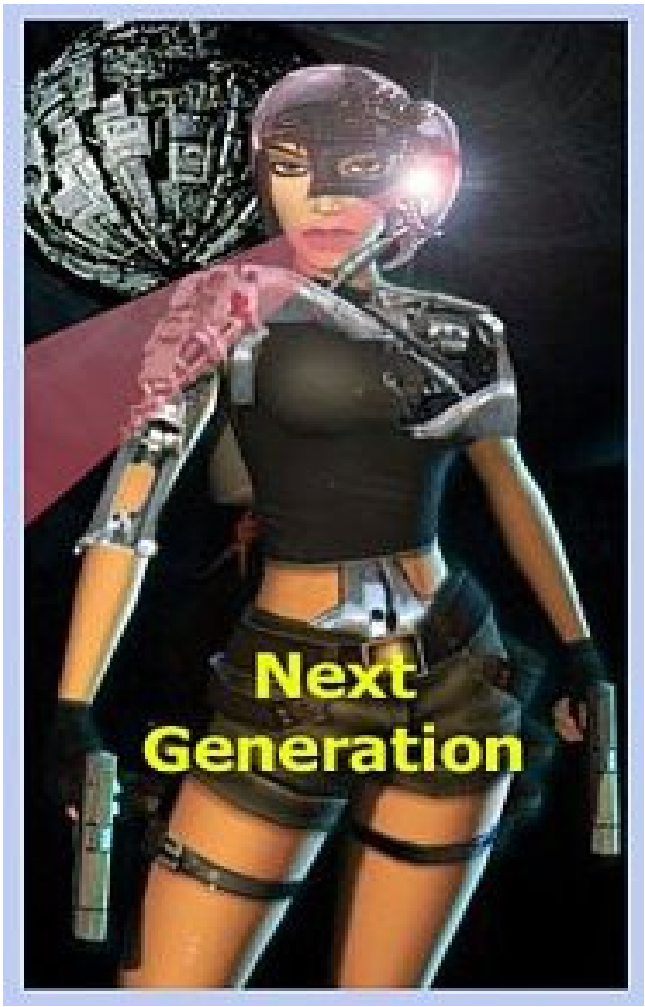
TOMB RAIDER
NEXT GENERATION



NEW SCRIPT COMMANDS

using : NG DLL 1.3.0.7

NEW SCRIPT COMMANDS

	PAGE
#DEFINE	4
#FIRST_ID	7
#INCLUDE	8
ADD_EFFECT	10
ANIMATION	16
ANIMATION_SLOT	21
ASSIGN_SLOT	23
COLOUR_RGB	25
COMBINE_ITEMS	26
CRS	28
CUSTOMIZE	29
CUT_SCENE	30
DAMAGE	31
DEFAULT_WINDOWS_FONT	34
DEMO	36
DETECTOR	39
DIAGNOSTIC	44
DIAGNOSTIC_TYPE	45
DIARY	46
DEMO	54
ELEVATOR	57
ENEMY	61
EQUIPMENT	65
FMV	66
FOG_RANGE	71
FORCE_BUMP_MAPPING	72
FORCE_VOLUMETRIC_FX	73
GLOBAL_TRIGGER	74
IMAGE	77
IMPORT_FILE	79
ITEM_GROUP	82
KEY_PAD	83

NEW SCRIPT COMMANDS

	PAGE
LARA_START_POS	85
LEVEL_FAR_VIEW	86
LOG_ITEM	87
MIRROR_EFFECT	88
MULTI_ENVELOPE_CONDITION	90
NEW_SOUND_ENGINE	92
ORGANIZER	93
PARAMETERS	95
PLUGIN	96
PRESERVE_INVENTORY	99
RAIN	100
SAVE_GAME_PANEL	101
SETTINGS	106
SHOW_LARA_IN_TITLE	107
SNOW	108
SOUND_SETTINGS	109
STANDBY	110
STATIC_MIP	113
SWITCH	115
TEST_POSITION	117
TEXT_FORMAT	124
TEXTURE_SEQUENCE	127
TRIGGER_GROUP	129
TURBO	134
WINDOW_FONT	136
WINDOW_TITLE	138
WORLD_FAR_VIEW	139

NEW SCRIPT COMMANDS

#DEFINE #Define

Syntax: #Define CONSTANT VALUE

Scope: Any

The define directive creates a temporary mnemonic constant with the wished value.

The main use of the **#define** directive is to easily get the remapping of a group of script commands. Some commands require to have an uni-vocal Id, when you insert in the block of commands in the script there is the risk of conflicts with the Ids.

To solve these conflicts build the script using a mnemonic constant defined by the **#define** directive to use as a base for the Id's of the commands.

For example:

```
TriggerGroup=      4, $8000, 112, $2C
TriggerGroup=      5, $2000, 271, $52
TriggerGroup=      6, $2000, 235, $70,  $2000, 232, $52
GlobalTrigger=     2, IGNORE, GT_CONDITION_GROUP, IGNORE, 4, 5, IGNORE
GlobalTrigger=     3, IGNORE, GT_TRNG_L_TIMER_EQUALS, 45, IGNORE, 6, INGORE
```

The above code shows the links between different commands using their Id's

The Global Trigger 2, "calls" the Trigger Groups 4 and 5.
The Global Trigger 3 calls the Trigger Group 6.

When there are more commands it becomes slow to find all of the Id's to change to move the Id range with higher numbers to avoid conflicts preserving the links.

Using the **#define** directive the above code can be changed in this way:

```
#DEFINE BS_TG 4  ;the first TriggerGroup index
#DEFINE BS_GT 2  ;the first GlobalTrigger index

TriggerGroup=      BS_TG, $8000, 112, $2C
TriggerGroup=      BS_TG+1, $2000, 271, $52
TriggerGroup=      BS_TG+2, $2000, 235, $70,  $2000, 232, $52
GlobalTrigger=     BS_GT, IGNORE, GT_CONDITION_GROUP, IGNORE, BS_TG,
                   BS_TG+1, IGNORE
GlobalTrigger=     BS_GT+1, IGNORE, GT_TRNG_L_TIMER_EQUALS, 45, IGNORE,
                   BS_TG+2, IGNORE
```

In this case the final compiled code will be exactly the same, but the advantage of a second version is that when a conflict is found and it is necessary to change all of the Trigger Group and Global Trigger indices, preserving the internal links between them,only change the two directives:

#DEFINE #Define

#DEFINE BS_TG 10
#DEFINE BS_GT 8

All id's will be moved higher keeping the correct inner links.

The problem of Id's remapping are usual when the level builders exchange blocks of commands to explain how to create some features. So, the code sample will be easily adapted to any script if the **#define directive** is set to all Id numbers.

Remark: See the #FIRST_ID directive to complete the above target.

Other uses of the **#define directive** are less useful, anyway some idea could be:
to use a define to use a single word for a group of official mnemonic constants often used:

#define DEBUG
DGX_LOG_SCRIPT_COMMANDS+DGX_COMMON_VARIABLES+DGX_LARA

and

#DEFINE USUAL DGX_LARA+DGX_SFX_SOUNDS +DGX_CHEATS

Easily change the Diagnostic command :

Diagnostic= DEBUG, 0
or
Diagnostic= USUAL, 0

Some notes about the #define directive:

The **#define directive** is local and it only works inside the current source.
If include files are used and therefore different sources any directive will only work in the source where it is.

The advantage to divide the script in different sources is that the same directive in any source can be used with different values.

The same CONSTANT name can not be used twice in the same source.

#DEFINE #Define

Example: #DEFINE MYNAME 12
#DEFINE MYNAME FT_BOTTOM_CENTER+FT_SIZE_ATOMIC_CHAR

You can use in the formula (to get the VALUE) the CONSTANT name set in another define directive of the same source but these directives have to be placed before the define that will use them:

This code is correct:
```
#DEFINE BS_ALL  10
#DEFINE BS_TG BS_ALL+30
#DEFINE BS_GT BS_TG+15
```

while this is wrong:
```
#DEFINE BS_ALL  10
#DEFINE BS_GT BS_TG+15
#DEFINE BS_TG BS_ALL+30
```

Because the value of BS_TG is not defined when the #DEFINE BS_GT directive is parsed.
There can not be blanks (spaces) between operands of the VALUE field.

This is wrong:
```
#DEFINE MYSUM ALFA +BETA
```

This is correct:
```
#DEFINE MYSUM ALFA+BETA
```

NEW SCRIPT COMMANDS

#FIRST_ID #FIRST_ID

Syntax: #FIRST_ID CommandName=FirstId

Scope: **Any**

This directive has been studied to be used in the include files of the script.
The only disadvantage using include files is the problem of Id's conflicts.
Many script commands have an Id as a first argument and these Id's have to be uni-voque
for the same type of command.

The **NG_Center compiler** is able to find these conflicts and signal them as errors after the compiling.
It should be OK to use in an include files Id's in a range of values that is different from that used in the main script or in other include files.

The **#FIRST_ID** directive tries to reach this target.
When an include file type is created,
in the first row there are one or more directives like the following:

#FIRST_ID TriggerGroup=40
#FIRST_ID GlobalTrigger=30

When typing a new script command and you hit F1 to find the first available Id for that command, the **Tomb Scripter** will compute the first available Id starting the count from the value typed in the specific FIRST_ID directive.

So, if the first Trigger Group command is typed in this include file
(but many others could exist in the main script or in other include files)
the **Tomb Scripter** will give the value "40" as the first id for the Trigger Group.

When typing a second trigger group the first id will be "41".
Decide from what Id to start all script commands in the include files,
reducing the risk of conflicts between different include files and/or the main script file.

The FIRST_ID directive is placed at the top of the include files also has an informative value when in the future this feature is used to find the Id ranges used in the old include file.

Define with the Mnemonic constant: **#DEFINE** BS_TG 30
#FIRST_ID TriggerGroup=BS_TG+10

Do not put spaces between the operands to the right of the "=" sign:

This is Wrong: **#FIRST_ID** TriggerGroup=BS_TG+ 10

The **#FIRST_ID** directive is local. This means it only works in the same source file.
To enhance the Id range swapping see the description of the **#define directive**.

NEW SCRIPT COMMANDS

#INCLUDE #Include

Syntax: #Include "ExternalScriptTextFile.txt"

Scope: **Any**

The **#include directive** is not a script command as it does not perform any change to the final **script.dat**.

It is only used to divide and to put in order a long **script.txt** moving some specific blocks of commands from the main script to another text file in the Script folder.

When the **Tomb Scripter** finds a **#Include** directive it temporary loads the content of the given include file and pastes it with the main script in the same position where that include directive was.

The response in the compiling is always the same.

The use of include directives is useful to put in order the script dividing it into logical blocks.

For example: When many script commands are created to realize a custom animation or other effect, copy all commands for that target in a separate text file with a meaning full name.

The script could become like this:

#INCLUDE "LaraHome.txt"
#INCLUDE "AnimClimbFromWater.txt"
#INCLUDE "DistanceSensory.txt"

There are advantages in this job planning.
When you are not interested in some completed feature, you do not have all those script commands in the main script file to create confusion.

When you work on that feature you can load (F5 key) the include file in the **Tomb Scripter** and have the meaningful commands in the editor to study and edit.

If you work with the include files read the description of the directive **#FIRST_ID**.
It is very useful to reduce the risk of Id's conflicts of the commands stored in the main script and include files

#INCLUDE **#Include**

Remarks: Theoretically you could use an **#include** directive in another included source up to 19 levels of depth.

I do not suggest to use this because the **F1** command to find the free **id** could not work and it becomes difficult to navigate between the sources since the **F6** command only allows you to come back to the main **script.txt**.

Remember that you can load an include file in the editor hitting the **F5** key while the caret is over the **#include** line.

NEW SCRIPT COMMANDS

ADD_EFFECT AddEffect=

Syntax: AddEffect=Id, EffectType (**ADD_**), FlagsEffect (**FADD_**), JointType (**JOINT_**), DispX, DispY, DispZ, DurateEmit, DuratePause, Extra param array

Scope: To use in the [Level] section

The Add Effect command is suggested only for advanced level builders.
Its more common use is to have animation commands to use in the animation editor of the **Wad Tool** program to add particles effects to a moveable only in specific frames of animation.

You can add: Blood, Flame, Smoke and Mist (like waterfall Mist).

Use an Add effect like common triggers.
A better use is to convert the trigger in the animation command and then add it in the animation of the moveable.

Arguments:

Id Field
This is a simple identifier for the AddEffect command in the script file.
It works like the first argument of the PuzzleItem or PuzzleCombo commands.

For example type "1" refers to the AddEffect in the trigger window choosing one effect to add.

EffectType
Set in this field the **ADD_** constant to choose the effect type to add.

FlagsEffect (FADD_)
Use flag effects (**FADD_**) to override other settings about the duration or behaviour of the effect.

See the description of the **FADD_** constants in the **MNEMONICS**.

JointType (JOINT_)
Specify the type of effect to set the exact position where to show the effect.
This computation is complicated and starts from a point corresponding to some joint of the moveable.

See the following fields DispX, DispY and DispZ to set the displacement from this chosen joint.

ADD_EFFECT AddEffect=

One of the the following joints can be chosen:

JOINT_SINGLE_MESH:0
JOINT_PUBIS:0
JOINT_LEFT_THIGH:1
JOINT_LEFT_KNEE:2
JOINT_LEFT_ANCKLE:3
JOINT_RIGHT_THIGH:4
JOINT_RIGHT_KNEE:5
JOINT_RIGHT_ANCKLE:6
JOINT_ABDOMEN:7
JOINT_NECK:8
JOINT_LEFT_SHOULDER:9
JOINT_LEFT_ELBOW:10
JOINT_LEFT_WRIST:11
JOINT_RIGHT_SHOULDER:12
JOINT_RIGHT_ELBOW:13
JOINT_RIGHT_WRIST:14

The value JOINT_SINGLE_MESH (0), will be used for the moveable using a single mesh.
In this situation the origin of the effect will be the center of the mesh.
The pivot is often the center of single meshes.

When a particle effect is added to the multi-mesh moveables set a joint as an origin and then modify this origin with DispX, DispY and DispZ fields to create the effect at the required position.

See the following field descriptions for more information.

DispX, DispY, DispZ
The three values DispX, DispY and DispZ are the distance between the origin of the effect set in the Joint field. To understand how to set these three values read the example.

To add smoke to Lara's mouth (as breath) choose a joint near to the target point.
Choose the neck, that is the **JOINT_NECK** is typed in the joint field.

Then move this origin.
The smoke is not in the neck but in a higher and more forward position.
To understand the axes X, Y, Z imagine watching Lara when she has her face in front of you.

Now the X, Y and Z axes have the following orientation:

X values will be negative going to the right hand of Lara and
X values will be positive going to the left hand of Lara.

Y values will be negative when moving to Lara's head (upwards) and
Y values will be positive when moving to Lara's feet (downwards)

Z values will be positive in front of Lara and
Z values will become negative behind Lara.

From the above orientation use the values to move the smoke in front of Lara's mouth.

From Lara's Neck as the origin: Move the smoke forwards,
because the mouth is in front of her Neck.

So from the Z co-ordinate use a positive value because the smoke will be in front of Lara.

The DispZ = +70

Move the smoke up because the Neck is below the mouth.
The Y axis has negative values for upward and positive values for downward.

Use the value: DispY = -70

DispX can be set at 0 because the Neck and the mouth are on the same vertical line.

This gives: JOINT_NECK, 0, -70, 70

The smoke should now be in front of Lara's mouth.

DurateEmit and DuratePause
For some effects, Smoke and Blood, it is necessary to alternate a phase of emitting particles with a pause phase to permit the particles to decay.

The values to type in the above fields is the number of frames.

For example: The DurateEmit has a value of 3 and DuratePause has a value of 10.
The effect will be emitted for 3 consecutive frames and will be suspended for 10 frames.

Chaining these two values sets the required intensity of the effect.

ADD_EFFECT AddEffect=

To set these two fields:

Fire and Smoke: Use the flag **FADD_CONTINUE_EMIT** to have continuous emission.
In this case the values in DurateEmit and DuratePause fields will be ignored.

For Blood: Set 1 for the DurateEmit and 30 for the DuratePause.

For Mist: Depends on the result wanted.
In some circumstances Mist works with the **FADD_CONTINUE_EMIT** flag. To simulate sprays use DurateEmit = 1 and DuratePause = 3.

Extra Param array
There are optional fields changing according to the effect type.
Currently there are only extra parameters for Mist and Light effects.

Extra parameters for Mist:

Extra1 = The Size of the Mist Ball. The default value is 12.

Extra2 = Number of Mist Balls.
The default value is 1.
A maximum of 4 Mist Balls can be set.
They will be placed over an ideal line.
The Mist line will be oriented following the facing of the current moveable.
The facing of the Mist line can be rotated using rotate flags.

Extra3 = Colour of the Mist.
Set a **MIST_COL_** value.

Extra4 = Persistence time of the Mist.
The default value = 6.
Use this field only if the moveable is moving faster and a wake is required.
Bigger values, longer wakes.

ADD_EFFECT AddEffect=

Extra Parameters for light effects:

Extra1 = Intensity of the light.
Bigger values will have more intensity and width of light.
The light of flares has intensity = 16.
Use values close to this reference.
This value works for flat light and blinking light
but has no effect with the spot-light.

Extra2 = The maximum distance for the spot-light.
This value only works for the **ADD_LIGHT_SPOT** effect.

The spot-light uses two 3d points:
The begin point and
The end point of the light cone.

The first 3d point is the moveable where the spot is added,
the final point is the last zone light by the spot-light.
Computing Source + Distance in given direction (facing).
The distance should be typed in sectors.

For example the headlight of the SIDECAR has a default of 20 sectors
in the old tomb4 and it was the **WorldFarView.**
If a super power light is not required reduce this distance.
If **IGNORE** is typed in this field the **TRNG** engine will use 12 sectors.

Extra3 = Colour of the light.
Use the same **MIST_COL** constants for the Mist effect.

ADD_EFFECT AddEffect=

Extra Parameters for ADD_FLAME effect:

Extra1 = Intensity.
The value for this field changes in accordance with the presence and value of the **FADD_FIRE_STRIP** flag.
When the **FIRE_STRIP** flag is used in the Extra1
set a value between +1000 to -8000.
A reasonable value is -4000.

When a common single fire is used set an OCB value of 0, 1, or 2 where increasing the number increases the size of the fire.

Extra2 = Setting to burn Lara.

OCB 1 = Burns Lara, 0 = No damage for Lara.

Extra3 = Sets the direction.
OCB 1= Vertical 0= Horizontal

This field is only used with the **FADD_FIRE_STRIP** flag.

To have the horizontal fire set 0 in this field.
If 1 is set there is a fire eruption like a little volcano.

NEW SCRIPT COMMANDS

ANIMATION **Animation=**

Syntax: Animation=AnimIndex, **KEY1_** , **KEY2_** , **FAN_** flags, **ENV_** Environment, Distance for Env, Extra, StateId (**STATE_**) or (-)AnimationIndex array (...)

Scope: To use in the [Level] section

With the Animation command new animations for Lara can be performed when the player hits a key.

This command is suggested only for advanced level builders to add new moves for Lara. To set the number of the animation and the hot key used to engage to set some condition about when this new Lara command will be permitted.

Consider the situation for performing an animation.
For example a kick while Lara is underwater or hanging on a rope is weird.

Arguments:

AnimIndex
Type the number seen in the Animation Editor of the **Wad Tool** program.

Key1 and Key2
Set a code (or two) in these fields to identify the activation key for the new animation.

Remark: The special value **KEY1_RELEASED** may be added to the **KEY1** field to invert the condition about keys.

When **KEY1_RELEASED** is present the condition is true only when the specified keys are NOT DOWN.

Two kinds of value can be set in the Key fields : **Keyboard scan codes**
Keyboard game commands.

ANIMATION **Animation=**

Both types identify a key of the keyboard but there are some differences:

Scan Codes permit a choice of any key (or almost) from the keyboard.

Game commands are only the prefixed keys seen in the Option screen of the Tomb Raider game.
The Game commands have the advantage to be activated by a joystick
If the player changes the setting in the Option screen of the **TRNG** engine the command
will continue to work.

Another difference is the way to type the values in the **KEY1** and **KEY2** fields.

See the KEYBOARD SCAN CODES.

To use game commands use the constant **KEY1_** and **KEY2_**

When game commands are used remember to place the value in the **KEY1_** field or
in the **KEY2_** field in accordance with the prefix **KEY1_** and **KEY2_**.

If scan codes are used to assign a new key, for example the scan code for the **F8** key, or for "**Q**"
literal. Two different scan codes can be set, writing the first in **KEY1_** and the second in **KEY2_**.

In this case the animation will be performed only when both keys are hit at the same time.

For example: Animation=500, 42, 11, ...
Since 42 = (Shift left) and 11= (Number 0), the animation will be started
when the player hits at the same time SHIFT + 0 (zero)

Remark: To use scan codes add the flag **FAN_KEYS_AS_SCANCODE** in the **FAN_** field.

Note: With the Key Board Game commands to insert two or more codes in the
same Key field type the sum of these values.

For example: For two or more game commands at the same time, add the constant value
KEY1_ with the other **KEY1_** and **KEY2_** with the other **KEY2_**.

For example: To combine the game commands Action Jump and Dash in the **KEY1_** field
KEY1_JUMP+KEY1_ACTION in the Key field gives the value
KEY2_DASH.

Remark: If the **KEY1_** or **KEY2_** fields are unused type **IGNORE** to signal that
it is not used.

ANIMATION **Animation=**

FAN_ Flags
Set one or more flags **FAN_** constants to to set the behaviour of the animation command.

See the **FAN_** constants

ENV_Environment
In the Environment field set a single specific condition about the environment around Lara.
Only use this field when the animation requires a correct environment around Lara.

