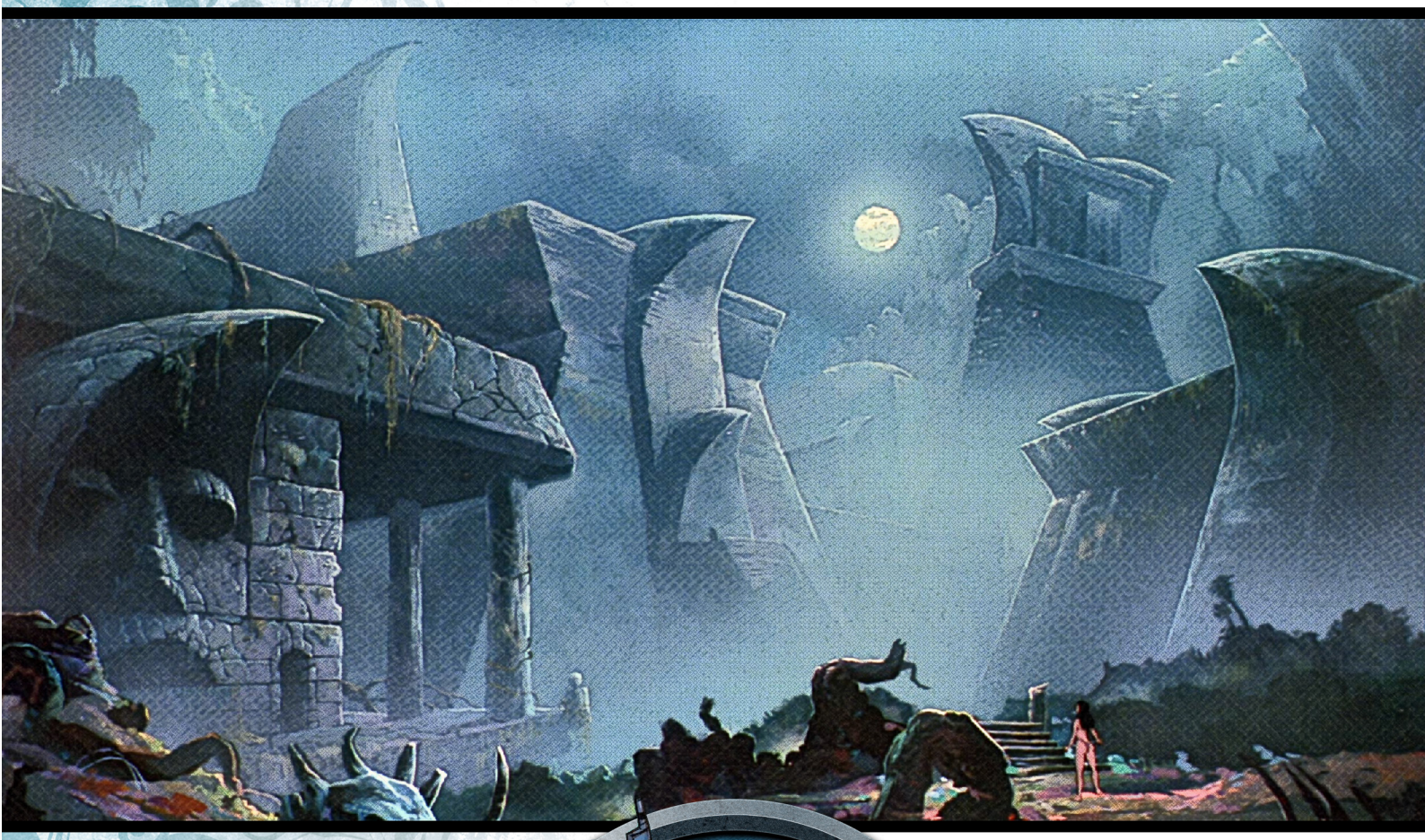


KIMMY
SCRIPT ENGINE

CREATE YOUR OWN FMV GAME

KARIS 2020/2024 - V4.01

Welcome!

This tutorial will show you how to create your own FMV game using SINGE 2.1. It is really simple and quick. If you've got a simple 10 minutes movie, then you can probably be ready to play it as a FMV in just one hour ! No programming at all, you just need to enter your list of moves in the KIMMY SCRIPT ENGINE.



A bit of History

Daphne is a laserdisc game emulator. It was created by **Matt Ownby**. It is called Daphne because it's the name of the princess in Dragon's Lair. Daphne was a pure emulator, using the original Roms of the game.

SINGE 1.0 (Singe is the name of the dragon) was an add-on to Daphne created by **Scott Duensing** in 2006. It was a dll which allowed, via a LUA script, to create new Laser discs. The script was a kind of «fake Rom». A few games were ported by **RDG** (mostly American Laser Games). But until 2019, nobody released a new game from scratch (RDG did some, but never released them). **Fire & Ice** was the first of its kind. I used a script from RDG and modified it in order to have all the needed resources in one single file. The LUA Singe Engine was then totally re-written and expanded until today (v4.0).

In 2020, Scott decided to give a new life to Singe and created **SINGE 2**. This new installment is no more a dll, so it doesn't need the good old Daphne to run. SINGE 2 is not an emulator, since it can't run original Roms. But the graphic performances and the scripting is now enhanced. Many games were ported or created and now it's your turn...

KIMMY SCRIPT ENGINE

KIMMY SCRIPT ENGINE is the script transforming your video into a game.
It was designed to emulate most of the behaviors you will encounter in many FMV games.
Do you want random levels ? Complex paths ? Show this and not that ? Everything is possible without programming a single line.

The script is evolving regularly with new features and moves.



What do you need?

- The **SINGE 2 program**. It's available for Windows, OSX and Linux,
- A **movie editing software** (like Adobe Premiere) to edit your movies and check the frames,
- A **text editor**: Notepad is enough, but Sublime Text is better (and free).
- Optionally, **a picture editing software** if you want to change some graphics (arrows, etc...)

Limitations

- Most video formats are supported. I use mostly H264 mp4,
- Max size of the video: 3.6 Gb (you may use several video up to this size but it's a more complex process),
- Any number Levels + 1 Secret Level (each regular level can be mirrored automatically),
- Any number of Scenes in each Level,
- Any number of moves within a Scene,
- Any number of death scenes (death can be mirrored).

General Concepts

You game will be divided in Levels. Each **Level** will be divided in Scenes. In each **Scene**, you will declare your moves, asking player to enter the **correct move** between frame X (Start), and frame Y (End). If the player enters a **wrong move** (or no move), this will trigger a **death scene** and restart at the beginning of the scene (or elsewhere if you change this option).

All those informations are located in a single file that you will edit simply with a text editor.

Features

Your game will have these features (of course you are free to use them or not):

- Keyboard and Pad controls
- Movement Arrows and up to 3 Action buttons,
- COMBO: you can ask the player to combine an arrow plus a button
- Special actions MASH, HOLD, LETGO, MULTI, LOOP, TIMED
- CHOOSE action to make a timed decision
- PATH and YESNO actions to choose between different moves, scenes, levels
- Difficulty levels : Easy to Extreme. Easy mode has no Special or Combo actions. Harder difficulty levels means less frames to enter your moves
- Selectable starting level and scene
- Secret Level: a Special level, triggered in certain cases
- Option screen for users
- Selectable gaming mode (Normal, Life Bar, Percent, Survivor, Die & Retry, Kid, Movie)
- Selectable gaming behavior (Normal, Random, Map, Tiers, Select)
- Selectable input behavior (Show moves, Nothing, Guess, Just sound, Cliffhanger style)
- Selectable Hints
- Selectable extra video (Get Ready, Extra Death, Level Cleared)
- Kid mode : No arrows, just button 1. The only input is the Action button
- Saves : 3 slots of save plus an auto save at the end of each scene. You can access those slots with a dedicated menu or directly from the game
- Death rewind : you can choose from the menu where to go back after a death. Either Scene, Level, Last Move (the move you just missed) or Next Move (you pass the missed move, like in Space Ace).

OK, I'm ready. Let's go!

The creation of your game will be divided in 3 parts. Video editing, files preparation and SINGE editing. In this tutorial i will cover the video editing part very slightly (there are plenty of good tutorials for this all over the web).

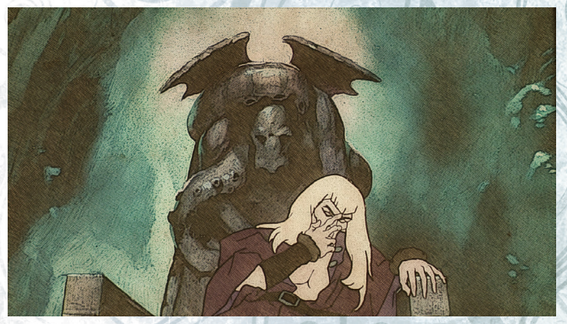
Part 1: Edit the video file

For a complete game you'll need to gather a few mandatory parts in your video file:

Title: this part is the first one that will be played when you start the game. Of course you can have just a static image than spans over a few frames. Don't make it too long because you can't do anything while it's playing.



