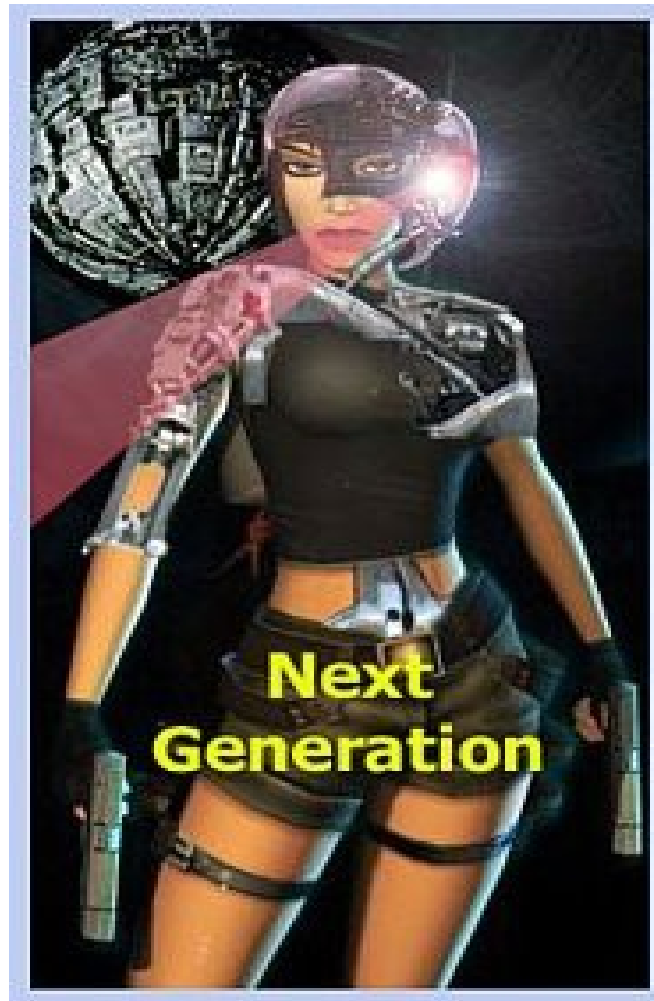


TOMB RAIDER NEXT GENERATION



MNEMONICS

(USE THESE IN THE NEW SCRIPT COMMANDS)

BASED ON : DLL 1.3.0.7

CONTENTS

MNEMONIC	USED IN COMMAND	PAGE
ADD_constants	AddEffect=	9
AMMO_constants	Customize=CUST_AMMO	11
ASF_constants	AnimationSlot=	12
BAR_constants	Customize=CUST_BAR	13
BINF_constants	Customize=CUST_BINOCULARS	15
BINT_constants	Customize=CUST_BINOCULARS	18
BKGDF_constants	Customize=CUST_BACKGROUND	20
BUGF_constants	Customize=CUST_FIX_BUGS	26
CDM_constants	Customize=CUST_CD_SINGLE_PLAYBACK	27
CL_constants	TextFormat=	28
CODE_constants	Plugin=	29
COLL_constants	Customize=CUST_SET_STILL_COLLISION	30
COLTYPE_constants	Parameters=PARAM_COLOR_ITEM	33
CUST_constants	Customize=	34
DEMF_constants	Demo=	100
DEMO_	Demo=	103
DENV_constants	Works with the ENV_ITEM_EXTRA... conditions	104
DGX_constants	DiagnosticType=	105
DIR_constants	Parameters=PARAM_MOVE_ITEM	112
DISABLED_constants		117
DMG_constants	Damage=	118
DRT_constants	Customize=CUST_DARTS	121
DTF_constants	Detector=	125
DWF_constants	DefaultWindowsFont=	127
EDGX_constants	DiagnosticType=	128
EF_constants	Elevator=	136
ENABLED_constants		141
ENV_constants	Animation=	142
	AnimationSlot=	
	MultEnvCondition=	
EXTRA_constants	Enemy=	167
FADD_constants	AddEffect=	168
FALSE_		172
FAN_constants	Animation=	173
	AnimationSlot=	
FBAR_constants	Customize=CUST_BAR	181
FCAM_constants	Customize=CUST_CAMERA	184
FFL_constants	Customize=CUST_FLARE	185
FGT_constants	GlobalTrigger=	186
FLI_constants	LogItem=	190

CONTENTS

MNEMONIC	USED IN COMMAND	PAGE
FMIR_constants	MirrorEffect=	192
FMOV_constants	Parameters=PARAM_MOVE_ITEM	193
FMV_constants	Customize=CUST_FMV_CUTSCENE	198
FO_constants	Organizer=	200
FR_constants	Customize=CUST_RAIN	203
FRB_constants	Customize=CUST_ROLLING_BOAT	204
FROT_constant	Parameters=PARAM_ROTATE_ITEM	206
FSB_constants	StandBy=	207
FSCA_constants	Parameters=PARAM_SCALE_ITEM	210
FSCAM_constants	Parameters=PARAM_SET_CAMERA	211
FSS_constants	Parameters=PARAM_SHOW_SPRITE	212
FT_constants	TextFormat=	215
FTYPE_constants	ImportFile=	218
GT_constants	GlobalTrigger=	219
GTD_constants	GlobalTrigger=	239
HAIR_constants	Customize=	240
HIT_constants	CUST_BIKE_VS_ENEMIES	242
HOLD_constants	Animation= used in ENV_extra field	244
HRP_constants	Customize=CUST_HARPOON	246
IF_constants	Image=	247
IGNORE_constants		253
IMPORT_constants	ImportFile=	254
JOINT_constants	AddEffect=	256
KEY_constants	Animation=	258
	AnimationSlot=	
KLH_constant	Customize=CUST_KEEP_LARA_HP	260
LDF_constants	Diary=	261
LGTN_constants	Parameters=PARAM_LIGHTNING	263
LOAD_constants	Equipment=	266
MIR_constants	MirrorEffect=	267
MIST_COL_constants	AddEffect=	268
MPS_constants	Plugin=	269
NEF_constants	Enemy=	270
OBJ_constants		274
OTYPE_constants	Parameters=PARAM_LIGHTNING	275
PARAM_constants	Parameters=	276
PB_constants	Customize=CUST_PARALLEL_BARS	304
PL_constants	Diary=	308
PLACE_constants	Animation=	313
QSF_constants	Customize=CUST_INNER_SCREENSHOT	314
RAIN_constants	Rain=	316

CONTENTS

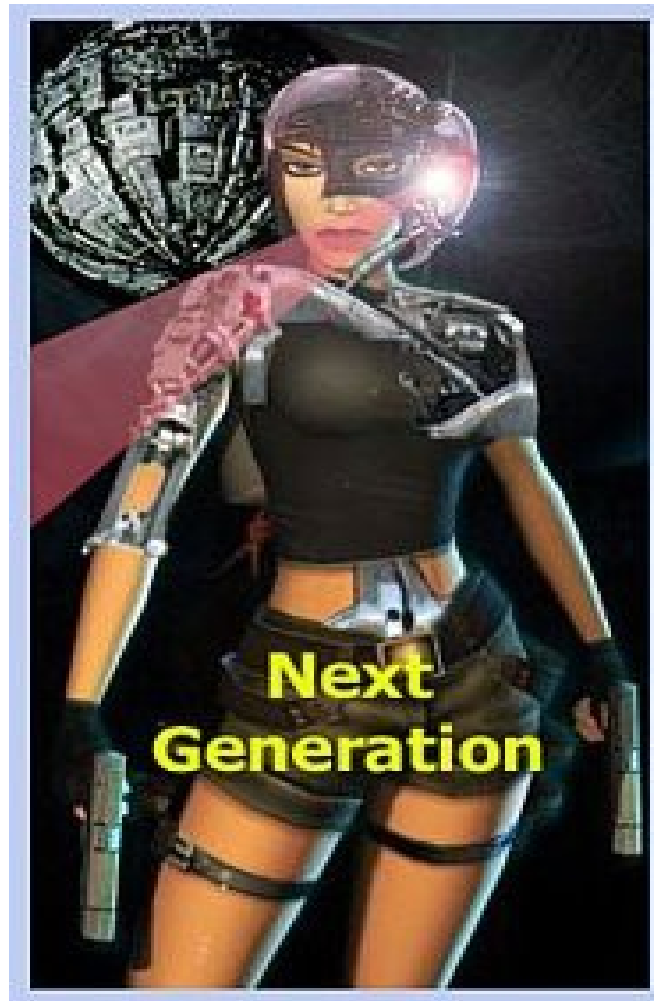
MNEMONIC	USED IN COMMAND	PAGE
RIB_constants	Parameters=PARAM_INPUT_BOX	317
ROOM_constants	with the ENV_ROOM	323
ROTH_constants	Parameters=PARAM_ROTATE_ITEM	325
ROTV_constants	Parameters=PARAM_ROTATE_ITEM	326
SC_constants	TextFormat=	327
SEQ_constants	TextureSequence=	328
SET_constants	Settings=	329
SEXT_constants	Obsolete.	333
SHOWC_constants	Customize=CUST_SHOW_AMMO_COUNTER	334
SNOW_constants	Snow=	336
SPC_constants	Parameters=PARAM_ACTOR_SPEECH	337
SPCF_constants	Parameters=PARAM_ACTOR_SPEECH	346
SPF_constants	SavegamePanel=	347
SPL_constants	SavegamePanel=	349
SQ_constants	SoundSettings=	351
STATE_constants	Animation=	352
	AnimationSlot=	
SWR_constants	Customize=CUST_STAR_WARS_ROBOT	366
SWT_constants	Switch=	368
TCF_constants		369
TCMD_constants	TriggerGroup=	370
TGROUP_constants	TriggerGroup=	380
TPOS_constants	TestPosition=	390
TRB_constants	Turbo=	395
TRUE_constant		398
TS_constants	Customize=CUST_SFX	399
TSB_constants	StandBy=	403
TSCR_constants	GT_TITLE_SCREEN Global	404
TT_constants	Customize=CUST_SET_TEXT_COLOR	405
WEAP_constants	Customize= CUST_WEAPON	409
WFF_constants	WindowsFont=	410
WTF_constants	Parameters=PARAM_WTEXT	416

MNEMONIC	PAGE
CUST_ADD_DEATH_ANIMATION	34
CUST_AMMO	34
CUST_BACKGROUND	39
CUST_BAR	40
CUST_BIKE_VS_ENEMIES	42
CUST_BINOCULARS	44
CUST_CAMERA	48
CUST_CD_SINGLE_PLAYBACK	52
CUST_DARTS	52
CUST_DISABLE_FORCING_ANIM_96	54
CUST_DISABLE_MISSING_SOUNDS	54
CUST_DISABLE_PUSH_AWAY_ANIMATION	54
CUST_DISABLE_SCREAMING_HEAD	55
CUST_ESCAPE_FLY_CAMERA	55
CUST_FIX_BUGS	56
CUST_FIX_WATER_FOG_BUG	56
CUST_FLARE	57
CUST_FMV_CUTSCENE	57
CUST_HAIR_TYPE	58
CUST_HARPOON	59
CUST_INNER_SCREENSHOT	61
CUST_KEEP_DEAD_ENEMIES	61
CUST_KEEP_LARA_HP	62
CUST_LIGHT_OBJECT	63
CUST_LOOK_TRANSPARENT	63
CUST_NEW_SOUND_ENGINE	64
CUST_NO_TIME_IN_SAVELIST	67
CUST_PARALLEL_BARS	67
CUST_PAUSE_FLY_CAMERA	68
CUST_RAIN	69
CUST_ROLLING_BOAT	71
CUST_ROLLINGBALL_PUSHING	73
CUST_SAVE_LOCUST	75
CUST_SCREENSHOT_CAPTURE	76

CUST_constants Customize=

MNEMONIC	PAGE
CUST_SET_CREDITS_LEVEL	77
CUST_SET_INV_ITEM	78
CUST_SET_JEEP_KEY_SLOT	79
CUST_SET_OLD_CD_TRIGGER	80
CUST_SET_SECRET_NUMBER	80
CUST_SET_STATIC_DAMAGE	81
CUST_SET_STILL_COLLISION	82
CUST_SET_TEXT_COLOR	84
CUST_SFX	85
CUST_SFX_DEMO	85
CUST_SHATTER_RANGE	86
CUST_SHATTER_SPECIFIC	87
CUST_SHOW_AMMO_COUNTER	88
CUST_SLOT_FLAGS	89
CUST_SPEED_MOVING	90
CUST_STAR_WARS_ROBOT	91
CUST_STATIC_TRANSPARENCY	92
CUST_TEXT_ON_FLY_SCREEN	92
CUST_TITLE_FMV	93
CUST_TR5_UNDERWATER_COLLISIONS	94
CUST_WATERFALL_SPEED	94
CUST_WEAPON	95

TOMB RAIDER NEXT GENERATION



MNEMONICS INFORMATION

(USE THESE IN THE NEW SCRIPT COMMANDS)

ADD_constants

3 \$0003: ADD_BLOOD

Used in the AddEffect= command.

Use this value to add blood (vlady) to the current object.

1 \$0001: ADD_FLAME

Used in the AddEffect= command.

Use this value to add fire to the current object

6 \$0006: ADD_LIGHT_BLINK

Used in the AddEffect= command.

Use this value to add a blinking light to some moveable.

The blinking light is the same as a flare light, where the intensity of the light changes a bit in a random way.

5 \$0005: ADD_LIGHT_FLAT

Used in the AddEffect= command.

Use this value to add a colour light to some moveable.

For "flat light" we mean a common non-blinking, non-spot light.

It is like a light-bulb.

8 \$0008: ADD_LIGHT_GLOVE

Used in the AddEffect= command.

This light has the shape of a sphere. This glove light is opaque, i.e. not transparent, like the yellow halo around the lights in a fogged night.

7 \$0007: ADD_LIGHT_SPOT

Used in the AddEffect= command.

This light works like a light-spot, i.e. it generates a cone of light following a given direction like the headlight of the SIDECAR.

For this light type it is important to choose the facing direction of the light cone.

The default facing is that of the moveable where you add this effect.

Change this facing using the **FADD_** flags in the **FlagsEffect (FADD_)** field.

Remarks: You could have trouble if the **JEEP** or boats use the header light, because this add effect uses the same light resources to generate this light-spot. With this kind of light type a very high Intensity value, about 200. You can not change the standard yellow-white colour for the spot light.

ADD_constants

4 \$0004: ADD_MIST

Used in the AddEffect= command.

Add Mist to the current object.

Mist are sprays of water.

You see the Mist in the waterfall.

Use the **ADD_MIST** and there are four additional fields in the **AddEffect** command and the new syntax of the **AddEffect** command becomes:

Syntax: AddEffect=Id, EffectType (**ADD_**), FlagsEffect (**FADD_**), JointType (**JOINT_**), DispX, DispY, DispZ, DurateEmit, DuratePause, Extra1(SizeMistBall), Extra2(NumberOfMistBalls), Extra3(ColorMist), Extra4(PersistenceOfMist)

These are the descriptions of the Extra parameters:

Extra1 = Size of Mist Ball. Default value is 12

Extra2 = Number of Mist Balls. Default is 1. You can set a maximum of 4 Mist Balls, they will be placed over an ideal line.
The Mist line will be oriented following the facing of the current moveable, rotate the facing of the Mist line using the rotate flags.

Extra3 = Color of Mist. Set a MIST_COL_ value.

See the **MIST_COL_** constant

Extra4 = Persistence time of Mist. Default value = 6.
Only use this field if the moveable is moving faster and you want a wake.
Bigger values, longer wakes.

2 \$0002: ADD_SMOKE

Used in the AddEffect= command.

Add smoke to the current object.

AMMO_constants

16 \$0010: AMMO_ADD_GUN_SHELL

Used in the Customize=CUST_AMMO command.

Used to add the shell falling down after shooting.

If this flag is used it will be use the mesh slot **GUNSHELL** for the shell.

32 \$0020: AMMO_ADD_SHOTGUN_SHELL

Used in the Customize=CUST_AMMO command.

It used to add the shell falling down after shooting.

If this flag is used it will use the mesh slot **SHOTGUNSHELL** for the shell.

4 \$0004: AMMO_PUSH_LARA

Used in the Customize=CUST_AMMO command.

When Lara shoots the recoil will move her backwards.

Use this flag with heavy ammo like a grenade.

Type in the **Extra** field the distance of movement.

The distance units are 1024 = one sector, 256 = one click.

2 \$0002: AMMO_PUSH_TARGET

Used in the Customize=CUST_AMMO command.

When the ammo hits the enemy he will be pushed (moved) as if the impact of the ammo had moved him.

Type in the **Extra** field the distance of movement.

The distance units are 1024 = one sector, 256 = one click.

64 \$0040: AMMO_REMOVE_SHOTGUN_SHELL

Used in the Customize=CUST_AMMO command.

This flag removes the shell case effect.

Only use this flag for shotgun ammo.

8 \$0008: AMMO_SET_GRENADE_TIMER

Used in the Customize=CUST_AMMO command.

Only used with grenade ammo.

The number of seconds required to explode the grenade can be changed.

The **default value is 4 seconds**.

Type the number of seconds in the **Extra** field.

ASF_constants

Used in the AnimationSlot= command.

NO FLAGS in the **tomb ide** list.

BAR_constants

2 \$0002: BAR_AIR

Used in the Customize=CUST_BAR command.

Display the Air Bar.

9 \$0009: BAR_COLD

Used in the Customize=CUST_BAR command.

Customize the bar shown in cold water rooms.

Please, note that damage bar colour could also be changed in the Damage= script command.

TRNG will use the colour of the Damage command in the case if you type **IGNORE** in the **Color1** field bar in the **CUST_BAR** command.

4 \$0004: BAR_CUSTOM1

Used in the Customize=CUST_BAR command.

The Custom bar is different from the other health, dash, air bars.

In this case there is no predefined target for it.

This is a free bar to use.

5 \$0005: BAR_CUSTOM2

Used in the Customize=CUST_BAR command.

Set the second customizable bar.

See the description of the **BAR_CUSTOM1** constant for more information about customize bars.

6 \$0006: BAR_CUSTOM3

Used in the Customize=CUST_BAR command.

Set the third customizable bar.

See the description of the **BAR_CUSTOM1** constant for more information about customize bars.

7 \$0007: BAR_CUSTOM4

Used in the Customize=CUST_BAR command.

Set the fourth customizable bar.

See the description of the **BAR_CUSTOM1** constant for more information about customize bars.

BAR_constants

8 \$0008: BAR_DAMAGE

Used in the Customize=CUST_BAR command.

Customize the bar shown in damage rooms.

Pease, note that the damage bar colour could also be changed in the Damage= script command.

TRNG will use the colour of the Damage command if type **IGNORE** in the **Color1** field bar in the **CUST_BAR** command.

1 \$0001: BAR_DASH

Used in the Customize=CUST_BAR command.

Displays the Run sprint Bar.

0 \$0000: BAR_HEALTH

Used in the Customize=CUST_BAR command.

Displays the Health Bar.

3 \$0003: BAR_LOAD_LEVEL

Used in the Customize=CUST_BAR command.

Displays the Load Level Bar.

BINF_constants

1 \$0001:BINF_COMPASS

Used in the Customize=CUST_BINOCULARS command.

To show information about the compass in the binocular view.

Add the **BINF_COMPASS** flag.

Then set in the **CompassRectAndFlags** field a **BINT_** value to choose the kind of compass view.

Choose the the **BINT_STRIP** value to use an image with graduate strip about the cardinal scale, or a **BINT_NUMERIC** value to show the degrees of the cardinal points in numeric format.

128 \$0080: BINF_LIGHT_SWITCH

Used in the Customize=CUST_BINOCULARS command.

Use this flag to enable a graphic switch to show when the player is using light in the binocular view. The switch will be a rectangle or circle zone where the colour will change according to switching the light on or off.

4 \$0004: BINF_LIGHTNESS

Used in the Customize=CUST_BINOCULARS command.

Use this flag and add information about the lightness intensity.

The lightness intensity will be computed on a little rectangle centred in the middle of the screen.

If the lightness information is shown in numeric format, it will use the ISO/DIN scale (0 / 39).

Or, using a bar, the full bar will be the maximum light while dark will be an empty bar.

512 \$0200: BINF_NOTATION_EXTENDED

Used in the Customize=CUST_BINOCULARS command.

When enabled the compass information will be shown as a number of degrees.

The short notation is: 123° where the degrees begin from 0 (north) increasing clockwise passing through 90° (EAST), 180° (SOUTH) and 270° (WEST).

You can use the extended notation where the degrees will only cover 0-89° degrees, beginning from the first cardinal point.

Example: "N 45°" means NE (or 45°)
 "E 20°" means "110°" (about) South-east direction

8192 \$\$2000: BINF_PROGRESSIVE_ZOOM

Used in the Customize=CUST_BINOCULARS command.

This flag changes the speed and progression of the zoom.

The old method was very fast and it always had the same speed.

The new (progressive) method tries to begin slowly, increasing the speed only when the player keeps down the scale key.

The progressive zoom works better having a fine tuning scale factor.

BINF_constants

2 \$0002: BINF_SEXTANT

Used in the Customize=CUST_BINOCULARS command.

To add a sextant to the binocular view add this flag.

Sextant gives information about the vertical degrees of the current view.

When Lara looks up the degrees will be increased,
when she looks down the degrees will decrease.

If this flag is used also set a **BINT_** value in the **SextantRectAndFlags** field to choose the kind of data to be shown by the sextant.

256 \$0100: BINF_SOUND_ON_LIGHT

Used in the Customize=CUST_BAR command

Customize the bar shown in the damage rooms.

Please note that the damage bar colour could also be changed in the Damage= script command.

TRNG will use the colour of the Damage command if you type **IGNORE** in the **Color1** field bar in the **CUST_BAR** command.

16384 \$4000: BINF_SOUND_ON_ZOOM

Used in the Customize=CUST_BAR command.

Customize the bar shown in the damage rooms.

Please note that the damage bar colour could also be changed in the Damage= script command.

TRNG will use the colour of the Damage command if you type **IGNORE** in the **Color1** field bar in the **CUST_BAR** command.

4096 \$1000: BINF_SUPER_ZOOM

Used in the Customize=CUST_BINOCULARS command.

This flag improves the zoom-in factor of the binoculars.

The maximum magnification will be from 4.2x to about 8.4 (double).

Note: This flag can be omitted from the script and enabled by using a flip effect.

This method may be useful in the game to simulate a change to the binoculars to improve them to some target reached by Lara in the game.

This means that it is possible to improve the binoculars in the game,
but it is not possible to do the opposite
i.e. removing the super_zoom once it has been supplied.

BINF_constants

1024 \$0400: BINF_SWING_COMPASS

Used in the Customize=CUST_BINOCULARS command.

When the compass strip is used add the **BINF_SWING_COMPASS** flag to simulate a random swinging of the compass every time the player moves the binoculars.

2048 \$0800: BINF_SWING_SEXTANT

Used in the Customize=CUST_BINOCULARS command.

This flag enables a simulation of swinging for the sextant value (in strip mode) when the player moves the binoculars

8 \$0008: BINF_ZOOM

Used in the Customize=CUST_BINOCULARS command.

This enables showing information about the current zoom-in factor.
See the description of the **ZoomRectAndFlags** field.

BINT_constants

16384 \$4000: BINT_BAR

Used in the Customize=CUST_BINOCULARS command.

For a solid bar to measure the intensity of some value in the binoculars screen, use the **BINT_BAR** value with an Id for the rectangle.

Note: Remember that when you use a bar it is necessary to set in the **PARAM_RECT** command (linked with the current binoculars information to show...) the values for the Foreground colour (the main colour of the bar) and the Back Ground colour (the empty colour of the bar).

8192 \$2000: BINT_NUMERIC

Used in the Customize=CUST_BINOCULARS command.

To show information about the compass in numeric format use the **BINT_NUMERIC** type in the **CompassRectAndFlags** field.

The default format is a simple number of degrees from 0 to 359.

It is possible to use the extended format adding to the flags (**BINF_**) field the **BINF_NOTATION_EXTENDED** flag

See the description of the **BINF_NOTATION_EXTENDED** flag for more information.

4096 \$1000: BINT_PATTERN_BAR

Used in the Customize=CUST_BINOCULARS command.

A pattern bar where the level of some value will be shown as a length of bar with a pattern instead of a solid colour.

To draw the pattern bar draw it in the binoculars screen at full size

Then in the rectangle (in the script) set the size of the bar type in the **BackGround** colour field of the **PARAM_RECT** command.

The colour below the pattern bar, i.e. the colour used to erase the pattern to simulate a shorter bar.

For example: If the pattern bar is above a black zone set a black colour in the background field.

BINT_constants

2048 \$0800: BINT_ROUND

Used in the Customize=CUST_BINOCULARS command.

Currently this only works in the **LightnessRectAndFlags** field to force the light switch to have a circle shape.

32768 \$8000: BINT_STRIP

Used in the Customize=CUST_BINOCULARS command.

You can show data as a strip for the compass and sextant.

A strip is a slim image with a graduate scale.

The strip for the compass should always have the size of 1400 x 64

Use the compass strip in the lightning demo level as a reference for position of the cardinal points and degrees.

BKGDF_constants

64 \$0040: BKGDF_ADD_COLOR_AND

Used in the Customize=CUST_BACKGROUND command.

This flag perform an **AND** operation between the game scene colours and the colour set in the **ColorRGB ID** in the Parameter field.

The **AND** operation returns a darker image with a preponderance of the colour linked to it. With the **BKGDF_ADD_COLOR_AND** flag it is better to set colour numbers with many bits set to 1, like 255, 127, 63, 31 or 15

For example to have the game scene changed to a dominant red colour use the colour:
ColorRGB= ID, 0, 127, 0

The above preserves the green tone while it is clear of all other tones.
If you do not want to loose all other colours set a value in the other tones with lower values:
ColorRGB= ID, 15, 127, 15

Note: This flag only works with **BKGD_T_INVENTORY type**
+ BKGDF_KEEP_GAME_SCREEN flag since it performs a change on the original game scene.

32 \$0020: BKGDF_ADD_COLOR_OR

Used in the Customize=CUST_BACKGROUND command.

This flag inverts all of the colours of the game scene creating an effect like a negative film. This effect works alone ignoring further colour set in the Parameter field.

Note: This flag only works with **BKGD_T_INVENTORY type**
+ BKGDF_KEEP_GAME_SCREEN flag since it performs a change on the original game scene.

BKGDF_constants

128 \$0080: BKGDF_ADD_COLOR_XOR

Used in the Customize=CUST_BACKGROUND command.

This flag perform a **XOR** (exclusive OR) operation between the game scene colours and the colour set in the **ColorRGB ID** in the Parameter field.

This flag creates a result similar to a abstract artwork, where all of the colours will be exchanged in a chaotic way.

Since each original colour will always be replaced with the same (new) colour the scene will remain understandable but with weird colours.

With this flag it is better to use numbers with many bits set, like 255, 127, 63, 31, 15

If you want only invert a colour tone to 0 other tones will not be changed.

For example to only change tones of the original game scene use as a colour:

ColorRGB= ID, 255, 0, 0

Note: This flag only works with **BKGD_T_INVENTORY** type
+ **BKGDF_KEEP_GAME_SCREEN** flag since it performs a change on
the original game scene.

BKGDF_constants

512 \$0200: BKGDF_ADD_DARKNESS

Used in the Customize=CUST_BACKGROUND command.

This flag add a dark shadow to the original game scene, giving the idea of a transparent effect like a plate glass over the game scene.

In the Parameter field set the level of darkness with a number between 3 and 255.

Where 3 is almost no darkness and 255 is full black.

Reasonable settings are in the range between 50 to 200

Note: This flag only works with **BKGD_T_INVENTORY** type
+ **BKGDF_KEEP_GAME_SCREEN** flag since it performs a change
on the original game scene.

256 \$0100: BKGDF_ADD_NEGATIVE_EFFECT

Used in the Customize=CUST_BACKGROUND command.

This flag inverts all colours of the game scene creating an effect like a negative film.

This effect works alone ignoring further colour set in the Parameter field.

Note: This flag only works with **BKGD_T_INVENTORY** type
+ **BKGDF_KEEP_GAME_SCREEN** flag since it performs a change
on the original game scene.

8 \$0008: BKGDF_COLORIZE

Used in the Customize=CUST_BACKGROUND command.

This flag works with the **BKGD_T_BINOCULAR** and **BKGD_T_LASER_SIGHT** types.

Add a colour to the screen when Lara is looking through the binoculars or laser sight.

Type in the Parameter field the Id of a ColorRGB= command for the colour to use.

1 \$0001: BKGDF_HIDE_LOADING_BAR

Used in the Customize=CUST_BACKGROUND command.

If customizing the **BKGD_T_LOADING_LEVEL** screen use this flag to hide the loading bar.

4 \$0004: BKGDF_HIDE_SPRITE_LASER_SIGHT

Used in the Customize=CUST_BACKGROUND command.

This flag works in accordance with the **BKGD_T_LASER_SIGHT** type.

To remove the little red point (a sprite), add this flag.

BKGDF_constants

16 \$0010: BKGDF_KEEP_GAME_SCREEN

Used in the Customize=CUST_BACKGROUND command.

If you do not want a custom image for the background but the current game screen, i.e. the last image in the game add this flag.

When the **BKGDF_KEEP_GAME_SCREEN** flag is used the image number in the image field can be omitted.

Note: This setting only works where there is a good game image, as happens for the inventory. It is not advisable for the Title and or the binoculars or laser sight types.

2 \$0002:BKGDF_MINIMAL_LOADING_TIME

Used in the Customize=CUST_BACKGROUND command.

This flag works according to the **BKGD_LOADING_LEVEL** type.

To show an image for a given time, add the **BKGDF_MINIMAL_LOADING_TIME** flag and set in the Parameter field the number of seconds that the image will be shown.

Note: In the case of loading the next level it will require a longer time to load. This loading time can not be reduced with this setting. Only use this flag to be sure that the player is able to see the image used for the loading screen. That is when there is text to read before the next level begins.

1024 \$0400: BKGD_SEMI_TRANSPARENT

Used in the Customize=CUST_BACKGROUND command.

This flag only works paired to the **BKGDF_KEEP_GAME_SCREEN** flag.

With the **BKGDF_SEMI_TRANSPARENT** flag you can force a custom image.

The **ImageId** is set in the **ImageId** field of the Customize=CUST_BACKGROUND command. over the current game scene. Choose the transparency level of your custom image.

Type in the Parameter field the opacity level of the custom image.

Where 0 = fully transparent and 255 = fully opaque.

Reasonable values will be in the range: 100 to 150.

BKGDF_constants

2048 \$0800: BKGDT_SKIP_LOADING_TIME

Used in the Customize=CUST_BACKGROUND command.

This flag only works with the **BKGDF_MINIMAL_LOADING_TIME** flag.

If you use the **BKGDF_MINIMAL_LOADING_TIME** flag, the background for the loading level will be kept on the screen until the time set.

If you want the player to be able to skip the background and go to the next level add the **BKGDF_SKIP_LOADING_TIME** flag. The player will be able to skip the background with the escape, space, or control key.

Note: In the background image when this flag is used add text to inform the player about the pasting of the image. Use text like "Escape to skip".

4 \$0004: BKGDT_BINOCULAR

Used in the Customize=CUST_BACKGROUND command.

Replace the old binocular mask with the image supplied in the Image command linked with the **CUST_BACKGROUND** script command.

Since the image has to be in full screen (other settings will be ignored) transparent settings for the image must be used.

1 \$0001: BKGDT_INVENTORY

Used in the Customize=CUST_BACKGROUND command.

Used to set the image for the inventory screen and all intermediate screens in the game: pause menu, options screen, statistic screen, load game and save game screens.

5 \$0005: BKGDT_LASER_SIGHT

Used in the Customize=CUST_BACKGROUND command.

Replace the mask of the laser-sight with the Image command linked with the **CUST_BACKGROUND** script command.

Since the image has to be in full screen (other settings will be ignored).

Transparent settings for the image must be used.

2 \$0002: BKGDT_LOADING_LEVEL

Used in the Customize=CUST_BACKGROUND command.

Used to set an image for the loading level phase.

Please note this is not the load game screen seen when the save games load.

It is when there is the progress bar and the next level loading.

BKGDF_constants

3 \$0003: BKGDT_TITLE

Used in the Customize=CUST_BACKGROUND command.

Used to change the title flyby camera with a fixed image.

This setting will also affect all other sub-screens that the user can reach from the title: new game, load game and options.

Note: The setting does not stop the flyby camera but the image will cover it.
To have an image with some transparent zones from those zones
to show below the flyby camera.

BUGF_constants

2 \$0002: BUGF_DART_NO_POISON_LARA

Used with the Customize=CUST_FIX_BUGS command.

In the default **Tomb4** there was a bug about poisoning with darts.

While the poison of the scorpion and harpies worked correctly with the screen deformation and Lara losing HP, the poison of the dart only remained for an instant and was then immediately removed.

4 \$0004: BUGF_LAND_WATER_SFX_ENEMIES

Used with the Customize=CUST_FIX_BUGS command.

In the default **Tomb4** there was a bug about the management of the sfx sound in the animation commands for some enemies.

There was no correct handling for the water/land with sound effect played as animation commands.

This bug was not present with all enemies.

For instance the land/water sounds worked fine with the skeleton or baddy1 or baddy2, but it did not work with the guide or the big scorpion.

1 \$0001: BUGF_TRANSPARENT_WHITE_ON_FOG

Used with the Customize=CUST_FIX_BUGS command.

Used to fix the bug in the water textures when there is the distance fog enabled in the level. The transparent texture will become full white.

When his fix is enabled the water textures will ignore the fog density.

Use it when the level has a depth fog.

This fix could create some trouble with fog bulbs close to water.

CDM_constants

1 \$0001: CDM_NO_SAVE

Used by the Customize=CUST_CD_SINGLE_PLAYBACK.

Using this setting the further single-playback CD will be ignored in the saving game and when the player reloads the save game the single CD will not be played.

3 \$0003: CDM_RESTORE_FROM_BEGIN

Used by the Customize=CUST_CD_SINGLE_PLAYBACK.

This setting forces the **TRNG** engine to save the single-playback CD and restore it from the start and not from the effective position it had in the game when it was saved.

This setting is advisable for a small audio track like a sound where Lara was speaking.

2 \$0002: CDM_RESTORE_FROM_MIDDLE

Used by the Customize=CUST_CD_SINGLE_PLAYBACK.

This is the default behaviour.

If the **CUST_CD_SINGLE_PLAYBACK** is not set to customize in the **script.dat** the **TRNG** engine will work using the **CDM_RESTORE_FROM_MIDDLE**.

With this setting the sound will be saved remembering its current position and it will restore playing from that precise position.

CL_constants

1 \$0001: CL_BLINKING_WHITE

Use with the **Customize=CUST_SET_TEXT_COLOR** command.

Sets a blinking white colour.

4 \$0004: CL_BLUE

Use with the **Customize=CUST_SET_TEXT_COLOR** command.

Sets a blue colour.

7 \$0007: CL_DARKMETAL

Use with the **Customize=CUST_SET_TEXT_COLOR** command.

Sets a dark metal colour.

6 \$0006: CL_GOLD

Use with the **Customize=CUST_SET_TEXT_COLOR** command.

Sets a gold colour.

5 \$0005: CL_METAL

Use with the **Customize=CUST_SET_TEXT_COLOR** command.

Sets a metal colour.

3 \$0003: CL_RED

Use with the **Customize=CUST_SET_TEXT_COLOR** command.

Sets a red colour.

2 \$0002: CL_WHITE

Use with the **Customize=CUST_SET_TEXT_COLOR** command.

Sets a white colour.

8 \$0008: CL_YELLOW

Use with the **Customize=CUST_SET_TEXT_COLOR** command.

Sets a yellow colour.

CODE_constants

2048 \$0800: CODE_ACTION

Used in the Plugin= command.

The **DisableFeatureArray** of the Plugin command can disable the changes for the **plugin** from a standard **TRNG** action trigger.

For instance if you do not want the changes with action 41 type in the **DisableFeatureArray** the value: **CODE_ACTION + 41**

The **TRNG** will stop the **plugin** changing the code for action 41

4096 \$1000: CODE_CONDITION

Used in the Plugin= command.

The **DisableFeatureArray** of the Plugin command disables the changes for the **plugin** from a standard **TRNG** condition trigger.

For instance if you do not want the changes with condition 23 type in the **DisableFeatureArray** the value: **CODE_CONDITION + 23**

The **TRNG** will stop the **plugin** changing the code for condition 23

1014 \$0400: CODE_FLIPEFFECT

Used in the Plugin= command.

The **DisableFeatureArray** of the Plugin command disables the changes for the **plugin** from a standard **TRNG** flipeffect.

For instance if you do not want the changes with flipeffect 132 type in the **DisableFeatureArray** the value: **CODE_FLIPEFFECT + 132**

The **TRNG** will stop the **plugin** changing the code for flipeffect 132

COLL_constants

4 \$0004: COLL_ANIMATING

Used with the Customize=CUST_SET_STILL_COLLISION command.

It enables the still collision when Lara touches an ANIMATING slot item.

2 \$0002: COLL_DOORS

Used with the Customize=CUST_SET_STILL_COLLISION command.

It enables the still collision when Lara touches a door.

Remark: This does not work with underwater doors or trap-doors.

32 \$0020: COLL_FAKE_WALLS

Used with the Customize=CUST_SET_STILL_COLLISION command.

This flag enables the still collision on some moveables used in the game to simulate walls, ceiling or floor:

SMASHABLE_BIKE_WALL,	SMASHABLE_BIKE_FLOOR,
FALLING_CEILING,	FALLING_BLOCK,
FALLING_BLOCK2,	BURNING_FLOOR,
ONEBLOCK_PLATFORM,	TWOBLOCK_PLATFORM,
RAISING_BLOCK1,	RAISING_BLOCK2,
EXPANDING_PLATFORM,	BRIDGE_FLAT,
BRIDGE_TILT1,	BRIDGE_TILT2
PUSHABLE OBJECTS.	

Remark: Some moveable used as a "floor" or "ceiling", the still collision will be applied only in the horizontal movement.

For example: When Lara splats her face on the edge of a bridge.

128 \$0080: COLL_FAST_TURNING

Used with the Customize=CUST_SET_STILL_COLLISION command.

When collision is enabled and Lara touches an item with an acute angle, she will not be stopped but only turned to continue her race in the correct direction to avoid the obstacle.

Using the **COLL_FAST_TURNING** flag the rotation speed can be increased.

Remark: The fast turning could be suggested when static items are placed with an intermediate facing of 45 degrees with respect to room walls.

COLL_constants

1024 \$0400: COLL_NO_SLIDING

Used with the Customize=CUST_SET_STILL_COLLISION command.

This sets a very drastic collision mode: removing all sliding features so Lara will be stopped with the current angle of impact with the item.

512 \$0200: COLL_NO_SPLAT

Used with the Customize=CUST_SET_STILL_COLLISION command.

The still collision tries to emulate the normal collision of Lara with the walls. When Lara touches an item with the correct height the "splat" animation will be performed. To disable the splat and only have an immediate frozen Lara add the **COLL_NO_SPLAT** flag.

8 \$0008: COLL_PANELS

Used with the Customize=CUST_SET_STILL_COLLISION command.

It enables the still collision when Lara touches some moveable PANEL item.

Remarks: The **PANEL_** items have been added with the **TRNG** engine. They are invisible moveables only used to set a specific collision on a side of the rooms. They have a slot in the range 472 to 479 and names like **PANEL_BORDER**, **PANEL_CORNER** etc.

The still collisions have a problem when it is necessary to decide about sliding/stopping with an item having a diagonal facing (with 45 degrees angle). This is because of the difficulty to correctly value the diagonal shape of the item.

Using the specific items: **PANEL_DIAGONAL** or **PANEL_MIDDLE_CORNER**, use a specific code to manage these diagonal items correctly. Therefore, if there is trouble with a diagonal item (when Lara continues to move her legs and arms) **try to solve the problem in this way:**

Disable the collision for the item
Place in the same position a **PANEL_DIAGONAL** item to cover the same collision position.

1 \$0001: COLL_STATICS

Used with the Customize=CUST_SET_STILL_COLLISION command.

It enables the still collision when Lara touches a static item.

COLL_constants

64 \$0040: COLL_STOP_ON_45_DEGREES

Used with the Customize=CUST_SET_STILL_COLLISION command.

The still collision method tries to simulate the same behaviour as a wall collision.
In the still collision in some circumstances Lara will be turned into the correct direction.

16 \$0010: COLL_VEHICLES

Used with the Customize=CUST_SET_STILL_COLLISION command.

It enables the still collision when Lara touches some earth vehicle like a **JEEP** or a **SIDECAR**.

Remark: This setting has no effect on water vehicles like boats.

COLTYPE_constants

1 \$0001: COLTYPE_SET_COLOR

Used in the Parameters=PARAM_COLOR_ITEM.

Use this to change the colour of the item using the first colour pointed to by the **Index1ColorRGB** field.

2 \$0002: COLTYPE_SET_PULSE

Used in the Parameters=PARAM_COLOR_ITEM.

The pulse only uses the first colour in the **Index1ColorRGB** field.

The **TRNG** engine will pulse the colour **Index1ColorRGB** up and down.

It is advisable to set a base colour that is not too light because the **TRNG** engine will increase the lighting and then reduce it.

For example: Using White (bad choice) as a colour will give no result because it has maximum lighting.
Set the speed of the pulse with a value in the **SpeedChange** field.

3 \$0003: COLTYPE_SHADE_COLORS

Used in the Parameters=PARAM_COLOR_ITEM.

The shade colours shade from the first colour (**Index1ColorRGB**) to the second colour (**Index2ColorRGB**) and vice versa.

Set the speed of shading by a value in the **SpeedChange** field.

CUST_constants

24 \$0018: CUST_ADD_DEATH_ANIMATION

Used with the Customize=

Adds a dying animation for a DEMIGOD or other immortal creature.

Syntax: Customize=CUST_ADD_DEATH_ANIMATION, SlotOfCreature, AnimNumber

10 \$000A: CUST_AMMO

Used with the Customize=

Used to customize ammunition of weapons.

Syntax: Customize=CUST_AMMO, SlotOfAmmo, Ammo flags (**AMMO_....**), Damage, ShotsForBox, ShotsWithWeapon, Extra, IdTriggerGroupWhenHitEnemy, DamageForExplosion, Speed, Gravity, IdAddEffectToAmmo, IdTriggerGroupAtEnd

Description of fields

SlotOfAmmo

Type in this field the slot of the ammo to customize.

Choose one of the following:

PISTOLS_AMMO_ITEM	
UZI_AMMO_ITEM	
SHOTGUN_AMMO1_ITEM	(normal)
SHOTGUN_AMMO2_ITEM	(wide shot)
CROSSBOW_AMMO1_ITEM	(normal)
CROSSBOW_AMMO2_ITEM	(explosive)
CROSSBOW_AMMO3_ITEM	(poisoned)
GRENADE_GUN_AMMO1_ITEM	(normal)
GRENADE_GUN_AMMO2_ITEM	(power)
GRENADE_GUN_AMMO3_ITEM	(lightning)
SIXSHOOTER_AMMO_ITEM	

Ammo flags (**AMMO_**)

Add in this field one or more **AMMO_** flags to set different features for the current ammo.

Type **IGNORE** for no **AMMO_** flag used.

See the **AMMO_** flags

Damage

The normal damage (to Lara) from this ammo can be changed.

If the ammo has an unforeseen damage

(like the **GRENADE_GUN_AMMO3_ITEM** (lightning)) the damage will be ignored.

Remark:

This is a normal damage.

If the ammo is also explosive set the damage for the explosion in the other field **DamageForExplosion**.

CUST_constants

Default values:	Ammo	Default	Max Value
	Pistols	1	255
	UZI	1	255
	Revolver	21	255
	ShotGun Normal	3 (*6)	255
	ShotGun Wide	3 (*5)	1000
	Grenade Gun Normal	20	255
	Grenade Gun Power	20	1000
	CrossBow Normal	5	255
	CrossBow Explosive	5	1000

Remarks: Some ammo have 0 damage because it is only for explosion, however a common damage can be set for them.

The wide shot shotgun ammo as not more powerful than normal shot.
 The damage is computed like a random rain of fragments where the wide property reduces the number of fragments reaching the target.
 Usually this number is only 5. For normal shotgun ammo it is 6.
 Therefore the normal shotgun ammo is more powerful than wide shot ammo.
 The only exception is when the enemy is very big.
 To make wide shot shotgun ammo more powerful set a value for damage in the field and it will be added to the compute for damage.

ShotsForBox

When Lara picks up an ammo box, a given number of single shots will be added in the Inventory for that ammo.

Use this field to set the number of shots for each ammo box of this kind.

Ammo	Default	MaxForBox
PISTOLS_AMMO_ITEM	Unlimited (-2)	1000
UZI_AMMO_ITEM	30	1000
SHOTGUN_AMMO1_ITEM	6	1000
SHOTGUN_AMMO2_ITEM	6	42
CROSSBOW_AMMO1_ITEM	10	1000
CROSSBOW_AMMO2_ITEM	10	255
CROSSBOW_AMMO3_ITEM	10	255
GRENADE_GUN_AMMO1_ITEM	10	255
GRENADE_GUN_AMMO2_ITEM	4	255
GRENADE_GUN_AMMO3_ITEM	4	255
SIXSHOOTER_AMMO_ITEM	6	1000

CUST_constants

Remarks: To keep the default value type **IGNORE** in this field

There are technical reasons (within the original tomb4 code) why some ammo have different maximum values that can be set. Do not exceed the limits otherwise there will be crashes or bad working of the **TRNG** engine.

The value -1 should mean "unlimited ammo" but this is not possible because the -1 value is the same as the **IGNORE** value used. To set a number of shot for the box with "unlimited ammo" use a negative number for example -2.

ShotsWithWeapon

Every time Lara picks up a weapon (not an ammo box) the **Tomb4** engine gives her some shots for that weapon.

This means shots are within the weapon when Lara picks it up.

The number of given shots for the weapon can be changed using this field.

Set 0 (no shots "present") for this value, or any other value.

Default values:	Given with weapon	Number Shots	MaxAllowedValue
	Pistols	Unlimited (-2)	1000
	Revolver	6	255
	UZI	30	255
	Shot Gun	6 (normal)	42
	Cross Bow	10 (normal)	255
	Grenade Gun	10 (normal)	255

Remarks: To leave the value unchanged type **IGNORE** in this field.

There are technical reasons (within the original tomb4 code) because some ammo have different maximum values that can be set. Do not exceed the limits otherwise there will be crashes or bad working of the **TRNG** engine.

The value -1 should mean "unlimited ammo" but this is not possible because the -1 value is the same as the **IGNORE** value used. To set a number of shot for the box with "unlimited ammo" use a negative number, for example -2.

Extra

This optional field can be used in some circumstances for special ammo.

See the **AMMO_** constant descriptions to discover the possible use of this field.

CUST_constants

IdTriggerGroupWhenHitEnemy

Perform a TriggerGroup script command when this ammo hits an enemy.
To use this feature type the Id of the Trigger Group in the script command.

Remarks: When the enemy is hit by ammo, the **TRNG** engine will perform the Trigger Group and set a "Found item" in the index of the enemy.
To perform a special ACTION stored in the Trigger Group to hit the enemy, remember to add to the exported trigger the flag **TGROUP_USE_FOUND_ITEM_INDEX**.
The action trigger placed in the Trigger Group will use the index of the enemy hit by the ammo instead of the moveable set originally in the exported action trigger. Use this method to give special functions to the ammo.

If No Trigger Group is to perform type **IGNORE** in this field.

DamageForExplosion

This field is used to set the damage for the explosive ammo, normal and powerful grenade ammo or explosive Crossbow ammo.

The **default value is 30** for all explosive ammo.
The Maximum value for damage is 1000.

Remark: When the Crossbow poisoned dart is customized use this field to set the intensity of the poison. The **default value is 1**

Speed

For visible ammo like grenades and crossbow darts the (horizontal) speed can be changed.

Default values:	Default	MaxValue
Grenade (all types):	128	1024
Darts (all types):	512	1024

Remark: Different speeds for each grenade or dart type can be set.
This means, an explosive dart moves faster than a poisoned dart.

CUST_constants

Gravity

For visible ammo like grenades and Crossbow darts the gravity effect can be changed.

Default values:	Default	MaxValue
Grenade (all types)	3	255
Darts (all types)	0	255

Remarks: The gravity is not simply a vertical (down) speed but an acceleration value used to increase the vertical speed.

For darts it was not foreseen to use gravity, but it can be applied if required

IdAddEffectToAmmo

This setting only works for visible ammo grenade and dart.

To set a special effect using a **AddEffect** script command type in this field the Id to attach the effect to the ammo every time it is shot.

For example: To create an **AddEffect** to add a blue Mist wake.
Type the Id in this field and the ammo will have a blue Mist wake.

IdTriggerGroupAtEnd

This setting only works for visible ammo grenade and dart.

This field works in a similar way to the field **IdTriggerGroupWhenHitEnemy**.
But in this case it is not important if the ammo hits or misses an enemy.

The Trigger Group with the Id will be performed when the ammo hits anything: enemies, wall, floor, statics etc.

In this Trigger Group the "found enemy index" will be the index of the current ammo.
If an action is placed in the Trigger Group it can be forced to perform this action on the ammo item (in final position) using the constant **TGROUP_USE_FOUND_ITEM_INDEX**.

CUST_constants

49 \$0031: CUST_BACKGROUND

Used with the Customize=

Used with the Customize Background command

Syntax: Customize= CUST_BACKGROUND, BackGroundType (**BKGD_**...), Flags (**BKGDF_**...), Parameter, ImageId

You can customize the background used for different game phases by using an image from the pix folder.

BackGroundType (**BKGD_**)

Type in this field the type of background to customize.

See the **BKGD_** flags

Flags (**BKGDF_**)

Add one or more flags to modify some feature of the chosen background

If you do not wish to use the flags type **IGNORE** in this field.

See the **BKGDF_** flags

Parameter

This is a field that can have different meanings according to the flags set in the Flags field.

See the **BKGDF_** flags

ImageId

Type the Id of the Image= command to use as a background.

Type an Image= command first above the Customize=CUST_BACKGROUND.

It is suggested to use the **IF_PRELOAD** flag in the Image command, to have a faster display in the game.

Note: Many fields and flags of the Image command will be ignored, like those about effects.

In some background types set the Image command to use an audio track and the background will use that value as background music.

Example:

Image= 3, 4, IF_PRELOAD+IF_PLAY_AUDIO_TRACK+IF_LOOP_AUDIO_TRACK, IGNORE, 105, IGNORE, IGNORE, IGNORE, IGNORE

Customize= CUST_BACKGROUND, BKGD_INVENTORY, IGNORE, IGNORE, 3

The above commands will set the "image4.bmp" (stored in the **\pix** folder) as a new background for the inventory/pause/load game/save game and it will play the audio track 105 in loop mode.

CUST_constants

25 \$0019: CUST_BAR

Used with the Customize=

Customize all the default **TRNG** engine Bars like the Air Bar, Health Points Bar, Loading Bar etc. setting the position, size, colours and animation mode.

Syntax: Customize=CUST_BAR, BarType (**BAR_**), FlagsBar (**FBAR_**), XOrigin, YOrigin, XSize, YSize, IdColor1, IdColor2, Extra

BarType (**BAR_**)

In this field type a **BAR_** constant to specify what bar is to be customized.
See the **BAR_** constants

FlagsBar (**FBAR_**)

Add two or more **FBAR_** flags to specify some animation mode (the variation of colour in dynamic bars).
For no flag type **IGNORE** in this field

XOrigin, YOrigin, XSize, YSize

These four fields change the position and size of the current bar on the screen.
To use the original position and size of the bar type four **IGNORE** values in these fields.

All values typed in these fields are in pixels and are computed to work with a game screen of 640x480.

This method is necessary since it is not known what the effective game screen size is when the game is played. The player could change the settings in the set up.

So compute the position and size using a reference when the game is working at 640x480 pixels, then, when the game changes its size, the **TRNG** engine will adapt in a proportional way the coordinates set.

The default values are:

TypeBar	OrgX	OrgY	SizeX	SizeY
HealthBar	8	8	150	12
DashBar	481	8	150	12
AirBar	481	26	150	12
LoadBar	20	444	600	15

Remark: There is some rounding in the change between the full screen and the windowed mode, usually with gaps of 1 or 2 pixels.

CUST_constants

IdColor1

In this field type an Id to identify a ColorRGB= script command with a RGB colour to use as the main colour for the bar.

For example: To have pure Red type: ColorRGB=3, 255,0,0

The Id is 3, so type the value 3 in the IdColor1 field to set RED as the main colour (the main colour is the colour of the full bar).

WARNING: For the ColorRGB command referenced with the **IdColor** it is necessary to type it FIRST in the Customize=CUST_BAR command otherwise when the **TRNG** engine is parsing the **CUST_BAR** command the ColorRGB (if it is typed lower down) will result in a error “missing” and a black colour will be set.

The main colour defaults:	BarType	Red	Green	Blue
	Health Bar	255	0	0
	Dash Bar	0	255	0
	Air Bar	0	0	255

IdColor2

This field works like the **IdColor1** (See the above description) but it sets the background colour of the bar The default value is Black (0,0,0)

Extra

This field accepts different values in accordance with further **FBAR_** flags.

See the **FBAR_** constants

CUST_constants

48 \$0030: CUST_BIKE_VS_ENEMIES

Used with the Customize=

Used with the Customize Cust Bike vs Enemies command

Syntax: Customize=CUST_BIKE_VS_ENEMIES, Slot+HIT_ flag array

This customize chooses what will happen when the bike collides with a specific slot. In the past the bike killed all mortal enemies and it passed across immortals like they were made of air.

From 1.2.2.5 version there is a new collision management to choose what happens to enemies: killed, pushed away, hurt etc.

Use the HIT_ flags to set the behaviour of the impact with the slot item.

For example: If the SKELETON was killed type the command:

Customize=CUST_BIKE_VS_ENEMIES, SKELETON + HIT_KILL

also place two or more flags:

Customize=CUST_BIKE_VS_ENEMIES, SKELETON+HIT_KILL+HIT_PUSH_AWAY

in the above case, the SKELETON will be pushed away and killed at the same time.

Many slots can be customized with the same **CUST_BIKE_VS_ENEMIES** command.

For example: Customize=CUST_BIKE_VS_ENEMIES, SKELETON + HIT_KILL,
KNIGHTS_TEMPLAR+HIT_PUSH_AWAY

CUST_constants

Note: If the [CUST_BIKE_VS_ENEMIES](#) customization is omitted the default behaviour will be:

```
HIT_PUSH_AWAY, // SKELETON
HIT_PUSH_AWAY, // GUIDE
HIT_PUSH_AWAY, // VON_CROY
HIT_PUSH_AWAY, // SETHA
HIT_PUSH_AWAY, // MUMMY
HIT_WALL, // SPHINX
HIT_WALL, // HORSEMAN
HIT_WALL, // SCORPION
HIT_PUSH_AWAY, // JEAN_YVES
HIT_WALL, // KNIGHTS_TEMPLAR
HIT_PUSH_AWAY, // MUTANT
HIT_WALL, // HORSE
HIT_PUSH_AWAY | HIT_HURT, // DEMIGOD1
HIT_PUSH_AWAY | HIT_HURT, // DEMIGOD2
HIT_PUSH_AWAY | HIT_HURT, // DEMIGOD
HIT_PUSH_AWAY | HIT_HURT, // AHMET
HIT_WALL, // LASER_HEAD
HIT_WALL, // LASER_HEAD_BASE
HIT_WALL, // LASER_HEAD_TENTACLE
HIT_HURT, // HYDRA
```

All other enemies will be killed at the first impact.

Note: The HIT_HURT cannot be used for immortal enemies, but the HIT_KILL+HIT_EXPLODE flags can and they will be killed by an explosion.

See the [HIT_](#) flags.

CUST_constants

51 \$0033: CUST_BINOCULARS

Used with the Customize=

Used with the Customize Cust_Binoculars command

Syntax: Customize=CUST_BINOCULARS, FLAGS (**BINF_**), Parameter, CompassImage, CompassRectAndFlags, SextantImage, SextantRectAndFlags, LightnessRectAndFlags, ZoomRectAndFlags, LightSwitchRectAndFlags, FontId

This customize allows the setting of many new features for the binoculars management but it only works according to the use of the Customize= CUST_BACKGROUND, BKGD_BINOCULAR ... command.

Once the **CUST_BACKGROUND** is added for the binoculars, further customization can be made with the **CUST_BINOCULARS** command.

If the customized background flag is omitted for the binoculars the **CUST_BACKGROUND** will have no effect.

IMPORTANT NOTE: When the **CUST_BACKGROUND** command is used with the binoculars the image for the background can have any size.

To customize the advanced features for the binoculars with the **CUST_BINOCULARS** command use a background image size of **1024 x 768**.

This requirement is necessary to give the correct alignment between the background and the different values, bars or strips that will be added to it.

FLAGS (**BINF_**)

In this field add the **BINF_** flags to set the work mode of the binoculars. Most of these flags are to set special features for the binoculars.

Parameter

In this field there could be an extra setting according to the **BINF_** flag. See the **BINF_** flags

CUST_constants

CompassImage

If the **BINF_COMPASS** flag is used in the Flags field then in the next

CompassRectAndFlags field set the **BINT_STRIP** type.

Also set in the current **CompassImage** field the number of images to use for the compass strip.

Note: If the **BINF_COMPASS** flag is not used or the **BINT_NUMERIC** type is used for the compass, type **IGNORE** in this field.

The number of the image to type in this field is **NOT an Id** of some Image script command but the number of the image in the **\pix** folder.

For example: If the image with the compass strip is **image13.bmp**, type 13 in this field.
The compass strip image has to be exactly 1400x64 pixels.

It is easy to understand how it should be drawn if you look at the example of the lightning demo where it is used.

The compass strip is a graduated scale with cardinal points (north, east, west, south) and different degrees that will be used to indicate where the binoculars are aiming at that moment.

Note: The compass strip has a double scale to preserve infinite rotation because it is necessary to have two instances for each cardinal point and degrees.
Look at the example of the demo lightning to study how it works.

CompassRectAndFlags

If the **BINF_COMPASS** flag is enabled, type the id of the **PARAM_RECT** command where the position is stored and the size of the rectangle where the compass data will be drawn.

Then add to this rect Id, a **BINT_** type to set the compass data as showing.
Only use the **BINT_STRIP** or **BINT_NUMERIC** types for the compass.

Set the origin and size of the rectangle using as a reference a **screen of 1024x768 pixels**.
The rectangle should have a base wider than the height when a **BINT_STRIP** type is used for the compass

For the **BINT_NUMERIC** the difference between the base and the height should be less.

Remember that in the **BINT_NUMERIC** type the compass data will be printed with a format such as "NE 32" and therefore the characters require more space in the base of the rectangle.

Notes: This field is not for the use of the **BINT_BAR** type.
The size of the font used to print the text when the **BINT_NUMERIC** type is used is given by the height of the **PARAM_RECT** rectangle.
If a bigger character for the compass data is required increase the height field in the **PARAM_RECT** command.

CUST_constants

SextantImage

To show a sextant with vertical degrees where the binoculars are looking set the **BINT_STRIP** type for the sextant data.

Type a number in this field for the image of the sextant strip.
The number of the image in this field **is NOT an Id** of some Image script command but it is the number of the image in the **\pix** folder.

The **sextant strip image has to be 64x1520 pixels**.
The compass strip image and the sextant image always have a vertical shape.

Note: Create a sextant image of resolution 64 x1520, in the game you can show a slimmer strip.
Define in the **SextantRect** a rectangle with a width less than 64 pixels and in this case the left strip of the original image will be taken.