For example climbable walls, holes, walls etc.
If no Environment is required type **IGNORE** in this field.

See the description of the **ENV_** constants

Remark: Using **FAN_** flags you cannot type the sum of any **ENV_** values.
Choose a single **ENV_** condition and only add to it **ENV_POS_** flags or an **ENV_NON_TRUE** flag to inverse the meaning of the condition.

Distance for ENV
Set **IGNORE** in the field and the **TRNG** engine will use a default value for the named Environment. A different value can be set in the Distance field to specify a size or height or range about the specific **ENV_** condition set in the Environment field.

The common situation is Distance, the height or depth of the wall or hole.
The units to use are 1024 for 1 sector, so 1 click = 256 and a half sector is 512.

If the Distance field is a range, like the Environment **ENV_MONKEY_CEILING**
the units will be different.

See the description of the **ENV_MONKEY_CEILING** value about how to set the Distance field for that case.

Remark: **IGNORE** can always be typed in this field and the **TRNG** engine will use a default value ideal for the current **ENV_** condition chosen.

Extra
This field may have different values according to the **FAN_** flags.
Currently it only accepts slot numbers.

If the flag is set for the **FAN_WHEN_ON_VEHICLE** in the **Extra** field,
type the slot number of the vehicle. (The slot for the animations of that vehicle.)

If the **FAN_USE_EXTRA_SLOT** is set in the **Extra** field,
type the number of the animation slot to find the Animation Index set.

ANIMATION **Animation=**

StateId or AnimationId array
After the **Extra** field type one or more values to set the State-Ids or the Animation indices to use as conditions to start the special animation.

The special animation will be performed only if at least one of the given state Ids and one of the animation indices is currently active for Lara in the game at the time the player hits the correct keystroke.

To recognize the State Id from the Animation indices use positive values for the State-Ids and negative values for the Animation Indices.

For example: the list: 96, 112, -28, -42

This is read as: "Perform my animation only if state Id of Lara = 96 or 112, and at the same time the current animation of Lara is 28 or 42.

State Id values can also set the **STATE_**constant.

If you do not understand what a "state-Id" is or the Animation index it can be interpreted as a number to signal what Lara is doing.

For example: The following table shows the most important state-Id's:

Climbing:	**$38 , $39, $3A, $3B, $3C, $3d**
Falling:	**$09, $1d**
Jumping:	**$19, $1A, $1b, $1C, $03**
Moving on all fours:	**$50, $51, $47, $48**
Rolling:	**5**
Running:	**1**
Walking:	**0**
Monkey:	**$4b, $4c, $4d, $4e, $4f, $52, $53**
Still, stand up:	**2**

The '$' sign means "hexadecimal value". Use the **OCB calculator** of the **NG_Center 1.5.7** in the **TIDE** folder to convert hex to decimal. Or type the number directly in hex with a $ sign in front.

Numbers can also be input in decimal. The State ids typed are used by the **TRNG** engine to understand when the animation is allowed and when it is not.

Normally a state Id is set with the value "2".
The animation will only be enabled if Lara is still on land and standing up.

You can create a animation to perform only when Lara is climbing a wall.

In this case type all values for climbing: $38 , $39, $3A, $3B, $3C, $3D
or only some of these values.

ANIMATION **Animation=**

To find all of the state Id values or animation indices to use in the **TRNG** engine use:

Diagnostic=ENABLED in the **script.txt** file and see the animation and state Id values.

Or use the **Wad Tool** program and go to the **Animation Editor**.
When an animation of Lara is selected see her "State-Id" value and Index of animation.

NEW SCRIPT COMMANDS

ANIMATION_SLOT AnimationSlot=

Syntax: AnimationSlot= Slot, ActionType (**ASF_**), AnimIndex, Key1, Key2, **FAN_** flags, **ENV_** Environment, Distance For Env_, Extra, StateId (**STATE_**)
or (-)AnimationIndex array (...)

Scope: To use in [Level] section
Max number of instances for level section: 511

The AnimationSlot is like the Animation command with the difference that the AnimationSlot works differently on a moveable with Lara.

Use it to create a new moveable or vehicle, or to change the animations of some already existing moveable.

Most of the fields and flags of the AnimationSlot are the same as the Animation command.
To get information about the common fields read the description of the Animation command.

Slot
In this field type the slot number or MNEMONIC Constant **(BADDY_1**, **MUMMY**) of the slot to change or create.

AnimSlotFlags (ASF_)
In this field type one or more **ASF_** values to affect the AnimationSlot command.

Other fields (common to Animation command)

All fields following the (above) ActionType field, are the same as the Animation command.
Read the description of the Animation command for information about these fields.

The difference between the two commands:

The AnimationSlot works on a specific (different Lara) slot.
The animation numbers and state-ids in the final State_id, AnimIndex array
will use the animation and state ids of that slot.

Except for the Vehicles, the **KEY1** and **KEY2** fields are unused in the AnimationSlot as the player does not control the enemies.

ANIMATION_SLOT **AnimationSlot=**

When building a vehicle work on two different slots:

the slot of the vehicle,
and the slot with the Lara's animation used for the vehicle.

Since Lara and the vehicle will always have the same animation number at the same moment (Lara is driving it), it is not a problem to set in the **AnimIndex** field the animation number.

It is not the same to type in the **Slot** field (the first field of the AnimationSlot command) the vehicle slot or Lara's animation slot because the condition for the environment and distance will be different depending on the choice of the moveable.

Set the vehicle slot for the animtype for the moving of the vehicle as it has a bigger collision box and it is this collision box that is the most important.
Use Lara's animation slot to compute other situations like the injury of Lara by enemies.

Remark: When a condition for Lara's animations is set the effective object that will be used will be Lara. The only difference between the animations used is that they will be those of the specific Lara's animation slot.

When the envelope condition: **ENV_CONDITION_TRIGGER_GROUP** flag is used, the Action and Conditions Triggers stored in that Trigger Group can be redirected to work on the enemy, vehicle, or Lara's animation slot using one of following Trigger Group flags:

TGROUP_USE_ITEM_USED_BY_LARA_INDEX
This is the vehicle driven by Lara.

TGROUP_USE_EXECUTOR_ITEM_INDEX
This is the index of Lara while she is driving,
or it is the index of the enemy you are changing or creating with the AnimationSlot command

TGROUP_USE_FOUND_ITEM_INDEX
This is the item that has been found with a TestPosition condition in the AnimationSlot command.

NEW SCRIPT COMMANDS

ASSIGN_SLOT **AssignSlot=**

Syntax: AssignSlot=MyUsedSlot , SlotType

Scope: To use in the [Level] section

Remark: **From version 1.2.2.2** the Assign Slot has new features so read the following updated description.

MyUsedSlot
Number or constant name to identify a slot position of the standard tomb4 wads
(example: "ENEMY_JEEP" or 34)
This slot identifies where the object is placed in the slot list.

SlotType
In this field type from where the features to assign to the object are taken.
In the new mode type as a Slot Type any slot and all of the features of this slot will be forced to the object placed in the **MyUsedSlot** slot.

For example: If a restyled crocodile is in the Animating1 slot and you want to give it the features of a real crocodile type in the [Level] section the following:

AssignSlot= ANIMATING1, CROCODILE

Use the Assign Slot to have different enemies with the same features but different looks, like two or more types of crocodile, or a different BADDY_1 enemy with different layouts, etc.

Remark: Not all slots will accept a reassignment.
The doors, waterfalls, switches and some emitters have trouble in a reassigned slot.
Only use this command to work with enemies.
When a slot is reassigned it is necessary that both of the slots are present in the wad.

For example: In the command: AssignSlot= **ANIMATING1**, **CROCODILE**

It is necessary that there is at least one real crocodile in the **CROCODILE** slot.

Some enemies like **BADDY_1**, **BADDY_2**, **VON_CROY** and the **GUIDE** require an alternative skin slot.

ASSIGN_SLOT **AssignSlot=**

By default the alternative skins are present in the following slots:

Alternate skin of **BADDY_1**	in	**MESHSWAP3**
Alternate skin of **BADDY_2**	in	**MESHSWAP2**
Alternate skin of **VON_CROY**	in	**MESHSWAP1**
Alternate skin of the **GUIDE**	in	**MESHSWAP2**

When re-assigning some of the above enemies supply a new alternative skin but do not place it in the mesh-swap slots. Place the alternative skin in the slot immediately following the one used.

For example: If you use the script command: AssignSlot= **ANIMATING4**, **BADDY_1** the **TRNG** engine looks for the alternative skin for **BADDY_1** in the **ANIMATING4_MIP** slot since this is the slot that follows the **ANIMATING4** slot with the main object (reassigned) to **BADDY_1**.

You can assign to new re-assigned enemies a new Health Points (vitality) different from the other Health Points of the original object.

In the above example: AssignSlot= **ANIMATING4**, **BADDY_1**

You can force the **BADDY_1** in the **ANIMATING4** slot to have Health Points different from the other **BADDY_1** slots, using the Enemy= command:

Enemy= ANIMATING4, 1400, IGNORE, IGNORE, IGNORE, IGNORE, IGNORE

A different damage to Lara for the **ANIMATING4** cannot be set.
A different value in an enemy command for damage will be ignored.
The damage of the reassigned enemy will be the same for the source assigned slot.

In the example it will be the same damage for all other **BADDY_1** objects.
To change the damage affected by a reassigned enemy use the ENEMY script command to modify the damage of the original object from where you got the re-assignment.
The reassigned enemies do not support a MIP version.
The reassigned enemies cannot have animations different from those of the original enemy.
The animations of the reassigned slot will not be used in any other way.

Trick: The only way to overcome the above limitation is to insert in an animation of the original enemy an exported flip-effect used to call a Trigger Group in the script.txt file.
If in this Trigger Group there is a starting condition about the slot Id of the current item (see TRNG Variables demo), you could perform some animation command only when that animation will be performed by the reassigned enemy.

In this animation command place the performing of another different animation.
This new animation should always be added to the original enemy and never to the reassigned enemy slot.

NEW SCRIPT COMMANDS

COLOUR_RGB **ColorRGB=**

Syntax: ColorRGB=IdColor, Red, Green, Blue

Scope: To use in [Level] section

With ColorRGB insert a RGB value to use later with some flip-effect or actions using the IdColor as a reference to locate it.

IdColor
Enter a progressive number to identify this colour from other colours set in the same [Level] section. When a flip-effect or action requiring a script color is used insert this IdColor number in the trigger to locate the color.

Red
Intensity of the Red.
Type a value between 0 and 255

Green
Intensity of the Green.
Type a value between 0 and 255

Blue
Intensity of the Blue.
Type a value between 0 and 255

NEW SCRIPT COMMANDS

COMBINE_ITEMS CombineItems=

Syntax: CombineItems=FirstItem (slot), SecondItem (slot), FinalItem (slot)

Scope: To use in the [Level] section

FirstItem, SecondItem, FinalItem
All fields accept a slot value that can be found in the **SLOT MOVEABLES.**

Remark: **DO NOT** use ANY slot but only the slot corresponding to the inventory items.
You can recognize these slots because they have in their name the word "_ITEM".

For example: Cannot use **PISTOLS_ANIM** or **LARA_HOLSTERS_PISTOLS**
Can use **PISTOLS_ITEM** or **PISTOLS_AMMO_ITEM**.

Description
This command allows the creation of a new combining rule about inventory items to get a new item.

The **TRNG** engine allows the combination of two items such as **PUZZLE_ITEM1_COMBO1** with **PUZZLE_ITEM1_COMBO2** to get the new item **PUZZLE_ITEM1** in the inventory.

Or combine the **LASERSIGHT_ITEM** with the **CROSSBOW_ITEM** to get the Crossbow with Laser-sight.

Using the Combine Items command get other targets such as:

To create any coupling not only with specific "combo" items or with laser_sight + weapon.

For example: Combine **PUZZLE1** with **KEY2** to get any other new item.

Set as a final item not only the Puzzle items but also weapons, medipacks, examine items and all other inventory items.

It is possible to build a weapon allowing the player to pick up different pieces and only when all the pieces are picked up will the working weapon be built.

COMBINE_ITEMS CombineItems=

An example for a final item that has 2 , 3, 4, 5 and more pieces.
To realize this target type two or more Combine Items command in the same [Level] section.

For example: If the final weapon is the Grenade Gun and there are three pieces to build it type the following script commands:

CombineItem= PUZZLE_ITEM1, PUZZLE_ITEM2, PUZZLE_ITEM3
CombineItem = PUZZLE_ITEM3, PUZZLE_ITEM4, GRENADE_GUN_ITEM

In the above commands use the PUZZLE_ITEM3 only as an intermediate item.
The three pieces to build the Grenade Gun will be:

(PUZZLE_ITEM1 + PUZZLE_ITEM2) + PUZZLE_ITEM4 = GRENADE_GUN_ITEM

Remark: It is not logical for a PUZZLE_ITEM3 with PUZZLE_ITEM1+PUZZLE_ITEM2 as there are already two combos to build a Puzzle Item.

The above example is given to show the method.
It would be more logical to have three pieces of the weapon using some default combo items.

For example:

PUZZLE_ITEM1_COMBO1 + PUZZLE_ITEM1_COMBO2 + KEY_ITEM1_COMBO1

Use the combine rule to get the PUZZLE_ITEM1
from (PUZZLE_ITEM1_COMBO1 + PUZZLE_ITEM1_COMBO2)
type this command:

CombineItem= PUZZLE_ITEM1, KEY_ITEM1_COMBO1, GRENADE_GUN_ITEM

Lara will pick-up COMBO1 and COMBO2 for PUZZLE_ITEM1.
Lara will pick-up the third piece, the KEY_ITEM1_COMBO1 and at the end when combined (with the standard combine rule) the COMBO1+COMBO2 for PUZZLE_ITEM1.

Then Lara will be able to combine the new PUZZLE_ITEM1 with the third piece of the weapon stored in the KEY_ITEM1_COMBO1 item.

Remarks: **DO NOT** use **IGNORE** in this command.
If invalid slot values are typed the corresponding Combine Items command will be ignored by the **TRNG** engine.

The maximum number of Combine Items in the same Level section is 67 instances.

NEW SCRIPT COMMANDS

CRS **CRS=**

Syntax: CRS=ENABLED/DISABLED

Scope: To use in the [Options] section

When CRS is enabled it is very improbable the game is stopped by a crash.

The use of CRS is to disable CRS (CRS=DISABLED) while building the level.
It is better to know if something does not work.

Only ENABLE CRS when developing the final release to avoid further crashes.

Remark: When CRS is disabled if a crash occurs a file named **Last_Crash#.txt** will be created in the current trle folder.

Another little file (always the same) will show that the game has been closed and what happened in the crash and where the Last_Crash file with the crash log is located.

NEW SCRIPT COMMANDS

CUSTOMIZE **Customize=**

Syntax: Customize= **CUST_** Customize Type, Arguments

Scope: To use in the [Level] section.

The Customize command permits changes for many settings replacing the old default values and behaviour of the standard tomb4 engine.

Type Customize commands in the same [Level] section.
By default the settings will only work for the current level.
To have a setting working for all levels insert the Customize= command in the Title level.

CUST_ Customize Type
In this field type a **CUST_** constant to specify what type of customizing is required.
See the **CUST_** constants

Arguments
The arguments that are typed after the **CUST_** type are variable numbers (in some circumstances they are absent) and are different according to the current **CUST_** type.

Read the description of each specific **CUST_** value to know its use.
See the **CUST_** constants

NEW SCRIPT COMMANDS

CUT_SCENE **CutScene=**

Syntax: CutScene=ENABLED

Scope: To use in the [Level] section

The Cut Scene command permits a signal to the current level like a Cut Scene level.

The operations performed by this command are only two:

All keyboard and joystick input will be disabled for the whole of the Cut Scene level.

If Lara has weapons or a flare in her hand it forces Lara to get a free hand.
The operation to free Lara's hands will be performed at the start of the Cut Scene level.

NEW SCRIPT COMMANDS

DAMAGE **Damage=**

Syntax: Damage=Flags **DMG_** , SecondsForDeath, SecondsForBarRestore, BarColor, BarName, BlinkPercentage

Scope: To use in the [Level] section

The Damage command allows the customized appearance and behaviour of the Damage feature in Damage Rooms and Cold Water Rooms in the game.

Remark: Do Not use any Damage command in the **script.txt** for Damage Rooms and Cold Water Rooms to work using the default settings. (see below).

Arguments:

FlagsDMG_
One or more constant values to set the behaviour of Damage.
Add different **DMG_** values using plus '+' sign to sum them.

See the **DMG_** flags

SecondsForDeath
The Damage procedure use seconds as units for Damage and speed of the Bar.
The number set in this field is the number of seconds necessary to fully decrease the whole bar.
If a large number is set (for example 40 seconds) the damage is very small.
If a small number of seconds is set the damage will be large and Lara will be killed quickly.

SecondsForBarRestore
This field works like the **SecondsForDeath** but in this case it controls the speed to fully restore the Health Bar when Lara goes from the Cold Water or Damage Room.

DAMAGE **Damage=**

BarColor
Type a RGB value to set the main colour of the Bar.
The Red Green Blue value is in hexadecimal format (use '$' prefix for hexadecimal values) is:
$RRGGBB

Where: RR = RED, GG=GREEN and BB=BLUE intensity.

Some possible values are:

$f924f1	PINK	(default color for the Cold Water Rooms.)
$F6F923	YELLOW	(default color for the Damage Rooms.)
$fb8953	ORANGE	
$FF0000	RED	
$00FF00	GREEN	
$0000FF	BLUE	

BarName
Set in this field a string to describe the Bar in the game.

For example: If the string "Temperature" is typed,
the text will be shown in the game at the bottom of the Damage Bar.

Remark: The text typed in this field must be present in the **english.txt** file in standard strings or in the ExtraNG strings sections.
A way around this limitation is to directly type the index value of the string instead of typing the real text.
To know the indices of the strings use the **Strings Panel**.

The index must be typed in the following way:

#12 For standard string with index = 12 present in [Strings] section
!12 For ExtraNG string with index = 12 present in [ExtraNG] section

If the index of the string is used it is not necessary that the string is present in the **english.txt** but only in the specific **language.dat** file used in the game.

Remark: If you do not want to show a name for the Bar type **IGNORE** or
the null sign for strings ***** (asterisk).

BlinkPercentage
Set from what percentage the Bar will start to blink.

DAMAGE

Damage=

For example: Set 30 as the BlinkPercentage, the Bar will blink when it 30 % or less of the full Bar.

Type **IGNORE** in this field and it will use the default value of 20%

Default values:

If a Damage command is not set in the script for the current level the settings will be:

Damage Rooms: Damage=DMG_INDIRECT_BAR + DMG_SLOW_DISAPPEARING + DMG_ALERT_BEEP, 16, 6, $F6F923, IGNORE

Cold Water Rooms: Damage=DMG_INDIRECT_BAR + DMG_SLOW_DISAPPEARING + DMG_ALERT_BEEP, 10, 5, $f924f1, IGNORE

NEW SCRIPT COMMANDS

DEFAULT_WINDOWS_FONT DefaultWindowsFont=

Syntax: DefaultWindowsFont= IdWindowsFont, FLAGS (**DWF_**), LineSpacing, MainMenuOffsets, NewGameTitle, LoadGameTitle, NewGameList, LoadGameList, OptionSettings, OptionCmdList, PauseScreen, StatList, InventoryItemName, ExamineText

Scope: To use in [Title] section

Type this command in the Title section of the script to force the tomb4 engine to use Windows font characters to show any text in the game.

All menu titles (New game/Load game/Options etc.) will be drawn using windows functions and a windows font set in the WindowsFont= command.

Remark: The advantage to use a windows font is to be able to support different character sets (different from the default western character set).

Practically using windows fonts it should be able to show texts in eastern languages. There are some problems in windows font management. Since the displaying of a windows font is based on the API windows function (and not DIRECTX functions), the drawing will be slower than the default DIRECTX management used in Tomb Raider.

This means there will be a light "flickering" of windows texts on screen.
Another risk is that the font chosen for the windows font could be missing in some computers and in that case the **TRNG** engine will try to locate a similar font but as it is a different format in the game it could be a bad display.

Use the DefaultWindwsFont= only when an eastern language not supported in the Tomb Raider defaults is used.

For western level builders and their players continue using the default font management because it grants better performance and is compatible with all computers.

IdWindowsFont
Type in this field the Id of the WindowsFont script command with the settings for the windows font to use as a default for all system strings.

It is important to place the WindowsFont= command in the script first and then the DefaultWindowsFont that it uses.

Example:
```
ColorRGB=            1, 255,255,255
ColorRGB=            2, 0,0,0
WindowsFont=         1, Arial, WFF_BOLD+WFF_SHADOW, 40, 1, 2
DefaultWindowsFont=  1, IGNORE
```

DEFAULT_WINDOWS_FONT **DefaultWindowsFont=**

Remark: Many settings in the WindowsFont command will be ignored in the game because it isthe game engine that decides when to use a blink text or a primary colour (white), or a secondary colour (yellow).

The size, font name and bold, italic (etc.) settings will be used.

Note: **From 1.2.2.7 version** the WindowsFont allows the FontName field (in the string, pointed by that field) to be typed in, also the character set in the format: Charset:FontName

NEW SCRIPT COMMANDS

DEMO Demo=

Syntax: Demo= DemoFlags (**DEMF_**), Parameter, InfoText ,
DemoLegendText, WaitingTime, DemoIndex array

Scope: To use in [Title] or [Level] sections

Demo command plays the demo data you recorded using the
EDGX_RECORDING_DEMO skill in the **DiagnosticType** command.

See the description of the **EDGX_RECORDING_DEMO** for more information

This command works to show a demo of the levels for the adventure when running the title level.
Place the Demo= command in the [Title] section of the script.

For a Demo command in a [Level] section it works in the custom scene mode,
with some operative differences.

