TRGameflow

By Troye, for the TR3 Level Building community.
https://github.com/Trxyebeep/TRGameflow

TRGameflow is a custom made script compiler for the Tomb Raider 3 game engine.

Using:

- Make sure your script and string files are in the same folder as the program.
- Launch the program.
- Type in the full name of your script file, eg. script.txt
- TRGameflow will output your new script file in the same directory.

TRGameflow produces 100% accurate results to the original script compiler.
With this package you should have received an example script.txt and english.txt, these are accurate recreations of the original Tomb Raider 3 script and the TRGameflow output is byte-for-byte accurate with the original English TOMBPC.DAT provided with the Steam version of the game.

In this document you will find a detailed explanation of how the script system works, list of all the possible commands/values you are able to use, and details on each command.

Colour codes:

Red denotes a command.
Blue denotes a value/string.
Purple denotes types/highlights.
Dark red denotes an example script line.

If at any time during the document you run into a technical term you are not familiar with, there is a glossary that lists some words at the bottom of the document.

The System

First things first, it is important to understand what the contents of the script files are.

The script is made of two files:

- Your main .txt file is the file that contains all the gameflow commands that directly alter how the game behaves (literally alters the game-flow)

- Your secondary .txt file is the file that contains game strings. Not every string is contained within this file, a few are taken directly from the main .txt file.

For the remainder of this document, the main file will be referred to as “script file”, and the secondary file as “strings file”.

Note: Command names/keywords are NOT case sensitive! For example, Frontend is the same as frontend, FRONTEND, or even FRoNTenD.

---

strings.txt contents

We will start with strings.txt as it is easier to cover.
There are two main categories in this file.

1- game_strings block
2- pc_strings block

-The first block contains the ‘general’ game strings, such as New Game, Load Game, etc.

-The second block contains the ‘PC Specific’ strings, such as Resolution, Gamma, etc.

-Both block keywords need to be followed by a colon : to inform the compiler that the list is starting. Eg. game_strings:

-Both blocks need to end with the keyword **END** followed by a colon : to inform the compiler that the list is ending.

-The amount/order of the strings, and where each string is used in the game is hardcoded on the engine side and should not be changed.

-The only thing you should change is the string contents, for example you can change New Game to Start Adventure, etc.

---

script.txt contents

First things first: Comments.

Comments are lines in the script that the compiler discards at compile time, they should be used as little notes for yourself in your script.

There are two types of comments:
1- One line comments //
2- Block comments /**/

One line comments, as the name suggests, are one single line, or part of a line.
One line comments need to start with a double slash //.

Couple examples:
//This line is a comment and will be discarded by the compiler
Level: Jungle //This is the first level in my new game!

In the first example, the entire line is skipped.
In the second example, the compiler will read the line Level: Jungle and discard the rest.

Block comments are entire blocks that are discarded for the compiler.
-Block comments need to start with a slash / directly followed by an asterisk *, making a /*.
-Each line in a block comment needs to start with an asterisk *.
-You can use multiple asterisks * in both these cases, as long as there is at least one.

Example:
```
/********************
* This entire block
** Will be skipped **
*** by the compiler!!!! ***
********************/
```

The entire block is skipped, no matter the contents.

Now that you know what is *not* part of the compilation process, it's time for the real commands.
Commands across the entire file are formatted like this:

command: value

Quite simple, right?
-Every command needs to be followed by a colon : to denote the end of the command, and the start of the value.

Note that not every command requires a value, some commands are there to simply tell the compiler that a new block is starting, or to set a simple setting, etc. You will see examples of this in this document.

Blocks

So, what exactly is the script? Is it just a bunch of commands? Well, yes and no.
The script is made of a few blocks, and each block contains some commands. All commands must be contained within a block, There are only two exceptions to this, they are the Description and Gamestrings commands, which will be explained within the document.

The script is made up of these blocks:

1- Options	//Special block
2- Title	//Special block
3- Frontend	//Sequence block
4- Gym	//Sequence block
5- Level	//Sequence block
6- DemoLevel	//Sequence block

-These block keywords are commands that simply tell the compiler that the block is starting. It is preferred to follow the above ordering of the blocks in your script.

-Special blocks each have their own set of commands, while sequence blocks all share the same list of commands.

-Each block end needs to be specified using the END: command.

The Commands

First let's explain the Description and Gamestrings commands.

Description command is a simple string that is printed on top of the output file, this is used as a small description of what the file is.