Attract: it is a longer video to present the story. It is longer because it will loop, waiting for the player to start the game. You can have up to 3 different Attract movies.



Death: in FMV games, you die a lot! All the death sequences are in this video. You need at least one. A good length for one death is around 50 frames (but of course, it can be 50 frame static image), it will give the player enough time to breath between failures.



Levels: this is where the game is played. You need at least 1. In those videos, the first part can be declared as an intro and skipped with the push of a button. It is useful to have an intro clip to present the action, but it's highly optional.

Menu: this part will contain all the UI of the game.



a **single frame** for the option menu



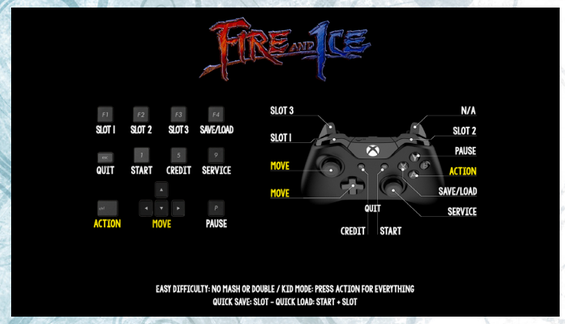
a **single frame** for victory



a **single frame** for the save menu



a **single frame** for rankings



a **single frame** for the Controls



a **video** for Game Over



a **video** for continuing the game



a **video** for High Score

All those parts will be shown by the program during special moments :

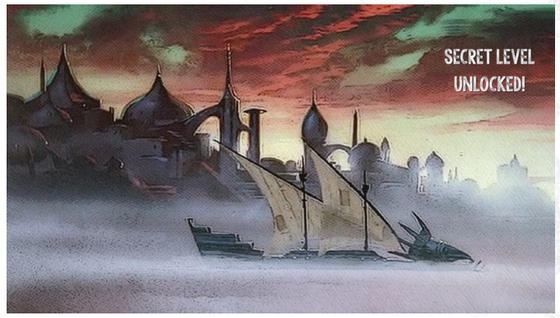
- Option menu when you hit the service key (9)
- Victory when you finish the game
- Save menu when you enter this menu (F4)
- Rankings and commands, during the Attract (you can hit Ctrl to swap between things)
- Game over when you die (after continue)
- Continue when you die (if you allowed continue in the option menu)
- High Score if your score is among the Top 10.

Video length

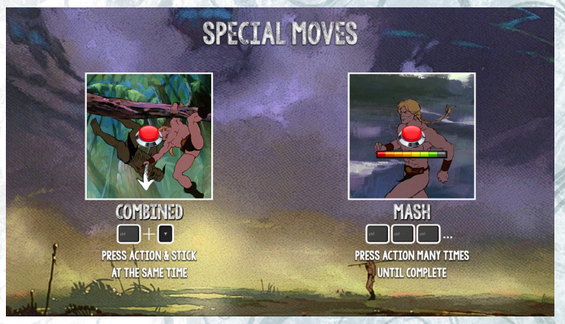
For those videos, you need to have the good length, because it will define the timing of the event (for example entering high score). So sometimes you don't need to be too long (Game Over is around 200 frames), and sometimes you need to be longer (High Score is around 700 frames).

Also note that any video can be a long static frame (for example in Fire & Ice, all the menu videos are one repeated static frame, with music).

Menu (optional things): for a better UI, or if you plan special features for your game, you may add some of those.



a **single frame** for the Secret Level



a **single frame** for the Special Controls



a **single frame** for the Trophies



an alternative (happy) Game Over **video**



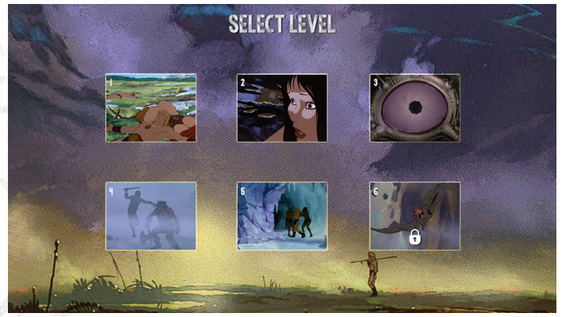
a Level Cleared **video**



a **video** announcing High Score



a **video** for Rankings



some select **videos**



a Get Ready **video**



an Extra Death **video**

All those parts will be shown by the program during special moments :
- Secret Level when you finish the game under specific conditions
- Special Controls right after Controls (during the Attract)
- Alt. Game Over when you beat a High Score or finish the game
- High Score will be placed before the part where you enter your name
- Rankings will be placed after you enter your name after a High Score
- Level Cleared, Extra Death and Get Ready are defined in the option menu (see later).

There are a few more things (Map, Select Difficulty). We'll discuss about them in the appendix.

The best way to edit

There are many tutorials all over the web to learn editing. Here are a few tips:

Mount each video individually. I mean, one video for the intro, one for each attract, one for each level, one for all the elements of the menu and on for the death. Keeping each level independent will help you to enter the moves later.

When you're done, gather all those videos altogether in a new file. Leave a few frames between each segment (in my case the space is 15 frames).

When your file is ready, export it as a single mp4 movie (remember the size limit). You can play with the compression parameter to fit. You are now ready for Part 2.

Should i put titles inside the videos or not ?

Both options are available. If you put your own titles, you may for example animate them, color them, choose their position, whatever you like...

If you are to lazy for this, or if you don't have Photoshop or After Effect, there is an option in the program to put titles automatically (see Part 3).

Part 2: Prepare the files

So now, it's time to prepare the files for SINGE. It will be short.

- Go into the Singe/Blank directory
- Rename Blank.singe with the name of your game- Copy your movie inside Blank/Movie
- Rename Blank directory with the name of your game
- Go back to the root and rename Blank.bat file with the name of your game
- Edit Blank.bat you should find this line:

```
Start Singe.exe -k -z -d Blank\Index -v Blank\Movie\Blank.mp4 -w Blank\Blank.singe
```

- Once again replace all the Blank with the name of your game and your movie like this (if the name of your game is Toto):

```
Start Singe.exe -k -z -Toto\Index -v Toto\Movie\Toto.mp4 -w Toto\Toto.singe
```

Save the bat and you're done! Now, it's time for the funny Part 3.

Part 3: Create your game

Now you are ready to create your game. It will be done in 4 steps. Go into your game directory and open Toto.singe (the name you just gave to the file), this is where you'll do everything.

Let's edit this file (you must change blue text, green is highly optional):

1) General Settings

```
dofile(«.Singe/Script/globals.singe»)
```

Just leave this line like it is (it calls all the other needed files)

```
MovieFPS = 23.976
```

Here, you need to put the speed (in frames per second) of your movie. The program will use this speed for some events. Common speeds are 23.976 or 29,97 or 25.
Note: this parameter is not used for reading the movie

```
altCfg = false
```

If altCfg is true, then the program will manage a second set of configurations (Options, High Score, Save). In that case the "main" game file will have altCfg = false and the other one altcCfg = true. It allows for example to propose 2 versions of the same game with different moves. A good example is Ninja Hayate (Arcade and PSX version).

2) Defining frames for the various elements of the game

In this section, you will tell SINGE where to find the videos for you Attact, Levels, Menu, etc. In order to achieve this, you need to open your edited video in Premiere and note the precise frame where each sequence is located.

```
offsetTitle = 0
offsetTitleend = 148
```

Frames for the beginning and the end of the Title video.

```
offsetIntro01 = 164
offsetIntro01end = 1512
offsetIntro02 = 75627
offsetIntro02end = 75706
offsetIntro03 = 0
offsetIntro03end = 148
```

Frames for the beginning and the end of each Attract video. If you've got less than 3 Attract videos, you can repeat the values of Intro01 or Title.

```
offsetIntroGame = 0
offsetIntroGameend = 0
```

Frames for a sequence played at the beginning of the game. Played only once after start.

```
offsetMenus = 74990
```

Frame for the beginning of the Menu video.

```
frameOptions = offsetMenus +0
frameVictory = offsetMenus +3
frameSave = offsetMenus +6
frameRankings =offsetMenus +9
frameControls = offsetMenus +12
frameSpecial = offsetMenus +15
frameSecret = offsetMenus +0
frameTrophy = offsetMenus +30
frameHints = offsetMenus + 24
frameQuit = offsetMenus +27
frameNewGame = offsetMenus +30
```

Frames for various elements of the menu (as defined in part 1). There is no end, cause the program is just waiting for a frame, not a video. The program just wants a number, so there are two way to do this: relative or direct (same result).
For example for frame Victory, offsetMenus +3 is exactly the same than 74993. If you created each video independently, you will prefer to work in a relative way, but if you're looking at the mounted video, you will go direct. If there are things you don't plan to use (like Special, Secret, Trophy, Hints) then the value is not relevant (you can leave it like it is).

```
offsetGameOver = 75038
offsetGameOverend = 75120
offsetGameOverAlt = 75136
offsetGameOverAltend = 75612
offsetNewHScore = 75940
offsetNewHScoreend = 75970
offsetEnterHScore = 75971
offsetEnterHScoreend = 76864
offsetRankings = 76880
offsetRankingsend = 77068
offsetContinue = 77084
offsetContinueend = 78163
```

offsetClear = 75722
offsetClearend = 75842
offsetGetReady = 75627
offsetGetReadyEnd = 75657
offsetSupDeath = 78179
offsetSupDeathEnd = 78249
offsetQuit = 0
offsetQuitend = 1

Videos for various elements of the menu. Each blue element needs to be defined, as those videos are important for the program. The green ones are only used if you choose them in the Options menu.