Remark: Remember that the two demo types have NO relationship between them.
When a Demo is in the title level to play demo.pak files,
it is different from a **demo.pak** demo played in a level.
The demo script command in a level section is only used when the player loads
the level.

DemoFlags (DEMF_)

Add one or more **DEMF_ flags** to customize how the demo command will work.
Read the description of **DEMF_** constants.

Parameter

It is possible that in the future some **DEMF_ flags** may require an extra parameter to
type in the Parameter field. Read the description of **DEMF_** constants.

DEMO Demo=

Info Text

To inform players that there is a demo mode and how to enable it,
add a string in the string section and then copy the same text into the **InfoText** field.
The colour and position of the text to be used is set in the
TextFormat= command in the title section.

The text is placed at the center of the bottom line of the screen.

The **InfoText** string can optionally contain a percentage % character that is replaced with the number of seconds remaining to launch the demo, that is a countdown.

For instance, setting a **WaitingTime** of 30 seconds and in the **InfoText** string the text:

A demo will start in %d seconds

In the title level the player will see the time counting down:

A demo will start in 30 seconds
A demo will start in 29 seconds
A demo will start in 28 seconds
A demo will start in 27 seconds etc.

In this case set the **DEMF_PLAY_ON_KEY** flag,
Always set a text to inform the player, otherwise he will not know what key to press.

So type Information Text like:

Hit "1" or "2" or "3" keys to see a demo of different locations of this adventure.

In the above case, there are three levels demo data, the first, second and third.

See description of the **DEMF_PLAY_ON_KEY** for more information.

DEMO Demo=

DemoLegendText

This field stores text to show while the demo is playing.

Do not confuse this text with the **InfoText** string.

The **InfoText** is shown in the title level to inform players about how to start the demo.
It should be text like "Demo will begin in %d seconds" or "Hit 1 or 2 to show a Demo".
The demo legend string is only drawn in the level (not the title) when the demo is
in progress. It could be text like: **Demo hit Escape for Menu**.

This text isprinted using the TextFormat setting but in this case it is the textformat command of the level where the demo is played.

Waiting Time

A value is used as a countdown to launch a demo.
The time in seconds will decrease.
Do not set any input command for the **WaitingTime** before launching a demo.

If **IGNORE** is set the countdown is not used.
In this case add the **DEMF_PLAY_ON_KEY** flag to the Demo Flags field, otherwise no demo will be launched.

Note: The countdown method and the **DEMF_PLAY_ON_KEY** flag can be combined.
In this situation the demo begins with a user input (hit the key) or because the elapsed time of the countdown has been completed.

DemoIndex array

In this array type one or more Ids for the **DemoId.pak** file created with the recording feature of the **EDGX_RECORDING_DEMO** flag.

Notes: The Demo#.pak files have to be stored in a Data subfolder
The name syntax of progressive Demo files is like this: **Demo1.pak**
Demo2.pak
Demo3.pak
Do not use **IGNORE** in the DemoIndex array fields.

NEW SCRIPT COMMANDS

DETECTOR Detector=

Syntax: Detector= Flags (**DTF_**), MetricScale, MetersOfRange, Target Items array

Scope: To use in the [Level] section.

In the **TRNG** engine there is a new object called the "Detector".

With the Detector Lara is able to locate the position of targets in real time.

There are two models of the Detector:

The **Pointer Detector** that is found in the ng2.wad

and The **Radar Detector** that is found in the ng.wad.

For both Detectors set a Detector script command in the **script.txt** to enable it and set its features.

Fields of Detector script command:

Flags (DTF_)
Add different **DTF_** constants in the first field of the Detector command to set its features.

Currently there are the following **DTF_** flags:
DTF_ENGAGE_ALWAYS
DTF_ENGAGE_IN_RANGE
DTF_ENGAGE_INVENTORY

The **DTF_** flags starting with **ENGAGE** set what way to start the Detector in the game.
Choose only one of the flags.

DTF_ENGAGE_ALWAYS
The Detector is always present on the screen.

DTF_ENGAGE_IN_RANGE
The Detector will automatically be shown when
Lara is close to some target within the given range (see the following fields).

ENGAGE_IN_RANGE
For Example: Set at 50 metres the Detector is shown when Lara
is 50 metres or less from the nearest target.
When Lara is further away the Detector is hidden.

DETECTOR **Detector=**

The engage mode **DTF_ENGAGE_INVENTORY** sets the Detector like Lara's other equipment.

She has to pick it up.
The Detector is in the **QUEST1** slot, (**AMULET OF HORUS**) and select it in the inventory.
When Lara selects it in the Inventory the Detector will be shown.
To remove it from the screen select it in the inventory choosing "Cancel".

DTF_REQUIRED_ITEM
This works with the **DTF_ENGAGE_INVENTORY** to set the **QUEST1** item as necessary to ENABLE the Detector.

For example: Use the **DTF_ENGAGE_ALWAYS + DTF_REQUIRED_ITEM** and the Detector is always shown if Lara has picked up the item.

If the **QUEST1** item is absent from the inventory the detector is not shown.
For the engage mode **DTF_ENGAGE_INVENTORY**
it is not necessary to set the **DTF_REQUIRED_ITEM** flag as it isadded by the **TRNG** engine.

For with **DTF_ENGAGE_IN_RANGE**, the **DTF_REQUIRED_ITEM** requires two conditions to show the Detector:

The **QUEST1** item has to be in the inventory (Lara picked it up) and a Target item is within the specified range.

DTF_RADAR_MODE
This sets the Detector in the Radar Mode.
If this flag is omitted the Detector will work in Pointer Mode.
There are big differences between the two modes and some fields and flags only work for a specific Detector mode.

In the Pointer Mode: The Detector has a pointer like a compass that shows the current target. The current target is the first target in the list present in the game.

The 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 to the second target in the list and so on.

In the Radar Mode: The Detector is able to scan all of the targets at the same time that are in the range of the radar.

Note: Remember that the range of the radar is different to the range for activation.
The range of radar is given by the formula **6 * MetricScale**,
where the digit "6" is the (fixed) number of grid sectors of radar.

DETECTOR **Detector=**

For example: Set 2 metres for the metric scale and the target is only shown on the radar when it is 12 metres or less from Lara. (6 * 2 = 12)

Radar Mode is more difficult than Pointer Mode to understand because the target in the Radar Mode is always shown with upwards = North.

In the 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 in the correct direction of the target.

In Radar Mode it is advisable that Lara looks to the north to understand if the targets are at her left or right.

DTF_FAST_RADAR_SCAN
This is only used with the Detector in the Radar Mode.
If the **DTF_FAST_RADAR_SCAN** is set the scanning of targets is faster.
In Radar Mode the position and distance of targets is only updated when the hand of the radar "touches" the target.

DTF_SWINGING_POINTER
This only works in the Pointer Mode.
By default the pointer points at the target.
It adds a swinging simulation to the pointer like the compass in the inventory to get a more realistic Detector.

DTF_INVERSE_VPOINTER
This only works in Pointer Mode.
The position of the line on the vertical scale at the right of the detector can be inverted.
By default when this flag is not used the floating line shows where Lara is.
The fixed red pointed line at the center shows the position of the target.
The floating point is the vertical position of the target.
The fixed red line is the vertical position of Lara.

DTF_NONE
If no flags are required use a **DTF_NONE** flag.

MetricScale
Set a value in metres that isused to assign the distance for each sign of the vertical scale or grid table for the Radar.

Set **IGNORE** in this field and the default value 2 metres is used.
A block in the game is 2 metres so setting 2 as a metric scale each line in the Detector will define a distance of 1 block (2 metres).

DETECTOR **Detector=**

For example: In the **Radar mode** the Detector shows a target at 3 squares (Radar) from Lara (center of the Radar panel).
This means that the target is 3 sectors and 6 metres from Lara.

In the **Pointer Mode** the metric scale only works for the vertical panel.
The vertical strip at the right of the Detector.

In **Radar Mode** it is very important that the value set is a metric scale because only targets in the radius of **6 * MetricScale** metres are valid.
The six value is the number of squares of radius of the Detector.

For example: If the Radar shows targets at 100 metres distance (50 blocks) set a metric scale (100 / 6) = (about) 16 metres.

So for the Pointer Mode it is better to use a low value for the metric scale (the default value "2" is good).

For the Radar Mode it is better to use big values.

MetersOfRange
This is only used if the **DTF_ENGAGE_IN_RANGE** is set.
The value is the distance (in metres) of the targets to show in the game Detector.
If the Radar Detector is used it should have a reasonable range set with the value of range visibility computed by the **MetricScale**.

For example: Set 2 (default) as the Metric Scale.
This means targets are shown when they are (2*6) 12 metres from Lara.
Now set the same value (12) for the **MetersOfRange** so the Detector is only shown when there is something to show
(a target in the visible range).

Target Items array
Type one or more indices of items (only moveables) seen in the **Tomb Editor** program.
If two or more indices are typed, split them with commas.

Example: Detector =

DTF_REQUIRED_ITEM+DTF_ENGAGE_IN_RANGE+DTF_SWINGING_POINTER +
DTF_INVERSE_VPOINTER, 2, 30, 365, 372, 373, 375, 377

In the above row the "2" value is the Metric Scale,
the "30" is the MetersOfRange,
all the other numeric values (365, 372, 373, 375, 377)
are indices of the moveables to monitor.

DETECTOR **Detector=**

Remark: The list of item array works differently for Radar or Pointer Mode.

Pointer Mode.
In Pointer Mode the Detector only shows the position of one item at a time.

In the above example the first position shown is the target with the index 365.
Only when the first index (365 in the sample) is picked up or killed (if it is a BADDY)
the Detector starts to follow the second item (372 index in the example).

Knowing this mode of working it is clear that non-killable or pickable items
(like a door or an animating) should not be used because the next item in the list will never be followed by the Detector.

To set a non- killable or pickable item as a target do the following:

Use the **Radar Mode Detector** as it works on all targets of the list at the same time.
Or place the index of the non killable item at the end of the list.

There is also another way: Use an ACTION trigger
to remove (kill) this item when Lara reaches it by walking
on a square with a kill trigger.

Radar Mode
In the Radar Mode all targets are followed at the same time.
Only targets within the visibility range (**MetricScale * 6**) are shown on the Radar Screen.
The Radar panel shows targets in different ways according to their vertical position.
When a target is at same height (floor) as Lara the target is shown as a little blinking circle.
When a target is in front of Lara (4 or more metres) it is shown as a triangle with the apex pointing to North.

When a target is behind Lara (4 metres or more) the triangle points South.

Same shapes are shown in the vertical panel (the row at the right of the Detector).
Targets at the same height of Lara (+2 to -2 sectors) are shown as a circle.

NEW SCRIPT COMMANDS

DIAGNOSTIC **Diagnostic=**

Syntax: Diagnostic= ENABLED/DISABLED

Scope: To use in the [Options] section

Enable the diagnostic with Diagnostic=ENABLED.
In the game information is shown on the screen in real time about the number of the current Lara animation, the current State-Id of Lara and the current effective frame rate in the game.

This information is useful to discover the number of some animation.
Use this number to locate the animation in the wad file using the **Animation Editor** of the **Wad Tool** program.

If there are problems reading the information in real-time:

Use the **Tomb4_Logger.exe** program to catch all of the diagnostic messages and store them in a text file.

Start the **Tomb4_Logger.exe** before starting the **TRNG** engine with

Diagnostic=ENABLED to have a log file.

To get information about the LoadCamera in the log file press the the **F1** key.

This is a sample of text that can be obtained in this way:

21656: LoadCamera= 6656, -2817, 14123, 6656, -2631, 12800, 2
21672: Animation=103 StateId=2 ($2)

NEW SCRIPT COMMANDS

DIAGNOSTIC_TYPE **DiagnosticType=**

Syntax: DiagnosticType =DiagnosticType (**DGX_**), Extra Dgx flags (**EDGX_**)

Scope: To use in the [Options] section

This command works with Diagnostic=ENABLED.
To enable the diagnostic set the type of Diagnostic information to show using the DiagnosticType command.

DiagnosticType (DGX_)
Type one or more **DGX_** constants to enable the specific Diagnostic information you want to see.

Extra Dgx flags (EDGX_)
This extra field is required by the **EDGX_** constant to customize the Diagnostic type chosen.

See the **DGX_** constants to know if they accept an **EDGX_** value in this field.

NEW SCRIPT COMMANDS

DIARY **Diary=**

Syntax: Diary= Id Diary, SlotDiaryItem, LaraDiaryFlags (**LDF_**), BackGroundImageId, Default PageLayout (**PL_**), FirstString, TitleWFontId, CommonTextWFontId

Scope: To use in the [Level] section

This command ENABLES Lara's Diary and permits the setting for the Diary.
Lara's Diary should be a Log where there are texts and images to describe the adventure that is going on.

There are flip effects to add new pages to Lara's Diary to update the Diary in accordance with the evolution of the adventure.

The player is able to select this item in the inventory like it was an Examine object going into the Diary mode to browse and read an unlimited number of pages, texts and images.

Remark: At the bottom of the description of the Diary= command, there is a description about the special text formatters used to insert text to show in the Diary.

Id Diary
There can be up to 10 different Diaries in the same level.
To distinguish the Diary to use specify its Id in a flip effect trigger.

SlotDiaryItem
Type the slot corresponding to Lara's Diary item.
The slot has to be an inventory item and it is suggested to use a QUEST item:

QUEST_ITEM2
QUEST_ITEM3
QUEST_ITEM4
QUEST_ITEM5
QUEST_ITEM6

Do not USE QUEST_ITEM1 as this is used for the Detector.

The example of "Lara's Diary" on the **TRNG** website uses the **QUEST_ITEM2.**

DIARY Diary=

LaraDiaryFlags (LDF_)
Type one or more **LDF_** flags to set the Background audio track when the Diary is displayed.

If **IGNORE** is typed no change will be made and the audio track of the game will continue as the Diary is displayed in the Inventory.

To stop the game audio track and set another custom background music while the player is watching the Diary set the correct **LDF_** flags.

An important flag is the **LDF_CONTINUE_DIARY**.
Use a unique meaningful value when a Diary is started from a previous level.

See the **LDF_** constants

BackGroundImageId
Lara's Diary in the Examine mode is a bi-dimensional item and it is necessary to set the image for the background.

In this field type a number for a bmp image in this format: IMAGE4.bmp

For example: To use the above image as a background type the value 4 into the BackGroundImageId field.

Remark: The bitmap image can have any size and it will be resized to fit the screen. However use an image of at least 640x480 pixels, or better i.e. 800 x 600 pixels or higher

If the **TRNG** engine detects a wide-screen monitor, the background image will NOT be resized to the full screen but will be enlarged to fit the height of the screen.
The width will be set to preserve the size ratio to avoid distortions of the image.

It could happen there are two vertical columns at the sides of the Diary image.
Supply the image#.bmp file with the level files.
The **TRNG** engine is able to find images in these folders: **root folder**
(i.e. in same folder as the tomb4.exe)
\PIX sub-folder

Currently the **TRNG** engine only supports images in a bitmap (.bmp) format.
Images in jpg format can be converted to bmp using the utility **CONVERTER.exe**

Default PageLayout (PL_)
Type the **PL_** flags to choose the default layout for the Diary.
This means the positions of text, title and images to display on the same page.
Different layouts are set to display the above elements.
Try to create a background image in accordance with the required layout.

See the **PL_** constants

DIARY **Diary=**

FirstString
To initialize the Diary with the first text string.
The text typed in this field has to be the same as that in an ExtraNG string.

Remark: Type in the ExtraNG string a file-string in the form: **@MyText.txt**
and then type the text in a file named **MyText.txt** and save it in the Script folder.
If no first string is required type **IGNORE** in this field.

Remark: Add strings to the Diary using the flip-effect **Diary. Add (&)NG String to Diary**.
It can type one or more pages at the same time.
In the ExtraNG string there is a #END_PAGE# tag to separate the pages.

TitleWFontId and CommonTextWFontId
To set the colour and size of the text type a WindowsFont= command script and then place an Identifier for that WindowsFont= command in the Diary.

The WindowsFont= commands used by the Diary command have to be typed before the Diary command.

In the **TitleWFontId** field type the Id for the title.
The title will have a larger font size than the common text.
Type the Id for the common text in the **CommonTextWFontId** field.

The Text Formatters to use in text pages:
The settings typed in the Diary command will work as the default values.
All of these settings can be changed for each page.
In each page insert special formatters used to inform the **TRNG** engine about the settings to change in the Diary.

The text used for these settings will not be displayed on the screen.
All of the textual settings have to be enclosed in a couple of tags <FORMAT> and <END_FORMAT>.

Type only one <FORMAT> tag for each page.
Type a FORMAT section in each page.
If a FORMAT section on a page is omitted that page will use the default settings typed in the Diary command and NOT the FORMAT setting of the previous page.

In the FORMAT section type one or more of the following text tags:

#TITLE_FONT#	=	Id of WindowsFont for title.
#TEXT_FONT#	=	Id of WindowsFont for common text.
#PAGE_LAYOUT#	=	***PL_*** constants.
#BG_AUDIO#	=	Identifier of the audio track to play as background for this page.
#BG_IMAGE#	=	Identifier of the image to use as a new background.
#POP_IMAGE#	=	Identifier of the image to show in this page.
#TITLE#	=	Text to show at the start of this page as a Title.

DIARY **Diary=**

How set the end page signal:
A very important tag to use in the Diary text is the end of page tag.
Type the text and indicate the page end and the page start position.
Type in the text the tag :#END_PAGE#.

It is important to place the #END_PAGE# on a empty line.
This means DO NOT END A PAGE LIKE THIS:

---------- **WRONG METHOD** --------
The key is in a bottomless pit. #END_PAGE#
Now Lara should find the key but to reach the pit she has to search...

The above position is wrong.
The correct mode is to place the end page like this:

---------- **RIGHT METHOD** ----------
The key is in a bottomless pit.
#END_PAGE#
Now Lara should find the key but to reach the pit she has to search...

To give an idea about how to use the Diary see the following example:
-------- **Example of text used for Diary** ---------

```
<FORMAT>
#TITLE_FONT#=1
#TEXT_FONT#=2
#PAGE_LAYOUT#=PL_WIDE_IMAGE+PL_ADD_INFO_BAR
#BG_IMAGE#=1
#POP_IMAGE#=2
#TITLE#=Mission Organizer
<END_FORMAT>
This is the Mission Organizer of Lara
In these pages you can verify the targets Lara has to reach.
New pages will be added as the mission goes on.
Remember to come back to read this document to get important information.
#END_PAGE#
<FORMAT>
#TITLE_FONT#=3
#PAGE_LAYOUT#=PL_DOUBLE_PAGE+PL_ADD_INFO_BAR
#BG_IMAGE#=3
#POP_IMAGE#=4
#TITLE#=Beginning...
#BG_AUDIO#=56
#FRAME_IMG#=20,10,400,500
#FRAME_T1#=520,10,400,500
#FRAME_T2#=20,600,400,400
<END_FORMAT>
```

DIARY **Diary=**

Lara has to find the man who stole the security key.
Last time he was seen in the dark castle on the hill.
... other text ....
The man is very dangerous because he has a mysterious power...
#END_PAGE#

---------- End of Example of text for Lara 's Diary ----------

That is a short sample of a Diary with two pages and different tag formatters.
Looking at the text in the first page is:

<FORMAT>
#TITLE_FONT#=1
#TEXT_FONT#=2
#PAGE_LAYOUT#=PL_WIDE_IMAGE+PL_ADD_INFO_BAR
#BG_IMAGE#=1
#POP_IMAGE#=2
#TITLE#=Mission Organizer
<END_FORMAT>

Read the above settings in this way:

"#TITLE_FONT#=1" Print the title using font setting stored in the script command: WindowsFont=1, ...

"#TEXT_FONT#=2"

Print the common text using the settings stored in the script command: WindowsFont=2, ...

"#PAGE_LAYOUT#=PL_WIDE_IMAGE+PL_ADD_INFO_BAR"
For the first page use a layout with a wide image at the top of the screen in the central position. At the bottom of the screen use the space for information Bar drawn on the background image for this first page.

"#BG_IMAGE#=1" As a background image use the file: IMAGE1.BMP
"#POP_IMAGE#=2" Display in the top half of the screen the wide image of the file: IMAGE2.BMP
"#TITLE#=Mission Organizer" At top of the wide IMAGE2.BMP, place the Title:

Mission Organizer

DIARY **Diary=**

Then display in the text:

<<
This is the Mission Organizer of Lara

In these pages verify the targets Lara has to reach.
New pages will be added as the mission goes on.
Remember to come back to read this document to get important information.

>>
All of the above are the introduction page for Lara's Diary.

Then there is the #END_PAGE# tag and the **TRNG** engine will wait until the player hits the RIGHT arrow key to go to the second page.

The settings for the second page are:

<FORMAT>
#TITLE_FONT#=3
#PAGE_LAYOUT#=PL_DOUBLE_PAGE+PL_ADD_INFO_BAR
#BG_IMAGE#=3
#POP_IMAGE#=4
#TITLE#=Beginning...
#BG_AUDIO#=56
#FRAME_IMG#=20,10,400,500
#FRAME_T1#=520,10,400,500
#FRAME_T2#=20,600,400,400
<END_FORMAT>

The meanings are:

"#TITLE_FONT#=3" Display the title using font (color and size) set in script command: WindowsFont=3

"#PAGE_LAYOUT#=PL_DOUBLE_PAGE+PL_ADD_INFO_BAR"

The layout will be a double page where the next image is shown at the top of the left page.
Keep the information bar at the bottom of the screen.

"#BG_IMAGE#=3" Use this image as a background.
IMAGE3.BMP
"#POP_IMAGE#=4" Show at top of the left page the image:
IMAGE4.BMP
"#TITLE#=Beginning..." Show over the IMAGE4.BMP the title:

DIARY **Diary=**

Beginning...