For the look of the image and how it works study the lightnings demo on the **TRNG** official website
(<http://www.trlevelmanager.net/ng.htm>) in the demo section.

SextantRectAndFlags

To use a sextant type the Id of the **PARAM_RECT** command where the position and size of the rectangle is stored that has the sextant data.
Also add to the rectangle Id the **BINT_** type about how to show the sextant data.
Choose the **BINT_STRIP** or **BINT_NUMERIC** types.

LightnessRectAndFlags

If the data for lightness is enabled, type the Id of the **PARAM_RECT** command with the position and size where it will be drawn.

Remember that the binoculars background is **always 1024x768 pixels** and the data of the rectangle should work according to this resolution.

Also add to the rect Id the **BINT_** type about how to show the sextant data.
Choose **BINT_BAR** or **BINT_NUMERIC** types.

Note: If the **BINT_NUMERIC** is chosen, the lightness level will be described in the ISO/DIN scale with values in the range between 1 and 39, where 39 is sunlight and 1 is dark.

ZoomRectAndFlags

To have the current zoom-in factor shown, add in the Flags field the **BINF_ZOOM** flag and type in the **ZoomRectAndFlags** field the Id of the **PARAM_RECT** where to show values about the current zoom factor.

This field can only be a **BINT_NUMERIC** type.

CUST_constants

LightSwitchRectAndFlags

If the **BINF_LIGHT_SWITCH** flag is added to the flags field, set the **LightSwitchRectAndFlags** field with the Id of the **PARAM_RECT** command with the information about the position for the light switch.

The light switch is a button with two states, on and off to show on the screen when the player enabled the illuminator in the binoculars view

In the **PARAM_RECT** use the **ForeColor** field to set the "on" colour of the button and the **backcolor** for the "off" state.

This button is a box of a given size, the only change is to add the **BINT_ROUND** type, to convert the box into a circle.

FontId

When a literal or numeric information is shown, choose the font to use for the text typing.

Note: The size of the font will be stretched or enlarged to be compatible with the height field of the rectangle for the given text.

CUST_constants

28 \$001C: CUST_CAMERA

Used with the Customize=

Change the properties of the Game Camera with this customize setting.

Syntax: Customize=CUST_CAMERA, Flags (**FCAM_**), DistanceChaseCam, VOrientChaseCam, HOrientChaseCam, DistanceCombatCam, VOrientCombatCam, DistanceLookCam, HeightLookCam, SpeedCamera

Remarks: The camera modes can be customized:
The "chase" camera that follows Lara,
The "combat" camera works like the "chase" camera but is engaged when Lara extracts the weapons

The "look" camera where Lara looks in the chosen direction.
The combat camera has some settings different from the "chase" camera.

Flags (**FCAM_**)

Type in this field one or more **FCAM_** flags to change some behaviour for the Camera.
See the **FCAM_** values

Remark: Type **IGNORE** in this field if no flag is set.

DistanceChaseCam

This value works for the "chase" (or "follow me") camera.
This field is the distance that the camera is behind Lara.
The **default value is 1536 (\$600)**, where 1024 (\$400) is one game sector.
If this value is reduced the camera will be closer to Lara.
If the value is increased Lara will appear further away.

Remark: If a negative value is typed in this field the camera will be in front of Lara instead of behind her.

It is interesting to note the **HOrientChaseCam** field with \$8000 value and the **DistanceChaseCam** with a negative value give the same target but only by using one of them at a time.

Type **IGNORE** in this field to keep the default value.

CUST_constants

VOrientChaseCam

This field works for the "chase" camera and it is the vertical difference of orientation (facing) of the camera with respect to Lara.

To understand "vertical orientation" try to look at the tutorial for animation and Test Position commands.

The **default value is -1820** and it is a difference (like degrees difference with respect to a line parallel to the floor). This means the camera is a bit higher than Lara.

If +1820 is chosen the camera will be a bit lower than Lara.

For -16384 the camera is exactly over the top of Lara (90 degrees with respect to the line parallel to floor).

For +16384 the camera is below Lara at a 90 degrees angle, but this setting will only work when Lara is monkey swinging or falling down.

For a 0 (zero) value the camera will have the same Y origin as Lara, because the camera and Lara are on the same line parallel to the floor.

Remark: Some settings in this field could be ignored when the value is large (absolute value) because it shows Lara upside down.

For example: The value -32767 should put a camera looking Lara in the face but with her appearing upside down.

Using this value (-32767) or -16384 gives the same result and this happens because the **TRNG** engine uses a cut-off function to stop Lara being shown upside down.

The valid range for this field is from -16384 to +16384

Type **IGNORE** in this field to use the default value.

CUST_constants

HOrientChaseCam

This works like the **HOrientDifference** for the Test Position command and it changes the facing of the Chase Camera

The orientation is named "facing" i.e. where an object is looking.

The values used could be:	\$0	default, the camera looks at Lara from the back
	\$4000	the camera looks at Lara from her left
	\$C000	the camera looks at Lara from her right
	\$8000	the camera looks at Lara from the front.

Type **IGNORE** in this field for the default value.

DistanceCombatCam

This field sets the distance where Lara is in modality with the combat camera.

The description about the concept of the "cam distance" is the same as the above field about the **DistanceChaseCam** .

The **default value is \$600**

VOrientCombatCam

This field works like the **VOrientChaseCam** field.

The **default value is -2730**

See the description for the **VOrientChaseCam** field to understand how it works.

Remark: There is no **HOrientCombatCam** field for the Combat camera because in "combat camera" mode the **HOrientCombatCam** is set by the **TRNG** engine according to the position of the selected enemy and Lara. No good result can be obtained by forcing the combat camera mode.

DistanceLookCam

This field works like the other "Distance ..." fields but with an opposite sign.

For example: The DistanceLookCam **default value is -1024** and this means that the camera is exactly one sector behind Lara.

If +1204 is used the Look camera will be forward of Lara and she will not be shown in the game screen.

CUST_constants

HeightLookCam

Type a Y displacement from Lara's neck.

The **default value is +16**. This means that the Y position of the camera is a bit lower than the centre of Lara's neck.

Remark: **DO NOT change this value.**

Using small values does give a noticeable difference while big values does not give good results.

SpeedCamera

This sets the time required to move the camera from the current position to an ideal position set by the field.

If a high speed is set the camera will move fast to reach the ideal position but the disadvantage could be a jerky movement.

Type **IGNORE** in this field to use the default value.

Remark: **DO NOT modify this value**

The default setting is an ideal value and the camera speed changes every time the **TRNG** engine requires to pass from one camera mode to another.

CUST_constants

23 \$0017: CUST_CD_SINGLE_PLAYBACK

Used with the Customize=

Customize the management of the CD audio track performed for a single playback.

Syntax: Customize=CUST_CD_SINGLE_PLAYBACK, CdMode value (**CDM_**)

This customize only works for saving and reloading a single-playback audio track in progress when the player saved the game.

See the different choices by looking at the **CDM_** constant descriptions

39 \$0027: CUST_DARTS

Used with the Customize=

Used by the Customize= Cust_Darts command.

Syntax: Customize=CUST_DARTS, IdCustDarts, Dart Flags (**DRT_**), IdAddEffect, Speed, EmittingTimer, PrimaryColorIDRgb, SecondaryColorIDRgb, IdTriggerGroup

This customize changes the speed and colour of the darts.

The darts are not very visible because they are fast.

Reduce the speed and change the colour to see them better.

Use this customize to restyle the dart by adding an effect or use them like a laser sensory ray.

IdCustDarts

Two or more customize profiles for darts can be set and then enabled at the same time by typing in the OCB field of the dart emitter object the **IdCustDarts** value to locate the customization to use.

For example: Create a command like: Customize=CUST_DARTS, 3,

When you want to customize a dart emitter in the **NGLE** with these settings type in the OCB field of this emitter the number "3"

Remark: The **IdCustDarts** value should be greater than 0.
Zero in the OCB field means: "No customize for this dart emitter"

Dart Flags (**DRT_**)

Add one or more **DRT_** flags to enable different features for the darts.

For no flag type **IGNORE** in this field.

See the **DRT_** constants

CUST_constants

IdAddEffect

If the **DRT_ADD_EFFECT** flag is used, type in the **IdAddEffect** field the Id of the AddEffect script command to add to each dart.

Speed

The speed value sets the value to add to the coordinates of the dart to move it in space.

Bigger values give higher speed.

The **default value is 256 (\$100)** and this is a very fast speed.

So, decrease the value to see the darts.

Type **IGNORE** in this field to use the unchanged speed.

EmittingTimer

This value is in tick frames, where one second = 30 tick frames.

The **default value is 24 tick frames** and this means the **TRNG** engine waits 24 tick frames (a bit less than one second) before shooting another dart.

Increase this value and there will be more time between shooting a dart and the next one.

Type **IGNORE** in this field to use the default value.

PrimaryColorIDRgb

To change the main colour of the dart type the Id of a ColorRGB= command with the RGB values to use.

The default colour is cream.

Type **IGNORE** in this field to use the standard colour.

SecondaryColorIDRgb

To change the secondary colour of the dart type the Id of a ColorRGB= command with the RGB values to use.

The secondary colour is black and is used for the borders.

It is a good colour for the border.

If the dart is used in a very dark room use white to give the idea of brightness to the darts.

Type **IGNORE** to use the standard colour.

IdTriggerGroup

If the **DRT_PERFORM_TRIGGERGROUP** flag is used type in the **IdTriggerGroup** field the Id of the Trigger Group to perform when a dart touches Lara.

See the description of the **DRT_PERFORM_TRIGGERGROUP** flag for more information.

CUST_constants

4 \$0004: CUST_DISABLE_FORCING_ANIM_96

Used with the Customize=

By default, when Lara is moving left or right in the hang mode and reaches a corner the **TRNG** engine forces animation 96.

Syntax: Customize=CUST_DISABLE_FORCING_ANIM_96

This behaviour is good, but if you are writing a new hanging animation the forcing of animation 96 could create trouble for the new hanging animation.

Using the custom value the **TRNG** engine will not force ANIM_96, allowing a custom animation to round the corner.

29 \$001D: CUST_DISABLE_MISSING_SOUNDS

Used with the Customize=

Used to customize the missing sounds.

Syntax: Customize=CUST_DISABLE_MISSING_SOUNDS

This setting is disabled in the **old tomb4** code for playing obsolete sound effects like the sound for bubbles (37 - LARA_BUBBLES) and for the pick up item (62 - LARA_PICKUP).

Use this setting when Boats or other new objects are used that use some of the above old sounds.

In fact, in an old example about boats on the **TRNG** website there was a boring bad sound when Lara swims underwater and also when she picked up an item.

The reason is the new boat object requires to use new sounds but there was no way to add a new sound slot. Unfortunately these missing sounds had a code in **tomb4** to play them so when the sounds for boats is added the **TRNG** engine played these sounds as bubbles or pick up.

Setting **CUST_DISABLE_MISSING_SOUNDS** the old codes for bubbles and pick up will no longer play sound 62 and 37, but new objects like boats will be able to play the new sounds assigned to the above sound slots.

50 \$0032: CUST_DISABLE_PUSH_AWAY_ANIMATION

Used in the Customize=CUST_BACKGROUND command.

This customize disables in the current level the push-away animation of Lara when she has been touched and moved by enemies.

The **push-away animations** are the **125, 126, 127 and 128** animations in the Lara slot.

CUST_constants

1 \$0001: CUST_DISABLE_SCREAMING_HEAD

Used with the Customize=

This disables the change from the normal Lara head (from Lara Skin slot) with the screaming head when she is shooting (from the Lara Scream slot).

Syntax: Customize=CUST_DISABLE_SCREAMING_HEAD

The reason to disable the change of heads could be when a standard swap mesh for Lara is performed but the screaming head remains untouched.

In this circumstance when a player shoots, the new Lara Head will be replaced with the old screaming head of Lara ruining the new mesh.

If this CUST value is set, Lara will have a single head avoiding the above problem after a swap mesh operation.

Remark: Another way to solve the problem is to use an advanced Swap mesh replacing the mesh in the Lara Scream slot, using a swap mesh where a "Shooting Head" is present.

20 \$0014: CUST_ESCAPE_FLY_CAMERA

Used with the Customize=

Used to allow a player to break a Flyby camera sequence by hitting the **ESC** key.

Syntax: Customize=CUST_ESCAPE_FLY_CAMERA, ENABLED/DISABLED, KeyBoardScanCode

To allow the break from a Flyby sequence set this customize as ENABLED.

The KeyBoard Scan code value will be the code of the keystroke used to exit the Flyby sequence.

Find the list of scan codes in the **KEYBOARD SCAN CODES**.

Type **IGNORE** in this field and the default value will be the **ESCAPE** key.

Remark: To permit skipping of the flyby cameras it is advisable to set some print text in the level to inform the player about this new change. Then you know to hit **Escape** if required during the Flyby sequences.

CUST_constants

53 \$0035: CUST_FIX_BUGS

Used with **Customize=** command.

Syntax: Customize=CUST_FIX_BUGS, BugsToFix flags (**BUGF_...**)

This customize accepts one or more **BUGF_...** flag to fix a bug in the tomb raider engine. The reason to let customizable bug fixing is because some level builders could have built a skill based on a bug in the old engine. Therefore fixing a bug in a critical module of the tomb raider engine could generate some unwanted collateral effect.

Note: In the **BUGF_** flags you will find some bug fixing already seen with other mnemonic constant in other script commands like

CUST_FIX_WATER_FOG_BUG or **DRT_FIX_POISON_BUG**.

You can fix these bugs two ways: with the old constants or with the new **BUGF_** constants. In the future all bugs will be fixed using **BUGF_** constants to find them in a single command.

See the **BUGF_** constant for more information.

32 \$0020: CUST_FIX_WATER_FOG_BUG

Used with the **Customize=**

Used to fix the bug in the water textures when there is Distance Fog enabled in the level.

Syntax: Customize=CUST_FIX_WATER_FOG_BUG, ENABLED/DISABLED

When this fix is enabled the water textures will ignore the Fog density.

Use it when the level has a depth fog.

This fix could create some trouble with fog bulbs close to the water.

CUST_constants

40 \$0028: CUST_FLARE

Used with the Customize=

Used to customize flares.

Syntax: Customize=CUST_FLARE, Flags for Flare (**FFL_**), SecondsOfLifeTime, Red, Green, Blue, Intensity

Flags for Flare (**FFL_**)

Add one or more **FFL_** flags to enable special features about the flare.

Type **IGNORE** to not set flags.

See the **FFL_** constants.

SecondsOfLifeTime

To override the duration of the flare type a value in seconds for the life-time of the lightning flare.

The **default value is 30 seconds**.

Type **IGNORE** to use the default value.

Red, Green, Blue

In these three fields set the middle colour for the flare light.

Remark: This colour will not be applied in the first and last phase of the life-time of the flare: the switching on and the switching off phases.

Type **IGNORE** in these three fields to use default values.
(Red = 128 , Green = 192, Blue=0)

Intensity

Change the middle light intensity of the flare.

The **default value is 16**.

Type **IGNORE** to use the default value.

31 \$001F: CUST_FMV_CUTSCENE

Used with the Customize=

Used to enable the customize viewing of FMVs (cut scenes based on videos).

Syntax: Customize=CUST_FMV_CUTSCENE, **FMV_** flags

Type one or more **FMV_** flags to set some features about the showing of videos (FMV) in the game.

See the **FMV_** constants

CUST_constants

17 \$0011: CUST_HAIR_TYPE

Used with the Customize=

Used to force the hair type of Lara in the game.

Syntax: Customize=CUST_HAIR_TYPE, HairType (**HAIR_**)

Choose the kind of hair by setting a **HAIR_** constant value.

See the **HAIR_** constants

Remark: This command is not able to change the mesh of the hair but it is used to inform the **TRNG** engine about what kind of hair object is copied in the wad file.

For example: A Young Lara (from Angkor wat) can be used but do not set it in the **script.dat** with the command YoungLara=ENABLED.

Young Lara will be set with the correct hair type using CUST_HAIR_TYPE and the weapons for the adult Lara because the **TRNG** engine will consider Lara as adult because there is no YoungLara=ENABLED command .

This customize value **CUST_HAIR_TYPE** is useful to divide the setting about hair from that about weapons.

If the old command Young Lara is used the setting for the hair cannot be separated from the setting for weapons.

CUST_constants

35 \$0023: CUST_HARPOON

Used with the Customize=

Used to customize the harpoon.

Syntax: Customize=CUST_HARPOON, HarpoonFlags (**HRP_**),
TopBorder,DistanceFromCam,Orient_X, Orient_Y, Orient_Z,
HarpoonSpeed, HarpoonGravity

This customize changes the Crossbow to get a Harpoon weapon like the Raider 3 adventure.

To use this customize it is necessary to copy in the wad file the **CROSSBOW_ANIM** slot found in the harpoon.wad

That slot has been changed from the original **HARPOON_ANIM** in the Tomb Raider 3 level.

HarpoonFlags (**HRP_**)

Add one or more **HRP_** flags to enable specific settings for the new Harpoon gun.

TopBorder

Adjust the position of the Harpoon moving up or down in the 2d view.

To move the Harpoon UP decrease this number.

To move the Harpoon DOWN increase this number.

Type **IGNORE** for the **default value =0**.

DistanceFromCam

Distance from the (virtual) camera.

This argument is useful to increase or reduce the size of the object in the Inventory.

If the distance is increased the object will become smaller

If the distance is decreased the object will become bigger.

This value is in game units where one square = 1024 units.

Type **IGNORE** to use the **default value: \$400**

Orient_X, Orient_Y, Orient_Z

These 3 fields set the orientation of the camera on the X, Y and Z axis.

The values for this argument could go from \$0000 to \$FFFF but usually only four values will be used:

\$0000	=	North (top view)
\$4000	=	East (right view)
\$8000	=	South (bottom view)
\$C000	=	West (left view)

Type **IGNORE** to use the default values:

Orient_X	=	\$B000
Orient_Y	=	\$C000
Orient_Z	=	\$C000

CUST_constants

HarpoonSpeed

The speed of the Harpoon can be changed.

Type **IGNORE** to use the default value,
that is in accordance with the current inclination with respect to the ground.
Reasonable values for this field are 100 to 300.

The speed will be different when the Harpoon travels in air or in water.
The speed in water will be a 1/4 less than the value used for air.

For example: For a value of 200 in this field,
The speed in air will be 200,
The speed underwater it will be 150.

HarpoonGravity

The gravity is an increment to move the Harpoon down, to simulate gravity.
The value is used in a different way if the harpoon is moving in air or in water.
When the harpoon is in air it will use the original **HarpoonGravity** value.
When the harpoon is travelling in water the gravity will be divided by 2 before using it.

For example: Type 16 in this field,
The gravity in air is 16.
The gravity in water is 8.
Type 0 and the Harpoon will not be affected by gravity.

Type **IGNORE** to use the **default value of 24**.

CUST_constants

30 \$001E: CUST_INNER_SCREENSHOT

Used with the Customize=

Used to enable the saving of the game image in all save games

Syntax: Customize=CUST_INNER_SCREENSHOT, QualityScreenshotFlags (**QSF_**)

When the **CUST_INNER_SCREENSHOT** is set in a level section all save games will be saved with a little image of the current game image.

This option is used by TRLM2009 to see an image corresponding to all save games and also for the new save game panel to set the quality of inner images.

QualityScreenshot (**QSF_**)

Set in this field two **QSF_** flags.

One **QSF_SIZE_** flag and (optional) a **QSF_TRUE_COLOR** flag.

The settings to use for the inner screen shots should be a good quality with a big size of save game, or a low quality with a small save game.

See the **QSF_** constants

18 \$0012: CUST_KEEP_DEAD_ENEMIES

Used with the Customize=

By default when an enemy dies, his body disappears.

To keep his body forever like in previous Raider adventures enable this command

Syntax: Customize=CUST_KEEP_DEAD_ENEMIES, ENABLED/DISABLED

Remark: The bodies killed by explosion will NOT be preserved.

CUST_constants

47 \$002F: CUST_KEEP_LARA_HP

Used with the Customize=

Used to customize Lara's health points.

Syntax: Customize=CUST_KEEP_LARA_HP, TargetLevel

This customize stops the automatic recharging of Lara Health Points when she jumps to another level from the current level.

TargetLevel

Type in this field the index of the target level where Lara will arrive keeping a previous Health Points value.

Example: Suppose that the adventure has three levels, where level 1 and level 2 are the two halves of a wide environment and Lara jumps in and back from these two levels many times.

Then there is a level 3 where Lara will only go after solving all the puzzles of the first two levels.

In this situation Lara should not be able to recharge her Health Points by going in and out through the gate between level 1 and level 2.

To keep the same Health Points in this situation place in the [Level] section of level 1 the command:

Customize=CUST_KEEP_LARA_HP, 2

While the in the [Level] section of level 2:

Customize=CUST_KEEP_LARA_HP, 1

For level 3 it is not necessary to type anything since this command is missing by default, Lara will be recharged when she jumps to level 3

Remark: To keep the Health Points of Lara, i.e. no recharge for all levels, use the special constants **KLH_ALL_LEVELS** instead of the target level:

Customize=CUST_KEEP_LARA_HP, KLH_ALL_LEVELS

For any level where Lara jumps, starting from the current, she will keep the old Health Points value.

CUST_constants

34 \$0022: CUST_LIGHT_OBJECT

Used with the Customize=

Used to customize a light object.

Syntax: Customize=CUST_LIGHT_OBJECT 34, SlotLight, Red, Green, Blue, Intensity,
Time

The colours and light intensity of these light objects can be modified: **AMBER_LIGHT**
WHITE_LIGHT
BLINKING_LIGHT

Change the main colour of these objects, their light intensity and in some circumstances the blinking time or other timing parameter.

SlotLight

Type the slot name or number of the light object to customize.

Currently the only light objects allowed are: **AMBER_LIGHT**,
WHITE_LIGHT,
BLINKING_LIGHT

Red, Green, Blue

Type the intensity of the **RED**, **GREEN** and **BLUE**.

Each colour intensity works in the range 0 to 255, where 255 is maximum intensity and 0 is null.

Intensity

Set the brightness and width of the light.

Type **IGNORE** to use the **default value of 18**.

Time

All lights have some blinking and this field changes the time of the blinking.

The default values for the light objects: **AMBER_LIGHT** = -2048
WHITE_LIGHT = 160
BLINKING_LIGHT = 30

16 \$0010: CUST_LOOK_TRANSPARENT

Used with the Customize=

ENABLE or Disable the transparency of Lara when she is Looking.

Syntax: Customize=CUST_LOOK_TRANSPARENT, ENABLED/DISABLED

To remove the transparency for Lara use DISABLED

CUST_constants

6 \$0006: CUST_NEW_SOUND_ENGINE

Used with the Customize=

Syntax: Customize=CUST_NEW_SOUND_ENGINE, NewSoundEngine flags (**NSE_**), SoundExtension (**SEXT_**), LongFadeOut, ShortFadeOut

To customize some features of the New Sound Engine based on the bass.dll created by Un4seen Developments Ltd and supported by the **TRNG** engine.

Remember that the new sound engine will be enabled as default, disable it using the script command: NewSoundEngine= DISABLED

Field description for CUST_NEW_SOUND_ENGINE: NewSoundEngine flags (**NSE_**) field

Change some features of the **New Sound Engine** by adding one or more **NSE_** flags. See the **NSE_** constants

Remark: **Currently there are no NSE_ constants available, but some new NSE_constants will be added in the future.**
For now type **IGNORE** in this field

SoundExtension (**SEXT_**)

This field is now obsolete. Typing a value will have no effect.

In **TRNG previous to 1.2.0.2**, it was possible to choose the extension of the audio track (.wav, .mp3, .ogg)

Now the **TRNG** engine uses an automatic method to detect the audio tracks to play. The **TRNG** engine checks for each track number (the first three digits "020" ...) and what extension is available.

If there is a single file with that track number that audio track will be played.

When there are two or more audio files for the same track number, the **TRNG** engine will play the first file in the following extension sequence: ".ogg", ".mp3", ".wav" ... all other supported audio format from bass.dll

CUST_constants

For example: If in the audio folder there are the following files:

- 034.wav
- 034_Interlude.mp3
- 159_FlyBy.mp3
- 159_TopView.ogg
- 005.wav

the **TRNG** engine will play the audio files:

- 034_Interlude.mp3
- 159_TopView.ogg
- 005.wav

This happens because the priority is for ".ogg" files,
if the .ogg file is missing it will search for a .mp3 file,
if it is missing the .wav file or other audio extensions (.mp2, .aiff etc.) will be played.

Remark: When there are two audio tracks with the same number and the same .wav extension, the **TRNG** engine will play the long name audio track.

For example: If in the audio folder there are the following files:

- 005.wav
- 005_Secret.wav

the **TRNG** engine will play the file "005_Secret.wav"

LongFadeOut

For "Fade out" there is a gradual reduction of volume to the null volume.
The fade out is used to stop a sound in a sweet way.
Type a value in microsecond units and it should be the time required to pass from the current volume to zero volume.

The "long fade out" is used when a sound is stopped for some change in the game like loading of a new level.

The **default value for a long fade out is 1000 ms**, i.e. one second.

Type **IGNORE** to use the default value.

CUST_constants

ShortFadeOut

For the meaning of "Fade out" read the description for the **LongFadeOut** field.

The "short fade out" is used when a CD sound is stopped to permit another CD sound to start in the same channel.

In this situation the fade out should be shorter and the **default value is of 300 ms**.

Type **IGNORE** to use the default value.

Remark: This customize command like all other customize commands, works for all levels if it is placed in the [Title] section

The Customize=CUST_NEW_SOUND_ENGINE can be placed in different levels

It is better not to change the default extension for audio files otherwise there could be trouble.

The values for the fadeout can be changed , level for level.

To use different **NSE_** flags in different levels read the specific description for the **NSE_** flags.

CUST_constants

26 \$001A: CUST_NO_TIME_IN_SAVELIST

Used with the Customize=

This setting disables the text about the game time in the save game list screen.

Syntax: Customize=CUST_NO_TIME_IN_SAVELIST, ENABLED/DISABLED

To remove the time game information type:

Customize=CUST_NO_TIME_IN_SAVELIST, ENABLED

This option is useful only when very big (in width size) text is set, when the font is too big the level name could overlap the time information with a very nasty effect.

If the time information is removed it is possible to use wider characters

27 \$001B: CUST_PARALLEL_BARS

Used with the Customize=

This item in the current level can be customized.

Syntax: Customize=CUST_PARALLEL_BARS, FlagsParallelBar (**PB_**), SpeedForSlide, MaxTurns

FlagsParallelBar (**PB_**)

Add one or more **PB_** flags to customize all parallel bars in the level.

See the **PB_** constants

SpeedForSlide

If the flag **PB_LARA_CAN_SLIDE** is used, set the speed for Lara sliding left or right.

The value has a unit for one block = 1024.

The **default value is 6**.

Type **IGNORE** to use the default value.

MaxTurns

By default the maximum number of turns is 10.

To increase the maximum power add turns.

Reasonable values are in the range 3 to 10

Type **IGNORE** to use the default value.

CUST_constants

21 \$0015: CUST_PAUSE_FLY_CAMERA

Used with the Customize=

Used to allow a player to stop the Flyby sequence temporary by pressing a key.

Syntax: Customize=CUST_PAUSE_FLY_CAMERA, ENABLED/DISABLED,
KeyBoardScanCode

The KeyBoard Scan code value will be the code of the keystroke to use to freeze a flyby sequence.

Find the list of scan codes in the **KEYBOARD SCAN CODES**.

Type **IGNORE** to use the default value 25 (Key P)

Remark: It is necessary to continuously press the key to freeze the Flyby camera.

CUST_constants

37 \$0025: CUST_RAIN

Used with the Customize=

Used to customize Rain.

Syntax: Customize=CUST_RAIN, FlagsForRain (**FR_**), DropSize, SprinklerAmount, MaxRain, MinRain, Float1, Float4, Float8, Float16, Extra

This command is not necessary for the Rain.

This command can be omitted.

Use the **CUST_RAIN** only when the **RAIN_ALL_OUTSIDE** is set in the Rain= command and the [Rain] intensity in the rooms of the level is omitted.

In this situation use the **CUST_RAIN** customize to set the features of the rain for the level.

Remember that if the **CUST_RAIN** is used and there are different settings for rain in different rooms (**RAIN_SINGLE_ROOMS** mode, with intensities 1/4 closed to [Rain] button) these differences will be small or absent because the custom rain overrides the individual settings for a room.

FlagsForRain (**FR_**)

Type one or more **FR_** constants.

Type **IGNORE** to omit the **FR_** flags.

DropSize

This is the width of a water drop.

Each drop is like a triangle where the base is the **DropSize** value.

Type **IGNORE** to use the **default value 2**.

SprinklerAmount

This is the number of Sprinkles for each drop when it touches the ground.

Type **IGNORE** to use the **default value 1**.

Remark: Increase in this value could create trouble with other particle resources like smoke and flames.

CUST_constants

MaxRain and MinRain

A range to set the density of rain, i.e. the number of drops (**MinRain**) in a given space (**MaxRain**).

When the **MaxRain**, is decreased the space covered by the rain in front of Lara will be reduced.

When the **MinRain** is increased the number of single drops will be increased.

Remember trying to get a wide space covered by thicker rain could give bad results.

Balance these two values: when one is increased, decrease the other and vice-versa.

Type **IGNORE** to use the default values:

MaxRain	=	\$800,
MinRain	=	\$80

Float1

Unknown

Float4

Unknown

Float8

Unknown

Float16

Unknown

Extra

Set an extra value required by some **FR_** flags.

See the description of the **FR_** constants to discover how to use the **Extra** field.

CUST_constants

44 \$002C: CUST_ROLLING_BOAT

Used with the Customize=

Used to customize the rolling boat.

Syntax: Customize=CUST_ROLLING_BOAT, SlotBoat, FlagsRollingBoats (**FRB_**),
SwingingSpeed, PitchingSpeed, RollingSFX

By default, all boats when Lara is not on board are still.
This is not normal when there are waves on the water surface.

With this customize set a swinging and/or pitching movement to get more realistic boats on water.

Use one or more Customize=CUST_ROLLING_BOAT commands in the same level section to have different rolling for different boats.

SlotBoat

Type the slot of the boat to roll.

Remark: Theoretically a slot different from boats can be used, for example, create a fake boat in an animating slot and move it in a realistic way.

Note: When the **KAYAK** is set in the **SlotBoat** remember that the rolling will only be performed when the kayak is on the water.

FlagsRollingBoats (**FRB_**)

Add two or more **FRB_** flags to set the level of the rolling movement.

The **FRB_** flags set the width of the rolling movement for pitching or sliding.

For example: Using the **FRB_PITCHING_HIGH** flag the boat will move above the peak above the water surface more than the **FRB_PITCHING_LOW** flag, where the peak will move only a bit over the water surface.

See the description of the **FRB_** constants for more information.

SwingingSpeed

The swinging rolls the boat for the waves that hit the boat on the sides.

Type 0 for no swinging.

The speed value in this field works like degrees but in a specific format of **Tomb4** where 90 degrees = \$4000.

Type a value that will be added (or subtracted) by the current facing 30 times a second
Reasonable values are in the range \$08 to \$100 (8 to 256).

CUST_constants

Tips: When a LOW value is set for swinging it is better to reduce the swing speed to compensate the reduction of "distance" covered in the swinging.

Normally the swinging is less visible than pitching because the shape of the boat is slim. It is better to set higher values for swing speed.

PitchingSpeed

The pitching rolls the boat for the waves that hit the boat on the peak.

Type 0 for no pitching.

Read the description of the **SwingingSpeed** field.

RollingSFX

Type the number for a SFX sound.

This sound will be played when the rolling is at a maximum value, creating the effect of a boat touching other boats or water surf.

Type **IGNORE** for no sound.

CUST_constants

5 \$0005: CUST_ROLLINGBALL_PUSHING

Used with the Customize=

Changes features activated with OCB = 4 or OCB = 8 for the Rolling Ball.

Use OCB = 4 or OCB = 8 to enable Lara to move the Rolling Ball pushing it like it was a moveable object.

Customize the effects : used animations
 time for moving,
 distance for activation of pushing etc.

Syntax: Customize=CUST_ROLLINGBALL_PUSHING, Distance, PushAnim,
 FailedAnim, FrameOfMoving, FrameOfActivation,
 FramesOfInvulnerability, Speed

Remark: Type **IGNORE** and the **TRNG** engine will use the default value in the hard-coded mode.

Distance

The distance between Lara and the centre of the Rolling Ball.

Only when Lara is at the correct distance will she be able to push it.

The **default distance value is 600**,

where 1024 is the size of a single sector of the game and one click is 256 units.

The default distance (600) is good for a Spiked Rolling Ball found in the tut1.wad.

If a non spiked rolling ball is used increase the distance because Lara will be further away from the Rolling Ball.

PushAnim

Use this field to change the animation used to show Lara pushing the Rolling Ball.

By default the animation is ANIM_316, i.e. the same used to push the big pushable button used as a switch.

FailedAnim

This is another animation number.

It is used when Lara is not able to move the Rolling Ball because there is some obstacle on the opposite side of the Rolling Ball

The default for the FailedAnim is animation 120.

CUST_constants

FrameOfMoving

This is the frame number of the **PushAnim** animation from the **TRNG** engine that will move the Rolling Ball in a passive way.

For "passive way" the Rolling Ball is not moved by itself but is pushed by Lara. The default value is 20th frame for the **PushAnim** animation.

The interval for the Rolling Ball to be moved in a passive way is the range between the **FrameOfMoving** and the **FrameOfActivation**.

Type a **FrameOfMoving** number bigger than the **FrameOfActivation** and the Rolling Ball will never move.

FrameOfActivation

This is the frame number of the **PushAnim** animation when the rolling is enabled and it will begin to move by itself.
The **default value is 50th frame**.

FramesOfInvulnerability

This is the number of frames of animation following the **PushAnim** animation when Lara will be invulnerable.

This invulnerability is necessary because Lara would be killed by rolling because she is very close to the Rolling Ball.
So give Lara time to get away from the Rolling Ball so there is a time of invulnerability.

The **default value is 30 frames** where 1 second = 30 frames.

Remark: While Lara is performing the **PushAnim** animation she is always invulnerable.

Speed

This is the increment used to move the Rolling Ball when Lara pushes it.

Each frame will be added to the "speed" increment to correct the coordinate (X or Z).
The **default value is 6**.

Increase this value and the pushable will be moved faster by Lara.

For example: If there is trouble seeing Lara in the Rolling Ball pushing phase, reduce the **FrameOfMoving** value to move the Rolling Ball and or increase the speed to move the rolling faster.

CUST_constants

33 \$0021: CUST_SAVE_LOCUST

Used with the Customize=

Used to customize the Locust save.

Syntax: Customize= CUST_SAVE_LOCUST, ENABLED

This customize fixes an old bug of the **tomb4** engine:

When the locust swarm is enabled
and the player saved the game;

at reload the locusts are missing.

The reason for this bug is that the Eidos' programmers probably omitted the saving of data for the locusts.

Now you can save all the data for current particle of the **LOCUST_EMITTER** slot,
just add in the [Title] (for all levels) or in a specific [Level] section:

Customize=CUST_SAVE_LOCUST, ENABLED

Remark: The save game will be bigger by about 2000 bytes to enable the saving of locusts.

CUST_constants

36 \$0024: CUST_SCREENSHOT_CAPTURE

Used with the Customize=

Used to customize the screenshot capture.

Syntax: Customize=CUST_SCREENSHOT_CAPTURE, SecondsOfDurate, FrameGap, QualityScreenshotFlag (**QSF_**)

This command should only be used in a temporary way since it could slow down the game.

By default when the **F3 key** (or "." key on US keyboards) is pressed the game saves the current screen in an image file with progressive names "shot1.bmp", "shot2.bmp" etc.

Using the **CUST_SCREENSHOT_CAPTURE** setting a long series of captured frames can be made instead of a single image like in the past.

This feature is useful when it is difficult to find the best image in a very fast action in the game.

SecondsOfDurate

Set how much time the capture sequence will work for.

For example: Type 3, the game screen will capture for 3 seconds continuously.

FrameGap

To reduce the number of captured images set an interval in the capture phase.

Type 0 , the game will create images for each frame (so, no gap in this case)

Type 1 , it will capture one frame every two.

Type 2 , it will capture one frame every three, etc.

QualityScreenshotFlag (**QSF_**)

Another reason to capture a sequence of images is to show dynamic action without using a movie. In this situation, sending many images via the internet it is advisable to reduce the size.

Type a **QSF_** value to reduce the size or the quality (omitting the **QSF_TRUE_COLOR** flag) of the captured images.

Type **IGNORE** to get a full screen image with maximum quality.

CUST_constants

3 \$0003: CUST_SET_CREDITS_LEVEL

Used with the Customize=

The credit level should be the last level of your adventure.

In **Tomb Raider The Last Revelation** the number was 39.

The target to reach is to have scrolling text with credits at the end of your level.

Substitute the credit texts with your custom texts.

Find this text in the [PSXString] of the language text file.

The first text is "Programmers" (index =246)and the credit list continues until the end of the [PSXString] section.

Syntax: Customize=CUST_SET_CREDITS_LEVEL, NumberOfEndLevel

CUST_constants

12 \$000C: CUST_SET_INV_ITEM

Used with the Customize=

Sets Inventory items like INVARIABLE items, i.e. items that will remain in the inventory after it is used, like the Binoculars.

Syntax: Customize= CUST_SET_INV_ITEM, SlotOfItem

Remark: Place one or more
Customize=CUST_SET_INV_ITEM commands
in the same level section so to assign the property to many inventory items.

CUST_constants

13 \$000D: CUST_SET_JEEP_KEY_SLOT

Used with the Customize=

Changes the slot used to store the JEEP key.

Syntax: Customize= CUST_SET_JEEP_KEY_SLOT, SlotForJeepKey

The reason to modify the default slot (default slot was number 175, corresponding to PUZZLE_ITEM1) JEEP key, when the JEEP and the SIDECAR are used in the same level, using the new features added in **TRNG from version 1.1.8.6**.

When the SIDECAR and the JEEP is used in the same level:

Use the last version of **Wad Merger** (**version 1.98.0.103 or higher, released in may 2008**), and copy in its installation folder the new "TR4Objects.dat" file you find in the Multiple Vehicles sample zip file.

Copy the **JEEP** slots (**JEEP** and **VEHICLE_EXTRA**) into the wad.

Copy the **MOTORBIKE** slot into the wad.

Copy the **VEHICLE_EXTRA** slot from the wad with the **SIDECAR** to a new slot **MOTORBIKE_LARA** in the wad.

To perform this operation click on the [Copy] button (in **Wad Merger**) keeping down the SHIFT key. It will show a list of all available slots and at the bottom of this list you will see the slot named **MOTORBIKE_LARA**.

It is the new slot created for the Lara's animations for the motorbike.

Performing the above operation means that the **JEEP** and the **SIDECAR** can be in the same level, but there is a small problem.

The slot used for the **JEEP** key is the same one used for the "Nitrous Oxide Feeder" used to enhance the speed of the **SIDECAR**.

Since both objects (JEEP key and Feeder) are in the slot **PUZZLE_ITEM1**, there is a conflict.

To solve this problem use this customize command to set another slot for the JEEP key. The **PUZZLE_ITEM1** will only work for the Nitrous Oxide Feeder, while the key to switch on the JEEP will be stored in another slot with this custom command.

For example: To store the JEEP key in the **PUZZLE_ITEM2** (slot 176) type in the [Level] section of the level the following script command:

Customize= CUST_SET_JEEP_KEY_SLOT, 176

or (same result): Customize= CUST_SET_JEEP_KEY_SLOT, PUZZLE_ITEM2

CUST_constants

19 \$0013: CUST_SET_OLD_CD_TRIGGER

Used with the Customize=

Used to restore the old behaviour of the CD trigger.

Syntax: Customize=CUST_SET_OLD_CD_TRIGGER, ENABLED/DISABLED

Using the new bass sound engine the CD trigger changes its behaviour.

The looped sounds play on channel1 while the single-shot CD (track number less than 105) play on channel2, allowing channel 1 to play the previous audio track.

To restore the old default method set the following command in your [Level] section:

Customize=CUST_SET_OLD_CD_TRIGGER, ENABLED

The old method played all of the CD trigger sounds on channel1.

When a single-playback sound stopped the previous looped audio track is restored on channel1 from the start.

2 \$0002: CUST_SET_SECRET_NUMBER

Used with the Customize=

Arguments: NumberOfSecrets

With this customize substitute text in the Statics screen of the game in the row:
Secrets Found ".. / 70".

The value for NumberOfSecrets will substitute the default "70" for the maximum amount of secrets.

To use this customize first way:

To show a different maximum amount of secrets, levels for levels,
i.e. each level has a different amount of secrets for that level.

Syntax: Customize=CUST_SET_SECRET_NUMBER, NumberOfSecrets

In each level of the script dat (excluding Title level)

To use this customize second way:

Modify the Global amount of secrets and it will be the same for all levels.

In this case use a single Customize line and place it in the level section of the Title level.

Typing a single **CUST_SET_SECRET_NUMBER** in the title level
will affect the setting for all levels.

CUST_constants

15 \$000F: CUST_SET_STATIC_DAMAGE

Used with the Customize=

Changes the value of damage or poison occurring to Lara when she touches a static with OCB= 32 or OCB= 256

Syntax: Customize=CUST_SET_STATIC_DAMAGE, Damage, PoisonIntensity

Damage

Type a value between 1 and 999.

If 999 is set Lara will be killed immediately.

The **default value is 10**.

PoisonIntensity

Type any value for poison.

Reasonable values are in the range 10 to 4096

A big scorpion and Harpy have a poison intensity of 2048, while a little scorpion is 512.

The **default is 256**.

CUST_constants

42 \$002A: CUST_SET_STILL_COLLISION

Used with the Customize=

Used in the Customize= command.

Syntax: Customize=CUST_SET_STILL_COLLISION, Collision Flags (**COLL_**),
LowerHeight, MoveableArray

This customize enables a still collision method when Lara touches static or moveable items.

Remark: Some people call the "still" collision "hard" collision.

By default, when Lara touches an item she continues to move her legs and arms.
This behaviour is different from Lara touching a wall, where she stops immediately.

Use the **CUST_SET_STILL_COLLISION** to give the item the same collision behaviour seen in the wall collision.

CollisionFlags (**COLL_**)

Type, one, two or more **COLL_** flags to enable the still collision with different types of items.

See the **COLL_** constants

LowerHeight

Type the lower (acceptable) height that an item has to have to enable the still collision.
Theoretically it can have 0 in this field and all selected items will enable the still collision.

Think about the problem:

When Lara touches a small (low) item with her feet, a one-click height item, it would be weird to see her stop immediately like the item was an impassable obstacle.

In this situation perhaps it is better to have the old "sliding" collision used with items.

When a low obstacle is moved up like a low wall, Lara should be able to walk over it, while with statics and moveables this is not possible.

Therefore there is a problem to solve about the realism.

Type **IGNORE** to use the (reasonable) value of 300 units as a lower acceptable height.

Remark: Remember the units used in this field are:

one click = 256 units,
one sector = 1024 units.

CUST_constants

MoveableArray

Type one or more moveable slots using the still collision.
This field can also be left empty.

Remember that different moveable classes can be selected using the **COLL_** flag, so use these **COLL_** flags.

When some moveable is not enclosed in the **COLL_** flags use the **MoveableArray**.

For example: For still collision with the moveable **POLEROPE** and with the **ROLLINGBALL** type a customize command like this:

```
Customize=CUST_SET_STILL_COLLISION, COLL_STATICS+COLL_DOORS,  
          IGNORE, POLEROPE, ROLLINGBALL
```

To have still collision with statics and doors and still have collision with the **POLEROPE** and **ROLLINGBALL**.

Another use of the **MoveableArray** is when some moveable must be excluded from the still collision.

This trick will have a sense only when some **COLL_** moveable types are input.

For example: Suppose the still collision for all doors (**COLL_DOORS**) is enabled but it is discovered that there is a problem with a door used for a "race verses time". In this situation when Lara touches the door while it is closing, Lara froze immediately.

To let the player "slide" over the closing door disable the still collision only for that type of door, preserving the other doors.
In this case type the number of the slot to exclude as a negative number.

For example: To exclude the **DOOR_TYPE4** with slot number 325, type this script command:

```
Customize=CUST_SET_STILL_COLLISION,  
          COLL_STATICS+COLL_DOORS, IGNORE, -325
```

Remark: It is better not to use the syntax "-DOOR_TYPE4" because the mixing of mnemonic constant and arithmetic sign "-" minus, could confuse the compiler in some circumstances.

CUST_constants

41 \$0029: CUST_SET_TEXT_COLOR

Used with the Customize=

Used to customize the text colour.

Syntax: Customize=CUST_SET_TEXT_COLOR, Text Type (**TT_**), Color for Text (**CL_**)

This customize changes the default colour for the text in the menu, options screen, inventory mode, statistics screen etc.

Set a new colour for the text type by typing the text to change, with a **TT_** constant value and typing the **CL_** colour.

For example: To have the main menu drawn in a red colour type in the [Level] section this command:

Customize=CUST_SET_TEXT_COLOR, TT_MAIN_MENU, CL_RED

Type two or more Customize=CUST_SET_TEXT_COLOR commands in the same [Level] or the [Title] section if required.

Remark: It is advisable to place this customize in the [Title] level to change text in the main menu, options or a new level screen.
The customize typed in the [Title] section will work for all levels until another specific customize is placed in a [Level] section.

CUST_constants

45 \$002D: CUST_SFX

Used with the Customize=

Used to customize SFX sounds.

Syntax: Customize=CUST_SFX, TypeSound (**TS_**), Sound Effect Number

This overrides the sound effect used in some **TRNG** features, like the Diary.

Use many Customize=CUST_SFX commands in the same [Level] section if required.

TypeSound (**TS_**)

Choose a **TS_** constant to set what sound type to change.

See the **TS_** constants

Sound Effect Number

Type the number of a new sound to use.

See the **SFX SOUNDS** for numbers.

2 \$0002: CUST_SFX_DEMO

Used with the customize command.

Syntax: Customize=CUST_SFX_DEMO, CleanerTouchLaraSfx, SWRobotShootsLaraSfx, CraneReleaseJawsSfx, CraneMovingSfx, CraneTouchFloorSfx

You can change the sound effect for the given sound type.

To change some sounds and leave unchanged others type **IGNORE** in the field to leave a default sound.

CUST_constants

8 \$0008: CUST_SHATTER_RANGE

Used with the Customize=

Syntax: Customize=CUST_SHATTER_RANGE, FirstStaticAsShatter, LastStaticAsShatter

By default there are only 10 statics with shatter attributes (from SHATTER0 to SHATTER9) but using this custom setting the number of shatter statics can be enlarged.

Default values are: FirstStaticAsShatter = 50 (SHATTER0)
LastStaticAsShatter = 59 (SHATTER9)

Look at the "Static list" in the Reference panel to see the name (and Ids) of all the static meshes.

For example: To have another 10 static meshes as shatter type this command:

Customize=CUST_SHATTER_RANGE, 50, 69

The 10 following statics (from EXTRA00 to EXTRA09) will become shatter objects.

If you do not want to change EXTRA to statics and prefer to use standard statics to shatter type the following command:

Customize=CUST_SHATTER_RANGE, 40, 59

All statics from **DEBRIS0** to **DEBRIS9** will become shatters.

Use the following short list as a reference for the above examples:

40 \$0029 DEBRIS0	50 \$0032: SHATTER0	60 \$003C: EXTRA00
41 \$0029: DEBRIS1	51 \$0033: SHATTER1	61 \$003D: EXTRA01
42 \$002A: DEBRIS2	52 \$0034: SHATTER2	62 \$003E: EXTRA02
43 \$002B: DEBRIS3	53 \$0035: SHATTER3	63 \$003F: EXTRA03
44 \$002C: DEBRIS4	54 \$0036: SHATTER4	64 \$0040: EXTRA04
45 \$002D: DEBRIS5	55 \$0037: SHATTER5	65 \$0041: EXTRA05
46 \$002E: DEBRIS6	56 \$0038: SHATTER6	66 \$0042: EXTRA06
47 \$002F: DEBRIS7	57 \$0039: SHATTER7	67 \$0043: EXTRA07
48 \$0030: DEBRIS8	58 \$003A: SHATTER8	68 \$0044: EXTRA08
49 \$0031: DEBRIS9	59 \$003B: SHATTER9	69 \$0045: EXTRA09

CUST_constants

54 \$0036: CUST_SHATTER_SPECIFIC

Used with the Customize= command.

Syntax: Customize=CUST_SHATTER_SPECIFIC, Slot1,Slot2, Slot3, SlotN

In this customize type the moveables that will be able to destroy the shatter objects with the OCB value 8192.

Only a moveable in this list will be able to destroy the statics with 8192 OCB

Notes:

Up to 250 moveable slots can be typed.

Only one **CUST_SHATTER_SPECIFIC** for level section.

To control an explosion remember to use the GRENADE (365) shot by the SAS.
For ammo type (see the following list) for explosive ammos fired by Lara.

To set a specific weapon that is able to destroy the given static, not a moveable (like enemy), type one of the slots for the weapon ammos:

PISTOLS_AMMO_ITEM

UZI_AMMO_ITEM

SHOTGUN_AMMO1_ITEM

SHOTGUN_AMMO2_ITEM

CROSSBOW_AMMO1_ITEM

CROSSBOW_AMMO2_ITEM

CROSSBOW_AMMO3_ITEM

GRENADE_GUN_AMMO1_ITEM

GRENADE_GUN_AMMO2_ITEM

GRENADE_GUN_AMMO3_ITEM

SIXSHOOTER_AMMO_ITEM

CUST_constants

11 \$000B: CUST_SHOW_AMMO_COUNTER

Used with the Customize=

Use this value to show the counter for the current ammo.

Syntax: Customize=CUST_SHOW_AMMO_COUNTER, Color, FormatFlags (**FT_**),
BlinkTime, SizeCharacter (**SC_**), ShowCounterFlags (**SHOWC_**)

If the CUST_SHOW_AMMO_COUNTER is set, there will be a counter for the ammo for the current weapon on the screen shown in real-time.

Remark: The ammo counter will only show when Lara holds a weapon.

Color, FormatFlags (FT_**), BlinkTime, SizeCharacter (**SC_**)**

All the fields have the same use and target of fields in the TextFormat= script command.

So read the description of the above field in **NEW SCRIPT COMMANDS**.

The fields set colour, size and position of the text for the ammo.

ShowCounterFlags (SHOWC_**)**

Set some **SHOWC_** constant flags to customize the showing of the ammo counter.

See the **SHOWC_** constants

CUST_constants

52 \$0034: CUST_SLOT_FLAGS

Used with the **Customize=** command.

Syntax: Customize=CUST_SLOT_FLAGS, Slot, FlagsForSlot (**FFS_**...)

This customize supplies a generic way to customize some feature for a given slot.

Note: To customize two or more slots type two or more
Customize=CUST_SLOT_FLAGS commands, one for each slot to customize.

Slot Field

Type the number or name of the slot to assign a **FFS_** flag.

FlagsForSlot (**FFS_**)

One or more **FFS_** flags can be added and linked with plus "+" signs.

Do not use two different Customize commands for the same Slot, this operation could confuse the **TRNG** engine.

CUST_constants

7 \$0007: CUST_SPEED_MOVING

Used with the Customize=

Syntax: Customize=CUST_SPEED_MOVING, Speed

When an **ACTION trigger** is used to move an animating or other moveables, the **TRNG** engine uses a **default speed value of 32**.

To use a different speed type in the OCB of the object.

This method only works for Animating objects because other moveables have (or could have) some use for the OCB field.

The **TRNG** engine will ignore the OCB field of the object that is not an Animating

To modify the default speed used by all moveables other than the Animating use the **CUST_SPEED_MOVING** constant.

For example: Customize=CUST_SPEED_MOVING, 50
will set 50 as the default speed.

CUST_constants

1 \$0001: CUST_STAR_WARS_ROBOT

Used with the customize command.

Syntax: Customize=CUST_STAR_WARS_ROBOT, RobotIndex, Flags (**SWR_...**),
ExtraParameter, TriggerGroupIdOnSupervisory

RobotIndex

The index of the **Star Wars Robot** customizing with the script command.
In the level there could be more SW Robots it is necessary to specify the index of any
Customize=CUST_STAR_WARS_ROBOT commands.

Have a Customize for each robot in the level, all SW Robots with no specific customize,
will have same settings of this customize command.

Flags (**SWR_**)

Type one or more **SWR_** flags linked with + operator, to set different skills of a robot.
See **SWR_** constants for information.

ExtraParameter

The meaning will depend on the **SWR_** flag used.
See the **SRW_** flags.

TriggerGroupIdOnSupervisory

Type the **id of a TriggerGroup** to perform when the robot detects Lara.

CUST_constants

14 \$000E: CUST_STATIC_TRANSPARENCY

Used with the Customize=

Change the value for transparency of a static that can be assigned to a static with the new flip effect trigger "Static. Transparency. .."

Syntax: Customize=CUST_STATIC_TRANSPARENCY, GlassOpacity, IceOpacity

To set the transparency set the value of opacity.

The values set for **GlassOpacity** and or **IceOpacity** are a percentage from 0 to 100

where: 0 = the object is fully invisible
100 = No transparency for the object

Using 50 % the object and background should be semi-transparent

Using 90 % the object is almost Normal (full opacity) but there is a light transparency

Using 10 % the object is almost invisible, it is seen like a ghost.

The default values used if there is no Customize=CUST_STATIC_TRANSPARENCY command

are: **GlassOpacity** = 50
IceOpacity = 81

22 \$0016: CUST_TEXT_ON_FLY_SCREEN

Used with the Customize=

Enables the printing of text on the screen during the Flyby Camera sequence.

Syntax: Customize=CUST_TEXT_ON_FLY_SCREEN, ENABLED/DISABLED

CUST_constants

46 \$002E: CUST_TITLE_FMV

Used with the Customize=

Used with the Customize command.

Syntax: Customize= CUST_TITLE_FMV, FmvNumber, TestMultiPlay

FmvNumber

Type the number of the FMV to play.

This is the same number in the file name of the FMV.

For example: To play the file fmv5.wmv movie, type 5.

TestMultiPlay

This field may be 0 (FALSE) or 1 (TRUE).

Type 0, the FMV will only be shown the first time the player launches the game.

Type 1, the FMV will be played every time the control passes to the title phase.

i.e., at the start of the game and when Lara dies

or the player exits from the game to go back to the title.

Example: To have a FMV at the boot-strap of the game, i.e. before beginning the title level, place in the [title] section the following commands: FMV= 8, 1

Customize=CUST_TITLE_FMV, 8, 0

In the above example the FMV to show at the boot is fmv8,

Enable the Escape for this FMV and it is played only once at the start of the game.

Remember to place the FMV file in the sub-folder named "FMV s" or "Store" and set in the **script.txt** in the [PCExtensions] section the extension used for the FMV.

For : "fmv8.wmv" type the line: FMV= WMV
"fmv8.avi" type the line: FMV= AVI
"fmv8.mpg" type the line: FMV= MPG

See the **NEW SCRIPT COMMANDS**.

CUST_constants

38 \$0026: CUST_TR5_UNDERWATER_COLLISIONS

Used with the Customize=

Used to customize underwater collisions.

Syntax: Customize=CUST_TR5_UNDERWATER_COLLISIONS

Adding this customize in the level enables the collision method used in **Tomb Raider Chronicles** for the underwater collisions of Lara.

The difference from the **Tomb4** collision is that in the **Tomb5** engine Lara will not be stopped when she touches a wall, but will move or rotate going on in a different direction.

In **Tomb4** any contact will stop Lara.

43 \$002B: CUST_WATERFALL_SPEED

Used with the Customize=

Used to customize the waterfall speed.

Syntax: Customize=CUST_WATERFALL_SPEED, PixelScroll

This changes the speed of the waterfall textures.

The default value is -7, set higher absolute values like -12 and the speed will be increased.

Use a positive value and the scrolling will be inverted (from down to up).

The maximum values are -63 to +63 but the meaningful maximum values are -31 to +31

CUST_constants

9 \$0009: CUST_WEAPON

Used with the Customize=

Customize a specific weapon.

Syntax: Customize= CUST_WEAPON, SlotOfWeapon, Weapon flags (**WEAP_**),
SoundForShot, FramesForRecharge, DurateFlash, Extra,
MaxDistanceForAiming, FrameToTakeWeapon,
FrameToLetWeapon, Random, VPositionOfWeapon,
Unknown, FrameCounter, FrameMinRange,
FrameMaxRange, OrigX, OrigY, OrigZ, OrigOrient

Remarks: Omit the last fields of this command and all omitted fields will be seen as **IGNORE** by the **TRNG** engine, i.e. the omitted fields will keep the default values.

This feature is present in all Customize commands from **1.1.8.1 dll version**.

Customize some features of the weapon using the **CUST_** constant named **CUST_AMMO**.

The current **CUST_WEAPON** only works for specific weapons.

The **CUST_AMMO** is used to customize single ammos.

Description of fields

SlotOfWeapon

Set the slot corresponding to the weapon to customize.

Choose between following slot values: **PISTOLS_ITEM**
UZI_ITEM
SHOTGUN_ITEM
CROSSBOW_ITEM
GRENADE_GUN_ITEM
SIXSHOOTER_ITEM

Weapon flags (**WEAP_**)

Add none, one or more **WEAP_** flags to set a special features for the current weapon.

Type **IGNORE** for no flag.

See the **WEAP_** constants

CUST_constants

SoundForShot

This field changes the default sound effect for the weapon.

This is the sound used when the weapon fires and not the sound when the shot hits some target.

Type **IGNORE** to use the default value.

Default values:	PISTOLS_ITEM	=	8
	UZI_ITEM	=	43
	SHOTGUN_ITEM	=	45
	CROSSBOW_ITEM	=	235
	SIXSHOOTER_ITEM	=	121
	GRENADE_GUN_ITEM		

(sounds are in the shooting animation with play flip anim command)

FramesForRecharge

This is the number of frames (30 per second) used for the auto-shot.

When a player continuously presses the action key the **FramesForRecharge** is used to set the pause between shots.

Bigger values will cause a longer pause time and hence less shots per second.

Default values:	PISTOLS_ITEM	=	9
	UZI_ITEM	=	3
	SHOTGUN_ITEM	=	9
	SIXSHOOTER_ITEM	=	16
	CROSSBOW_ITEM		Fixed values
	GRENADE_GUN_ITEM		Fixed value.

DurateFlash

This sets the number of frames (30 per second) of duration for fire flash when the weapon shots.

Default values:	PISTOLS_ITEM	=	3
	UZI_ITEM	=	3
	SHOTGUN_ITEM	=	3
	SIXSHOOTER_ITEM	=	3
	GRENADE_GUN_ITEM	=	2
	CROSSBOW_ITEM	=	2

Extra

This field can have different values according to the **WEAP_** constant flag.

Read the descriptions of the **WEAP_** constants for the value to type in this field.

CUST_constants

MaxDistanceForAiming

The automatic aiming allows Lara to point at an enemy when he is (about) in front of her and within a specific maximum aiming distance

The **default value for all weapons is 8 sectors**.

Different aiming distances can be set for each weapon.

If zero is set the weapon will not have automatic aiming at enemies.

If 3 (sectors) are set the weapon will work with automatic aiming on enemies within 3 sectors of 3d distance.

Remark: Increasing this value could compromise the speed of the game engine because the compute to locate enemies will be enlarged to very wide zones wasting CPU time.

FrameToTakeWeapon and FrameToLetWeapon

These two fields set the number of frames to get on and get off in Lara's hands animation. Only change these values when trying to change the animation to extract weapons.