If you have no Level Clear, Extra Death or Get Ready video, you can put 0 for both values and they won't appear in the Options. For the red ones, you need to enter something even if you don't have those video cause the program will search for them. For GameOverAlt, enter the same values than for GameOver. For NewHScore and Rankings, enter 0 for both values.

offsetMovieEnd = 74975

Last frame of the last Level. Used by the Movie Mode.

totalDeath = 28
offsetDeath = 78270

Total number of Death videos and Frame for the beginning of the Death video.

Death[1] = {offsetDeath +0, offsetDeath +28}
Death[2] = {78314, 78355}
and so on (as long as you wish)....

In each line, put Start and End frames for each death video. It can be a long list depending on the number of different death you have. It can be short too. You need at least one death scene (it can be the same for all moves). As for the menu it can be presented relative (like Death[1]) or direct (like Death[2]), you choose.

3) Levels

Here are the basic informations for your levels.

finalstage = 5
PlayOrder = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}

finalstage tells the program where the game ends. For example if you put 1, SINGE won't search for a second level and the game will end after stage 1. You can put any value between 1 and 16. PlayOrder indicates the normal order of play for your levels. Normally, you don't need to change this line (even if you have less than 16 levels). But if you want to alter this order for any reason you can change it here.

AllowSecret = true

True or false. It will tell SINGE if you've got a secret level. If you do not have one, put false.

Access the Secret Level

There are two ways to launch the secret level :

- Finishing the whole game with one credit
- Special combination of keys (L-Shift + L-Alt + UP + RIGHT) during the Attract video.

RelativeFrames = true

Defines how you will enter the frames numbers inside the levels. True: frame 1 is at the start of each level. False: Frame 1 is at the start of the disc.

Level[1] = {"Title 1", 9749, 9799, 2, 0, 0,-1}
Level[2] = {"Title 2", 1302, 15232, 8, 0, 0,-1}
Level[3] = {"Title 3", 20787, 21164, 13, 0, 0,-1}
Level[4] = {"Title 4", 27789, 27847, 9, 0, 0,-1}
Level[5] = {"Title 5", 34945, 34946, 10, 0, 0,-1}
Level[levelSecret] = {"Secret Level", 78327, 78447, 4, 0, 0}

Each Level has a few settings, before entering the moves you need to set them. There is one line for each level with important informations. Let's have a look at level 1:

Level[1] = {"Title 1", 9749, 9799, 2, 0, 0,-1}

- 1 is the number of the level. It just means these are the informations for Level 1.
- Title 1 is the Title of the Level (between brackets). You can change it of course. It will appear in the Options Menu and at the beginning of the Level during Play.
- The first figure (9749) is the Frame where the level starts on the disc (intro).
- The second figure (9799) is the end of the intro of the Level on the disc. During intro, players

can skip. IF you've got no intro, just put the value of the first figure +1 (ex: 9750)

- The third figure (2) is the total number of Scenes inside the Level. Level 1 has 2 scenes.

- The fourth figure (0) is only used if you have an exact mirror (where Left is Right and Right is Left) for this Level. In that case, you don't need to setup both. The program will randomize this for you, invert Left and Right and play either the original or the mirror. For this, you just need to tell how much to add (for example if you enter 10000, then it means the mirror starts at frame 19749 on the disc).
- The fifth figure (0) is exactly the same mirror for the deaths.
- The last figure is the behavior of the program in case of death. -1 (default value) means that this level will replay until you pass it. 0 means that if you die, the level will be skipped. You'll pass to the next one and won't replay it. 1 means that you will replay the level once. In case of new death it will be skipped. Any other value will queue the level for one replay later. For example with a value of 4, the level will be reordered in 4th position and replayed once. The program will automatically reorder levels.

On the last line, i also defined a secret level. It works exactly the same than a normal level but has to be called levelSecret. It will only launch if AllowSecret = true (previous page).

Just remember that you need to put the same number of level lines that you defined for finalstage (previous page). Secret Level does not count.

Relative frames...or not

For the frames inside your levels, it is easier to use relative frames. So for example, let's say that RelativeFrames is set to true and the first move starts at frame 135. This is not frame 135 of the whole video, but frame 135 of the Level 1 video. This is why it is better to mount each level independently and then put them altogether in a single video.
But in some cases, you can't do this (for example if your video was already complete). In that case, just put RelativeFrames to false and you can enter the frames as they are on the video.

4) Moves

This is where the moves are defined. This is done within one big function, Level by Level, Scene by Scene and Move by Move:

```
function setupMoves(thisLevel, thisScene)

    if thisLevel == 1 then

        if thisScene == 1 then

            sceneStart = 101
            sceneEnd   = 370
            totalMoves   = 5

            move[1] = {135, 157, RIGHT, 1}
            move[2] = {161, 176, LEFT, 2}
            move[3] = {178, 198, BUTTON1, 1}
            move[4] = {271, 290, BUTTON1, 1}
            move[5] = {298, 318, BUTTON1, 2}

        elseif thisScene == 2 then

            sceneStart = 371
            sceneEnd   = 500
            totalMoves   = 3

            move[1] = {390, 400, LEFT, 3}
            move[2] = {442, 457, BUTTON1, 3}
            move[3] = {477, 497, ACTRIGHT, 3}

        end
```

This is a (short) Level. As you can see, it is divided in 2 Scenes. It may look like it's complicated, but there are only few things to set.

```
function setupMoves(thisLevel, thisScene)
```

Don't touch this!

```
if thisLevel == 1 then
```

Here you declare which Level it is. The first Level is obviously 1.

```
if thisScene == 1 then
```

Here you which Scene of Level 1 it is. The first Scene is obviously 1.

```
sceneStart = 101
```

This is the start of the first Scene. Normally it starts right after the intro of the level. As the length of our intro is 100 (starts at 9749 and ends at 9849) then the value is 101. Note that these are relative frames. The scene starts at frame 101 of the Level 1, which is not frame 101 of the disc.

```
sceneEnd   = 370
```

This is the end frame of the first Scene. All the moves of that Scene will be between sceneStart and SceneEnd.

```
totalMoves   = 5
```

This is the total number of moves for the Scene. Be careful to have the same number here than the number of moves you enter below, or the program will fail. There is no limit for the number of moves and each scene needs at least one move.

```
move[1] = {135, 157, RIGHT, 1, (option1), (option2)}
```

This is the first move. It contains 4 important informations:

- 135 is the frame where the move starts
- 157 is the frame where the move ends (so 22 frames long)
- RIGHT is the required move (the possible moves are presented in the appendix)
- 1 is the death played in case of failure. You can also put 0 for a random death movie.
- option 1 and option 2 are extra informations you might want to add for the move. In this demo, there are only simple moves. No extra informations are needed. In the first Appendix you will find a detailed description of the moves (and options).

So, between frames 135 and 157, an Arrow going to the right will show on the screen. The player must go to the right or it will be a failure and Death 1 will be played.

```
move[2] = {161, 176, LEFT, 2}
move[3] = {178, 198, BUTTON1, 1}
move[4] = {271, 290, BUTTON1, 1}
move[5] = {298, 318, BUTTON1, 2}
```

Then, each other move of the scene, following the same pattern. Note that it's better to have a security gap of at least 1 between moves (for example if move 2 ends at 176, then move 3 starts at 178). Also, the end of the last move must be inferior to SceneEnd.

Of course you can add many more moves. in that case you just have to add lines like this:

```
move[6] = {330, 345, UP, 2}
move[7] = {350, 362, DOWN, 1}
```

and so on...