"#BG_AUDIO#=56" Play the audio track stored in the file:

trle\audio\056.wav
(but it could also be 056.mp3 or 056.ogg)

"#FRAME_IMG#=20,10,400,500" Set the position and size where to display the image on the page.

In the Diary Layout Page flags set the **PL_CUSTOM_LAYOUT**.
This setting will apply to following pages.
If another **PL_ LAYOUT** is set this setting of "#FRAME_IMG#=" will only be used for this page.

Other pages will use the **PL_ LAYOUT** set in the Diary command.

The values for this tag (20,10,400,500) are in micro units,
where 1 = 1/1000 of width or height of the screen size.

To compute these values use the utility [**Get Screen Frames**] found in the **Tools Panel** of the **TIDE\ NG_Center 1.5.7** program. It is necessary to load a background image.

"#FRAME_T1#=520,10,400,500"

Set the position and size of the first frame used for the text on the page.
The "first" frame is where the first block of text is placed on the page.
In the layout there can be two text frames.
In this case the "#FRAME_T1#=" tag sets the first of two text frames.
This tag requires micro units.

See the description of the **#FRAME_IMG#=** tag for more information.

"#FRAME_T2#=20,600,400,400"

Set the position and size for the second text frame.
When the layout has two text frames use the #FRAME_T2#= tag to set the position of the second block of text on the page.
This tag requires micro units.

See the description for the **#FRAME_IMG#=** tag for more information.

DIARY **Diary=**

Remarks:

The background sounds in the Diary mode will only work if the BASS features in the level are ENABLED.

If an #END_PAGE# is placed too low to make the text visible on the screen the player will not be able to view some of the current page and no extra text will be shown over the information Bar.

When a tag formatter is missing the **TRNG** engine will use the default setting of the Diary script command.

To stop a background sound set in a previous page,
but not start another audio track,
type the tag formatter: #BG_AUDIO#=-1

The previous audio track will stop and the default audio track in the
Diary= script command will be played.

To use large text use the trick to set in the ExtraNG strings the file reference:
45 @FirstPages.txt

The **FirstPages.txt** has to be in the Script folder when the **script.txt** is built.
Once the **script.dat** and the **english.dat** are built the **FirstPages.txt** file is not necessary as its contents are compiled into the **english.dat** or another language file.

NEW SCRIPT COMMANDS

DEMO Demo=

Syntax: Demo= DemoFlags (**DEMF_**), Parameter, InfoText , DemoLegendText, WaitingTime, DemoIndex array

Scope: To use in the [Title] or [Level] sections.

The Demo command is of service to play the demo data recorded using the **EDGX_RECORDING_DEMO** skill in the DiagnosticType command.

See the description of the **EDGX_RECORDING_DEMO** for more information.

This command works to show a demo of the levels of the adventure when the player is in the title level.

In this situation place the Demo= command in the Title section of the script.

If a Demo command is used in a Level section it will work in a custom scene mode, with some operative differences.

Remark: Remember that the two demo types have NO relationship to each other.
When a Demo is set in the level to play a **demo.pak** file,
it is different from the title where the **demo.pak** is played.
The Demo script command in a level section will only be used when the player loads that level and not the Demo mode from the title.

DemoFlags (DEMF_)
Add to this field one or more **DEMF_** flags to customize how the Demo command works.
Read the description of the **DEMF_** constants

Parameter
It is possible that in the future some **DEMF_** flag will require an extra parameter in the Parameter field. Read the description of **DEMF_** constants

DEMO Demo=

InfoText
To display text in the title level to inform players that a demo mode exists and how to enable it, add a string in the string section and then copy the same text into the **InfoText** field.

The colour and position of the text uses the setting of the TextFormat= command in the title section, and the text is placed at the center of the bottom line of the screen.

The **InfoText** string can optionally contain a percentage % character that is replaced with the seconds remaining to launch the demo, like in a countdown.

For instance, to set as WaitingTime 30 seconds set in a InfoText string the text:

A demo will start in %d seconds

Then in the title level the player will see the time counting down:

A demo will start in 30 seconds
A demo will start in 29 seconds
A demo will start in 28 seconds
A demo will start in 27 seconds etc.

In this case set the **DEMF_PLAY_ON_KEY** flag.
Always set a text to inform the player what key to press.

So for example type a **InfoText** such as:

Hit "1" or "2" or "3" keys to see a demo of different locations of this adventure.

In this example case, there are demo data for three levels, the first, second and third.

See the description of the **DEMF_PLAY_ON_KEY** constant for more information.

DemoLegendText
This field stores a text to show when the demo is playing.
Do not confuse this text with the InfoText string.

The **InfoText** is shown in the title level to inform players about how to start the demo and it should be a text like: **Demo will begin in %d seconds** or **Hit 1 or 2 to show a Demo**.

The **DemoLegendText** string is only drawn in the level.
When the demo is in progress it should display a text like: **Demo hit Escape for Menu**.

This text is printed using the **TextFormat** setting but in this case it is the **TextFormat** command in the level where the demo is played.

DEMO Demo=

WaitingTime
Set in this field a value different from **IGNORE** (or -1).
This value is a seconds countdown to launch a demo.
The time in seconds decreases.
Do not set any input command for the WaitingTime before launching a demo.

If **IGNORE** is set the countdown method is not used.
In this case add the **DEMF_PLAY_ON_KEY** flag to the DemoFlags field,
otherwise no demo is launched.

Note: The countdown method and the **DEMF_PLAY_ON_KEY** flag can be added.
In this situation the demo begins for an user input (hit the key) or
because the elapsed time of countdown has been completed.

DemoIndex array
Type one or more Ids for the **DemoId.pak** file created with the recording feature
of the **EDGX_RECORDING_DEMO** flag.

Notes: The **Demo#.pak** files have to be stored in the Data subfolder
The name syntax of the progressive Demo files is: **Demo1.pak**
Demo2.pak
Demo3.pak

Do not use **IGNORE** values in the **DemoIndex** array fields.

NEW SCRIPT COMMANDS

ELEVATOR Elevator=

Syntax: Elevator=ElevatorIndex, ClickFloorDistance, NumberOfFloors, Elevator flags (EF_), FirstDoorIndex, InnerKeyPadIndex, Speed, Frame items array

Scope: To use in the [Level] section.

Arguments:

ElevatorIndex
Index in the **Tomb Editor** in the yellow box when you click on the elevator item.

ClickFloorDistance
The number of clicks for the height of each floor.
For example if there are three blocks between each floor type "12" in the field as each block is 4 clicks.

NumberOfFloors
The number of floors for the elevator.
For example if you type 3 it means the elevator will move to 3 different floors.

ElevatorFlags
Look in the **EF_**constants for the Elevator flags information.
Place one or more flags using the "+" sign or type the sum of the values.
Remember that some flags are incompatible to use at the same time, like Multi-doors and single-door flags.

$0001 EF_MULTI_DOORS
If the **TRNG** engine has to handle all doors on any floor set this flag and then set in the **IndexFirstDoor** the index of the door at the first floor.

The management of the **TRNG** engine is very simple.
When the elevator reaches a floor the door opens.
When the elevator leaves a floor the door on that floor will close.
If you use the multi-doors handling it is important to place all of the doors at the same vertical distance set in the **ClickFloorDistance** field.

It is important because if a door is at a different height it will not be found by the **TRNG** engine and there will be problems.

Place each door at the same height where the elevator stops floor for floor.

ELEVATOR **Elevator=**

$002 EF_SINGLE_DOOR

The multi-door flag can not be set when using the **SINGLE_DOOR** mode.

In the SINGLE_DOOR mode the elevator has a single door linked with the elevator cage.
This door will be moved up and down with the elevator.
When the elevator reaches a floor the door will open.
When the elevator starts from a floor the door will close.
If this flag is set in the FirstDoorIndex field the index for the door is placed near to the elevator.

Remark: Use a "fake" door found in the slot **ANIMATING2** in the ng2.wad
This animating will slide horizontally like a door if the **SINGLE _DOOR** mode is used and its index is set in the **FirstDoorIndex** field.

There is a reason to use a fake door like this.
When a real door is used in some circumstances
(it depends by door type and by position of door in respect to linking rooms)
the **TRNG** engine will add an invisible sector collision in front of the door when it is closed. The fake animating door avoids this problem.

$004 EF_INNER_KEYPAD

To insert the keypad to choose the floor inside the elevator set this flag and in **Tomb Editor** place the keypad oriented on the elevator wall.

Then type the index of the keypad in the **InnerKeyPadIndex** field.
Remember to use the "fake" keypad found in the **ANIMATING16_MIP** of the ng.wad or the ng2.wad and not the real keypad in the **SWITCH_TYPE1**.

The reason for this is because of a technical problem.
The SWITCH_TYPE requires a switch trigger to work but on the elevator floor there are dummy triggers. The management of the "fake" keypad of the **ANIMATING16_MIP** will be performed in a hard-coded mode by the **TRNG** engine.

The keypad will work in the same way as a **SWITCH_TYPE1** but with no need for a special trigger.

$0008 EF_MODE_YO_YO

YO_YO mode is a bit curious.
The elevator will move up and down without stopping at the floors.
Using this mode there are NO DOORS as there is no time to open and close them.
Lara will have to jump in and out of the elevator.
An interesting application of the **YO_YO** mode is to give the elevator the role of a Trap.
Lara can be killed if she is under the elevator when it moves down, or over the elevator when it moves up to the ceiling.

Remark: **Do NOT mix YO_YO mode and STOP_AND_GO mode.**
If the elevator is set for **YO_YO mode** the number of floors will always be two (2) and the distance between the floors will be the height of the whole movement that the elevator will cover before reversing direction.

ELEVATOR **Elevator=**

$0010 EF_MODE_STOP_AND_GO
The elevator stops and stays for some time at each floor.
Two seconds for an elevator with no doors and three seconds for an elevator with doors.
Place the doors (multi or single) and set a number of floors.
The **STOP_AND_GO mode** may hurt Lara.

$0000 EF_NONE
If you do set any flags type **EF_NONE** or 0 (zero)

FirstDoorIndex
The Index of the first door, the single door or multi-door.

InnerKeyPadIndex
The Index of the keypad inserted in the elevator.
Set the **EF_INNER_KEYPAD** flag when there is an inner keypad

Speed
Set the speed for the elevator.
Typing **IGNORE** in this field sets the default speed of 20.
Remember Speed units are 1/1024 of sector. i.e. : 1024 = 1 sector, 256 = 1 click.
Each second will change the position 30 times.
DO NOT exaggerate the speed value.
The maximum value suggested is 50.
Larger values will give trouble with collision when the elevator hits Lara.

Frame Items array
After the Speed field there is an optional list of item indices.
It is optional so you do not have to type anything after the Speed field.

The frame array is useful for many targets.
All items typed in this array will move up and down with the elevator movement, giving the impression of being linked to the elevator cage.

ELEVATOR **Elevator=**

For example: A chain can be created to place over the ceiling of the elevator to get a more realistic movement of the elevator.
In the ng2.wad, **ANIMATING5** is used in the samples for the **YO_YO** and **STOP_AND_GO** Elevators.

Place different instances of this chain to cover all of the height for the elevator moving.
Then type all the indices of the chains in the elevator script command.
When the elevator is moved in the game the chain will be moved.

Another use of the frame array is to place invisible collision panels around the elevator.
The collision panel is found in the **ANIMATING1** of the ng2.wad (1x2 sectors) and the **ANIMATING1_MIP** (1x1 sector to use when a little door for the elevator cage is set up).
Only use the collision panels if the elevator is outdoors, i.e. without room walls around it.

If the elevator is at the center of a large environment it is necessary to place collision panels.
Otherwise Lara is able to walk through the elevator walls of the cage.
In the **YO_YO** and **STOP_AND_GO** elevator sample collision panels are used.
Place the collision panel over each elevator wall.
Cover all of the walls except the door of the elevator cage.

Then type all of the indices of the collision panels in the Frame Array and the collision will move up and down making the elevator cage a real closed box.

Place about 23 frame indices in the Frame array.
Collision panels can have a maximum of 8 items.

Remarks: When linking the elevator and other items with the elevator keypad, SINGLE_DOOOR, chain, collision panel etc. in the project the elevator must be at the first floor.

To have the elevator at another floor at the start of the level use the elevator flip-effect to move it to the required floor.
The elevator **MUST BE** on the first floor in the project phase.

The maximum number of floors is 10.
Lara selects the 10th floor with the keypad value "0"

There can be up to 20 elevators for a level working with different modes in accordance with the different Elevator script commands

NEW SCRIPT COMMANDS

ENEMY **Enemy=**

Syntax: Enemy=Slot, HP (vitality), **NEF_** flags, TombFlags, **EXTRA_** flags, DAMAGE1, DAMAGE2, DAMAGE3

Scope: To use in the [Level] section

This command permits different settings for the specific Slot objects.
Set the Health Points (vitality, i.e. resistance to shot by Lara),
the Damage they give to Lara (DAMAGE1, DAMAGE2 ..).

Use flags like NG Enemy Flags (**NEF_**) or Tomb Enemy Flags to set a particular behaviour of the slot.

Only use this command Slot Enemy= for moveable enemies (BADDY) or Traps.
This command can be used to change all of the moveables but many settings have no effect.

Arguments:

Slot
Wad slot to modify with this command.
See the **SLOT MOVEABLE** list

HP (vitality)
Set the resistance of this enemy to Lara's shots.

For example: Set a Large value (1000) will make the enemy difficult to kill.
Set a small value (20) and it will be killed with a few revolver shots.

ENEMY Enemy=

NEF_ flags
This field accepts values of **NEF_** constants.

See the **NEF_** constants

Currently these are the **NEF_** flags:

NEF_EXPLODE	When the creature is killed it explodes.
NEF_EXPLODE_AFTER	When the creature is killed the body will explode only after the death animation is complete.
NEF_HIT_BLOOD	Can only be set once.
NEF_HIT_DEFAULT	
NEF_HIT_FRAGMENTS	Can only be set once.
NEF_HIT_SMOKE	Can only be set once.

The **NEF_HIT** flags set the behaviour of Lara's shots on the creature.
If no **NEF_HIT** is set the shots will have no effect

Using the **NEF_HIT_DEFAULT** the response to shots will not be changed.

NEF_NON_TARGET	Enemy becomes non-target from Lara and it will be immortal.
NEF_NONE	Reserved, do not use. For no **NEF_** flags type **IGNORE** in this field.
NEF_ONLY_EXPLODE	The enemy will become like a Skeleton or Mummy. Common ammunition will have no effect but Explosive ammunition can destroy it.
NEF_SET_AS_CREATURE	Experimental. You can use non-enemy and non-trap slots and use this flag to transform it into an ENEMY. Lara will aim at this object and kill it.
NEF_SET_AS_MORTAL	This flag is be used with slot for the Demi God to transform it into a MORTAL ENEMY. When this flag is set the ENEMY can be killed with common ammunition.

ENEMY **Enemy=**

TombFlags
This field is only for experiment.
It corresponds to the Flags field of the slot structure.
Wrong values in this field can cause problems.
Set this field to **IGNORE** and only use the **NEF_** flags to modify the behaviour of the object.

To experiment with the the TombFlags field here are some values for well-known slots:

$3A7B	=	Used for **SKELETON**.
$327B	=	Used for **VON_CROY**
$3072	=	Used for **GUIDE**
$267B	=	Used for **BADDY_1**
$0472	=	Used for **BIG SCORPION**

EXTRA_ flags
For most slots set this field to **IGNORE**.
There are some slots that require an extra value to be typed into this field.
Each **EXTRA_** constant only works for one specific SLOT.

See the **EXTRA_** constants

Damage1/2/3
Type **NONE**, or one or more values for the Damage that the current enemy causes to Lara.
The reason to accept a variable number of values is because some Slots do not damage Lara, while other moveables can damage Lara in different ways.

BADDY_1 can hurt Lara using a UZI or a sword.
In this situation type a different damage value, one for each type of damage.
To know when an ENEMY has one, two, three or NO DAMAGE effect

ENEMY **Enemy=**

Remark: To modify only one damage value but no others use **IGNORE** to specify "do not change this damage kind".

For example to only change the Grenade Damage for the SAS but not change the UZI Damage type the following line:

Enemy=SAS, IGNORE, IGNORE, IGNORE, 0, IGNORE, 1000

The above command does not change Health Points, NEF_ flags, TombFlags.

Set the EXTRA to 0 and IGNORE for the UZI Damage, change the Grenade Damage to 1000.

Remark: In the Damage1/2/3 fields type only positive values.
In some circumstances type Negative values but the result will be to recharge Lara's vitality instead of hurting her.

Follow the Range limits
See the **DAMAGE ENEMY** list.

Setting a value outside of the range could give unpredictable results.

NEW SCRIPT COMMANDS

EQUIPMENT **Equipment=**

Syntax: Equipment= SLOT item, Amount

Scope: To use in the [Level] section

Equipment forces the amount of the specific Slot item in the inventory.
By default Lara starts the first level or a level after a Reset-Hub command with the same equipment for MEDIPACK, WEAPONS, BINOCULARS etc.

Using the Equipment command can force a different amount for each inventory item.

SLOT Item
Type in this field a value to identify the slot corresponding to the item to set.

The full list of slot items can be found in the **SLOT MOVEABLES.**

Remark: Not all slot names will be accepted so only use slot names with the word "_ITEM".

Amount
In the amount field set the quantity of the specific item in the inventory.

For example: For NO MEDIPACK for Lara at the start of the current level use

these two commands:
Equipment= BIGMEDI_ITEM, 0
Equipment= SMALLMEDI_ITEM, 0

NEW SCRIPT COMMANDS

FMV **FMV=**

Syntax: FMV= NumberFmv, EnableEscape

Scope: To use in the [Level] section.

NumberFmv is the number of the FMV file.

EnableEscape can have a value of 0 or 1.
Value 1 means: "permit player to skip the movie using the Escape key"

The file name must have the following syntax: "FMV" + [Number] + [Extension]

For example: The following names are correct for FMV files: **fmv4.wmv**
fmv1.mpg
fmv43.avi

It is **WRONG TO USE**:

fmv04.wmv	There is a no meaningful "0" in front of "4".
fmv0.avi	Numbering starts from "1".
fmv41	Missing extension.
fmv5.bik	.Bik extension is not supported by the **Tomb Editor.**

In the script command Fmv=FirstNumber (NumberFmv) is the number within the FMV file name.

For example: The file name **fmv3.wmv** should be written in the script as a command : FMV=3, 0

Remember to set the extension with the command FMV= used in the [PCExtensions] section.

This FMV= command has the same name but **DO NOT CONFUSE IT** as it is a different script command.

The FMV= command is not a new Next Generation command as it was present in the standard **script.dat**.

The **TRNG** engine can now show a FMV in the game.
The **TRNG** engine will look for the FMV file in the following sub-folders:

Folder named **FMVs** in the current folder **\FMVs\...** fmv files
Folder named **Store** in the current folder **\store\...** fmv files

FMV **FMV=**

The **TRNG** engine can play **.AVI, .MPG, .WMV** files.

General Notes for Setting up FMV files

Use the **SET_FORCE_SOFT_FULL_SCREEN** command in the script in the [Options] section.
It permits a non-exclusive full screen mode.
It is the "exclusive" mode that stops FMVs from playing.

See SET_FORCE_SOFT_FULL_SCREEN from version 1.2.2.7

[Options]

Settings= SET_SOFT_FULL_SCREEN **up to version 1.2.2.6**

Settings= SET_FORCE_SOFT_FULL_SCREEN **from version 1.2.2.7**

In the Level block type the FMV= command as defined in the section above.
Place the video files in one of the following sub folders **FMVs** or **Store**.

The video can be customized by the script command:

Customize= CUST_FMV_CUTSCENE, FMV_SHORT_BLACK_RESTART + FMV_FADE_OUT

placed in the Level Block of the script.txt file.

Playing a video at the start of the Level

To have a Video at the start of a level set on the same square as Lara the FMV Trigger to start the video and a Flip-Effect Trigger to "hide the screen".
This stops the game being visible before playing the FMV.

Playing a video during the Level

Set an FMV Trigger with a Flip-Effect Trigger to hide the screen.
Define the FMV in the level section for the script.

Perform a change in game when a FMV has been completed

In most cases use a video to illustrate some important facts in the game.
At the end of the FMV set some change in the game to give a sense to the FMV.
In this case the problem is when to perform the triggers to get these changes.
If these triggers are on the same square as the FMV Trigger the changes could be visible for an instant before the FMV starts playing. This could be bad.

FMV FMV=

To solve this problem use a new Global Trigger named **GT_FMV_COMPLETED**.
Use this Global Trigger command and specify the number of the FMV to detect and then type the id of the Trigger Group to perform when that condition is True
(the FMV has finished).

Global Trigger + Trigger Group used to move Lara's position at the end of FMV2

Exporting: TRIGGER(257:0) for FLIPEFFECT(79)

<#> : Lara. (Move) Move Lara in LARA_START_POS with <&>OCB value in (E)way
<&> : OCB= 1 of LARA_START_POS(517) in sector:(3,1) of Room19
(E) : Keep original sector displacement of Lara

Values to add in script command: $2000, 79, $101

Exporting: TRIGGER(296:0) for ACTION(623)
<#> : JEEP Id:623 in sector:(7,10) of Room0
<&> : Enemy. Move immediately <#>enemy in LARA_START_POS with (E)OCB setting
(E) : OCB= 1 of LARA_START_POS(517) in sector:(5,2) of Room19

Values to add in the script command: $5000, 623, $128

TriggerGroup= 1, $2000, 79, $101, $5000, 623, $128
GlobalTrigger= 1, IGNORE, GT_FMV_COMPLETED, 2, IGNORE, 1

The Trigger Group will move Lara and the JEEP.
In the video Lara drives her JEEP to reach a far place.
See the **NG DEMO section FMV Video**.