For pistols, UZI and six shooter set the same frame for both fields and the **TRNG** engine will use that frame -1 to place the weapon in the holster and the frame +1 to place the weapon in Lara's hands.

The reason for this method is the **TRNG** engine will use the same animation to extract the weapon or place it in holsters. Only the frame will change +1 or -1 according to the extract or holster animation.

Big weapons like the shotgun, grenade-gun and crossbow have two different frames for the extract or holder operation.

The following table shows the default values:

	PlaceWeaponInHand	RemoveWeaponFromHand
Pistols	13 (it becomes 14)	13 (it becomes 12)
Revolver	15 (it becomes 16)	15 (it becomes 14)
UZI	13 (it becomes 14)	13 (it becomes 12)
Shotgun	10	21
Grenade Gun	10	21
Cross Bow	10	21

Remark: Looking at the original animations to extract weapons the number of frames are bigger than the total number of frames of the animation.
It is a hard-coded value.

CUST_constants

Random

Used to randomly change the direction or intensity for some shooting effect.

Default values:	Pistols	=	1456
	Revolver	=	728
	UZI	=	1456
	Shotgun	=	0
	Grenade Gun	=	1456
	Cross Bow	=	1456

VPositionOfWeapon

This field is a Coordinate Y value (up/down in the 3d world) to compute the height of the weapon in Lara's hands.

When the **TRNG** engine detects that water is touching the weapon it will holster the weapon.

Default values :	Pistols	=	650
	Revolver	=	650
	UZI	=	650
	Shotgun	=	500
	Grenade Gun	=	500
	Cross Bow	=	500

Unknown

This is not used when Lara extracts the weapon and neither when she is shooting. For all weapons the default value is 1820.

FrameCounter

This is the number of frames and it is decreased by 1 or each game cycle.

It is a "frame rate" field to multiply the Duration of the animation to slow down the extract weapon animation.

This field is only used for the Pistol, UZI and revolver.

Default values:	Pistols	=	4
	Revolver	=	7
	UZI	=	4

CUST_constants

FrameMinRange and FrameMaxRange

These two values are the minimum and maximum values of a range.
They are only used for Pistols, Revolver and UZI.

If the frame to extract the weapon is changed

(see the **FrameToTakeWeapon** and **FrameToLetWeapon** fields)

remember that the number to change the weapon has to be within the range of **FrameMinRange** and **FrameMaxRange** fields, otherwise the animation will not change the weapon in the correct way.

Default values:	Pistols	=	minimum: 5	maximum: 24
	Revolver	=	minimum: 8	maximum: 29
	UZI	=	minimum: 5	maximum: 24

OrigX, OrigY, OrigZ, OrigOrient

These fields only work for weapons launching visible ammo like
the **Grenade Gun** and **Cross Bow**.

If the animation or the shape of the weapon meshes is changed it may be required to change the origin of the bullets at the start of the movement.

The **OrigX**, **OrigY** and **OrigZ** are + or - values and they will be added to the default current position for the bullet.

To understand how to set these three values read the description of the **AddEffect** script command about the **DispX**, **DispY**, and **DispZ** field.

The **OrigOrient** field sets the degrees for the direction of the bullet.

Set 0 and the direction will not change from the default value.

Set a positive or negative value and the horizontal direction will be changed.

The way to set degrees is: **90 degrees to East will be 16384 (\$4000)**
 90 degrees to West will be -16384 (\$C000)

Intermediate values can be set.

Study the above to understand how to compute other values.

DEMF_constants

256 \$0100: DEMF_CINEMA_SCREEN

Used in the Demo script command.

This flag adds the cinema screen effect while the demo is playing.

For cinema effect we mean when two black bars, at the top and bottom of the screen give a wide screen effect. This is like the effect seen in the flyby sequence but in this case the black bars are higher to cover the **hp bar** to distinguish demo cutscenes from the flyby camera.

Note: A problem with the cinema screen is that the top and bottom legend demo will be covered by the black bars. If the cinema screen is enabled use a vertical centre alignment for the demo legend or disable the demo legend text.

8 \$0008: DEMF_CROSS_FADE

Used in the Demo script command.

By default the first demo will begin with a fade-in (from black to game screen).

If **DEMF_PLAY_LEVEL_SEQUENCE** or **DEMF_PLAY_ALL_SEQUENCE** is used add the cross fade to swap the two demos while the game screen is black.

If this flag is omitted in a demo sequence Lara will disappear from the last position of the first demo and she will suddenly appear in the new room and position at the start point of the second demo. The disadvantage of cross fade is that the last frames of the first demo and the first frames of the following demo will be darked.

1024 \$0400: DEMF_MUTE_SFX

Used in the Demo script command.

This flag silences the sound effects but plays the background music from the Audio folder.

This solution is good to give the demos a more understated layout avoiding the shoot or jump sounds but allowing soft background music (audio track). In this situation it is interesting playing a different audio track in the demo mode to entertain the player.

Note: To silence the sound effects the **TRNG** engine will temporarily change the sfx sound volume to zero. Oddly with this null volume the strongest sfx sounds can still be heard in a quiet way.

512 \$0200: DEMF_MUTE_TRACK

Used in the Demo script command.

This silences the audio tracks while the demo is playing with the flip effect.

This flag does not affect the sound effects.

DEMF_constants

64 \$0040: DEMF_PERFORM_AT_START

Used in the Demo script command.

Used in the [Level] section, (cutscene mode).

For a Demo command in a [Level] section add the **DEMF_PERFORM_AT_START** flag to force the **TRNG** to play the demo at the start of the level like a introduction cutscene.

Note: If the **DEMF_PLAY_LEVEL_SEQUENCE** flag is present all demos cutscenes will be played in a sequence with no chance for the player to interact with the game until the last demo has been played.

2 \$0002: DEMF_PLAY_ALL_SEQUENCE

Used in the Demo script command.

This silences the audio tracks while the demo is playing with the flip effect. This flag does not affect the sound effects.

1 \$0001: DEMF_PLAY_LEVEL_SEQUENCE

Used in the Demo script command.

Used in the [Title] section.

This allows a demo array to be played. If the following demo is the same level, that will be performed and it will go on for all of the demos with the same level number.

The advantage using this flag is to avoid the "load-new-level" phase followed by the "come-back-to-title-level" phase. All single demos of the level will be played in sequence when the level has already been loaded.

DEMF_constants

32 \$0020: DEMF_PLAY_ON_KEY

Used in Demo script command.

Used in [Title] section.

Add the **DEMF_PLAY_ON_KEY** flag to enable the player to choose and launch different demos. Type a text information string to explain to the player what keys he can press.

Note: The keys are digits from 1 to 0, where 0 is for 10.
If demos are in three levels of your adventure use a text information string like this: "Hit "1", "2", or "3" to show demos."

The number of the key is not the position of the Demo Id in array demo fields, or directly the Demo ID. It is the level number when a demo is played.

If there is more than one demo for the chosen level the choice will depend on the **DEMF_RANDOM** or **DEMF_PLAY_LEVEL_SEQUENCE** flags.

As a general rule, **TRNG** will avoid playing a demo twice and it will give precedence to demos not played.

4 \$0004: DEMF_QUIT_WITH_ESCAPE

Used in the Demo script command.

To allow the player to quit a demo while it is playing add the **DEMF_QUIT_WITH_ESCAPE** flag to the Demo Flags.

When the player presses Escape the current playing will be stopped and the control will come back to the title level if the demo was in [Title] section, or back to the common game if the demo was in the [Level] section.

16 \$0010: DEMF_RANDOM

Used in the Demo script command.

Used in the [Title] section.

For demos (supposing you had more than a demo binary file) to have no fixed order to be played add the **DEMF_RANDOM** flag.

In this situation there is no relevance for the order of the Demo IDs in the array demo fields. The order is changed randomly every time the game is launched.

Note: The **DEMF_RANDOM** flag does not work with **DEMF_PLAY_LEVEL_SEQUENCE** and **DEMF_PLAY_ALL_SEQUENCE** because the same level number set in the array is changed randomly.

DEMO_constants

2048 \$0800: DEMO_FRAME

Used with the [SPC_PAUSE](#) command of Parameters=PARAM_ACTOR_SPEECH

This is a special flag that can only be used by adding it to the [SPC_PAUSE](#) command to force an absolute pause until the current demo reaches the given frame.

See the [SPC_PAUSE](#) command for more information.

DENV_constants

16384 \$4000: DENV_FLAG_FACE2BACK

Works in the same way as the **DENV_FLAG_FACE2FACE** flag but in this case the condition is TRUE when Lara is behind the object.

See the description of the **DENV_FLAG_FACE2FACE** flag for more information.

32768 \$8000: DENV_FLAG_FACE2FACE

Works with the **ENV_ITEM_EXTRA...** conditions, but adds this flag to the **DistanceForEnv** field and not to the **EnvCondition** field.

Use it to acquire a specific couple of Orientation (facing) for Lara and the object.

For example: To start the animation only when Lara is face to face with the SKELETON (slot id = 35) and about a half sector (512) distance from it type:

In the ENV Condition field	:	ENV_ITEM_EXTRA_IN_FRONT
In the DistanceForENV	:	512 +DENV_FLAG_FACE2FACE
In the Extra field	:	35

Remark: The **DENV_constants** are **DistanceENV** flags to add to the distance value to set some other settings.

For the **ENV_ITEM_EXTRA** conditions add two **DENV_** constants to the **DistanceforENV** to set the position for the orientation of Lara and the object.

Choose from two DENV_constants: **DENV_FLAG_FACE2FACE**
Lara and object are looking face to face

DENV_FLAG_FACE2BACK
Lara is looking at the back of the object

Remark: If no **DENV_** constant is added to the distance the facing will be **IGNORED**.

DGX_constants

8192 \$2000: DGX_ADJUSTMENT_MODE

Used in the **DiagnosticType=** command.

This allows the alignment of **TRNG** objects like the **Detector** and the **Keypad** to have the correct coupling between the object and the text.

In some circumstances when the game screen has a high resolution it could give some misplacement between the object and the text or the radar detector and the sprites of detected targets.

Use this adjustment when some misalignment is discovered with a higher resolution except those already adjusted.

It is better to enable in the adjusting phase the **DGX_ADJUSTMENT_MODE**, so type the following command in the [Option] section of the **script file**:

```
Diagnostic = ENABLED  
DiagnosticType= DGX_ADJUSTMENT_MODE, 0
```

In the game many variables will be printed on the screen about the position and text formatting of **TRNG** objects.

Keyboard commands in adjustment mode

To choose the variable to change use the: **R** and **F** keys

To change the value of the current selected variable use the keys: **Y** and **U** keys

Another keyboard command in the adjustment mode is the: **S** key.

If the **TOMB4_log** utility is launched, when **S** is pressed all current variables and values will be printed in a log file.

Always perform this operation at the end of the adjustment phase.

To locate the correct data in the log file for "ADJUSTMENT DATA" text, copy and paste all text between the "adjustment data" and "end adjustment" .

For example:

```
6985: ----- ADJUSTMENT DATA (640 x 480) -----
6985: RADAR DETECTOR:
6985: TargetX=1587
6985: TargetY=13669
6985: SizeTextX=128.00
6985: SizeTextY=170.67
6985: VLineX=2624
6985: VLineY=13715
6985: GapX=112
6985: GapY=149.33
6985: TextX=736
6985: TextY=15541
6985: COMPASS DETECTOR:
6985: VLineY=13845
6985: KEYPAD:
6985: OrgX=11056
6985: OrgY=17685
6985: TextY=2304
6985: ----- END ADJUSTMENT -----
```

Remarks: The adjustment mode only allows a temporary fix for the problem.
At the next game restart the values will come back to the original (bad) values
The adjustment should be done with the final target to send the correct adjusted value to [PAOLONE] and I will insert the new best values in the next version of the **trng dll**

If you have a computer with a higher resolution you can create adjusted values for these extra resolutions.

In the radar detector section many values can be changed but it is advisable to change only three values:

TargetY	Sets the Y position of detected objects on the main (planar) panel
VLineY	Sets the Y position of detected objects on the secondary (vertical) panel
TextY	Sets the Y position of text (distance xx metres)

DGX_constants

2 \$0002: DGX_ANIMATION

Used in the DiagnosticType= command.

Shows the debug data about the Animation script command.

The Animation command works in the Debug mode with a negative animation number.

In this situation the diagnostic will show if the conditions to enable the animation are correct and when this is true it will show "YES".

16 \$0010: DGX_AUDIO_TRACKS

Used in the DiagnosticType= command.

It enables the viewing of information for audio tracks:

The enabled channel

The number of the CD track playing.

32 \$0020: DGX_CHEATS

Used in the DiagnosticType= command.

Shows the current typed keystrokes to set some **TRNG** cheats.

The cheats allowed in **TRNG** are all based on four characters words:

KILL	kill all enemies
ROOM	reverse all flip ROOM s
IAIR	Infinite AIR
GODS	set Lara as invulnerable i.e. she becomes a DEMIGOD
DOOR	open all DOORS of the level
STAR	gives Lara a constant and she glows like a STAR

Remark: The cheats work every time the Diagnostic mode is enabled
Diagnostic=ENABLED and the FlyCheat=ENABLED.

128 \$0080: DGX_COMMON_VARIABLES

Used in the DiagnosticType= command.

Shows on screen the values of the **TRNG** variables.

This flag enables the viewing for common variables,
i.e. all variables except the Store variables.

DGX_constants

32768 \$8000: DGX_ERRORS

Used in the **DiagnosticType=** command.

By default the **TRNG** engine is able to give many error messages when it detects some problem with script commands, wrong indices of moveables (in some triggers) and other DirectX errors.

Unfortunately most level designers ignore them because to see these messages they should launch the **tomb4_log.exe** program.

At the end of the game see the log that has been caught from the logger.

There is an easier way to have the error messages shown on the game display in real time. Add the **DGX_ERRORS** flag to the **DiagnosticType=** command.

Note: Since in the game the errors could be many to cover all of the screen, **TRNG** uses a buffering method: It keeps on the screen an error message for only five seconds and ignores it when a new error message is the same as one already printed on the screen.

At the end of the game find a text file named "**error_parsing_log.txt**" with all of the error messages.

If at the end of playing there is no "**error_parsing_log.txt**" file this means that there has been No Errors.

Remark: Some error messages could be very long.
To keep them all on the screen use a small font.
In the **error_parsing_log** file all of the errors will be recorded with no truncation.

8 \$0008: DGX_FAR_VIEW

Used in the **DiagnosticType=** command.

This flag enables the viewing of the current **WorldFarView** distance applied in the game.

Remark: This information requires a Turbo script command in the current level using the **AdaptiveFarView**, otherwise it will not be shown.

512 \$0200: DGX_FLYBY

Used in the **DiagnosticType=** command.

Enables the information for the current Fly By sequence.

Remark: The information is shown only when a Fly By sequence is running.

DGX_constants

64 \$0040: DGX_FOG

Used in the DiagnosticType= command.

Shows the information for the Fog Distance: **StartFog**, **EndFog** and **ColorFog**.

Remark: To see the information set the **DGX_FOG** flag it is also necessary that in the current [Level] section there is a FogRange= command.

16384 \$4000: DGX_FPS

Used in the DiagnosticType= command.

Shows the current FPS (frames per second).

Note: It will show a couple of values: **the first is the async frame rate**
the second is the sync frame rate.

1 \$0001: DGX_LARA

Used in the DiagnosticType= command.

Shows in the Diagnostic the information about Lara: coordinates, orientation, room, flags

DGX_constants

2048 \$0800: DGX_LOG_SCRIPT_COMMANDS

Used in the **DiagnosticType=** command.

This Diagnostic mode works in a different way with respect to other **DGX_** modes.

It does not show anything on the screen but it creates a full log using the **TOMB4_LOG.exe** utility.

This Diagnostic mode enables a Debug mode for the **Global Trigger, Organizer** and **Trigger Group** commands showing all further variables changed by the above commands

It is advisable to run Tomb Raider in windowed mode to have a chance to see the **tomb4_log** window with the latest messages.

Consult the full log at the end of **tomb4** running using the command in the **tomb4_log** program [Show/Last tomb.log]

Tips & Tricks:

Add in the **Extra** field of the **DiagnosticType** command the constant **EDGX_CONCISE_SCRIPT_LOG** to reduce the quantity of messages in the log.

See the description of the **EDGX_CONCISE_SCRIPT_LOG** constant for more information.

Use the function Key **F9** to suspend/resume the log about script commands.
Use this feature to temporary suspend the log when Lara is far from the "zone" you want to check and then resume the log when ready.

This trick is useful to reduce the size of the log file and get the correct portion of log you are interested in.

The debug mode of script could slow down the game.

Remember to disable it to play normally.

DGX_constants

4 \$0004: DGX_SFX_SOUNDS

Used in the DiagnosticType= command.

This enables the viewing on screen of the current sound sample played and information about the missing sound samples.

256 \$0100: DGX_STORE_VARIABLES

Used in the DiagnosticType= command.

Shows on screen the values of all the Store variables.

1024 \$0400: DGX_TEXT_VARIABLES

Used in the DiagnosticType= command.

Shows on screen the content of test variables.

4096 \$1000: DGX_WEAPON_ANIMATION

Used in the DiagnosticType= command.

Adding this flag will display on screen the information about the current weapon animation.

The animations and state Id for weapon management are not visible in Lara's information because the weapon works on specific slots like the **SHOTGUN_ANIM**, **GRENADE_GUN_ANIM** etc.

DIR_constants

27 \$001B: DIR_DIRECTION_LARA_LEADING_ACTOR

Used in the Parameters=PARAM_MOVE_ITEM script command.

The direction will be the line from Lara to the leading actor.

Understand that this is different from the **DIR_LARA_FACING** since the

DIR_DIRECTION_LARA_LEADING_ACTOR flag has no importance where Lara is looking.

Lara and the leading actor will be used as two points to draw the direction.

Note: Remember to add the **DIR_INVERT_DIRECTION** flag to have the opposite direction. In this case the direction is from the leading actor towards Lara.

17 \$0011: DIR_DOWN

Used in the Parameters=PARAM_MOVE_ITEM command.

Absolute moving down.

1 \$0001: DIR_EAST

Used in the Parameters= PARAM_MOVE_ITEM command.

Absolute moving to east.

18 \$0012: DIR_FORWARD

Used in the Parameters=PARAM_MOVE_ITEM script command.

Relative direction.

This flag will move the item forward with respect to its current facing.

For instance if the item is looking north the item will move to north.

DIR_constants

30 \$001E: DIR_HEAD_FOR_EXTRA_ACTOR

Used in the Parameters=PARAM_MOVE_ITEM script command.

Direction from the current position of the item to move towards the position of extra actor.

28 \$001C: DIR_HEAD_FOR_LARA

Used in the Parameters=PARAM_MOVE_ITEM script command.

This direction is given from the current position of the item to be moved and the position of Lara as the ending position.

29 \$001D: DIR_HEAD_FOR_LEADING_ACTOR

Used in the Parameters=PARAM_MOVE_ITEM script command.

Direction from the current position of the item to move and the position of the leading actor.

DIR_constants

256 \$0100: DIR_INVERT_DIRECTION

Used in the Parameters=PARAM_MOVE_ITEM script command.

This is a flag that is added to any **DIR_** direction value.

When this flag is added it gives a relative direction with turning.

For instance the pair: **DIR_TURNING_RIGHT_90+DIR_INVERT_DIRECTION**

will move the item turning it right.

25 \$0019: DIR_LARA_FACING

Used in the Parameters=PARAM_MOVE_ITEM script command.

The direction will be the same where Lara is looking.

This direction could be useful when Lara throws away an item (or bullet) in front of her.

26 \$001A: DIR_LEADING_ACTOR_FACING

Used in the Parameters=PARAM_MOVE_ITEM script command.

This direction will be the same as where the leading actor is looking.

This maybe useful when the leading actor throws an item in front of him.

Note: Remember to set an enemy as the leading actor with the Action 87.

21 \$0015: DIR_LU_TURNING_180

Used in the Parameters=PARAM_MOVE_ITEM script command.

Relative direction with turning.

This flag will move the item forward continuously changing its facing, turning left, until it is facing the opposite direction from its start.

See the description of **DIR_TURNING_LEFT_90**

0 \$0000: DIR_NORTH

Used in the Parameters= PARAM_MOVE_ITEM command.

Absolute moving to north.

DIR_constants

24 \$0018: DIR_RU_TURNING_180

Used in the Parameters=PARAM_MOVE_ITEM script command.

Relative direction with turning.

This flag will move the item forward continuously changing its facing, turning right, until it is facing the opposite direction from its start.

2 \$0002: DIR_SOUTH

Used in the Parameters= PARAM_MOVE_ITEM command.

Absolute moving to south.

20 \$0014: DIR_TURNING_LEFT_45

Used in the Parameters=PARAM_MOVE_ITEM script command.

Relative direction with turning.

This flag works like **DIR_TURNING_LEFT_90** flag but in this case the turning will only be 45 degrees.

19 \$0013: DIR_TURNING_LEFT_90

Used in the Parameters=PARAM_MOVE_ITEM script command.

Relative direction with turning.

Setting this flag the item will move forward but its facing will be changed smoothly until it is turned by 90 degrees with respect to its forward movement.

With this setting an item can move to circumvent a corner of 90 degrees on the left.

Set in the **Extra** field the turning speed typing a value that will be added for each frame to the current item facing.

The units work in this way:

45 degrees = \$2000

90 degrees = \$4000

Any intermediate value can be set. It is better to set a small value since the value typed in the **Extra** field will be added to the current facing 30 times a second.

Reasonable values are in the range 32 to 1024

Note: Only set positive values. The direction of turning will be set by the **DIR_** value direction. When a left turning is set the turning speed will be subtracted from the current facing of the item. Only type positive values.

DIR_constants

22 \$0016: DIR_TURNING_RIGHT_45

Used in the Parameters=PARAM_MOVE_ITEM script command.

Relative direction with turning.

This flag moves the item forwards changing its facing until it is turned right by 45 degrees with respect to its original facing.

23 \$0017: DIR_TURNING_RIGHT_90

Used in the Parameters=PARAM_MOVE_ITEM script command.

Relative direction with turning.

This flag moves the item forwards changing its facing until it is turned right by 90 degrees with respect to its original facing.

16 \$0010: DIR_UP

Used in the Parameters= PARAM_MOVE_ITEM command.

Absolute moving up.

3 \$0003: DIR_WEST

Used in the Parameters= PARAM_MOVE_ITEM command.

Absolute moving to west.

DISABLED_constants

0 \$0000: DISABLED

Disabled.

DMG_constants

256 \$0100: DMG_ALERT_BEEP

Used with the Damage=

Only use with the **DMG_INDIRECT_BAR** flag.

When the bar goes down to 15 % (or less) it will blink and a "beep" sound will be performed.

Remark: When this flag is absent the bar will blink but no sound will be performed.

32 \$0020: DMG_BURNING_DEATH

Used with the Damage=

When the time for the Damage Bar is complete Lara is killed burning with fire.

Only use for waterless Damage rooms.

This is used to simulate a room where the temperature increases until Lara burns.

Lara will burn when: **The indirect Bar is empty**
The Health Points bar is empty
The indirect Bar for the Damage room is not enabled

128 \$0080: DMG_BURNING_SCREAM

Used with the Damage=

Only use with the **DMG_BURNING_DEATH** flag.

With the **DMG_BURNING_DEATH** flag set Lara is forced to scream when she takes fire.

Remark: This flag only works with the indirect Bar.
When the indirect bar is not set,
Lara burns and dies and has no time to scream.

8 \$0008: DMG_COLD_WATER

Used with the Damage=

Set this to customize the Cold Water Rooms in the current level.

If this flag is not used the setting will effect the Damage Rooms.

DMG_constants

4 \$0004: DMG_INCREASE_BAR

Used with the Damage=

Only use with the **DMG_INDIRECT_BAR** flag.
By default the Damage Bar starts full and decreases with time.
When it reaches the empty status Lara will be damaged faster.
Use this flag to force the Damage Bar to start empty and increase over time.
This flag simulates an increase of temperature in the current room.

01 \$0001: DMG_INDIRECT_BAR

Used with the Damage=

Use this flag to have a progress bar on the screen to signal the level of the current damage in the current room.

With the indirect bar the vitality of Lara remains untouched until the indirect bar is empty (or full if the **DMG_INCREASE_BAR** flag is set).

When set the Damage works in this way:

The Damage Bar decrease but not until it is empty is there any damage for Lara.
When the Damage Bar is empty the Health Points of Lara start to decrease.

If this flag is not set the Damage will directly affect the Health Points Bar of Lara.

512 \$0200: DMG_LITTLE_TEXT

Used with the Damage=

When a **BarName** is used in the Damage= command choose the flag to use small text to show the string with the Bar Name.

2 \$0002: DMG_ONLY_PAD

Used with the Damage=

Restricts the Damage time when Lara is touching the floor of the current Damage Room.
This flag is **IGNORED** in Cold Water Rooms.
Use this for simulation of an electric floor or a very hot floor.

DMG_constants

16 \$0010: DMG_POISON_LARA

Used with the Damage=

Lara enters the current room and she will be poisoned with a deforming effect of the screen and a life bar with a yellow colour.

64 \$0040: DMG_SLOW_DISAPPEARING

Used with the Damage=

Set the behaviour of the damage bar when Lara goes from a Damage Room to a safe room.

When Lara goes from the Damage or Cold Water Room the Bar will increase slowly until it is full and then it will be removed from the screen.

If it is omitted the Damage Bar will disappear immediately, not when Lara exits from the Damage Room.

Only use with the **DMG_INDIRECT_BAR** flag.

The Damage Bar will invert its direction of filling or emptying and it will disappear when it reaches the full state. Like at the start of entry in the Damage Room.

DRT_constants

2 \$0002: DRT_ADD_EFFECT

Used in the Customize=CUST_DARTS command.

Adds an effect for each dart.

Type in the **IdAddEffect** field the Id of the **AddEffect** script command where the effect to add to the dart is stored.

Remarks: **Warning, this feature could be CPU intensive.**
If there are many dart emitters in the level disable the emitters already visited by Lara, using an Untrigger to free the resource.

Another suggestion is to set a big Emitting timer to slow down the emitting and or a high Speed value to avoid having too many darts enabled at the same time.

In the **AddEffect** command pointed to by the **IdAddEffect** field some fields will be ignored: **DurateEmit** and **DuratePause**.

Other fields should have the same value:

The **JointType** should be set to a **JOINT_SINGLE_MESH** value.

The **TRNG** engine will only be interested in the type of effect, while the emitting time will be constant for the life-time of the dart.

32 \$0020: DRT_FIX_POISON_BUG

Used in the Customize=CUST_DARTS command.

In the default **Tomb4** there was a bug about poisoning with darts. The poison of the Scorpion and Harpy worked correctly with the screen deformation and Lara losing Health Points. The poison of the dart only remained for an instant and was removed. To fix this bug add the **DRT_FIX_POISON_BUG** flag.

16 \$0010: DRT_HIDE_DART

Used in the Customize=CUST_DARTS command.

This makes the dart invisible.

The only case where this setting could be used is when a effect is added to the dart with the **DRT_ADD_EFFECT** flag and only the **Mist** or **fire effect** is to be visible.

This setting helps to free the CPU from 3d graphic operations.

DRT_constants

1 \$0001: DRT_NO_POISON

Used in the Customize=CUST_DARTS command.

By default when Lara is hit by a dart she is poisoned.

Use this to disable this feature.

8 \$0008: DRT_NO_SMOKE

Used in the Customize=CUST_DARTS command.

This removes the smoke when the dart hits the wall and also the sound of the crash.

Use this when the dart looks like a laser ray.

DRT_constants

4 \$0004: DRT_PERFORM_TRIGGERGROUP

Used in the Customize=CUST_DARTS command.

Add this to transform the dart emitter to a sensor line.
When a dart touches Lara it will perform like a trigger.

Practically it could be an electronic alarm.

In this case type in the **IdTriggerGroup** field the Id of the Trigger Group to perform.

Remarks: Reduce the speed and change the colour (for example Red or Green) of the dart emitting to look like a laser ray.

When the dart emitter is used like a sensory line it is advisable to disable the poison with the **DRT_NO_POISON** flag. It makes no sense being poisoning by a laser ray.
Also disable the smoke and crash sound with the **DRT_NO_SMOKE** flag.

When a dart touches Lara the **Trigger Group** will perform and the index of the dart emitter will be stored in the internal variable [Found item index].

Then insert in the **Trigger Group** an action trigger using the

TGROUP_USE_FOUND_ITEM_INDEX flag
to disable the dart emitter with this action trigger:

```
; Exporting: TRIGGER(44:0) for ACTION(0)
; <#
: LARA          ID:0    in sector:(8,7) of Room3
; <&
: Trigger. (Moveable) Untrigger <#>
```

```
Object with (E)Timer value
; (E) : Timer= +00
; Values to add in script command: $5000, 0, $2C
```

and the values will be added in the Trigger Group command in this way:

TriggerGroup= , \$5000 + TGROUP_USE_FOUND_ITEM_INDEX, 0, \$2C

to use the found index instead of that set in the original trigger (it was Lara)

The best way to have a realistic laser sensory line avoiding at the same time the slow down of the engine :

Disable the showing of darts with the **DRT_HIDE_DART** flag to free CPU time.
Disable the smoke and crash sounds with the **DRT_NO_SMOKE**.
Disable the poisoning with the **DRT_NO_POISON** flag.

DRT_constants

Now decrease the emitting timer to show the dart.

This is necessary to avoid Lara passing the sensory line without touching the darts.

Now the only problem is that the sensory line is invisible.

To solve this problem use a static object with no collision built like the tight-rope bar and add to it the same texture used for the waterfall to give it a pulse/flushing effect.

Then in the **Tomb Editor** place the dart emitting in a direction and height with the static sensory bar placed at the same position where the darts should be emitted to give a pulsing semi-transparent laser sensory in the level.

Each sensory line could be used like a trigger to activate enemies, doors, flip maps etc.

DTF_constants

32 \$0020: DTF_ENGAGE_ALWAYS

Used in the Detector= command.

The detector will be present on screen until the targets exit the level.

8 \$0008: DTF_ENGAGE_IN_RANGE

Used in the Detector= command.

The detector will be automatically shown when Lara is close to a target within the given range.

With the **DTF_ENGAGE_IN_RANGE** set for example 50 metres the detector will be shown when Lara is 50 metres or less from the nearest target.

When Lara is further away the detector will be hidden.

4 \$0004: DTF_ENGAGE_INVENTORY

Used in the Detector= command.

Lara has to pick up the Detector object and engage it in the inventory to select it

64 \$0040: DTF_FAST_RADAR_SCAN

Used in the Detector= command.

The pointer will turn very fast

128 \$0080: DTF_INVERSE_VPOINTER

Used in the Detector= command.

This only works in the pointer mode.

The position of the line in the vertical scale (at the right of the detector) can be inverted.

By default when this flag is not set the floating line shows where Lara is.

The fixed red pointed line at the centre shows the position of the target.

If it is not logical use this flag to invert the situation.

With this flag the floating point is the vertical position of the target, while the fixed red line is the vertical position of Lara.

0 \$0000: DTF_NONE

Used in the Detector= command.

DTF_constants

2 \$0002: DTF_RADAR_MODE

Used in the Detector= command.

This sets the Detector in Radar Mode.

If omitted the detector will work in the Pointer mode.

There are big differences between the two modes and some fields and flags will only work for a specific detector mode.

In Pointer Mode: The detector has a pointer like a compass that shows where the current target is. The Current target is the first target in the list present in the game. Pointer Mode only works on **one target** at a time:
When the first target has been picked up (or killed) the detector will start to point at the second target in the list and go on.

In Radar Mode: The detector is able to scan all targets at the same time that are in the range of the radar.

Note: Remember that the range of radar is different from the range for activation. The range of radar is given by a formula
6 * MetricScale,
where 6 is the (fixed) number of grid sectors of radar.

For example: Set 2 metres for the metric scale and the target will be shown on the radar only when it is 12 metres or less from Lara.

Radar Mode is more difficult to understand than **Pointer Mode** because the target is shown with up side = North.

In **Pointer Mode** the pointer is always relative to where Lara is looking.

In **Pointer Mode** when the pointer is on the red sign this means Lara is looking at the target.

In **Radar Mode** it is advisable that Lara looks North to understand if targets are on her left or right.

16 \$0010: DTF_REQUIRED_ITEM

Used in the Detector= command.

The Detector will only work if Lara picked it up, or she has it from the start of the level in the Inventory

1 \$0001: DTF_SWINGING_POINTER

Used in the Detector= command.

This only works in Pointer Mode.

By default the pointer points to the target.

This flag adds a swinging simulation to the pointer like the compass in the Inventory to get a more realistic detector.

DWF_constants

not in mnemonics list

EDGX_constants

128 \$0080: EDGX_ANIMATION_SLOT

Used in the Diagnostic Type script command.

Adding this gives an on screen signal when the condition for the animation slot with a negative **AnimIndex** is true or false.

This works in the same way as the **DGX_ANIMATION** flag.

Type an animation slot command where the **AnimIndex** field has a negative value.

The animation will not be performed but the conditions will be tested and when they are true or false the result will be drawn on screen.

1 \$0001: EDGX_CONCISE_SCRIPT_LOG

Used in the Diagnostic Type script command.

Add this flag in the **Extra** field of the Diagnostic Type command to create a shorter debugging log. When omitted the log will be very large with messages about false conditions, frame for frame.

If the full log is too big create a concise log with this flag.

64 \$0040: EDGX_CUTSCENE_LOG

Used in the Diagnostic Type script command.

Enable the creation of a log for the cutscenes.

The log will be saved in the folder with a name "**custscene_log.txt**" and it will give information about:

Demo game commands for the current **demo.pak** playing.

The game commands are "action", "jump", "left", "right" etc.

Speech commands for the currently enabled **PARAM_ACTOR_SPEECH**

Organizer commands linked with the current demo

(only for the organizer with **FO_DEMO_ORGANIZER** flag)

8 \$0008: EDGX_LARA_CORD_IN_LOG

Used in the Diagnostic Type script command.

This extra flag only works when the **DGX_LARA** is enabled to have Lara's data (animations, state Ids) on the screen.

If this flag is added in the **ExtraDgxFlags** of the Diagnostic Type command the log will show the coordinates of Lara.

EDGX_constants

32 \$0020: EDGX_RECORDING_DEMO

Used in the Diagnostic Type script command.

This flag enables the in game demo recorder.

You can read the available keyboard commands on the screen but it is better to read the following notes and hints.

The maximum duration for the recording is about 25 minutes.

If you reach this limit the recording will automatically stop.

In the first row of the Recorder there will be the name of the **demo#.pak** file that is currently active. If this name has a # sign in front it means that the **demo#.pak** file already exists on the hard disk. Take care that when you enable the recording with a demo file with the # sign it is going to be overwritten.

You can change the active current **demo#.pak** file using the keys **"Q"** and **"W"**.

"Q" reduces by 1 the current **Demo index**.

"W" increases by 1 the current **Demo index**.

Valid range is between 1 to 999

Editing of demo files includes a simple backup method.

Every time the current **demo.pak** is erased using the **"E" (erase)** command, **TRNG** will rename the current file **demo#.pak** to **demo#.backup** .

The **demo#.pak** file is not erased but its content is saved in a corresponding **demo#.backup** file. A backup file is also created when a new (fresh) recording is selected and it has an existing **demo#.pak** file.

To restore the backup file and make it the main **demo.pak** file (cancelling the last erase/recording overwrite operation) use the **"R" (restore)** command.

In the demo editing information on screen you can also see the indices of the currently selected **Leading actor ("L-actor")** or **Extra Actor ("E-actor")**.

If you do not see the above descriptions it means that you have not assigned a role for that (missing) actor role.

If in the script the organizer is linked with the demo you are recording or playing, it will be performed in the editing mode.

This can be disabled (but only in the editing mode) with the **F4 key**.

EDGX_constants

At the start the game will increment to a higher free demo index.
This means a new demo file can be recorded avoiding over writing.
To play a demo file from the disk use the **Q and W keys** to choose the demo file and then hit the **F10 key**.

When an existing demo file (it has the # sign in front of demo name) is chosen, but the information to play is missing "**F10 to play the demo**", this means that the demo has not been recorded in the current level.

When playing a demo file: press the **Escape key** to stop it.
Trim it with the **F11 or F12 keys**.

Escape only stops the the current demo playing with no changes.

The **F11** key will remove part of the demo from the start up to the current position.
The **F12** key will stop the the demo playing and truncate it to the end.
Use the **F11** and **F12** keys to edit a demo file, removing the beginning or end part.

If the **Escape** command does not work in the level section where the **demo.pak** file is playing it is because the demo= script command is missing the **DEMF_QUIT_WITH_ESCAPE** flag.

During editing the **demo.pak** files in a game, it is better to add the flag to a demo script command. Or temporary remove the demo script command from the script with a semicolon ";" character.

When playing is in progress it can be paused, freezing the game by pressing and holding the **F8** key. This is useful to read the current frame number. Then this frame information can be used with a **demo organizer**. Or a condition **C86** or **C87** can be set to perform a trigger when the demo is playing the frame to give better synchronization for special effects.

Every time the **F8** key is pressed to stop demo, the frame number is saved to a log file **cutscene_log.txt**. So stops at different moments can be made and then the stop-point frames can be read.

Choosing **C86** and **C87** conditions or **Demo organizer (FO_DEMO_ORGANIZER flag)** synchronized with the demo.
This depends on the number of triggers to perform in the progress of the demo.
If there are only one or two triggers it is easier to use **C86** and **C87** condition.
If there are more triggers it is better to use a **Demo organizer**.

The **F9** key allows the "**Add recording**" operation.
This operation continues the recording of the current demo, after a previous recording or playing phase.

EDGX_constants

It is not always possible to have the add recording:

If a demo is selected with **Q/W** keys, the add recording will not be possible until the current demo is stopped.

It is necessary that Lara is in the final position of the current demo to continue the recording from final position.

To replace the second part of a demo with a new recording with the **F9** key.

Play the demo and then stop it with the **F12** or **Escape** key.

The next **F9** key will add a recording after the previous stop point, overwriting the (previous) final part.

The add recording operation maybe useful in two circumstances:

When a long and complicated demo is recorded and the first part is to be kept, but the final part needs to be edited.

To add many special effects to the cutscene/demo but the position of the frame where to set these triggers is required before continuing. In this case record a part of the demo, then find the frame where to add a trigger. Then repeat the demo and at the end, press the **F9** key to continue the recording.

Remark: When a demo is stopped and the recording is to continue, choose a stop point where Lara is still, otherwise the final demo playing will have problems of synchronization.

Do not stop a recording while Lara is falling, because when recording starts again Lara will continue to fall and the position of Lara will not be her real final position from the previous recording. The same applies to Lara's speech, running or performing other actions as she does.

Remember that the demo just recorded does not always works fine.

For example in the recording phase, Lara changed some items in the game that will be different (missing/killed, moved, already enabled etc.)

For instance if Lara, in the recording phase, enables an enemy and then she kills him, when you perform the play the enemy will be missing as he has already been enabled and killed.

To avoid this problem it is better to save the game before beginning a recording.

Play the demo reloading the save game to come back to the previous game situation and then hit the **F10** key to play the demo from same initial status.

Another situation where the playing could be different from the recording is when Lara interacts with enemies. Since there are some random actions in some enemies, these different movements could affect the behaviour of Lara or when an enemy touches her moving (or turning) her. The the more certain demos are those where Lara has no enemy to aim at and no enemies that touches her.

EDGX_constants

The demo files are saved and loaded from the Data folder. In the data folder there is also a **DemoInfos.txt** file with a descriptive list of **demo#.pak** files.

There are some limitations about the start position of Lara for recording.
Lara can not be hanging on a rope or on a pole-rope or driving a vehicle.
It is not possible to begin the recording from the inventory or other "paused" screen.
Lara can do all the actions but only in the middle of the recording.
This means Lara starts from the stand-up position and then moves to hang on a pole-rope, drive the jeep, enter in inventory or chooses an item etc.

Theoretically it should be possible to start a recording while Lara is standing-up, climbing, monkeying, running, jumping, shooting, falling, floating on water or swimming underwater.
However it is problematic starting with Lara walking (**SHIFT+Arrows**) because the mode to detect down/release of the shift key is very particular.

During the middle phase of recording avoid loading save games since a demo should only work within a level environment.

The demo recorder is only able to record the standard game commands, those seen in the Option/Control Configuration screen, plus the digits between 0-9. If custom scancodes are used to perform a special animation or other new features, these will be not be recorded and played.

Use the **Action (CTRL)** key to confirm numbers and not the **[Enter/Return]** key.

Death of a demo file.

The demos will probably be recorded on the final version of your levels.
If something is changed after recording a demo, you have to learn when these changes could create troubles for demo playing.

Avoid the change of equipment in the inventory items. In the demo Lara may pick an item from the inventory which is missing, the movement to select the correct item could change. In this situation a trick could be to record the demo when the selected item is the first item selected in the inventory and one that Lara will choose.

It may be dangerous to change a Lara object in the wad file, adding or removing animations, frames or meshes.

Do not change the geometry of the room where Lara will move through in the demo. It is not serious to add new rooms or change textures, triggers, sounds or other non-collisional stuff.

EDGX_constants

Remember the fundamental improvement of the **C25 condition trigger**: now there is the new mode "**Demo/Cutscene mode**" for this condition.

Thank to this condition you can place trigger using the **C25** condition that only be enabled in demo mode. This could be useful to improve the demo/cutscene with many triggers that will be not be triggered in the common game mode. Also if Lara comes back in the room where you pass her in the demo. Another important benefit is that these triggers (working only in demo mode) to prepare the demo with all the correct enemy enabled, door open, weapon/pickups etc.

For instance...

To show a demo where Lara fights the final monster of the level using a new weapon that in the normal game she will only get after picking up many sub-items. The demo needs to have Lara close to the monster's place with the special weapon in the inventory and the monster enabled.

It is important to understand that the demo does not save all features of the game, only the Lara position+status and her following movements, given from input control.

So for a demo from the title Lara will begin when the level starts so she will have NO special weapon and she will be far from the place with monster.

The distance is not a problem if the recording begins when Lara is close to the monster room. To solve other problems, place in the sector where the recording begins, the **C25** triggers to add in the inventory the special weapon and enable the monster. Only record from this sector close to the monster where Lara will trigger the adding of the special weapon and the monster.

All of the **C25** condition triggers are enabled ONLY in the demo mode. They will not affect the game playing when Lara moves over them in the normal game. It is necessary to understand that the triggers have to be present in the final release of the level, otherwise the demo will not work. So you can not temporary change the level (adding triggers) to record the demo and then remove these triggers.

To reconstruct the situation that Lara should have a different moment in the level, place **C25** triggers that will reproduce equipment and situations that in the normal play require a lot of time.

EDGX_constants

Tips and tricks:

Adding a different view of Lara with a fixed camera, flyby camera, maxtrix, or portrait effect improve the demo. However it complicates the recording of the demo to move Lara correctly when there is not a normal view of the game.

The trick to solve the problem is to add in triggers with **C25** condition to enable different viewing of the game. Record and drive Lara with a common view to avoid errors. After the recording is completed place the **C25+camera trigger** in the path of Lara in the demo and in the final playing there will be many camera views.

There is a new flip effect **(F379)**:

"Cutscene. Perform the **demo.pak** with [&index] in (E)way"

to launch a demo from a [Level] section to work like a cutscene.

To create a cutscene the **C25** triggers are very important:

- To change the camera view,**
- To set animating with special custom animations,**
- To add texts**
- To set a new background sound or music, etc.**

EDGX_constants

2 \$0002: EDGX_SLOW_MOTION

Used in the Diagnostic Type script command.

Use this flag with any combination of the **DGX_** diagnostic type.
Slow down the frame rate in the game by using the **F11** key.

Press **F11** once to enable the slow-motion.
To have an extreme slow motion press the **F11** key again.
To disable the slow-motion hit the **ESC** key.

The slow motion is useful to study custom animations or other dynamic movements where you want to view all details of the action.

4 \$0004: EDGX_SWAP_VIEW

Used in the Diagnostic Type script command.

Use this to enable a change of view when the **F12** key is pressed.
The view is from the direction where Lara looked.
This feature is useful to study an animation or mesh of Lara that is not visible from the back view.

16 \$0010: EDGX_TRIGGER_TIMING

Used in the Diagnostic Type script command.

The trigger timing will show on screen the distance in frame ticks between a trigger activation and the following.

This could be useful to synchronize **cutscenes** with the **organizer** in the script.
Another interesting use is to convert the old method of rolling ball
(to enable other triggers in the sequence) with an organizer.

The Organizer will reduce the used triggers on the map moving them into the script.

The time seen on the screen will only be the tick frames elapsed by the last trigger activation.

EF_constants

32 \$0020: EF_DOUBLE_DOOR

Used in the Elevator= command.

Do Not confuse this flag with **EF_SINGLE_DOOR** or **EF_MULTI_DOORS**.

These flags specify the number of doors to use.

If a single door or multiple doors move up and down with the elevator and there is one door for each floor, the double door specify the TYPE of single (or multi) door.

The double door is a door composed of two separate doors.

The double door is able to cover two sectors width, like the width of the elevator.

The reason to use this flag is to have two doors opening at the same time when the elevator reaches a floor to be able to cover the width of the elevator.

This flag has been added to the target and it requires some attention to make it work correctly.

If the **EF_DOUBLE_DOOR** flag is set, it is necessary that the secondary door has an index higher by 1 from the main door.

The "main" door has the index typed in the Elevator= command.

The "secondary" door is the door paired to the main door to open and close at the same time.

This second door has to have an index exactly +1 of the main door.

For example: If the main door on the first floor in a multi-door elevator has an index = 23, the secondary door should have an index = 24 (+1 of main door).

If this rule is violated the double doors will not work in the game.

If there is a problem to having two consecutive indices use this simple trick:

Place a door in a free zone of your room, this zone is not in front of the elevator, it will only be a "store" temporary zone.

Continue to place doors in this zone, checking the index assigned from **Tomb Editor** for each new door.

When there are two doors with consecutive indices delete the two doors and place the two doors in the correct position in front of the elevator.

Place other double doors for the other floors of the elevator if there is a multi-door elevator.

When the placement of the double doors is completed, come back to the temporary zone and delete all of the doors placed to realize this trick.

EF_constants

Remark: If a "single-door" animating is used to place a door with the correct collisions, the behaviour of this animating door will be different according to the use of the **EF_DOUBLE_DOOR** flag.

If the **EF_DOUBLE_DOOR** flag is missing the animating door will be moved sliding in an internal direction, i.e. the door will open moving the door over the other panel of the elevator

When the **EF_DOUBLE_DOOR** flag is used, the two doors will be opened with an external movement: the left door will be moved to the left, while the right door will be moved to the right.

For animating doors use the rule of consecutive indices: the secondary door should have the index of main door plus one.

In the case of animating double doors it is also important that the two doors should have the same orientation (facing).

Another rule about animating doors is for the door set as the main door.

The index typed in the Elevator command should be the door on the left of the elevator, looking at the elevator from the door side.

This last rule is not important when real doors are used.

EF_constants

4 \$0004: EF_INNER_KEYPAD

Used in the Elevator= command.

To insert the keypad to choose the floor inside the elevator and in **Tomb Editor** place the keypad oriented on the elevator wall.

Type the index of the keypad in the **InnerKeyPadIndex** field.

Remember to use the "fake" keypad found in ANIMATING16_MIP of ng.wad or ng2.wad and not the real keypad in SWITCH_TYPE1.

The reason for this choice is given by a technical problem:

The SWITCH_TYPE requires a switch trigger to work, but the elevator floor has dummy triggers. Therefore it is not possible to place both special triggers in some sectors.

The management of "fake" keypad (of ANIMATING16_MIP) will be performed in the hard-coded mode by the **TRNG** engine.

The keypad will work in the same way with the SWITCH_TYPE1 but with no need of a special trigger to handle it.

16 \$0010: EF_MODE_STOP_AND_GO

Used in the Elevator= command.

STOP_AND_GO mode is similar to **YO_YO** mode.

In this case the elevator stops and stays for some time at each floor.

2 seconds for an elevator with no doors,

3 seconds for an elevator with doors,

Use **STOP_AND_GO** mode and place the doors (multi or single) and set a number of floors.

STOP_AND_GO mode may hurt Lara (like all elevators)

EF_constants

8 \$0008: EF_MODE_YO_YO

Used in the Elevator= command.

YO_YO mode is a elevator that will move UP and DOWN endlessly without any stops at floors.

Using this mode doors cannot be used because there is no time to open and close them.

Lara will have to jump in and out of the elevator.

An interesting application of **YO_YO** mode is to give elevators the role of a trap.

If Lara is touched by the elevator she will be killed.

That is:

- If she is under the elevator moving down,
- If she is over the elevator when it moves up and there is a ceiling
- If she is "disturbed" climbing the wall when the elevator touches her.

Remarks: **YO_YO** mode and **STOP_AND_GO** mode cannot be mixed together.
If the elevator is set for **YO_YO** mode the number of floors will always be two (2).
The distance between floors will be the height of the whole movement that the elevator covers before changing direction.

EF_constants

1 \$0001: EF_MULTI_DOORS

Used in the Elevator= command.

Set this for the **TRNG** engine to handle all doors on any floor and then set in the **IndexFirstDoor** the index of the door on the first floor.

The management of the **TRNG** engine is very simple.

When the elevator reaches a floor the door on that floor will be opened.

When the elevator leaves a floor the door on that floor will be closed.

If multi-doors handling is used it is important to place all doors in the project at the same vertical distance set in the **ClickFloorDistance** field.

It is important because if a door is placed at a different distance that door will not be found by the **TRNG** engine.

Place each door at the same height where the elevator will stop, floor for floor.

0 \$0000: EF_NONE

Used in the Elevator= command.

2 \$0002: EF_SINGLE_DOOR

Used in the Elevator= command.

Set for a single door mode.

In single door mode the elevator has a single door linked with the elevator cage.

This door will move up and down with the elevator movement.

When the elevator reaches a floor the door will be opened.

When the elevator leaves a floor the door will be closed.

Set the type in the **FirstDoorIndex** field the index for the door placed near the elevator.

Remark: Use a "fake" door found in the slot Animating2 in the ng2.wad
This animating will be moved, sliding horizontally like a door
by the **TRNG** engine.

If the single-door mode is used set its index in the **FirstDoorIndex** field.

There is a reason to use a fake door:

When a real door is used, in some circumstances the engine will add an invisible sector collision in front of the door when it is closed.

It depends on the door type and position of the door with respect to rooms linking.

ENABLED_constants

1 \$0001: ENABLED

Enabled.

ENV_constants

48 \$0030: ENV_ANIM_COMPLETE

Works on the current animation of Lara.

The condition is true only if the current animation is at the last frame.

32 \$0020: ENV_CLIMB_LEFT_IN_CORNER

Verify if Lara is near to at an internal corner on the left.

See the description of **ENV_HANG_LEFT_IN_CORNER** constant to understand the difference between internal and outside corners.

33 \$0021: ENV_CLIMB_LEFT_OUT_CORNER

Same meaning as the **ENV_CLIMB_LEFT_IN_CORNER** but for the outside corner.

36 \$0024: ENV_CLIMB_LEFT_SPACE

Works like the **ENV_HANG_LEFT_SPACE** but it should be used when Lara is in climb mode

34 \$0022: ENV_CLIMB_RIGHT_IN_CORNER

Same meaning as the **ENV_CLIMB_LEFT_IN_CORNER** but for the right side.

35 \$0023: ENV_CLIMB_RIGHT_OUT_CORNER

Same meaning as the **ENV_CLIMB_RIGHT_IN_CORNER** but for the outside corner.

37 \$0025: ENV_CLIMB_RIGHT_SPACE

Works like the **ENV_HANG_RIGHT_SPACE** but it should be used when Lara is in climb mode

7 \$0007: ENV_CLIMB_WALL_AT_LEFT

Condition is true when the specific wall in the same sector with Lara is climbable.

ENV_constants

6 \$0006: ENV_CLIMB_WALL_AT_RIGHT

Condition is true when the specific wall in the same sector with Lara is climbable.

16 \$0010: ENV_CLIMB_WALL_BACK

Condition is true when the specific wall in the same sector with Lara is climbable.

5 \$0005: ENV_CLIMB_WALL_IN_FRONT

Condition is true when the specific wall in the same sector with Lara is climbable.

ENV_constants

58 \$003A: ENV_CONDITION_TRIGGER_GROUP

This **ENV_** condition allows a **Trigger Group** to perform when an exported condition is true

In the **DistanceForEnv** type the **IdTriggerGroup** to check.

If the final condition of the Trigger Group is true
the **ENV_** condition for the Animation command is true.

31 \$001F: ENV_DISTANCE_CEILING

Used in the Animation= command.

Condition is true if the distance from Lara to the ceiling is greater or equal to the value in the **DistanceForEnv**.

For example: Type in the **DistanceForEnv** the value \$300 (3 clicks).
The condition will be true if the coordinate Y of Lara
(on her feet when she is standing up) is 3 clicks or more
from the ceiling in the current sector.

64 \$0040: ENV_DISTANCE_EAST_WALL

Used in the Animation= command.

This condition will verify if there is a wall within the **DistanceForEnv** field to the East of Lara.

39 \$0027: ENV_DISTANCE_FLOOR

Used in the Animation= command.

Verify if the distance between Lara and the floor is greater or equal to the value set in the **DistanceForEnv** field

Remark: For the opposite condition,
i.e. the distance from the floor is less than or equal to the
DistanceForField add to the
ENV_DISTANCE_FLOOR the special flag
ENV_NON_TRUE.
This method works for all other conditions where
it is required to invert the condition.

ENV_constants

62 \$003E: ENV_DISTANCE_NORTH_WALL

Used in the Animation= command.

This condition will verify if there is a wall within the **DistanceForEnv** field to the North of Lara.

WARNING: This condition works in an absolute way looking only at the cardinal point, ignoring the current facing of Lara so it should only be used with other **ENV_** conditions to apply to a specific room in a hard-coded way.

This condition considers a "wall" on any floor higher than the current position of Lara.

63 \$003F: ENV_DISTANCE_SOUTH_WALL

Used in the Animation= command.

This condition will verify if there is a wall within the **DistanceForEnv** field to the South of Lara.

65 \$0041: ENV_DISTANCE_WEST_WALL

Used in the Animation= command.

This condition will verify if there is a wall within the **DistanceForEnv** field to the West of Lara.

71 \$0047: ENV_ENEMY_SEE_LARA

Used in AnimationSlot= command.

It checks if the current enemy is able to see Lara.

In the **DistanceForEnv** field type the angle of view of the enemy.

Type the view angle in degrees, where 180 means the enemy is able to control a semi cycle in front of him.

If 360 is typed this means that the facing (orienting of enemy) will not be used to verify the chance to see Lara but only the presence of obstacles (walls, items) in the line between the enemy and Lara.

A reasonable value for the **human view angle** is **160 degrees**.

Remark: The angle of view works for the horizontal view
The vertical view will use the prefixed value of 90 degrees.
This means that if Lara is higher or lower than an enemy he will not be able to see her.

Another prefixed limitation is the distance.
The enemy will never be able to see Lara when she is at 0 sectors or too far away.

ENV_constants

45 \$002D: ENV_FLOATING

Used in the Animation= command.

Performs an animation only if Lara is floating over the water surface

50 \$0032: ENV_FLYING_DOWN

Used in the Animation= command.

Perform an animation only when Lara is jumping and she is in a fall down phase

49 \$0031: ENV_FLYING_UP

Used in the Animation= command.

Perform an animation only when Lara is jumping and she is in an upward phase

40 \$0028: ENV_FRAME_NUMBER

Set in the **StateId** Animation array a negative number for the current animation to specify the precise frame when the condition will be true.

For example: For a special animation to start only when Lara is performing frame number 21 of the animation 325, type "-325" in the state-Id animation array (last field of Animation command) and set frame 21 in the envelope condition **ENV_FRAME_NUMBER** and in the **DistanceForEnv** field type 21.

43 \$002B: ENV_FREE_HANDS

Used in the Animation= command.

The animation will only be started when Lara has no object in her hands.

If she is taking weapons, a crowbar or she is hanging on a wall, climbing or monkey swinging the animation will NOT be performed

Note: This flag ignores the flare, because when Lara has it in her hand the game engine considers Lara has free hands.

ENV_constants

25 \$0019: ENV_HANG_LEFT_IN_CORNER

Used in the Animation= command.

Verify if Lara is near to a corner on her left when she is in hang mode.

In the **DistanceForEnv** type the minimum distance from the corner for the condition to be true.

Remark: The "_IN_" means internal corner.
The internal corner are found inside a box,
while the "_OUT_" (outside) corners are outside a box.

ENV_constants

26 \$001A: ENV_HANG_LEFT_OUT_CORNER

Used in the Animation= command.

Same meaning as the [ENV_HANG_LEFT_IN_CORNER](#) but for an Outside corner.
See the description of the [ENV_HANG_LEFT_IN_CORNER](#) more information about internal and outside corners.

29 \$001D: ENV_HANG_LEFT_SPACE

Used in the Animation= command.

Verify if there is space on the left of Lara in hang mode.

In the **DistanceForEnv** type the space required on the left for a true condition.

Remark: A value cannot be specified for the distance that passes over the current sector.
If the value is too big the **TRNG** engine will reduce it to reach the left border of the current sector.

27 \$001B: ENV_HANG_RIGHT_IN_CORNER

Used in the Animation= command.

The Same meaning as the [ENV_HANG_LEFT_IN_CORNER](#) but for the right side.
See the description of the [ENV_HANG_LEFT_IN_CORNER](#) constant

28 \$001C: ENV_HANG_RIGHT_OUT_CORNER

Used in the Animation= command.

Same meaning as the [ENV_HANG_RIGHT_IN_CORNER](#) but for the outside corners.
See the description of the [ENV_HANG_LEFT_IN_CORNER](#) constant

30 \$001E: ENV_HANG_RIGHT_SPACE

Used in the Animation= command.

Same as the [ENV_HANG_LEFT_SPACE](#) but for the right side.
See the description of the [ENV_HANG_LEFT_SPACE](#) for more information.

ENV_constants

2 \$0002: ENV_HANG_WITH_FEET

Used in the **Animation=** command.

Used only when Lara is hanging on the wall corner.

The condition is only true when the wall where Lara hangs is high enough.

In the **Distance** field change the height of the wall.

The **default value is \$300** (3 clicks)

In some circumstances it is better to use a lower Distance value, like \$200 (2 clicks)

57 \$0039: ENV_HOLD_EXTRA_ITEM_IN_HANDS

Used in the **Animation=** command.

The condition is true when Lara holds in her hands the **HOLD_** item typed in the **Extra** field.

See the **HOLD_** constants

9 \$0009: ENV_HOLE_BACK_CEILING_CLIMB

Used in the **Animation=** command.

Use the condition only when Lara is monkey swinging on the ceiling and an animation is to be created so that Lara can pass from the current ceiling to a higher climbable wall, passing directly from the ceiling to a climbable wall.

13 \$000D: ENV_HOLE_FLOOR_AT_LEFT

Used in the **Animation=** command.

The condition is true when there is a hole on the left, i.e. a zone where Lara is able to go down or fall.

In the **Distance** field set the depth of the hole.

12 \$000C: ENV_HOLE_FLOOR_AT_RIGHT

Used in the **Animation=** command.

The condition is true when there is a hole to the right, i.e. a zone where Lara is able to go down or fall.

In the **Distance** field set the depth of the hole.

ENV_constants

14 \$000E: ENV_HOLE_FLOOR-BACK

Used in the Animation= command.

The condition is true when there is a hole behind Lara.

That is a zone where Lara is able to go down or fall.

In the **Distance** field set the depth of the hole.