Just remember to adjust: totalMoves = 7 or those moves will be ignored.

```
elseif thisScene == 2 then

sceneStart = 371
sceneEnd   = 500
totalMoves   = 3

move[1] = {390, 400, LEFT, 3}
move[2] = {442, 457, BUTTON1, 3}
move[3] = {477, 497, ACTRIGHT, 3}
```

Scene 2 is a lot like Scene 1. Scene 2 starts right after Scene 1 so sceneStart is sceneEnd of Scene 1 plus 1. This is not mandatory, If you put an higher value, then the program will jump to the sceneStart value for scene 2 after reaching the end of scene 1.

Of course you can add more Scenes. in that case you just have to add lines like this:

```
elseif thisScene == 3 then

sceneStart = 501
sceneEnd   = 800
totalMoves   = 2

move[1] = {590, 610, DOWN, 3}
move[2] = {630, 670, MASH, 3}
```

Remember to adjust the total number of scenes for your Level or your scene won't play.

Proceed to the end of the level. Note that normally, the end frame of your last Scene will match the end frame of the Level.

You can now add up to more Levels and a secret Level. It works exactly the same way.

```
if thisLevel == 2 then

if thisScene == 1 then

sceneStart = 2213
sceneEnd   = 2720
totalMoves = 4
```

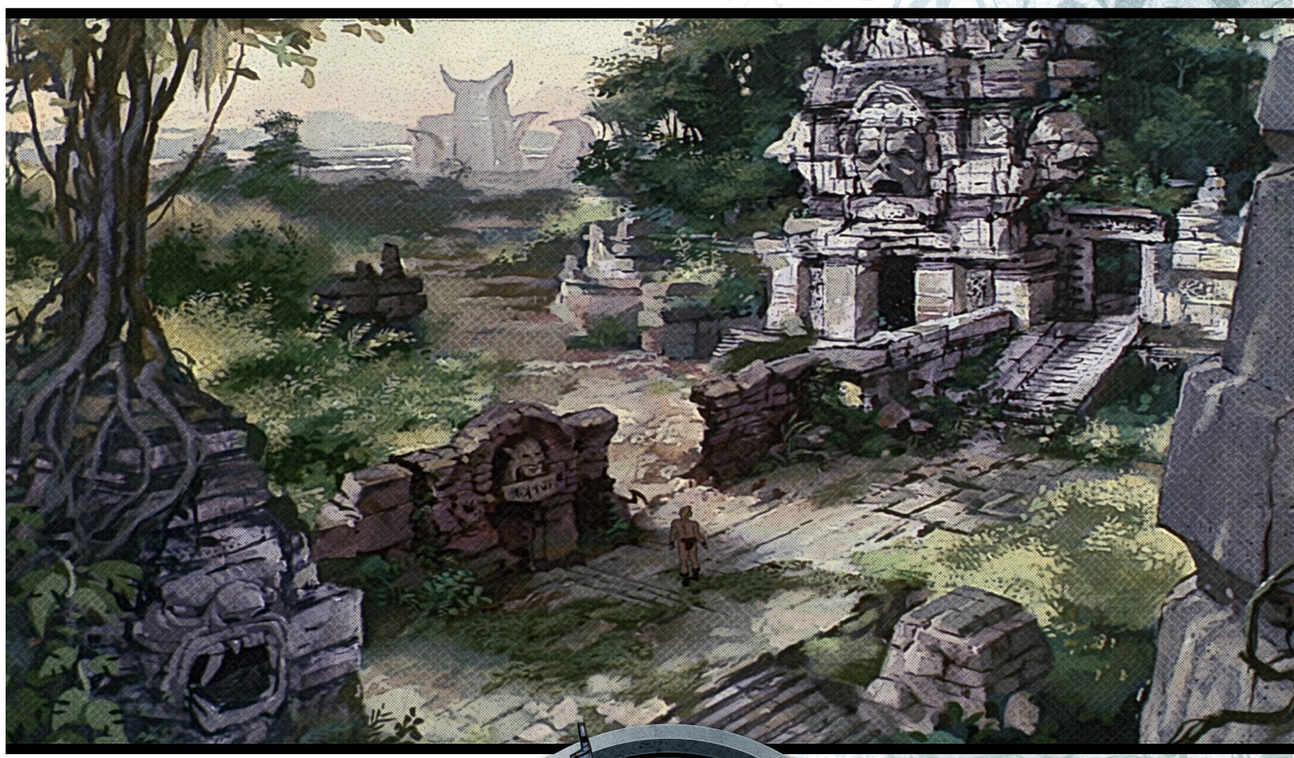
```
move[1] = {2229, 2242, DOWN, 14}
move[2] = {2305, 2318, RIGHT, 23}
move[3] = {2411, 2424, LEFT, 12}
move[4] = {2508, 2521, RIGHT, 12}
```

And so on. When you're done, the game is finished!

Step by Step

At the beginning of your game, you may ignore how many Levels you will have, and that's not a problem.
A good solution is to create Levels one by one. So just set finalstage to 1 and create a line just for Level 1. When you're done with part 4 (set the moves for Level 1 - see next page) you can then set finalstage to 2, create a line for Level 2, and set the moves. And so on...





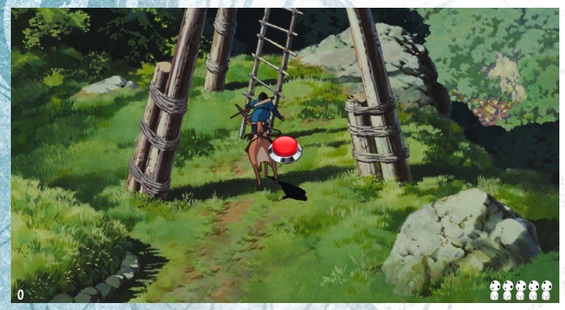


APPENDIX

MOVES

Here we will discuss about the possible moves (you can also check the table on the last page).

A) BASIC MOVES



BUTTON1, BUTTON2, BUTTON3



UP, DOWN, LEFT, RIGHT

These are the most common moves you will enter. You can also use some shortcuts with the same effect (B1, B2, B3 for Buttons, U,L,D,R for directions).

move[1] = {390, 400, BUTTON1, 1} or move[1] = {390, 400, B1, 1}

between frame 390 and 400, the player has to push BUTTON1 or it will trigger Death 1.

move[3] = {450, 470, UP, 3} or move[3] = {450, 470, U, 3}

between frame 450 and 470, the player has to go UP or it will trigger Death 3.



UPLEFT, UPRIGHT, DOWNLEFT, DOWNRIGHT

The 4 diagonals are also Basic Moves (shortcuts are UL, UD, DL, DR):

move[1] = {390, 400, UPLEFT, 1} or move[1] = {390, 400, UL, 1}

between frame 390 and 400, the player has to push UP and LEFT or it will trigger Death 1.

Options for all the Basic moves: in position 5 you can define another Basic move that will be accepted too. This is mainly used for some games where the input is not shown on screen.

move[1] = {390, 400, UL, 1, B1}

Here UPLEFT is still the good move but BUTTON1 will be accepted too.

B) COMBO



COMBO is any **combination of 2 Basic moves** (previous page) you must enter **at the same time**. Option 5 and Option 6 are **mandatory** if you use the move COMBO.

move[1] = {390, 400, COMBO, 1, BUTTON1, UP}

between frame 390 and 400, the player has to push BUTTON1 and UP or it will trigger Death 1.

Order is not relevant so you can also write it this way (same effect):

move[1] = {390, 400, COMBO, 1, UP, BUTTON1}

Any COMBO of Basic moves is possible (here a diagonal UP+LEFT and BUTTON 3):

move[1] = {390, 400, COMBO, 1, UL, B3}

Be careful because some COMBO might be impossible to do with a stick (here LEFT and RIGHT):

move[1] = {390, 400, COMBO, 1, LEFT, RIGHT}

Shortcuts for COMBO

For the most common COMBO moves there are shortcuts (mainly for retro compatibility). They work exactly the same way than a COMBO, but are quicker to write if you prefer:

ACTUP: BUTTON1 + UP
ACTDOWN : BUTTON1 + DOWN
ACTLEFT: BUTTON1 + LEFT
ACTRIGHT: BUTTON1 + RIGHT

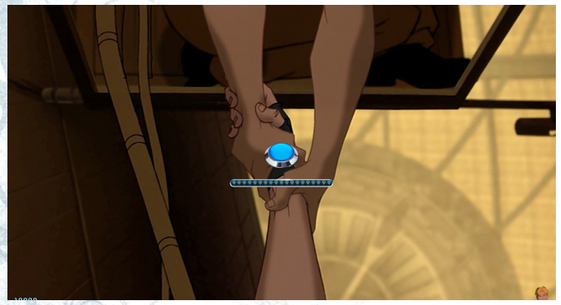
So for example:

move[1] = {390, 400, ACTUP, 1}

is exactly the same than:

move[1] = {390, 400, COMBO, 1, BUTTON1, UP}

C) MASH



MASH

MASH will require to push any Basic move many times, fast enough to fill the gauge (which decrease automatically between hits, this is the Mash Resistance).

move[1] = {390, 450, MASH, 1}

means you need to mash BUTTON1 between frames 390 and 450 until you fill the gauge.