Play a video at the end of the Level and load another Level.

There is a final FMV for the level and then at the end of the FMV the next level loads :
Use another Global Trigger **GT_FMV_COMPLETED**,
where a Trigger Group loads the next level.
The only issue is the NG Flip Effect Trigger to load a level with delay:

"Delay. Load <&>level in (E)seconds" - "Forever (use other action/effect to disable it)"
Use this Flip-Effect because is not possible to export a common "FINISH" trigger.

Global Trigger + Trigger Group to load next level
Exporting: TRIGGER(2:1) for FLIPEFFECT(82)
<#> : Delay. Load <&>level in (E)seconds
<&> : 2
(E) : Forever (use other action/effect to disable it)

Values to add in the script command: $2000, 82, $2
TriggerGroup= 2, $2000, 82, $2
GlobalTrigger= 2, IGNORE, GT_FMV_COMPLETED, 3, IGNORE, 2

FMV **FMV=**

For an Export TRIGGER Hiding the Screen
Exporting: TRIGGER(0:0) for FLIPEFFECT(54)

<#> : Screen. Hide screen for <&>time in (E) way
<&> : Forever (use other action/effect to disable it)
(E) : Black screen

Values to add in script command: $2000, 54, $0

EXAMPLE SCRIPT.TXT FILE FROM NG PROJECTS FMV Video.

```
[PCExtensions]
Level=.TR4
Cut=    .TR4
FMV=.WMV

;--------------------------------------
; Options
;--------------------------------------

[Options]
LoadSave=     ENABLED
Title=        ENABLED
PlayAnyLevel=       ENABLED
InputTimeout=       18000          ; frames * seconds = 60x30
FlyCheat=     DISABLED
Security=     $55
DemoDisc=     DISABLED
WorldFarView= 45
Settings= SET_SOFT_FULL_SCREEN              up to version 1.2.2.6
Settings= SET_FORCE_SOFT_FULL_SCREEN    from version 1.2.2.7
Diagnostic= DISABLED
CRS=DISABLED

;--------------------------------------
; Levels
;--------------------------------------
[Level]
Name=         Video FMV Cut Scene
Horizon=            ENABLED
Layer1=             128,96,64,7
Puzzle=             1,Ignition Key,       $0008,$0400,$2000,$3000,$4000,$0002
Puzzle=             3,Canopic Jar 2,      $0001,$0320,$0000,$0000,$0000,$0002
PuzzleCombo=        1,1,Sun Disk,         $0000,$0180,$0000,$0000,$0000,$0002
PuzzleCombo=        1,2,Sun Goddess,      $0000,$04b0,$0000,$0000,$0000,$0002
Puzzle=             5,Golden Vraeus,      $0003,$0300,$0000,$0000,$0000,$0002
Puzzle=             7,Guardian Key,       $0009,$0300,$0000,$0000,$0000,$0002
Key=                2,Hypostyle Key,      $0000,$0400,$0000,$c000,$0000,$0002
```

FMV **FMV=**

;commands for FMVs: ----------

Customize= CUST_FMV_CUTSCENE, FMV_SHORT_BLACK_RESTART + FMV_FADE_OUT
FMV= 1, 1
FMV= 2, 1
FMV= 3, 0 ; forbid skipping of fmv with escape

; this Global Trigger + Trigger Group has been used to move Lara position at end of fmv2
; Exporting: TRIGGER(257:0) for FLIPEFFECT(79)
; <#> : Lara. (Move) Move Lara in LARA_START_POS with <&>OCB value in (E)way
; <&> : OCB= 1 of LARA_START_POS(517) in sector:(3,1) of Room19
; (E) : Keep original sector displacement of Lara
; Values to add in script command: $2000, 79, $101
; Exporting: TRIGGER(296:0) for ACTION(623)
; <#> : JEEP Id:623 in sector:(7,10) of Room0
; <&> : Enemy. Move immediately <#>enemy in LARA_START_POS with (E)OCB setting
; (E) : OCB= 1 of LARA_START_POS(517) in sector:(5,2) of Room19
; Values to add in script command: $5000, 623, $128

TriggerGroup= 1, $2000, 79, $101, $5000, 623, $128
GlobalTrigger= 1, IGNORE, GT_FMV_COMPLETED, 2, IGNORE, 1,IGNORE

;Global Trigger + Trigger Group to load next Level
; Exporting: TRIGGER(2:1) for FLIPEFFECT(82)
; <#> : Delay. Load <&>level in (E)seconds
; <&> : 2
; (E) : Forever (use other action/effect to disable it)
; Values to add in script command: $2000, 82, $2

TriggerGroup= 2, $2000, 82, $2
GlobalTrigger= 2, IGNORE, GT_FMV_COMPLETED, 3, IGNORE, 2,IGNORE

;end commands for FMV ----------
LoadCamera= 89366,-258,48077,88372,-1300,45701,0
Level= DATA\VIDEOFMV,110

NEW SCRIPT COMMANDS

FOG_RANGE FogRange=

Syntax: FogRange=StartLimitDistanceFog, EndLimitDistanceFog

Scope: To use in the [Level] section

Default values in tomb4: StartLimitDistanceFog = 12 sectors
EndLimitDistanceFog = 20 sectors (or LevelFarView)

With the FogRange= command change the limits for the Distance Fog changing the position and density of the Fog.

The Distance Fog is **ONLY VISIBLE** when **Volumetric FX** is **DISABLED**.

StartLimitDistanceFog
The number of sectors from where the fog will be visible.
Only used for Distance Fog.
To increase the effect of the Fog type a negative number in this field.
The larger the number the stronger the Fog density.

Remark: It is not good using negative values because you get an abnormal intensity of Fog color in the transparent texture of the level (like water) or objects (transparent or shining textures).

In some circumstances it is better to reduce the **EndLimitDistanceFog** field when a Distance Fog with high density is required.

In this case set the **StartLimitDistanceFog** field to 0 avoiding negative values.

EndLimitDistanceFog
This value is set with the same value of **WorldFarView** (or **LevelFarView**).
For thick Fog reduce this value.
The value is the number of sectors that Lara can see.

The formula for the Fog Distance works in this way:

It begins a Fog effect from the Start Limit Distance Fog sector from Lara and the Fog effect increases up to the 100% density at the End Limit Distance Fog.

Setting a large value for the **EndLimitDistance** Fog will allow Lara to see to the Far Distance.

Set a small value in the **EndLimitDistance** Fog to only see to the **EndLimitDistance** Fog distance.

NEW SCRIPT COMMANDS

FORCE_BUMP_MAPPING **ForceBumpMapping=**

Syntax: ForceBumpMapping=ENABLED/DISABLED

Scope: To use in the [Options] section

For prepared Bump Map textures use this command to override the settings in the tomb4 set-up and force enabling of the Bump Map in the game.

NEW SCRIPT COMMANDS

FORCE_VOLUMETRIC_FX **ForceVolumetricFX=**

Syntax: ForceVolumetricFX=ENABLED/DISABLED

Scope: To use in the [Level] section

Force Volumetric FX command enables or disables the Volumetric FX setting in the current level. This command is useful to mix Fog bulbs with Distance Fog.

Remark: Enable or disable the Volumetric FX in a dynamic way using new flip-effects with the **Tomb Editor** in the game.

This trick is interesting, "switch off" the fog bulb and "switch on" when an effect of gas or smoke is to be created after an explosion.

If no Force Volumetric FX command is used for the current level then the Set up Menu for **TRNG** engine should be used.

NEW SCRIPT COMMANDS

GLOBAL_TRIGGER GlobalTrigger=

Syntax: GlobalTrigger=IdGlobalTrigger, Flags Global Trigger (**FGT_**),
Global Trigger (**GT_**), Parameter, IdConditionTriggerGroup,
IdPerformTriggerGroup, IdOnFalseTriggerGroup

Scope: To use in the [Level] section

Maximum number of instances for level section: 499
Global Trigger enables some Triggers when a Global event occurs.

To understand the meaning of the Global Trigger start from the local Triggers.

The local Triggers are the common Triggers used in the **Tomb Editor**
(HEAVY, PAD, FLIPON, FLYBY).

These triggers are local because they only work in the located space.
The Global Trigger works for the whole level and will be enabled when some Global event happens.

Arguments:

IdGlobalTrigger
Type a progressive number,
1 for the first Global Trigger of the level, "2" for the second and so on.

It is used to identify the Global Trigger when a flip effect is used to enable or disable a Global Trigger.

Flags Global Trigger (FGT_)
Set options for the behaviour of the Global Trigger command.

See the **FGT_**constants

Remark: For no **FGT_** flags type **IGNORE** in this field.

GLOBAL_TRIGGER GlobalTrigger=

Global Trigger (GT_)
Type a constant value starting with **GT_** prefix to choose the required Global Trigger.

See the **GT_** constants

Current Global Triggers:

GT_ENEMY_KILLED: Specify in the Parameter field the index of the moveable to monitor. When the given moveable is killed the Global Trigger will be started.

Remarks: Read the index of the moveable when in the **Tomb Editor** program and single mouse click on that item.

The index is shown in a yellow box.

You can also get a trigger activation when a creature dies with a local (common) trigger Switch.

If you trigger the enemy 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 of this method is that it is necessary to cover a big surface with triggers.
Using a Global Trigger the **GT_ENEMY_KILLED** the trigger will be activated with any position of the enemy.

GT_SCREEN_TIMER_REACHED:
Start the Global Trigger when the screen timer reaches the supplied number of seconds.
Type the 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, ..."

GT_USED_INVENTORY_ITEM:
This Global Trigger is used to detect when Lara chooses a specific item from the inventory.
Specify the slot Id in the Parameter field.

For example: To enable a Global Trigger when Lara selects the **QUEST_ITEM2** with slot Id = 253, type

GlobalTrigger=1, IGNORE, GT_USED_INVENTORY_ITEM , 253, ...

GLOBAL_TRIGGER GlobalTrigger=

Remarks: 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 this flip-effect is not used the selected item will remain in the inventory for further activations.

Try to avoid using QUEST_ITEM1 for the Global Trigger because it is used for the Detector activation.

Remark: There are many other Global Trigger **GT_** constants.

Parameter
The value to type depends on the type of the Global Trigger chosen in the previous field **GT_ Global Trigger**.

Read the description for the specific Global Trigger to get more information.

IdConditionTriggerGroup
This is optional.
By default the **GT_** Global Trigger sets a Global condition.

To perform the final **IdPerformTriggerGroup** only when further conditions are **TRUE** create a Conditional Trigger Group script command.

Then set in this field the Id of that Conditional Trigger Group to perform another condition. When a Conditional Trigger Group is used the final condition will only be true when the other conditions set in the Global Trigger and the conditional Trigger Group are **TRUE**.

For no additional conditions type **IGNORE** in this field.

IdPerformTriggerGroup
Type the Id of the Trigger Group command that will be performed when the Global condition of the **GT_ Trigger** is **TRUE**.

The Trigger Group can have condition triggers.
These further condition triggers will only be verified if the **GT_** Global Trigger is **TRUE**.

IdOnFalseTriggerGroup
To perform a Trigger Group when the Global condition is **FALSE**,
Type the Id of the Trigger Group to perform on a **FALSE** condition.

NEW SCRIPT COMMANDS

IMAGE **Image=**

Syntax: Image=IdImageCommand, IdImageFile, ImageFlags (**IF_**), EffectTime, AudioTrack, XPosition, YPosition, SizeX, SizeY

Scope: To use in the [Level] section

The Image command sets data to be used for the flip-effect trigger "Show Image ..."

IdImageCommand
Type a progressive number that is used to identify this Image command
from some flip effect trigger.

IdImageFile
Type the number of the image to show.
This number (or Id) refers to the image in the format: "IMAGE<Id>.BMP"

For example: Type 5 in this field to show the image: IMAGE5.BMP

The **TRNG** engine will look for the images in the **\PIX folder** and if it is missing in the PIX folder the **TRNG** engine will look for the image in the **root folder**, i.e. the folder where the **tomb4.exe** program is located.

ImageFlags (IF_)
Specify the **IF_** flags to set features for the image or about the mode to view it.
Type **IGNORE** and the image is shown freezing the game in the position
and size set in the following fields of the command.

EffectTime
This only works for further effects chosen for the image.
If no effects are to be applied to the image this field will be **IGNORED**.
The time is in tick frame units the internal time of the **TRNG** engine.
One tick frame value is 1/30 of second.
Set an effect for the current image and the Effect Time will inform the **TRNG** engine about how many frames will be necessary to complete the effect.
Small values and the effect is fast.
Large values and the effect will be slower.
Type **IGNORE** to use the default value of 30 ticks, i.e. one second.

IMAGE **Image=**

AudioTrack
Type a number between 0 and 255 to choose the audio track in the audio folder to play.
For example typing 35 the sound 035.wav will be played.

XPosition and YPosition
In these two fields type the origin with respect to the Tomb Raider screen where the image is placed. It is important to understand that the units used for position and size are **not in pixels**.

The reason is because of the difficulty of knowing in advance the current Tomb Raider screen resolution.

When a level is created the **script.txt** does not know if the game will be played at 640 x 480, 800 x 600 or 1024 x 768 etc.

Setting the origin of the image or its size in pixels will be a problem.
The origin and size of the image is given in proportional units named "micro units" (1/1000 of screen).

For example: To locate the center of the game screen the values will be XPosition = 500 and YPosition=500.

The whole width and height of the game screen will be 1000 x 1000 micro units.

SizeX and SizeY
In these two fields set the size of the image once it has been drawn on the game screen.
The values are in micro units.

See the description of the **XPosition** and **YPosition** fields.

There may be a problem to set the current ratio of the image because the height of the screen (about Y size) will be computed in micro units (1/1000).

For example: If the source image is 100 x 100 pixels,
it cannot set a target size 200 x 200 micro units,
because the size Y 200 computed on Screen Y should be bigger
than computed on Screen X as the screen usually has a ratio of 1.3 (4:3).

To solve this problem use the utility
GetScreenFrames in the **Tools Panel** of the
TIDE\ NG_Center 1.5.7 program.
It is necessary to load a background image.

With this utility using the mouse select a rectangular region and get the four values **XPos, YPos, SizeX and SizeY** in micro units.

NEW SCRIPT COMMANDS

IMPORT_FILE ImportFile=

Syntax: ImportFile= IdImport, PathFile, FileType **(FTYPE_)** , ImportType (**IMPORT_**)

Scope: To use in [Options] section

Use the Import File command to insert in the **script.txt** file the Binary image of any file.

There are two advantages to using Import File:

To have a fast starting of some files in the game, like sounds. In this case use a memory import to force the **TRNG** engine to load the file into the RAM.

To use the **script.txt** like a container for some special files that would be copied in some folder in the target trle folder.

For example: To have a "help" subfolder in the trle folder with files like "start.htm" and some images use the import file in temporary mode to have the installation at the start of the **tomb4** program.

Arguments:

IdImport
The progressive number is used to locate the Import Slot.
The location is important when some new flip-effect is used to import the file.

For example: To import a file like **wagner.mp3** in the memory mode in order to be able to play the audio file using the flip-effect
Sound (CD) Play <&>imported file ...
and the "imported file" select the IdImport number.

PathFile
The path to import the file.
It is **not possible** to use an absolute path like **c:\.....\audio\003.wav**.
The path has to be relative using the current folder as the ROOT.

For example: To import the wav file **003.wav** located in the audio folder type a path like this: **audio\003.wav**

To import the load.bmp file into the trle folder
type a path: **load.bmp**

IMPORT_FILE **ImportFile=**

Remark: To import the file in the temporary mode (IMPORT_TEMPORARY value) the file will be exported (extracted) at the first start of the **tomb4.exe** and written in some relative path typed in the **script.txt**.

This is important when some of the files are in subfolders with non-standard names.

For example: For a subfolder **Mp3** inside the trle folder import a file from this subfolder in this way: ImportFile=1, mp3\bach.mp3, FTYPE_SOUND, IMPORT_TEMPORARY

The above command will work in the importing phase but it gives an error when the **TRNG** engine exports the file because when it is extracted it creates the subfolder **Mp3** and then copies the file with the name **batch.mp3**.

The problem is that the **TRNG** engine is not able to read audio tracks from folders different from the audio subfolder.

To avoid this problem there are two choices:

Always put the sound files to import in the audio folder of trle together with the other common adpmc wav files. **or...**

Import the file using a memory import. When a file is imported by the **IMPORT_MEMORY mode**, the folder name is not important as the file will never be exported.

The files imported in memory will be loaded in their Binary image and will be played directly from memory. In this case the path name will only be used in the importing phase and the name will be ignored when the game is played on the target computer.

FileType (FTYPE_)
Type a **FTYPE_** value to inform the **TRNG** engine about the nature of the file.

See the **FTYPE_**constants

ImportType (IMPORT_)
Choose an import mode: **IMPORT_MEMORY** or **IMPORT_TEMPORARY**

IMPORT_MEMORY
In this case the file is loaded directly into the memory at the start of the game.
The advantage is fast start for the file avoiding the delay to access the hard disk.
Reading the file from disc the game will be stopped for a short period.
Reading from memory (pre-load and start from the **script.txt** file) the audio track will start with no slow-down.

IMPORT_FILE **ImportFile=**

IMPORT_TEMPORARY

In the temporary mode the file will be stored in the **script.dat** file but it will be extracted from the disc at the start of the game.

This means it is used in the game in the traditional way: reading it from disc.

The advantage to using a file imported in Temporary Mode
(rather than enclose the specific separated file) is that it avoids confusing the program Level Manager during the installation phase.

Files like **.ogg** will be ignored by the Level Manager and also by many beginners in the Level Editor world.

mp3 files are handled by the Level Manager with the result they will be converted into **.wav** format.

To play a sound file directly in **.mp3** format avoiding the automatic conversion to wav by **TRLM**,

import the **mp3** file so the Level Manager will not be able to see it in the installation phase, but the **TRNG** engine will export it at the start of the game.

NEW SCRIPT COMMANDS

ITEM_GROUP **ItemGroup=**

Syntax: ItemGroup=IdGroup, FirstIndexItem, Other indices for the items

Scope: To Use in the [Level] section

Item Group permits a storage of a group of moveable indices.
These indices are from the **Tomb Editor** when an object is clicked to see its index shown as the first number in a yellow box.

Once a list of moveable indices is set in an Item Group operations can be performed on all of the objects referenced to by the Item Group with a single trigger.
Flip effects starting with the description: "ItemGroup. ....".

The advantage of working with an Item Group are as follows:

Multiple items can be used like a single moveable.

For example moving, enabling, rotating or hiding all of the items at the same time with a single trigger.

To perform a particular operation activation sequence with doors or enemies.

See the specific flip-effect triggers starting with the "Item Group. " descriptive text.

Field description:

IdGroup
This is the Id used to locate the Item Group command script
when it is required to perform an operation activated by a trigger.

Start with a "1" for the first Item Group in the [Level] and increase the Ids: 2, 3, 4 etc.

FirstIndexItem, Other indices for items
After the Id field type one or more indices for the moveables as read in the **Tomb Editor**.

Type a maximum of 83 indices for each Item Group.

NEW SCRIPT COMMANDS

KEY_PAD KeyPad=

Syntax: KeyPad=AtStartAnimation, FrameStartPopUp, AtEndAnimation, ClickSound

Scope: To use in the [Level] section

The Key Pad command is optional and it is used to change the default animations and sounds for the Key pad **SWITCH_TYPE1** found in the ng.wad

Only use this command to modify the animations or sounds to enable when Lara activates the Key pad switch in the game.

Parameters:

AtStartAnimation
The number of the Lara animation to show when she engages with the Action command for the keypad.

For no animation at the start type **IGNORE** in this field.

The default value is 197.

FrameStartPopUp: Only used if a valid value is set in the **AtStartAnimation**. In this case the keypad image is only shown when the start animation reaches this frame number.

For example: For 20 in this field the keypad will be shown 20 frames after the start animation is started.

Remarks: If No value is set for the **AtStartAnimation** this field will be ignored.
If a valid value is set for the **AtStartAnimation** field,
a valid value must be set in the **FrameStartPopUp** field.
It cannot be set to **IGNORE**.

AtEndAnimation
The number of the animation to show when the keypad is closed.

Remarks: If a final animation is not required type **IGNORE** in the field.
The **AtEndAnimation** is only shown if the player goes off from the keypad with the Enter/[*] key.

If an EXIT is made using the **Escape** key no final animation will be performed.

KEY_PAD KeyPad=

ClickSound
The number of the sound effect to play when the player hits the key on the keypad.

Remarks: If a sound is not required type **IGNORE** in this field.
The list of sound effects can be found in the **Catalogue folder.**

NEW SCRIPT COMMANDS

LARA_START_POS LaraStartPos=

Syntax: LaraStartPos= RoomOfLsp, OcbOfLsp

scope: To use in the [Level] section

This command forces the start position of Lara into the same position of the **LARA_START_POS** item for a given room or OCB.

Note: The main target of this command is to have two or more [Level] sections in the script, pointing to the same tr4 file.

In this situation set for each [level] section set a different start point for Lara, giving the player the feeling that they were different levels.

RoomOfLsp
The room index where the **LARA_START_POS** item is placed where Lara begins the level.
Type in the **OcbOfLsp** field a OCB value that is the only one in the level for the **LARA_START_POS** item.

To omit this field type **IGNORE**.

OcbOfLsp
The OCB value of the **LARA_START_POS** item where Lara begins the level.
There must be a valid Room number in the previous field and in that room there is only one **LARA_START_POS** item.

To omit this value type **IGNORE**.

NEW SCRIPT COMMANDS

LEVEL_FAR_VIEW LevelFarView=

Syntax: LevelFarView=NumberOfBlocks

Scope: To use in the [Level] section

NumberOfBlocks
The number of sectors shown in the game view.
The default value in the old tomb4 was 20 sectors.
This means that a mesh more than 20 sectors away from Lara becomes Black.