3 \$0003: ENV_HOLE_FLOOR_IN_FRONT

Used in the Animation= command.

The condition is true when there is a hole in front of Lara.

That is a zone where Lara is able to go down or fall.

In the **Distance** field set the depth of the hole.

8 \$0008: ENV_HOLE_IN_FRONT_CEILING_CLIMB

Used in the Animation= command.

Use this condition only when Lara is monkey swinging on the ceiling and an animation is to be created so that Lara can pass from the current ceiling to a higher climbable wall, passing directly from the ceiling to a climbable wall.

ENV_constants

52 \$0034: ENV_IN_LEFT_SIDE_SECTOR

Used in the Animation= command.

This condition checks only the position of Lara with respect to the current sector.

If Lara is near to the left border of the current sector the condition is true.

This means the condition is true when Lara is (virtually) touching with her left arm the wall in the sector on the left

In the **DistanceForEnv** fields change the compute for distance.

If a big value is set the condition is true when Lara is not close to the left border of the current sector

If a small value is set the condition is only true when Lara is close to the left border of the current sector.

Remark: Use this condition together with others because alone it is not useful.
It does not check if the sector on the left of the current sector has a wall that Lara could touch.

Use this condition with the
ENV_NON_TRUE+ENV_NO_BLOCK_AT_LEFT condition to verify
if Lara could touch the wall on the left side.

The default value for **DistanceForEnv** is 128 (1024 = one sector).

This is a good value to compute touching of the left wall when Lara is in the stand up position.

Type **IGNORE** in the **DistanceForEnv** field for a default value of 128.

ENV_constants

53 \$0035: ENV_IN_RIGHT_SIDE_SECTOR

Used in the **Animation=** command.

This condition works in a similar way to the **ENV_IN_LEFT_SIDE_SECTOR** condition but in this case it works for the right side.

See the description of the **ENV_IN_LEFT_SIDE_SECTOR** constant to understand the logic of this condition.

47 \$002F: ENV_IS_STILL

Used in the **Animation=** command.

Perform an animation only when Lara has no horizontal or vertical speed

54 \$0036: ENV_ITEM_EXTRA_AT_LEFT

Used in the **Animation=** command.

This condition works like the **ENV_ITEM_EXTRA_IN_FRONT** but it analyses for an item at the left of Lara.

Insert the slot for the wanted item in the **Extra** field and the distance in the **DistanceForEnv** field.

55 \$0037: ENV_ITEM_EXTRA_AT_RIGHT

Used in the **Animation=** command.

This condition works like the **ENV_ITEM_EXTRA_IN_FRONT** but it analyses for an item at the right of Lara.

Insert the slot for the wanted item in the **Extra** field and the distance in the **DistanceForEnv** field.

21 \$0015: ENV_ITEM_EXTRA_IN_FRONT

Used in the **Animation=** command.

Verify if there is an object (moveable) with a slot number equal to the number set in the **ExtraSlot** field of the Animation command.

In the **DistanceForEnv** field type the distance where the object is wanted. (1024 = sector).

ENV_constants

22 \$0016: ENV_ITEM_EXTRA_OVER

Used in the **Animation=** command.

Similar to the **ENV_ITEM_EXTRA_IN_FRONT** but in this case the object should be over Lara on the same sector.

In the **DistanceForEnv** field type the height above Lara.

See the description of the **ENV_ITEM_EXTRA_IN_FRONT** for more information.

23 \$0017: ENV_ITEM_EXTRA_UNDER

Used in the **Animation=** command.

Similar to the **ENV_ITEM_EXTRA_OVER** but in this case the object is below Lara.

If the **DistanceForEnv** field is set to 0 the object is near to the feet of Lara.

56 \$0038: ENV_ITEM_TEST_POSITION

Used in the **Animation=** command.

This condition is used to detect if there is an object of given **TestPosition** in the correct position with respect to Lara.

This has the same target in the **ENV_ITEM_EXTRA_** conditions but in this case it can accurately compare for an item in any position with respect to Lara and in absolute precise mode.

The other **ENV_** conditions have some problems to detect the exact distance when the pivot of Lara and the item are different.

To use type in the script a "TestPosition=" script command with all the necessary data for the detection.

Then type in the **DistanceForEnv** field of **Animation=** or **MultEnvCondition=** commands the **IdTestPosition** (first argument of TestPosition).

Remark: The **ENV_ITEM_TEST_POSITION** condition like the other **ENV_ITEM_EXTRA_** condition is limited to only check for items in the same room as Lara.
To detect an item in a room away from Lara set in the **TestPosition** command the flag **TPOS_TEST_ITEM_INDEX**
Set in the **Slot** field of **TestPosition** an index of a specific item instead of a generic slot type.

See the **Test Position** script command in the **NEW SCRIPT COMMANDS**.

ENV_constants

66 \$0042: ENV_LARA_IN_MICRO_STRIP

Used in the Animation= command.

This environment condition works in a similar way to the **ENV_POS_STRIP_1/2/3** flags. Remember that it is a condition and not an **ENV_POS** flag so it cannot add the **ENV_LARA_IN_MICRO_STRIP** value to another **ENV_** condition.

Place this **ENV_LARA_IN_MICRO_STRIP** condition in a MultEnvCondition script command and add to this condition an **ENV_POS_** flag like the **ENV_POS_HORTOGONAL** flag.

The **ENV_LARA_IN_MICRO_STRIP** allows the **ENV_POS_STRIP_1/2/3** flags the best precision, since the **ENV_POS_STRIP_1/2/3** flags can choose between three "fat" strips. The **ENV_LARA_IN_MICRO_STRIP** can choose any specific range within the sector where Lara should be to get a true for this condition.

When the **ENV_LARA_IN_MICRO_STRIP** condition is used type in the **DistanceForEnv** field the range (minimum and maximum) in micro-strips units where Lara should be enclosed.

The value to type in the **DistanceForEnv** field is given by the following formula: **Max*256 + Min**

Where **Min** and **Max** have values in the range 0 to 31.

Practically imagine a sector divided by 32 strips in the direction where Lara is looking and choose in what range of 32th strips there should be a true condition for Lara.

For example: To have Lara close to the outside border for a condition to touch a wall in the next sector choose the range 0 to 2.
The condition is true when Lara is in 0,1 or 2 micro strip.

Remember : The first strip (value= 0) is close to the border where Lara is looking,
The last strip (value = 31) is close to the border at the back of Lara.

ENV_constants

4 \$0004: ENV_MONKEY_CEILING

Used in the **Animation=** command.

The condition is true when Lara is below a monkey ceiling.

In the **Distance** field set the range of the distance of the monkey ceiling.

The distance is given by the formula : **MinClick** + **MaxClick** * 256

For example: For the condition to be true when the monkey ceiling is in the range 5 clicks to 7 clicks type in the distance field the value $(7*256 + 5) = 1797$

Remark: In hexadecimal ('\$' sign) it is easier to understand the **Min** and **Max** value. For example the number 1797 in hexadecimal is \$0705 where the **MaxClick** is "07" and the **MinClick** is "05".

If Lara is under the monkey ceiling but the height is outside the range set in the distance field the condition will be FALSE and the animation will not be performed.

ENV_constants

24 \$0018: ENV_MULT_CONDITION

Used in the Animation= command.

This signals a multiple condition.

The effective ENV_ condition flags are stored in a **MultiEnvCondition** script command.

In the **DistanceForEnv** field type the **Id** of the **MultiEnvCondition** command that will be used. The final condition will be true only if all the conditions in the **MultiEnvCondition** command are true at the same time.

To set in the animation command two ENV_ conditions, like: **ENV_CEILING_HEIGHT** with a height distance ENV = \$300 (3 clicks) + another condition:

ENV_HOLE_FLOOR_AT_LEFT with depth (distance ENV) = \$400 (4 clicks)

create this Multi Envelope Condition command:

MultiEnvCondition= 1, ENV_CEILING_HEIGHT, \$300, ENV_HOLE_FLOOR_AT_LEFT, \$400

and then type in the Animation command the reference Id of the above MultiEnvCondition (1) and the ENV_ condition: ENV_MULT_CONDITION

For example: Animation=447, KEY1_LEFT,
IGNORE,IGNORE,ENV_MULT_CONDITION, 1,IGNORE,-445, -448

38 \$0026: ENV_MULT_OR_CONDITION

Works like the ENV_MULT_CONDITION but the final condition will be true if a single condition of the Multi Envelope Condition command is true.

See the description of the **ENV_MULT_CONDITION** for more information.

ENV_constants

11 \$000B: ENV_NO_BLOCK_AT_LEFT

The condition is true only if there is no wall in the sector on the left of Lara.
In the **Distance** (next field) type the height of the wall.
See the **DistanceForEnv** field.

10 \$000A: ENV_NO_BLOCK_AT_RIGHT

The condition is true only if there is no wall in the sector on the right of Lara.
In the **Distance** (next field) type the height of the wall.
See the **DistanceForEnv** field.

15 \$000F: ENV_NO_BLOCK_BACK

The condition is true only if there is no wall in the sector behind Lara.
In the **Distance** (next field) type the height of the wall.
See the **DistanceForEnv** field.

1 \$0001: ENV_NO_BLOCK_IN_FRONT

Used in the Animation= command.

Perform a special animation only if Lara has NO block in front of her face.
In the **DistanceForEnv** field set the height of the block.

For example: To set 1024 (height of one sector) the **TRNG** engine will consider true the condition "no block in front" if the floor height in front of Lara is less than 1024 units with respect to the current floor where Lara is standing. If the floor in front of Lara is higher than 1024 units a block will be detected in front of Lara.

ENV_constants

68 \$0044:: ENV_NO_BOX_AT_LEFT

Used in the AnimationSlot= command.

Check to the left of the current enemy for NO grey box sector.
See the description of the [ENV_NO_BOX_IN_FRONT](#) constant for more information.

69 \$0045: ENV_NO_BOX_AT_RIGHT

Used in the AnimationSlot= command.

Check to the right of the current enemy for NO grey box sector.
See the description of the [ENV_NO_BOX_IN_FRONT](#) constant for more information.

70 \$0046: ENV_NO_BOX_BACK

Used in AnimationSlot= command.

Check to the back of the current enemy for NO grey box sector.
See the description of the [ENV_NO_BOX_IN_FRONT](#) constant for more information.

67 \$0043: ENV_NO_BOX_IN_FRONT

Used in the AnimationSlot= command.

Check to the front of the current enemy for NO grey box sector.
This condition detects the grey boxes used in the [Tomb Editor](#) to stop the moving of enemies.
In the **DistanceForEnv** field set the distance to check in front of the enemy.

The "BOX" are the grey box sectors placed in the [Tomb Editor](#) to inhibit the enemies movement.
See the [Tomb Editor Manual](#).

Note: The _NO_BOX_ conditions should be used with the **AnimationSlot** command on a enemy.

The use with Lara is theoretically possible but not very useful.

ENV_constants

128 \$0080: ENV_NON_TRUE

Used in the Animation= command.

Invert the current ENV_ condition.

Add this value to an ENV_ condition when to make the condition true when it is false and vice versa.

For example: Looking at the ENV_ conditions find the ENV_NO_BLOCK_IN_FRONT condition. If this is used on its own it means "perform the special animation only if Lara has NO Block in front of her face".

To perform an animation requiring a block in front of Lara use the same condition ENV_NO_BLOCK_IN_FRONT + ENV_NON_TRUE to invert the condition.

In this combination there is a double negative that means:

"when Lara HAS a block in front.. . perform the special animation".

Add the ENV_NON_TRUE flag to any ENV_ condition and remember that the ENV_NON_TRUE flag has NO effect on the ENV_POS ... flags.

42 \$002A: ENV_ON_VEHICLE

Used in the Animation= command.

By default the start of the animation cannot happen if Lara is driving a vehicle.

To create a new animation for Lara while she is on a vehicle use this ENV_ condition and type in the Extra field the Slot number of the vehicle.

Remark: The name of the slot cannot be put in the Extra field.
Input the number read on the left of the slot name in the Slot moveable list.

46 \$002E: ENV_ONLAND

Used in the Animation= command.

Only perform an animation if Lara is on land.

ENV_constants

60 \$003C: ENV_PLAYER_IS_SLEEPING

Used in the Animation= command.

This is an original condition. It verifies how much time there is between keyboard commands.

Type in the **DistanceForEnv** field the number of tick frames (one second= 30 tick frames) required to enable this condition.

For example: For a true condition when the player does not perform a game command for 3 seconds, type in the **DistanceForEnv** field the value 90 (because $3 * 30 = 90$).

This condition could be used to start an animation in stand-by mode, i.e. when the player is not active (no keyboard signals).

The Standby animation could be Lara scratches her head, or turns her face to look at the player saying: "Are you sleeping?"

To use this animation in **standby mode** remember to set in this animation a state Id different from the standard state-Ids. For example, use the state Id **STATE_CONTROLLED** to avoid a new game command causing a bad change between a current custom animation and a new standard animation. Using the state Id neutral, the new animation will be performed until Lara comes back to a standard state Id.

Create an animation with a state Id = STATE_CONTROLLED and as a next animation the default animation of still stand-up.

Create a custom animation where Lara starts from a stand-up position, performs this move and then come back to the stand-up animation with the next animation having the standard stand-up animation.

ENV_constants

61 \$003D: ENV_PLAYER_WOKE_UP

Used in the Animation= command.

This ENV_ condition is the opposite of the ENV_PLAYER_IS_SLEEPING.

It happens when there is any game command.

In the ENV_PLAYER_IS_SLEEPING description is a way to create a stand-by animation for death times in the game. There is also another method and it is the situation ENV_PLAYER_WOKE_UP.

To perform a custom animation when there is no game commands (player is sleeping) put Lara in a new position in continue mode.

For example: She sits down and waits.

In this situation, since you used a non-standard state-Id, no keyboard command will be accepted, so it should be your Animation command and condition the ENV_PLAYER_WOKE_UP, to force another custom animation having the target move Lara from the sit down position to the stand-up position.

In this new animation set the next animation, the standard still stand-up animation.

Lara will now be able to receive game commands.

In the game the player could see a little show: Lara (bored) sit down and look left and right then, when the player hits a game command, for example the JUMP, Lara will not jump but she will come back to a stand-up position and if the player hits the jump key (or another commands) she will come back to respond to these commands.

ENV_POS_CENTRAL Please do not confuse this flag with the previous
ENV_POS_IN_THE_MIDDLE.

The ENV_POS_CENTRAL works fine together with some

ENV_POS_STRIP.. field, to set a central position in the specific strip you choose.

The ENV_POS_IN_THE_MIDDLE cannot be used with the ENV_POS_STRIP because that flag sets the position in the sector (at the centre).

1024 \$0400: ENV_POS_CENTRAL

Please don't confuse this flag with the ENV_POS_IN_THE_MIDDLE.

The ENV_POS_CENTRAL works fine with some

ENV_POS_STRIP.. field, to set a central position in specific strip you chose.

While the ENV_POS_IN_THE_MIDDLE can not be used with the ENV_POS_STRIP because that flag sets exactly the position in the sector (at the centre).

ENV_constants

2048 \$0800: ENV_POS_HORTOGONAL

Hortogonal means :non diagonal.

For many animation it is necessary that Lara is facing a wall otherwise she will be not able to climb, hang etc.

Add this **ENV_POS** flag and the condition will be **FALSE** if Lara is diagonal to the wall lines.

4096 \$1000: ENV_POS_IN_THE_MIDDLE

Lara is (almost) exactly at the centre of the current sector.

For example: This is the position for Lara when she is able to jump over a rope.

256 \$0100: ENV_POS_LEFT_CORNER

Works like the **ENV_POS_RIGHT_CORNER** but for the left side.

See the description of the **ENV_POS_RIGHT_CORNER** for more information.

512 \$0200: ENV_POS_RIGHT_CORNER

This condition is true when Lara is in the right corner of the current sector using as a reference the direction where she is looking.

ENV_constants

32768 \$8000: ENV_POS_STRIP_1

Lara is in the first (more forward) strip of three ideal strips of the current sector in respect where she is looking.

Use this flag if you want Lara to be near a switch or door or other items attached to a wall in front of her.

Remark: Add to the current ENV condition one or more [ENV_POS_](#) flags to specify the position of Lara with respect to the current sector.

16384 \$4000: ENV_POS_STRIP_2

Like the above, but in this case Lara is in the middle strip of three stripes of the current sector. See the description of the [ENV_POS_STRIP_1](#) for more information.

8192 \$2000: ENV_POS_STRIP_3

Like the above, but in this case Lara is in the third strip.

See the description of the [ENV_POS_STRIP_1](#) for more information.

59 \$003B: ENV_ROOM_IS

Used in the Animation= command.

This condition checks if the current room for Lara is the same number typed in the **DistanceForEnv** field.

Using this condition a custom animation can be created that will work in a room for the level.

To check for more than one room insert the [ENV_ROOM](#) condition list in a [MultEnvCondition](#) command.

Type one of the [ROOM_](#) constants to test if Lara is in the wanted room type.

See the [ROOM_](#) constants

Remark: In the **DistanceForEnv** field type the room number to check.

20 \$0014: ENV_SUPPORT_IN_BACK_WALL

Used in the Animation= command.

See the description for the [ENV_SUPPORT_IN_FRONT_WALL](#)

ENV_constants

17 \$0011: ENV_SUPPORT-IN_FRONT_WALL

Used in the Animation= command.

All the **ENV_SUPPORT** check where Lara could hang on the wall in front or back, left, right. Use this condition to start an animation where Lara jumps and hangs on a wall in front of her. Verify if there is a wall with the correct height in front of Lara.

For example: By default the **TRNG** engine allows Lara to hang on a wall if it has a height in the range 4 clicks to 7 clicks.
If the wall is 8 clicks or more Lara will not be able to hang with an up jump.

To set the same condition insert in the **DistanceforENV_** field the value: \$0174

For the **ENV_SUPPORT** condition the distance **ENV_** is complex..

The values to set are: The minimum acceptable height (4 clicks in the example)
The maximum acceptable height (7 clicks in the example)
The minimum space over the support to place Lara hands (1 click)

The formula is: **MinHeight** + **MaxHeight***16 + **ClickSpace** * 256

In decimal this value is : $4 + 7 * 16 + 1 * 256 = 372$

In hexadecimal format it is easy to understand the position of each sub-field:
372 decimal = \$174 in hexadecimal. i.e. in hexadecimal the digits are:

\$0BCA where: 'A' is click of minimum height required
'B' is click of minimum height acceptable
'C' is number of clicks of space over the support

Hexadecimal numbers use a single digit: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
A (10 decimal), B (11 decimal), C (12 decimal), D (13 decimal), E (14 decimal) and F (15 decimal).

Another example: To verify if Lara hangs and has enough space to move up over the new floor.

Use a distance like \$333 because: 3 clicks for minimum and maximum height the support has to be in a specific position, i.e. at Lara's hands.

The value 3 is the distance from Lara's feet.
The last "3", is the space necessary for Lara to stand up.
The above condition will be true only if the height of the space over the support where Lara hangs is at least 3 clicks high.

ENV_constants

19 \$0013: ENV_SUPPORT_IN_LEFT_WALL

Used in the Animation= command.

See the description for the [ENV_SUPPORT_IN_FRONT_WALL](#)

18 \$0012: ENV_SUPPORT_IN_RIGHT_WALL

Used in the Animation= command.

See the description for the [ENV_SUPPORT_IN_FRONT_WALL](#)

44 \$002C: ENV_UNDERWATER

Used in the Animation= command.

Perform the animation only if Lara is underwater

41 \$0029: ENV_VERTICAL_ORIENT

Used in the Animation= command.

Set a condition in accordance with the current vertical orientation of Lara.
Normally Lara has zero vertical orientation but in some animations this value could change.

Set a condition about the current vertical orientation using the condition:

ENV_VERTICAL_ORIENT and set in the **DistanceForEnv** field the range of orienting values computed in this way: $\text{MinValue} + \text{MaxValue} * 256$

For example: To set a range from 3 to 8, type \$83

Remark: For a range enclosing the zero (0) invert the maximum and minimum values.

For example: To enable a range accepting the following values: 13, 14, 15, 0, 1, 2, 3 type 13 minimum value and 3 maximum value, i.e. in hexadecimal: \$D3 (D = 13)

ENV_constants

51 \$0033: ENV_WALL_HOLE_IN_FRONT

Used in the Animation= command.

This condition works in a similar way to the ENV_SUPPORT_IN_FRONT_WALL condition. In this case it checks for a support in the front wall.

The TRNG engine will also check the height of the hole in the wall to set the minimum and the maximum height limits for the hole.

EXTRA_constants

1 \$0001: EXTRA_MUTANT_NO_LOCUSTS

Used in the Enemy= command.

Add this flag in the **Extra** field of the Enemy command to customize the **MUTANT** slot with an Enemy script command to disable the swarm locust attack.

1 \$0001: EXTRA_TEETH_NO_DAMAGE_ON_WALKING

Used in the Enemy= command.

Enable the NO Damage for Lara when she moves slowly (walking).
Set in the **Extra** field of Enemy= command with the **TEETH_SPIKES** slot.
Lara will be able to walk through **TEETH_SPIKES** with no damage.

1 \$0001: EXTRA_WRAITH_BURN_LARA

Used in the Enemy= command.

Type in the Enemy script command to customize a Wraith slot.
Enable this flag to burn Lara when a Wraith touches her.

FADD_constants

256 \$0100: FADD_CONTINUE_EMIT

Used in the AddEffect= command.

Set the value in the **DurateEmit** and **DuratePause** will be ignored.

Use this flag for effect types requiring a continuous emission, like Mist and flames.

32768 \$8000: FADD_DURATE_ANIMATION

Used in the AddEffect= command.

This works in a similar way to the flag **FADD_DURATE_STATEID**.

In this case it checks for the condition emit/disable for the animation number.

4096 \$1000: FADD_DURATE_SINGLE_FRAME

Used in the AddEffect= command.

This forces the emitting to only work for a single frame.

Then the effect will be disabled.

Only Blood can work with this flag.

4096 \$1000: FADD_DURATE_STATEID

Used in the AddEffect= command.

Keeps the effect active until a moveable is in the current state Id.

Using this flag the time **Durate** set in the trigger window will be ignored and the effect will be removed only when the moveable changes its state Id with a value different from that used at the start of the effect.

FADD_constants

4 \$0004: FADD_FIRE_STRIP

Used in the AddEffect= command.

This flag is used together with the **ADD_FLAME** type.

By default the **ADD_FLAME** shows a little fire.

If the **FADD_FIRE_STRIP** flag is used it will create a horizontal strip of fire like the flames shoot from a dragon or from a air-jet (from the back using the **FADD_ROTATE_180** flag).

2 \$0002: FADD_IGNORE_STATUS

Used in the AddEffect= command.

By default the **AddEffect** trigger will ignore the command of the **AddEffect** if the moveable has not been enabled in the game or it has already been killed or removed.

In some circumstances this compute could be wrong and with some special items the result could be to fail the adding effect operation.

For special items like some pickups add the **FADD_IGNORE_STATUS** flag.

Then the **TRNG** engine will ignore the status of the moveable and apply the effect if the item is invisible or disabled.

1 \$0001: FADD_NO_SOUND

Used in the AddEffect= command.

This flag disables the sound linked with effects like Mist or Fire.

0 \$0000: FADD_NONE

Used in the AddEffect= command.

If no **FADD_** constant is used set the **FADD_NONE** constant in the **FADD_** field.

16384 \$4000: FADD_ROTATE_180

Used in the AddEffect= command.

This works like the **FADD_ROTATE_90** flag but using a rotation of 180 degrees.

See the **FADD_ROTATE_90** flag

FADD_constants

2048 \$0800: FADD_ROTATE_270

Used in the AddEffect= command.

This works like the **FADD_ROTATE_90** flag but using a rotation of 270 degrees.

See the **FADD_ROTATE_90** flag

8192 \$2000: FADD-ROTATE_90

Used in the AddEffect= command.

The **FADD_ROTATE** flags are only used in circumstances when an effect type that uses the horizontal orientation of object to compute the direction of emitting.

For example, for the Mist strip to be oriented to the side instead of in front of the object use the **FADD_ROTATE_90** flag.

1024 \$0400: FADD_SMOKE_EXHAUST

Used in the AddEffect= command.

This flag is only used for the **ADD_SMOKE** effect.

There are two different smoke emitters: **Default is smoke that goes up and disappears,**
Exhaust smoke seen with the **JEEP** or **SIDECAR**.

16 \$0010: FADD_VORIENT_180

Used in the AddEffect= command.

This changes the vertical orientation of the moveable by 180 degrees before applying the compute of **DispX**, **DispY**, **DispZ**.

See the description of the **FADD_VORIENT_90** flag to understand the situation.

32 \$0020: FADD_VORIENT_270

Used in the AddEffect= command.

This changes the vertical orientation of the moveable by 270 degrees before applying the compute of **DispX**, **DispY**, **DispZ**.

See the description for **FADD_VORIENT_90** flag to understand the situation.

FADD_constants

8 \$0008: FADD_VORIENT_90

Used in the AddEffect= command.

This flag rotates the item 90 degrees in a clockwise direction on the vertical axis.

This temporary change is useful with some moveables where the position of the meshes has a different vertical orientation with respect to the position in the first animation.

This situation can be discovered by using **Wad Tool** and verifying the mesh when there is no animation selected.

It is important to understand that all **FADD_VORIENT_** constants are different from the **FADD_ROTATE_** constants.

The **FADD_ROTATE_** constants are used to give a horizontal direction (facing) for those effects moving on an ideal line like the Mist strip or Fire strip.

The direction of effects cannot be changed by using the **FADD_VORIENT_** constants but these flags can fix the bad vertical orientation of some moveables like the **JEEP**.

FALSE_constants

0 \$0000: FALSE_

FAN_constants

2048 \$0800: FAN_ALIGN_TO_ENV_POS

Used by the Animation= script command.

From version 1.1.9.8 this FAN_ has new properties.

When this flag is added and a valid ENV_POS_ constant flag is set in the **Environment** field (see the following description) the position of Lara will be forced to the ideal position in accordance with the ENV_POS_ values before performing the custom animation.

For example: If the ENV_POS_HORTOGONAL is set before starting the special animation the Lara orientation will be aligned with the Hortogonal axis.

Remark: By default the alignment operation is poor, because the TRNG engine moves Lara to an ideal position without any progressive movement.

From versions 1.1.9.8, when the FAN_ALIGN_TO_ENV_POS is set in accordance with the ENV_ITEM_TEST_POSITION condition there is a progressive self adjustment in the game as seen in the default tomb4 when Lara opens a door or pushes a pushable object.

This feature will use the data set in the TestPosition command and extracts from it the ideal position to reach using intermediate values for each difference range set.

This feature does not work if Lara is performing a state Id different from the stand-up position or swimming underwater.

Warning: When the ENV_ITEM_TEST_POSITION is set and the FAN_ALIGN_TO_ENV_POS orientation information will be read from the Test Position command, so further ENV_POS_ constant flags added in the EnvCondition will be ignored.

128 \$0080: FAN_DISABLE_GRAVITY

Used by the Animation= script command.

This flag works in a similar way to the FAN_ENABLE_GRAVITY.
In this case it disables the gravity compute.

See the description of the FAN_ENABLE_GRAVITY for more information.

FAN_constants

16 \$0010: FAN_DISABLE_PUSHAWAY

Used by the Animation= script command.

Add this flag if Lara is not to change her animation when she has been touched by enemies. The push-away animation is a **hard coded animation**, where Lara suspends a previous animation and moves herself slowly while the enemy is pushing her away. The **push-away animations have numbers: 125 126 127 128**

Note: This flag disables the push away animation only when the animation of Lara is the same as set in the current Animation command.

If an animation chain is used, where after the first animation there is another custom animation you could see Lara with a push-away animation while there is this second animation.

To avoid this problem create another Animation command for the second animation with the **FAN_DISABLE_PUSH_AWAY** flag.

The trick is that the other animation could never be performed but uses a non existing state Id as a condition.

Store the animation numbers about when the disabling push away animation for Lara is used.

Another method to solve the problem is to disable the push away animation forever and for all of Lara's animations.

Apply the **CUST_DISABLE_PUSH_AWAY_ANIMATION** customize command.

FAN_constants

64 \$0040: FAN_ENABLE_GRAVITY

Used by the Animation= script command.

This enables the gravity compute for the animation.

When Lara is flying or falling the **TRNG** engine stores a flag to remember to apply the gravity rules.

Some circumstances need to enable gravity for the animation by using this flag.

32 \$0020: FAN_KEEP_NEXT_STATEID

Used by the Animation= script command.

This is a very technical flag.

My suggestion to use it or not is simply to try.

At the start create an Animation command without using this flag, because it is more common not to use it.

If you see that it does not work, try adding the flag to see the result.

Technically use this flag when the animation is for a single execution and then Lara will come back to a standard animation.

Do Not use this flag when the animation has to be performed in a loop like an upward jump and many other fluid animations.

8 \$0008: FAN_KEYS_AS_SCANCODE

Used by the Animation= script command.

Add this flag to use the **KeyBoard** Scan Code in the **KEY1_** and **KEY2_** fields.

Normally the value typed in the **KEY1** and **KEY2** fields will be interpreted like a Game command, i.e. mnemonic constants **KEY1_..** and **KEY2_...**

Remark: This behaviour is the opposite from the first version.
In previous versions the flag is set to specify game commands.
Now the game commands are the default setting.

FAN_constants

256 \$0100: FAN_PERFORM_TRIGGER_GROUP

Used by the Animation= script command.

This changes the working mode of the Animation command.

When this flag is added the value typed in the first field of the Animation command, i.e. the field named **AnimIndex**, will be seen as an **IdTriggerGroup** to perform when all of the conditions set in the Animation command are true.

This means a custom animation could start by typing in a **Trigger Group** a flip effect to start a specific animation for Lara, adding in the same Trigger Group many other effects and then use an Animation command to start the Trigger Group when the player hits the correct key and all conditions are true.

In many circumstances it is required to perform a custom animation and then some special effect to change something in the game to add new skills to Lara. A way to realize this target is to create a custom animation where a frame is set with a NG AnimCommand to perform these "special effects" (i.e. exported triggers to perform somewhat). When the animation command starts the custom animation, the NG Animation will be performed and it could perform many changes in the game.

Using the **FAN_PERFORM_TRIGGER_GROUP** flag can get the same result in a different way. Create a Trigger Group with many exported triggers to change some situation in the game and then in the same Trigger Group have a flip effect to perform the custom animation.

The difference between the two methods is that, using the Trigger Group method something can be performed before the animation is started.

With the NGAnimCommand in the custom animation these effects can only be done when the custom animation has been started.

Remark: Theoretically you could not add any "perform animation" flip effect in the Trigger Group.

Use the Animation command to perform some operations when the player hits some keystrokes.

Remember to find a way to avoid the Trigger Group from continuously performing until the conditions are true.

The **TRNG** engine uses a method to avoid multiple performing, so disable this Animation/Trigger Group until the key used to perform it is pressed.

It is better if a trigger is placed in the Trigger Group to change some condition to avoid repetitions of the Trigger Group.

FAN_constants

8192 \$2000: FAN_RANDOM

Used by the **Animation= script** command.

Use this to perform different animations in a random way.

Type in the **Extra** field the number of random animations.

For example: To perform one of the following animations: 512, 513 or 514, when all of the conditions in the animation command are true:

Type in the **AnimIndex** field the first animation 512, in the **Extra** field type 3 to inform there is a block of three random animations, starting from animation 512.

When all conditions are true the animation 512 or 513 or 514 will be performed. The target of this feature is to create a simulation of random events in animations for Lara.

For example: A falling animation, create different kinds of falling chosen in a random way.

FAN_constants

4096 \$1000: FAN_SET_ADDEFFECT

Add a particle effect when the animation is performed.

Create an `AddEffect=` script command to specify what particle effect to use.

Then add the current flag **FAN_SET_ADDEFFECT** in the **FAN** field and type the number of the **AddEffect** command in the **Extra** field of the **Animation=** command.

Remark: By default the effect will have an infinite duration.
To override this setting use the correct flag in the `AddEffect=` command to only have the **DurateOfEffect** when some state Id or animation number is on.

1024 \$0400: FAN_SET_BUSY_HANDS

This flag works in a similar way to **FAN_SET_FREE_HANDS** flag but with an opposite target. If this flag is used the animation will be performed and the **TRNG** engine will be informed that Lara's hands are busy and she is not able to grab a wall.

Remark: This flag is rarely used because by default when the animation has the correct **stateId**, for example climbing or monkey swinging the **TRNG** engine will set this flag itself.

1 \$0001: FAN_SET_FREE_HANDS

The current flag **FAN_SET_FREE_HANDS** performs a specific action to inform the **TRNG** engine that Lara's hands are free and she is able to hang, climb and pick up objects or weapons.

Remark: The operation to free hands will only be performed if the special animation has been started.

Only use this flag when a special animation starts from a position where Lara had busy hands, climbing, or hanging and the animation moves her from a wall to do a jump.

In this situation inform the **TRNG** engine she has free hands, otherwise Lara will not be able to grab or climb a wall after the jump.

FAN_constants

512 \$0200: FAN_SET_FREE_HANDS_TEMP

This flag works like the **FAN_SET_FREE_HANDS**.

In this case it only sets the "Free Hands" status.

The previous status when Lara was hanging will be restored at the end of the custom animation. The "free hands" is necessary when a custom animation is performed when Lara holds a weapon or a torch.

When the "Free hands" is missing Lara's arms will not follow the custom animation.

Another difference about the old **FAN_SET_FREE_HANDS** is that the

FAN_SET_FREE_HANDS_TEMP fixes a bug where the "free hands" command did not work when Lara held the Torch.

16384 \$4000: FAN_SET_LARA_PLACE

Used by the Animation= script command.

This **FAN_** flag is be used when the custom animation moves Lara between two different places, such as from "ground" to "water", or vice-versa.

Type the new place where Lara should be at the end of the animation in the **Extra** field.

Type one **PLACE_** constant in the **Extra** field to set the new environment for Lara.

Remark: When an animation is created to move Lara between different places it is possible to add a standard [SetPosition] command to move Lara to a new position and also adjust the pivot position used in accordance with ground, underwater, floating etc.

4 \$0004: FAN_SET_NEUTRAL_STATE_ID

Used by the Animation= script command.

This value forces the **TRNG** engine to change the real **stateId** number of the animation, replacing it with a neutral **stateId**.

Use this **FAN_** value when the special animation is stopped before reaching the last frame, because Lara has problems with collisions or status (water, falling, climbing).

When an animation is performed with a neutral **StateId** (69 number) the **TRNG** engine checks for basic collisions but it does not perform any specific control about what Lara is doing or where she is.

FAN_constants

2 \$0002: FAN_START_FROM_EXTRA_FRAME

Used by the Animation= script command.

To start an animation from some specific frame number add the **FAN_START_FROM_EXTRA_FRAME** flag in the **FAN_** field and type in the **Extra** field the frame number to begin the animation.

FBAR_constants

8 \$0008: FBAR_DRAW_ALWAYS

Used in THE Customize=CUST_BAR command.

This flag will always keep on screen the current bar.

The **FBAR_DRAW_ALWAYS** flag does not work with all bars.

Use it with: **BAR_HEALTH**
BAR_DASH
BAR_AIR
BAR_DAMAGE
BAR_COLD

2 \$0002: FBAR_SHOW_BAR_NAME

Used in the Customize=CUST_BAR command.

This only works for the custom bars (**BAR_CUSTOM1/2/3/4**)

Add this flag to the **FBAR_** field of the customize command and the **TRNG** engine will draw the bar with the text under it.

The texts to use are prefixed: **BAR_CUSTOM1** : Extra NG String with index = 301
BAR_CUSTOM2 : Extra NG String with index = 302
BAR_CUSTOM3 : Extra NG String with index = 303
BAR_CUSTOM4 : Extra NG String with index = 304

For example: To add text for the **BAR_CUSTOM1** add the **FBAR_SHOW_BAR_NAME** flag in the customize command for the **BAR_CUSTOM1** and in the **ExtraNG** list add text: **301: Jumping Power**

The previous index text is removed when you click on the [Add string] button.

FBAR_constants

1 \$0001: FBAR_SOUND_BAR_ANIM

Used in the Customize=CUST_BAR command.

This creates floating colours in the current bar like the audio bar in the Options screen.

When using this flag type in the **Extra** field an **IdColor** (pointing to a ColorRGB= command) to set the mask colour.

The audio bar works in a particular way: There is a floating effect using two colours,
Main Color **(IdColor1)**
Background colour **(IdColor2)**.

The mask colour will be used to paint the side of the bar.

For example: Use the flag with the **BAR_HEALTH** and Lara has 50 % Health Points, the left half of the bar will be coloured with the Mask Color to differentiate the right (empty) half.

The mask colour does not fully paint the bar but it will be added to the current floating effect.

For example: Setting the Mask Color with RGB values 0,0,0 gives no difference between the left and right side of the bar.
Adding 0,0,0 changes nothing.

If the mask colour is White with RGB values 255,255,255 the side of the bar will be white losing the floating effect.

It is complicated to set a good mask colour.

Try a mask colour where the RGB value is not 0 or 255.

For example: A colour like: 128,128,128 the default mask colour if **IGNORE** is used in the **Extra** field), or 63,63,63 (used in tomb4 for the audio bar).

FBAR_constants

4 \$0004: FBAR_USED_FOR_BOAT_FUEL

Used in the Customize=CUST_BAR command.

For the boat OCB 's values: 32 (Fuel management) + 128 (show fuel bar)
it is necessary to set a customize script command to prepare the fuel bar to show on screen.

The fuel bar is always the **BAR_CUSTOM4** bar.

You can type this customize command but with two differences:

Add the **FBAR_USED_FOR_BOAT_FUEL** flag.

In the **Extra** field of customize, type the maximum value for the fuel.
That is the value for the boat full fuel to correspond to a full bar on the screen.

The value to type in the extra field works in this way.

The fuel for the boat is the number of frame ticks for the operating time of the boat.

As the boat engine is running the fuel/time will be decreased by 1 for each frame tick.
When it reaches 0 the boat engine will be switched off and the fuel bar will be empty.

When the boat is running in turbo mode the fuel will be decreased by 2 units.

Frame ticks are 30 per second.

To give a full fuel operating time of 15 minutes type the value $30 \times 60 \times 15 = 27000$

Notes:

The maximum value to type in this field is 65534
That corresponds to about 36 minutes.

In the game this is a huge time so you can reduce it if required.

Remember that the fuel management for the boat uses in hard coded mode,
the local long Delta variable (#0072) to store the current fuel of the boat.

You can use this variable to show on screen the remaining time,
or, converting it to litres to give a representation of the current litres in the boat tank.
You can also use triggers to increase or decrease the current fuel.

FCAM_constants

1 \$0001: FCAM_DISABLE_COMBAT_CAM

Used in the Customize=CUST_CAMERA command.

This disables the combat camera mode and it does not affect the automatic aiming feature.
Lara's arms will point to the furthest enemy.
The Chase camera or other camera mode that was working at the time is kept.

2 \$0002: FCAM_INVISIBLE_LARA_ON_LOOK_CAM

Used in the Customize=CUST_CAMERA command.

This makes Lara invisible when the player is using the Look camera.
By default Lara is semi-transparent in this situation.

FFL_constants

2 \$0002: FFL_ADD_FIRE

Used in the Customize=CUST_FLARE command.

Adds a little flame to the flare.

8 \$0008: FFL_ADD_GLOW_LIGHT

Used in the Customize=CUST_FLARE command.

Adds a coloured light bulb to the flare.

Remark: The glow light has the same colour set in the
Customize=CUST_FLARE command.

4 \$0004: FFL_ADD_SMOKE_TO_SPARKS

Used in the Customize=CUST_FLARE command.

This adds a little smoke when adding sparks to the flare.

This only has an effect when used with the **FFL_ADD_SPARKS** flag.

1 \$0001: FFL_ADD_SPARKS

Used in the Customize=CUST_FLARE command.

Adds emission Sparks to the flare.

16 \$0010: FFL_FLAT_LIGHT

Used in the Customize=CUST_FLARE command.

By default the flare light is blinking.

To have a stable common light add this flag.

FGT_constants

8 \$0008: FGT_DISABLED

Used in the GlobalTrigger= command.

By default a Global Trigger works from the beginning of the current level until it is performed with a flag **FGT_SINGLE_SHOT** to disable it.

The Global Trigger can be disabled at the start of the level and then enabled by using the flip effect **Enable/Disable the Global Trigger...**

8192 \$2000: FGT_HIDE_IN_DEBUG

Used in the GlobalTrigger= command.

This does not change the behaviour of the Global Trigger at runtime but it removes the debugging messages in the debug mode **DGX_LOG_SCRIPT_COMMANDS** diagnostic for the Global Trigger.

The reason to use this is when you are studying another script command and a Global Trigger continues to fill the log with messages you are not interested in.

2 \$0002: FGT_NOT_TRUE

Used in the GlobalTrigger= command.

Use this to invert the Global Trigger condition.

For example: To have a Global Trigger engaged when Lara is NOT poisoned type a Global Trigger command script like:

GlobalTrigger=1, FGT_NOT_TRUE, GT_LARA_POISONED, 0,

Remark: This does not affect the further Condition Trigger Group. In the Trigger Group there is a similar flag to invert the Single Condition Triggers.

FGT_constants

4 \$0004: FGT_PUSHING_COLLISION

Used in the GlobalTrigger= command.

This only works when used with Global Triggers: **GT_COLLIDE_ITEM** or **GT_COLLIDE_SLOT**

The **TRNG** engine computes three types of collision:

- Bound Box Collision:** This collision is easy to compute but not very precise. The **TRNG** engine compares two collision boxes for the current frame for both moveables and gives a positive if the two boxes are overlapped in some 3d space position. This is the default collision used for Global Triggers **GT_COLLIDE_ITEM** or **GT_COLLIDE_SLOT**.
- Pushing item collision:** This collision works like the Bound Box Collision but in this case it is necessary that an object pushes other object to give a positive. Theoretically this collision could be considered more precise than the common Bound Box Collision. If the **FGT_MOVING_COLLISION** flag is added the Global Trigger will use this type of "pushing" collision.
- Mesh on Mesh collision:** This is the most complex and precise computes for collisions: each mesh of the first moveable will be compared (using a sphere mesh) with each mesh of the second moveable to detect a collision. It is performed only in very rare circumstances
- For example:** A collision of Lara with some traps, like blades and swords.

FGT_constants

16 \$0010: FGT_REMOVE_INPUT

Used in the GlobalTrigger= command.

This only works with the: **GT_GAME_KEY1_COMMAND**,
GT_GAME_KEY2_COMMAND,
GT_KEYBOARD_CODE Global Triggers.

Add in the Flags field the **FGT_REMOVE_INPUT** flag.

When the condition is true, the received game/keyboard command will be removed and the **TRNG** engine will not be able to detect it.

The game commands can be filtered or redirected.

The filtering is used when it is discovered that a given game command has been received and a Trigger Group or Global Trigger is added for some further actions in accordance with that command.

The redirection is to remove the effect of that game command and replace them with other effects.

64 \$0040: FGT_REPLACE_MANAGEMENT

Used in the GlobalTrigger= command.

This makes the **TRNG** engine abort the default procedure in the **Tomb4** engine used to manage the event the Global Trigger intercepted.

This should be used when a trigger group is used to launch by the Global Trigger command to manage the event and you do not want the **Tomb4** engine to interfere with the operations.

Currently the only Global Trigger that uses this flag is **GT_SELECTED_INVENTORY_ITEM**

1 \$0001: FGT_SINGLE_SHOT

Used in the GlobalTrigger= command.

This sets the current Global Trigger for a single execution.

When the Global Trigger is executed it will be disabled to avoid further executions.

FGT_constants

32 \$0020: FGT_SINGLE_SHOT_RESUMED

Used in the GlobalTrigger= command.

This is similar to the **FGT_SINGLE_SHOT** flag but with a big difference:

The **FGT_SINGLE_SHOT** performs the Trigger Group when the condition is true for the first time and then is disabled.

The new **FGT_SINGLE_SHOT_RESUMED** will be resumed when the condition results give a new false.

The **FGT_SINGLE_SHOT_RESUMED** works like a multi shot flag but avoiding activating the Trigger Group if the condition is continuously true.

THE DIFFERENT WORKING MODES ARE DESCRIBED BY THIS TABLE:

RESULT OF			
CONDITION	MULTI-SHOT (NO FLAG)	FGT_SINGLE_SHOT	FGT_SINGLE_SHOT_RESUMED
FALSE	No-Run Enabled	No-Run Enabled	No-Run Enabled
TRUE	Run Enabled	Run Enabled	Run Enabled
TRUE	Run Enabled	No-Run Disabled	No-Run Enabled
TRUE	Run Enabled	No-Run Disabled	No-Run Enabled
FALSE	No-Run Enabled	No-Run Disabled	No-Run Enabled
FALSE	No-Run Enabled	No-Run Disabled	No-Run Enabled
TRUE	Run Enabled	No-Run Disabled	Run Enabled
TRUE	Run Enabled	No-Run Disabled	No-Run Enabled

FLI_constants

8 \$0008: FLI_DISTANCE_IN_SECTORS

Used in **LogItem=** command.

When the **FLI_SHOW_DIFFERENCES** flag is set, the distance between Lara and the logged item is drawn by default in game units, where one sector is 1024 game units. To have the distance shown in sectors divide by 1024.

Note: This flag gives rapid information about the distance between Lara and a given static item (also using the **FLI_STATIC_ITEM** flag) to set the **StaticMIP=** script command for that kind of item.

It is used to discover how many sectors of distance will be required to make a given static item invisible.
(to set a value in the **CLimit** with -1 **CStaticSlot** field, to skip its drawing).

Enable the diagnostic using the logitem for that static and then verify at what distance that static will not be visible when the **StaticMip** script command is not in the **script.txt**.

2 \$0002: FLI_SHOW_BOUND

Used in the **LogItem=** command.

This shows the current size and coordinates of the 3d bounding box of the Item added. This information is useful when you are not sure about the size of an item being tested.

1 \$0001: FLI_SHOW_DIFFERENCES

Used in the **LogItem=** command.

Set this to display on screen the difference between Lara and the item of the **LogItem**. The values are similar to those required in the **TestPosition** command so use these values as a reference to type reasonable ranges into the **TestPosition** command.

Remark: The differences for Orienting are the same as used in the **TestPosition** command. The coordinate distances (**X ,Y, Z**) values shown on screen do not consider the relative axis used in the **TestPosition** but will show as absolute differences.

The difference between absolute and relative distance is the orientation of Lara. If Lara and an item are both on the same axis, for example Z axis the absolute difference for Z will be the same as the relative difference.

See the description of the **Test Position command** in the **MNEMONICS**.

FLI_constants

4 \$0004: FLI_STATIC_ITEM

Used in **LogItem=** command.

Add this when a static index is used rather than a moveable item index in the **LogItem** command. The information drawn on the screen will be the same used for moveables but some values about animation, frames etc. will have a useless (null) value.

FMIR_constants

8192 \$2000: FMIR_ADJUST_X

Used in the MirrorEffect= command.

Add this to the index of an animating when the reflex of the item has a bad position. This could happen in the floor mirror when the animating does not have the pivot at the centre of the mesh but in a corner or on a side. This problem happens with doors. When this flag is used the **TRNG** engine tries to analyse the collision box of the item and then adjusts the X coordinate of the reflex.

Note: The X axis is in the South/North direction in the **Tomb Editor**.

16384 \$4000: FMIR_ADJUST_Z

Used in the MirrorEffect= command.

Add this to the index of an animating when the reflex of the item has a bad position. This could happen in the floor mirror when the animating does not have the pivot at the centre of the mesh but in a corner or on a side. This problem happens with doors. When this flag is used the **TRNG** engine tries to analyse the collision box of the item and then adjusts the Z coordinate of the reflex.

Note: The Z axis is in West/East direction in the **Tomb Editor**.

4096 \$1000: FMIR_ALTERNATE_REFLEX

Used in the MirrorEffect= command.

This may be added to the index of an animating when in a first attempt you are not sure about its reflex.

The technical explanation is that there are two different ways to simulate the reflex of an item.

In the first method if **FMIR_ALTERNATE_REFLEX** is bad,

add the **FMIR_ALTERNATE_REFLEX**,

and rebuild the script to see if it is better.

The general rule for animating with a good reflex is to have at least two opposite sides that are specular between them.

FMOV_constants

1792 \$0700: FMOV_ANVIL_GRAVITY

Used in the Parameters=PARAM_MOVE_ITEM script command.

This enables the maximum level of anvil gravity.
The "anvil" name has been chosen as symbol of a very heavy item.
In this logic think of an anvil like weight more than a man.

1280 \$0500: FMOV_APOLLO_GRAVITY

Used in the Parameters=PARAM_MOVE_ITEM script command.

This moves the item up with a slow vertical speed for a short time.
When it reaches the maximum speed it will continue upwards.

The maximum vertical speed with a percentage of variation is typed in the **Speed** field.

See the description of the **FMOV_FROG_JUMP_GRAVITY** to know how to compute the formula to type in that field.

1024 \$0400: FMOV_APPLE_GRAVITY

Used in the Parameters=PARAM_MOVE_ITEM script command.

This is the most common gravity.

Begin slowly and increase faster until a middle vertical speed downward is achieved.
Then increase slowly the middle value. Like an apple falling from the tree.

The middle value for vertical speed with a percentage of variation is typed in the **Speed** field.

See the description of **FMOV_FROG_JUMP_GRAVITY** to know how compute the formula to type in this field.

12288 \$3000: FMOV_CAR_SPEED

Used in the Parameters=PARAM_MOVE_ITEM script command.

This will move the item, beginning from a slow speed that will be increased gradually, until it reaches a maximum speed that will remain constant.

The value typed in the **Speed** field will be used as a percentage to change the preset speed of this effect.

For 100, no change will be performed.

For 200, the speed will double.

For 50, the speed will reduce by half.

FMOV_constants

768 \$0300: FMOV_EXPLOSION_GRAVITY

Used in the Parameters=PARAM_MOVE_ITEM script command.

This will move the item upwards at the start with the highest speed and then it will decrease fast.

The maximum start speed can be changed using a percentage of variation in the **Speed** field.

See the description of **FMOV_FROG_JUMP_GRAVITY** to know how to compute the formula to type in the field.

4096 \$1000: FMOV_EXPLOSION_SPEED

Used in the Parameters=PARAM_MOVE_ITEM script command.

This will give to the item a speed beginning with a maximum value that decreases slowly.

This happens when an explosion throws all items close to it.

Note: Warning there is no check about collisions.

The item will only stop when it reaches the distance set in the **Distance** field.

The value typed in the **Speed** field will be used as a percentage to change the preset speed of the explosion movement.

For 100, no change will be performed.

For 200, the speed will double

For 50, the speed will reduce by half the preset speed.

256 \$0100: FMOV_FROG_JUMP_GRAVITY

Used in the Parameters=PARAM_MOVE_ITEM script command.

This will move the item upwards and then it will fall down, like the jump of a frog.

The shape of the jump should be regular.

That is the ascent part will have the inverse shape in the descent part.

Type a percentage variation of vertical speed in the Speed field using following formula:

PercentageOfChangeVerticalSpeed * 256 + HorizontalSpeed

Remember not to type a value higher than 255.

The value of the **PercentageOfChangeVerticalSpeed** works around the 100 value, that is "no change".

To double the vertical speed, use 200, to half preset vertical speed use 50.

FMOV_constants

8 \$0008: FMOV_HEAVY_ALL

Used in the Parameters=PARAM_MOVE_ITEM command.

Enable all the heavy triggers that an item meets in its path.

Use this feature to create a chain of events in the game.

2 \$0002: FMOV_HEAVY_AT_END

Used in the Parameters=PARAM_MOVE_ITEM command.

Enable a heavy trigger in the final sector of the path.

64 \$0040: FMOV_IGNORE_FLOOR_COLLISION

Used in the Parameters=PARAM_MOVE_ITEM script command.

When the movement affects a new direction or speed or gravity mode (new management from **1.2.2.7 version**).

By default the engine will forbid the item to "sink" into the floor.

This means that when a UP/DOWN direction or a gravity mode that moves the item downward is set, when it will reach the floor it will stop.

With horizontal movement if the item reaches a point on the floor that is higher than its position, the engine will move the item upwards to maintain the distance above the floor.

To allow the item to sink into the floor add the **FMOV_IGNORE_FLOOR_COLLISION** flag and the engine will not perform any control for the floor collision.

1 \$0001: FMOV_INFINITE_LOOP

Used in the Parameters=PARAM_MOVE_ITEM command.

The movement will be repeated in an endless loop.

512 \$0200: FMOV_LEAF_GRAVITY

Used in the Parameters=PARAM_MOVE_ITEM script command.

When the item is far from the floor, it will move down very slowly with no acceleration, like a light leaf moving slowly down.

The only collision checked is for the floor, not for other items or walls or ceiling.

The preset vertical speed is typed as a percentage of variation in Speed field.

See the description of **FMOV_FROG_JUMP_GRAVITY** to know how to compute the formula to type in the field.

FMOV_constants

8192 \$2000: FMOV_MAGNET_SPEED

Used in the Parameters=PARAM_MOVE_ITEM script command.

This will set a non constant horizontal speed for the item.

The item will begin slowly to increase its speed increasing faster.
Like an item attracted from a (far) source.

The value typed in the **Speed** field will be used as a percentage to change the preset speed of this effect.

For 100, no change will be performed.

For 200, the speed will double

For 50, the speed will reduce by half the preset speed.

1536 \$0600: FMOV_MAN_GRAVITY

Used in the Parameters=PARAM_MOVE_ITEM script command.

This will move the item down when it is above the floor.
The vertical speed will be a big man falling in space.

4 \$0004: FMOV_TRIGGERS_ALL

Used in the Parameters=PARAM_MOVE_ITEM command.

Enable all of the common triggers that an item meets in its path.

The "common" triggers are the triggers enabled by Lara.
Use this to work an item like the **Mechanical Scarab** to enable some trap.

FMOV_constants

32 \$0020: FMOV_USE_EXTRA_ACTOR_INDEX

Used in the **Parameters=PARAM_MOVE_ITEM** script command.

This replaces the value in the **IndexItem** field with the index of the current enemy set as a extra actor.

This is an interesting way to use the same **Parameter=PARAM_MOVE_ITEM** to move different enemies.

Just set an **EXTRA_ACTOR** as a different enemy and use the same **PARAM_MOVE_ITEM** to move this new enemy.

162 \$0010: FMOV_USE_LEADING_ACTOR_INDEX

Used in the **Parameters=PARAM_MOVE_ITEM** script command.

This replaces the value in the **IndexItem** field with the index of the current enemy set as a leading actor.

This is an interesting way to use the same **Parameter=PARAM_MOVE_ITEM** to move different enemies.

Just set an **EXTRA_ACTOR** as a different enemy and use the same **PARAM_MOVE_ITEM** to move this new enemy.

128 \$0080: FMOV_WAIT_STAND_ON_FLOOR

Used in the **Parameters=PARAM_MOVE_ITEM** script command.

By default, the movement will be completed when the distance supplied in the **Distance** field has been reached.

When some kind of gravity is set and once the distance is covered the item is still above the floor, falling down.

The stop moving causes a weird effect: it stops in space at some height from the floor.

To avoid this situation add the **FMOV_WAIT_STAND_ON_FLOOR** flag and the movement will be stopped only when the distance has been covered and the item has reached the floor falling down.

FMV_constants

1 \$0001: FMV_FADE_OUT

Used in the Customize=CUST_FMV_CUTSCENE command.

For a Fade Out before the start of the FMV add the **FMV_FADE_OUT** in the **FMV_** field of the Customize=CUST_FMV_CUTSCENE command.

Remark: Fade-Out means the game screen becomes darker to black and after this change the video is played.

4 \$0004: FMV_LONG_BLACK_RESTART

Used in the Customize=CUST_FMV_CUTSCENE command.

This flag is like the **FMV_SHORT_BLACK_RESTART**, so read the description for that flag for more information.

The long version of the black restart gives a black screen for one second after the end of the FMV. There are some reasons to use this long black restart instead of the shorter version. Some changes in the game require time to be engaged.

For example: A rope enabled will have a starting animation, a change of position of Lara will require a moving of the follow-me camera to look at Lara correctly.

If the changes performed after the FMV require some time use the long black screen to set a stable game position.

8 \$0008: FMV_NO_AUDIO_RESTART

Used in the Customize=CUST_FMV_CUTSCENE command.

To stop the restart of the **CD** track at the end of the **FMV**. By default the **TRNG** engine stops the **CD** track before starting the **FMV** and restarts the same **CD** in the same position when the **FMV** is completed.

For example: To change the **CD** track at the end of a **FMV** or load another level. In this situation the **CD** track restarted should be stopped to start a new track. Use the **FMV_NO_AUDIO_RESTART** flag to obtain silence at the end of the **FMV**.

FMV_const

16 \$0010: FMV_PRE_CACHE

Used in the Customize=CUST_FMV_CUTSCENE command.

This flag tries to fast start the first **FMV** played in the current game session. It is used like a PRELOAD flag for images, but in this case the **FMV** will NOT be loaded in memory, but inserted in the Windows cache.

A file (like a FMV file) has to be loaded from a folder. Windows spends time to scan the folder and locate the file. So, to insert the **\FMVs** folder in the windows cache use this flag and the **TRNG** engine will perform a first access to the **FMV's** folder when the current level is loaded.

2 \$0002: FMV_SHORT_BLACK_RESTART

Used in the Customize=CUST_FMV_CUTSCENE command.

When the **FMV** is completed the screen is black for two frames (less than 1/10 of second). In some circumstances it is useful to have a black screen to stop the player seeing the game image on the screen before the **FMV** playback.

This situation could be a problem for a game situation after the **FMV**. Using the **FMV_SHORT_BLACK_RESTART** in accordance with a Global Trigger **GT_FMV_COMPLETED**, perform a trigger to change the game situation after the movie, stopping the player seeing the old game screen.

Remark: In spite of the **GT_FMV_COMPLETED** Global Trigger, if a black restart is omitted the old screen game could be visible for 1/30 of second.

Therefore to modify the game screen after a **FMV** it is necessary to add a **BLACK_RESTART** flag, short or long.

FO_constants

8 \$0008: FO_DEMO_ORGANIZER

Used by the Organizer command.

This dramatically changes the operative mode of the current organizer to work according to the **demo.pak index** set in the Parameter field of the Organizer= script command.

A demo organizer is used to perform triggers in precise frames for a linked **demo.pak** while playing or recording. The aim is to create a cutscene.

The main differences of a demo organizer are:

The **FO_ENABLED** flag is ignored.

The organiser is enabled when the demo whose id is typed in the Parameter field of the **organizer** has been started. It is the demo to start by the organizer.

The **FO_LOOP** flag is ignored.

The **FO_TICK_TIME** flag is ignored

The **demo organizer** always works in tick frames and never seconds.

The **"time"** field of the time/trigger group pair array now works as an absolute frame from beginning of the demo.

For instance for a demo with a length of 100 seconds for 3000 frames, to perform a trigger at the first frame and another at the last frame type the time/trigger array with following values: **0, 1, 3000, 2**

Remember that to find the precise frame in a demo enabling the diagnostic for the demo recording with the script commands.:

Diagnostic= ENABLED

DiagnosticType= IGNORE, EDGX_RECORDING_DEMO

When a demo is played pause it and read the current frame by keeping down the **F8** key.

FO_constants

Remarks:

To launch a common organizer at the same time that a demo is performed could result in a leak. The demo works in the inventory and paused menu, changing its internal frame counter, while the common organizer is frozen.

The frame counter used by the common organizer uses the draw frame cycle.
The demo organizer uses the read-input cycle.