Example:
Description: Tomb Raider III Script. E3 Release (c) Core Design Ltd 1998

This command is optional, but, if used, is preferred to be at the very top of the script, before all your blocks.

Gamestrings is a very important command, it specifies the name of your strings file.

Example:
Gamestrings: english.txt

This command must exist in your script. It is preferred to be at the very bottom of the script, after all your blocks.

The Options Block

The best way to describe this block is "Main Game Options". It contains the main options for how the game acts in a few events.

Possible commands are:

- language: Possible language values: English, French, German, American, Japanese, Italian, Spanish. This option controls the “Assault Course” and "Quadbike Track" strings in Lara’s home. These two strings are hardcoded to this option. Also, using German changes blood colour from Red to Purple.

- secret_track: unused (leftover from TR2).

- cypher_code: Defines the encryption/decryption xor key for strings. Remove this option for the strings to not be encrypted. Possible values: 0 .. 255.

- firstoption: The first thing the game does after booting up. For possible values check the option types list.

- title_replace: This command is to be used ONLY with the title_disabled command. This command specifies the level index to load in place of title.

- ondeath_demo_mode: Specifies what the game does if Lara dies during a rolling demo. For possible values check the option types list.

- ondeath_ingame: Specifies what the game does if Lara dies in game. For possible values check the option types list. For possible values check the option types list.

- on_demo_interrupt: Specifies what the game does if the demo playback is interrupted with input. For possible values check the option types list.

- on_demo_end: Specifies what the game does when the demo ends naturally. For possible values check the option types list.

- noinput_time: The time the game waits in the title before kicking off demo playback, in frames. (1 second = 30 frames).

- singlelevel: Level index. Using this option forces the game to only load this level.

- demoversion: Use this command to launch the game in demo mode: forcing the game to return to title when a level end trigger is hit, and changing the "Exit to Title" string to "Exit Demo". No value required.

- title_disabled: For use with the title_replace. No value required.

- cheatmodecheck_disabled: Use this command to disable ingame cheats (not DOZY). No value required.

- noinput_timeout: Use this command to enable demo interruption with input. (I.e if you don't include this option, demos cannot be ended with input.)

- loadsave_disabled: Use this command to disable saving and loading. No value required.

- screensizing_disabled: unused.

- lockout_optionring: Use this command to disable going into the options ring in the inventory. No value required.

- dozy_cheat_enabled: Use this command to enable DOZY cheat. No value required. This command is obsolete with the tomb3 project.

- select_any_level: Use to enable level select. No value required.

- cheat_enable: Secondary check for DOZY cheat. No value required. This command is obsolete with the tomb3 project.

- securitytag: unused.

- end: Denotes the end of this block, must be used at the end of the block.

---

option types: These are the values possible for the designated commands (see above).
- EXIT_TO_TITLE: Forces the game to go to the title.
- LEVEL: Use example: "Level 1" - Forces the game to load the specified level.
- DEMO: Use example: "Demo 1" - Forces the game to load the specified demo.
- Sequence: same as LEVEL.
- ExitGame: Forces the game to quit if used within title, reloads the last reloaded save if used within a level.

Example of this block:

```
Options:
Language: English
cypher_code: 166
firstoption: EXIT_TO_TITLE
//select_any_level:
//title_disabled:
//title_replace: 1
demoversion:
end:
```

The Title Block

This block specifies a file list for the game's main title routines, and the main theme to play during the title. The file list must be in this order
1- Title level file path.
2- Title image background file path.
3- Copyright image background file path.

Possible commands are:

- Game: Adds the level file path of the title level to the file list. **Must be the first command in this block.**

- PCFile: Adds an image file path to the file list.

- track: Specifies the number of the music track that plays in the title.

- end: Denotes the end of this block, must be used at the end of the block.

Example of this block: (or, how this block needs to be, rather)

```
Title:
Game: data\title.TR2          //This must be first
PCfile: pix\titleuk.bmp       //Title image must be second
PCfile: pix\copyrus.bmp       //Copyright image must be third
track: 5
end:
```

The Frontend Block

This block is a sequence block. The frontend block must only contain starting FMV info. I.e the file names of the intro FMVs

For possible values, refer to the Sequence Block Commands portion of this document.

Example:

```
Frontend:
fmv: FMV\LOGO.RPL
fmv: FMV\INTR_ENG.RPL
end:
```

The Gym Block

This block is a sequence block. The Gym block contains the sequence of commands to be executed when selecting Lara's Home option in the title.