Default is BUTTON1 but you may use option 5 to define any Basic move :

move[1] = {390, 450, MASH, 1, LEFT}

means you need to mash LEFT between frames 390 and 450 until you fill the gauge.

The number of times you need to Mash is calculated by the engine depending on several parameters (like FPS and lenght of the move), but it's better to have at least 20 frames (and up to 300).

Mash Resistance

The automatic decrease of the gauge depends on the MASH resistance parameter (in Options Menu), the difficulty, and option 6 (not mandatory):

Mash resistance is a global parameter in the Option Menu #3. Everyone is different and every button too ! So you can set it yourself to a value between 0 and 15 depending on your taste and your system (default is 5). At 0 there is no resistance at all. At 15, there will be a very strong decrease between 2 hits.

You can also adjust this on a particular MASH with option 6

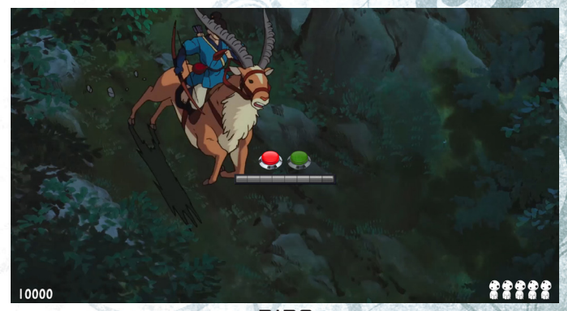
move[1] = {390, 450, MASH, 1, BUTTON2, 3} will add 3 to the resistance

move[1] = {390, 450, MASH, 1, BUTTON2, -3} will remove 3

You can set any value, but basically between 3 and -3 is OK. This is useful if you want a particular Mash to be more/less difficult.



LR and UD



B1B2

Instead of a basic move you can use other keywords for option:

LR will require to move the stick LEFT/RIGHT, fast enough to fill the gauge.

UD will require to move the stick UP/DOWN, fast enough to fill the gauge.

B1B2 will require to push BUTTON1 and BUTTON2 **alternatively**, fast enough to fill the gauge.

These moves also depend on the Mash Resistance defined in the Option Menu. Also, you can still use the option 6 to adapt the Mash Resistance.

move[1] = {390, 450, MASH, 1, LR}

means you need to mash LEFT/RIGHT between frames 390 and 450 until you fill the gauge.

move[1] = {390, 450, MASH, 1, UD, 3}

means you need to mash UP/DOWN with 3 extra resistance.

move[1] = {390, 450, MASH, 1, B1B2, -3}

means you need to mash BUTTON1/BUTTON2 with 3 less resistance.

Option Menu

In the Option Menu #3, you can decide how you want to set the Mash action done with the stick.

Good Hand means the move is made with your good hand (directional hand).

Bad Hand means the move is made with your non directional hand. The engine will ask less mash and the move will be easier.

Two Buttons means that you don't use the stick, but buttons instead (every UD and LR action is converted to a B1B2 action).

Basic Mash means that UD, LR and B1B2 are converted to a MASH action with BUTTON1

D) MULTI



MULTI is any sequence Basic moves you must enter **one after the other**. There is no limit to the size of the sequence (or maybe the size of the screen), but you must enter all the moves to validate the MULTI.

move[1] = {390, 410, MULTI, 1, "L,B1,L,L"}

between frame 390 and 410, the player has to enter the 4 moves LEFT, BUTTON1, LEFT, LEFT. As you type the moves, they will disappear from the overlay so you always know where you are.

As you can see, the sequence of moves relies on a **mandatory** option 5 that must be a string of characters (meaning between brackets: "L,B1,L,L"). The characters are intuitive:

Characters for the string (separated by , and no space)

U for **UP**	**B1** for **BUTTON1**	**UL** for **UPLEFT**
D for **DOWN**	**B2** for **BUTTON2**	**UR** for **UPRIGHT**
L for **LEFT**	**B3** for **BUTTON3**	**DL** for **DOWNLEFT**
R for **RIGHT**		**DR** for **DOWNRIGHT**

Note : if a character doesn't match this list, it will be replaced by BUTTON1 to avoid bugs.

So for example:

move[1] = {390, 500, MULTI, 1, "L,U,B1,R,D,UL,B2"}

will ask you to enter LEFT, UP, BUTTON1, RIGHT, DOWN, UPLEFT, BUTTON2 (yeah, maybe that particular sequence is not that useful but it's still possible).

Memory effect

By entering a number in Option 6, you can change the behavior of MULTI:

move[1] = {390, 450, MULTI, 1, "L,B1,L,L", 20}

The number 20 is a number of frames. It means that during the 20 first frames of the move (from 390 to 410) the overlay will show the MULTI with a "WAIT". No move can be entered.
During the remaining frames, the overlay shows "???" and the player has to enter the move.

E) LOOP



LOOP

LOOP will require the player to make a full, half or quarter circle with the stick either going to clockwise or counterclockwise. On top of this you may ask the player to push any button once the loop is achieved. This is done with a **mandatory** option 5:

Characters for the LOOP (option 5)

1) STARTING POINT: U, D, L or R

2) DIRECTION: + (clockwise) or - (counterclockwise)

3) LENGTH: Q (quarter), H (half), or a number (1 for a full turn, 2+ for several turns)

4) BUTTON (optional): B1, B2 or B3 for a button push right after the loop.

move[1] = {390, 450, LOOP, 1, "L,-,1"}

means you need to make a full circle starting and ending LEFT clockwise (so LEFT/DOWN/RIGHT/UP/LEFT).

move[1] = {390, 450, LOOP, 1, "U,+,H"}

means you need to make a half circle starting UP clockwise (so UP/RIGHT/DOWN).

move[1] = {390, 450, LOOP, 1, "U,+,H,B1"}

same loop than earlier but with an added button (so UP/RIGHT/DOWN/BUTTON1).

Memory effect

By entering a number in Option 6, you can change the behavior of MULTI:

move[1] = {390, 450, LOOP, 1, "U,+,H", 20}

The number 20 is a number of frames. It means that during the 20 first frames of the move (from 390 to 410) the overlay will show the MULTI with a "WAIT". No move can be entered.
During the remaining frames, the overlay shows "???" and the player has to enter the move.

Note: the move is 4 directional (not counting diagonals) in order to be achieved easily on keyboard. If you struggle, you can still replace it automatically in the Option Menu #3. In that case all LOOP moves will be replaced by HOLD.

F) HOLD and LETGO



HOLD

HOLD will require to push a direction and stay still until the gauge is full. Hold is adaptive but, due to the latency, you need at least 20 frames (and up to 300). HOLD works with any Basic move. This is defined in the **mandatory** option 5.

move[1] = {390, 450, HOLD, 1, LEFT}

means you need to hold LEFT between frames 390 and 450 until you fill the gauge.



LETGO

LETGO is **always** after HOLD. It will require the player to release the HOLD key quickly. LETGO is generally short (10 frames or less). LETGO will fail if you release the key before or after the instruction.

move[1] = {390, 450, HOLD, 1, LEFT}
move[2] = {452, 459, LETGO, 1}

means you need to hold LEFT between frames 390 and 450 until you fill the gauge. Then you need to release the stick between frames 452 and 459.

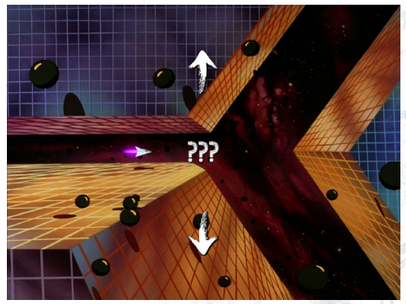
Notes: HOLD can be standalone (without LETGO). Unlike MASH, HOLD will fail if you press the directional key before the instruction.

G) SKIP

SKIP allows to skip a part of the level where nothing happens. If you've got a long part of storytelling, it's a good idea to give the players the opportunity to skip it when they play it several times. You can also add an additional on-screen text (between brackets) as option 5 :

move[1] = {390, 450, SKIP, 1, "SOME TEXT"}

H) PATH and YESNO



PATH

PATH allow to choose between several moves/scenes/levels and jump to another location. PATH action is more complicated and require a string (a bit like MULTI).

move[4] = {200, 220, PATH, 3, "U,5,D,7", 9}

The sting is filled with up to 4 moves (U/D/L/R, buttons are also possible but no diagonals) and up to 4 moves numbers. In the above example the choice is between UP and DOWN.