In some circumstances these two cycles have different frame amounts.
So it is not possible to synchronize a common organizer with a demo.

Theoretically the last valid frame index inside a demo is given to the **AmountofFrames – 1**.

So in the above example it should be 2999 and not 3000.

It is also acceptable to use the total amount of frames.

The difference between the **Amount-1** and **Amount** is very little.

The frame in the game is not same as the **Amount-1** frame in the demo-playing mode.

The Amount frame index is the first frame after quitting the current demo.

This difference is meaningful only in the case of condition triggers working on a demo playing status.

FO_constants

1 \$0001: FO_ENABLED

Used in the Organizer= command.

By default the **organizer** is disabled at the start and a flip effect is called to enable it.
To enable the **organizer** from the start of the level add the **FO_ENABLED** flag.

2 \$0002: FO_LOOP

Used in the Organizer= command.

Adding the **FO_LOOP** flag the list of (time + **PerformGroup**) will be performed in an endless loop.

After the last **PerformGroup** has been performed, the **Organizer** will restart from the first **PerformGroup**.

Use this flag to perform a **Perform Group** every "number of seconds".

Remember to use a flip effect to start or stop an **Organizer**.

This feature is useful when you want to perform in a loop mode an **organizer** for a specific section of the game or for some time interval.

4 \$0004: FO_TICK_TIME

Used in the Organizer= command.

Changes the time measurement unit.

By default the time is in seconds.

For more precise units of time **use this flag** in the **FlagsOrganizer** field of the Organizer.
The time will be interpreted as frame ticks.

A **tick frame is 1/30 of second**, so to set a time

type the number of seconds * 30.

For example:	15	=	half second
	300	=	10 seconds

FR_constants

2 \$0002: FR_ADD_DRIPS_TO_LARA

Used in the Customize=CUST_RAIN command.

Set this to Enable the drips on Lara after she has been in the rain.

By default there are no drips when Lara is in the rain but only after she has been in a pool.

4 \$0004: FR_CORRECT_SPRINKLERS

Used in the Customize=CUST_RAIN command.

By default the code for elaboration of the sprinklers when the rain drop touches the ground was a leak.

If the room had an irregular floor that is not flat at the same height, many sprinklers were not visible because they were generated underground. To correct this bug add the **FR_CORRECT_SPRINKLERS** flag.

1 \$0001: FR_PLAY_SFX

Used in the Customize=CUST_RAIN command.

This enables the playing of a sound effect when the rain drops hit the floor. Type the number of the sound effect to play in the **Extra** field of the Customize=CUST_RAIN command.

Remarks: By default there is no sound for Rain.
It is advisable to use a sound sample and set the loop mode.

FRB_constants

16 \$0010: FRB_ALLOW_DRIFT

Used with the Customize=CUST_ROLLING_BOAT command.

When the swinging or pitching of a boat is enabled the drift effect can be enabled.

The drift is when the boat moves by itself from its position in time, sliding in some direction.

This effect is a simulation of real behaviour of the non-moored boats. Take care using this effect because the boat moves from the start position placed in the level map. It could move a distance from the start point in the time necessary for the player to reach it. In this case place some barrier to stop the drift movement from the start point. It is not easy to know or set the direction of the drift because this movement is like a collateral effect of pitching and swinging of the original tr2/3 code.

Changing the value of the pitch, swing speed, and height of the rolling changes the direction of the boat.

Remark: For rubber boats and motor boats there is a native procedure for drift. For the Kayak and further fake boats in animating slots the drift procedure has been created in the **TRNG** engine.

This drift simulation, avoids risks to move the kayak into positions not reachable by Lara.

Use the new kayak to avoid problems by keeping the kayak at least one sector from the walls or beach and avoid shallow water.

Knowing the above features an invisible barrier can be created to stop drift from the start position by having shallow water around the kayak, or walls on at least three sides around the kayak.

FRB_constants

8 \$0008: FRB_PITCHING_HIGH

Used with the Customize=CUST_ROLLING_BOAT command.

Set a high level for the boat pitching. The boat will have a large movement.

4 \$0004: FRB_PITCHING_LOW

Used with the Customize=CUST_ROLLING_BOAT command.

Set a low level for boat pitching. The boat will have very little movement.

0 \$0000: FRB_PITCHING_NORMAL

Used with the Customize=CUST_ROLLING_BOAT command.

Normal level for pitching.

Remark: If a **FRB_PITCHING**_constant is omitted,
the NORMAL will be used by default.

2 \$0002: FRB_SWINGING_HIGH

Used with the Customize=CUST_ROLLING_BOAT command.

Set a high level for the boat swinging. The boat will have large movements.

1 \$0001: FRB_SWINGING_LOW

Used with the Customize=CUST_ROLLING_BOAT command.

Set a low level for the boat swinging. The boat will have very little movement.

0 \$0000: FRB_SWINGING_NORMAL

Used with the Customize=CUST_ROLLING_BOAT command.

Normal level for swinging.

Remark: If a **FRB_PITCHING**_constant is omitted,
the NORMAL will be used by default.

FROT_constant

1 \$0001: FROT_LOOP

Used in the Parameters=PARAM_ROTATE_ITEM command.

Performs an infinite rotation.

When set the Angle of rotation value will be ignored because the item will rotate continuously.

FSB_constants

2 \$0002: FSB_DISABLE_ON_COMBAT

Used in the StandBy= command.

When used the standby will not be enabled while Lara is in combat mode.

That is when she is holding weapons.

The reason to use this flag is to give the player a chance to disable the standby when he does not want it. That is keep the weapons in Lara's hands to disable the StandBy.

1 \$0001: FSB_DISABLE_ON_CRAMPED_SPACE

Used in the StandBy= command.

This should be used when the standby is used in the native mode.

That is there is when there is automatic activation after a time of inactivity.

As it is not known where Lara will be when the standby starts, it could happen that Lara is in a cramped space.

In this situation the rotation of the camera could be difficult.

To avoid this problem set the **FSB_DISABLE_ON_CRAMPED_SPACE** flag and the standby will NOT begin while Lara is near a wall.

Remark: This method only works when the distance set is less than 2048, while for higher values some distant wall could not be correctly computed and it could create problems.

4 \$0004: FSB_EXIT_ON_ATTACK

Used in the StandBy= command.

This will quit the Standby mode when Lara is hurt by some enemies or traps.

8 \$0008: FSB_FLIP_DISTANCE

Used in the StandBy= command.

This influences the matrix and portrait modes to smoothly change the distance.

The distance is increased and decreased with a floating range from

(Distance - 50%) to (Distance + 50%).

FSB_constants

32 \$0020: FSB_FLIP_H_ORIENT

Used in the StandBy= command.

This slides the horizontal orientation of the camera in a range from -45 degrees to +45 degrees.

Remark: This setting could only be used with the portrait mode, since it makes no sense using it with a matrix effect. That is because the matrix changes the horizontal orientation continuously.

64 \$0040: FSB_FLIP_SPEED

Used in the StandBy= command.

This changes the rotation speed in a random way.
The **RotateSpeed** range will be between - 50% to + 50%
It is better to use this setting with the matrix effect.

16 \$0010: FSB_FLIP_V_ANGLE

Used in the StandBy= command.

This slowly changes the vertical angle moving it from a minimum of (**VAngle** - 50%) to a maximum of (**VAngle** + 50%).

128 \$0080: FSB_FREEZE_ENEMIES

Used in the StandBy= command.

This prevents Lara being attacked during the standby mode.
That is the activity of all enemies is stopped.
All of the **BADDIES** are frozen while the StandBy is active.

256 \$0100: FSB_FREEZE_LARA

Used in the StandBy= command.

This stops the player making game input during the Standby mode.
As the player is not able to move Lara to exit the Standby mode this setting should only be used when a Standby is started with the flip effect trigger **Perform <&StandBy mode for (E)seconds**

FSB_constants

1024 \$0400: FSB_IMMEDIATE

Used in the StandBy= command.

Usually the camera will start from the current standard position and will slowly reach the position required by the settings for the StandBy command.

To start the camera in a new position use the **FSB_IMMEDIATE** flag and the camera will move immediately to the new position.

512 \$0200: FSB_OVERLAP_AUDIO

Used in the StandBy= command.

With a valid value set in the **AudioTrack** field,
decide if the Standby audio will replace the current audio track.
Or the new audio will overlap the old audio without stopping it.

For the Standby audio to overlap the old audio track use the **FSB_OVERLAP_AUDIO** flag.

To stop the old audio track before playing the new Standby audio track omit the
FSB_OVERLAP_AUDIO flag.

Remark: Only use the **FSB_OVERLAP_AUDIO** flag when the audio track in the level is environment sounds, wind, sea song etc.
If the audio is music, it is better to stop it before playing the Standby audio because two music sounds at the same time could be chaotic.

FSCA_constants

2 \$0002: FSCA_ENDLESS

Used with the Parameters=PARAM_SCALE_ITEM.

This should only be used with dynamic effects.

Add the **FSCA_ENDLESS** flag to create a continuous inflating and deflating of the item, like a pulse effect.

Once the item has reached the final size, it will go back to the **BeginSizePercentage** and go on, forever.

1 \$0001: FSCA_IMMEDIATE

Used with the Parameters=PARAM_SCALE_ITEM.

Sets an immediate scaling with no dynamic effect.

In this situation the **BeginSizePercentage** and the **PercentageSpeed** fields will be ignored (you can type **IGNORE** for them) and the item will be resized to the size set in the **FinalSizePercentage** field.

4 \$0004: FSCA_ITEMGROUP_INDEX

Used with the Parameters=PARAM_SCALE_ITEM.

Use this and the scaling effect works on a item group instead of a single item.

When using the **FSCA_ITEMGROUP_INDEX** flag, the **ItemIndex** field of the **PARAM_SCALE_ITEM** command is used like an Id for a Group Item present in the same [level] section.

For example: Two script commands in the [Level] section:

ItemGroup= 5, -143,-144,-154,-155 ;remember to have negative indices for static items

Parameters=PARAM_SCALE_ITEM, 1, 5,
FSCA_ITEMGROUP_INDEX+FSCA_IMMEDIATE, IGNORE, 85,
IGNORE

When the trigger performs to engage the above Parameters command all four items with indices: 143, 144, 154, 155, will be resized to 80 % of their original size.

The use of the **FSCA_ITEMGROUP_INDEX** flag is advisable when creating a forest with trees :
one group of trees is 80% of the size of the original
another group of trees is 120 % of the size of the original.

FSCAM_constants

1 \$0001: FSCAM_DISABLE_COMBAT_CAM

Used in the Parameters=PARAM_SET_CAMERA command.

This disables the Combat camera until the **PARAM_SET_CAMERA** is working.

2 \$0002: FSCAM_DISABLE_LOOK_CAM

Used in the Parameters=PARAM_SET_CAMERA command.

This disables the Look camera until the **PARAM_SET_CAMERA** is working.

FSS_constants

16 \$0010: FSS_ANIMATE

Used with the Parameters=PARAM_SHOW_SPRITE script command.

Add the **FSS_ANIMATE** to create an animated sprite where a sequence of sprites will be shown to create the effect of the animation.

Use the **Extra** field to customize the animation.

The number in the **Extra** field is given from the formula:

$$\text{FrameDurate} * 256 + \text{NumberOfSprites}$$

Where:

FrameDurate

This is the number of frame that each frame will remain on the screen.

NumberOfSprites

This is the number of sprites that forms the animation.

For example: For an animation that uses **8 sprites** from the **index = 4** and the animation has the maximum speed, where each sprite remains on the screen for only one frame:

Type **4** in the **SpriteIndex** field,

Type in the **Extra** field: $1 * 256 + 8 = 264$ (\$0108)

Remark: There are two kinds of sequence:

Standard sequence, where the indices grows up to the maximum and then repeats from the minimum index.
Like: 123412341234

For Back sequence, where the index reaches the maximum and then decreases down to the minimum index:
Like: 1234321234321

To use the for-back sequence add the **FSS_ANIMATE_FOR_BACK** flag.

FSS_constants

32 \$0020: FSS_ANIMATE_FOR_BACK

Used with the Parameters=PARAM_SHOW_SPRITE script command.

Used together with the **FSS_ANIMATE** flag to create an animated sprite with a forward - backward sequence,

The sprite index will increase to the maximum and then decrease to the minimum like: 1234321234

Read the **FSS_ANIMATE** flag for more information.

4 \$0004: FSS_CLONE_SPRITE

Used with the Parameters=PARAM_SHOW_SPRITE script command.

This only works with the **FSS_SHOW_SPRITE_GRID** flag.
To show the same sprite many times, place it in a grid and use both flags
FSS_SHOW_SPRITE_GRID + FSS_CLONE_SPRITE.

64 \$0040: FSS_EFFECT_FROM_BOTTOM

Used with the Parameters=PARAM_SHOW_SPRITE script command.

This is an effect to move the sprite from the bottom of the screen to the final position set in the **OriginX**, **OriginY** fields.

In the **Extra** field type the number of frames required to complete the movement.
A small value gives a fast movement.

256 \$0100: FSS_EFFECT_FROM_LEFT

Used with the Parameters=PARAM_SHOW_SPRITE script command.

This is an effect to move the sprite from the left of the screen to the final position set in the **OriginX**, **OriginY** fields.

In the **Extra** Value field type the number of frames required to complete the movement.

FSS_constants

512 \$0200: FSS_EFFECT_FROM_RIGHT

Used with the **Parameters=PARAM_SHOW_SPRITE** script command.

This is an effect to move the sprite from the right of the screen to the final position set in the **OriginX**, **OriginY** fields.

In the **Extra** field type the number of frames required to complete the movement.

128 \$0080: FSS_EFFECT_FROM_TOP

Used with the **Parameters=PARAM_SHOW_SPRITE** script command.

This is an effect to move the sprite from top of the screen to the final position set in the **OriginX**, **OriginY** fields.

In the **Extra** field type the number of frames required to complete the movement.

8 \$0008: FSS_EFFECT_ZOOM

Used with the **Parameters=PARAM_SHOW_SPRITE** script command.

With this effect the sprite is shown with a zoom effect.

At the start the sprite will be drawn at 1/10 of its size,

then in the [**Extra**] frames it will be magnified to reach the size and position chosen in the **OriginX**, **OriginY**, **Width**, **Height** fields.

In the **Extra** field set the number of frames (1/32 of second) required to complete the zoom effect.

A small value makes a fast zoom.

1 \$0001: FSS_SHOW_SPRITE_GRID

Used with the **Parameters=PARAM_SHOW_SPRITE** script command.

Each sprite has as maximum size of 256 x 256 pixels

2 \$0002: FSS_TRANSPARENT

Used with the **Parameters=PARAM_SHOW_SPRITE** script command.

To have a semi-transparent image add the **FSS_TRANSPARENT** flag.

Remember that only transparent sprites can use the add colour feature.

FT_constants

8192 \$2000: FT_BLINK_CHARS

Used in the Parameters=PARAM_PRINT_TEXT command.

1 \$0001: FT_BOTTOM_CENTER

Used in the Parameters=PARAM_PRINT_TEXT command.

6 \$0006: FT_BOTTOM_LEFT

Used in the Parameters=PARAM_PRINT_TEXT command.

7 \$0007: FT_BOTTOM_RIGHT

Used in the Parameters=PARAM_PRINT_TEXT command.

3 \$0003: FT_CENTER_CENTER

Used in the Parameters=PARAM_PRINT_TEXT command.

4096 \$1000: FT_NARROW_CHARS

Used in the Parameters=PARAM_PRINT_TEXT command.

2048 \$0800: FT_SIZE-ATOMIC_CHAR

Used in the Parameters=PARAM_PRINT_TEXT command.

192 \$00C0: FT_SIZE_DOUBLE_CHAR

Used in the Parameters=PARAM_PRINT_TEXT command.

128 \$0080: FT_SIZE_DOUBLE_HEIGHT

Used in the Parameters=PARAM_PRINT_TEXT command.

64 \$0040: FT_SIZE_DOUBLE_WIDTH

Used in the Parameters=PARAM_PRINT_TEXT command.

48 \$0030: FT_SIZE_HALF_CHAR

Used in the Parameters=PARAM_PRINT_TEXT command.

32 \$0020: FT_SIZE_HALF_HEIGHT

Used in the Parameters=PARAM_PRINT_TEXT command.

FT_constants

16 \$0010: FT_SIZE_HALF_WIDTH

Used in the Parameters=PARAM_PRINT_TEXT command.

256 \$0100: FT_SIZE_MICRO_CHAR

Used in the Parameters=PARAM_PRINT_TEXT command.

Different from the other **FT_SIZE_** constants this setting ignores all the shape and pre-set size of the current font and forces characters of the same size: squared and small.
The same font size is used to show digits in the Keypad switch

FT_constants

1024 \$0400: FT_SIZE-NO_BORDER Unused in 1.2.2.7 or later.
worked with an old print for special national characters but now it has no effect.

2 \$0002: FT_TOP_CENTER

Used in the Parameters=PARAM_PRINT_TEXT command.

4 \$0004: FT_TOP_LEFT

Used in the Parameters=PARAM_PRINT_TEXT command.

5 \$0005: FT_TOP_RIGHT

Used in the Parameters=PARAM_PRINT_TEXT command.

10 \$000A: FT_UNDER_LEFT_BARS

Used in the Parameters=PARAM_PRINT_TEXT command.

The text will start from the top left corner of the screen, under the lowest bar (Damage Bar)

11 \$000B: FT_UNDER_RIGHT_BARS

Used in the Parameters=PARAM_PRINT_TEXT command.

The text will start from the top right corner of the screen, under the lowest bar (Cold Bar)

FTYPE_constants

3 \$0003:FTYPE_SETTINGS

Used in the **ImportFile=** command.

This is used with some script commands to have an external binary setting file when the required settings for that command are too big to be hosted by the usual command arguments.

Note: The **FTYPE_SETTINGS** type always require as ImportType the **IMPORT_MEMORY** setting.

2 \$0002: FTYPE_SOUND

Used in the **ImportFile=** command.

Use the **FTYPE_SOUND** for all sound files with extensions .wav, .mp3, .ogg, .aiff, .mp2, .mp1
That is all sound files supported in the new sound engine implemented in **TRNG** that are to be used in the game with the specific flip effect trigger : **Play imported sound with <Id**

1 \$0001: FTYPE_USERFILE

Used in the **ImportFile=** command.

Use the **FTYPE_USERFILE** for each file not recognized by the **TRNG** engine, or each file that will not be used directly by the game engine.

When the **TRNG** engine finds an imported file the **FTYPE_USERFILE** will ignore it and the only operation performed is to save it in the correct folder.

It is not logical to use the import mode **IMPORT_MEMORY** for **FTYPE_USERFILE** files because the player will not be able to see them in the game.

For example: To give the player some **.html** files like an introduction for the adventure, or **.txt** files, or an image to be viewed then use the type **FTYPE_USERFILE**.

GT_constants

31 \$001F: GT_AFTER_RELOADING_VARIABLES

Used in the GlobalTrigger= command.

This condition is true when the player has loaded a save game, or when Lara enters a new level using a Finish Trigger with no ResetHub command.

If some variables have been saved in the memory using the **GT_BEFORE_SAVING_VARIABLES** Global Trigger, they can now be restored.

When the variables have been reloaded the same situation is created in the game prior to saving. In the **Parameter** field type the **level number** after the reloading of the game about when to perform this Global Trigger.

So, if Lara is in level 1 and touches a finish trigger to go to level 2, type "2" to reload from the beginning of the level.

32 \$0020: GT_ALWAYS

Used in the GlobalTrigger= command.

This is a special Global Trigger.

Usually a Global Trigger performs the Trigger Group when a specific Global condition is true in the game. The **GT_ALWAYS** Global Trigger will be performed if it is enabled, frame by frame during the game. **Use this Global Trigger to have a Trigger Group working continuously.**

GT_constants

30 \$001E: GT_BEFORE_SAVING_VARIABLES

Used in the GlobalTrigger= command.

This Global Trigger is enabled before the **TRNG** engine saves the variables in the save game or in the HUB section which is used when Lara steps to a new level with a Finish Trigger.

In the **Parameter** field type the **level number** where this trigger performs.

It is important to set the correct level number because the operation to save and restore could be valid for some levels but not others.

Remember that the store variables are all global variables and they will be changed and reloaded for all levels.

See the tutorial for **TRNG** variables in the zip file: **trng_variables.zip**

For example: If some variables modify a moveable in level 1 it is important to save the value in a specific variable, different from level 2 where this moveable could be missing.

Use this Global Trigger to save the variables previously typed to change the moveables or physics of the game. This operation is necessary only when the "poke" operation in some critical memory is only temporary and it will be lost when the player reloads a save game. In this case use the Global Trigger to save some variable changes and then restore them with another Global Trigger: **GT_AFTER_RELOADING_VARIABLES**

15 \$000F: GT_COLLIDE_CREATURE

Used in the GlobalTrigger= command.

This Global Trigger works like the **GT_COLLIDE_SLOT**

In this case the condition will be true for each moveable that is a "creature" type.

A creature is a moveable following AI rules that is able to move by itself.

This Global Trigger will ignore the value typed in the **Parameter** field.

13 \$000D: GT_COLLIDE_ITEM

Used in the GlobalTrigger= command.

This detects if Lara touches a moveable with an **index** set in the **Parameter** field.

For more information read the description of the **GT_COLLIDE_SLOT** Global Trigger and the **FGT_PUSHING_COLLISION** flag.

GT_constants

14 \$000E: GT_COLLIDE_SLOT

Used in the GlobalTrigger= command.

This Global Trigger works like the **GT_COLLIDE_ITEM**.

In this case type a **slot number** and a specific **item index**.

Use the **GT_COLLIDE_SLOT** to test the collision of Lara with any item for a given slot.

For example: Create this Global Trigger:
GlobalTrigger=1, IGNORE, GT_COLLIDE_SLOT, 41, ...
Since 41 is the slot for the BADDY_1 object, every time Lara collides with any BADDY_1 object the Global Trigger will be engaged.

Remark: When the collision is detected the item of the moveable colliding with Lara will be saved to be used with triggers stored in the Trigger Group.
Use the flag **TGROUP_USE_FOUND_ITEM_INDEX** together with a trigger that works on a moveable.

For example: If the Global Trigger detects a collision with a moveable in slot 41 and this moveable has the index = 132,
this index will be used for each trigger in the Trigger Group where the flag is set for **TGROUP_USE_FOUND_ITEM_INDEX**.

See the description of the **TGROUP_USE_FOUND_ITEM_INDEX** for more information.

This is an example how the **TGROUP_USE_FOUND_ITEM_INDEX** flag works together with the **GT_COLLIDE_SLOT** Global Trigger:

For the following lines in the script.txt file:

```
TriggerGroup = 1, $5000+TGROUPE_FOUND_ITEM_INDEX, 749, $E
GlobalTrigger = 1, IGNORE, GT_COLLIDE_SLOT, 43, IGNORE, 1
```

Lara will kill each **BADDY_2** enemy by touching him.

The number **43** is the slot for **BADDY_2**, the flag **TGROUP_USE_FOUND_ITEM_INDEX** in the Trigger Group command has been used to substitute the original target of that trigger

(it was an Action trigger to kill a crocodile) with the **index** of the moveable found in the Global Trigger **GT_COLLIDE_SLOT**

That is **BADDY_2** was colliding with Lara.

GT_constants

18 \$0012: GT_COLLIDE_STATIC_SLOT

Used in the GlobalTrigger= command.

The condition is true when Lara touches a static of a given static slot kind defined in the Parameter field.

Remarks: Do not confuse the moveable slots with static slots.

To perform a Trigger Group when Lara touches a specific static, it is not necessary to use a Global Trigger.

Add in the OCB of that static the value 2048 and then place a heavy trigger in the sector under that static. When Lara touches the static the heavy trigger will be performed.

38 \$0026: GT_COMPLETED_SCALING_ON_ITEM

Used in the GlobalTrigger= command.

This Global Trigger is enabled when the static or moveable item with the index in the Parameter field of the Global Trigger command has been resized and its rescaling is complete.

Note: This Global Trigger only works for dynamic (but not endless) rescaling. Practically when a dynamic rescaling from percentage to percentage with a given resizing speed is performed.

GT_constants

11 \$000B: GT_CONDITION_GROUP

Used in the GlobalTrigger= command.

If a **GT_** Global Trigger is not wanted to test the condition, but some common CONDITION triggers stored in a Trigger Group command script, use the value **GT_CONDITION_GROUP** for the Global Trigger field.

With this value the **TRNG** engine will only check the conditions in the Trigger Group in the **IdConditionGroup** field.

Remark: **Do Not confuse** the use of the **GT_CONDITION_GROUP** value.
This value cannot be added to other **GT_** Global Trigger values.
This value is only used when you do not want to type any specific Global Trigger but only want to use conditions of the Trigger Group.

If a Global Trigger like **GT_USED_LITTLE_MEDIPACK** for example is used and it is required to add a condition group set a valid **IdTriggerGroup** value in the **IdConditionGroup** field.

The **GT_CONDITION_GROUP** is only used to set a null Global Trigger with a condition Trigger Group because if **IGNORE** is in the Global Trigger field the Global Trigger will never be performed and any **IdConditionGroup** will be ignored.

GT_constants

40 \$0028: GT_CREATED_NEW_ITEM

Used in the GlobalTrigger= command.

This global trigger will be enabled every time in the game when an item has been created in the same slot typed in the Parameter field.

When the global trigger is engaged the index of the new created item (always a moveable) will be stored as found in an item index. You can elaborate by adding to exported triggers the **TGROUP_USE_FOUND_ITEM_INDEX** flag.

Items that can be created are:

FLARE_ITEM (when Lara throws it away)

BURNING_TORCH_ITEM (when Lara throw it away. It is not always "burning")

CLOCKWORK_BEETLE (when Lara places it on the floor)

GRENADE (shot by SAS, from Lara and from enemy jeep)

DARTS (emitted from dart emitter. Warning: if you intercept this item you'll have a huge quantity of items to manage)

CROSSBOW_BOLT (shot by Lara)

new created item with slots:

PISTOLS_ANIM (*)

UZI_ANIM (*)

SIXSHOOTER_ANIM (*)

SHOTGUN_ANIM

CROSSBOW_ANIM

GRENADE_GUN_ANIM

GT_constants

9 \$0009: GT_DAMAGE_BAR_LESS_THAN

Used in the GlobalTrigger= command.

This Global Trigger checks the level for damage only when Lara is in a Damage room.

Type the value to compare in the Parameter field.

This condition will check the value of damage in this way:

When Lara enters the Damage room the value of Damage Bar is 1000.
If she remains in the Damage room the damage level will go down to 0,
then Lara will be damaged or burned.

GT_constants

12 \$000C: GT_DISTANCE_FROM_ITEM

Used in the GlobalTrigger= command.

This Global Trigger verifies the distance between Lara and a moveable.
When the distance is less than that set in the Parameter field the trigger will be engaged.

Remark: The value in the Parameter field is a combination of two values:

The index of the moveable to check and
The distance (in real units) to compare.

Note: From 1.2.2.4 version the formula is different.
The new formula is: **MoveableIndex** + (**RealUnitDistance** * 8192)

RealUnitDistance

This is the units used in the game, instead use the number of sectors or clicks.

For example: One sector = 1024,
One click = 256,

so to check the distance for two sectors the value will be 2048
The formula for the Parameter: **ItemIndex** + 8192 * 2048

say the **ItemIndex** = 307, the result will be: $307 + 8192 * 2048 = 16777523$

The maximum value to use as a distance is 130048, which is 127 sectors.
The distance is computed with trigonometric precision as the distance between the two 3d points **X, Y, Z** for Lara and the **X2, Y2, Z2** for the Item.

If the different height (Y coordinates) is ignored add to the formula the constant value
GTD_IGNORE_HEIGHT

The above example will become: $130048 + \text{GTD_IGNORE_HEIGHT}$

A possible reason to use the **GTD_IGNORE_HEIGHT** mode is when a fine precision on the whole side of an object is required like a column.

In this case, if the **GTD_IGNORE_HEIGHT** value is omitted and the compute encloses the **Y** value, when Lara is above the column she will seem further from it because there is the distance between the **Y** coordinate of Lara and the **Y** coordinate of the column on the base of the floor.

If the **GTD_IGNORE_HEIGHT** value is used the distance will be computed in a planar way.

Remark: When the **GTD_IGNORE_HEIGHT** value is used a conditional Trigger Group is used to verify if Lara is in the correct room or at the correct height limit (vertical condition trigger). This avoids the condition is true when Lara is in a room above or below the item but on the same **X, Z** position in the planar view.

GT_constants

19 \$0013: GT_DISTANCE_FROM_STATIC

Used in the GlobalTrigger= command.

This Global Trigger verifies the distance between Lara and a specific static object. When the distance is less than the value set in the Parameter field the trigger will be engaged. In the Parameter field type the result of the formula for the static index and the distance.

See the formula in the [GT_DISTANCE_FROM_ITEM](#)

301 \$012D:GT_DRIVING_CRANE_QUIT

Used in the GlobalTrigger= command. (Plugin_Cranes)

Stops Lara driving the crane.

When Lara quits the driving of the crane it will engage this global trigger. Remember to type in the Parameter field the index of the crane to detect the end of driving.

Obtained from the [plugin_obj_demo](#) script available on the Paolone website Demo downloads.

300 \$012C:GT_DRIVING_CRANE_START

Used in the GlobalTrigger= command. (Plugin_Cranes)

Enables Lara to drive the crane.

This global trigger detects when the crane driving begins in the game. Type in the Parameter field the index of the Swinging Crane item to detect for the driving mode.

Note: The global trigger is engaged when the first animation of Lara over the Crane Control Panel has just been completed.

Obtained from the [plugin_obj_demo](#) script available on the Paolone website Demo downloads.

GT_constants

29 \$001D: GT_ELEVATOR_STARTS_FROM_FLOOR

Used in the GlobalTrigger= command.

This condition verifies when the given elevator has left the given floor.
Type the elevator index and floor number in the Parameter field.
The mode compacts these two numbers in to a single number.

See the **GT_ELEVATOR_STOPS_AT_FLOOR** Global Trigger

28 \$001C: GT_ELEVATOR_STOPS_AT_FLOOR

Used in the GlobalTrigger= command.

This Global Trigger is enabled when the given elevator has stopped at a given floor.
Use this trigger to open a door in front of the elevator.
Type the elevator index and the floor number in the Parameter field.

To put two numbers in to a single field it is necessary **to compact the two numbers**.
The formula is: **(FloorNumber * 4096) + ElevatorIndex**

For example: To verify when the elevator with **Index = 137**, reaches the first floor
Floornumber= 1,

the compute will be: $(1 * 4096) + 137 = 4233$

Remarks: At the start of the level all elevators will give the
GT_ELEVATOR_STOPS_AT_FLOOR event with the
floor number = 1.
All elevators have to begin with the cage at the first floor.
If a Global Trigger is used to have a door open in front of the floor
where the elevator is it will work from the start of the level.

There is a limitation for both values to set in parameter field:

The elevator index cannot be greater than 4095
The floor number cannot be greater than 15

GT_constants

3 \$0003: GT_ENEMY_KILLED

Used in the GlobalTrigger= command.

When the given moveable is killed the Global Trigger will activate.

Read the index of the moveable in the **Tomb Editor** program when you perform a single mouse click on the object. The index will be shown in a yellow box.

Theoretically you can get a trigger activation when a creature dies with a local (common) trigger Switch. If the enemy is triggered with a switch trigger and then add to this trigger some common trigger to enable doors, enemies etc., when the creature dies the trigger will be activated.

The problem with this method is that you have to cover a big surface with triggers because it is necessary to know where the creature dies.

Using a Global Trigger **GT_ENEMY_KILLED** the trigger will be activated for any position of the enemy.

For this Global Trigger it is strongly suggested to use the flag **FGT_SINGLE_SHOT**, otherwise when the condition becomes true it will be performed 30 times per second forever.

26 \$001A: GT_FMV_COMPLETED

Used in the GlobalTrigger= command.

This Global Trigger is engaged when the **FMV** with the number in the Parameter field is finished.

This Global Trigger is useful when something in the game changes at the end of a video cut scene (FMV).

Use a **FMV** to show some fact or actions about Lara and at the end of the movie something in the game changes: like loading a new level, or enabling a flip map or moving Lara in the level.

In all the above circumstances use this Global Trigger to be sure that the triggers stored in a Trigger Group called by this Global Trigger command is performed in the first frame following the ending of the given **FMV**.

GT_constants

23 \$0017: GT_GAME_KEY1_COMMAND

Used in the GlobalTrigger= command.

The condition is true when the game command (jump, action, look etc.) set in the Parameter field is activated by the Player.

Type in the Parameter field the **KEY1_** game command to detect.

To detect a **KEY2_** command use the twin Global Trigger **GT_GAME_KEY2_COMMAND**.

Remark: This condition is true when the game command has been sent.
This means that the command has not been elaborated by the **TRNG** engine when the condition is true.

For example: If the command is **KEY1_JUMP**, Lara is not jumping yet.
If you want to remove the game command to prevent the game engine receiving it, use the flag **FGT_REMOVE_INPUT**

See the **FGT_REMOVE_INPUT**

24 \$0018: GT_GAME_KEY2_COMMAND

Used in the GlobalTrigger= command.

This Global Trigger works in the same way as the **GT_GAME_KEY1_COMMAND**.
In this case set in the Parameter field a **KEY2_** game command.

See the **GT_GAME_KEY1_COMMAND** for more information.

GT_constants

25 \$0019: GT_KEYBOARD_CODE

Used in the GlobalTrigger= command.

This Global Trigger detects when a player hits a specific key.

Type the Scan Code value in the Parameter field.

All valid Scan Codes are in the **KEYBOARD SCANCODES**

Remark: This command should not be used to detect standard game commands, use the **GT_GAME_KEY1_COMMAND** or **GT_GAME_KEY2_COMMAND** Global Triggers for the target.

Remember that the keystrokes linked with the game commands could be changed in the Options screen by the player.

So you cannot be sure that a given Scan Code is always the same to jump or roll.

Use the **GT_KEYBOARD_CODE** trigger to have a custom hot key to start some extra feature. The scan code chosen should be different from all other keyboard commands that the player could choose in the Options screen.

For example select some unused function keys like **F7** or **F8** to start the operation.

GT_constants

36 \$0024: GT_KEYPAD_REMOVED

Used in the GlobalTrigger= command.

This Global Trigger is the opposite of the **GT_KEYPAD_SHOWED** flag.
When the editable keypad is removed this Global Trigger is enabled.
Use this Global Trigger to remove the text shown by : **GT_KEYPAD_SHOWED**.

35 \$0023: GT_KEYPAD_SHOWED

Used in the GlobalTrigger= command.

This Trigger is enabled when the **Key Pad** with the index in the **Parameter** field is shown.
Use this Global Trigger to show text at the same time as the big editable keypad is shown to describe to the player what the target is.

20 \$0014: GT_LARA_HOLDS_ITEM

Used in the GlobalTrigger= command.

This Global Trigger is true when Lara is holding in her hands the item specified in the Parameter field.

The value set in the Parameter could be a vehicle, pole or rope.
The Parameter field cannot hold a SLOT value but a **HOLD_** constant.

See the **HOLD_** values

5 \$0005: GT_LARA_HP_HIGHER_THAN

Used in the GlobalTrigger= command.

This Global Trigger is enabled when the life value for Lara is higher than the value in the Parameter field.

The valid range for **Lara's vitality** is from 0 to 999

GT_constants

4 \$0004: GT_LARA_HP_LESS_THAN

Used in the GlobalTrigger= command.

This Global Trigger is enabled when the life bar for Lara (Health Points) is less than the value set in the Parameter field.

Remember that the health value for Lara is in the range 0 (she dies) to 1000 (full line and no bar shown in the game).

For example: To reduce some capability of Lara when she has less than 100 Health Points type a Global Trigger:

GlobalTrigger=1, FGT_SINGLE_SHOT, GT_LARA_HP_LESS_THAN, 100, ...

Use a trigger like the one above to remove some capability from Lara.

Remember to set some way to return this capability to Lara when her Health Points recover over the limit (100).

For example use the **GT_LARA_HP_HIGHER_THAN** or
GT_USED_BIG_MEDIPACK or
GT_USED_LITTLE_MEDIPACK.

10 \$000A: GT_LARA_POISONED

Used in the GlobalTrigger= command.

A poisoned condition for Lara.

When Lara is poisoned the Life Bar becomes yellow and the screen will be deformed.

Use this Global Trigger to perform an action, for example show text when Lara is poisoned.

In the Parameter field type the value used for comparison.

The formula used is **IF POISON_IN_LARA** is greater than the Parameter then ... perform Global Trigger.

To enable the Global Trigger for any poison value type 0 (zero) in the parameter field.

If the activation is only when the poison level is higher type a value, like 400 or more.

The Potency of Poison is:

Darts	=	160
Little scorpion	=	512
Big scorpion	=	2048
Start screen Harpy	=	2048

Deforming screen when the poison is higher than 256

Max level for poison = 4096

GT_constants

16 \$0010: GT_LOADED_SAVEGAME

Used in the GlobalTrigger= command.

The condition is true when the player has started the loading of a save game from the Load Game screen.

37 \$0025: GT_NO_ACTION_ON_ITEM

Used in the GlobalTrigger= command.

This Global Trigger is true when the supplied item has no **TRNG** action enabled on it.

Type the **index** of the item to check in the **Parameter** field of the Global Trigger command. The **TRNG** actions are procedures used to turn, move and other time consuming effects on moveables. Many Action triggers found in the Set Trigger Type window begin a progressive action on that moveable. This Global Trigger is useful to discover the exact moment when an item completes its movement, for example play a sound effect, to synchronize another action starting when the first action has been completed.

Remark: This Global Trigger verifies if the item has no action enabled on it.

Use the following flags for this Global Trigger:

FGT_DISABLED flag, to avoid it being engaged at the beginning of the level, since in that moment the item has no action in progress.

FGT_SINGLE_SHOT or **FGT_SINGLE_SHOT_RESUMED** flags, otherwise the Trigger Group linked with this global trigger will be continuously performed after the last progressive action has been completed on the item.

17 \$0011: GT_SAVED_SAVEGAME

Used in the GlobalTrigger= command.

The condition is true when the player has saved the current game in some save game.

GT_constants

2 \$0002: GT_SCREEN_TIMER_REACHED

Used in the GlobalTrigger= command.

Start the Global Trigger when the screen timer reaches the specified number of seconds in the Parameter field.

For example: To start the trigger when the timer screen reaches 240 seconds (4 minutes)
type: "GlobalTrigger=1, IGNORE, GT_SCREEN_TIMER_REACHED, 240, ..."

Remark: For this Global Trigger use the flag **FGT_SINGLE_SHOT**, otherwise when the condition becomes true it will perform 30 times per second.

39 \$0027: GT_SELECTED_INVENTORY_ITEM

Used in the GlobalTrigger= command.

This Global Trigger is similar to **GT_USED_INVENTORY_ITEM** but there are important differences.

The **GT_USED_INVENTORY_ITEM** only works with inventory items where there is no hard coded feature and only when Lara is still in the stand up position.

With the **GT_SELECTED_INVENTORY_ITEM** Global Trigger you detect the use of the item, whose slot had been typed in the Parameter field so that it is the first that the tomb4 engine processes

This fact implies some situations to remember:

- 1) Since there is no check about the current animation of Lara or environment (ground, underwater, floating) it should be you, with some Conditional Trigger Group, to verify if it is possible in that moment to really use the item.

For example: If the item should perform an animation where Lara takes the item in her hand, verify for the right conditions if Lara has free hands and she is in a reasonable stateId to manage this situation.

- 2) The Global trigger will be engaged when there is a hard coded management like for the Binoculars, Medipack, Weapons, or when there is no management but, usually the tomb4 refuses to directly manage that item like the Lasersight item.

To replace the default management of the hard coded item add in the flags of the GlobalTrigger command the **FGT_REPLACE_MANAGEMENT** flag.

See the description of the **FGT_REPLACE_MANAGEMENT** for more information.

GT_constants

27 \$001B: GT_TITLE_SCREEN

Used in the GlobalTrigger= command.

This Global Trigger is only used in the [Title] section.

Use the **GT_TITLE_SCREEN** to discover what screen is shown in the title phase:
main titles, new game, load game, or options.

Type in the Parameter field a **TSCR_** value for the wanted screen type.

33 \$0021: GT_TRNG_G_TIMER_EQUALS

Used in the GlobalTrigger= command.

Enabled when the Global **TRNG timer** has the same value set in the Parameter field.
The value in the Parameter field is in frame ticks, where 1 second = 30 frame ticks.
So to detect when a timer reaches 40 seconds type "1200" , since $40 * 30 = 1200$.

Remark: This Global Trigger only works when the respective timer is running.
This means it is not necessary to use the **FGT_SINGLE_SHOT** or **FGT_SINGLE_SHOT_RESUMED** flags.

To check the variation of the **TRNG timer** performed in a different way from the standard flip effect **Variables. Timer. Start the <& TRNG Timer in (E)Mode** avoid these Global Triggers and use instead the **GT_CONDITION_GROUP** Global Trigger placing in the Condition Trigger Group the Id of a Trigger Group with exported CONDITION triggers to test the content of the variable TIMER.

34 \$0022: GT_TRNG_L_TIMER_EQUALS

Used in the GlobalTrigger= command.

Enabled when the Local **TRNG timer** has the same value set in the Parameter field.

Read the **GT_TRNG_G_TIMER_EQUALS** Global Trigger for more information.

6 \$0006: GT_USED_BIG_MEDIPACK

Used in the GlobalTrigger= command.

Trigger enabled when Lara selects and uses a Big Medipack.

Remark: If Lara has a full life and the player selects a Big Medipack from the inventory the condition will be false.
In this situation the Medipack will not be used.

GT_constants

1 \$0001: GT_USED_INVENTORY_ITEM

Used in the GlobalTrigger= command.

This Global Trigger detects when the player chooses a specific item from the inventory. Specify the slot Id in the Parameter field.

For example: To enable a Global Trigger when a player selects the QUEST_ITEM2 with slotId = 253,
type "GlobalTrigger=1, IGNORE, GT_USED_INVENTORY_ITEM , 253, ...".

Remark: To remove the chosen item from the inventory when the Global Trigger has been engaged insert in the Trigger Group the specific flip effect **Inventory-Item. Remove <& inventory-item from inventory.**

If you do not use this flip effect the selected item will remain in the inventory for further activations.

Please, try to avoid the use of QUEST_ITEM1 for the Global Trigger because QUEST_ITEM1 is used for the Detector activation.

7 \$0007: GT_USED_LITTLE_MEDIPACK

Used in the GlobalTrigger= command.

The Trigger is enabled when Lara selects and uses a Small Medipack.

Remark: If Lara has full life and a player selects a Small Medipack from the inventory the condition will be false.
In this situation the Medipack will not be used.

8 \$0008: GT_USING_BINOCULAR

Used in the GlobalTrigger= command.

The Trigger is enabled when Lara is using the Binoculars.

WARNING: Place the correct value in the Parameter field.

To check the condition for Lara using the Binoculars
type **128** or **\$80** in the Parameter field.

To check the condition for Lara NOT using the Binoculars
type the value **0** (zero) in the Parameter field.

GT_constants

21 \$0015: GT_VSCROLL_COMPLETE

Used in the GlobalTrigger= command.

This Global Trigger is true when the vertical scrolling text of a given string index has been completed.

In the Parameter field type the **index** of the **EXTRA_NG** string used for the vertical scrolling text to monitor.

Remark: The most common use for this Global Trigger is to create the following operation:
Show a long scrolling text on the screen and freeze Lara disabling the keyboard input.

In this situation use the Global Trigger to enable new keyboard input when the scrolling text is complete.

22 \$0016: GT_VSCROLL_LAST_VISIBLE

Used in the GlobalTrigger= command.

This Global Trigger is true when last row of the vertical scroll text is fully visible on the screen.

Different from **GT_VSCROLL_COMPLETE**, with **GT_VSCROLL_LAST_VISIBLE** the condition is true when the vertical scroll is not complete but the last row of text enters the visible area of the screen.

In the Parameter field type the **index** of the **NG string** used for the scrolling operation to check. Use this Global Trigger to create a chain of different scrolling operations with different size and character colour.

For example: Create text with a single row using very big characters.
Use this single row as a title for the following scrolling text with a small text size. In this situation use a **GT_VSCROLL_LAST_VISIBLE** Global Trigger linked to the title scrolling to show the remaining text (other scrolling text) when the title is visible on the screen.

The final effect will be to watch a single long text using different size or colour of character but in reality it will be two different scrolling texts working at the same time.

GTD_constants

1073741824 \$40000000: GTD_IGNORE_HEIGHT

Used in the GlobalTrigger= command.

Used in the Parameter field of the Global Trigger command when the **GT_DISTANCE_FROM_ITEM** or **GT_DISTANCE_FROM_STATIC** Global Triggers are used.

See the description of Global Triggers for more information.

HAIR_constants

3 \$0003: HAIR_ONE_PONYTAIL

Used in the Customize=

Set the hair look for the adult Lara with a single ponytail from her nape.

4 \$0004: HAIR_ONE_TR5_PONYTAIL

Used in the Customize=

Set the hair of the adult Lara of Tomb Raider Chronicle type.

In the Tomb Raider Chronicle the pony tail has its hook in a position different from that of the Tomb Raider The Last Revelation.

Remark: In Raider Chronicles there are three models of Lara:

Young Lara,
Adult Lara1,
Adult Lara2.

The Adult Lara1 has the same joints as Lara of The Last Revelation,
The Adult Lara2 has a different joint for her ponytail.

1 \$0001: HAIR_PAGE_BOY

Used in the Customize=

This hair mode disables any type of plaits or ponytails.

This hair mode disables the floating hair for Lara.

If you set this value cover the hole that Lara has in her neck nape,
changing the mesh of Lara's head and the screaming head.

HAIR_constants

2 \$0002: HAIR_TWO_PLAITS

Used in the Customize=

Set the hair look of Young Lara in Angkor Wat.

The only reason to use this value is to have Young Lara with all weapons.

To do this:

Type in the corresponding [Level] section these two commands:

Customize= CUST_HAIR_TYPE, HAIR_TWO_PLAITS

Customize= CUST_DISABLE_SCREAMING_HEAD

DO NOT type any YoungLara= command, otherwise Lara will have no weapons, we want the **TRNG** engine to believe that Lara is adult Lara.

Start **Wad Tool** and load in the left side the main wad used for the level.

It is important that this level has all slots for the adult Lara.

Now load in the right side the wad Angkor.

Only copy these two slots: **LARA_SKIN** and **LARA_SKIN_JOINTS**

Now the level will have Young Lara with all weapons, flares and crowbar animations working.

HIT_constants

32768 \$8000: HIT_BIKE_EXPLODE

Used with the CUST_BIKE_VS_ENEMIES customization.

In this case the bike explodes and Lara is killed.

Mix this flag with the **HIT_EXPLODE** flag to have a double explosion: the item and the bike.

16384 \$4000: HIT_EXPLODE

Used with the CUST_BIKE_VS_ENEMIES customization.

The enemy hit by the vehicle will explode at first impact.

Only use this when an immortal enemy is killed that has no dying animation.

4096 \$1000: HIT_HURT

Used with the CUST_BIKE_VS_ENEMIES customization.

The enemy hit by the bike is injured.

The amount of damage will depend on the speed of the bike at the moment of the impact.

This does not work with immortal creatures.

1024 \$0400: HIT_KILL

Used with the CUST_BIKE_VS_ENEMIES customization.

The enemy touched by the bike is killed

HIT_constants

2048 \$0800: HIT_PUSH_AWAY

Used with the CUST_BIKE_VS_ENEMIES customization.

The enemy is pushed away from the bike.

When the bike is very fast the enemy will fly for some distance.

The distance and speed of the pushed enemy will depend on many factors: the speed of the bike, the angle of impact and the size of the object.

When the object is bigger it will be moved a shorter distance.

Remarks: This can be used with a moveable that is not an enemy.
For example use it with animations.

Use with Other Special Moveables

Take care about the possible results.

Pickups: Could have trouble picking up items after pushing away because the position will not be at the centre of the sector.

Pushables: If they lose the alignment with the sector grid Lara could not be able to push them in a common way.

Rolling ball: It is activated in the direction of the impact but aligned in the Horizontal directions.
The activation of the Rolling ball with the **HIT_PUSH_AWAY** flag will work with the OCB codes typed in the Rolling Ball.

Another effect that the **HIT_PUSH_AWAY** flag will have on the activated Rolling balls is to injure Lara when it is moving in a collision direction.

8192 \$2000: HIT_WALL

Used with the CUST_BIKE_VS_ENEMIES customization.

The enemy remains unmoved by the impact with the bike.

Use this when the enemy is very big or very powerful.

The bike will crash like it was a wall.

Note: It is important that the enemy does not have fast movements to use this flag.
Otherwise he could enter the collision box of the bike with weird results.

HOLD_constants

16 \$0010: HOLD_ANY_TORCH

Lara is holding a torch. Works for a fired or an unlit torch.

6 \$0006: HOLD_CROSSBOW

Lara is holding the crossbow.

9 \$0009: HOLD_FIRED_TORCH

Lara is holding the fire torch.

7 \$0007: HOLD_FLARE

Lara is holding the flare.

5 \$0005: HOLD_GRENADEGUN

Lara is holding the grenade gun.

10 \$000A: HOLD_JEEP

Lara is driving the jeep.

17 \$0011: HOLD_KAYAK

Lara is rowing the kayak.

13 \$000D: HOLD_MOTOR_BOAT

Lara is driving the motor boat.

HOLD_constants

8 \$0008: HOLD_CUT_TORCH

Lara is holding the cut torch.

1 \$0001: HOLD_PISTOLS

Lara is holding the pistols.

15 \$000F: HOLD_POLE

Lara is holding the climbing pole.

2 \$0002: HOLD_REVOLVER

Lara is holding the revolver.

14 \$000E: HOLD_ROPE

Lara is holding the swinging rope.

12 \$000C: HOLD_RUBBER_BOAT

Lara is rowing the rubber boat.

4 \$0004: HOLD_SHOTGUN

Lara is holding the shotgun.

11 \$000B: HOLD_SIDE CAR

Lara is driving the motor bike.

3 \$0003: HOLD_UZI

Lara is holding the uzi pistols.

HRP_constants

8 \$0008: HRP_DISABLE_LASER_SIGHT

Used in the Customize=CUST_HARPOON command.

Add this to disable the use of the laser sight with the harpoon gun.

4 \$0004: HRP_DOUBLE_AMMO

Used in the Customize=CUST_HARPOON command.

As the harpoon weapon uses the slot for the crossbow by default the harpoon has three different ammo types (normal, poison, explosive).

Set this to have only two ammo types: normal and explosive.

1 \$0001: HRP_NO_SWIM_UNDERWATER

Used in the Customize=CUST_HARPOON command.

Set this and Lara will not be able to swim when she holds the harpoon in her hands.

This setting is more realistic when Lara has busy hands as it is not logical that she is not able to swim underwater.

2 \$0002: HRP_SINGLE_AMMO

Used in the Customize=CUST_HARPOON command.

As the harpoon weapon uses the slot for the crossbow by default the harpoon has three different ammo types (normal, poison, explosive).

Set this to have the classic harpoon gun with a single type of ammo in the inventory.

IF_constants

16384 \$4000: IF_CRYPTIC

Used in the Image= command.

If this is added to the Image command, the corresponding image will be encrypted and its name will be changed from: Image#.bmp to @Image#.bmp

Once the image has been encrypted it will not be possible to load and view the image in a common way. The encrypted image is recognized by the "@" character in front of the name.

Playing the level the TRNG engine will decrypt the image to show it in the game. This encrypting feature has been thought of to stop the "old-fox" player looking at the images in the Pix folder to see secret maps and other useful information while playing the level.

Notes: **Perform a backup of original images into another folder because once they have been encrypted there is no way to decrypt them. Therefore there is no chance to modify them without a copy.**

No decrypting tool has been supplied. When you crypt an image it will always be yours.

The decrypting phase performed in game is faster than the encrypting process. The decrypting will require some time so it is better using cryptic images and the IF_PRELOAD flag.

The decrypting phase will then happen at the launch of the game. In the game when the image is triggered it will have been decrypted and will show faster.

32768 \$8000: IF_EFFECT_CROSS_FADE

Used in the Image= command.

The cross fade is when there is a fade out. That is the screen changes from normal colour to black followed by a fade in. That is the screen changes from black to normal light.

The image is displayed ready in the black phase.

Note: This effect can not be mixed with other effects like zoom or moving from sides. The transparent flag does not work with cross fade because when the image is displayed the tomb raider screen is black. The cross fade only works with overlapped images, not with pop up images.

IF_constants

128 \$0080: IF_EFFECT_FROM_BOTTOM

Used in the **Image=** command.

This effect is similar to the **IF_EFFECT_FROM_TOP** effect but in this case the image will enter the screen from the bottom.

See the description of the **IF_EFFECT_FROM_TOP** flag for more information.

256 \$0100: IF_EFFECT_FROM_LEFT

Used in the **Image=** command.

This effect is similar to the **IF_EFFECT_FROM_TOP** effect but in this case the image will enter the screen from the left side.

See the description of the **IF_EFFECT_FROM_TOP** flag for more information.

512 \$0200: IF_EFFECT_FROM_RIGHT

Used in the **Image=** command.

This effect is similar to the **IF_EFFECT_FROM_TOP** effect but in this case the image will enter the screen from right side.

See the description of the **IF_EFFECT_FROM_TOP** flag for more information.

64 \$0040: IF_EFFECT_FROM_TOP

Used in the **Image=** command.

This effect moves the image from the top of the screen to reach the final position set in the fields **PosX**, **PosY**, **SizeX** and **SizeY**.

The whole moving effect is performed in the frames set in the **EffectTime** field. Small values in the **EffectTime** field will produce a fast moving image.

Remark: At the start of this effect the image will be outside the screen, so apply this effect for images to show in full screen size.

IF_constants

32 \$0020: IF_EFFECT_ZOOM

Used in the Image= command.

With this effect the image will show with a zoom effect.

At the start the image will be drawn at 1/10 of its size.

Then in the **EffectTime** frames it will be magnified to reach the size and position chosen with the **XPos**, **YPos**, **SizeX** and **SizeY** fields.

In the **EffectTime** field set the number of frames (1/32 of second) required for the zoom effect.

A small **EffectTime** gives a very fast zoom.

A long **EffectTime** gives a slow zoom

Remark: This effect could slow down the game when applied to pop up images.

4 \$0004: IF_FULL_SCREEN

Used in the Image= command.

This setting will show the image resizing to cover the whole game screen.

The fields are used to set the origin and size of the image.

XPosition, **YPosition**, **SizeX**, **SizeX** will be ignored,
so fill them with four **IGNORE** values.

8192 \$2000: IF_LOOP_AUDIO_TRACK

Used in the Image= command.

To enable a background audio track for the image,
use the **IF_PLAY_AUDIO_TRACK** flag.

To play the audio track in a loop (infinite) add the **IF_LOOP_AUDIO_TRACK** flag.

4096 \$1000: IF_OVER_FIXED_CAMERA

Used in the Image= command.

By default the images will be hidden when there is a fixed camera.

To keep the image on the game screen when there is a fixed camera add the **IF_OVER_FIXED_CAMERA** flag.

IF_constants

8 \$0008: IF_OVER_FLYBY

Used in the **Image=** command.

This only works with pop up images.

By default when a Flyby sequence begins the pop up image will be hidden like other special objects (keypad, detector or picked up items).

To display the pop up image over a Flyby sequence add this flag to the **ImageFlags** field.

1024 \$0400: IF_PLAY_AUDIO_TRACK

Used in the **Image=** command.

Add this to start a sound at the same time as viewing the image and type in the **AudioTrack** field a number between 0 and 255 to locate an audio track in the audio folder.

For example: To play the track "034.wav" type 34 in the **AudioTrack** field.

2 \$0002: IF_POP_IMAGE

Used in the **Image=** command.

This will change the working mode for viewing the image.

A pop up image is drawn over the game screen as the game plays.

A non pop up image is drawn over the game screen when the game is paused.

The pop up image is like a TV logo and is usually smaller than the game screen.

The non pop up image is drawn covering the whole game screen.

It stops when the viewing time is complete.

IF_constants

2048 \$0800: IF_PRELOAD

Used in the Image= command.

By default the image is loaded from the disk and then drawn on the screen.
This operation requires some time and the game remains frozen for a short period.

To avoid this the **TRNG** engine preloads the image when it is loading other data for the current level. The loading bar is on the screen.

The image is stored in the RAM memory and displayed immediately.

Remark: As bmp images can be very big (one MB or more) it is advisable not to have a large number of preloaded images for the same level.
Keep within a limit of 15 Mb for preloaded images.
The size of images preloaded in different levels is not cumulative because the **TRNG** engine will discard them when a level is discarded.

16 \$0010: IF_QUIT_ESCAPE

Used in the Image= command.

This only works for NON pop up images.

Set this when an image freezes the game to permit a player to skip the image with the **Escape**/Inventory keystroke.

IF_constants

1 \$0001: IF_TRANSPARENCE

Used in the Image= command.

Add this to apply a transparency effect on the current image.

The transparent zone will be where there are pixels with RGB colour = \$FF00FF (red = 255; green=0; blue=255).

This is the traditional transparent colour for Tomb Raider textures.

For example: A transparent round image over the game screen.
The visible squared corners are filled with normal colour.

Remarks: The transparency effect is slow to create so it is advisable to use it moderately.

When you create transparent images in **PHOTOSHOP** disable the anti-aliasing feature because the line between the solid colour and the transparent colour could be altered with colours similar to the transparent colour.

Only the colour 255, 0, 255 is transparent. (Magenta).

IGNORE_constants

-1 \$FFFF: IGNORE

IMPORT_constants

1 \$0001: IMPORT_MEMORY

Used in the ImportFile= command.

Select the ImportType value **IMPORT_MEMORY**.

The corresponding file will be loaded into the RAM memory and used only from the memory.

The advantage is that the file will start faster because it does not require disc access to load it into memory. This is very important for sound files where the access from the hard disc could create a slow down in frame rate of the game.

A problem is the use of excessive RAM memory. This is a problem if too many big files are loaded using the **IMPORT_MEMORY** mode.

For example: Importing into memory all of the audio files from the audio folder (from 000.wav to 111.wav) will use about 110 Mb of RAM memory. Then adding the memory used for the **TRNG** engine and memory for other level files there is a risk of performance because Windows will be forced to use virtual memory to run the game.

It is obvious that this problem changes according to the quantity of the RAM memory installed on the PC. For a PC with 256 MB of RAM, limit the total amount of memory of imported files to less than 20 Mb. Use memory imported files only when it is strictly necessary. That is when sound files or images are required to be started quickly.

It is better to avoid the **IMPORT_MEMORY**, preferring the **IMPORT_TEMPORARY** import mode.

See the description of **IMPORT_TEMPORARY** value to compare the differences.

Remark: Up to 200 files can be supported in the **IMPORT_MEMORY** mode.

IMPORT_constants

2 \$0002: IMPORT_TEMPORARY

Used in the **Import File=** command.

When a file is imported into the **script.dat** using the import type **IMPORT_TEMPORARY** it is immediately exported at the start of the game.

Exported means that a new file is created with the original name and folder.

For example: To import a file use the following **Import File** command:

```
ImportFile=1, Help\Start.htm , FTYPE_USERFILE, IMPORT_TEMPORARY
```

The contents of the **Start.htm** file in the sub-folder **help** will be imported into the **script.dat** file. When the game starts the **TRNG** engine will create a new sub-folder named **help** and save the file with the name **Start.htm**

The temporary import mode works using the **script.dat** like a simple "container" store and then transfers files to install before the game starts.