For possible values, refer to the Sequence Block Commands portion of this document.

This block's command keyword, Gym, specifies the name of the level. Example:
Gym: Lara's House

Example block:

```
Gym: Lara's House          //The name of the gym level
Load_pic: pix\house.bmp    //The loading screen
track: 2                   //The background ambience
game: data\house.TR2       //The level file path
Key1: Racetrack Key        //The inventory name for the KEY_ITEM1 slot
end:
```

Notice how the Load_pic and track commands are used before the game command. This is because the gym block is a sequence block. Meaning these commands are executed ingame in this same order specified in the script. If, for example, the game command was used before the Load_pic command, you will not get a loading screen before the level loads.

The Level Block

This block is a sequence block. The level block adds a new level to the list of game levels, and contains the sequence of commands to be executed when playing this level.

For possible values, refer to the Sequence Block Commands portion of this document.

This block's command keyword, Level, specifies the name of the level. Example:
Level: Jungle

Example of two level blocks:

```
//Level 6
Level: Crash Site                    //Level name
track: 33                            //background ambience
Load_pic: pix\southpac.bmp           //Loading screen
StartInv: PICKUP1
game: data\crash.TR2                 //Level file path
Pickup1: Swamp Map                   //Specify PICKUP_ITEM1 slot's inventory name
Key1: Commander Bishop's Key         //same for key 1
Key2: Lt. Tuckerman's Key            //and 2
track: 65                            //Audio track to be played in the cutscene
cutangle: 16384                      //starting camera angle of the cutscene
cut: cuts\cut4.TR2                   //cutscene file
Complete:                            //End the level, show statistics
end:

//Level 7
Level: Madubu Gorge                  //Level name
track: 36                            //background ambience
Load_pic: pix\southpac.bmp           //Loading screen
game: data\rapids.TR2                //Level file
complete:                            //End the level, show statistics
end:
```

Just like the Gym block, note how the Load_pic and track commands are used before the game command. Now notice how the second track command is used after, and actually does not affect the actual level's ambience, but the cutscene ambience, as the command for the cutscene comes after it.

And as you can see, level 6's block ends, then level 7's block starts. **This list of level blocks is what determines the order and gameflow of the entire game.**

The last level in your game must contain the gamecomplete command. This command tells the game to show credits. More importantly, following this level block there must be one extra level, the bonus level (All Hallows in original TR3). The bonus level is unfortunately

NOT optional currently. Setting up the bonus level is very simple and similar to the rest of the level blocks.

Example:

```
//Level 19
Level: Meteorite Cavern
track: 26
Load_pic: pix\antarc.bmp
game: data\chamber.TR2
fmv: FMV\ENDGAME.RPL
gamecomplete:
end:

//Bonus level
Level: All Hallows
track: 30
Load_pic: pix\london.bmp
game: data\stpaul.TR2
Key1: Vault Key
complete:
End:
```

As you can see, the 'last' Level block does not use the complete command to avoid showing the statistics screen, and instead uses the gamecomplete command to go straight to the credits.
Now, after the game is finished showing the credits, it checks if the player got all the secrets in the game, if they didn't, it goes back to the title. If they did, it loads the bonus level. It is important for the bonus level to end with complete. Do NOT use gamecomplete again, or you will keep going back to the bonus level every time it ends.

The DemoLevel Block

This block is a sequence block. The DemoLevel block adds a new demo level to the list of demo levels. Demo levels must be defined **after** the regular level blocks. Their format is very close to the regular level blocks.

For possible values, refer to the Sequence Block Commands portion of this document.

This block's command keyword, DemoLevel, specifies the name of the level. Example:
DemoLevel: Aldwych Demo

Example:

```
DemoLevel:    Aldwych Demo
track:        74
Load_Pic:     pix\london.bmp
PCDemo:       data\sewersDemo.TR2
end:
```

Sequence Block Commands

This is the list of commands available for use in sequence blocks:

- FMV: Adds the value to the FMV file list, and adds a “Play FMV” command to the sequence.

- PCFmv: Adds the string to the FMV file list, without adding a “Play FMV” command to the sequence. There’s no real reason to use this command.

- GAME: Specifies the level file name and adds a “Play this level” command to the sequence.

- CUT: Specifies a cutscene file name and adds a “Play this cutscene” command to the sequence.