Valid range: Minimum = 1 Maximum = 127

The Number of Blocks set with this command should be less or equal to the value set in the [Options] section with the WorldFarView= command, otherwise it will be ignored.

Insert in the [Options] section a WorldFarView= command with a large number and then for the different levels set a smaller value with the LevelFarView= in accordance with the speed of the game.

Remember that not only the Non-resistance affects the speed of the game but also the number of meshes, transparent textures and moveables present in the level.

For this reason you could be forced to reduce the **LevelFarView** for levels with wide spaces and many moveables/meshes/transparent textures to enhance the speed.

In other levels use a larger value for the **LevelFarView** with no problem because the scene is simpler to draw for the **TRNG** game engine.

NEW SCRIPT COMMANDS

LOG_ITEM **LogItem=**

Syntax: LogItem=FlagsLogItem (**FLI_**), IndexOfItem

Scope: To use in the [Options] section

This command only works when the Diagnostic is ENABLED with the command:
Diagnostic=ENABLED in the [Options] section.

The Log Item permits to have on screen the same information as seen for Lara,
but in this case it is for another moveable.

The main target for this command is to get the information for a **TestPosition** command.

FlagsLogItem (FLI_)
Type one or more **FLI_** flags in this field.
If No flags are set type **IGNORE** in this field.

See the **FLI_** flags

IndexOfItem
Type the index of the moveable to monitor.
Find this index in the **Tomb Editor map** by performing a left mouse click on the item.

NEW SCRIPT COMMANDS

MIRROR_EFFECT MirrorEffect=

Syntax: MirrorEffect= InFrontRoom, HiddenRoom, MirrorType (**MIR_**), Animating array

Scope: To use in the [Level] section

The Mirror Effect replaces the old Mirror= command.
The old Mirror script command is still working.

The New Mirror type CANNOT be used with the Old Mirror command.

Arguments:

InFrontRoom
The room number in front of the Mirror.
This is a real room where Lara will be able to enter and move about.

HiddenRoom
The room number placed behind the Mirror where Lara cannot enter.

MirrorType
Specify a **MIR_**constant to set the Mirror type.

See the **MIR_**constants

Currently the following values can be used:

MIR_WEST_WALL
West wall is the setting used for the Old Mirror.
West is the position of the mirror from Lara's position looking at the room in the **Tomb Editor.**

MIR_FLOOR
The mirror is on the floor of the **InFrontRoom**.

MIR_CEILING
The Mirror is on the ceiling of the **InFrontRoom**.
To use a ceiling mirror it is advisable to use a low ceiling room,
otherwise Lara will not be able to look at the reflected Lara on the ceiling.

MIR_INVERSE_WEST
Inverse west is a horizontal Mirror on the west side of the **InFrontRoom**.
It is like the **MIR_WEST_WALL** but in the inverse mirror Lara and the other objects
will be inverted like in the Old Tomb Raider 1.

MIRROR_EFFECT MirrorEffect=

Animating array
From this field set one or more indices of animating present in the **InFrontRoom**.

Using this array the **TRNG** engine will place all of the animating in the correct position and orientation in the hidden room to simulate their mirror image in accordance with the mirror type.

A couple of animating objects have to be placed:

The main animating is placed in the required position in the **InFrontRoom**
and The clone of this animating is placed in the hidden room.

It is not necessary to place the clone animating in the correct position but it is useful to place it in the same vertical or horizontal line according to the mirror type.

If a **Vertical Mirror** is used place the main animation in the **InFrontRoom** with the correct orientation. Then place another animating of the same type in the hidden room.

In this situation it is important to carefully place the clone animating in the same vertical line. (i.e. same 2d visual sector).

The height is not as important for the clone animating or its orientation because these settings will be set by the **TRNG** engine at run-time.

For a **Horizontal Mirror** (like **MIR_WEST_WALL**)

Place the clone animating in the same sector row as the main animating.

See the help file for more information about the correct position for clone animating.

Remark: Type indices for other moveables in the Animating array, in this circumstance the moveable will dynamically update. To work the moveables like ENEMY place them in a hidden room and the trigger to enable them will be in the **InFrontRoom** where it is also the trigger to enable the enemy in the front room.

The room numbers for the **InFrontRoom** and the **HiddenRoom** field can be found in the room list of the **Tomb Editor**.

Remember to choose the smaller number when the numbers are different

For example: If the text for the Hidden (InFrontRoom) is:

Mirror Hall (34:30) The real room number is "30" ,
type this value in the MirrorEffect command.

NEW SCRIPT COMMANDS

MULTI_ENVELOPE_CONDITION MultEnvCondition=

Syntax: MultEnvCondition=IdMultCondition, **ENV_** condition,
DistanceForEnv, Extra field,
array of tripled of {**ENV_** Condition, DistanceForEnv, Extra field}

Scope: To use in the [Level] section

This command is used to store multiple **ENV_** Environment conditions for the Animation script command.

To set two or more **ENV_** conditions for the Animation command create a MultEnvCondition command with all of the **ENV_** conditions and then set the Id of the current MultEnvCondition in the field **DistanceForEnv** for the Animation command.

In the field **EnvCondition** of the Animation command type the value of the **ENV_MULT_CONDITION** to inform the **TRNG** engine that the real conditions are stored in the MultEnvCondition command with an Id = **DistanceForEnv** in the Animation command.

IdMultCondition
Type a number to identify the command.
Use the **DistanceForEnv** field for the Animation command to link it with the Animation command.

ENV_ condition
This field works in the same way as the **EnvCondition** field in the Animation command.
Type an **ENV_** condition value + (optionally) some **ENV_POS_** flags.

See the description of the **EnvCondition** field in the Animation command Section for more information.

DistanceForEnv
This works in the same way as the **DistanceForEnv** field in the Animation command
Type a value in this field to set the distance about the current (previous) **ENV_** condition.

See the description of **DistanceForEnv** field in the Animation command for more information.

Extra
This has the same use as the Extra Slot in the Animation command.
If an **ENV_** condition is used requiring a value in the **ExtraSlot** field type this value in the Extra field.

MULTI_ENVELOPE_CONDITION MultEnvCondition=

Array of Triples of {ENV_ condition**, DistanceForEnv, Extra} fields**
Type multiple triples of fields in the **EnvCondition**, **DistanceForEnv**, **Extra**
up to a maximum of 125 couples.

Examples:

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 a depth (distance ENV) = $400 (4 clicks)

as a first step create this MultEnvCondition command:

MultEnvCondition= 1, ENV_CEILING_HEIGHT, $300, IGNORE,
ENV_HOLE_FLOOR_AT_LEFT, $400, IGNORE

and then type in the Animation command the reference for the Id of the MultEnvCondition (1)

and the ENV_ condition: ENV_MULT_CONDITION

For example: Animation=447, KEY1_LEFT, IGNORE,IGNORE,ENV_MULT_CONDITION,
1,IGNORE,-445, -448

NEW SCRIPT COMMANDS

NEW_SOUND_ENGINE NewSoundEngine=

Syntax: NewSoundEngine=ENABLED/DISABLED

Scope: To use in the [Option] section

The new sound engine should be the **BASS 2.4 sound library** created by Un4seen Developments Ltd.

This new engine is based on Bass.dll and permits two channels for the CD audio sound (background and foreground or channel1 and channel2) at the same time.

Other features supported are:

Play a sound file different from **.wav** like **.mp3** or **.ogg** files
Fade out effect to close sweetly the previous sound before starting another CD track.
Dynamic change of frequency or volume in the game using new flip-effect triggers.

ENABLED/DISABLED field **The New Sound engine is ENABLED by default.**

Only use this command to DISABLE the New Sound Engine

command: NewSoundEngine=DISABLED

Remarks: It is not necessary to change the sound format when the new sound engine is used. The **bass.dll** is able to support the common ADPCM wav files found in the audio folder.

It is not advisable to add the bass.dll library to the level because the **TRNG** engine uses a build-in the **bass.dll** library.
Another bass.dll library could create conflicts.

The **bass.dll** used by the **TRNG** engine is **nextgeneration.dll** and the version is 2.4.0.1

It is better to use this version.
The internal **bass.dll** library will be extracted at-fly when the **TRNG** engine is started.

NEW SCRIPT COMMANDS

ORGANIZER **Organizer=**

Syntax: Organizer=IdOrganizer, Flags Organizer (**FO_**), Parameter, FirstTime,
PerformFirstIdTriggerGroup, SecondTime,
PerformSecondIdTriggerGroup,
{Time and TriggerGroup Array }

Scope: To use in the [Level] section

The Organizer performs a list of triggers, organized in Trigger Group script commands, at a specific time

For example: A list of Trigger Group can be created and organized with the Organizer command to open a door and after 4 seconds activate a BADDY then after another 22 seconds enable a flip-map etc.

Parameters:

IdOrganizer
Specify the Id of the current organizer.
This number will be used to locate the Organizer command with the flip-effects to enable or disable the Organizer.
Assign 1 to the first Organizer in the level section and 2 for the second Organizer, etc.

Flags Organizer (FO_)
Add one or more **FO_** constants to control the behaviour of the Organizer command.

See the **FO_**constants

Parameter
Currently unused.
It could be used in future versions.
Type **IGNORE** in this field.

Couples of Time + IdTriggerGroup
After the Parameter field store one or more couples of fields: Time to wait before starting.

Trigger Group
You can imagine these as Appointment information.
At this hour remember to do this, at other hour do this etc.
In this case the seconds will be used.

ORGANIZER **Organizer=**

For example: Create this Organizer command: Organizer=1, IGNORE, IGNORE, 6, 1, 5, 2, 4, 3

Read the above data in following way:

After 6 seconds from enabling of the Organizer perform the Trigger Group 1
After another 5 seconds perform the Trigger Group 2
After another 4 seconds perform the Trigger Group 3

We can understand this data using a table:

<PRE>

Time	**Trigger Group**		
6	Trigger Group	=	1
5	Trigger Group	=	2
4	Trigger Group	=	3

</PRE>

Remark: In the time field input a number up to 65535, i.e. about 18 hours.

NEW SCRIPT COMMANDS

PARAMETERS **Parameters=**

Syntax: Parameters= Type of parameters (**PARAM_**), IdParameterList, parameter array

Scope: To use in the [Level] section

The Parameters command is a general purpose command for parameters used by some triggers.

Since the trigger window is able to display only one or two parameters,
when a trigger requires many parameters it could require a
Parameters= command in the **script.txt** file to get all of the needed parameters for its operations.

Type of parameters (PARAM_)
In this field type a **PARAM_** constant to describe what the trigger is.

See the **PARAM_** constants

IdParameterList
This is a progressive number to identify the Parameters= command script in the trigger window of the **Tomb Editor**.

Type 1 for the first Parameter command, 2 for second etc.

Use the same **IdParameterList** when two Parameters are used in the command with
a different **PARAM_** constant.

It is necessary to set different Id's only for the Parameters with the same **PARAM_** constant.

Parameter array
From the third parameter are the parameters used by the Trigger.
The number and meaning of these parameters change according to the **PARAM_** type.

See the **PARAM_** constants

NEW SCRIPT COMMANDS

PLUGIN **Plugin=**

Syntax: Plugin= PluginId, PluginName, MainPluginSettings (**MPS_**),
DisableFeatureArray

Scope: To use in [Options] section.

You have to use a Plugin command for each plugin you mean to use.
This command has some differences with respect to other commands.

You cannot type it into an include file but only in the [Options] section.

You cannot use a variable defined with the **#define** directive for its **PluginId** field.

Read carefully the description of the **PluginId** field because it is the most important of the Id commands.

PluginId
The Id of the Plugin is very important for the exported (**from Tomb Editor**) triggers that are used in the Trigger Group commands.

When **Tomb Editor** exports a trigger (for the script) and that trigger uses a Plugin,
in the three numbers exported (mainly only in the first number) there is the hidden Id of the Plugin.

The problem is that the Id is only a number but to link this Id to an
effective **plugin_Name.dll** library it is necessary to do some more work.

In the **Tomb Editor** program just before exporting the trigger it will read the **script.txt** to discover the Id of the plugin (the field we are describing now**)** to set in the exported trigger the correct Id corresponding to the Plugin name that owns that trigger.

PLUGIN **Plugin=**

A complicated matter...

Remember some rules:

Always type in a plane way the Id of the plugin in the **Plugin=** command,
This is because the ability of the **Tomb Editor** program to parse the script is not as advanced as that of the **NG_Center compiler.**

The **Tomb Editor** is not able to parse **#define** values or texts in **#include** files.

Before exporting triggers from the **Tomb Editor**, take care to have already saved on disc the **script.txt** otherwise the **Tomb Editor** will read an old **script.txt** from disc.

Do not change the Ids of Plugin commands.
Never do that, only add a new **Plugin=** command for a Plugin not yet loaded.
The value inserted in previous Trigger Groups could not work as the value remains from an old plugin Id and it will not be updated.

When Trigger Group commands come from a friend and they used Triggers from a Plugin, you have to know what the plugin was and its **pluginId** in the script of the friend.

Then use the **#define @plugin...** directive to give the information to the **NG_Center compiler** in the script.

See the description of the **#define** command, in the section about **@plugin_name** and the **#define** "clear" attribute.

Every time a trigger is exported from the **Tomb Editor** handled by some Plugin, the **Tomb Editor** in its exporting report will also add information about the **#define @plugin** directive to use to enclose the Trigger Group that will host the trigger.

That directive is not necessary when only you are using the trigger in the script.
It will become useful when you give someone that Trigger Group.
In this case the receiver will need information about the Id of the Plugin stored in the triggers.
It is better to keep the information about the **#define @plugin** directive and any comments for that Trigger Group.

It is not necessary for your script because the **Tomb Editor** will place in the trigger the same Plugin Id that it read from the **script.txt** file.

In this case it will work omitting the **#define @plugin** directive.

Read information about the **#define @plugin** directive in the **NEW SCRIPT COMMANDS.**

PLUGIN **Plugin=**

PluginName
In this field type the name of a given Plugin.
Since this is a text, type the same text in the strings section of the **Tomb IDE Scripter.**
The name should not contain any extension.

The extensions are **.dll**, **.script** and others that some plugin files could have.

For instance for the plugin **Plugin_Marble.dll** type the name : **Plugin_Marble**

MainPluginSettings (MPS_)
In this field type **MPS_** flags to customize some basic features of the given Plugin.

The description of **MPS_** values

Only use the **MPS_** flags for that plugin and not for any others.

Each author of a plugin will set his **MPS_** flags**.**

If there is a plugin named **Plugin_Alfa**,
in the **Plugin=** script command only type the **MPS_** flags handled by the **Plugin_Alfa**.

To quickly locate these constants go to the **[Plugin]** panel of the **Tomb IDE** and choose from the combo box the "**Plugin_Alfa**" name (in **our example**) and then only see the constants of that Plugin.

The only exception to this rule (about the owner) is for the **MPS_DISABLE** value,
that is from **TRNG** and disable that Plugin to verify if some problem was affected by it.

DisableFeatureArray
This array and this means that from this field type one, more (or none) values in the same format.

The format for each item of this array is a sum between a **CODE_** constant + one operand value.

The generic use of this fields is to disable some changes (about standard trng features)
that did not please you.

See the description of the **CODE_** constants

NEW SCRIPT COMMANDS

PRESERVE_INVENTORY **PreserveInventory=**

Syntax: PreserveInventory= ENABLED

Scope: To use in the [Level] section

The Preserve Inventory command can be used when a ResetHUB= command is used in the same [Level] section.

With the **ResetHUB** command the level number is set that will be started from the current level with a FINISH trigger.

If Lara's meshes need to be reloaded or there is a change of vehicle use the ResetHUB.

The problem with the **ResetHUB** is that the Pickup items will be cleared from the ResetHUB command.

If a **ResetHUB** is used and it is required to preserve the Pickup items from previous levels add it in the [Level] section of the command:

PreserveInventory=ENABLED

NEW SCRIPT COMMANDS

RAIN **Rain=**

Syntax: Rain= RAIN_ constant

Scope: To use in [Level] section

A Rain Room in the current [Level] section is only allowed with a command Rain= in the **script.txt** file,

otherwise the setting for the Rain in the **Tomb Editor** will be ignored.

Argument:

Choose one of following constants:

RAIN_DISABLED
This is the same as the NO Rain= command in the **script.txt** file

RAIN_SINGLE_ROOMS
This setting will show for specific rooms signed as "Rain" in the **Tomb Editor** with outside status.

RAIN_ALL_OUTSIDE
For Rain in all outside rooms without setting the "Rain" button for each room in the **Tomb Editor** use this command in the **script.txt** file.

To enable the **RAIN_ALL_OUTSIDE** setting ENABLE the "Rain" button in the First Room that will be visited by Lara and set the Water Intensity field at the right of the multi-state button water/rain/snow intensity (from 1 to 4).

The intensity for all of the level will then be set.

If rain rooms are not required in the current level use the command

Rain=RAIN_DISABLED

or omit the

Rain= command in the **script.txt** file to inform the **TRNG** engine.

NEW SCRIPT COMMANDS

SAVE_GAME_PANEL SavegamePanel=

Syntax: SavegamePanel=SavegamePanelFlags (SPF_), BackGroundImageId,
NumberOfSave, NumberOfVisibleSave,
SavegamePanelLayout (SPL_),
InfoFormatString, IdListWindowsFont, IdInfoWindowsFont,
InTitleWindowsFont

Scope: To use in [Level] section

The Save Game Panel command allows the customization of a new Save Game Panel to load or save the game

Type this command in a [Level] section and the old load/save screen will be replaced by this new save game panel.

The advantage of this new Save Game Panel is to be able to view the inner image of each save game to give a better choice of the correct save game to load.

The panel helps the player avoid overwriting important save games.

Use this new Save Game Panel together with the customize to enable the saving of the image inside the screen shot.

Add the following to the **script.txt** file:

Customize=CUST_INNER_SCREENSHOT, QSF_SIZE_320x240+QSF_TRUE_COLOR

The QSF_ flags could be different .
Add a Customize=CUST_INNER_SCREENSHOT command in the level section to enable the creation of a screen shot image for each save game of the adventure.

The number of save games can be customized (in the old Raider panel this was 15 save games).
The information to show when a save game is selected can be customized.

SavegamePanelFlags (SPF_)
Add two or more SPF_ flags to customize a feature of the Save Game Panel.
If No flags are required type IGNORE in this field.

See the SPF_flags

BackGroundImageId
Supply a background image for the Save Game Panel.
Type the number of the image located in the PIX subfolder.
For example for a background image using the file **Image8.bmp** type 8 in the field.

SAVE_GAME_PANEL SavegamePanel=

NumberOfSave
Type the quantity of save games that the Save Game Panel will handle.
In the **Old Tomb4** this was 15 (from save game.0 to save game.14).
The maximum number that can be set is 100.
A reasonable value could be 30.
To use 15 save games type **IGNORE** in this field.

NumberOfVisibleSave
The number of visible save games is different from the previous field **NumberOfSave**.
When a large number of save games is set in the **NumberOfSave** field it is not possible to show all of the list on the screen.

To solve this problem the **TRNG** engine is able to show a few save games on the screen and then allow the scrolling of the list to show another "page" of save games.

The **NumberOfVisibleSave** field is the number of save games displayed for each visible page on the screen.

For example: The **Old Save Game** Panel showed 15 save games.
To handle 50 save games set the **NumberOfVisibleSave** games to 15 and then the **TRNG** engine will show the first 15 save games.

When the player hits the DOWN key the **TRNG** engine will show the next group of 15 save games and go on.

Remark: The **TRNG** engine will page if a specific **SPF_** flag is set.
Otherwise it will slowly scroll through the list.

Note: If **IGNORE** is typed in this field the **TRNG** engine will consider that all of the **NumberOfSave** will be shown on the screen at the same time,
i.e. the **NumberOfVisibleSave** and **NumberOfSave** fields are the same value.

The general rule to follow is Set the fonts, image and layout for the Save Game Panel and test it in the game to see how it looks on the screen.
Then set the number in the **NumberOfVisibleSave** field.

WARNING: If a correct number is not set in the **NumberOfVisibleSave** field there will be a big mess because the **TRNG** engine will not verify if all of the required save games will fit on the screen.

SAVE_GAME_PANEL SavegamePanel=

SavegamePanelLayout (SPL_)
Type a single **SPL_** value to set the position of the different frames on the screen.
Choose where to place the image on the screen (left, right or centre) and the list of save games.

See the **SPL_** constants

There is also another (optional) frame: The information frame.
The information frame shows data about the currently selected save games.

For example: Show the number of secrets, or the weapons available in that selected save game.

See the description of the next field about the choice.

InfoFormatString
Use a Savegame Panel Layout with the Information frame type in the **InfoFormatString** field
The string is used to format the information to show the selected save game.
The information frame is a zone of the screen where it will show information about the save game currently selected.

Use the trick of external NG strings to store the format text, i.e. type the text in a text file, save it in the Script folder and then link this text in ExtraNG strings using "@" + "name.txt".

For example: Type the format text in a file named **Info_savegames.txt** and add a new ExtraNG string and type the text **@Info_savegames.txt** into it.

In this text form type what is required to display on screen.

Remember to use a special place-folder to signal a specific value in the save game.
All place folders are enclosed in round parenthesis "( )" and have fixed names that the **TRNG** engine will be able to recognize and replace with real values read from the save game.

For example: Type text information in the following format:

Found Secrets	(SECRETS)
Large Medipacks	(L-PACKS)
Small Medipacks	(S-PACKS)

Then in the game when the player selects a save game from list on the screen it will show the values in that save game. So for example it could read:

Found Secrets	5
Large Medipacks	3
Small Medipacks	12

SAVE_GAME_PANEL **SavegamePanel=**

Remark: To have a good alignment it is advisable to select an information text windows font with a fixed width, like "courier" or "new courier".