Temporary Import: The advantage consists of having a raw installer removing problems with special files like .ogg or images.

The disadvantage is that there is a lower speed in starting these imported files because they will be loaded from the hard disk before being used in the game.

There is no limit for the number or the size of imported files in **IMPORT_TEMPORARY** mode.

JOINT_constants

7 \$0007: JOINT_ABDOMEN

Used in the AddEffect= command.

Abdomen joint.

3 \$0003: JOINT_LEFT_ANCKLE

Used in the AddEffect= command.

Left ankle joint.

10 \$000A: JOINT_LEFT_ELBOW

Used in the AddEffect= command.

Left elbow joint.

2 \$0002: JOINT_LEFT_KNEE

Used in the AddEffect= command.

Left knee joint.

9 \$0009: JOINT_LEFT_SHOULDER

Used in the AddEffect= command.

Left shoulder joint.

1 \$0001: JOINT_LEFT_THIGH

Used in the AddEffect= command.

Left thigh joint.

11 \$000B: JOINT_LEFT_WRIST

Used in the AddEffect= command.

Left wrist joint.

8 \$0008: JOINT_NECK

Used in the AddEffect= command.

Neck joint.

JOINT_constants

0 \$0000: JOINT_PUBIS

Used in the AddEffect= command.

Pubis joint.

6 \$0006: JOINT_RIGHT_ANCKLE

Used in the AddEffect= command.

Right ankle joint.

13 \$000D: JOINT_RIGHT_ELBOW

Used in the AddEffect= command.

Right elbow joint.

5 \$0005: JOINT_RIGHT_KNEE

Used in the AddEffect= command.

Right knee joint.

12 \$000C: JOINT_RIGHT_SHOULDER

Used in the AddEffect= command.

Right shoulder joint.

4 \$0004: JOINT_RIGHT_THIGH

Used in the AddEffect= command.

Right thigh joint.

14 \$000E: JOINT_RIGHT_WRIST

Used in the AddEffect= command.

Right wrist joint.

0 \$0000: JOINT_SINGLE_MESH

Used in the AddEffect= command.

Single mesh joint.

KEY_constants

64 \$0040: KEY1_ACTION

Used in the Animation= command.

Action key pressed. (**Control key**)

2 \$0002: KEY1_DOWN

Used in the Animation= command.

Down Arrow key pressed.

32 \$0020: KEY1_DRAW_WEAPON

Used in the Animation= command.

Draw weapon key pressed. (Space bar).

16 \$0010: KEY1_JUMP

Used in the Animation= command.

Jump key pressed. (**Alt key**).

4 \$0004: KEY1_LEFT

Used in the Animation= command.

Left Arrow key pressed.

512 \$0200: KEY1_LOOK

Used in the Animation= command.

Look key pressed. (**0 key numeric pad.**)

32768 \$8000: KEY1_RELEASED

Used in the Animation= command.

8 \$0008: KEY1_RIGHT

Used in the Animation= command.

Right Arrow key pressed.

KEY_constants

4096 \$1000: KEY1_ROLL

Used in the Animation= command.

Roll key pressed. (**End key**.)

1 \$0001: KEY1_UP

Used in the Animation= command.

Up Arrow key pressed.

128 \$0080: KEY1_WALK

Used in the Animation= command.

Walk keys pressed. (**Shift key** and **Up Arrow key**.)

16384 \$4000: KEY2_DASH

Used in the Animation= command.

Sprint key pressed. (**/ key**)

8192 \$2000: KEY2_DUCK

Used in the Animation= command.

Crouch down key pressed. (**> key**.)

8 \$0008: KEY2_USE_FLAREKEY1_ACTION

Used in the Animation= command.

Use a flare key pressed. (**? key**).

KLH_constant

-2 \$FFFE: KLH_ALL_LEVELS

Used in the Customize=CUST_KEEP_LARA_HP command.

Use this to force keeping the current **HP** value for all levels where Lara could jump from the current level.

See the **CUST_KEEP_LARA_HP** constant for more information.

LDF constants

512 \$0200: LDF PLAY TRACK

Used in the Diary= command.

Use this for new background music for the Diary.

Choose the audio track in the audio folder in the range 0 to 255.

For example: To have as background music the audio track 104.wav
or 104.mp3
or 104.ogg

type in the **DiarySoundFlags** of the Diary command this text:

LDF PLAY TRACK + 104

Remark: The **LDF_PLAY_TRACK** flag does not stop the game audio track but overlaps the new audio track for the Diary.

It could be useful when the **LDF_PLAY_TRACK** flag is used to add the **LDF_SILENT** flag to stop the game audio track:

LDF PLAY TRACK + 104 + LDF SILENT

Remark: When the diary quits, the previous audio track for the game is restored

256 \$0100: LDF SILENT

Used in the Diary= command.

Set this when Lara's Diary is shown. The background audio track for the game is suspended.

This setting is useful when custom audio tracks are used for single pages of Lara's Diary.

Remark: When the Diary quits, the previous audio track for the game is restored.

1024 \$0400: LDF SOUND EFFECTS

Used in the Diary= command.

This is used to enable some sound effect in the Diary viewing.

The sound effects are: Click on page changing (71: GENERIC_SWOOSH)
 Sound for starting zoom phase (356: LIGHT_BEAM_JOBY)
 Sound for non valid change of page (2: LARA_NO)

Remark: To use sounds remember to add the sound 356 LIGHT_BEAM_JOB_Y into the wad2. (xml file). The two other sounds should always be present.

LDF_constants

8192 \$2000: LDF_TRANSPARENT_BKG

Used in the Diary= command.

This informs the **TRNG** engine that the background images are transparent zones.

The transparent colour is the standard one used in tomb4: \$FF00FF(red = 255; green=0; blue=255)

Use this setting if the Diary is not a rectangular shape.

For example: to have a round shape place transparent colour in the four corners of the screen.

Remarks: This setting has no effect on pop up images shown in small frames on a page.
The pop up images are always shown supporting the transparent colour.

The **LDF_TRANSPARENT_BKG** flag does not work correctly with the **PL_FIX_WIDE_SCREEN** flag for the page layout.

When there is a wide-screen the background image is stretched in the transparent zones.

If the **LDF_TRANSPARENT_BKG** is used with the **LDF_ZOOM_START** the image shown in the zoom effect will have no text and no pop-up images.

Practically the **TRNG** engine will only zoom the first background image for the Diary and when the zoom effect is complete text and pop up images are added.

To set a transparent zone in the background of the Diary it is necessary that all (further) background images have the transparency in the same zones.
Otherwise the transparency zone from the previous page is shown on the screen.

2048 \$0800: LDF_ZOOM_START

Used in the Diary= command.

Starts the Diary image with a zoom effect.

LGTN_constants

4 \$0004: LGTN_ADD_GLOVE_LIGHT

Used with the Parameters=PARAM_LIGHTNING command.

There is a glove light around the target position when the lightning is working.

8 \$0008: LGTN_EXPLODE_TARGET

Used with the Parameters=PARAM_LIGHTNING command.

Use this for a target position set using a moveable index or a static item.
It can be destroyed with an explosion.

16 \$0010: LGTN_FIRE_LARA

Used with the Parameters=PARAM_LIGHTNING command.

To burn Lara use this for a target position using a Lara index.

Note: The fire will kill Lara in a few seconds if the **LGTN_KILL_TARGET** is also used. If it is missing Lara will be damaged and have a chance to survive by diving into water.

2 \$0002: LGTN_FLASH_SCREEN

Used with the Parameters=PARAM_LIGHTNING command.

Add this for a flash in the level while the lightning is running.

128 \$0080: LGTN_GLOBAL_SOUND

Used with the Parameters=PARAM_LIGHTNING command.

By default, if this is omitted the sound will be local.
That is it is played at the target position. When Lara is far away the sound will be faint.

The global sound will always have the same (maximum) volume at any point in the level.
Only add the **LGTN_GLOBAL_SOUND** for a random target from the sky (lightning).

Do Not use it for the electric conductor.

16384 \$4000: LGTN_EARTHQUAKE

Used with the Parameters=PARAM_LIGHTNING command.

This adds a short earth quake effect when Lara is close to the target position of the lightning.
The distance to enable the earthquake is about 6 sectors or less.

LGTN_constants

8192 \$2000: LGTN_INCLINED_RANDOM

Used with the Parameters=PARAM_LIGHTNING command.

Use **IGNORE** as a source position and the **TRNG** chooses a random position over the target position.

By default **TRNG** tries to set a source point almost perpendicular to the floor and the target item.

To see inclined lightning in a random way add the **LGTN_INCLINED_RANDOM** flag.

Note: Only use this when the target item is far from a wall and another obstacle. Otherwise the lightning could pass through the walls.

64 \$0040: LGTN_KILL_TARGET

Used with the Parameters=PARAM_LIGHTNING command.

Use this to kill a moveable.

The target position is set with a moveable **index**.

32 \$0020: LGTN_LARA_SCREAM

Used with the Parameters=PARAM_LIGHTNING command.

This only works if the target item is Lara.

There are two kinds of scream: **the long scream**, set the **LGTN_FIRE_LARA**

a simple moan, **LGTN_FIRE_LARA** is missing.

1 \$0001: LGTN_PLAY_SOUND

Used in the PARAM_LIGHTNING command.

This enables the playing of a SFX sound typed in the **SoundEffect** field.

LGTN_constants

4096 \$1000: LGTN_RANDOM_COLOR

Used with the Parameters=PARAM_LIGHTNING command.

The colour of the light will change by the adding of a floating value between 0 and 47, to each gradient (red, green, blue) that is not 0 and less than 208.

Avoid using the 0 and 208 limitations so that the random changes do not modify too dramatically the chosen main colour.

Set to 0 or greater than 208 the colour gradient that is not to be changed.

LOAD_constants

8 \$0008: LOAD_AMMO_TYPE1

Used in the Equipment= command.

For a weapon slot (weapon, not ammo),
add to the **Amount** field the **LOAD_AMMO_TYPE1** flag to set the ammo currently loaded as Normal ammo (ammo type 1)

Remark: **PISTOL**, **REVOLVER** and **UZI** can only use **AMMO_TYPE1**.
For **SHOTGUN** the **AMMO_TYPE1** is the Normal ammo.

Example: To set the **SHOTGUN** in the inventory and load Normal ammo:
Equipment=SHOTGUN_ITEM, 1+LOAD_AMMO_TYPE1

If the ammo type currently loaded in the weapon is omitted the engine will use the default **AMMO_TYPE1** for that weapon.

16 \$0010: LOAD_AMMO_TYPE2

Used in the Equipment= command.

For a weapon slot (weapon, not ammo),
add to the **Amount** field the **LOAD_AMMO_TYPE2** flag to set the ammo currently loaded with **AMMO_TYPE 2**.

The **SHOTGUN**, **CROSSBOW** and **GRENADE_GUN** can use **AMMO_TYPE2**

32 \$0020: LOAD_AMMO_TYPE3

Used in the Equipment= command.

For a weapon slot (weapon, not ammo),
add to the **Amount** field the **LOAD_AMMO_TYPE3** flag to set the ammo currently loaded to **AMMO_TYPE3**.

The **CROSSBOW** and **GRENADE_GUN** can use **AMMO_TYPE3**.

4 \$0004: LOAD_LASERSIGHT

Used in the Equipment= command.

For a weapon slot (weapon, not ammo),
add to the **Amount** field the **LOAD_LASERSIGHT** flag to set that the ammo has loaded the **Laser Sight**.

Note: Only the **REVOLVER** and the **CROSSBOW** can use the **LASERSIGHT**.

MIR_constants

2 \$0002: MIR_CEILING

Used in the MirrorEffect= command.

Mirror on the ceiling.

Remark: To see the reflected Lara on the ceiling use a room with limited height or a fixed camera. Otherwise Lara is not able to rotate her neck enough to see the reflected Lara.

4 \$0004: MIR_EAST_WALL

Used in the MirrorEffect= command.

Mirror on the East wall.

1 \$0001: MIR_FLOOR

Used in the MirrorEffect= command.

Mirror on the floor.

3 \$0003: MIR_INVERSE_WEST

Used in the MirrorEffect= command.

Mirror on West wall but every movement and displacement of Lara or other moveables will be inverted like in Tomb Raider 1 when Lara meets an alien.

6 \$0006: MIR_NORTH_WALL

Used in the MirrorEffect= command.

Mirror on the North wall.

5 \$0005: MIR_SOUTH_WALL

Used in the MirrorEffect= command.

Mirror on South wall.

0 \$0000: MIR_WEST_WALL

Used in the MirrorEffect= command.

Mirror on the West wall of the current room.

This was the default and the only setting in the old tomb4 engine.

MIST_COL_constants

5 \$0005: MIST_COL_AZURE

Used in the AddEffect= command.

Set the Mist colour Azure.

3 \$0003: MIST_COL_BLUE

Used in the AddEffect= command.

Set the Mist colour Blue.

2 \$0002: MIST_COL_GREEN

Used in the AddEffect= command.

Set the Mist colour Green.

6 \$0006: MIST_COL_PURPLE

Used in the AddEffect= command.

Set the Mist colour Purple.

1 \$0001: MIST_COL_RED

Used in the AddEffect= command.

Set the Mist colour Red.

0 \$0000: MIST_COL_WHITE

Used in the AddEffect= command.

Set the Mist colour White.

4 \$0004: MIST_COL_YELLOW

Used in the AddEffect= command.

Set the Mist colour Yellow.

MPS_constants

. \$40000000: MPS_DISABLE

Used with SPC_PAUSE command of Parameters=PARAM_ACTOR_SPEECH command.

This is a special flag that can only be used by adding it to the **SPC_PAUSE** command to force an absolute pause until the current demo reaches the given frame.

See the **SPC_PAUSE** command for more information.

NEF_constants

256 \$0100: NEF_EASY_HEAVY_ENABLING

Used in the Enemy= command.

This gives the current enemy the ability to enable heavy triggers.

By default only some moveables enable heavy triggers and it is necessary to place an **AI_** object in the target sector with the heavy trigger to enable it.

When the enemy moves over the heavy trigger it is activated.

1 \$0001: NEF_EXPLODE

Used in the Enemy= command.

When the enemy has no health vitality it will explode

2 \$0002: NEF_EXPLODE_AFTER

Used in the Enemy= command.

Perform a death animation and at the end it explodes.

1024 \$0400: NEF_HIT_BLOOD

Used in the Enemy= command.

Show Blood when it has been hit.

8 \$0008: NEF_HIT_DEFAULT

Used in the Enemy= command.

Use standard behaviour.

3072 \$0C00: NEF_HIT_FRAGMENTS

Used in the Enemy= command.

Show fragments when it has been hit.

2048 \$0800: NEF_HIT_SMOKE

Used in the Enemy= command.

Show smoke when it has been hit.

NEF_constants

4 \$0004: NEF_NON_TARGET

Used in the Enemy= command.

Lara will be not able to aim at this moveable.

0 \$0000: NEF_NONE

Used in the Enemy= command.

4096 \$1000: NEF_ONLY_EXPLODE

Used in the Enemy= command.

The enemy will become like a skeleton or mummy.

The common ammunition will have no effect.

Explosive ammunition can destroy it.

16384 \$4000: NEF_SAVE_MESH_VISIBILITY

Used in the Enemy= command.

Use the action trigger to get an invisible mesh for a moveable.

This forces the **TRNG** engine to save and restore from a save game the visibility status of each mesh for the moveable.

32 \$0020: NEF_SET_AS_BRIDGE_FLAT

Used in the Enemy= command.

This forces the specified animating to work like a **BRIDGE_FLAT** item.

Place below the **BRIDGE_FLAT** item a dummy trigger and Lara will be able to walk over it.

Remark: It is advisable to use an Animating item with this flag. Some moveables will not work because they have more than a mesh or a pivot for a first frame that is not compatible to work like a bridge item. A good way to avoid these problems is to copy an original **BRIDGE_FLAT** object in an animating slot with **Wad Tool**

Once copied into the original **BRIDGE_FLAT** the object can be modified with **Metasequoia**.

NEF_constants

64 \$0040: NEF_SET_AS_BRIDGE_TILT1

Used in the **Enemy=** command.

This forces the specified animating to work like a **BRIDGE_TILT1** item.
The **TILT1** has a slope with a difference of one click between the opposite ends.

See the description for the **NEF_SET_AS_BRIDGE_FLAT** flag for more information.

128 \$0080: NEF_SET_AS_BRIDGE_TILT2

Used in the **Enemy=** command.

This forces the specified animating to work like a **BRIDGE_TILT2** item.
The **TILT2** has a slope with a difference of two clicks between opposite ends.

See the description for the **NEF_SET_AS_BRIDGE_FLAT** flag for more information.

NEF_constants

16 \$0010: NEF_SET_AS_CREATURE

Used in the Enemy= command.

This is experimental: Used to give to a common moveable some features of creatures

8192 \$2000: NEF_SET_AS_MORTAL

Used in the Enemy= command.

This is used with the slot of a demigod to transform it in to a mortal enemy.

When this is set the enemy can be killed with common ammos.

32768 \$8000: NEF_SET_AS_SEMIGOD

Used in the Enemy= command.

Use this and the enemy will become a semi god and it will not be possible to kill him.

OBJ_constants

2 \$0002: OBJ_MOTOR_BOAT

Define the slot with the motor boat OBJECT.

6 \$0006: OBJ_MOTOR_BOAT-ANIM

Define the slot with Lara ANIMATIONS for the motor boat.

1 \$0001: OBJ_RUBBER_BOAT

Define the slot with rubber boat OBJECT.

5 \$0005: OBJ_RUBBER_BOAT_ANIM

Define the slot with Lara ANIMATIONS for the rubber boat.

OTYPE_constants

16384 \$4000: OTYPE_AI_DATA

Used with the Parameters=PARAM_LIGHTNING command.

Add this value to the **OCB** value stored in the null mesh object used, to inform **TRNG** that is an **AI_DATA** item.

0 \$0000: OTYPE_MOVEABLE

Used with the Parameters=PARAM_LIGHTNING command.

Add this value to a **moveables index** to inform **TRNG** that it is a Moveable item.

8192 \$2000: OTYPE_STATIC

Used with the Parameters=PARAM_LIGHTNING command.

Add this value to a **static index** to inform **TRNG** that it is a Static item.

PARAM_constants

17 \$0011: PARAM_ACTOR_SPEECH

Used with the Parameters= command.

Syntax: Parameters= **PARAM_ACTOR_SPEECH**, SpeechId, SpeechFlags (**SPCF_**), Parameter, FrameRate, SpeechSlot, HeadSlotMesh, FirstMeshIndex, SpeechMeshAmount, CommandArray

The data of the **PARAM_ACTOR_SPEECH** command will be used to change in sequence the head of the current actor (set via action trigger) to simulate talking.

SpeechId

Set a progressive number to identify the current **PARAM_ACTOR_SPEECH** command to choose with the action trigger.

SpeechFlags (**SPCF_**)

Add one or more **SPCF_** flags to customize the talking effect.
You can type **IGNORE** to omit flags.

See the description of the **SPCF_** flags.

Parameter

This field could store an optional value working in accordance with some **SPC_** flag.

FrameRate

FrameRate means the relation for the animation of the head. That is swapping heads to simulate an animated face and the number of frames.

The default **FrameRate** is 5.

Type **IGNORE** in this field to keep the default value: 5 frames for each head type.

Type 6 and each different head will be kept for 6 frames.

To slow down the moving of the mouth increase the **FrameRate**.

To speed up the moving of the mouth reduce the **FrameRate**.

Note: The **FrameRate** will only affect speech commands where the number of frames to keep that head have been set.

PARAM_constants

SpeechSlot

Type the slot name (or number) from where to take the different heads to simulate talking.

All further heads will be taken from following meshes.

Different slots can be used, each of them with only an head mesh (old method), or from a single slot with many head meshes (new method).

HeadSlotMesh

Type the index of the mesh corresponding to the head of the actor.

For Lara it is 14, for Jean Yves it is 18, for Von Croy it is 21 etc.

Note: When the **SPCF_OLD_SPEECH_SLOTS** method is used, the **HeadSlotMesh** index will be the same as the **FirstMeshIndex** field.

For the new method the **FirstMeshIndex** could be any value.

Often it will be 0 for the first speech mesh in the first mesh of the speech slot.

FirstMeshIndex

Type the index of the first mesh for talking in the **SpeechSlot** object.

For the old format **SPCF_OLD_SPEECH_SLOTS** the mesh index for Lara will be 14.

This is the index for the head of Lara in the **LARA_SPEECH_HEAD1/2/3/4** slots.

For the new method the index should be 0 for Lara.

For other actors it will be the index of the first talking head for the actor.

SpeechMeshAmount

Type the number of different heads used to simulate talking.

Remember that this amount is different for different heads.

To simulate expressions (happy, sad, angry, surprised etc.)

The expression heads are used with the direct command **SPC_MESH** and they should not be counted in the **SpeechMeshAmount** value.

For instance: For the common **HEAD_SPEECH_LARA1/2/3/4** type **4** in the **SpeechMeshAmount** field.

CommandArrays

Type a series of different values divided by commas.

Each value is a command and it will have a **SPC_** (Speech Command) value with a (+ sign)

a value to set a relative mesh number or number of frames, according to the speech command used.

The general formula is: **SPC_command** + **Frame***64 + **MeshIndex**

Frame

This is the number of frames to show the current mesh.

Range for frames is 1 to 63.

If this value is not set (0), it will use the **FrameRate** value as the frame duration.

PARAM_constants

MeshIndex

This is the relative index beginning from the first speech mesh for the given slot.
Range for the **MeshIndex** is 0 to 63.

See description of [SPC_](#) to know the right syntax for each command.

PARAM_constants

7 \$0007: PARAM_BIG_NUMBERS

Used in the Parameters= command.

Use to store big numbers to use in some flip effects.

Syntax: Parameters=PARAM_BIG_NUMBERS, Numbers are separated by commas

For technical reasons it is not possible to type in the trigger type window a number greater than 255.

Use the script command with the **PARAM_BIG_NUMBERS** value to store big numbers to use in some flip effects or actions.

The trigger uses these big numbers and they have a description like this:

Value 0 index in Parameters=PARAM_BIG_NUMBERS script command

Value 1 index in Parameters=PARAM_BIG_NUMBERS script command

Value 2 index in Parameters=PARAM_BIG_NUMBERS script command

So to use in the trigger the value 23430 type the script command:

Parameters=PARAM_BIG_NUMBERS, 23430

In the trigger window select the index =0 for the PARAM_BIG_NUMBERS command.

Up to 254 numbers can be stored in the same PARAM_BIG_NUMBERS command.

Choose a number by its index,

where the first number has index= 0, the second = 1, the third = 2 etc.

Remark: Numbers bigger than 65536 or \$FFFF cannot be input.
All numbers used must have positive values.

PARAM_constants

12 \$000C: PARAM_CIRCLE

Used in the Parameters= command.

Used to store the circle data to use with a fragmented trigger.

Syntax: Parameters=PARAM_CIRCLE, IdParamList, xCenter, yCenter, Radius

IdParamList

Define an Id number, different from other Parameters=PARAM_CIRCLE in the same [Level] section.

xCenter and yCenter

In these two fields type the centre of the circle.

Remember that the origin of the current square game sector is (0,0) and it is in the top-left corner.

The coordinates of the game square go from 0 to 1023.

The X axis in the NGLE view is the West to East direction.

The Y axis in the NGLE view is North to South direction.

Tips & Tricks: Place the centre outside the current game sector but the effect will only work inside the current game square. A reason to place the centre outside is when you want to have a very light bend. In this case having a light bend in the south side of the current sector could set the centre of the circle in the sector south of the current sector. Say, at (512, 1600). Then to have a zone in the current sector set a big radius for example 800

Note: Negative numbers can be used for the centre and it is necessary when the centre is placed outside and in a sector west or north of the current game square.

Radius

The radius uses game units.

So a circle centred in the current square sector to touch all sides will have a radius of 512.

PARAM_constants

4 \$0004: PARAM_COLOR_ITEM

Used in the Parameters= command.

Use as the first value in the Parameters script command.

It is used by a trigger to change the colour of a item.

Syntax: Parameters=PARAM_COLOR_ITEM, IdParamList, ColorType (**COLTYPE_**),
ItemIndex, Index1ColorRGB, Index2ColorRGB, SpeedChange

IdParamList

This is a progressive number to identify the "Parameters=PARAM_COLOR_ITEM" command script in the trigger window.

ColorType (**COLTYPE_**)

Choose a **COLTYPE_** constant to set the work mode of this **PARAM_COLOR_ITEM**.

Read the description for **COLTYPE_** constants.

Remark: Only a single **COLTYPE_** constant can be chosen,
two or more values cannot be added.

ItemIndex

The index of the static or moveable that you want change the colour of.

Index1ColorRGB

Type the **IdColor** of the ColorRGB= with the RGB value used as a first colour.

The ColorRGB= have to be in the same [Level] section where the
Parameters=PARAM_COLOR_ITEM command has been placed.

Index2ColorRGB

If the **COLTYPE_** requires two colours, type the **IdColor** of the ColorRGB= command.

SpeedChange

To use a **COLTYPE_** requiring a dynamic effect (like PULSE or SHADE)

type in the **SpeedChange** field the number of the frame for each cycle.

Remember that 30 frames = 1 second.

So for a pulse from a dark colour to a light colour in two seconds type 60.

Remark: The valid range for speed is minimum= 1, maximum= 255 (about 8 seconds)

PARAM_constants

18 \$0012: PARAM_INPUT_BOX

Syntax: Parameters= PARAM_INPUT_BOX, InputBoxId, BckImageId, WFontId, MaxChars, SfxSound, Flags (**RIB_** values), ExtraParam

The data of the **PARAM_INPUT_BOX** parameters will using a flip effect show a background image where the player will be able to type characters letters and or digits on the screen.

At the end of this operation some **TRNG** variables

LastInputNumber and **LastInputText** will host the typed text and when it is only digit text the final number is typed by the user.

When the Input Box is used in "only digits" mode it will work like a 2D graphic Keypad.

Now it is possible to have words in the **password** accepting letters and digits.

InputBoxId

Set a progressive number to identify the current **PARAM_INPUT_BOX** command to choose it with a flip effect trigger.

There is a maximum of 100 **PARAM_INPUT_BOX** for a level with **Id** values up to 999.

BckImageId

Supply a background image for the Input Box feature.

Type the **Id** of an Image= script command with data about the image to use as a background for the input box.

Note: The Input Box feature is a bit different from a common image.
The settings in the Image= script command should follow some rules:

The Image= command has to have

IF_FULL_SCREEN + **IF_QUIT_ESCAPE** flags

The **IF_POP_IMAGE** flag should be avoided as only "overlapped" images work with the Input Box feature where the program waits for user input.

As the image is NOT a pop-up image type in **XPosition**, **YPosition**, **SizeX** and **SizeY** fields. The micro unit values are used to set the position of the text typed by the user and not to set the position of the image.

PARAM_constants

WFontId

This is the **Id** of a WindowsFont= script command to set the type of font and colour used to write on screen what the player types on the keyboard.

Note: Character sets different from western sets are not supported.
The low-level keyboard reading of the Input Box process, reads a standard US/UK QWERTY keyboard and for this reason, non-western characters will not be seen by the Input Box procedure.

MaxChars

This is the maximum number of characters that the user will be able to type on the screen. This maximum length value is important to avoid too many characters typed, ruining the layout of graphic on the screen with the text moving outside the screen borders.

Perform a test to verify how many characters will be visible on the screen according to the font size chosen and the start position of the text rectangle.
Then type a value of **MaxChars** less than the maximum visible number of characters.

SfxSound

This is the Index of the SFX sound effect to play every time the user hits a key on the keyboard. Note that if **IGNORE** is set it does not mean that there will be no sound (silent mode), but only that a preset sound is used for hitting the key: the sfx sound 109 (**MENU_SELECT**).

The switching on or off of the sounds will only be affected by use of the **RIB_SOUND_ON_KEY** flag.

Flags (RIB_ values)

Insert one or more **RIB_** (Read Input Box) flags linked with "+" operator.

If **IGNORE** is in this field it will use the default settings corresponding to the following flags:

RIB_BLINK_CARET+RIB_SOUND_ON_KEY+RIB_ONLY_CAPS

ExtraParam

This field keeps a parameter according to a **RIB_** flag.

See the descriptions of the **RIB_** constants

PARAM_constants

13 \$000D: PARAM_LIGHTNING

Used in the Parameters= command.

Used to store data to shoot a lightning bolt.

Syntax: Parameters=PARAM_LIGHTNING, IdParamList, Lightning flags (**LGTN_**), SourcePosItem, TargetPosItem, IdColorRGB, Intensity, SoundEffect, Size, ParticleDurate, IntervalTime, Alfa, Beta

IdParamList

Define a **Id** number, different from other

Parameters=PARAM_LIGHTNING in the same [Level] section.

Then use this **Id** to pass to the trigger to set the current parameters.

Lightning flags (**LGTN_**) .

Type one or more **LGTN_** flags to customize the lightning effect.

Type **IGNORE** in this field to set no flag.

See the meaning of the **LGTN_** flags.

SourcePosItem

Set a source position for the item and **TRNG** will use the position for the source.

Type **IGNORE** and **TRNG** will use a random position in the sky above the target position as a source position.

To set a precise source position type in this field:

A moveable index + the **OTYPE_MOVEABLE** flag

Note: In reality it can be omitted because missing the other **OTYPE_** flags the moveable type will be used

A null mesh item OCB value (the null mesh items are like the **AI_** objects or the LARA_START_POS item + the **OTYPE_AI_DATA** flag

Remark: To use a **LARA_START_POS** type its OCB value and NOT the index. This means to set in the null mesh object an OCB value different from any other in the level.

A static object **index** + the **OTYPE_STATIC** flag

TargetPosItem

To set a position supply an item and **TRNG** will use the position of that item as a target.

Type in this field: A moveable **index** + the **OTYPE_MOVEABLE** flag

PARAM_constants

Note: In reality the flag can be omitted because missing other **OTYPE_** flags the moveable type will be used as a default.

A null mesh item OCB value (the null mesh items are like the **AI_** objects or the **LARA_START_POS** item) + the **OTYPE_AI_DATA** flag

Remark: To use a **LARA_START_POS**, type its OCB value and NOT the index. This means set in the null mesh object an OCB value different from any other in the level.

To keep different OCB values for the whole level compute the OCB value multiplying the room number by 10 and then use this as the first OCB value for items in that room.

A static object **index** + the **OTYPE_STATIC** flag

IGNORE can also be typed in this field and in this case gives a random lightning. Use **IGNORE** in the **TargetPosItem** field when Lara is in a wide outside landscape. With **IGNORE** the **TRNG** engine will choose a random target placed far away from Lara from 8 to 20 sectors choosing the side where Lara is able to see it.

IdColorRGB

Type the **Id** of a **ColorRGB=** command where the colour to use for the lightning is stored. Type **IGNORE** in this field to use the white colour.

Intensity

Type values that are in the range from 30 to 100.
The intensity increases the light of the lightning and its duration.
Type **IGNORE** in this field and **TRNG** will use the most common value for this setting

SoundEffect

Type the number of the sound effect to play when the lightning is performed.

Note: To supply a sound effect add to the Lightning field the **LGTN_PLAY_SOUND** flag.

Size

This should be the diameter size of the lightning but it does not have a precise reference to the effective size. It was a magnifying factor about the "lightning" sprite used to create this effect. Reasonable values for this field are in the range from 20 to 40.
Type **IGNORE** in this field and **TRNG** will use the most common value for this setting

ParticleDurate

This value is the persistence of particles used to create the lightning. Reasonable values are in the range from 20 to 40.
Type **IGNORE** in this field and **TRNG** will use the most common value for this setting

PARAM_constants

IntervalTime

This field is complicated to explain but it is very important to simulate the lightning conductor. Type **0** (or **IGNORE**) to simulate the sky lightning.

The value in this field is used to compute a pause between a shooting of a lightning and the next. When set in the trigger a long delay time of 30 tick frames (one second) or more. The problem is that lightning uses many particles and when set to show lightning for some time, it will be a big number of lightnings, one for each frame.

If a pause is not set between a lightning and the following one something happens that you do not want to see in your level. To avoid this collapse of the particle system and lightning it is necessary to set a pause that should be about the time used from the particles for the previous lightning to them vanishing in the normal way.

The value in this field will work in the inverse mode with the **ParticleDurate** field. Type the denominator of a fraction where there is 1 as the numerator. This will be the time that the lightning will shoot while the other time will be a pause.

For example: 1/4 means it will shoot once every four times.

A further complication: Only choose for this field powers of 2, like 2, 4, 8, 16, 32 etc.

A reasonable value for this field is 4 or 8, but it depends on the value typed in the **Intensity** field and in the **ParticleDurate** field.

Bigger values for those fields should have bigger **IntervalTime**.

Note: If a huge value (the maximum value is 128), the collapsing problem is solved and a long pause between a lightning and the next one is obtained.

To compute the pause time perform the computation:

IntervalTime / 30, to get the seconds of pause between a shot and another.

For example: Typing 32 gives a lightning every second.

Lightning goes on for about half a second

so the time with no lightning effect on the screen will be shorter.

Alfa

The generic name **Alfa** is because the meaning of this argument is unknown. It always has small values (0,1,3 are the most used).

Experiment to discover what changes modify this argument.

Or type **IGNORE** in this field and it will use the most common value.

PARAM_constants

Beta

The generic name **Beta** is because the meaning of this argument is unknown.

In this case it appears as the intensity of a sharpening, blurring effect of the lightning.

Increasing this value makes it like smoke and increasing it more it becomes transparent until it disappears.

This argument usually has 3 or 5 values.

Type **IGNORE** and the **TRNG** will use the most common value.

PARAM_constants

2 \$0002: PARAM_MOVE_ITEM

Used in the Parameters= command.

Use as a first value in the Parameters script command.

Set the values used to move a moveable or a static using the specific "Move." flip effect.

Syntax: Parameters=PARAM_MOVE_ITEM, IdParamList, Flags (**FMOV_**), IndexItem, Direction (**DIR_**), Distance, Speed, MovingSound, FinalSound

Field descriptions:

IdParamList

This is a progressive number to identify the

Parameters=PARAM_MOVE_ITEM command script in the trigger window.

Type 1, for the first PARAM_MOVE_ITEM command, 2 for second etc.

Flags (**FMOV_**)

Set one or more **FMOV_** values linked with + (plus) sign

See the **FMOV_** constants

Remark: Type **IGNORE** to not use any flag.

IndexItem

This is the index read when an item is clicked in the **Tomb Editor**.

Remark: Use static or moveables and the correct flip effect in accordance with the static or moveable nature of the item.

Remember the static is owned by the room and so avoid moving a static out of its room.

Direction (**DIR_**)

Choose one **DIR_** constant to set the direction of moving.

See the **DIR_** list

Distance

Set the distance of moving. The units are:

1 sector = 1024,
half-sector = 512,
quarter-sector = 256 etc.

Always use multiples of 256 (one click) to avoid trouble.

The maximum value for distance is 64512, corresponding to 63 sectors.

PARAM_constants

Speed

This is the number of units that will be added to the current position to move the item.
The units are the same as the Distance field: 1 click = 256 units.

Remember that this speed will be added 30 times per second,
so it is better not to set big values for speed.
A reasonable speed is in the range from 8 to 64.

Remark: It is advisable to set a speed value that is a perfect multiple of distance,
otherwise there will be an error in the compute for the final distance.

Moving sound

Optional. To play a sound effect in a looped mode while the item is moving type the number
of a sound effect.

The list of sound effects is in **SOUNDS**.

For no sound type **IGNORE** in this field.

Final sound

Optional. To play a sound effect when the item reaches the final position.

Type a number for the sound effect.

The list of sound effects is in the **SOUNDS**.

For No sound type **IGNORE** in this field.

PARAM_constants

5 \$0005: PARAM_PRINT_TEXT

Used in the **Parameters=** command.

Used to store all information used to print text.

Syntax: Parameters=PARAM_PRINT_TEXT, IdPrintText, Color (**CL_**), FontType (**FT_**), BlinkTime, DurateTime, X_Position, Y_Position

This parameter list is required by the new flip effect trigger **Text. Print formatted ... string**

IdPrintText

This is the number chosen in the Set Trigger Window for the **print formatted text** trigger.

Color (**CL_**)

The colour for the text.

See the **CL_** constants.

FontType (**FT_**)

Set two or more **FT_** constants to describe the size or the preassigned position for the text.

Remark: Omit the **FT_** value to set the position in pixel coordinates typing two valid values in fields: **X_Position** and **Y_Position**.
See the description of these fields.

BlinkTime

A numeric value to signal the interval for blinking when the **FT_BLINK_CHARS** flag is set in the **FontType** field.

For the blink time use only power of **2** values : 1, 2, 4, 8, 16, 32, 64, 128 etc.

DurateTime

The number of seconds to show the text on the screen.

To set a forever time type **-1** or **IGNORE** in this field.

Remarks: If forever (**IGNORE**) is set for the duration, remove the string from screen using the flip effect **Text. Print. Remove (&)Extra NG String from screen**

If this parameter data is used for vertical or horizontal scrolling text, the value typed in the **DurateTime** field will be interpreted by the **TRNG** engine as the speed value used for scrolling text.

PARAM_constants

THE VALUES FOR SCROLLING SPEED ARE AS FOLLOWS:

0:	Abs. Normal Speed	(30 fps)
1:	Abs. Slow Speed	(15 fps)
2:	Abs. Very Slow Speed	(10 fps)
3:	Abs. Fast Speed	(60 fps)
4:	Abs. Very Fast Speed	(90 fps)
5:	Prop. Normal Speed	(30 fps)
6:	Prop. Slow Speed	(15 fps)
7:	Prop. Very Slow Speed	(10 fps)
8:	Prop. Fast Speed	(60 fps)
9:	Prop. Very Fast Speed	(90 fps)

Set **IGNORE** in this field (used by scrolling text) to use the default value **0** i.e. "Abs. Normal Speed (30 fps)"

X_Position and Y_Position

Set two valid values in these two fields. More **FT_** values to set the position will be ignored. To use a **FT_** prefixed position type **IGNORE** in the **X_Position** and **Y_Position** fields. The values typed are in pixels because the resolution in the game is unknown. (each player could set a different resolution).

Set the coordinates as if the screen is 1024x768 pixels resolution. Then if the game has a different resolution the **TRNG** engine will change the position proportionally in accordance with the real resolution.

For example: Set the **Y_Position** value 384 (i.e. $768 / 2$) the text will be shown in the vertical half on the screen.

Remark: The **Y_Position** is not the top pixel of the character but the baseline character.

This means that the **Y_Position** for the 0 value of the character will not be visible.

PARAM_constants

11 \$000B: PARAM_QUADRILATERAL

Used in the Parameters= command.

Used to store the four vertices to define a quadrilateral to use with a fragmented trigger.

Syntax: Parameters=PARAM_QUADRILATERAL, IdParamList, Xa, Ya, Xb, Yb, Xc, Yc, Xd, Yd

A quadrilateral could be a rectangle, a square but also a trapezium or a rhombus.
As it has four sides it is a quadrilateral.

It is not important what first point is used for the A(x,y) vertex,
It is necessary to type the vertices following the perimeter in a clockwise order.

Define a quadrilateral with four 2D vertices.

Each vertex is a point of a game square of 1024x1024 units.

The origin is the top-left corner (North-West corner in the planar view).

The X axis moves from 0 (West side of the square) to 1023 (East side of the square).

The Y axis moves from 0 (North side of the square) to 1023 (South side of the square).

IdParamList

Define an **Id** number, different from other

Parameters=PARAM_QUADRILATERAL in the same [Level] section.

Use the **Id** number to use the data with the fragmented trigger for the quadrilateral.

Xa, Ya

Define the A(x,y) vertex.

Xb, Yb

Define the B(x,y) vertex.

Xc, Yc

Define the C(x,y) vertex.

Xd, Yd

Define the D(x,y) vertex.

PARAM_constants

15 \$000F: PARAM_RECT

Used in the **Parameters=** command.

Syntax: Parameters=PARAM_RECT, RectId, XOrigin, YOrigin, Width, Height, ForeColor, BackColor

This parameter could be used to store information about the screen rectangle.

RectId

Id to identify the **PARAM_RECT** parameters.

Use this value to reference the rectangle in the script commands or triggers that require it.

XOrigin

This is the **Xorigin** of the rectangle. It is the left side of the rectangle.

YOrigin

This is the **Yorigin** of the rectangle. It is the top side of the rectangle.

Width

This is the **Xsize**, the width of the rectangle.

Height

This is the **Ysize**, the height of the rectangle.

ForeColor

Optional field. Where it is required type the **Id** of a ColorRGB command to set the foreground colour of the rectangle.

If it is not foreseen to supply a Foreground colour type **IGNORE** in this field.

Note: Usually the foreground colour is the frame that bounds the rectangle

BackColor

Optional field. Where it is required type the **Id** of a ColorRGB command to set the background colour of the rectangle.

If it is not foreseen to supply a Background colour type **IGNORE** in this field.

Note: Usually the Background colour is the inside (wider) zone of the rectangle.

PARAM_constants

3 \$0003: PARAM_ROTATE_ITEM

Used in the Parameters= command.

Used as a first value in the Parameters script command for rotating of items, moveables or statics.

Syntax: Parameters=PARAM_ROTATE_ITEM, IdParamList, FlagsRotation (**FROT_**), ItemIndex, DirHRotation (ROTH_), HRotationAngle, SpeedHRotation, DirVRotation (ROTV_), VRotationAngle, SpeedVRotation, MovingSound, FinalSound

Description of fields:

IdParamList

This is a progressive number to identify the

Parameters=PARAM_ROTATE_ITEM command script in the trigger window.

Type 1, for the first **PARAM_ROTATE_ITEM** command, 2 for second etc.

Use numbers from 1 to 99.

FlagsRotation (**FROT_**)

Type one or more **FROT_** constants linked with + (plus) sign.

See the description for the **FROT_** constants.

ItemIndex

This is the index of the item to rotate.

Read it in the frame that appears when you click on a item in the **Tomb Editor**.

Choose a moveable or static but remember to perform the correct flip effect in accordance with the nature of the static or moveable item.

DirHRotation (**ROTH_**)

To have a horizontal rotation type in this field a **ROTH_** value to set the direction of rotation (clockwise or opposite).

HRotationAngle

Set the wanted horizontal angle rotation.

The unit of measurement for this field is a bit weird but it was used in the **tomb4** engine.

These are the references:

\$2000 = 45 degrees

\$4000 = 90 degrees

\$8000 = 180 degrees

For example: To rotate an object by 90 degrees type the value \$4000 (16384).

Remark: If the **FROT_** flag is set for endless rotation the **HRotationAngle** field will be ignored, as the item will rotate continuously.

PARAM_constants

SpeedHRotation

The speed is a value that will be added to the current horizontal orientation.
The unit of measurement is the same as the horizontal or vertical **RotationAngle**, but in this field type a value smaller than the final angle.

Remember that the value of speed will be added to the current orientation 30 times per second.
It is advisable to use a speed value that is a multiple of the power by 2.
In hexadecimal it is easy to remember good speeds: \$80 , \$100 , \$180, \$200, \$280, \$300 etc.
If these values are not used there is the risk that the speed value is NOT a multiple of the final Rotation Angle and the object may not be positioned correctly.

For a good value of **RotationAngle** and **Speed** divide the Rotation by Speed.

DirVRotation (ROTV_)

Have a vertical rotation **ROTV_** value and set the direction of rotation forwards or backwards.

Remark: Unfortunately statics cannot have a vertical rotation, so a vertical rotation can only be applied to moveable items.

VRotationAngle

Set the vertical rotation angle.
The unit of measurement for this field is used in the **tomb4** engine.

These are the references: \$2000 = 45 degrees
 \$4000 = 90 degrees

For example: To rotate an object by 90 degrees type the value \$4000 (16384).

Remark: If the **FROT_** flag is set for endless rotation the **VRotationAngle** field will be ignored as the item will rotate continuously.

SpeedVRotation

The speed is a value that will be added to the current vertical orientation.
The unit of measurement is the same as the vertical **RotationAngle**,
However type a value smaller than the final angle.

Remember that the value of speed will be added to the current orientation 30 times per second.

MovingSound

Optional field. To assign a sound item while it is rotating type a number for the sound SFX.
For No sound set **IGNORE** in this field.

FinalSound

Optional field. To perform a sound when the rotation is complete type a sound SFX in this field.
For No final sound set **IGNORE** in this field.

PARAM_constants

8 \$0008: PARAM_SCALE_ITEM

Used in the **Parameters=** command.

Used to store data for scaling effects.

Syntax: Parameters=PARAM_SCALE_ITEM, IdScaling, ItemIndex, Flags Scaling (**FSCA_**), BeginSizePercentage, FinalSizePercentage, PercentageSpeed

IdScaling

Used to locate the specific command in the script.

To use the data choose the number in the set Trigger Type window to choose the Parameters with the scaling data.

ItemIndex

Type the index of the item to resize.

This is an index given by performing a right mouse click on the item placed in the map in the **Tomb Editor**.

Read the index in the frame in the rounded parenthesis.

Flags Scaling (**FSCA_**)

This is the flag to customize the scaling effect.

Type **IGNORE** if this is not used.

See the **FSCFA_** constants.

BeginSizePercentage and FinalSizePercentage

The dynamic scaling effect works by dynamically changing the size of an item from one size to another.

The **BeginSizePercentage** is the start size of the item in the game.

The **FinalSizePercentage** is the finish size of the item to complete the effect.

The values are in percentage, where 100 = the original size (in the wad2 file) of the item.

For example: To double the size of the item from the original size use:

BeginSizePercentage = 100 and **FinalSizePercentage** = 200

Remark: Type **IGNORE** in the **BeginSizePercentage** field when using this script command to resize an item to the new size **FinalSizePercentage** field with no dynamic effect.

PercentageSpeed

Add or subtract the value to the **BeginSizePercentage** at each frame to reach the **FinalSizePercentage** value.

Larger values will give a faster dynamic resize effect.

Important: The value is not given in 100th but in 1000th.
That is one unit = 1/1000.
So a very slow speed can be set.

PARAM_constants

6 \$0006: PARAM_SET_CAMERA

Used in the Parameters= command.

Use to store information in the Parameters command to use with the flip effect to change the camera mode.

Syntax: Parameters=PARAM_SET_CAMERA, IdSetCamera, Flags (**FSCAM_**), DistanceCam, VOrientCam, HOrientCam, SpeedCamera

IdSetCamera

This is an identifier to locate the Parameters in the [Level] section using the value in the flip effect trigger to temporary set a new camera mode.

Flags (**FSCAM_**)

Type one or more **FSCAM_** constants to affect the behaviour of the command.

Type **IGNORE** for no flag.

DistanceCam

This is the distance of the camera from Lara.

A reasonable value is +1600.

To keep the default value type **IGNORE** and the standard distance value set for "chase" (follow-me) camera will be used.

For more information about the concept of "cam distance"

See the description of the **CUST_CAMERA**

VOrientCam

This is the degrees of difference computed on the horizontal line (parallel to floor) between the camera and Lara.

Type **IGNORE** to use the **VOrientCam** value set for the "chase" camera.

See the **CUST_CAMERA** description to read more information about the Vertical Orientation field.

HOrientCam

Set the horizontal orientation (facing) of the camera with respect to Lara.

Type **IGNORE** and the **TRNG** the engine will use the **HOrient** value set for the "chase" camera.

See the **CUST_CAMERA** description to read more information about the Horizontal Orientation field.

PARAM_constants

SpeedCamera

This value sets the speed of the camera to pass from a previous camera mode and position to the current new position.

The speed works in an opposite way: **large values = slow movements,**
 small values = fast movement.

Type a small value and the camera will reach the new position faster but the movement could be jerkily and not good to see.

Type a big value and the camera will take more time to reach the new position.
A reasonable value for fast movement is 1, while for slow movement is 10.

Type **IGNORE** and the **TRNG engine** will use the default speed value 10.

PARAM_constants

9 \$0009: PARAM_SHOW_SPRITE

Used in the **Parameters=** command.

Used to store data to show sprites on the screen with the flip effect show sprite.

Syntax: Parameters=PARAM_SHOW_SPRITE, IdParamShowSprite, Flags Show Sprites (**FSS_**), OriginX, OriginY, Width, Height, SlotSprite, SpriteIndex, IdColorRGB, GridX, GridY, Extra Value

IdParamShowSprite

Type a progressive number to distinguish the

Parameters=PARAM_SHOW_SPRITE command from others in same level section.

Use the number to signal the flip effect Trigger to look for the sprite show data.

Flags Show Sprites (**FSS_**)

Type the **FSS_** constant values to customize the feature for showing the sprite operation.

See the **FSS_** constants

OriginX and OriginY

In these two field type the origin of the sprite (top-left corner).

Both values are in micro units.

Width and Height

Set the size that the sprite will have on the game screen.

Type these values in micro units.

Use the [**Get Screen Frames**] tool to choose a rectangle on the screen and then copy both four values in micro units in the **OriginX**, **OriginY**, **Width** and **Height** fields.

Remark: Use a sprite grid (with the **FSS_SHOW_SPRITE_GRID** flag) choose the width and height of a single sprite and not the final size of the sprite grid.

SlotSprite

Type the sprite slot for the sprite to use.

For example: Type the **DEFAULT_SPRITES**, **MISC_SPRITES** or another slot that contain sprites.

SpriteIndex

Type the index of the sprite to use within the given **SlotSprite**.

The first sprite has an index = 0

IdColorRGB

Type in this field the **Id** number of a previous ColorRBG= script command to add colour to the sprite. The add colour feature only works when the transparent feature for the sprite is enabled by adding the **FSS_TRANSPARENT** flag.

For full opaque sprites the **IdColorRBG** will be ignored.

PARAM_constants

GridX and GridY

These fields are only used with the **FSS_SHOW_SPRITE_GRID** flag.

For example: To show an image created with 2 x 1 sprites: type 2 in the **GridX**
type 1 in the **GridY**.

See the description of the **FSS_SHOW_SPRITE_GRID** flag for more information.

If no **FSS_SHOW_SPRITE_GRID** flag is set the **GridX**, **GridY** will be ignored.

Extra

This field is used according to the **FSS_** constant.

See the descriptions of the **FSS_** constants to discover when this field is used.

PARAM_constants

16 \$0010: PARAM_SWAP_ANIMATIONS

Used in the **Parameters=** command.

Syntax: Parameters=PARAM_SWAP_ANIMATIONS, ASwapId, SourceFirstAnim, TargetFirstAnim, NumberOfAnimations

This parameter holds data for flip effect 393.

The **F393** trigger swaps animations from the **SourceFirstAnim** to the **TargetFirstAnim**.

The animation slots swapped will be **NumberOfAnimations**.

Usually another swap is performed for the same set of animations to restore the previous situation.

ASwapId

A progressive id is used to identify the parameter data.

This **id** is chosen in the **F393** trigger to link it to the swapping data for this specific

Parameters=PARAM_SWAP_ANIMATIONS command.

SourceFirstAnim

Type the animation number of the first animation in the group of animation slot to swap.

Remember that it is necessary that both group of animations to swap are in the same slot.

That is a store for the same object type. The indices are zero based. For example to replace all 8 animations type 0 to choose first animation slot.

TargetFirstAnim

Type the number of the first animation of the second group to swap the animations.

Note that nothing changes if the values in the **TargetFirstAnim** field and the **SourceFirstAnim** are inverted. As this is a swapping operation at the end both group animation will still be present in the slot but in a different (inverted) position.

NumberOfAnimations

Type the number of animations to swap between groups.

To swap a single animation type 1, or a bigger number to swap more.

For instance this parameter is wrong:

Parameters=PARAM_SWAP_ANIMATIONS, 0, 4, 6

The **NumberOfAnimations** to swap (6) is bigger than the difference between the first **SourceFirstAnim** (0) and the **TargetFirstAnim** (4).

This will mess up the animations for the slot.

PARAM_constants

10 \$000A: PARAM_TRIANGLE

Used in the Parameters= command.

Used to store the three vertices to define a triangle to use as a fragmented trigger.

Syntax: Parameters=PARAM_TRIANGLE, IdParamList, Xa, Ya, Xb, Yb, Xc, Yc

Define a triangle with three 2D vertices.

Each vertex is a point of a game square of 1024x1024 units.

The origin is the top-left corner (North-West corner in planar view).

The X axis moves from 0 (West side of the square) to 1023 (East side of the square).

The Y axis moves from 0 (North side of the square) to 1023 (South side of the square).

IdParamList

Define an **Id** number, different from other

Parameters=PARAM_TRIANGLE in the same [Level] section.

Use this number to use the data with the fragmented trigger for custom triangles.

Xa, Ya

Define the A(x,y) vertex.

Xb, Yb

Define the B(x,y) vertex.

Xc, Yc

Define the C(x,y) vertex.

Example: To define a triangle corresponding to a North-East corner triangle with a size of 512, use the following command:

Parameters=PARAM_TRIANGLE, IdParamList, 512, 0, 1023, 0, 1023, 512

Using the custom triangles and exporting the condition triggers that use them into a Conditional Trigger Group can cover any shape for the trigger.

To create a detailed shape add different triangles and use the **TGROUP_OR** flag in the Trigger Group command to link the different triangle triggers.

Mix the triangles with other shapes, like circle or grid triggers, linked with the (default)

TGROUP_AND to have a trigger zone where all fragmented triggers are true at the same time.

PARAM_constants

14 \$000E: PARAM_WTEXT

Used to store parameters to print a string with the Windows font.

Syntax: Parameters=PARAM_WTEXT, IdParameter, Flags (**WTF_...**), WindowsFontId, TimeDurate, Left, Top, Right, Bottom

IdParameter

Type an **id** to identify the parameters command to link it to a trigger to print the windows texts.

Flag (**WTF_**)

Type one or more **WTF_** Windows Text Flags.

Type **IGNORE** to omit flags.

WindowsFontId

This is the **id** of a WindowsFont command for a font and other settings to use in the printing.

It is better that the WindowsFont command is placed before the current

Parameters= command.

TimeDurate

Type the number of tick frames (one tick frame = 1/30th of second) that the text will be shown on the screen.

Type **IGNORE (or -1)** for the text to show forever.

To remove the text call another **F364** trigger.

Left, Top, Right, Bottom

When printing windows text it is necessary to identify a bounding box on the screen

where the text is aligned according to the alignment flags set in the WindowsFont command.

The values are in micro units, where 1000 is the maximum width or height of the screen.

For example the box: 0 , 0, 1000, 1000

This covers the whole screen and is good for a long multi line central alignment.

PB_constants

8 \$0008: PB_DOUBLE_FACE

Used in the Customize=CUST_PARALLEL_BARS.

By default, the Chronicles parallel bar has a weird property:

**It works with long jumps only from one side,
while from the opposite side the jump is disabled.**

Lara performs a single pirouette and remains in the closed sector.

Probably this behaviour has been made to stop the parallel bars being used to go backwards in the level in some situations.

To make the parallel bar work the same for both directions customize it with the **PB_DOUBLE_FACE** flag.

2 \$0002: PB_LARA_CAN_SLIDE

Used in the Customize=CUST_PARALLEL_BARS.

By default Lara is not able to move when she hangs on the bar.

She can only turn.

Adding the **PB_LARA_CAN_SLIDE** in the **CUST_PARALLEL_BARS** Lara will be able to slide to the left or to the right.

Remarks: Place two or more **PARALLEL_BARS** items to create a long bar and Lara will be able to move along all of this path.

The power of the jump is affected by the last parallel bar item touched by Lara before leaving the bar to perform the jump.

For a long bar using different **PARALLEL_BARS** set different OCB values in these items and Lara will jump with different power according to the OCB value stored in the parallel bar.

PB_constants

4 \$0004: PB_MULTIPLE_ENDINGS

Used in the Customize=CUST_PARALLEL_BARS.

By default Lara will perform a correct jump when she lets go of the bar.

It is not possible to go wrong because the jump is always the same.

Use the **PB_MULTIPLE_ENDINGS** flag for situation changes: **now Lara could go wrong, turning, falling vertically, headlong or a stand-up position.**

Remarks: This option is not perfect because the **state id** change should require a correct, specific, next animation to continue the source animation of turning, frame for frame.

Theoretically create a new custom animation and link it to **animation 462 State id = 128** using the change **State id** editor in the **Wad Tool** program.

To create a new animation where Lara loses the hanging at a different point of the turn, create a change **State id** and set the range of frames for the 462 animation where Lara can let go of the bar.

Then set the **State id = 129** and the number of the new animation.
To perform this attempt do not use the **PB_MULTIPLE_ENDINGS** value because this flag performs a change of **State id** in a hard-coded mode and this could interfere with the changes of the **State id**.

Technically the code performs this computes:

When it detects that Lara lets go of the **Action key**: set as next **State id = 129**

This change does not happen immediately but only when the infinite animation 462 finds a correct frame range in the **State id** change.

By default this happens between frame 8 and 9 of animation 462 and the next animation will be animation 463.

Add the new animation starting from some intermediate position of rotation and then apply to the specific frame range where Lara has her starting position.

PB_constants

1 \$0001: PB_PROGRESSIVE_CHARGE

Used in the Customize=CUST_PARALLEL_BARS.

By default the power of the jump is given by the OCB value set in the parallel bar object in the **Tomb Editor** program.

If the **PB_PROGRESSIVE_CHARGE** flag is added the power of the jump will depend on the number of full turns around the bar.

For each turn the power of the jump will be increased by a number in the OCB value.

The maximum power is 10 turns. Over this limit the power will not be increased.

For example: Type 50 in the OCB field and Lara performs 3 full turns before letting go of the the **Action key** ; the power will be $50 * 3 = 150$

Remark: Use the **PB_LARA_CAN_SLIDE** flag to move Lara left or right on the hang bar. Every time Lara moves left or right the counting of the jump power will be cleared, restarting from the OCB Value.

This is useful when the player needs to perform a correct number of turns but also has to move Lara to a different position on the bar.

It is possible to set the correct number of turns:

When the player passes over the correct number of turns move Lara left or right to reset and restart the counting.

PB_constants

16 \$0010: PB_SHOW_CHARGE_BAR

Used in the Customize=CUST_PARALLEL_BARS.

This flag only works when the **PB_PROGRESSIVE_CHARGE** flag is set.

When the **PB_PROGRESSIVE_CHARGE** flag is enabled Lara is able to increase the power of her jump turning many times.

It is advisable to add this feature because the player could have some difficulty understanding how many turns were made.

32 \$0020: PB_SHOW_CHARGE_COUNTER

Used in the Customize=CUST_PARALLEL_BARS.

This flag only works when the **PB_PROGRESSIVE_CHARGE** flag is set.

The charge counter is text showing the current number of turns.

Use this flag and the counter text will be printed on the screen while Lara is turning on the bar.

If required use this flag together with **PB_SHOW_CHARGE_BAR**.

Remark: The source of the charge counter text is always the PSX string with index = 220.

To change the text or create another language version remember to use the PSX 220 string and use the formatter **%02d** at some point in the text because the formatter will be replaced with the current number of turns.

PL_constants

256 \$0100: PL_ADD_INFO_BAR

Used in the Diary= command.

Different from other **PL_** flags, add the **PL_ADD_INFO_BAR** to another but only one **PL_** flag.

Adding the **PL_ADD_INFO_BAR** to some other layout informs the **TRNG** engine that the current layout on the bottom row of the screen should be reserved and no Diary text should be typed over it.

Use this flag to use the bottom strip of the background image to show arrows or some short information about the keyboard commands to manage Lara's Diary.

When the **TRNG** engine detects this flag it formats the text avoiding the bottom row of the screen so it can be a dedicated zone for small icons or text information.