- CUTANGLE: Specifies the orientation angle of cutscenes, Possible values: -32767..32767

- DEMO: Specifies the demo level file name and adds a “Run this demo” command to the sequence.

- TRACK: Specifies the ambient track for the level that follows it. For example, using this command before the GAME command sets the ambience for the level, using it after GAME and before CUT sets the audio track for the cutscene. See example in page 11.

- SUNSET: TR2 leftover, unused. No value needed.

- LOAD_PIC: Specifies the name of the loading screen file and adds a “Display this loading screen” command to the sequence.

- DEADLY_WATER: Unused.

- REMOVE_WEAPONS: Removes Lara’s weapons from her inventory, use **before** the GAME command.

- REMOVE_AMMO: Removes Lara’s ammo, flares, and medipacks from her inventory (keeps one small medipack, though), Use **before** the GAME command..

- NOFLOOR: Specifies the world Y coordinate at which Lara dies if she goes below. (Eg. in Thames Wharf, or TR2 Floating Islands). To get the value in Tomb Editor, Select a square at the height you wish Lara to die at, and look at the bottom bar, you will see a
Global area = (x, z) -> y = [y floor, y ceiling]
The value you want is **y floor multiplied by 256, then negate the result**. For example if your y floor is 17, your NOFLOOR script value is 17*256 = 4352. Then we negate the result, which makes it -4352

- BONUS: Unused in the original game, but the functionality is supported in the engine. When picking up the three “SECRET_ITEM”s, (slots 221, 222 and 223), the game adds the specified item to Lara’s inventory. Can use multiple BONUS lines to add multiple items. Use this command **before** the GAME command. For possible values, refer to the AddInv Items List table at the end of this list.

- STARTINV: Adds the specified item to lara’s inventory at the start of the level. Use this command **before** the GAME command. For possible values, refer to the AddInv Items List table at the end of this list.

- STARTANIM: Specifies an animation state number to force Lara to at the start of the level. The animation must be in the Lara extra anims slot. Use this command **before** the GAME command. Note this command requires extra data in the level file for the camera movement, which is done by setting up a flyby sequence in Tomb Editor.

- SECRETS: Originally unused, repurposed for tomb3. Check the new commands document.

- KILLTOCOMPLETE: Not used, do not use this command.

- PUZZLE1: Specifies the inventory name for the PUZZLE_ITEM1 item.

- PUZZLE2: Specifies the inventory name for the PUZZLE_ITEM2 item.

- PUZZLE3: Specifies the inventory name for the PUZZLE_ITEM3 item.

- PUZZLE4: Specifies the inventory name for the PUZZLE_ITEM4 item.

- KEY1: Specifies the inventory name for the KEY_ITEM1 item.

- KEY2: Specifies the inventory name for the KEY_ITEM2 item.

- KEY3: Specifies the inventory name for the KEY_ITEM3 item.

- KEY4: Specifies the inventory name for the KEY_ITEM4 item.

- PICKUP1: Specifies the inventory name for the PICKUP_ITEM1 item.

- PICKUP2: Specifies the inventory name for the PICKUP_ITEM2 item.

- PCDEMO: Same as DEMO.

- COMPLETE: Adds a “Show Statistics” command to the sequence. Use **after** the GAME command.

- GAMECOMPLETE: Adds “Show Credits” and “Check for bonus level” commands to the sequence. Use **after** the GAME command.

- END: Specifies the end of the block.

AddInv Items List:

These are all the possible values for the BONUS and STARTINV commands:

- PISTOLS
- SHOTGUN
- AUTOPISTOLS
- UZIS
- HARPOON
- M16
- ROCKET
- GRENADE
- PISTOLS_AMMO
- SHOTGUN_AMMO
- AUTOPISTOLS_AMMO
- UZI_AMMO
- HARPOON_AMMO
- M16_AMMO
- ROCKET_AMMO
- GRENADE_AMMO
- FLARES
- MEDI
- BIGMEDI
- PICKUP1
- PICKUP2
- PUZZLE1
- PUZZLE2
- PUZZLE3
- PUZZLE4
- KEY1
- KEY2
- KEY3
- KEY4
- CRYSTAL

Glossary

- Command: A keyword for the program telling it to do a specific action.

- Value: A representation of user-inputted data that may be required for any certain commands. Can be of any type, depending on the command.

- String: A sequence of characters.

- Type: The way a certain value is represented. Can be a number, a single character, a keyword, a string (sentences, words) etc.

- Block: A collection of commands