The 2nd value indicates where to go if the first choice (UP) is made. The 4th value indicates where to go if the second choice (DOWN) is made. Last but not least, option 6 defines the Jump Value. Jump value is the final destination for each PATH.

So, in the example above, if the player goes UP the game will go to move 5 then 6. When 6 is completed the game then jumps to move 9 (because 7 is the value for moving down) . If the player goes DOWN the game will play move 7, then 8 and then 9.

Another example with more options :

move[4] = {200, 220, PATH, 3, "L,2,D,5,B1,7,R,9", 12}

Here the choice LEFT will play moves 2/3/4 and then 12. The choice DOWN will play moves 5/6 and then 12. The choice BUTTON1 will play moves 7/8 and the 12. The choice RIGHT will play 9/10/11 and then 12.
As you can see you cannot put the moves in any order, you must follow the order of the movie. First the move leading to the closest move, and so on.

Wrong choices: you can also decides that you want one of the path to be mortal. For example, i want the UP direction to lead the player directly to death #25. In that case, i will write my pass table that way:

move[6] = {200, 220, PATH, 3, "U,1025,D,7", 9}

Entering a value above 1000 means that the path leads to death. The death scene to be played is simply determined by adding the number of the death scene to 1000. So, in this example, if the player goes UP the game will play death #25. If the player goes DOWN the game will play moves 7,8 and then 9. Of course, this can be done with any of the 4 defined moves.

PATH is really a powerful action cause you can even use it to jump between scenes/levels:

move[6] = {200, 220, PATH, 3, "U,2,D,3,R,4"}

When the Jump value is not defined (no option 6), the game will jump scenes/levels depending on the situation (if you want to prevent this you must enter a jump value, 0 will do):

- If the current scene is **not the last scene** of the level, then PATH will **jump to another scene.** In the above example, by going UP the player will go to scene 2, DOWN to scene 3 and RIGHT to scene 4.
- If the current scene is **the last scene** of the level, then PATH will **jump to another level.** In the above example, by going UP the player will go to level 2, DOWN to level 3 and RIGHT to level 4.

Note: the PATH action has a death move. It is the death scene that will be played if the player don't choose any direction during the time allowed for the move.



YESNO

YESNO is a lot like PATH in a simpler way. With YESNO you define two ways by pushing or not pushing the BUTTON1.

move[3] = {150, 170, RIGHT, 3}
move[4] = {200, 220, YESNO, 3, "5,7", 9}
and so on...

As you can see there are only 3 figures here. The first value (5) defines where to go when the button is pressed. The 2nd value (7) where to go if nothing is done. The option 6 is the jump value (9) So, in this example, if the player press BUTTON1 the game will play moves 5 and 6 and then jump to move 9. If the player don't press, the game will play moves 7,8 and then 9. If there is no jump value, then the game will jump scenes or level (see PATH).

H) WAY, TOSCENE, TOLEVEL

WAY is a simple "placeholder" move.

move[3] = {150, 170, RIGHT, 3}
move[4] = {200, 220, WAY, 0}
and so on...

Sometimes (most of the time right after a PATH), you'll need to show the result of the move. This is not really an action, but this is to short for a SKIP. In that case you'll use WAY which is neutral and does not score. But sometime, this move you want to show is not directly connected to the other moves, so you need to "jump back on track". In that case, change the **death value** from 0 (default) **to a number**, and after the WAY action, you jump to **that move**.

TOSCENE is like WAY but allows to **jump to another Scene** inside the Level.

TOLEVEL is like WAY but allows to **exit the Level** even if you're **not on the last Scene**.

For **TOSCENE** and **TOLEVEL**, the **death value** indicates **where to jump**.

I) CHOOSE



CHOOSE

CHOOSE will require to choose between 2 to 4 options. You move between options with UP and DOWN arrows and select with ACTION. One choice is good. Other choices are bad and lead to death. This move requires a long time (at least 200 frames to read and choose) and special lines:

move[5] = {794, 1023, CHOOSE, 3}
choice[1] = {«GO UP», true, 0}
choice[2] = {«GO DOWN», false, 24}
choice[3] = {«SHOOT THE MONSTER», false, 23}

Here, the move 5 is CHOOSE. The death number following CHOOSE (here 3) sets the number of choices (2 to 4). After this, you need to insert lines. For each line enter the text between quotation marks, then true for the good answer (here choice 1) and false for the wrong answers. The last figure is the death played in case of wrong choice (irrelevant for true).

Note: the choices are randomized on-screen. The good choice can appear in any line.

J) TIMED



TIMED

TIMED is a move used when you want a really precise timing. For example, crossing a bridge above a lava flow. You want to cross when the lava is down and not up. But most of the time, the amount of frames is really low, so just showing an arrow at the right time is not enough cause players won't react fast enough. Timed action is the solution:

```
move[4] = {12350, 12414, TIMED, 81}
timed[4] = {LEFT,12366,12371,82}
```

Here, the move 4 is TIMED. The death number following TIMED (here 81) is the death played if the player does nothing. Timed uses a special table where the number must match the number of the move which called it:

```
timed[4] = {LEFT,12366,12371,82}
```

LEFT is the accepted move. Any other move will play DEATH 82. 12366 and 12371 are the "accepted" frames for the move. So during move 4, if you press LEFT at the right time, you'll pass the bridge.

A bit more complex case: there are several chances to cross the bridge. It would be possible to make several short TIMED actions, but if the time is too short, you can gather several "good gaps" within the same action:

```
move[4] = {12350, 12414, TIMED, 81, 1, 3}
timed[1] = {LEFT,12366,12371,82}
timed[2] = {LEFT,12388,12393,82}
timed[3] = {LEFT,12410,12414,82}
```

For the TIMED action, we add 2 values after the death move. They will define which part of the timed table is concerned by this move (timed[1] to timed[3]). The whole sequence runs from frames 12350 to 14414 (64 frames) and there are 3 small gaps inside this sequence where the lava is down. Each timed line is one of those gaps.

Last thing. In the last example, let's say i cross the bridge by pushing LEFT at frame 12368. This is the first defined gap, so it will work. But defined like this, i'll have to wait for frame 12414 (end of the move) to see the result of that action and go to move 5. Since i used the first try i would prefer to see it immediately. It is possible:

```
timed[1] = {LEFT,12366,12371,82, 5}
timed[2] = {LEFT,12388,12393,82, 5}
timed[3] = {LEFT,12410,12414,82}
```

This new value inserted at the end of the table tells the program to jump to move 5. This is the same kind of "jump" used in the PATH action. So here if i press LEFT at frame 12368, the program jumps directly to the frame 12414 and i cross directly. Note that, i did not inserted a value for timed[3] cause it's close enough to the actual end of the move and in this case, i let it finish naturally.

K) ANYTHING, NOTHING



ANYTHING



NOTHING

These very special moves will ask the player to hit Anything or to hit Nothing during the defined time for the move. Any button or direction will succeed the move ANYTHING and will kill the player for the Move NOTHING.

L) Optional Moves (scoring or not)

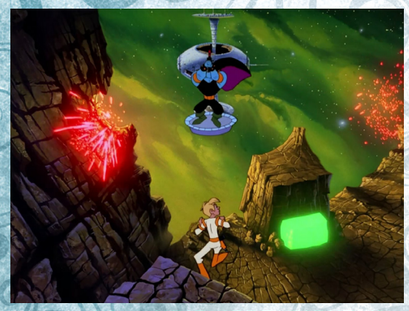


Any move can be turned as Optional easily. It means that this move will be shown on-screen but not mandatory (not causing death). For any move, if **Death is set to -1**, then this move is now **optional** and will not score. For example, it is used at the beginning of Dragon's Lair, where you can head for the Drawbridge.

If **Death is set to -2**, then this move is **optional and will score anyway**. For example it is used in Dragon's Lair elevator scene when you wait before leaving the elevator.