Remark: The information to insert in the Information Bar are the description of the keyboard commands accepted by Lara's Diary:

RIGHT ARROW	=	Show next page
LEFT ARROW	=	Show previous page
HOME	=	Show first page of diary
END	=	Show last page of diary
ESCAPE	=	Exit from Diary
SPACE	=	Zoom Image (show a small image at full screen). [SPACE to come back to original size].

6 \$0006: PL_CENTRAL_IMAGE

Used in the Diary= command.

In this layout the image will be shown at the centre of the screen in the top-half side.

Three widths of image in this layout is a bit larger than the left/right layouts:

the 60% of screen versus the 40% of the previous layouts.

PL_constants

512 \$0200: PL_CUSTOM_LAYOUT

Used in the **Diary=** command.

If you are not satisfied with the pre-set layouts create your own layout using this flag.

Set the **PL_CUSTOM_LAYOUT** (omitting all other **PL_** layouts) and the **TRNG** engine will expect to find in the text information about the layout to use.

In the <FORMAT section of text place these three tag formatters:

```
#FRAME_IMG#=XPos,YPos,SizeX,SizeY  
#FRAME_T1#=XPos,YPos,SizeX,SizeY  
#FRAME_T2#=XPos,YPos,SizeX,SizeY
```

and the **TRNG** engine will use these frames to show the text and further images.

All values are in micro units.

A different layout can be set for each page.

When the **TRNG** engine shows a page there is no custom layout.

Set a **PL_** layout in the **Diary=** command and also a format tag on the page.

That page will display using the format tag,
while the following pages with no format tag will be shown in the **PL_** layout
set in the **Diary=** command.

To know all tags see <FORMAT section of text.

See the description of the **Diary=** command in the **NEW SCRIPT COMMANDS**.

PL_constants

1 \$0001: PL_DOUBLE_PAGE

Used in the Diary= command.

This layout divides the screen into two ideal pages.

The text will be formatted in two columns:

and **first** the left column (left half of screen).
then the right column (right half of screen).

IMAGE: If there is a small image to show in this layout it will be placed at the top of the left page and it will fit across the whole width of the left page

TITLE: If there is a title for this page it will be placed at the top of the left page.

If there is also an image the sorting (in left page) will be: **[Image]**
[Title]
..text ..

Remark: If a long image is used that covers the left side the right side can only contain text.

1024 \$0400: PL FIX WIDE SCREEN

Used in the Diary= command.

When the game is played full screen on a wide-screen monitor the images of the Diary will be stretched and Lara will become an obese dwarf.

To avoid this deformation set this flag.

When the **TRNG** engine finds the **PL_FIX_WIDE_SCREEN** flag it will show the background image of the Diary at full-screen but with a width smaller than the screen to preserve the normal ratio between width-height. [ratio of 1.3 [width] divided by [height] = 1.3]

Viewing the Diary, at the left and right there will be two empty columns.

This look could be good for a background image drawn in the style of a book.

PL_constants

2 \$0002: PL_LEFT_IMAGE

Used in the Diary= command.

This layout works like a single wide page.

The image will be shown in the top left corner of the screen and its size will be about 40% of the Tomb Raider screen.

The title will be at the top of the top-right corner and below it will be the common text.

In this layout the text will be shown in two frames:

A first frame in the top-right corner of the screen with same height of the image at the left.

A second and final frame in the bottom half of screen.

This last frame will have the width of the whole Tomb Raider screen.

3 \$0003: PL_LEFT_IMAGE_LOGO

Used in the Diary= command.

This layout is like the **PL_LEFT_IMAGE** layout about the size and position of the small image.

In this layout there is a single text frame in the bottom half of the screen.

The top-right zone will be empty.

The "hole" in the top-right corner could seem "ugly" but it depends on the image set as a background for the Diary.

Create a fixed image in the top-right corner of the background image to show a logo of the adventure like Tomb Raider 2 and 3.

4 \$0004: PL_RIGHT_IMAGE

Used in the Diary= command.

In this layout the image will be shown in the top-right corner.

The text is formatted in two frames:

The first frame is at the top-left corner of the screen

The second frame will fit the bottom half of the screen.

Remark: The title will be shown at the top of the top-right frame before showing common text.

PL_constants

5 \$0005: PL_RIGHT_IMAGE_LOGO

Used in the **Diary=** command.

This layout is similar to the **PL_RIGHT_IMAGE** layout.

In this case there is a single text frame in the bottom half size of the screen.

The frame at the top-left of the screen will be left empty.

Like the **PL_LEFT_IMAGE_LOGO** use the "empty" top-left frame for the logo for the adventure drawing the image in the background image for the Diary.

Remark: When the **PL_LEFT_IMAGE_LOGO**
or **PL_RIGHT_IMAGE_LOGO**
layout is selected and the page has no small image to show,
the whole top-half of the Tomb Raider screen will be left empty
to avoid text in the logo zone.

7 \$0007: PL_WIDE_IMAGE

Used in the **Diary=** command.

This layout is similar to the **PL_CENTRAL_IMAGE** layout.

In this case the image will fit 90% of the screen.

Use this layout and create a wide image otherwise it could cover the whole screen.

The use of this layout is useful to show two small images on each page.

The trick is to create a unique image where there are two pictures,
one on the left and the other on the right.

PLACE_constants

2 \$0002: PLACE_FLOATING

Used in the Animation= command.

Lara is floating on a water surface.

0 \$0000: PLACE_GROUND

Used in the Animation= command.

Lara is on the ground, i.e. she is out of water.

Ground means Lara is falling down, climbing, monkey, jumping.

That is ground = not in water

4 \$0004: PLACE_LOW_WATER

Used in the Animation= command.

Lara is in shallow water.

Low water is when Lara can wade with her feet on the bottom.

3 \$0003: PLACE_SPECIAL

Used in the Animation= command.

Lara is in a vehicle or she is in **DOZY mode**.

This status means: **Lara is performing hard-coded animations where the physics rules about gravity or water floats will be ignored.**

1 \$0001: PLACE_UNDERWATER

Used in the Animation= command.

Lara is underwater.

Do not confuse this situation with **PLACE_FLOATING**.

Lara is only underwater when she is under the water surface and she is not able to breathe.

QSF_constants

0 \$0000: QSF_SIZE_260x200

Used with the **Customize=CUST_INNER_SCREENSHOT**.

The size of the image with this resolution is 153 Kb for true colour.

For compressed 256 colours (non true colour) (about) 53 Kb.

1 \$0001: QSF_SIZE_320x240

Used with the **Customize=CUST_INNER_SCREENSHOT**.

The size of the image with this resolution is 225 Kb for true colour.

For compressed 256 colours (non true colour) (about) 77 Kb.

2 \$0002: QSF_SIZE_390x300

Used with the **Customize=CUST_INNER_SCREENSHOT**.

The size of the image with this resolution is 343 Kb for true colour.

For compressed 256 colours (non true colour) (about) 115 Kb.

3 \$0003: QSF_SIZE_468x360

Used with the **Customize=CUST_INNER_SCREENSHOT**.

The size of the image with this resolution is 493 Kb for true colour.

For compressed 256 colours (non true colour) (about) 163 Kb.

4 \$0004: QSF_SIZE_520x400

Used with the **Customize=CUST_INNER_SCREENSHOT**.

The size of the image with this resolution is 609 Kb for true colour.

For compressed 256 colours (non true colour) (about) 199 Kb.

5 \$0005: QSF_SIZE_640x480

Used with the **Customize=CUST_INNER_SCREENSHOT**.

The size of image with this resolution is 900 Kb for true colour.

For compressed 256 colours (non true colour) (about) 288 Kb.

QSF_constants

256 \$0100: QSF_TRUE_COLOR

Used with the Customize=CUST_INNER_SCREENSHOT.

Set a true colour 24 bits for the inner image.

By default when this flag is missing the image will be saved at 256 colours in a compressed format.

The RGB images are better than 256 compressed but the size of RGB images is large and for a mix of true colour with a large save game could reach 1 Mb.

This is not a good choice because players often want to exchange save games and large save games are not good for exchanges on-line.

RAIN_constants

2 \$0002: RAIN_ALL_OUTSIDE

Used in the Rain= command.

To have rain in all outside rooms without the setting the **Rain** button for each room in the **Tomb Editor** use this command setting.

Remarks: To use the **RAIN_ALL_OUTSIDE** setting the trick is to set the rain intensity to enable the **Rain button** in the first room that will be visited by Lara.
Set in the Water Intensity field (right of the multi state button water/rain/snow...) the intensity (from 1 to 4).

The intensity for all of the level will be set by that room.
If another "rain" room has different intensity,
when Lara enters the room the rain intensity will change.

If no rain is used it is better to use the command **Rain=RAIN_DISABLED**
(or omit to type Rain= command)
to inform the **TRNG** engine not to compute further rain rooms.

0 \$0000: RAIN_DISABLED

Used in the Rain= command.

Use Rain=RAIN_DISABLED in the script
The same result can be obtained by not using the Rain= command in the script.

1 \$0001: RAIN_SINGLE_ROOMS

Used in the Rain= command.

With this setting the rain will only be shown in specific rooms signed as "Rain" in the **Tomb Editor** with outside status.

RIB_constants

1024 \$0400: RIB_ADD_SCANCODE_LIST

Used with Parameters=PARAM_INPUT_BOX.

This flag expands the Scan Codes recognized by the keyboard parsing procedure.

By default the input box parses and only recognizes certain keys for all QWERTY keyboard layouts but ignores keys that change for different nationalized keyboards, like "?", "!", "@" etc.

This flag uses the **InputBox** procedure to also check for other keys, giving information about Scan Codes and characters to display on the screen.

When the **RIB_ADD_SCANCODE_LIST** flag is added, also add a NG extra string with a fixed number "666" (only to remember it better).

The syntax of the 666 ng string will be : 666: AABBCc, A1B1C1, A2B2C2 ... etc....

Practically type a series of hexadecimal numbers (omitting the "\$" prefix) divided by commas.

Each hexadecimal number gives information to the input box procedure, about what Scan Code read by the keyboard should be parsed and the ASCII code to draw on screen. (the Scan Codes are those in **KEYBOARD SCANCODES**).

The "AA" pair will be the ASCII code (UTF-7 set), while the BB (and further, optional, "CC" and "DD") will be single Scan Code keys at the same moment, to draw the "AA" ASCII code.

For instance: The "?" character in the Keyboard Scan Codes list has no Scan Code value.

The reason is that the Scan Codes only show the code for each single key of the keyboard. Most keys have two or more symbols on each key.

So for US/UK keyboards, the "?" is over the "/" symbol, so look for the "/" character in the Keyboard Scan Codes list. That is : **\$35** "Backslash /"

The Scan Code is **\$35** but to have the "?" on screen add the "SHIFT" key.

The ASCII code of the "?" character is **\$3F** (or 63 decimal).

To complete the required information check for the scan code of SHIFT: **\$2A**

Create the hex number to add the "?" character to the input box: **\$3F352A**

Read the number like a series of 2 hex digits: "3F": on screen print ASCII "3F", i.e. the "?" character

"35" when user hits the key "/" and in same moment there is the "2A" key down (SHIFT).

RIB_constants

Notes:

There are two shift keys, left and right, with different scan codes, **TRNG** will extend the left shift to the same right shift. This means that the hex scan code for only one "SHIFT" key is required and the **TRNG** will **check** for the other (left or right shift) key.

For the ALT keys, tomb4 has a limited scanning and does not recognize [ALT GR] as right ALT but sees all ALT as one scan key code.

For CTRL two different CTRL keys give the same scan key code.

The Maximum amount of numbers (divided by commas) is 64.

Remember that the position of symbols changes between different keyboards according to nationality.

So to support inter punctuation characters (like ",", ":", ";", etc.) there are only two ways:

Method 1

Use only US/UK keyboard layout and warn players that some keys do not work showing a different character instead from that displayed on their keyboards.

Method 2

Create two or more **langauge.dat** files with different scan codes according to the layout "German" "French" "Italian" etc.

Theoretically keys with no visible character to display can be scanned, like "F1" "F2" function keys etc.

In this case it is better to also use the **RIB_SHORTCUT_KEY+RIB_HIDE_TEXT** flags. Then supply as ASCII code a special character to verify its presence after the **inputbox**.

RIB_constants

64 \$0040: RIB_ALIGN_CENTER

Used with the Parameters=PARAM_INPUT_BOX command.

Set a central alignment of text typed by the user.

The text is centred inside a rectangle set in the Image script command.

Notes: If this flag is omitted the text will be aligned left.
It is not advisable to use central alignment with a blinking caret because :

Every time the caret disappears, the text will become shorter
and the central alignment will move position,
giving the feeling of text shortening.

8 \$0008: RIB_BLINK_CARET

Used with the Parameters=PARAM_INPUT_BOX command.

Perform a blinking of the caret while it is waiting for input.

The blinking simulates the old MS-DOS prompt.

4 \$0004: RIB_HIDE_CARET

Used with the Parameters=PARAM_INPUT_BOX command.

Omit to draw a underscore character "_" to simulate the cursor.

By default a cursor will always be drawn to the right of the last typed character.

4096 \$1000: RIB_HIDE_TEXT

Used with the Parameters=PARAM_INPUT_BOX command.

This flag will stop the keys hit by the user being drawn on the screen.

In spite of this behaviour all valid keys typed by the user will be stored in the **LastInputText** variable and the **LastInputNumber** (if only digits).

Usually use this flag together with the **RIB_HIDE_CARET** flag and manage a single key command with the **RIB_SHORTCUT_KEY** flag.

RIB_constants

512 \$0200: RIB_INPUT_BELOW_BIG_TEXT

Used with the Parameters=PARAM_INPUT_BOX command.

This flag could only be used with the **RIB_PRINT_BIG_TEXT** flag.
When the printing of text (that in **TRNG** BigText variable) is enabled on the screen, before waiting for user input, use the **RIB_INPUT_BELOW_BIG_TEXT** flag to change the Y position (in height axis) of the input box rectangle to show input text immediately below the last row of BigText drawn.

|The input zone will change according to the last printed text.

Note: This flag only affects the Y position of the input box rectangle, the relative height of the rectangle or its width and X origin will not be modified.

128 \$0080: RIB_ONLY_CAPS

Used with the Parameters=PARAM_INPUT_BOX command.

Forces all text typed by the user into capital letters.
By default the text will be drawn in lower or capital letters according to the use of the **SHIFT** key.

1 \$0001: RIB_ONLY_DIGITS

Used with the Parameters=PARAM_INPUT_BOX command.

Forces the Input Box to only accept digits (0,1,2,3,4,5,6,7,8,9).
Further letters (a,b,c etc.) typed by the user will be ignored.

When this flag is used the user will only be able to type a number (with any digits).
At the end press ENTER.
The digits typed will be stored as a number in the **LastInputNumber** **TRNG** variable.

Then it is possible to check the value against another using the flip effect for **TRNG** variables.

Notes: If the user quits the input box with the **ESCAPE** command and the **RIB_ONLY_DIGITS** flag is used, in the **LastInputNumber** there will be the "-1" value.

This flag is incompatible with the **RIB_ONLY_LETTERS** flag.

If either the **RIB_ONLY_DIGITS** or the **RIB_ONLY_LETTERS** flags is omitted the input box will accept numbers and letters.

If the **RIB_ONLY_DIGITS** flag is omitted the **LastInputNumber** variable will not be affected by the Input Box.

RIB_constants

2 \$0002: RIB_ONLY_LETTERS

Used with the **Parameters=PARAM_INPUT_BOX** command.

Ignores digits (0123456789).

Digits typed by the user will be ignored.

256 \$0100: RIB_PRINT_BIG_TEXT

Used with the **Parameters=PARAM_INPUT_BOX** command.

Set this flag when the background image of the input box has been drawn.

Read and draw the text typed by the player stored in the **TRNG** variable **BigText**.

When this flag is set it is necessary to type in the **ExtraParam** field of the **Parameters=PARAM_INPUT_BOX** script command, the Id of a **Parameters=PARAM_RECT** script command.

The data of the **PARAM_RECT** will be used to set the position of the rectangle (in micro units) to draw the **BigText** string.

The data of the **PARAM_RECT** will be used to set the colour of the text, the **BackColor** field will be ignored.

The idea behind the **RIB_PRINT_BIG_TEXT** flag is to have a generic input box. For instance with a background image showing a computer screen, it is possible to use it over and over with different text requiring different passwords or input commands to perform operations in the game.

Remember to perform the F409 trigger to show the input box within a Trigger Group. The first F409 trigger will be a trigger to copy some text (from strings of .dat files) in the **BigText** variable.

In this way customize every time in run-time the texts to show in the "pc/control panel" input box in the game.

RIB_constants

32 \$0020: RIB_PRINT_ONLY

Used with the **Parameters=PARAM_INPUT_BOX** command.

The input box will draw on the screen the **LastInputText** typed text and the further **Bigtext** without allowing the user the chance to type text.

The input box works like a common overlapped image with some printed text. There is only one advantage using the Input box in this way which is to have all settings about position, colour and font, to print the text to the right of the previous inserted text.

Only use this flag with a **Parameters=PARAM_INPUT_BOX** command to display error messages.

The input box with this flag does not allow any kind of text input but only allows the Escape command to quit the input box.

2048 \$0200: RIB_SHORTCUT_KEY

Used with the **Parameters=PARAM_INPUT_BOX** command.

Adding this flag the input box will quit when the ENTER command is pressed immediately after the first valid key chosen by the user.

It should be used for instance to handle a menu with a list of commands labelled with a number (1. 2. 3.) or letters (A. B. C. D.)

Note: This flag does not affect any other behaviour of the input box. So for the caret to be hidden or the inserted text not drawn on screen, add flags like **RIB_HIDE_CARET** and **RIB_HIDE_TEXT**.

16 \$0010: RIB_SOUND_ON_KEY

Used with the **Parameters=PARAM_INPUT_BOX** command.

Perform a sound effect every time the player hits a key. The sound effect number is typed in the **SfxSound** field of the Parameters command.

ROOM_constants

-4 \$FFFC: ROOM_COLD

Used in accordance with the ENV_ROOM condition in the Animation or MultEnvCondition commands.

Lara is in a Cold Room.

-3 \$FFFD: ROOM_DAMAGE

Used in accordance with the ENV_ROOM condition in the Animation or MultEnvCondition commands.

Lara is in a Damage Room.

-9 \$FFF7: ROOM_MIST

Used in accordance with the ENV_ROOM condition in the Animation or MultEnvCondition commands.

Lara is in a room marked as a Mist Room.

-7 \$FFF9: ROOM_OUTSIDE

Used in accordance with the ENV_ROOM condition in the Animation or MultEnvCondition commands.

Lara is outside.

-2 \$FFFE: ROOM_QUICKSAND

Used in accordance with the ENV_ROOM condition in the Animation or MultEnvCondition commands.

Lara is in a Quicksand Room.

-5 \$FFFB: ROOM_RAIN

Used in accordance with the ENV_ROOM condition in the Animation or MultEnvCondition commands.

Lara is in a Rain Room.

-8 \$FFF8: ROOM_REFLEX

Used in accordance with the ENV_ROOM condition in Animation or MultEnvCondition commands.

Lara is in a room with water light reflex.
Usually the Reflex Rooms are those over the Water Room.

ROOM_constants

-6 \$FFFA: ROOM_SNOW

Used in accordance with the ENV_ROOM condition in the Animation or MultEnvCondition commands.

Lara is in a Snow Room

-1 \$FFFF: ROOM_WATER

Used in accordance with the ENV_ROOM condition in the Animation or MultEnvCondition commands.

Lara is in a Water Room.

Please note that Lara is in a Water Room when she is floating on the water, or when she is swimming underwater.

She is not in water when she is wading in shallow water.

ROTH_constants

1 \$0001: ROTH_CLOCKWISE

Used in the Parameters=PARAM_ROTATE_ITEM command.

Rotate in a horizontal clockwise direction

2 \$0002: ROTH_INV_CLOCKWISE

Used in the Parameters=PARAM_ROTATE_ITEM command.

Rotate in a horizontal anti-clockwise direction

65535 \$FFFF: ROTH_NONE

Used in the Parameters=PARAM_ROTATE_ITEM command.

Disable the horizontal rotation

ROTV_constants

2 \$0002: ROTV_BACKWARD

Used in the Parameters=PARAM_ROTATE_ITEM command.

Vertical backward rotation

1 \$0001: ROTV_FORWARD

Used in the Parameters=PARAM_ROTATE_ITEM command.

Vertical forward rotation

65535 \$FFFF: ROTV_NONE

Used in the Parameters=PARAM_ROTATE_ITEM command.

Disable the vertical rotation

SC_constants

5 \$0005: SC_DOUBLE_HEIGHT

Used in the TextFormat= command.

Set the double height for characters.

The width remains the same.

6 \$0006: SC_DOUBLE_SIZE

Used in the TextFormat= command.

Set double width and double height for characters.

4 \$0004: SC_DOUBLE_WIDTH

Used in the TextFormat= command.

Set double width for characters.

The height remains the same.

2 \$0002: SC_HALF_HEIGHT

Used in the TextFormat= command.

Reduced the height of characters by half.

The width remains the same.

3 \$0003: SC_HALF_SIZE

Used in the TextFormat= command.

Reduce by half the width and height of characters

1 \$0001: SC_HALF_WIDTH

Used in the TextFormat= command.

Reduce by half the width of characters.

The height remains the same.

0 \$0000: SC_NORMAL

Used in the TextFormat= command.

Set normal text.

SEQ_constants

1 \$0001: SEQ_LOOP

Used in the TextureSequence= command.

The sequence will be performed in an endless way.

Use this flag and use the specific flip effect to stop the sequence, otherwise it will be performed continuously.

2 \$0002: SEQ_LOOP_INVERSE

Used in the TextureSequence= command.

This flag only works when the **SEQ_LOOP** flag is set.

The single **SEQ_LOOP** flag sets an infinite loop with this sequence,

For example: Suppose 4 textures from 0 to 3: 0 1 2 3 0 1 2 3 0 1 2 3

When the SEQ_LOOP_INVERSE flag is also set,
the sequence will be: 0 1 2 3 2 1 0 1 2 3 2 1 0 ..

4 \$0004: SEQ_STOP_AT_FIRST

Used in the TextureSequence= command.

Set this flag to force the **TRNG** engine to set the first texture of the range when the animation is completed.

If this flag is omitted the last texture shown will depend on the type of animation.

If no loop is set:	The last texture shown will be the texture of the last index of the array sequence.
--------------------	--

If a loop is set:	The texture shown will depend on the time when the Stop texture sequence flip effect is activated.
-------------------	---

SET_constants

32 \$0020: SET_ACCEPT_EXTRA_TAILINFOS

Used in the Settings= command.

This setting forces the **TRNG** engine to accept up to 32767 tail informations in the tr4 files.

Using this setting the limit for the **NGLE** remains the old limit of 1024 tail informations. Therefore the only reason to use this setting is to use the **Meta2tr** program to replace the room meshes in the tr4 file.

Meta2tr increases the number of tail informations to support the new extra tail information.

2 \$0002: SET_BLIND_SAVEGAMES

Used in the Settings= command.

To stop players using a save game editor to change save games. Any attempt by players to modify save games will cause a crash at reload.

8 \$0008: SET_CRYPT_SCRIPT

Used in the Settings= command.

Setting this flag the **script.dat** file will be encrypted to stop decompilation by other utilities.

If security issues like **SET_DISABLE_CHEATS** are set it is advisable to set encrypting of the **script.dat**, otherwise the file can be decompiled and recompiled to remove the **SET_DISABLE_CHEATS** flag.

1 \$0001: SET_DISABLE_CHEATS

Used in the Settings= command.

Disable the hidden cheats to skip the level to get infinite weapons or items.

Remark: To disable the FlyCheat use the old command FlyCheat = DISABLED.

SET_constants

64 \$0040: SET_FORCE_NO_WAITING_REFRESH

Used in the Settings= command.

This setting, together with the [SET_FORCE_SOFT_FULL_SCREEN](#) setting, changes the setting of the Tomb Raider game to solve the problem of FMV playing or flickering images.

See the [SET_FORCE_SOFT_FULL_SCREEN](#) description:

**The player is now able to set this option so it is better not to use this setting in the script.
(From Version 1.2.2.7)**

Technically this setting disables the waiting setting in the flip DIRECTX method.

Some experiment demonstrates that this disabling is able to remove the flickering of images from the exclusive full screen resolution.

SET_constants

16 \$000F: SET_FORCE_SOFT_FULL_SCREEN

Used in the Settings= command.

Note: From 1.2.2.7 version the previous SET_SOFT_FULL_SCREEN has been removed.

The current SET_FORCE_SOFT_FULL_SCREEN constant performs the same job. Now the player is able to set or remove the soft full screen setting because it has been included in the tomb raider set up window.

For this reason the name of the SET_SOFT_FULL_SCREEN is different from that in past versions.

Now any soft full screen setting can be omitted and the player will be able to enable or disable it.

It should be better not to use this forcing of soft full screen because if it is in the script the player will not be able to disable it from the set up window. It is more logical that the player can choose to use or disable this option as he knows his computer.

When the game (tomb4) works in full screen it is difficult for TRNG to start a movie (FMV) because the screen mode, known as "full screen", is an Exclusive mode, i.e. the game catches the screen and does not release it to other DirectX tasks (like the FMV viewer).

With this setting the Exclusive Full Screen Video Mode will be converted into a Software (cooperative) Full Screen mode.

Practically the game will be shown in full screen but the DirectX mode will not be exclusive and therefore the FMV will be shown. So no blinking or slow time at the start or end of the FMV.

Remark: This setting will have no effect if the user sets tomb4 to work in windowed mode. This setting only works when tomb raider is set to work in full screen mode.

SET_constants

4 \$0004: SET_PERFORM_FROM_CD

Used in the Settings= command.

With this constant the game will be able to be played from a CD or DVD and generally with only read storage support.

The reason to use this setting is to give the game stored on CD and played directly from the CD to a friend with no need of installation of the level editor (trle folder) on the local disk.

When this setting is present the **TRNG** engine will save and load the save game from a new folder on the local drive C: with the name of the first level.

For example: With this option a level described in the **script.txt** **City_Of_The_Dead**, the **TRNG** engine will create a folder named: **C:\City_Of_The_Dead** and it will save and load save games in this folder.

Remark: The **start.exe** and autorun.inf files should be in the root of the CD, i.e. outside the trle folder.

To create an autostart CD perform the following steps:

Add in the **script.txt** file in the [Options] section :

Settings = SET_PERFORM_FROM_CD

Build the script.txt

When the trle folder has all the level files (.dat , .tr4) necessary to play the game, save the trle folder including the name of the trle folder in the CD image to burn onto CD.

Before burning the image add to the image the following files found in the Extra_NG_File.zip:

START.exe
autorun.inf

Now burn the CD and at the end browse the new created CD to check the format:

START.exe
autorun.info
trle (Folder)

Now insert the CD in a CD ROM drive and the game will start and play by itself.

SEXT_constants

5 \$0005: SEXT_AIFF Obsolete.

See the description for the **CUST_CUST_NEW_SOUND_ENGINE** customize flag.

4 \$0004: SEXT_MP1 Obsolete.

See the description for the **CUST_CUST_NEW_SOUND_ENGINE** customize flag.

3 \$0003: SEXT_MP2 Obsolete.

See the description for the **CUST_CUST_NEW_SOUND_ENGINE** customize flag.

2 \$0002: SEXT_MP3 Obsolete.

See the description for the **CUST_CUST_NEW_SOUND_ENGINE** customize flag.

100 \$0064: SEXT_MULTIPLE Obsolete.

See the description for the **CUST_CUST_NEW_SOUND_ENGINE** customize flag.

1 \$0001: SEXT_OGG Obsolete.

See the description for the **CUST_CUST_NEW_SOUND_ENGINE** customize flag.

0 \$0000: SEXT_WAV Obsolete.

See the description for the **CUST_CUST_NEW_SOUND_ENGINE** customize flag.

SHOWC_constants

1 \$0001: SHOWC_OMIT_AMMO_NAME

Used in the Customize=CUST_SHOW_AMMO_COUNTER command.

With this setting the current name of the selected weapon ammo will be omitted.

For example: Instead of displaying **Normal Cross Bow Ammo 12** it will only display **12**.

2 \$0002: SHOWC_USE_GRAPHIC_AMMO

Used in the Customize=CUST_SHOW_AMMO_COUNTER command.

This setting removes the literal name of the ammo and replaces it with a single graphic character.

Use the **TRNG** engine and select characters from the font set.

One single character for each kind of ammo will be displayed on the screen

The graphic characters used for each ammo is in the following list:

Name_in_font_Editor	Ammo_Type
127: [GRAPHIC]	Pistol Ammo
129: [GRAPHIC]	Revolver Ammo
138: S	Uzi Ammo
140: OE	Shotgun Normal Ammo
141: [GRAPHIC]	Shotgun Wide shot Ammo
142: Z	Grenade gun Normal Ammo
143: [GRAPHIC]	Grenade gun Super Ammo
144: [GRAPHIC]	Grenade gun Flash Ammo
154: s	Crossbow Normal Ammo
156: oe	Crossbow Poison Ammo
157: [GRAPHIC]	Crossbow Explosive Ammo

Remark: To add icons in the NG font character set, perform the following operations:

Go in to the **[Tools]** panel of the **NG_Center**
Click on the **[NG Font Editor]** button.

In the **NG Font Editor** window click on the **[Choose Wad]** button
Select a wad where there is a NG font object.

Select in the combo box (below the **[START IMPORTING]** button) a character, taking care to choose the correct number read from the above list.

Now click on the **[Import BMP]** button close to the small single character image and select the bmp file where the image for that char/ammo is stored.

Select in the **[Type]** list the voice "Graphic" and click the button **[Recompute]**
Repeat until all of the images for all of the ammo are assigned.
Click on the **[Exit and Save wad]** button.

SHOWC_constants

4 \$0004: SHOWC_USE_GRAPHIC_WEAPON

Used in the Customize=CUST_SHOW_AMMO_COUNTER command.

This setting removes the name of the ammo type and replaces it with a graphic character to signal the current weapon.

SNOW_constants

2 \$0002: SNOW_ALL_OUTSIDE

Used in the Snow= command.

The snow is shown in all outside rooms, ignoring the multi state button.
This means that a room without "snow" will have snow if it is not a Water Room and it has the [O] outside status.

Remarks:

For the **SNOW_ALL_OUTSIDE** set a single room with the "Snow" attribute in the **NGLE** and set the value for snow intensity.
That value will be used for the whole level.

If there is no snow it is better to use the command

Snow=SNOW_DISABLED

to inform the **TRNG** engine to avoid a lot of computes to locate snow rooms.

0 \$0000: SNOW_DISABLED

Used in the Snow= command.

This setting is the same as not inserting the Snow= command.

1 \$0001: SNOW_SINGLE_ROOM

Used in the Snow= command.

Only the room with the "Snow" label will have snow in the game.

Each room will have its own snow intensity.

SPC_constants

49152 SC000: SPC_ANIMATION

Used with the Parameters=PARAM_ACTOR_SPEECH command.

Add a **index** for a moveable to play the custom animation.

Example: **SPC_ANIMATION+15**

Will perform the animation 15.

40960 SA000: SPC_HEAD_NOD

Used with the Parameters=PARAM_ACTOR_SPEECH command.

Add a **index** for a moveable to play the custom animation.

Example: **SPC_ANIMATION+15**

Will perform the animation 15.

36864 \$9000: SPC_HEAD_SHAKE

Used with the Parameters=PARAM_ACTOR_SPEECH command.

This simulates the shaking head to show the feeling to say: "no, I don't...", "it's not true" etc.

Two values can be set:

The speed of the shaking movement.

The angle in degrees of the head turning, computed for each side.

To add the values use the formula: **SPC_HEAD_SHAKE + Speed * 64 + Angle**

Speed

This is the turning increment given in degrees.

Each increment is added at each frame.

For a shaking **Angle** of 20 degrees at a **Speed** of 3 degrees per frame:

$$\text{SPC_HEAD_SHAKE} + (20 * 64) + 20$$

Where: **190 = 3 (degrees) * 64**
 20 is the angle for a single side.

Note: Remember that for both arguments the maximum value is 63.

SPC_constants

32768 \$8000: SPC_LOOK_DOWN

Used with the Parameters=PARAM_ACTOR_SPEECH command.

Move the head downward for required degrees and frame duration.

See the description of the [SPC_LOOK_RIGHT](#) command for the formula.

24576 \$6000: SPC_LOOK_LEFT

Used with the Parameters=PARAM_ACTOR_SPEECH command.

Move the head left for required degrees and frame duration.

See the description of the [SPC_LOOK_RIGHT](#) command for the formula.

28672 \$7000: SPC_LOOK_UP

Used with the Parameters=PARAM_ACTOR_SPEECH command.

Move the head upwards for required degrees and frame duration.

See the description of the [SPC_LOOK_RIGHT](#) command for the formula.

SPC_constants

20480 \$5000: SPC_LOOK_RIGHT

Used with the Parameters=PARAM_ACTOR_SPEECH command.

Move the head right for required degrees and frame duration.

Two values can be set:

The horizontal turning angle.

The time for turning and still in the turned position.

Both arguments can not be bigger than 63.

To increase this the format is not in frame and single degrees but rather:

The angle is in double degrees. That is for 45 the turning is 90 degrees.

The duration is in tenths of a second.

So for a maximum value of 63 gives six seconds of right looking.

The time in tenths of seconds has to be multiplied by 64 in the formula.

For example: For a turning of 44 degrees, kept for 2.5 seconds

The formula is: **SPC_LOOK_RIGHT+Angle+Duration**

SPC_LOOK_RIGHT+22+1600

The 22 will be multiplied by 2 by the **TRNG** to get 44

The duration in tenths of seconds has to be multiplied by 64.

2.5 seconds is 25 tenth's of seconds. So **Duration** = 25 * 64 = 1600

Notes:

All **SPC_LOOK_** and **SPC_HEAD_** commands only work with Lara.

To have the same effect for another moveable it is necessary to create a new custom animation to move the head and then use a **SPC_ANIMATION** command to force that animation.

All **SPC_LOOK_** and **SPC_HEAD_** commands will overlap the following commands in their playing.

This means that this command will start the turning of the head immediately but in the same frame the next command will begin.

For example to move an actor's mouth.

To make the **TRNG** wait for the turning to complete before performing the next command add after the **LOOK** command a **SPC_PAUSE** command with the same time duration as the **LOOK** command.

SPC_constants

0 \$0000: SPC_MESH

Used with the Parameters=PARAM_ACTOR_SPEECH command.

Choose the head mesh to show.

Type the number to add to the SPC_MESH command.

It is a relative index beginning from zero for the first speech mesh.

For example: SPC_MESH+4

This means the fifth mesh in the sequence (0,1,2,3,4 are 5 meshes).

To set a frame duration different from the **FrameRate** field add the number of frames for the duration:

The number of frames has to be multiplied by 64:

For example: SPC_MESH+256+4

Means duration for 4 frames = $(4 * 64) = 256$

Show the mesh with index=4 (fifth mesh)

Note: As the SPC_MESH command has a value of 0, omit typing it.
Avoid inserting any other SPC_ command:

For instance: 256+4

SPC_constants

61440 \$F000: SPC_NEXT_STATEID

Used with the Parameters=PARAM_ACTOR_SPEECH command.

This Forces the next **State id** to the value given as a argument.

For example: **SPC_NEXT_STATEID+4**

Sets as the next **State id** for the current actor the value **4**.

16384 \$4000: SPC_PAUSE

Used with the Parameters=PARAM_ACTOR_SPEECH command.

Forces a speech pause with a closed mouth for the actor for the given frames.

SPC_PAUSE + FramesOfPause

For instance: **SPC_PAUSE + 45**

The actor will remain mute (closed mouth) for about one and a half seconds (45 frames).

By default the number of frames is the duration of the pause but the **SPC_PAUSE** can also be used to align speech commands with the frames of the current demo playing.

To force an absolute frame add to the argument the special constant **DEMO_FRAME**.

For example: **SPC_PAUSE+DEMO_FRAME+136**

This means: Wait until the demo frame counter reaches the 136th frame.

So you are sure that the next command will be performed on demo frame 136.

SPC_constants

57344 SE000: SPC_PERFORM_TG

Used with the Parameters=PARAM_ACTOR_SPEECH command.

This performs the Trigger Group with the given number.

For example: **SPC_PLAY_TG+24**

This will perform the Trigger Group = **24** from the current level section of the **script.txt** file.

53248 SD000: SPC_PLAY_CD

Used with the Parameters=PARAM_ACTOR_SPEECH command.

This will play the audio with the supplied number.

For example: **SPC_PLAY_CD+112**

This will play the **112.wav** file from the **audio** folder in single playing mode.

45056 SB000: SPC_PLAY_SFX

Used with the Parameters=PARAM_ACTOR_SPEECH command.

This will play the sfx sound set as a argument.

For example: **SPC_PLAY_SFX+104**

This will play the **104** sound (HECKLER_KOCH_STOP)

The Range of values is 0 to 4095.

SPC_constants

12288 \$3000: SPC_SEQUENCE

Used with the Parameters=PARAM_ACTOR_SPEECH command.

This command allows a custom sequence of meshes for other expressions.
It simulates the mouth opening and closing with a sequence of heads.
Other expressions like smiling, angry etc. can be made.

Type two numbers: **The index of the first mesh of the animated sequence**
 The number of meshes used in the sequence

It is not possible to set the frame duration for the animation.

The default frame rate is used.

The formula to use is: **SPC_SEQUENCE + NumberOfMesh* 64 + IndexFirstMesh**

For instance: **SPC_SEQUENCE + 256 + 12**

The mesh sequence is 4 so = **NumberOfMesh * 64 = 4 * 64 = 256**

The animated sequence begins from **IndexFirstMesh** with index = 12

SPC_constants

4096 \$1000: SPC_SYLL

Used with the Parameters=PARAM_ACTOR_SPEECH command.

This will show the simulation of a syllable.

The syllable is defined as a single animation from closed mouth to open mouth or, open mouth to closed mouth.

If the actor has a closed mouth, the syllable command will change heads from a closed mouth to an open mouth.

If the actor has an open mouth, the syllable command will change heads from an open mouth to a closed mouth.

The **SPC_SYLL** command is not a single mesh but an animated sequence of meshes, from the minimum to maximum opening of the mouth.

If no value is added to this command it will use the default frame rate, so each head of the animated sequence will be shown for the **FrameRate** duration.

To have a different frame rate only for the command add the number of frames for each head multiplying it by **64**:

SPC_SYLL+192

As **192** means **3** frames ($3 \times 64 = 192$), each head of the animated sequence will be displayed for **3** frames.

If a value between 1 to 64 is added, it will be the number of syllable to be played with this command.

For instance: **SPC_SYLL+5**

SPC_constants

8192 \$2000: SPC_TEXT

Used with the Parameters=PARAM_ACTOR_SPEECH command.

This draws the **EXTRA_NG** string with the index typed as argument.

For example: **SPC_TEXT+23**

This will draw the **EXTRA_NG** string with index = **23**.

It is important to remember the flip effects **F399 F400 F401**.

They set the start position and colour of text for each actor.

Set the string with the **SPC_TEXT** command and the text will draw with the colour and position for the actor linked to the speech command.

Note: All texts will remain on the screen until another **SPC_TEXT** command is set for the same actor, or the following :

SPC_TEXT+0

The string index = 0 means: remove the previous text and do not print any string for the current actor

SPC_TEXT with no string index supplied this has the same meaning as **SPC_TEXT+0**

SPCF_constants

4 \$0004: SPCF_FREEZE_HAIR

Used with the Parameters=PARAM_ACTOR_SPEECH command.

There is a bug. When there is a swap mesh of Lara's head her hair flicks.
After remapping the ponytail vertices with the right numbers the problem is still present.
To reduce this problem add the **SPCF_FREEZE_HAIR** flag and the ponytail will stay down.
This solution is OK as when Lara talks, you look at her head on and her ponytail is not visible.

2 \$0002: SPCF_LOOP

Used with the Parameters=PARAM_ACTOR_SPEECH command.

To control only Lara speaking for a given number of frames in a random way, add the **SPCF_LOOP** flag and type in the Parameter field the number of frames for the speaking sequence.
The **TRNG** will perform the commands inserted in the loop mode up to the number of frames set in the parameter field.

For example:

Parameters= PARAM_ACTOR_SPEECH, 1, SPCF_LOOP, 120, IGNORE,
LARA_SPEECH_HEAD1, 14, 0, 4, SPC_SYLL, SPC_PAUSE, SPC_SYLL+6,
SPC_PAUSE, SPC_SYLL+3

With the above parameters Lara will talk for 4 seconds ($4 * 30 = 120$)
moving the mouth following the sequence of commands typed for 4 seconds.

1 \$0001: SPCF_OLD_SPEECH_SLOTS

Used with the Parameters=PARAM_ACTOR_SPEECH command.

To use the old multiple slots **LARA_SPEECH_HEAD1/2/3/4** or **ACTOR1_SPEECH_HEAD1/2/3/4** method, add the **SPCF_OLD_SPEECH_SLOTS** flag.

With this old method **TRNG** engine will look for the first head mesh at index = 14 (15th mesh) for the given speech slot. If this is omitted, **TRNG** will use the first mesh (**index=0**) which is the mute face.

In the old method the sorting was inverted:

MaxOpenMouth
OpenMouth
LittleOpen
Mute

In the new method the sorting is:

Mute
LittleOpen
OpenMouth
MaxOpen

Use this flag to manage both situations.

SPF_constants

4 \$0004: SPF_BLINK_SELECTED

Used in the `SavegamePanel=` script command.

By default the selected save game in the list is highlighted with a constant inversion of colours.

To have a blink of this highlighting use the `SPF_BLINK_SELECTED` flag.

8 \$0008: SPF_NO_PANEL_TITLE

Used in the `SavegamePanel=` script command.

By default it is in a central position on the first row of the title panel,
save game: **Load Game** or **Save Game**.

To omit the text use this flag.

The only reason to omit the title of the panel is when a title is on the background image.

1 \$0001: SPF_NO_TIME_IN_LIST

Used in the `SavegamePanel=` script command.

By default the save game list is shown in the old Tomb Raider format:

002 Name of Level 1 days 02:41:11

The game time is “1 days 02:41:11” part.

Using the `SPF_NO_TIME_IN_LIST` flag the game time is omitted
and the width required for the list is less.

Use this flag when the layout requires a save game list in the left or right half of the screen.

SPF_constants

16 \$0010: SPF_PRELOAD_BKG_IMAGE

Used in the SavegamePanel= script command.

By default the **TRNG** engine will load and show the background image of the save game panel when the player requires the save and load panel.

The operation to load it from disc is a delay.

To avoid this add the flag and the background image is loaded into memory in advance when the level is loaded.

Remark: The only reason to omit this flag is when there are a lot of preloaded images in a level, or when a bug is detected in the preloaded background image.

In very seldom circumstances the preloaded image of the save game panel could appear to be damaged.

This problem disappears on exiting and entering the game.
It is probable that this problem happens when there is a preloaded image that damages the memory where the image is stored.

2 \$0002: SPF_SCROLL_PAGE

Used in the SavegamePanel= script command.

When the save game list is only partly shown the player is able to scroll the save games.

SPL_constants

7 \$0007: SPL_CENTRAL_IMAGE

Used in the SavegamePanel= script command.

The image is at the centre in the top half of the screen.

The image is wider than other layout formats,

so this layout gives more scene to the image and works in wide screen mode.

The save game list is shown in the bottom half of the screen.

2 \$0002: SPL_LEFT_IMAGE_BOTTOM_INFO

Used in the SavegamePanel= script command.

The image is at the top left corner.

The information frame is below the image.

The save game list is in the right half of screen.

3 \$0003: SPL_LEFT_IMAGE_NO_INFO

Used in the SavegamePanel= script command.

The image is at the top left corner.

This layout has no information frame.

The top-right corner is an "empty" zone so a logo of the adventure could be placed here in the background image.

The list of save games is in the bottom half of the screen.

1 \$0001: SPL_LEFT_IMAGE_RIGHT_INFO

Used in the SavegamePanel= script command.

The image is in the top-left side.

The information frame is at the right of the image.

The list of save games is in the bottom half of the screen.

5 \$0005: SPL_RIGHT_IMAGE_BOTTOM_INFO

Used in the SavegamePanel= script command.

The image is at the top right corner.

The information frame is under the image.

The list of save games is on the left side of the screen.

SPL_constants

4 \$0004: SPL_RIGHT_IMAGE_LEFT_INFO

Used in the SavegamePanel= script command.

The image is in the top-right corner.

The information frame is left of the image.

The list of save games is in the bottom half of the screen.

6 \$0006: SPL_RIGHT_IMAGE_NO_INFO

Used in the SavegamePanel= script command.

The image is in the top-right corner.

There is no information frame.

The top-left corner is an "empty" zone so a logo of the adventure could be placed here in the background image.

The save games list is in the bottom half of screen.

SQ_constants

44100 \$AC44: SQ_HIGH_QUALITY

Used in the SoundSettings= command.

High quality sound.

11025 \$2B11: SQ_LOW_QUALITY

Used in the SoundSettings= command.

Low quality sound.

22050 \$5622: SQ_MEDIUM_QUALITY

Used in the SoundSettings= command.

Medium quality sound.

STATE_constants

16 \$0010: STATE_BACK

Used in Animations: 038 039 040 041 061 062

25 \$0019: STATE_BACK_JUMP

Used in Animations: 074 075 182 183
209 211 212

61 \$003D: STATE_CLIMB_DOWN

Used in Animations: 168 169

59 \$003B: STATE_CLIMB_END

Used in Animations: Not used.

58 \$003A: STATE_CLIMB_LEFT

Used in Animations: 171

60 \$003C: STATE_CLIMB_RIGHT

Used in Animations: 170

56 \$0038: STATE_CLIMB_START_AND_STANDING

Used in Animations: 162 163 164 166 167 172 173

57 \$0039: STATE_CLIMB_UP

Used in Animations: 160 161 165

15 \$000F: STATE_COMPRESS

Used in Animations: 073

89 \$0059: STATE_CONTROLLED

Used in Animations: 313 314 315
400 401 402
403 417 418 422 423 426
427 428 429 430 431 432
439 443 444
461 464 465 466

STATE_constants

117 \$0075: STATE_CONTROLLED_117

Used in Animations: 412

92 \$005C: STATE_CONTROLLED_92

Used in Animations: 316

93 \$005D: STATE_CONTROLLED_93

Used in Animations: 317

94 \$005E: STATE_CONTROLLED_94

Used in Animations: Not used.

97 \$0061: STATE_CONTROLLED_97

Used in Animations: 324

95 \$005F: STATE_CONTROLLED_LET

Used in Animations: 319 320
347 348 349 350 351 352

96 \$0060: STATE_CONTROLLED_LET_96

Used in Animations: 321 322 323

73 \$0049: STATE_DASH

Used in Animations: 223 224 225

74 \$004A: STATE_DASH_DIVE

Used in Animations: 230 231 232 240 241 242
308 309

8 \$0008: STATE_DEATH

Used in Animations: 025 133 138 145 149 155 301
372 438 442 443

70 \$0046: STATE_DEATH_SLIDE

Used in Animations: 215

STATE_constants

35 \$0023: STATE_DIVE

Used in Animations: 112 113 115 119 152 154

118 \$0076: STATE_DOZY

Used in Animations:

71 \$0047: STATE_DUCK

Used in Animations: 217 222 245 259 265 274
304 305 306 307 310 311

72 \$0048: STATE_DUCK_72

Used in Animations: 218 219 220 247

105 \$0069: STATE_DUCK_LEFT

Used in Animations: 353

106 \$006A: STATE_DUCK_RIGHT

Used in Animations: 354

107 \$006B: STATE_EXTERNAL_CORNER_LEFT

Used in Animations: 355 356 363 364

108 \$006C: STATE_EXTERNAL_CORNER_RIGHT

Used in Animations: 357 358 365 366

29 \$001D: STATE_FALL_BACK

Used in Animations: 093

5 \$0005: STATE_FAST_BACK

Used in Animations: 088 089 090

53 \$0035: STATE_FAST_DIVE

Used in Animations: 153 208

STATE_constants

9 \$0009: STATE_FAST_FALL

Used in Animations: 022 023 030 032 033 036 037
045 049 083 084 085 098 318 330

20 \$0014: STATE_FAST_TURN

Used in Animations: 044 069

14 \$000E: STATE_FAST_TURN_14

Used in Animations: 378

4 \$0004: STATE_FAST_TURN

Used in Animations: Not used.

3 \$0003: STATE_FORWARD_JUMP

Used in Animations: 016 017 018 019 034 035 046
076 077
207 210 213 216

18 \$0012: STATE_GLIDE

Used in Animations: 087 198 199 200

10 \$000A: STATE_HANG

Used in Animations: 029 096 187 188 194 201 202

30 \$001E: STATE_HANG_LEFT

Used in Animations: 136

31 \$001F: STATE_HANG_RIGHT

Used in Animations: 137

82 \$0052: STATE_HANG_TURN_LEFT

Used in Animations: 271

83 \$0053: STATE_HANG_TURN_RIGHT

Used in Animations: 272

STATE_constants

132 \$0084: STATE_HEADGEAR_VCI_REMOVE

Used in Animations:

131 \$0083: STATE_HEADGEAR_VCI_USE

Used in Animations:

109 \$006D: STATE_INTERNAL_CORNER_LEFT

Used in Animations: 359 360 367 368

110 \$006E: STATE_INTERNAL_CORNER_RIGHT

Used in Animations: 361 362 369 370

27 \$001B: STATE_LEFT_JUMP

Used in Animations: 078 079

79 \$004F: STATE_MONKEY_180

Used in Animations: 257

76 \$004C: STATE_MONKEY_FORWARD

Used in Animations: 236 237 238 239 252

77 \$004D: STATE_MONKEY_LEFT

Used in Animations: 253

78 \$004E: STATE_MONKEY_RIGHT

Used in Animations: 255

75 \$004B: STATE_MONKEY_STILL_OR_HANG_SWING

Used in Animations: 150 233 234 235 254 256
283 284 285 286

STATE_constants

116 \$0074: STATE_NULL_116

Used in Animations: 399 433 434 435

19 \$0013: STATE_NULL_19

Used in Animations: 026 097 102 174 214 342 473

50 \$0032: STATE_NULL_50

Used in Animations: Not used.

51 \$0033: STATE_NULL_51

Used in Animations: Not used.

54 \$0036: STATE_NULL_54

Used in Animations: 159

62 \$003E: STATE_NULL_62

Used in Animations: Not used.

63 \$003F: STATE_NULL_63

Used in Animations: Not used.

64 \$0040: STATE_NULL_64

Used in Animations: Not used.

68 \$0044: STATE_NULL_68

Used in Animations: Not used.

69 \$0045: STATE_NULL_69

Used in Animations: Not used.

87 \$0057: STATE_NULL_87

Used in Animations: 287 288

STATE_constants

86 \$0056: STATE_ON_ALL_FOURS_BACK

Used in Animations: 275 276

81 \$0051: STATE_ON_ALL_FOURS_FORWARD

Used in Animations: 260 261 262 267

80 \$0050: STATE_ON_ALL_FOURS_STANDING

Used in Animations: 258 263 264 266 268 273
277 278 279 280 281 282

88 \$0058: STATE_ON_ALL_FOURS_TO_HANG

Used in Animations: 289 290 302

84 \$0054: STATE_ON_ALL_FOURS_TURN_LEFT

Used in Animations: 269

85 \$0055: STATE_ON_ALL_FOURS_TURN_RIGHT

Used in Animations: 270

137 \$0089: STATE_OPEN_BOX_PICKUP_VCI

Used in Animations:

128 \$0080: STATE_PB_HANGING

Used in Animations: 462 - Swing around horizontal pole

129 \$0081: STATE_PB_LEAP_OFF

Used in Animations: 463 - Jump off horizontal pole

39 \$0027: STATE_PICK_UP

Used in Animations: 130 135 291 292
419 424 425 471

98 \$0062: STATE_PICK_UP_98

Used in Animations: 325

STATE_constants

67 \$0043: STATE_PICK_UP_FLARE

Used in Animations: 204 206 312

101 \$0065: STATE_POLE_DOWN

Used in Animations: 334 335 336

102 \$0066: STATE_POLE_LEFT

Used in Animations: 332 343 344

103 \$0067: STATE_POLE_RIGHT

Used in Animations: 333 345 346

99 \$0063: STATE_POLE_STATIC_99

Used in Animations: 326 327 328 331 337

100 \$0064: STATE_POLE_UP

Used in Animations: 329 338

37 \$0025: STATE_PULL_BLOCK

Used in Animations: 122

104 \$0068: STATE_PULLEY

Used in Animations: 339 340 341

36 \$0024: STATE_PUSH_BLOCK

Used in Animations: 123

38 \$0026: STATE_PUSH_PULL_READY

Used in Animations: 120 121

11 \$000B: STATE_REACH

Used in Animations: 094 095 100 101 248 249 250 251
386 406 407 409 410 411

STATE_constants

26 \$001A: STATE_RIGHT_JUMP

Used in Animations: 080 081

23 \$0017: STATE_ROLL_23

Used in Animations: 147

45 \$002D: STATE_ROLL_45

Used in Animations: 047 048 146 151

111 \$006F: STATE_ROPE

Used in Animations: 371 373 374 377 379 408

114 \$0072: STATE_ROPE_114

Used in Animations: 387 394 396 397 398

115 \$0073: STATE_ROPE_115

Used in Animations: 395

113 \$0071: STATE_ROPE_CLIMB_DOWN

Used in Animations: 375 384 385

112 \$0070: STATE_ROPE_CLIMB_UP

Used in Animations: 376

90 \$005A: STATE_ROPE_LEFT

Used in Animations: 392

91 \$005B: STATE_ROPE_RIGHT

Used in Animations: 393

1 \$0001: STATE_RUN

Used in Animations: 000 006 008 010 043 052 055 056
092 180 181 243 244

STATE_constants

24 \$0018: STATE_SLIDE

Used in Animations: 070 071 072 246

32 \$0020: STATE_SLIDE_BACK

Used in Animations: 104 105 106

46 \$002E: STATE_SPECIAL

Used in Animations: 139

12 \$000C: STATE_SPLAT

Used in Animations: Not used.

22 \$0016: STATE_STEP_LEFT

Used in Animations: 065 066

21 \$0015: STATE_STEP_RIGHT

Used in Animations: 067 068

2 \$0002: STATE_STOP

Used in Animations: 011 014 015 024 031
042 050 051 053 054
082 099
103 148 175 184 185 221 226
227 228 229 303

47 \$002F: STATE_SURF_BACK

Used in Animations: 140 141 142

48 \$0030: STATE_SURF_LEFT

Used in Animations: 143

49 \$0031: STATE_SURF_RIGHT

Used in Animations: 144

STATE_constants

34 \$0022: STATE_SURF_SWIM

Used in Animations: 116 118

33 \$0021: STATE_SURF_TREAD

Used in Animations: 110 114 117

52 \$0034: STATE_SWAN_DIVE

Used in Animations: 156 157 158

17 \$0011: STATE_SWIM

Used in Animations: 086 109

40 \$0028: STATE_SWITCH_ON

Used in Animations: 063 129 195 197 413 414
415 416 420 470

126 \$007E: STATE_SWITCH_ON_126

Used in Animations: 460 – Examine and turn dove switch (Rome-levels)

41 \$0029: STATE_SWITCH_ON_41

Used in Animations: 064 196

127 \$007F: STATE_TIGHTROPE_REGAIN_BALANCE

Used in Animations: 453 456 (Tightrope animations)

124 \$007C: STATE_TIGHTROPE_STAND_WALK_OUT

Used in Animations: 458 (Tightrope animations)

120 \$0078: STATE_TIGHTROPE_TURN_AROUND

Used in Animations: 451 (Tightrope animations)

125 \$007D: STATE_TIGHTROPE_WALK_OFF

Used in Animations: 459 (Tightrope animations)

STATE_constants

122 \$007A: STATE_TR_FALL_122

Used in Animations: 452 454 (Tightrope animations)

123 \$007B: STATE_TR_FALL_123

Used in Animations: 455 457 (Tightrope animations)

119 \$0077: STATE_TR_POSE

Used in Animations: 447 448 449 (Tightrope animations)

121 \$0079: STATE_TR_WALK

Used in Animations: 446 450 (Tightrope animations)

13 \$000D: STATE_TREAD

Used in Animations: 107 108

7 \$0007: STATE_TURN_LEFT

Used in Animations: 013

6 \$0006: STATE_TURN_RIGHT

Used in Animations: 012

44 \$002C: STATE_UNDERWATER_DEATH

Used in Animations: 124 132

STATE_constants

130 \$0082: STATE_UNKNOWN
Used in Animations:

133 \$0085: STATE_UNKNOWN
Used in Animations:

134 \$0086: STATE_UNKNOWN
Used in Animations:

135 \$0087: STATE_UNKNOWN
Used in Animations:

136 \$0088: STATE_UNKNOWN
Used in Animations:

STATE_constants

28 \$001C: STATE_UP_JUMP

Used in Animations: 027 028 091

42 \$002A: STATE_USE_KEY

Used in Animations: 131 437

43 \$002B: STATE_USE_PUZZLE

Used in Animations: 134

65 \$0041: STATE_WADE

Used in Animations: 176 177 178 179 186

0 \$0000: STATE_WALK

Used in Animations: 001 002 003 004 005 007 009
020 021 057 058 059 060
125 126 127 128 189
293 294 295 296 297 298 299 300
380 381 382 383 388 389
390 391 398
404 405
421 436 437 440 441

55 \$0037: STATE_WATER_OUT

Used in Animations: 111 190 191 192 193

66 \$0042: STATE_WATER_ROLL

Used in Animations: 203 205

SWR_constants

1 \$0001: SWR_SUPERVISORY

Used with the **Customize=CUST_STAR_WARS_ROBOT** command.

This enables the Supervisory mode.

When the Star Wars Robot is in the Supervisory mode it tries to detect the presence of Lara and when it detects her the TriggerGroup, with **id** typed in the TriggerGroupIdOnSupervisory argument of the

Customize=CUST_STAR_WARS_ROBOT command is performed.

Note: Supervisory mode disables the **SWR_KILLING** or **SWR_HURTING** modes.

2 \$0002: SWR_HURTING

Used with the **Customize=CUST_STAR_WARS_ROBOT** command.

This enables the Star Wars Robot to injure Lara with electrical lightning. Set in the **ExtraParameter** field the damage for Lara for each frame duration of the shake.

4 \$0004: SWR_KILLING

Used with the **Customize=CUST_STAR_WARS_ROBOT** command.

This turns the Star Wars Robot into a killing machine. It shoots lightnings that will kill Lara, burning her.

Note: This is not compatible with the **SWR_SUPERVISORY** or **SWR_HURTING** flags

8 \$0008: SWR_DISABLE_INSPECTION

Used with the **Customize=CUST_STAR_WARS_ROBOT** command.

By default the Star Wars Robot, when enabled has at least one flag **SWR_SUPERVISORY**, **SWR_HURTING** or **SWR_KILLING**.

It will try to look for Lara, moving its head.
It will move up to reach a higher wall, up to one sector.

In this inspection mode the robot is slow, as it stops to look around. This skill can be disabled by adding the **SWR_DISABLE_INSPECTION** flag to the Flags field of the **Customize=CUST_STAR_WARS_ROBOT** command.

SWR_constants

16 \$0010: SWR_INJURING_ON_TOUCH

Used with the **Customize=CUST_STAR_WARS_ROBOT** command.

Adding this the Star Wars Robot will injure Lara when it touches her.

-2 \$FFFE: ANY_SW_ROBOT

Used with the **Customize=CUST_STAR_WARS_ROBOT** command.

To use in the **RobotIndex** field to set the current customize for all of the Star War Robots in the current level.

SWT_constants

1 \$0001: SWT_BASE_ZERO

Used in the Switch= command.