Proportional fonts (like "Arial") could give a bad alignment:

Found Secrets 5
Large Medipacks 3
Small Medipacks 12

The full list of place-folders is as follows:

LEVEL-NAME	The name of the level in the save game
SAVE-NUMBER	The progressive inner number of the save game
SECRETS	The found secrets
L-PACKS	The number of Large Medipacks
S-PACKS	The number of Small Medipacks
FLARES	The number of flares
WEAPONS	Show a list of the weapons in the save game. See (1) note in following remark section.
GAME-TIME	The game time. See (2) note in following remark section
DATE-TIME	The date when the current save game was created or the last time it was changed. The format is DD/MM/YYYY (HH:MM:SS)
KM-DISTANCE	Number of Km of distance
METERS-DISTANCE	Number of remaining metres of distance (See (3) note)

1. Since the weapon list could be long, it is better to let a whole line display this value using a format text like this:

   ----- **Example of information format text** ----
   Available Weapons:
   (WEAPONS)
   ---- **end information format text sample** -----

   The list will have a row of information frame

2. The game-time will be shown in the save game list as in the default Save Game Panel. The game time can be removed from the save game list with the **SPF_NO_TIME_IN_LIST** flag and then add the value in the specific description of the selected save game with a place folder (GAME_TIME).
   The advantage of this method is to be able to display the save game list in a short frame.

3. The reason to have two different values for KM and metres is to allow the words "km" and "metres" in the language or a different language, one for each language file.

SAVE_GAME_PANEL **SavegamePanel=**

For example: Type the information format text:

Distanza: (KM-DISTANCE) chilometrie (METERS-DISTANCE) metri.

The above example shows the description of distance in Italian,
using the words "chilometri" and "metri" instead of "kilometres" and "metres"

In the game it would read: Distanza: 2 chilometrie 43 metri

IdListWindowsFont
Type the Id of the WindowsFont=IdNumber command where the style of text used is set for the save game list.

It is important to place the WindowsFont command before the SavegamePanel command that uses it.

IdInfoWindowsFont
Type the Id of the WindowsFont=IdNumber command to set the style of text used for the information shown for the selected save game.
Since there could be a lot of information use a small font.

Remark: It is important to place the WindowsFont command first in the SavegamePanel command that uses it in the **script.txt** file.

InTitleWindowsFont
Type the Id of the WindowsFont=IdNumber command to set the style of the text used to display the Title of the Panel,

i.e. the text "Save Game" or "Load Game" in accordance with the current language.

Remark: It is important to place the WindowsFont command before the Save game Panel command that uses it in the **script.txt** file.

NEW SCRIPT COMMANDS

SETTINGS **Settings=**

Syntax: Settings= **SET_** constants

Scope: To use in the [Options] section

This command will be placed in the [Options] section to set some Global settings.

Arguments:

SET_ constants
Type one or more **SET_** constants separated by plus (+) sign.

See the **SET_**constants

NEW SCRIPT COMMANDS

SHOW_LARA_IN_TITLE **ShowLaraInTitle=**

Syntax: ShowLaraInTitle=ENABLED/DISABLED

Scope: To use in the [Options] section

By default Lara is not visible in the Title level but using this command:

ShowLaraInTitle=ENABLED in [Options] section.

Lara will be drawn.

NEW SCRIPT COMMANDS

SNOW **Snow=**

Syntax: Snow= **SNOW_** constant

Scope: To use in [the Level] section

The snow is only shown in the current level if the current [Level] section has the command

Snow=SNOW_SINGLE_ROOM
or Snow=SNOW_ALL_OUTSIDE

SNOW_DISABLED
This setting is the same as omitting the Snow= command.
Rooms labelled as "Snow" room in the **Tomb Editor** will be ignored,
i.e. no snow effect will be shown.

SNOW_SINGLE_ROOM
Only the room with a "Snow" label in **Tomb Editor** and with outside status will have the snow in the game.

Each room will have its own snow intensity read from the **WaterIntensity** field.

SNOW_ALL_OUTSIDE
The snow is shown in all outside rooms, ignoring the multi-state button.

A room without "snow" will have snow if it is not a Water room and has the outside status.

Remark: If the **SNOW_ALL_OUTSIDE** setting is used to set the intensity for snow in all of the level set a single room with the "Snow" attribute in the **Tomb Editor** and set the value for the snow intensity.

That value will be used for the whole level.
For No snow in the current level use the command
Snow=SNOW_DISABLED.

NEW SCRIPT COMMANDS

SOUND_SETTINGS SoundSettings=

Syntax: SoundSettings=Sound Quality (**SQ_**), MusicVolume, SoundEffectVolume

Scope: To use in the [Options] section

Sound Settings allows setting the music quality, Volume and the sound effects (SFX) volume.

Sound Quality field **Choose three different Qualities for music**:

SQ_LOW_QUALITY	(11025 Hz)
SQ_MEDIUM_QUALITY	(22050 Hz)
SQ_HIGH_QUALITY	(44100 Hz)

Type **SCRIPT_IGNORE** and the quality will not change and it will use the value currently set in the game.

MusicVolume
Type values between 0 (silent) and 100 (maximum volume).
Type **SCRIPT_IGNORE** in this field and the music volume will not change from the value currently set in the game.

SoundEffectVolume
Type values between 0 (silent) and 100 (maximum volume).
Type **SCRIPT_IGNORE** in this field and the SFX volume will not change from the value currently set in game.

NEW SCRIPT COMMANDS

STANDBY StandBy=

Syntax: StandBy= IdStandBy, Type StandBy (**TSB_**), WaitTime, Flags StandBy (**FSB_**), Text, NumTexts, AudioTrack, VAngle, RotateSpeed, Distance, IdTriggerGroupBegin, IdTriggerGroupEnd

Scope: To use in the [Level] section

The Stand By command enables an automatic Pause-mode in the game.
The StandBy will begin when the player does not press any key for a time that is set in the **WaitTime** field.

Choose what will happen in the game:

The main effect is a matrix effect where the camera turns around Lara.
Add text and a new audio track to underline the Stand By mode.

IdStandBy
The Stand By starts when the correct conditions are met.
It is added as a flip-effect to execute a Stand By mode.
With this flip-effect the settings of the Stand By command perform an in game camera effect.
To select the Stand By choose the corresponding **IdStandBy** value.

IdStandBy value = 1.

StandBy (TSB_)
Type a single **TSB_** value in this field to set the main effect for the Stand By.

See the **TSB_** constants

WaitTime
Set the time of NO KEY INPUT to activate the Stand By
Type in the number of seconds.

Flags StandBy (FSB_)
Set the **FSB_** flags to customize the main method set with the **TSB_** value
Type **IGNORE** in this field to omit any **FSB_** flag.

See the **FSB_** constants

Text
Type the text that will be printed on the screen when the Stand By is on.
For example show text like: **Hit a Key To Continue**.

STANDBY StandBy=

NumTexts
Set the number of texts to show during the Stand By mode.
Usually it will be "1" to show the single string typed in the previous "text" field.

Show different strings in sequence.
When a number greater than "1" is set, the game will show the string typed in the "Text" field and after 4 seconds the following string in the ExtraNG strings list.

For example: Type in the [NG Strings] section of the script the following strings:

23: Pause - Hit a Key
24: The longest Night
25: Created by **PAOLONE**

To show the sequence on screen, type in the **Text** field for the first string,
followed by the **NumTexts** field the number of strings to show ("3"):

StandBy= .... , Pause - Hit a Key, 3, ...

AudioTrack
For a new audio track to play during the StandBy mode type in the number of the CD track to play.
For no audio type **IGNORE** in this field.

VAngle
This field is set if the camera is looking at Lara from down or up.
Positive values place the camera BELOW Lara and therefore the camera will look up.
Negative values move the camera up to look at Lara.
With a zero value the camera will have the same height as Lara

The valid range of values is from -16384 to +16384 but it is better to avoid the 16384 value.
It is advisable to use zero or negative values for this field because large positive values make the camera go under the floor.

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

Note: 16384 = 90 degrees, 8192 = 45 degrees.

RotateSpeed
The speed of the camera turning around Lara has units like degrees.
A full revolution will be 65536 ($10000). The value typed in the field will be added to the current angle 30 times per second so compute the value to type to reach the target.

For example: For the camera to perform a quarter of a revolution (90 degrees) in one second 16384 (i.e. 90 degrees) / 30 (i.e. one second) = 546.

Setting too high a value for rotation speed will make it difficult to watch the StandBy.

STANDBY StandBy=

Distance
This is the distance between the camera and Lara.
One sector is 1024 units, one click = 256 units.
The reasonable range of values could be 256 / 4096.

Values greater than 2048 should be avoided as the camera could have problems turning around Lara and colliding with walls.

IdTriggerGroupBegin
The Stand By command supplies many features, camera effects, background audio and printing of texts.

To customize the Stand By further use this field to set a Trigger Group to perform when it starts in the Trigger Group.

For example: Call an Organizer where scrolling or sliding texts will be shown, or a Trigger that starts a Fly By sequence.

For no Trigger Group type **IGNORE** in this field.

IdTriggerGroupEnd
When a Trigger Group is set at the start of the Stand By mode it is possible to use another Trigger Group to remove the effect added by the first Trigger Group.

For example: For text displayed on the screen for an infinite time use the End Trigger Group to clear the screen.

NEW SCRIPT COMMANDS

STATIC_MIP StaticMIP=

Syntax: StaticMIP= MainStaticSlot, BStaticLimit, BStaticSlot, CStaticLimit, CstaticSlot

Scope: To use in [Level] or [Title] section

The target is to get a faster view and better frames per second (fps) in the game when there are wide scenes with many static items.

Using the **StaticMip** command and creating low quality copies for most used static items, to draw when they are far from the source view will give an improvement in speed.

This command requires the **TRNG** engine to show statics with different quality from the given Main Static, according to the current distance between the static and current source camera. Usually from Lara's position.

This is like the AnimatingMIP command.
In this case set two different limits for a total of three different static kinds to draw in the game.

MainStaticSlot
Type the slot number of the main static.
The static slots are in the **STATIC** list.

The **MainStatic** is the static object placed in the level map and it should have the highest quality.

BStaticLimit
Type the min distance, in sectors (1024 game units),
between the source view and the static to draw the **BStaticSlot** replacing the **MainStaticSlot**.
Apply when the distance is equal to or greater than the **BstaticLimit**.

Less than the **CstaticLimit** draw the static from the **BStaticSlot** instead of the **MainStaticSlot**.

Create a copy of the **MainStaticSlot** in the **BStaticSlot**, having the same size and look but using less meshes and less (or no) transparent textures.

Remember that transparent textures use a lot of CPU (or GPU) time.

BStaticSlot
This is the static slot for the **BStaticLimit** and is less than the **CStaticLimit**.
Try to understand that this change of static to draw does not affect the slot item set in the level or the static OCB or its light.

Only the mesh of the **BStaticSlot** will be used but all other features will be taken from the **MainStaticSlot**.

Type **IGNORE** in this field and the drawing of the **MainStaticSlot** is skipped when the distance is greater than the **BstaticLimit**.

STANDBY StandBy=

CStaticLimit
This field works like the **BStaticLimit** but in this case it will affect the drawing of the **CstaticSlot**.

The **CStaticLimit** should always be greater than the **BstaticLimit**.

Type **IGNORE** in this field means not to supply another raw copy of the **MainStaticSlot.**

When **IGNORE** is typed as the **CStaticLimit** the next **CStaticSlot** will be ignored.

CStaticSlot
The slot of the static typed in this field is only drawn when the distance is bigger or equal to the **CStaticLimit** distance.

The copy of the **MainStaticSlot** set in the **CStaticSlot** should be of a low quality, like a simple shadow and should have a great distance in the **CStaticLimit** field.

When **IGNORE** is typed in this field this means to skip the drawing of the **MainStaticSlot** when the distance is bigger or equal to the **CStaticLimit** field.

NEW SCRIPT COMMANDS

SWITCH Switch=

Syntax: Switch= SwitchId, VariablePlaceFolder, FlagsSwitch (**SWT_**),
TriggerGroupIndices

Scope: To use in the [Level] section.

Introduction
The Switch command permits a choice between a list of Trigger Group indices in accordance with the value in a variable.

For example: Input a number from the **Key Pad** and then perform different Trigger Groups according to the number chosen by Lara.
The same result can be obtained with a Trigger Group filled with a list of Condition Trigger and **ELSE TGROUP_** flags.

Using the switch command obtains the result in an easier and faster way,

SwitchId
This is the Id chosen in the flip-effect to perform a Switch command.
Type different progressive numbers for the Ids in the same [Level] section.

VariablePlaceFolder
Type the Place Folder to identify a specific **TRNG Variable**.

See the list of Place Folders in the VARIABLE PLACEFOLDERS

For example: To use the variable **Local Short Beta1**

Type the Place Folder: #0052.

Type **IGNORE** in this field and the Switch will use the variable **LastInputNumber**, which is the variable that receives the value typed with the Key Pad object.

FlagsSwitch (SWT_)
Type one or more **SWT_** constants to affect the working mode of the Switch command.
Type **IGNORE** in this field if the flag is not set.

See the **SWT_** constants

SWITCH **Switch=**

Trigger Group Indices
From this field type indices of Trigger Group commands.

The first Trigger Group will be performed when the variable is "1",
the second when it is "2", etc.

If **IGNORE** is typed in the field there will be NO Trigger Group for that number.

For example: If no Trigger Group is required when the variable is "2"

Type a list like this: ..., **4**, IGNORE, **7**, **8**

When the variable is "1" it will perform the **Trigger Group=4**.
When the variable is "2" nothing will be performed.
When the variable is "3" it will perform the **Trigger Group=7**
When the variable is "4" it will perform the **Trigger Group=8**

Up to 120 indices can be put in this array.

NEW SCRIPT COMMANDS

TEST_POSITION **TestPosition=**

Syntax: TestPosition= IdTestPosition, Flags (**TPOS_**), Slot Moveable, XDistanceMin, XDistanceMax, YDistanceMin, YDistanceMax, ZDistanceMin, ZDistanceMax, HOrientDiffMin, HOrientDiffMax, VOrientDiffMin, VOrientDiffMax, ROrientDiffMin, ROrientDiffMax

Scope: To use in the [Level] section

Use the **TestPosition** to verify if Lara is in the correct position with respect to other items (all moveables in the given Slot).

Type the **IdTestPosition** and the **Extra** field of

Animation=
or MultEnvConditon=

script commands to transform it into an effective condition.

The TestPosition= is a complicated command.
There is an example zip file to understand how to use TestPosition.

The zip file is called **AnimationTestPosition.zip** and it contains some images to understand the following explanations.

The zip file also contains a project + script to test the **ENV_** condition with the TestPosition command.

Descriptions of TestPosition fields:

IdTestPosition
Type an Identifier number to call this **TestPosition** command from the other commands like Animation= or MultEnvConditions= (type this Id in the Extra field).

Each **TestPosition** command input for the [Level] section should have a progressive Id number.

Flags (TPOS_)
Type one or more **TPOS_** constant values in this field to set some special features.
If it is not required type **IGNORE** in this field.

See the **TPOS_** constants

TEST_POSITION **TestPosition=**

Slot Moveable
Type the slot (the number or the name) of items to check.

Remark: If the Flags field of the **TPOS_TEST_ITEM_INDEX** has a value in the **SlotMoveable** field type the index of the item to check.

See the description of the **TPOS_TEST_ITEM_INDEX** for more information.

XDistanceMin
Type the minimum valid distance from the X origin of Lara and the X origin of the item to test.

XdistanceMax
Type the maximum valid distance from the X origin of Lara and the X origin of the item to test.

To understand what the X axis is,
look at the image **ExTestPosition_Axis.jpg** in the **AnimationTestPosition.zip** file.

The X axis and its orientation is in a Light Blue color.
The effective difference to compare from these two fields is given by:
X_Lara_origin – X_Item_Origin.

The range to type is always a negative value for **XDistanceMin** and a positive value for **XDistanceMax**

For example: XDistanceMin = -256
XDistanceMax = +256

This means a tolerance of one click (a sector is 1024 units, so a click is 256 units)
of difference between Lara and the item on X axis.

Example: Set 0 for the minimum and maximum X distance.
This means that Lara's nose is on the other item (for example a Ninja).

Example: For X difference range: The item to test is a coffee cup and the cup gives a difference on the X axis of +100 .

This means that the cup would be close to the left hand of Lara.

If the difference was -100 the cup would be near the right hand of Lara.

When the valid range for the X axis (or others) is computed keep in mind the concept of **Relative axis** used in the TestPosition comparison.

TEST_POSITION TestPosition=

RELATIVE AXIS description
The axes are relative to the current orientation of Lara.
Usually in the 3d world we think of fixed x, y, z coordinates.

When the difference for the X relative axis is computed the **TRNG** engine fixes the origin of the 3d world to the origin of Lara and it rotates the 3d world to have Lara looking in the same direction as the required axis.

If a Z difference of 256 units is set (Z axis).
See the ExTestPosition_Axis.jpg picture
and the condition is true,
it means that the item will be one click in front of Lara.

For example: In the 3d world Lara is to the left or right of an item on the screen.

The position of the origin will be changed before performing the comparison to have Lara at 0,0,0
and then look along the axis at the item.

Where Lara is looking will now become the new Z axis direction.
This method has been created by EIDOS programmers.

YDistanceMin
Type a value for the minimum distance on the Y axis.
The Y axis has a White color.

YDistanceMax
Type a value for the maximum distance on the Y axis.
The Y axis has a White color.

To understand what the Y axis is,
look at the image **ExTestPosition_Axis.jpg** in the **AnimationTestPosition.zip** file

The description for the X difference range are valid for the Y range.

In this case there is a further complication because while the X and Z origin of Lara are always the same at half of her body height,
the Y origin of Lara changes according to the current animation.

Use the Diagnostic to understand where the Y origin of Lara is set in the animation.
Then it will give the value to launch the custom animation.

Type in the [Options] section of the **script.txt** file the
command: Diagnostic= ENABLED

In the game the values for Lara will be displayed.

TEST_POSITION **TestPosition=**

Look at the "Cy=" value.
To understand this:
Build a sample project where the floor is at 0 clicks,
then extrude a sector to 4 clicks high and a hole in the floor of 4 clicks.

The basis of this test is to see what the Cy of Lara is when she is on the floor.
The Cy should be "0" if the floor of the room is 0 click.

When Lara moves up a sector 4 clicks high the Cy should be
Cy = -1024 because upward the Y axis is negative.

If Lara goes down in a hole 4 clicks deep the Cy should become +1024
because there are 4 clicks of difference up to the floor.

This experiment is simple because the results are known.
However it shows the method to discover the reference of the Y origin of Lara for all animations.

For example: If Lara hangs on the edge of an elevated 4 click block,
there is an animation that of hanging
and in this animation the Y origin is not Lara's feet but her knees.

For example: Create a project with the floor at 0 clicks and a pool room
to discover other Y origins for particular standard animations.

When Lara is floating on the water surface the Y origin of Lara
is the same as the Y coordinate of the water surface.
When Lara is floating up and down on the waves the Cy value remains
the same. In this case the Y origin is Lara's neck.

Use a **TestPosition** command to verify what point of Lara is used for the Y origin
of the animation when the custom animation is launched. Build the Y range on this reference.

Now perform an example using the stand-up position of Lara when the Y origin is on her feet.
If the item is a small ball and it is in front of Lara's head,
the difference will (about) + 600.

If the difference is negative,
for example - 1024 it means the ball is in a hole below Lara.

TEST_POSITION **TestPosition=**

For example: To put a ball in front of Lara's head

Set Y ranges: YDistanceMin = 550
YDistanceMax = 650

There is no negative value in the above values.
This is because Lara's height is about 3 clicks.

So 256*3 = 768 units (exactly 800 units in the 3d world).

For a ball in front of Lara's head,
the ball cannot be lower than Lara's neck which is about 512 units from the her feet (Y origin).

Set a minimum Y distance of 550,
if it was less it would mean the ball is closer to Lara's feet.
The ball should also be less than 768 so that it is not above her head.

Remember that the axes are relative so it is not important if the Y origin of Lara or the item is positive or negative.

The difference will work in the correct way.
When the Y difference (in the example of the ball) is 0 it means the ball is in front of her feet.

SIZE of Lara: The following sizes can be used for Lara:

Height in stand-up position	=	800
Width stand-up, from hand to hand	=	320
Depth of Lara by profile	=	130
Height in all fours position	=	320
Height in climb position	=	600

ZDistanceMin
Set the minimum distance on the Z axis.
The Z axis is the direction where Lara is looking.

ZDistanceMax
Set the maximum distance on the Z axis.
The Z axis is the direction where Lara is looking.

When the Z difference (Lara.Z - Item.Z) is negative this means that the item is in front of Lara.
If the difference is positive Lara has passed over the item and it is behind her.

TEST_POSITION **TestPosition=**

HorientDiffMin
Set the acceptable minimum Horizontal relative difference between Lara and the Item.
The horizontal orientation is given by Lara.HorizonalOrient – Item.HorizontalOrient.

HOrientDiffMax
Set the acceptable maximum Horizontal relative difference between Lara and the Item.
The horizontal orientation is given by Lara.HorizonalOrient – Item.HorizontalOrient.

To understand "Horizontal" facing look at the picture **ExTestPosition_H_Orienting.jpg** that is in the **AnimationTestPosition.zip** file.

The default Horizontal facing of items (Lara or others) is 0 and this is the position seen in the **Tomb Editor** when an item is placed in the level map.

To understand how the difference between the orientation changes
see the project in the AnimationTestPosition.zip file.

If the correct distance ranges are set but the horizontal orientation is wrong
Lara gets closer to the item but the item but it is not on the correct side.

For example: Create a cabinet and an animation where Lara forces the lock to open it.