IN DEPTH EXAMPLE (WITH PATH, TIMED and WAY)

Let's face it. PATH and TIMED are quite complex moves and you may even ask yourself why they are here. So, here's a small illustrated sequence from Space Ace Level 1, Scene 1 which use both. The moves 1 to 4 are very basic so i won't comment them. Move 5 is the PATH.



move[1] = {1310, 1320, RIGHT, 1}



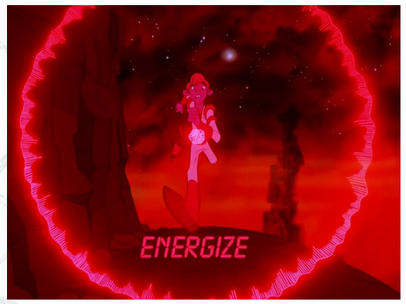
move[2] = {1347, 1358, RIGHT, 1}



move[3] = {1438, 1453, LEFT, 1}



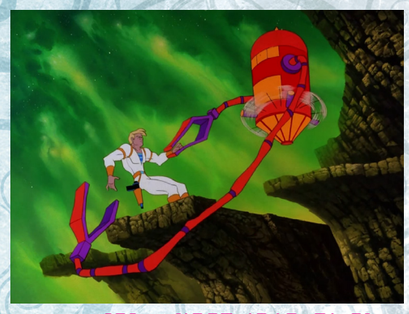
move[4] = {1492, 1506, LEFT, 1}



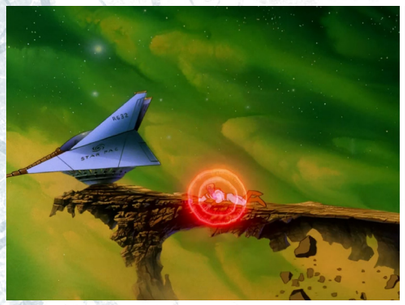
move[5] = {1527, 1550, PATH, 5,"B1,6,D,13", 19}

The options for the PATH are "B1,6,D,13" and 19 (Jump Value), what does it means ? If the player pushes BUTTON1 (B1) he will turn into Ace and the **Ace sequence (moves 6 to 12)** will be played, then the game will jump to move 19 (end of the Level). If the player goes DOWN, he will play as Dex and the **Dex sequence (moves 13 to 18)** will be played, then the game will jump to 19 (end of the level).

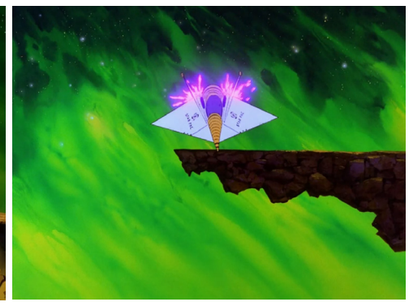
Ace sequence is quite basic so i won't comment (below you can see the first and last move).



move[6] = {1627, 1640, B1, 2}



move[12] = {1899, 1913, LEFT, 6}



move[19] = {2251, 2309, WAY, 0}

On move 12 Ace turns back into Dex, and must enter the ship. Then we jump to move 19 which is a WAY. Why ? For the story. WAY is only showing the ship's departure, but this is a more logical end for the level. This is the last move, so after WAY the game goes to level 2.



move[13] = {1914, 1954, WAY, 0}

Dex sequence is a bit more complex. It begins with a WAY. Once again, this move is only here for narrative reasons: we just show the result of going DOWN on the PATH action and fill the gap until the next moves, explaining what happens next.

The moves 14 to 17 are TIMED and 17 is special (2 chances to cross). As those 4 moves are timed there are timed tables linked to each move (14 to 16 for moves 14 to 16, 17 and 18 for the move 17)



move[14] = {1977, 2007, TIMED, 3}



move[15] = {2029, 2059, TIMED, 3}



move[16] = {2083, 2114, TIMED, 3}



move[17] = {2147, 2207, TIMED, 3, 17, 18}

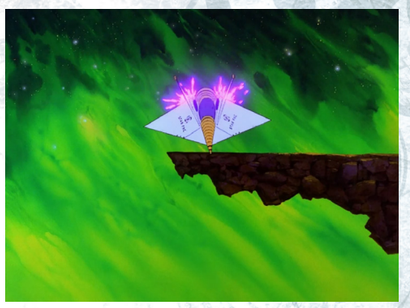
They all work the same way,asking the player to go LEFT during a precise gap.

timed[14] = {LEFT,1995,2007,4} for move 14
timed[15] = {LEFT,2047,2059,4} for move 15
timed[16] = {LEFT,2101,2114,4} for move 16
timed[17] = {LEFT,2157,2170,4,18} for 17, goes to move 18
timed[18] = {LEFT,2191,2207,4} for move 17 2nd chance

Why TIMED and not just LEFT ? Because the timing is quite short and if you show the arrow during this timing he would be very off-beat. It is also very important for move 17 because if the player takes the first chance to go left, the timed action allows to jump to move 18 without showing another cycle of the machine (timed[18] will be skipped).



move[18] = {2208, 2250, WAY, 0}



move[19] = {2251, 2309, WAY, 0}

Dex sequence ends with another narrative WAY (going to ship) and then level ends (ship departure).

MAIN OPTIONS MENU



In the Options Menu (key 9), you can set different parameters for your game. Change the line with UP/DOWN. Change the options with LEFT/RIGHT or Action. Exit with Exit or key 1.

- Game type: Normal/Percent/Life Bar/Kid/Movie

Note : in Kid mode, you just have to push Action at the right moment. Movie allows you to watch the movie without playing. You can use arrows to scroll chapters and Start or Esc to exit.

- Difficulty: Easy/Normal/Hard /Extreme/In-Game* (* appears if defined in the Singe file)

Note : in Easy difficulty, there is only Basic move and Diagonals. All other moves are simplified.

- Play Style: Normal/Random*/Tiers*/Map*/Select (* appears if defined in the Singe file)

- Start Level: Goes from 1 to finalstage

- Start Scene: depends on the Level. Goes from 1 to total scenes (only in Normal/Life Bar mode)

- Death Rewind: Scene/Level/Last Move/Next Move. Defines the behavior of the game when you die (where you reset). Scene is default and is forced for Life Bar mode. (not used in Percent mode)

- Lives per Credit: 1 to 5 (not used in Percent and Life Bar modes).

- Allow Continue: None/1/2/3/Unlimited

- Show Moves: Nothing/Sound/Guess/Cliff/Normal

Nothing is mainly for games with clues included in the video. Guess is a visual clue (???) with no indication about the move. Cliff is an advanced clue (Stick/Action/Special).

- Show Hints : Yes or No. Suggested if you don't want to show moves.

- Show Next Input : Yes or No. Will show the next input (not the move) on the downright corner.

- Tilt sensitivity : from NO TILT to EXTREME (see later).

Note: there are 3 pages in the menu. Page 2 allows to tweak the graphical options of the game like skins, font, positions of score. Page 3 allows to tweak things that may change performances, like resolution of the overlay, or the behavior of some actions.

PIMP YOUR GAME

The blank directory is delivered with graphics, sounds, font. But you might want to change those to adapt them to your game (or simply, you can change the skin and colors in OptionMenu #2).

- Graphics are located in the Overlay directory. You can replace them with other png files.



- Sounds are located in the Sounds directory. You can replace them with wav files (32 bits).

- Font is located in the Fonts directory (font.ttf as main and fontlo.ttf for Low res if needed). You can replace with any ttf file. If your font is too big or too small, you can adjust its size in the second page of the Options Menu.

IMPORTANT: For the Singe Overlay, pure black (#000000) is considered transparent. So if one of your assets contains that color it will not appear on-screen. You can still change that color to something close to pure black (like #090000) if you want it to appear black.

CONTROLS

Controls are located in the Controls.cfg file. You may edit this file with a notepad and change it.

```
DEAD_ZONE = 12000

INPUT_UP          = { SCANCODE.UP, GAMEPAD_0.DPAD_UP, GAMEPAD_0.AXIS_LEFT_Y_U }
INPUT_LEFT        = { SCANCODE.LEFT, GAMEPAD_0.DPAD_LEFT, GAMEPAD_0.AXIS_LEFT_X_L }
INPUT_DOWN        = { SCANCODE.DOWN, GAMEPAD_0.DPAD_DOWN, GAMEPAD_0.AXIS_LEFT_Y_D }
INPUT_RIGHT       = { SCANCODE.RIGHT, GAMEPAD_0.DPAD_RIGHT, GAMEPAD_0.AXIS_LEFT_X_R }
INPUT_1P_COIN     = { SCANCODE.MAIN_5, GAMEPAD_0.BUTTON_BACK }
INPUT_2P_COIN     = { SCANCODE.MAIN_6 }
INPUT_1P_START    = { SCANCODE.MAIN_1, GAMEPAD_0.BUTTON_START }
INPUT_2P_START    = { SCANCODE.MAIN_2 }
INPUT_ACTION_1    = { SCANCODE.LCTRL, SCANCODE.RETURN, GAMEPAD_0.BUTTON_A }
INPUT_ACTION_2    = { SCANCODE.LALT, GAMEPAD_0.BUTTON_B }
INPUT_ACTION_3    = { SCANCODE.LSHIFT }
INPUT_ACTION_4    = { SCANCODE.RSHIFT }
INPUT_SKILL_EASY  = { SCANCODE.F1, GAMEPAD_0.BUTTON_LEFT_BUMPER }
INPUT_SKILL_MEDIUM = { SCANCODE.F2, GAMEPAD_0.BUTTON_RIGHT_BUMPER }
INPUT_SKILL_HARD  = { SCANCODE.F3, GAMEPAD_0.AXIS_LEFT_TRIGGER_P }
INPUT_SERVICE     = { SCANCODE.MAIN_9, GAMEPAD_0.BUTTON_RIGHT_STICK }
INPUT_TEST_MODE   = { SCANCODE.F4, GAMEPAD_0.BUTTON_X }
INPUT_RESET_CPU   = { SCANCODE.F10 }
INPUT_SCREENSHOT  = { SCANCODE.F12 }
INPUT_QUIT        = { SCANCODE.GRAVE }
INPUT_PAUSE       = { SCANCODE.P, GAMEPAD_0.BUTTON_Y }
INPUT_CONSOLE     = { SCANCODE.ESCAPE, SCANCODE.Q }
INPUT_TILT        = { SCANCODE.T }
INPUT_GRAB        = { SCANCODE.G }
```