By default the Switch command performs the first Trigger Group in the list when the **PlaceFolderVariable** value is "1", it performs the second when the variable = 2, etc. If the first value is "0" for the first Trigger Group then use this flag.

2 \$0002: SWT_RANDOM_MODE

Used in the Switch= command.

To introduce random elements in the adventure use this to perform different triggers in a random way.

When the **SWT_RANDOM_MODE** flag is set the field **VariablePlaceFolder** is ignored and the number used to select the Trigger Group to perform is generated in a random way.

For example: Place three Trigger Group **ids** in a list,
the **TRNG engine** will generate a random number in the range (1 to 3)

(or 0 to 2 if the **BASE_ZERO** flag is set)

and then then perform the Trigger Group using the random number.

TCF_constants

2 \$0002: TCF_CREATURE

A creature is a moveable that lives and dies.
It has AI to move itself.

16 \$0010: TCF_LAND_CREATURE

4096 \$1000: TCF_ONLY_EXPLODE

This moveable can only be killed by an explosion, like a mummy or a skeleton.

8 \$0008: TCF_SAVEPOS

Save the information about the current moveable in the save game file.

TCMD_constants

2 \$0002: TCMD_EXIT

Used in the TriggerGroup= command.

This should be used as the second of three values,
where the first is the **TGROUP_COMMAND**.

The **TCMD_EXIT** command quits the execution of the current Trigger Group,
returning the value typed in the third (and next) argument.

The returned value can be **TRUE** or **FALSE**.

In conditional Trigger Groups the final result can be set.

If **TRUE** is set as the next argument the condition is true.

If **FALSE** is set as the next argument the condition is false.

Note: When the Trigger Group with the **TCMD_EXIT** command is
NOT a condition but a series of triggers to affect some effect
in the game, return a value for the third argument that is **TRUE**.

Examples:

The triple values: TGROUP_COMMAND, TCMD_EXIT, TRUE

This will quit the execution of the current Trigger Group,
returning **TRUE**. If the current Trigger Group is a condition,
the condition is **TRUE**.

The triple values: TGROUP_COMMAND, TCMD_EXIT, FALSE

This will quit the execution of the current Trigger Group,
returning **TRUE**. If the current Trigger Group is a condition,
the condition is **FALSE**.

TCMD_constants

1 \$0001: TCMD_GOTO

Used in the TriggerGroup= command.

This is the second of three values.

The first value must be the **TGROUP_COMMAND** constant.

The **TCMD_GOTO** forces a jump to another trigger in the same Trigger Group command.

The third value is the **index** of the trigger (group of three values) to perform.

Example:

```
TriggerGroup= 1, $5000, 118, $2A,           ;0 index
               TGROUP_COMMAND, TCMD_GOTO, 3 ;1 index, jump to trigger "$2000, 406, $1"
               $5000, 137, $0129,           ;2 index
               $2000, 406, $1                ;3 index
```

Using **GOTO** makes it easier to manage Trigger Groups with a complicated sequence of **IF ELSE** conditional groups.

For instance in natural language, a sequence of conditions and commands in a Trigger Group:

```
TriggerGroup=id, ACondition           ; 0
                        TriggerA1      ; 1
                        TriggerA2      ; 2
                        ELSE BCondition ; 3
                        TriggerB1      ; 4
                        TriggerB2      ; 5
                        TriggerC1      ; 6
                        TriggerC2      ; 7
                        TriggerC3      ; 8
```

Above is a symbolic description of a Trigger Group.

Where you read **Condition** there are three values for a exported conditional trigger.

Where you read **Trigger** there are three values for a exported trigger.

Now to perform **TriggerA1 TriggerA2** when **ACondition** is true
and **TriggerB1 TriggerB2** when **BCondition** is true

Say, we want to perform the **TriggerC1 TriggerC2 TriggerC3**, in all cases after
TriggerA1 Trigger A2 and after **TriggerB1 TriggerB2**.

TCMD_constants

In the last release of the **TRNG** it was possible in only one way:
duplicating the **TriggerC1 TriggerC2 TriggerC3** in this way:

```
TriggerGroup=Id, ACondition ; 0
    TriggerA1 ; 1
    TriggerA2 ; 2
    TriggerC1 ; 3
    TriggerC2 ; 4
    TriggerC3 ; 5
ELSE BCondition ; 6
    TriggerB1 ; 7
    TriggerB2 ; 8
    TriggerC1 ; 9
    TriggerC2 ; 10
    TriggerC3 ; 11
```

It happened because after a condition, ONLY the triggers after that condition are performed and the first of the next ELSE statement.

When the execution entered in **ACondition** is true the execution stops at the next found ELSE.

Using a **GOTO** command can get over this limit in this way:

```
TriggerGroup=Id, ACondition ; 0
    TriggerA1 ; 1
    TriggerA2 ; 2
    COMMAND GOTO 7 ; 3
ELSE BCondition ; 4
    TriggerB1 ; 5
    TriggerB2 ; 6
    TriggerC1 ; 7
    TriggerC2 ; 8
    TriggerC3 ; 9
```

In the above when the **ACondition** is true,
TriggerA1 TriggerA2 will be executed and then the **GOTO**
also the **TriggerC1 TriggerC2 TriggerC3** avoiding to have to duplicate them.

From the above example the **GOTO** command is often used as the first of an **ELSE** statement,
that executes other triggers below in the Trigger Group.

There is another new approach that can be used with the **GOTO** command.

In past **TRNG** versions, the execution of triggers, inside a given Trigger Group, was only from
top (first row) to bottom (last row) with skipping of some triggers according to the result of some
conditions.

TCMD_constants

Now, using **GOTO** it is possible to newly perform a previous trigger generating a loop. When this is done take care to avoid endless loops otherwise the game will freeze.

There are two situations to use loops (**GOTO** commands) with no trouble:

Situation 1: When a **TRNG** variable is used to keep track of a number of cycles (amount of times) that some triggers are performed.

In this method insert in the cycle a condition to verify if it is the moment to quit the loop. This method is used to manage the arrays (vectors).

A small (meaningless) example:

```
TriggerGroup= Id, Set A1=0                ; 0
      TriggerAlfa                        ; 1
      TriggerBeta                        ; 2
      TriggerDelta                       ; 3
      A1=A1+1                            ; 4
      If A1 less or even than 4          ; 5
          COMMAND GOTO 1                 ; 6
      ELSE TriggerGamma                  ; 7
          TriggerOmega                    ; 8
```

The above example is a loop, where the group of triggers (**Alfa**, **Beta** and **Delta**) are executed four times. At the end of the cycle, the **Trigger Gamma** and **Trigger Omega** are performed.

If you follow the execution and keep in mind the value of A1, modified by $A1=A1+1$ This is then checked by the **IF A1 is less than or equal to 4 condition**.

Note: Do not try to use a loop to move items on the screen, or to wait for a different condition different of **TRNG** variables or memory zones changed in a loop otherwise the game will freeze.

The right way to continuously perform a Trigger Group is to use a trigger in the level to always activate the Performance.

Trying to get the same result with a cycle based on a **GOTO** command will fail because the game requires to draw meshes, read input commands from the keyboard to work. **It can not be stopped in an endless cycle inside a Trigger Group.**

Situation 2: When you want to recycle a series of triggers typed in a huge Trigger Group.

This situation is like that already shown to skip the **ELSE** statement. The only difference is that a **GOTO** could also be used to come back to some previous trigger block.

TCMD_constants

8 \$0008: TCMD_LOG

Used in the TriggerGroup= command.

This should be used as the second of three values.
Where the first value is the **TGROUP_COMMAND**.

This is only used for script debugging.

Enable or disable the log like you had changed the script command:

DiagnosticType= DGX_LOG_SCRIPT_COMMANDS, EDGX_CONCISE_SCRIPT_LOG

In the third argument type **ENABLED** to enable the log, or **DISABLED**, to disable the log.

Note that this command is not able to enable or disable the diagnostics but only the **DGX_LOG_SCRIPT_COMMANDS** flag.

Use it in this way:

Type in the [Options] section of the script the commands:

Diagnostic= ENABLED
DiagnosticType= 0, EDGX_CONCISE_SCRIPT_LOG

Note: The **DGX_LOG_SCRIPT_COMMANDS** flag can be omitted.

In the Trigger Group that you want to have a log type as the first trigger:

TGROUP_COMMAND, TCMD_LOG, ENABLED

and at end of the Trigger Group disable the log:

TGROUP_COMMAND, TCMD_LOG, DISABLED

In this way you can remove most of the log messages about script commands that you are not interested in to clean up the log.

Note: This command can not remove the system log messages about loading a level, creation of lights etc.
It affects only log messages about script commands.

TCMD_constants

9 \$0009: TCMD_PAUSE

Used in the TriggerGroup= command.

This should be used as the second of three values.
Where the first value is the **TGROUP_COMMAND**.

This is only for script debugging.

This can force a pause in the game when this command trigger is performed.
The only reason to use this is to verify by watching the screen what is happening in the game in that precise moment.

There are two ways to perform the pause:

Set in the third value of the command trigger the number of microseconds for the pause.
After this elapsed time the game starts to run.

Set a scan code.
When a scan code is set the pause goes on until the user hits that key.

To recognize **scan codes** from **microseconds** use the sign.

Negative numbers are seen as **microseconds** to wait
Positive numbers are seen as a **scan code** to be hit to quit the pause.

6 \$0006: TCMD_SET_EXTRA_CONDITION

Used in the TriggerGroup= command.

This is used as the second of three values.
Where the first value is the **TGROUP_COMMAND**.

It changes the following (real) trigger of the current Trigger Group.
It replaces the **Extra Timer** with a **TRNG** variable set in the third value of the current command trigger.

The Range is 0 to 31
It is only Used with a Conditional trigger. It is the (E) Extra parameter.

TCMD_constants

5 \$0005: TCMD_SET_EXTRA_TIMER

Used in the TriggerGroup= command.

This is the second of three values.

Where the first value is the **TGROUP_COMMAND**.

It changes the following (real) trigger of the current Trigger Group.

It replaces the **Extra Timer** with a **TRNG** variable set in the third value of the current command trigger.

The Range is 0 to 127

It is used with Flip effects and Action triggers. It is the (E) Extra parameter.

First Example: Change the time of self-closing of a door,
in accordance with the number of found secrets.

In the game play this means: Look for and find secrets and at the end of the level you will have more time to reach the final exit-door.

The Trigger Group to realise this is based on a trigger to modify dynamically with the **TGROUP_COMMAND** like this:

```
; Set Trigger Type - ACTION 43  
; Exporting: TRIGGER(1323:0) for ACTION(175) {Tomb_NextGeneration}  
; (#) : DOOR_TYPE1 ID 175 in sector (2,2) of Room 0  
; (&) : Trigger. (Moveable) Activate (#)Object with (E)Timer value  
; (E) : Timer= +05  
; Values to add in script command: $5000, 175, $52B
```

The trigger will open the Door and then wait 5 seconds before closing it.

The first exported trigger "\$5000, 175, \$52B", used a command trigger:

TGROUP_COMMAND, TCMD_SET_EXTRA_TIMER, #0800

Change the (E)timer value of the following trigger using the value stored in the Current Value variable (#800).

Before doing this we have to use **TRNG** variables to read the amount of secrets and then multiply it by some factor and store this value in the **Current Value** variable. This is used to modify the number of seconds to keep the door open.

TCMD_constants

Second Example: Building a new custom elevator.

Use the **InputBox** to receive a number (number of floor, for instance).
Multiply the value (stored in the **LastInputNumber** variable) by some factor.

Then use the action trigger:

```
; Set Trigger Type - ACTION 30  
; Exporting: TRIGGER(30:0) for ACTION(4) {Tomb_NextGeneration}  
; (#) : DOOR_TYPE1 ID 4 in sector (3,3) of Room0  
; (&) : Move. Move up (#)animating for (E) clicks  
; (E) : 1 clicks  
; Values to add in script command: $5000, 4, $1E
```

Use a bridge as animating to move.

In the Trigger Group set the **TGROUP_COMMAND** to change the Extra timer of the trigger using the variable saved in the computation:

LastInputNumber * (some factor, the number of clicks for the floor).

4 \$0004: TCMD_SET_FULL_TIMER

Used in the TriggerGroup= command.

This is the second of three values.

Where the first value is the **TGROUP_COMMAND**.

It changes the following (real) trigger of the current Trigger Group.

It replaces the **Full Timer** with a **TRNG** variable set in the third value of the current command trigger.

The Range is 0 to 32767

This is only used with Flip effects having NO (E) Extra parameter.

It is the "Timer (Parameter (&))" parameter.

TCMD_constants

7 \$0007: TCMD_SET_OBJECT

Used in the TriggerGroup= command.

This is the second of three values.

Where the first value is the **TGROUP_COMMAND**.

It changes the following (real) trigger of the current Trigger Group.

It replaces the **Object value** with a **TRNG** variable set in the third value of the current command trigger.

The Range is 0 to 4095 or 0 to -4095 (see note).

This is used with Condition and Action triggers.

It is the "(Object to trigger (#))" parameter.

Note: If the value is a moveable index,
remember that there are two kinds of indices about moveables:

The index in the tr4 file used in run-time.

The NGLE index, that is the index you see in NGLE room editor.

Use the negative sign (-) to declare a tr4 moveable index.

The range is -1 to -4096

If the value of the source variable is positive, it should be a tomb (tr4) moveable index or another kind of parameter different from moveable indices.

TCMD_constants

3 \$0003: TCMD_SET_TIMER

Used in the TriggerGroup= command.

This is the second of three values.

Where the first value is the **TGROUP_COMMAND**.

It changes the following (real) trigger of the current Trigger Group.

It replaces the **Timer value** with a **TRNG** variable set in the third value of the current command trigger.

The the Range is 0 to 255

This is only used with Flip effects.

It is the "Timer (Parameter (&)) parameter.

TGROUP_constants

8 \$0008: TGROUP_AND

Used in the TriggerGroup= command.

Set a operator for the current condition with the **AND** operator.

This operator will be kept for the following condition trigger if no other **TGROUP_** flag for boolean operators is to be used.

The **AND** operation is the default operation if no **TGROUP_** flags set operators in the Trigger Group.

That is all conditions will be linked with **AND** operators.

When other operators like **OR** or **NOT** are set it could be necessary use the

TGROUP_AND flag to force a new **AND** operator to link the different conditions.

Example of list of conditions + boolean operators:

Condition1

TGROUP_OR	+	Condition2
TGROUP_AND	+	Condition3
TGROUP_NOT	+	Condition4

The above list of condition triggers is read in this way:

If (Condition1 OR Condition2) is **TRUE AND**

Condition3 is **TRUE AND**

Condition4 is **NOT TRUE** then

perform the following (non-conditional) trigger sequence.

TGROUP_constants

3 \$0003: TGROUP_COMMAND

Used in the TriggerGroup= command.

Different from other **TGROUP_** values, the **TGROUP_COMMAND** value should be used alone(*) to introduce a fake trigger.

Usually in Trigger Groups there is data about real triggers exported, but in the case of **TGROUP_COMMAND** it is only used to set a command to handle the execution of other triggers in the current Trigger Group.

Once the first of three values is typed in the **TGROUP_COMMAND** set in the second value a **TCMD_** to set what kind of command it is.

The third value is the argument for a **TCMD_** command.

See the description of the **TCMD_** for more information.

Note: The only exception to use the **TGROUP_COMMAND** alone is for the **TGROUP_ELSE**. That is to add to the **TGROUP_COMMAND** when it is the first trigger of an **ELSE** block.

No other **TGROUP_** can be added to the **TGROUP_COMMAND**.

In this case use a **TGROUP_COMMAND** with the **TCMD_SET_** command to modify a condition trigger.

The **TGROUP_OR** **TGROUP_AND** **TGROUP_NOT** operators should always be added to real (following) condition triggers, never to a **TGROUP_COMMAND** fake trigger.

TGROUP_constants

64 \$0040: TGROUP_ELSE

Used in the Trigger Group command.

Set the start of a **ELSE** group in the current Trigger Group.

The **ELSE** is important to perform an alternative trigger when the first condition was **FALSE**.

For example: Create a Trigger Group with conditions to load a different level according to some condition, like the following:

```
Condition1 is TRUE
Perform trigger "Load level 4"
ELSE
Perform trigger "Load title level"
```

The above conditional sentence could be used to perform a Bonus level (4) only when some condition (condition1) is TRUE,

while if the condition is FALSE (ELSE) it will load the title level,

to complete the adventure with no Bonus level.

Place one, two or more **ELSE** in the Trigger Group.

The rule to follow to understand how the Trigger Group will work is:

If the first conditions are TRUE:

perform the first group non conditional trigger
following the first conditions.

If the first conditions are FALSE,
skip all the following non conditional triggers,
until an **ELSE** flag is reached.

If an flag has been found,
the parsing will start from that condition /trigger
as if it was the start of the whole Trigger Group.
The trigger with the **ELSE** flag could also be another condition.

If the conditions are TRUE perform the following non conditional triggers.
If the conditions are FALSE, skip all triggers up to another **ELSE** flag.

Using multiple **ELSE** to perform different triggers
according to different conditions.

TGROUP_constants

For example: Open different doors according to the last number typed in the Keypad switch:

Condition: If KeyPad Value = 1

Trigger: Open door 12

ELSE

Condition: If KeyPad Value=2

Trigger: Open door 7

ELSE

Condition: If keypad Value =3

Trigger: Open door 18

TGROUP_constants

32 \$0020: TGROUP_NOT

Used in the TriggerGroup= command.

Set as an operator to link the current condition with previous conditions use the **NOT** operator.

The **NOT** operator is read as "AND NOT" because it requires the previous condition is **TRUE** and that the current condition is **FALSE**.

Use the **NOT** operator to invert a condition.

This is very useful to create a condition not in the Trigger Condition list.

For example: If there is in the trigger window the condition
Lara collide with object <# but it is missing the inverse condition
Lara is NOT colliding with object <# can be used,
and then add a **TGROUP_NOT** flag to the first value of that exported
condition to transform it into **Lara is NOT colliding with object**.

See the description of the **TGROUP_AND** flag for more information about the use of boolean operators in the Trigger Group.

16 \$0010: TGROUP_OR

Used in the TriggerGroup= command.

Set as an operator to link the current condition with previous conditions using the **OR** operator.

When the **OR** operator is used the previous condition
or the current condition is **TRUE** to get a **TRUE** condition.

See the description of the **TGROUP_AND** flag for more information about the use of boolean operators in the Trigger Group.

TGROUP_constants

1024 \$0400: TGROUP_SINGLE_SHOT

Used in the TriggerGroup= command.

Add this flag in the first number of the first trigger of the Trigger Group to create a single-shot Trigger Group.

This Trigger Group will only be performed once in the current level.

The single-shot trigger groups are very important to fix the limitation about the trigger zone where the same trigger is placed in two or more closed sectors in the map.

When all of this trigger zone works like a single-shot trigger there is a problem. Each trigger will work in a one-shot way but individually and this means that Lara could engage the same trigger when she touches another sector in that trigger zone.

Currently the only way to avoid this problem is to export the trigger in script format and copy it in a Trigger Group command.

Add to the first number the **TGROUP_SINGLE_SHOT** flag and at the end replace in the level the original trigger with a "perform trigger group" flip effect.

The Trigger Group will only be performed once and when Lara passes over another trigger sector of that same zone nothing will happen.

Remark: There is also a new flip effect to use when a new single-shot Trigger Group already performed can be reset.

TGROUP_constants

8 \$0008: TGROUP_SINGLE_SHOT_RESUMED

Used in the TriggerGroup= command.

This flag works like the **TGROUP_SINGLE_SHOT** flag (see its description) but in this case it is not necessary to use the flip effect 345 to enable it again.

A Trigger Group with the **TGROUP_SINGLE_SHOT_RESUMED** in (first) flags will be enabled again ready when the current Trigger Group finishes.

This means that the "single shot" attribute will not only work in continuous triggered sectors but when Lara goes out from this trigger zone the trigger will be ready for further actions.

256 \$0100: TGROUP_USE_EXECUTOR_ITEM_INDEX

Used in the TriggerGroup= command.

Add this flag to the first triplex parameter of the exported trigger data.

This will replace the index that performs some action with the item that executes the trigger and enables the Trigger Group.

Use this flag when a heavy trigger is used to perform the Trigger Group and all of the Action Triggers will be redirected to a moveable that has enabled the Trigger Group.

For example: Export an Action Trigger to kill a moveable and then perform this Trigger Group with an heavy trigger.

When any moveable moves on it the action (to kill) will be applied on this moveable.

Practically with this method create a trigger to kill enemies (or another action on them) where it will be the same moveable to enable it.

Remark: To do this trick use an Enemy= command with the **NEF_EASY_HEAVY_ENABLING** to permit different moveables to enable all heavy triggers they meet in their path.

TGROUP_constants

1 \$0001: TGROUP_USE_FOUND_ITEM_INDEX

Used in the TriggerGroup= command.

Used to force the use of the index of the last moveable detected by some condition.
The found index will be used in all of the following triggers or conditions of the Trigger Group that require an index for the moveable.

Using this flag the index found from a condition will become the index to use for the following condition and triggers, ignoring the further moveable index for those triggers.

This overriding of original indices will only be stopped when it is found that another **TGROUP_** flag is used to change the default index of the moveable to use.

Other **TGROUP_** flags to change the default index for moveable are:

TGROUP_USE_OWNER_ANIM_ITEM_INDEX
TGROUP_USE_TRIGGER_ITEM_INDEX

See the descriptions for the above flags.

Remark: Not all condition trigger may be used to set a moveable index.

Add the flag **TGROUP_USE_FOUND_ITEM_INDEX**
to the first value of the exported condition.

Only verify the presence of some moveable on the condition.

For example: The condition trigger
Lara collides with moveable with [&] slot or the ENV condition
Lara has in front the object with slot = ExtraSlot

This forces another index to use the following triggers to handle the situation where it is not known WHAT moveable Lara will collide with or have in front of her.

For example: Build a Trigger Group to handle fighting:

Conditions: **When right hand of Lara touches head of [&] slot moveable**

Then: (above condition is **TRUE**) Toggle vitality of the current enemy moveable.

In the above Trigger Group,
when the condition trigger is built set the slot of the moveable
for the condition.

TGROUP_constants

For example:

Use the slot **95 SAS**,
In the game Lara hits a SAS moveable
it could be one of many **SAS** in the level
and its index is unknown when the condition trigger is built.

For an unknown index it is not possible to toggle the vitality.

To solve the problem there are some **TGROUP_** flags used
get the index of the moveable to use in a dynamic way:

When the **TRNG** engine detects a moveable
with some condition it uses that index for the following trigger.

TGROUP_constants

2048 \$0800: TGROUP_USE_ITEM_USED_BY_LARA_INDEX

Used in the TriggerGroup= command.

Use this to override the original item index of the exported Action Trigger with the index of the last item that Lara used or is using in the interactive mode.

4 \$0004: TGROUP_USE_OWNER_ANIM_ITEM_INDEX

Used in the TriggerGroup= command.

Set the moveable index to use in the following and current triggers, the index of the moveable animation command performed.

This flag could only be used when the current Trigger Group command has been started by an animation command.

In this situation the flag for the index of the moveable to use will be the index of the moveable that owns the animation that started the current Trigger Group.

See the description of **TEGRUP_USE_FOUND_ITEM_INDEX** flag f or more information about the mechanism of changing of moveable index.

2 \$0002: TGROUP_USE_TRIGGER_ITEM_INDEX

Used in the TriggerGroup= command.

Set a index of the moveable to use for the following and current triggers, the real source index set in the exported trigger.

This flag only requires to be used when the default index is changed in previous triggers or conditions.

If no **TEGRUP_** flag is set to modify the index to use it is not necessary use this flag.

See the description for the **TEGRUP_USE_FOUND_ITEM_INDEX** flag for more information about the mechanism for changing the moveable index.

TPOS_constants

1 \$0001: TPOS_DOUBLE_HORIENT

Used in the **TestPosition=** command.

To detect that the correct position could be in front of the object (with given H Orient) or on the opposite side.

When in the **TestPosition** there is the **TPOS_DOUBLE_HORIENT** flag.

The **TestPosition** condition will be **TRUE** if Lara is on the opposite side from that set in the range of **TestPosition**.

For example: For an item like a gate, use the **TPOS_DOUBLE_HORIENT** to allow Lara to be in a correct position for both sides of the gate.

Remark: This flag only works if the item has its pivot in the same position of the visible mesh.

Unfortunately, in many circumstances the pivot (origin x,y,z of the object) is not at the centre of its mesh but in a corner of the sector where it will be placed.

8 \$0008: TPOS_FAST_ALIGNMENT

Used in the **TestPosition=** command.

This flag affects the alignment phase when the **FAN_ALIGN_TO_ENV_POS** constant is used in the Animation command that called the **TestPosition** command.

Use an Animation command and use the **FAN_ALIGN_TO_ENV_POS**.

When Lara is in a good position for the **TestPosition** command she will be moved to an ideal position before performing the animation.

The old method used to move Lara to an ideal position had a problem of a continuous loop when she was too near to the target position.

To solve this problem in the **TestPosition** command use the **TPOS_FAST_ALIGNMENT** flag and Lara will be moved to the correct position avoiding the risk of looping.

Remark: Use the fast alignment only when a small tolerance is set in the **TestPosition** command,
i.e. when the condition of **TestPosition** is **TRUE** and Lara is very near to an ideal position.

TPOS_constants

2 \$0002: TPOS_FOUR_HORIENT

Used in the TestPosition= command.

The correct position is set for each side of the object.

For example: A square pushable object,
where Lara is able to interact with every side of the pushable.
To work it is necessary that the object has its pivot
origin x,y,z of object in the centre position of the object mesh.

16 \$0010: TPOS_OPPOSITE_FACING

Used in the TestPosition= command.

The alignment with some objects could fail because the main mesh has an opposite facing with respect to the animation 0 displacement.

Discover this situation with the Animation Editor of **Wad Tool**:

When the position of the mesh with "No Animation" is the opposite
of the mesh in "Animation 0" the self alignment could fail.

To fix this bug use the **TPOS_OPPOSITE_FACING** flag in the **TestPosition** command
used by the Animation command.

TPOS_constants

32 \$0020: TPOS_ROUND_HORIENT

Used in the **TestPosition=** command.

This works ignoring the facing of the object but it only checks Lara's facing.

To understand the situation think about the alignment of Lara with the pole-rope.
In this case it is not important for the facing of the pole-rope but only that Lara is looking at it at the correct distance.

Since the round facing is very particular there are some special rules and limitations:

The couple (XDistanceMin / XDistanceMax) and (ZDistanceMin / ZDistanceMax) should have the same values.

The **TRNG** engine will only read the pair (ZDistanceMin / ZDistanceMax).

The range (HOrientDiffMin / HOrientDiffMax) is the only orienting pair to be verified.

If this flag is used in a **TestPosition** command with an Animation command having the **FAN_ALIGN_TO_ENV_POS** flag, Lara will be moved to an ideal position, but only in an immediate way (no sliding or smoothed movement).

It is better to use limited ranges for Max / mix distance and DiffMin / DiffMax horizontal orient to avoid a jerk movement in the game.

TPOS_constants

256 \$0100: TPOS_SELF_FIXING

Used in the **TestPosition=** command.

The self fixing works to recognize a problematic situation and it tries to fix it itself. Technically, the problematic situation is when the difference between the HOrient of Lara and that of Item is about \$8000.

The problem is that, in short values (the way tomb4 manages the H Orient values) the \$7fff value is a large positive number (+32767), the \$8000 value is a small negative number (- 32768).

This situation creates the paradox that, the "hOrientDifMin" value, set in the script command will be higher than "hOrientDifMax". This is bad and the tomb4 procedure returns false. The self-fixing recognizes this situation and tries to fix it.

4 \$0004: TPOS_TEST_ITEM_INDEX

Used in the **TestPosition=** command.

Add this value in the Flags of the **TestPosition** to change the mean of the **SlotMoveable** field.

Use the **TPOS_TEST_ITEM_INDEX** flag and the **TRNG** engine will consider the value in the **SlotMoveable** field like an index of the item to check.

Find the index in the **NGLE** program by clicking on the item and see the index in a yellow frame. The advantage to using an index is that it can test a moveable if it is external to Lara's room.

In other cases (checking for slot) the **TestPosition** only detects items in the same room as Lara. The disadvantage to using this flag is that it cannot detect all objects in a general way.

TPOS_constants

128 \$0080: TPOS_TURN_FACING_180

Used in the TestPosition= command.

This flag fixes a problem with some items.

In some circumstances an item has a wrong facing, turned by 180 degrees.
Discover this situation by enabling the self-alignment feature to see that Lara moves into the wrong position.

The only way to discover if this flag fixes the problem is trying to use it and seeing the final movement of Lara.

Note: This flag is different from the **TPOS_TURN_FACING_90** flag and will only affect the alignment phase, it will be ignored in the Test Position phase.

40 \$0028: TPOS_TURN_FACING_90

Used in the TestPosition= command.

This flag fixes a problem with some items.

In some circumstances an item has a wrong facing, turned by 90 degrees.
Discover this situation by enabling the self-alignment feature and see that Lara moves into the wrong position.

The only way to discover if this flag fixes the problem is trying to use it and seeing the final movement of Lara.

TRB_constants

16 \$0010: TRB_ADAPTIVE_FARVIEW

Used in the Turbo= command.

This enables a complex compute to adapt the maximum far view (**LevelFarView**) in the current level in accordance with the current performance.

When the **TRNG** discovers that it is not able to keep the medium value of 30 fps (Frames per second) it will progressively reduce the **LevelFarView** until it can keep 30 fps.

When it reaches the optimal frame rate and keeps it for three seconds it will try to increase the **LevelFarView** very slowly to reach the value of the **LevelFarView** set in the **script.dat** file.

This method allows a variable distance to show far objects to always preserve a good frame rate.

Remark: The **TRB_ADAPTIVE_FARVIEW** setting could not work with the **TRB_ASYNC_FRAMES** setting because this setting creates an irregular response in the internal frame rate.

The **TRB_ADAPTIVE_FARVIEW** uses its own frame rate value to perform the changes in the far view it could be confused by the **TRB_ASYNC_FRAMES**

To use the **TRB_ADAPTIVE_FARVIEW** it is advisable not to exaggerate the setting for the **LevelFarView** in the level.

Many level builders set the **LevelFarView** = 127 because it is the maximum value. This number is huge because it is very rare that more than 60 sectors are visible in a direct line.

By setting 127 when a bad frame rate happens the **TRNG** engine will spend time decreasing the far view from 127 sectors to a meaningful value like 20 or 30 sectors.

When the far view has been dramatically reduced and then recovers the **TRNG** engine will spend time to increase the far view to 127 sectors.

An ideal value for adaptive FarView is to set the LevelFarView = 60 while the required fps to set is (about) 24 fps.

TRB_constants

4 \$0004: TRB_ASYNC_FRAMES

Used in the Turbo= command.

This removes the double synchronization of the frames keeping only a single synchronization.

This setting could be an advantage to the game engine with some modern video cards. It permits some time gain with the fast frame to invest in the following frames requiring a longer time.

When using this setting the fps value seen on the screen in the Diagnostic mode is not very meaningful because the fps are very high (for example 80 fps) the game will always be in the correct limits to avoid fast movement.

Discover the advantage of this setting by looking at the game to recognize a more fluid movement.

Remark: This setting does not work in the Title level because the input synchronization for moving in the title menu does not allow the use of an asynchronous frame rate.

From 1.2.2.7 version, this setting works in a different way.

Now when the game loses some frames, the turbo command performs two or more frames in a shorter time to recover the previously lost frame.

The result in this situation is that Lara will cover the distance in the same time as a full frame rate but the movements will not always be fluid.

In spite of this solution not being perfect it is better than the slowdown seen when the engine is not able to support the standard 30 frames per second.

TRB_constants

2 \$0002: TRB_HIGH_PRIORITY

Used in the Turbo= command.

This sets a higher class priority for the whole tomb4 process to increase the time supplied to tomb4 with respect to the other processes.

Theoretically this setting could be an advantage in the game engine when there are other programs working or some background services like anti-virus.

8 \$0008: TRB_OPTIMIZE_SORTING

Used in the Turbo= command.

This optimizes the sorting of the polygons list used by the **TRNG** engine before performing draw operations.

Usually this optimizing gives a gain of about 5 or 6 fps (frames per second)

1 \$0001: TRB_SELECTIVE_VIEW

Used in the Turbo= command.

This excludes all computes about static objects not visible in the game because they are behind the current camera view.

Theoretically the DIRECTX functions perform this cut off, but looking at the source code it appears that many computes have been performed in spite of these statics objects being outside the visible world.

This setting allows time to be saved that would otherwise be wasted in useless computes.

TRUE_constant

1 \$0001: TRUE

TS_constants

6 \$0006: TS_AFTER_FALLING_AS_INVULNERABLE

Used with the Customize=CUST_SFX command.

When Lara is invulnerable and she falls from a great height, she will hit the floor not die and come back to a stand up position after a short animation.

13 \$000D: TS_ANIMATING_DOOR_CLOSE

Used with the Customize=CUST_SFX command.

When an animating (fake) door for the elevator is used this is the sound made when it closes.

12 \$000C: TS_ANIMATING_DOOR_OPEN

Used with the Customize=CUST_SFX command.

When an animating (fake) door for the elevator is used this is the sound made when it opens.

19 \$0013: TS_BINOCULAR_LIGHT

Used with the Customize=CUST_SFX command.

This is a single-shot sound played when the player switches the illumination on or off for the binoculars.

Default value is 369 LARA_CLICK_SWITCH

18 \$0012: TS_BINOCULAR_ZOOM

Used with the Customize=CUST_SFX command.

This is the sound when the customized binoculars is zooming in or out.

Default value is 309 MAPPER_MOVE

9 \$0009: TS_DAMAGE_ROOM_BEEP_ALERT

Used with the Customize=CUST_SFX command.

When in a damage room the time (bar) is almost terminated the beep sounds for the blinking bar ending time.

8 \$0008: TS_DAMAGE_ROOM_SCREAM_BURNING

Used with the Customize=CUST_SFX command.

When in a damage room and the bar is empty Lara will burn and scream.

14 \$000E: TS_DETECTOR_SHOW

Used with the Customize=CUST_SFX command.

When the detector is shown on the screen for the first time.

TS_constants

1 \$0001: TS_DIARY_CHANGE_PAGE

Used with the Customize=CUST_SFX command.

This sound is played when the user changes the page of the Diary to the next or previous page.

2 \$0002: TS_DIARY_NO_PAGE

Used with the Customize=CUST_SFX command.

This sound will be played when the player tries to change a page of the Diary but there are no more pages in that direction (next or previous).

3 \$0003: TS_DIARY_ZOOM_START

Used with the Customize=CUST_SFX command.

This sound is played when the zoom effect is enabled at the first activation of the Diary.

15 \$000F: TS_ELEVATOR_SQUASHED_LARA

Used with the Customize=CUST_SFX command.

This sound is played when an elevator has squashed and killed Lara.

11 \$000B: TS_MISSING_REQUIRED_ITEM

Used with Customize=CUST_SFX command.

This sound is played when the trigger

Inventory-Item.Pop up inventory screen to select the <& Item

has been performed and the required item is missing. (by default Lara says "No").

16 \$0010: TS_MIST_EMITTER_WITH_OCB

Used with the Customize=CUST_SFX command.

This is the sound of the Mist emitter but only when it is customized with an OCB value different from 0.

TS_constants

17 \$0011: TS_PUSHED_ITEM_IMPACT

Used with the Customize=CUST_SFX command.

This is the sound that is played when an item has been pushed away from the bike and it hits the wall or another object.

The default value is 72 GENERIC_HEAVY_THUD

7 \$0007: TS_SAVEGAME_PANEL_SELECTED

Used with the Customize=CUST_SFX command.

This is the sound played when the user changes the selected save game in a customized save game panel.

TS_constants

4 \$0004: TS_SCREENSHOT_CAPTURE

Used with the Customize=CUST_SFX command.

This sound is played when the **F3 key is pressed** to capture the game screen image.

20 \$0014: TS_SHOT_HARPOON_UW

Used with the Customize=CUST_SFX command.

This is a single-shot sound that will be played when the crossbow is customized as a harpoon-gun.

The sound will only work when Lara is underwater and she has a harpoon.
The preset value for this sound is **68 PENDULUM_BLADES**

5 \$0005: TS_VIBRATE_RESUME_FROM_FROZEN

Used with the Customize=CUST_SFX command.

This is the short sound, repeated many times,
when an enemy previously frozen is thawed out

10 \$000A: TS_WHIRLPOOL_SUNKED_LARA

Used with the Customize=CUST_SFX command.

The sound is played when Lara and the boat is sunk in the whirlpool.

TSB_constants

1 \$0001: TSB_MATRIX

Used in the StandBy= command.

The matrix effect creates a camera turning around Lara.

0 \$0000: TSB_NO_CHANGE_CAMERA

Used in the StandBy= command.

This standby type disables any changes for the Look camera.
Only use this when the Trigger Group is used to customize the Standby mode.

3 \$0003: TSB_PANORAMA

Used in the StandBy= command.

The Panorama type works like a matrix type but the distance, the vertical angle and the speed of rotation will continue to change in a random way.

Create a large range for distance.

For the Matrix type the maximum difference between the minimum and maximum limits are +/- 50% of the given distance.

In the Panorama type some settings for the StandBy command to be interpreted in a particular way:

The further settings for the **FSB_FLIP_DISTANCE**, **FSB_FLIP_SPEED** and **FSB_FLIP_V_ANGLE** flags are ignored, because the Panorama type flips the distance and the vertical angle.

The value in the **Distance** field is used like a Maximum Distance, the minimum distance will always be 400 (near to Lara).

The value in the **VAngle** field is used like the allowed **MaxVerticalAngle**.

Remember that negative numbers in **VAngle** mean "the camera is above and looking at Lara".
When a large maximum distance is set avoid large positive values for **VAngle** because the risk is that the camera tries to go underground.

The only exception is when Lara is on the top of a pyramid where the ground around her is lower than her position.

2 \$0002: TSB_PORTRAIT

Used in the StandBy= command.

In the Standby the camera remains in front of Lara.

In this mode use a short distance value to look at the beauty of Lara's eyes.

TSCR_constants

2 \$0002: TSCR_LOAD_GAME

Used with the GT_TITLE_SCREEN Global Trigger.

Detects the [Load Game] screen.

0 \$0000: TSCR_MAIN_TITLES

Used with the GT_TITLE_SCREEN Global Trigger.

Detects the Main Title screen,
i.e. the screen where the "New Game", "Load Game", "Options" and "Exit" is displayed.

1 \$0001: TSCR_NEW_GAME

Used with the GT_TITLE_SCREEN Global Trigger.

Detects the [New Game] screen, where a specific level name can be selected.

3 \$0003: TSCR_OPTIONS

Used with the GT_TITLE_SCREEN Global Trigger.

Detects the [Options] screen

TT_constants

1 \$0001: TT_ACTION_INVENTORY_MENU_OFF

Used in the Customize=CUST_SET_TEXT_COLOR command.

Switch off the Inventory Menu, Combine, Examine, Separate etc.

2 \$0002: TT_ACTION_INVENTORY_MENU_ON

Used in the Customize=CUST_SET_TEXT_COLOR command.

Switch on the Inventory menu, Combine, Examine, Separate etc.

3 \$0003: TT_AMMO

Used in the Customize=CUST_SET_TEXT_COLOR command.

Text for the ammo quantity.

27 \$001B: TT_CAMERA_VIEW

Used in the Customize=CUST_SET_TEXT_COLOR command.

Text shown when the user hits the F1 key to see the LoadCamera values in the game.

13 \$000D: TT_CREDITS

Used in the Customize=CUST_SET_TEXT_COLOR command.

Credit texts shown at the end of the adventure.

5 \$0005: TT_EXAMINE1_BOTTOM

Used in the Customize=CUST_SET_TEXT_COLOR command.

The Bottom text for the EXAMINE2 item. **Rules of Senet second part.**

4 \$0004: TT_EXAMINE1_TOP

Used in the Customize=CUST_SET_TEXT_COLOR command.

The top text for the EXAMINE2 item. **Rules of Senet first part.**

TT_constants

6 \$0006: TT_EXAMINE3

Used in the Customize=CUST_SET_TEXT_COLOR command.

Text for the EXAMINE3 item.

0 \$0000: TT_ITEM_NAME

Used in the Customize=CUST_SET_TEXT_COLOR command.

The name of an item selected in the Inventory.

12 \$000C: TT_LEGEND

Used in the Customize=CUST_SET_TEXT_COLOR command.

Text for the Legend at the start of a level.

9 \$0009: TT_LEVEL_NAME_OFF

Used in the Customize=CUST_SET_TEXT_COLOR command.

Turn off the name of a level in the New Game screen.

10 \$000A: TT_LEVEL_NAME_ON

Used in the Customize=CUST_SET_TEXT_COLOR command.

Turn on the name of the level in the New Game screen.

11 \$000B: TT_MAIN_MENU_OFF

Used in the Customize=CUST_SET_TEXT_COLOR command.

Turn off the Menu text for the main menu, New Game/ Load Game/ Options / Exit.

26 \$001A: TT_MAIN_MENU_ON

Used in the Customize=CUST_SET_TEXT_COLOR command.

Turn on the Menu text for the main menu, New Game/ Load Game/ Options / Exit.

TT_constants

8 \$0008: TT_NEW_LEVEL_ARROWS

Used in the Customize=CUST_SET_TEXT_COLOR command.

Scroll up/down arrows for the new game screen.

20 \$0014: TT_OPTION_DESCRIPTIONS

Used in the Customize=CUST_SET_TEXT_COLOR command.

Text used to describe the option, like the "Control Method".

21 \$0015: TT_OPTION_VALUES

Used in the Customize=CUST_SET_TEXT_COLOR command.

Text for the possible value for options, like "Joystick/Keyboard".

25 \$0019: TT_PAUSED_MENU_ITEMS

Used in the Customize=CUST_SET_TEXT_COLOR command.

Text for the paused menu, "Statistics", "Options", "Exit to Title".

24 \$0018: TT_PAUSED_MENU_TITLE

Used in the Customize=CUST_SET_TEXT_COLOR command.

Title for the "Paused" menu. This text is the "Paused" word.

16 \$0010: TT_SAVEGAME_DESCRIPTION_OFF

Used in the Customize=CUST_SET_TEXT_COLOR command.

Turn off the text for the descriptive line for the save game i.e. "012 Coastal Ruins 3 day 22:13".

17 \$0011: TT_SAVEGAME_DESCRIPTION_ON

Used in the Customize=CUST_SET_TEXT_COLOR command.

Turn on the current selected save game description.

TT_constants

15 \$000F: TT_SAVEGAME_PANEL_TITLE

Used in the Customize=CUST_SET_TEXT_COLOR command.

Title for the save game panel. "Load Game" or "Save Game".

14 \$000E: TT_SCREEN_TIMER

Used in the Customize=CUST_SET_TEXT_COLOR command.

Text for the screen timer like the Angkor Wat level.

7 \$0007: TT_SELECT_LEVEL

Used in the Customize=CUST_SET_TEXT_COLOR command.

Title "Select Level" for the new game screen.

22 \$0016: TT_STATISTICS_DESCRIPTIONS

Used in the Customize=CUST_SET_TEXT_COLOR command.

Text used to describe the statistics, like "Distance Travelled".

23 \$0017: TT_STATISTICS_VALUES

Used in the Customize=CUST_SET_TEXT_COLOR command.

Text for statistic values, like "43m".

WEAP_constants

1 \$0001: WEAP_DISABLE_AUTO_SHOT

Used in the Customize= CUST_WEAPON command.

Disables the repeat function for shooting.

By default when the player presses the Action key the weapon will continue to shoot.

Using this flag the weapon will shoot once and the player will have to release the action key to shoot again.

2 \$0002: WEAP_DISABLE_CHANGE_WEAPON

Used in the Customize= CUST_WEAPON command.

By default the **TRNG engine** will change the current weapon with pistols when the current weapon runs out of ammo.

To disable the changing of the weapon when the ammo runs out add this flag.

WFF_constants

1 \$0001: WFF_BOLD

Used in the WindowsFont= command.

A Bold character has a "heavy" layout.
The depth drawn is larger than the normal or light character.

If the background where the text is shown is not a solid tint, it is better to use a Bold or ultra_bold character otherwise in some combinations of text colour - background colour the border is not easily visible.

512 \$0200: WFF_CENTER_ALIGN

Used in the WindowsFont= command.

This will centre the text at half the text frame.
This align mode is suggested for titles.

1024 \$0400: WFF_FORCE_FIXED_PITCH

Used in the WindowsFont= command.

This flag forces the **TRNG engine** to choose a font with fixed pitch characters.

Theoretically it should be sufficient to choose a font name of fixed pitch like **Courier** but the windows font procedure prefers to choose a different face name font for parameters like size of font.

To avoid this doubt add the **WFF_FORCE_FIXED_PITCH** flag.
The fixed pitch (all characters will have the same width) is very useful when text tables are created like the save game panel.

8192 \$2000: WFF_FROM_RIGHT_TO_LEFT

Used in the WindowsFont= command

The text will be displayed from right to left.

Example: The string: "ABCDEF" will be shown in this way: "FEDCBA"

Note: **This flag Only works with Hebrew and Arabic computers with the the correct character set in the WindowsFont.**

WFF_constants

16 \$0010: WFF_ITALIC

Used in the **WindowsFont=** command.

Sets the italic font.

128 \$0080: WFF_LEFT_ALIGN

Used in the **WindowsFont=** command.

The text will be aligned on the left margin of the text frame.

This is the default value.

Used if IGNORE is typed in the **WindowFontFlags** field.

4 \$0004: WFF_LIGHT

Used in the **WindowsFont=** command.

This setting generates slim characters.

Only use this setting when the background is a solid tint,

otherwise the border of the characters will be difficult to see on a multi coloured background.

256 \$0100: WFF_RIGHT_ALIGN

Used in the **WindowsFont=** command.

The text will be aligned on the right margin of the text frame.

2048 \$0800: WFF_ROTATE_90

Used in the **WindowsFont=** command.

The text will be turned by 90 degrees in a clockwise mode.

Note: There are some limitations for this flag.

4096 \$1000: WFF_ROTATE_INV_90

Used in the **WindowsFont=** command.

The text will be turned by 90 degrees in a inverted clockwise mode.

Using this flag there are some limitation and technical requirements.

See the description of **WFF_ROTATE_90** flag for more information

WFF_constants

8 \$0008: WFF_SHADOW

Used in the **WindowsFont=** command.

Add this flag in the **WindowsFontFlags** field and each character will have a shadow.

Technically this effect requires the same text to be printed twice, moving the second by a few pixels and using two different colours.

By default the shadow is black but it can be changed by using the **ShadowColorRgbId** field.

The shadow has two targets:

- Create a similar-border to get a more visible text on multi coloured backgrounds.
- and Give a 3d effect.

This effect works with bold or ultra-bold characters but less well for small/light characters.

2 \$0002: WFF_ULTRA_BOLD

Used in the **WindowsFont=** command.

This is the maximum heavy for characters:

The depth drawn will be the maximum available.

To have "fat" characters this is the ideal setting but it works when the size set for this font is big. Using a small font an ultra bold could cause a very leaky text.

32 \$0020: WFF_UNDERLINE

Used in the **WindowsFont=** command.

All of the text will be underlined.

This setting is useful for text used as a title etc.

WFF_constants

16384 \$4000: WFF_UNICODE

Used in the `WindowsFont=` command.

To show a text with an eastern character set.

Unicode is UTF16, a fixed 2 bytes (16 bits), encoding method.

To use uni-code type the string using the "binary hex string" format.

This is a necessary "trick" to get around the problem that **NG_Center** program does not support uni-code texts.

To add a uni-code string in the string panel of the NG_Center:

Open block notes or other (simple) text editor of your computer

Type the original text in your language, included in two "\$\$\$\$" markers.

Example:

If the text is "MyText" type into the block notes: \$\$\$\$MyText\$\$\$

Save this text in a file on disk.

Go to the **String** panel of the **NG_Center** and enable [**ExtraNG**] section.

Click to [Add new string].

Now click on [Import Binary String] button.

Choose the file where the original text is saved.

Now choose to keep the text in an external file ("@FileName.txt") or keep the binary string directly in hex format.

WFF_constants

32768 \$8000: WFF_UTF8

Used in the **WindowsFont=** command.

Add this flag to inform the **TRNG** to print text with eastern fonts encoded with multi-bytes format.

When the WFF_UTF8 flag is set the **TRNG** engine will manage the text to print as if it was an UTF8 text.

This means that the **TRNG** will convert it into uni-code before printing it.

When the **WFF_UTF8** flag is added do not add the **WFF_UNICODE** flag as this operation will be performed by the **TRNG**.

For the choice between UTF8 and UNICODE:

See if the **NG_Center** with the correct character set is able to support the language.

To set a character set for the **NG_Center** different from the default ANSI/western character set use the translation file. If you do not have a translation file for the **NG_Center** create a fake translation file:

Open block notes, and type this text with your LanguageName and CharSetNumber:

```
[START_CONFIGURATION]
Version=LanguageName
Charset=CharSetNumber
PropFont=Arial
FixedFont=Courier
[END]
```

WFF_constants

Note: In the above text, replace the "[" character with the "less than" character, and the "]" character, with the "greater than" character.

Now save this text in the **NG_Center** folder, giving the name "**my_scripter_constants.txt**"

Now, close the **NG_Center**, and relaunch it.

Now in the **Settings panel** set the language (typed in Version= command).
If it is not selected, select it now

Now the **NG_Center** will support the character set typed (CharSetNumber) and the font.

Try to type text in the [Strings] section, in the language.
After saving, quit the program and launch new.

In the case where the language is not yet supported by the **NG_Center** use binary strings.
A binary string is a sequence of hex values with the codes used to print a text.

Read the description of the **WFF_UNICODE** flag to understand how to get a binary string from the original text.

The choice between UTF8 and UNICODE (it should be UTF16) depends on the language.

Languages like Chinese and Japanese work with UNICODE,
while other middle-east languages, like Russian, Turkish or Greek, could work with UTF8.

There are also simplified versions of Chinese and Japanese that may be supported by UTF.8.

EXPERIMENT TO DISCOVER THE BEST SETTINGS FOR YOUR LANGUAGE.

WTF_constants

4 \$0004: WTF_CHANGE_COLOR

Used with the Parameters=PARAM_WTEXT script command.

The colour of text will change between the foreground colour and the colour of the shadow mask slowly, giving the look of a blinking text.

1 \$0001: WTF_FLYING_TEXT

Used with the Parameters=PARAM_WTEXT script command.

This flag generates a flying/zoom effect, where the text begins to change slowly and the effect is magnified quickly until it reaches the bigger size set.

The final position of the text will be given by the rectangle (left, top, right, bottom fields) set in the **PARAM_WTEXT** parameters script command.

To have text at the centre of the screen that will fit the whole screen at the end of the flying set 0,0,1000,1000 as values for the rectangle.

Notes: The time (tick frames, 30th of seconds) set in the **TimeDurate** field of the **PARAM_WTEXT** command will be used to set the time that the text will remain on the screen after it reaches the final size.

Type IGNORE or 0 in the **TimeDurate** and the text will remain forever.

The flip effect F364 trigger is used to remove the text.

The smallest time to set in the **TimeDurate** field for this effect is 2 (1/15th of second).

If 1 is set the effect will fail.

It is advisable to always use the **WFF_CENTER_ALIGN** flag in the WindowsFont command for this effect.

This effect works better with single row texts, i.e. text where there is no new line character. If new line characters are used in the text this effect will work in a different way:

The text will not be vertically centred at the start of its flying but will appear in the top part of the screen as if it was coming from the sky.

The same **PARAM_WTEXT** command cannot be used for two (or more) flying effects (also if they are using different texts) at the same time.

Note: The Size typed in the WindowsFont command linked with the current **PARAM_WTEXT** command will be used as the ending (and larger) size. The start size is set by default to be so small as to look like a distant row.

WTF_constants

128 \$0080: WTF_OVER_BINOCULAR

Used with the Parameters=PARAM_WTEXT script command.

Keep the text over the binoculars view.

64 \$0040: WTF_OVER_FIXCAMERA

Used with the Parameters=PARAM_WTEXT script command.

Keep the text when there is a fixed camera.

32 \$0020: WTF_OVER_FLYCAMERA

Used with the Parameters=PARAM_WTEXT script command.

Keep the text when there is a flyby camera.

16 \$0010: WTF_OVER_IMAGE

Used with the Parameters=PARAM_WTEXT script command.

This flag allows windows text to stand over the full screen image that is currently drawn on the screen.

By omitting this flag the text will be disabled when a full screen image has been shown on the screen.

8 \$0008: WTF_OVER_INVENTORY

Used with the Parameters=PARAM_WTEXT script command.

This text will be shown over the inventory and pause screen.

If this flag is omitted, when the inventory pops up the text will be removed and then it will be shown again when the inventory screen quits.

256 \$0100: WTF_OVER_LASER_SIGHT

Used with the Parameters=PARAM_WTEXT script command.

Keep the text over the laser sight view.

WTF_constants

2 \$0002: WTF_PULSING_TEXT

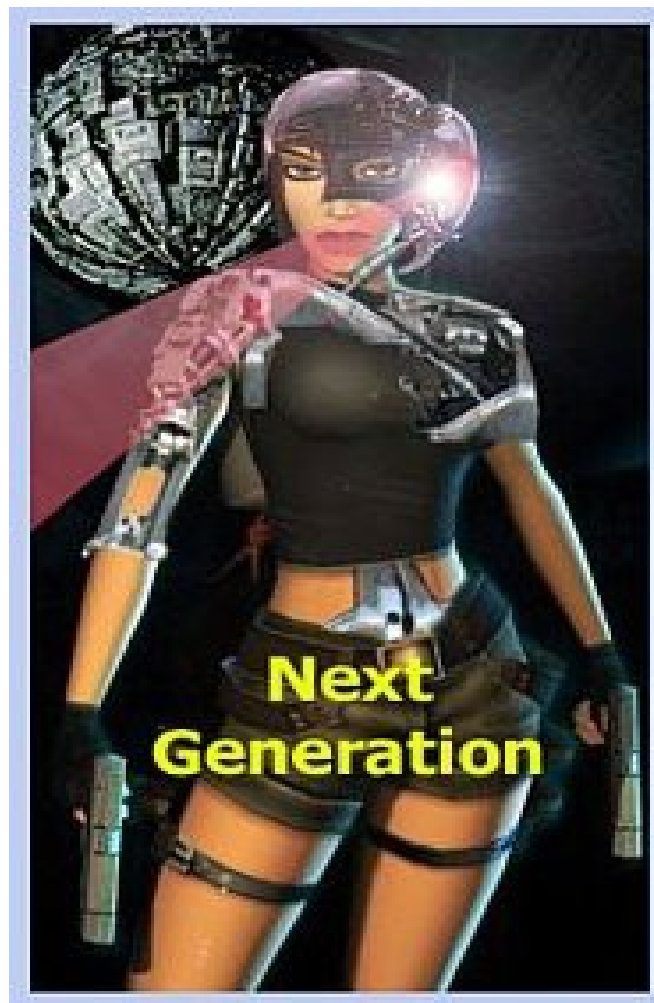
Used with the Parameters=PARAM_WTEXT script command.

This flag generates text where the size will change from a maximum to a minimum size, giving the idea of a pulsing heart.

Note: The Size typed in the WindowsFont command linked with the current **PARAM_WTEXT** command will be used as the smallest size.
The larger size will be the current font size + 30 %.

It is advisable to always use the **WFF_CENTER_ALIGN** flag in the WindowsFont command for this effect.

TOMB RAIDER NEXT GENERATION



MNEMONICS NAMES APPENDIX

ADD	Add an effect Blood, Flame, Light, Mist, Smoke Flags
AMMO	Ammunition Add Remove Flags
ASF	Animation Slot Flags
BAR	Health Damage Bars Flags
BINF	Binoculars Flags
BINT	Binoculars Target View Flags
BKGDF	Background Game Display Flags
BUGF	Bug Flags
CDM	Compact Disc Media Flags
CL	Colour for text Flags
CODE	Codes for Plugins
COLL	Collision Settings Flags
COLTYPE	Colour Types Flags
CUST	Customize Object Flags
DEMF	Demo Flags
DEMO	Demo Frame Flags
DENV	Demo Envelope Flags
DGX	Diagnostics Flags
DIR	Direction Flag for Lara
DISABLED	Disabled
DMG	Damage to Lara Flags
DRT	Dart Flags
DTF	Detector Flags
DWF	Default Windows Font Flags
EDGX	Envelope Diagnostic Flags
EF	Elevator Flags
ENABLED	Enabled
ENV	Envelope Condition Flags
EXTRA	Extra Flags for an Enemy
FADD	Flags for the Add Effect
FALSE	False
FAN	Flags Animation
FBAR	Flags for the Custom Bar on the Screen
FCAM	Flags for the Custom Camera
FFL	Flags for the Custom Flare
FGT	Flags for the Global Trigger
FLI	Flags Log Item
FMIR	Flags for the Mirror Effect
FMOV	Flags for the Moveable
FMV	Flags for the Multimedia Video
FO	Flags for the Organizer
FR	Flags for the Rain Effect
FRB	Flags for the Rolling Boat
FROT	Flags for the Rotating Item

FSB	Flags for the Stand By Condition
FSCA	Flags for Scale Animation
FSCAM	Flags Set Camera
FSS	Flags Show Sprite
FT	Flags for Text
FTYPE	Flags for Import File Type
GT	Global Trigger Flags
GTD	Global Trigger Distance Flag
HAIR	Lara's Hair Flags
HIT	Custom Bike Hitting Enemy Flags
HOLD	Lara Holding items Flags
HRP	Harpoon Flags
IF	If a condition Flags
IGNORE	Ignore
IMPORT	Import File into Memory Flags
JOINT	Object Mesh Joints Flags
KEY	Key pressed Flags
KLH	Keep Lara's Health Flag
LDF	Log Diary Flags
LGTN	Lightening Flags
LOAD	Load Item Flags
MIR	Mirror Flags
MIST_COL	Mist Colour Flags
MPS	Moveable Pause Flag (Actor Speech)
NEF	Now (current) Enemy Flags
NSE	New Sound Engine Flags
OBJ	Object Flags (Motor boat, Rubber boat)
OTYPE	Object Type Flags
PARAM	Parameter Flags
PB	Parallel Bar Flags
PL	Place Log Flags (Diary)
PLACE	Location of Lara Flags (ground, water etc.)
QSF	Query Size Flags (Saved Image)
RAIN	Rain Flags
RIB	Read Input Box Flags
ROOM	Room type Flags
ROTH	Rotate Object Horizontal Flags
ROTV	Rotate Object Vertical Flags
SC	Set Character Text Flags
SEQ	Sequence Flags
SET	Settings Flags
SEXT	Set External Flags (Sound Engine) (obsolete)
SHOWC	Show Custom Ammo Count Flags
SNOW	Snow Flags

SPC	Speech Flags
SPCF	Speech Flags
SPF	Save Game Panel Flags
SPL	Save Game Panel Location Flags
SQ	Sound Quality Flags
STATE	Lara's State Flags
SWR	Star Wars Robot Flags
SWT	Switch Flags
TCF	Trigger Creature Flags
TCMD	Trigger Group Command Flags
TGROUP	Trigger Group Flags
TPOS	Test Position Flags
TRB	Turbo Flags
TRUE	True
TS	Trigger Sound Flags
TSB	Trigger Stand By Flags
TSCR	Title Screen Flags
TT	Trigger Text Flags
WEAP	Weapon Flags
WFF	Windows Font Flags
WTF	Windows Text Flags