It is necessary not only for Lara to be close to the cabinet,
but also to be at the front to open the lock.
If Lara has the back or the side of the cabinet in front of her,
the distance could be correct but the animation would make no sense.

Find the ideal difference between the Horizontal orientation and
then create a range where there is at least some tolerance.

For the cabinet the difference could be $8000 (32768) (face to face)
and the range should have a Minimum,
for example $7f00
the maximum is bigger by twice the tolerance
(+/- $100) i.e. $8100.

TEST_POSITION TestPosition=

VOrientDiffMin
Set the minimum difference for the Vertical Orientation.
See the image **ExTestPosition_vOrienting.jpg** to understand how the V Orientation works.

VOrientDiffMax
Set the maximum difference for the Vertical Orientation.
See the image **ExTestPosition_vOrienting.jpg** to understand how the V Orientation works.

This range is not so important because 99 times out of 100 both items,
Lara and the other item will have VOrientation = 0

Set a range Min= -200 and Max = +200

A rare case where Lara has a different VOrient is the sprint run,
when she swims underwater or diving.

See the sample project in AnimationTestPosition.zip file to verify this situation.

RorientDiffMin
Set the minimum difference for the Rotating Orientation between Lara and an item.

ROrientDiffMin and ROrientDiffMax
Set the maximum difference for the Rotating Orientation between Lara and an item.

The values seldom change.
They are 0,
so the difference between Lara and the item should be zero.

This value changes when Lara is sprinting and she is turning left or right.

Type a range -200 +200 but use 0 and 0 for most situations.

NEW SCRIPT COMMANDS

TEXT_FORMAT TextFormat=

Syntax: TextFormat=Color (**CL_**), FormatFlags (**FT_**) , BlinkTime, SizeCharacterMenu (**SC_**)

Scope: To use in the [Level] section or [Title] section

Arguments:

Color
CL_constant sets the default colour for the print string flip-effect.

FormatFlags
The **FT_**constant sets the default position of text,
the default size and further blink or stretch characters.

Note: Add to the FT values for the text position the two flags **FT_NARROW_CHARS** and **FT_BLINK_CHARS** and all the **FT_SIZE** flags using the '+' character.

Example: To set text in Gold at the Top aligned Central with Blinking characters.:

FormatText=CL_GOLD, FT_TOP_CENTER + FT_BLINK_CHARS,16

Remark: Remember this setting only works for text printed using the print text flip-effect. To change the size of the **TRNG** characters set a **SC_** constant in the **SizeCharacters** field.

BlinkTime
This is a Numeric value to signal the interval for the Blinking when the **FT_BLINK_CHARS** flag is set in the **FormatFlags** parameter.

For the Blink time only use powers of 2 values, like: 1, 2, 4, 8, 16, 32, 64, 128

TEXT_FORMAT **TextFormat=**

SizeCharacterMenu
Use the value **SC_ TYPE** to change the default size of the characters in the **TRNG** menu.

Remark: Values of **FT_** flags is the value set in the **SizeCharacter** field and will only effect the **TRNG** Default texts (Menu, Options, but not the Legend string).

This setting will have no effect on the flip-effect print text.
Only use this field in the [Title] section in this way:

TextFormat=IGNORE, IGNORE,IGNORE, SC_HALF_HEIGHT

To use the **TextFormat** in the [Level] section type **IGNORE** in this field, otherwise the text for the **TRNG** menu will change in an irregular way according to the last level played.

Currently ONLY ONE one of the following values in this field can be selected:

SC_NORMAL	Do not change anything. The characters will have default size
SC_HALF_WIDTH	Text will be reduced by half the width of the characters, while the height will be not changed
SC_HALF_HEIGHT	Text will be reduced by half the height of the characters, while the width will be not changed
SC_HALF_SIZE	This value reduce by half both the width and height of the characters
SC_DOUBLE_WIDTH	Set double width for the characters, untouched height
SC_DOUBLE_HEIGHT	Set double height for the characters, untouched width
SC_DOUBLE_SIZE	Set double width and double height for the characters.

TEXT_FORMAT **TextFormat=**

Default values for the Text Format command:

TextFormat= CL_WHITE, FT_BOTTOM_CENTER, 16, SC_NORMAL

If no Text Format command is set for the current level the default values will be used:

Remarks:

Change the settings for the text in the game with flip-effects.
It is better to use this script command to set default values.

Use some **FT_** constant to set the position of text to add to a specific string
some format characters to change the position of the text on the screen.

Example: **FT_TOP_LEFT** is used but the text overlaps into the Vitality Bar.
Start the text with a character couple: "\n" (n = newline).
A empty line will be printed and the visible text will start in a lower position.

Example: To move a LEFT aligned text to the right, start each line with a character couple: "\t" (t = tabulation).

\tHi world\n\tI am very Happy to be here

NEW SCRIPT COMMANDS

TEXTURE_SEQUENCE TextureSequence=

Syntax: TextureSequence=IdTexSeq, FramePerSec, **SEQ_** flags, Tex Indices array {...}

Scope: To use in the [Level] section

IdTexSeq
A number to identify the Texture Sequence when it is enabled using the flip-effect trigger.

Use "1" for the first Texture Sequence command,
"2" for the second Texture Sequence command, etc.

FramePerSec
Set the speed of the sequence.
The value is in frames per second.
The maximum value is 30, i.e. 30 frame for second, the smallest value is 1.

SEQ_ flags
Set two or more **SEQ_** flags.

Currently there are the following flags:

SEQ_LOOP
The sequence will be performed in a loop.
If this flag is set use a specific flip-effect to stop the sequence,
otherwise it will be performed continuously.

SEQ_LOOP_INVERSE
This flag works only if the **SEQ_LOOP** flag is set.
When the single **SEQ_LOOP** flag is set an infinite loop with four textures
(from 0 to 3): 0 1 2 3 0 1 2 3 0 1 2 3 ....

When the **SEQ_LOOP_INVERSE** flag is set the sequence
will be: 0 1 2 3 2 1 0 1 2 3 2 1 0 ...

TEXTURE_SEQUENCE **TextureSequence=**

SEQ_STOP_AT_FIRST

If this flag is set the **TRNG** engine is forced to set the first texture of the range when the animation is completed.

If this flag is OMITTED the last texture shown depends 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.

Text Indices array {...}
The text indices array can have up to 1000 different indices.
Separate each value with a comma ',' .

Each index will describe the index position for each texture within the specific animation range.

For example: If four textures are set in the TGA map,
each texture shows "A" the first then "B", "C" and "D

When a frame displays "A" type 0 (zero),
when "B" is wanted type 1, etc.

For example: TextureSequence= 1, 4, IGNORE, 3, 2, 1, 0
The sequence has Id = 1.
This is the number set in the flip-effect trigger to start/stop the sequence.

The animation is slow because it is set at 4 frames per second.
NO LOOP, because the IGNORE for SEQ_ flag is set.

The sequence shown in the game
will be with the following images:

D (3)
C (2)
B (1)
A (0)

Remarks: The ranges animated with a Texture Sequence command have to be set like "P-Frames" in the **Tomb Editor**.
The maximum number of textures to animate with the Texture Sequence is 16.
This means a number in the range 0 to 15 in the Indices Array fields.
The P-Frame range can be animated with different Texture Sequence commands.
The Id of the P-Frame range to animate will be set in the flip- effect trigger.

NEW SCRIPT COMMANDS

TRIGGER_GROUP TriggerGroup=

Syntax: TriggerGroup= IdGroup, ExportValue1 + **TGROUP_** flags, ExportValue2,
ExportValue3,
{Other Values 1/2/3 of exported triggers or conditions}

Scope: To use in the [Level] section

The Trigger Group permits two or more Triggers to perform at the same time using a single flip-effect called "Perform TriggerGroup..".

A Trigger Group can be activated using a condition trigger
Check condition of <&>TriggerGroup.

In this case the condition will be true if all of the conditions in the Trigger Group command are true.
To obtain data to type in the Trigger Group command use the Trigger Window of the **Tomb Editor** program by clicking on the button named "Export Script Trigger".

If no button displays with that name it means that the specific trigger cannot be exported in script format.

The following triggers can be exported: **Flip-effects**, **Actions**, **Conditions**.

A Trigger is exported as a script trigger to give a text file containing information like this:

---- Begin file ---------
Add following three values in your script command:
... , $2000, 96, $0
Information about exported trigger

Type: FLIPEFFECT
<#> : Lara. Disarm Lara in <&>way
<&> : Remove All. (Weapons + Ammos)
(E) :

--- End file ----

In the example the values to add to the Trigger Group command are the values: $2000, 96, $0

Other information read in the text files are given only for reference, so remember that the above three numbers perform the trigger "Lara. Disarm Lara" .

TRIGGER_GROUP TriggerGroup=

Arguments:

IdGroup
This is a number typed in to recognize the Trigger Group from other Trigger Group commands in the same [Level] section.

Use the IdGroup number to start via a Trigger the Trigger Group.

In the trigger window select a flip-effect **Perform <&>Trigger Group in the <&>** field.

Choose a Trigger Group from the Trigger Groups listed and set a different IdGroup value.
Start with "1" for the first Trigger Group and continue with "2" and then "3"
when another Trigger Groups is added to the [Level] section.

Exported Trigger values
After the IdGroup field type a list of triple values.
Each Group of three values is obtained from an "Export Script Trigger" operation.

Type these three values in the same order seen in the text from the "Export Script Trigger" button.

The general syntax for the Trigger Group becomes:

TriggerGroup= IdGroup, {Alfa1, Alfa2, Alfa3}, {Beta1, Beta2, Beta3}, {...},
{Omega1, Omega2, Omega3}

Remark: The "{" "}" parenthesis do not have to be written.
They are only shown to identify the optional grouping of the Trigger Data
that can be added in the Trigger Group command.

In the above **Alfa1, Alfa2, Alfa3, Beta1, Beta2, Beta3,** and **Omega1, Omega2, Omega3** are specific exported triggers.

DO NOT modify the values obtained from the exporting.
It is possible to add to the first value (like **Alfa1**, **Beta1** or **Omega1** in above example)

Some **TGROUP_** flags give specific meaning to the current trigger.
Add the following TGROUP to the first value of each exported trigger:

TGROUP_AND
TGROUP_NOT
TGROUP_OR

The above flags are Boolean operators.
They are used to link Condition Triggers.
If NO Boolean operator is used the Default value is always AND.
This means all conditions typed in sequence should all be True to get a True final condition.

TRIGGER_GROUP **TriggerGroup=**

For example: For a final TRUE condition when "Lara is climbing" or "Lara is monkey" use the **TGROUP_OR** operator in this way:

Condition1 (For Lara is climbing)
TGROUP_OR + Condition2 (For Lara is monkey)

If one condition is **TRUE** then the Final condition is **TRUE**.

TGROUP_NOT
This works in a different way.
It is like **AND_NOT** because it is linked with a previous condition.
An **AND_NOT** inverts the meaning of the current condition.

Use the **NOT** when the required condition is not required but the opposite in the list of the Trigger Window.

Use this opposite condition with a **NOT** to invert the meaning of that condition.

TGROUP_ELSE
Add **TGROUP_ELSE** to conditions or common triggers.
When this flag is added an "ELSE block" is started.
The ELSE block is only performed if a previous condition block gave a Negative condition.
Use ELSE to perform one of two different triggers (or block of triggers).

For example: For a Bonus level.
If Lara has found all of the secrets DO the Bonus level
If Lara has missed some of the secrets GOTO the Title level
to end the adventure.

Use an **ELSE** in this way:

Condition (Lara found <&> secrets)
(if true) Perform trigger (Load bonus level)
TGROUP_ELSE + Trigger (Perform title level)

If Lara has all of the secrets the Bonus level will be loaded,
ELSE,
the Title level.

None, two or more ELSE flags can be placed in the Trigger Group making very complex conditions.

A way to use ELSE is to test the conditions and perform a different trigger depending on the result.

TRIGGER_GROUP **TriggerGroup=**

For example: Condition: "Lara found 1 secret"
Trigger: Load Level 5
ELSE Condition: "Lara found 2 secrets"
Trigger: Load Level 6
ELSE Condition: "Lara found 3 secrets"
Trigger: Load level 7

In the example ELSE is attached to the conditions.
The rule used by the game to scan a Trigger Group is:

IF current_trigger is a condition: verify if the condition is True
IF it is True verify the other conditions.

If there are no other conditions perform the first non- condition trigger found and all the following non-condition triggers until the end of the Trigger Group or up to the first ELSE flag found.

IF the current trigger is not a condition:

perform it and all the following and then stop at the first ELSE flag found.

IF the current_trigger is a condition and the condition is FALSE,
skip the current condition block and skip the following non-condition block.

Continue parsing to find an ELSE flag and start to verify the conditions or perform a trigger from this ELSE flag.

TGROUP_USE_FOUND_ITEM_INDEX
TGROUP_USE_OWNER_ANIM_ITEM_INDEX
TGROUP_USE_TRIGGER_ITEM_INDEX

The TGROUP flag modifies the "ITEM_INDEX".
The "item index" is the index of the moveable used by the Trigger like Action or conditions.

By default each Trigger has an index when Exported.
Sometimes we need to perform actions (triggers) on a moveable not yet known.

TRIGGER_GROUP TriggerGroup=

Example for a Fight Situation:

When Lara hits an enemy the enemy will lose vitality.

The problem is to identify the index of an ENEMY because it could exist as different ENEMY TYPES for that slot type in the level.

When typing a trigger it is not known what ENEMY will meet Lara first.
To solve this problem use the flags to specify the index to use for the following triggers or conditions.

This method is possible when there is a way to locate dynamically a moveable.

For a condition like **Lara collides with moveable of <&>slot**,
when the condition is True the **TRNG** engine will find the index of the moveable that is touching Lara if the condition is attached with the flag **TGROUP_USE_FOUND_ITEM_INDEX** .

The found index in the current condition will be used in the following triggers or conditions to perform a specific trigger over this ENEMY to Damage it.
Remove vitality to <#> ENEMY

When using the above flag type the source trigger choosing any object because it will use another index of the object to perform that action.

The flag **TGROUP_USE_OWNER_ANIM_ITEM_INDEX** is easy to use.
Use this flag when the current Trigger Group is performed by an animation command inserted in some animation of a moveable.

In this circumstance the **TGROUP_USE_OWNER_ANIM_ITEM_INDEX** flag will force an index to use the index of the moveable where this animation command has been inserted.

The **TGROUP_USE_TRIGGER_ITEM_INDEX** flag, resets the Index of the moveable forcing it back to its original value.

If a previous trigger or condition used a flag like
TGROUP_USE_OWNER_ANIM_ITEM_INDEX or
TGROUP_USE_FOUND_ITEM_INDEX, use the real source indices in the current level and the following trigger or conditions using the flag **TGROUP_USE_TRIGGER_ITEM_INDEX**.

This flag informs the **TRNG** engine to use the effective index typed originally in that trigger.

See the **TGROUP_** constants

NEW SCRIPT COMMANDS

TURBO **Turbo=**

Syntax: Turbo=Flags (**TRB_**), FPStoKeep

Scope: To use in the [Level] section

The Turbo command allows tricks to be set to enhance the speed of the **TRNG** engine.

Only use this command when the level has a problem with the frame rate,
i.e. when the Frames Per Second goes below 29 fps.

Remark: At the bottom of the description for the Turbo command are some suggestions to enhance the speed of the level using some tricks in the project phase.

Description of fields:

Flags (TRB_)
Set one or more (linking with +) **TRB_** constant values.

See the **TRB_** values

FPStoKeep
This is only used if the **TRB_ADAPTIVE_FARVIEW** is set in the **Flags** field.
Type the frames per second (fps) value to Keep.

If **IGNORE** is typed the Default value is 29 fps.
This means that when the frame rate goes below 29 fps the **TRNG** engine will reduce the **FarViewDistance** until the frame rate recovers to 29 fps.

If the level has a very low frame rate, for example 15 fps,
it is better not to set a high frame rate in the keep FPS field.
Otherwise the **FarViewDistance** will be decreased too much.

For example: If 20 fps is set the **TRNG** engine will only reduce the distance of the **FarViewDistance** when the frame rate goes down to 20 fps.

How important is the best frame rate and the maximum far view distance ?
If the best frame rate is essential set 29 fps (the maximum value) and the **TRNG** engine will keep this frame rate but reduce the **FarViewDistance**.

If the **FarViewDistance** is important then set a low value like 20 fps.
The **FarViewDistance** will be preserved except when the game frame rate is very low.

TURBO **Turbo=**

If the **TRB_ADAPTIVE_FARVIEW** setting is not used type **IGNORE** in this field.

Some tips to enhance the speed of your level
If the Turbo command is not enough to get the result,
use some tricks while building the level to enhance the speed.

The problem of statics...
The static objects require more time to be shown than moveable objects.
For this reason many "slow down" problems are made because there are too many statics in the same room or nearby.

To solve this problem replace the static objects with moveable objects that look the same.

For example: Replace the static copying its meshes and textures with the **STRPIX** program. Use the "Export dxf" on the original static object and the command "Import dxf".

The transparency problem...
For complex reasons the **TRNG** game engine performs extra computes when an object has transparent textures. Hence it is not always possible to give up this use,
in some circumstances replace the transparent texture with some more detailed opaque mesh.

For example: To simulate a circle, use a square mesh with a transparent texture where the circle is the opaque side.
Create a real circular mesh using some triangles.
This avoids the use of transparent texture without giving up the shape required.

An alternative to Adaptive frame rate setting
If the working of the Adaptive frame rate is not satisfactory using the Turbo command, set in the level some specific zones that have a reduced **FarViewDistance**.

Use a new flip-effect trigger to set dynamically a larger **FarViewDistance** in the current level.
Using this method disable the **AdaptiveFarView** in the turbo command otherwise there will be conflicts.

Study the level to understand what zones are important to have a gain in frame rate reducing the **LevelFarView**.

For example: If the level has a wood with trees very close together it is possible to have a problem with the frame rate speed.
In this case place a strip of triggers to enable the **FarViewDistance** to be reduced when Lara enters the wood
Another strip of triggers is set to give a larger **FarViewDistance** when Lara leaves the wood.

NEW SCRIPT COMMANDS

WINDOWS_FONT WindowsFont=

Syntax: WindowsFont= IdWindowFont, WindowFontName, WindowsFontFlags (**WFF_**), SizeFont, ColorRbgId, ShadowColorRgbId

Scope: To use in the [Level] section

The Windows Font permits the choice for the settings for the Windows Font.

Do not confuse the common game font used in Tomb Raider with the Windows Font.

The Windows Fonts can only be used in the **TRNG** engine special page layout like Lara's Diary.
Other **TRNG** engine features will be added in the future.

IdWindowFont
Type a progressive number used as an identifier to reference this font setting
from other script commands.

WindowFontName
Type a standard Font name.
There are many Windows Fonts.
Use a common Font otherwise there is the risk that the font chosen is missing on a player's computer.

The most common font names are:

Arial
Comic Sans MS
Courier
Courier New
Ms Sans Serif
Times New Roman
Verdana

Remark: The name in this field is also in the **Strings section.**

For example: To use the "Courier New" Font for the text in this field:

WindowsFont= 1, Courier New, ...

and then type this text into the [Strings] or [ExtraNG] strings section.

Remark: If the font name is omitted by typing **IGNORE** the **TRNG** engine will find a generic font responding to other settings, like bold and size.
To get the best results it is better to type a font name.

WINDOWS_FONT WindowsFont=

WindowsFontFlags (WFF_)
Type one or more **WFF_** flags linked with "+" sign.

If **IGNORE** is typed in this field the **TRNG** engine will use the Default setting:

WFF_BOLD+WFF_SHADOW+WFF_LEFT_ALIGN for common text settings.

The Title setting is: WFF_CENTER_ALIGN+WFF_UNDERLINE+WFF_ULTRA_BOLD

See the **WFF_** constants

SizeFont
This value is the height in pixels of a large character at a screen resolution 1024 x 768, where the width of the char is 50 % of the SizeFont typed.

This size is stretched according to the current effective resolution of the game screen.

Remark: Be aware of the size of the text for different screen resolutions
It is important to set a couple of [Size Font] and [Frame Size for the text] to have some free space available in the frame text to minimize any further adjustment.

ColorRbgId and ShadowColorRgbId
These two fields can set the colour for the font (ColorRbgId) and a shadow colour (ShadowColorRgbId).

The value typed is the identifier of the ColorRGB in the [Level] section.

Choose a shadow color different from the text color.

For example: If the text colour is White choose Black for the shadow color and vice-versa.

The ColorRGB commands are typed BEFORE the Windows Font command that uses them.
Otherwise the **TRNG** engine cannot locate the correct ColorRGB command.

NEW SCRIPT COMMANDS

WINDOW_TITLE WindowTitle=

Syntax: WindowTitle= NameOfCurrentLevel

Scope: To use in the [Level] or the [Title] section.

With this command text can be set that will be shown in the Window Title Bar of the game. This command only works when the game is in a Windowed mode.

Example: WindowTitle= The Last Revenge by **PAOLONE**

Remark: Remember to add the same text in the **english.tx**t file.
For different language versions type the text in the English section and then click on the different language combo box and overwrite the text shown in the selected position.

NEW SCRIPT COMMANDS

WORLD_FAR_VIEW **WorldFarView=**

Syntax: WorldFarView=MaxSectorDistance

Scope: To use in the [Options] section

The default value in the original tomb4 engine:
WorldFarView=MaxSectorDistance = 20 sectors

Valid range: Minimum = 1 Maximum = 127

The **WorldFarView** sets the maximum distance to use in any level of the game.

Set a larger number with the **WorldFarView** and smaller values in the **LevelFarView** for different levels according to the problem with frame rate for wide and complex scenes.

See the **LEVEL_FAR_VIEW** command and the **FOG_RANGE** command for more information.