Here is my default control files (some of the inputs are not used). For example, you can see that Input_Up is assigned to both arrow UP on the keyboard (SCANCODE.UP), the Dpad (GAMEPAD_0.DPAD_UP) and the left stick (GAMEPAD_0.AXIS_LEFT_Y_U). Unless you've got some really specific designs, this control file will work without any change.

LEVEL SELECT SCREEN





The v4 engine now includes an automated level selection screen. Automated because you don't need to do anything to create this screen (but it is possible to pimp it).

There are 2 ways to access the screen :

- Automatic with some gaming mode (Percent and Kid)
- Option Menu #1 in the Play Style line

The frame shown by the engine is by default the 100th frame of each level, But you can change any or all of those frames by indicating another frame at the end of each level:

Level[1] = {"Title 1", 9749, 9799, 2, 0, 0,-1, 10249}

For level 1, the frame shown would normally be 9849 (9749+100), but as there is another frame indicated (10249) it will be shown instead.

NEXT INPUT





In the Option Menu #1, you can decide if you want to show the Next Input that the game will ask you (something inspired by Tetris). With this option, the game will be way easier to play.

The Next Input is shown on the lower right corner of the screen. For the most complex moves (like PATH or CHOOSE) nothing is indicated. Please note that Next Input doesn't mean Next move. For example, for a BUTTON1 MASH you will see BUTTON1, but not the MASH.

TILT!





Obviously inspired by pinball, this feature will allow you to make the game even more difficult (or balanced).

You can set the Tilt sensitivity in Option Menu #1 (NO TILT, LOW, MEDIUM, HIGH or EXTREME). No Tilt is the old behavior of the game. Other settings check what you do in between moves and react accordingly. Of course Low is a forgiving setting and Extreme is not.

In most games (with no tilt) you can prepare the moves in advance. This is a huge advantage, but a bit unfair. On the other hand, sanctioning early moves with instant death would be over the top. So, if you enter the move too early, a (secret) gauge will progress. At some point (depending on sensitivity setting) it will send you a warning on the top of the screen (left image). If you keep entering early moves you will Tilt the game (right image)! At some point (depending on the mode), the warning may disappear.

Tilt behavior depends on the Gaming mode:

- Normal and Survivor: the warning will reset after each scene. Tilt is immediate Death.
- Life Bar: then warning will reset after several good moves. Tilt lowers the Bar
- Percent: when you tilt you don't die but you will lose 5% for each tilt.

Note: some moves like LETGO, PATH or TIMED are not checked.



GAMING MODE: PERCENT



Percent is a new gaming mode. It can be launched by setting the game type to "Percent" from the Options menu. In this new mode only one level is played (through Level Selection Screen).

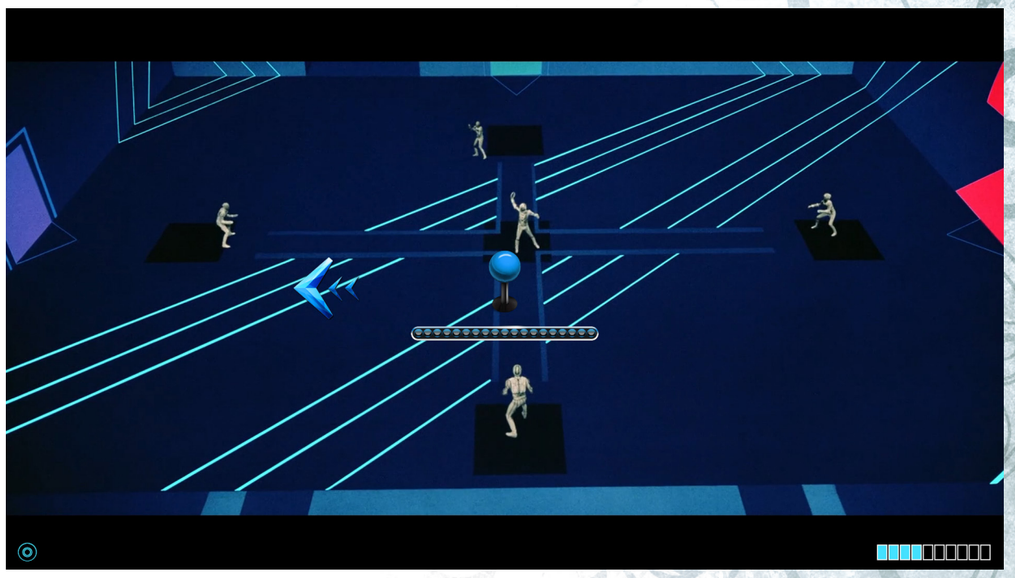
The only difference with the Normal game is that the player don't die if they make a mistake. Instead, they will just hear an "error" sound and the game will continue normally until the end of the level.

At the end, the game will show a result screen with the percentage of good moves with the possibility to enter a High Score (there is one High Score per level). You may then Quit (Button 1), Retry (Button 2) or go to another level (Button 3). In this mode, saves are disabled.

Percent mode is a really good way to discover a new game and also a good trainer. Try it!



GAMING MODE: LIFE BAR



Life Bar is a new gaming mode. It can be launched by setting the game type to "Life Bar" from the Options menu. In this new mode, failures are not mortal. The game behave like a regular "hit points" video game.

The only difference with the Normal game is that the player don't die if they make a mistake. Instead, they will just hear an "error" sound and lose life slots in the bar. The game will continue normally until the life bar is empty. They can also gain life slots during the game, if they complete a few good moves.

The Life Bar settings are located in the Script/Addons.singe (only if you want to change them).

```
BarSize = 10
BarMin = 2
BarBonus = 3
```

- BarSize defines the size of the bar (here 10 slots is default. Don't make it too long)
- BarMin defines the number of emptied slots when you make a mistake.
- BarBonus defines the number of good moves you need to make in order to have one slot back.

If BarMin value = 0, then you cannot die. If BarBonus value is 0, then you never gain life slots.

Note: those values will change with difficulty. BarBonus goes -1 in Easy (easier to win some life) and +1 in Extreme. BarMin goes +1 in Hard and Extreme (player will lose more for each failure).

Life Bar mode is an in-between Percent and Normal modes. With the Life Bar mode, you will play like a regular video game, so it's less "Die and Retry" than the Normal mode. Some will like it, others not, but you may decide.

GAMING MODE: SURVIVOR

Survivor is a new gaming mode. It can be launched by setting the game type to "Survivor" from the Options menu. In this new mode, each failure is mortal. In other terms, you've got only one life, no continue and no save.

The game starts from level 1 and follows the regular play order.

Survivor mode has a dedicated high score system. So, even if you don't finish the game, you may enter your name with a good performance. Of course, Saves are disabled.

This mode is of course very tough! Some will like it, others not, but here again you may decide.



GAMING MODE: DIE & RETRY

Die & Retry is the last gaming mode. It can be launched by setting the game type to "Die and Retry" from the Options menu. In this mode, you will have unlimited lives to complete the game. Each Death is recorded and your final score is the number of death.

Of course, to achieve a High Score, you need the lowest possible score. There are only 4 champions in the Die and Retry mode: One for each difficulty.

ADVANCED CONFIGURATION (ADDONS.SINGE)

If your game has special rules, there is a special place where you can change things. In the Script directory, you will find addons.singe. This file contains both advanced configurations and empty functions which are called by the program. By entering code in those functions, you can change a few things like scoring, levels or even advanced things like object management in DL2e. Of course this is quite advanced so i advise you to check the various examples if you want to follow that path. The following pages will present you some possibilities.

Most configuration entries are pretty straightforward. So, i will only comment the tricky ones only:

```
miscR = 0
miscG = 225
miscB = 255
```

If you choose MISC for the text in the option menu, you can create your own text color. In that case fill miscR, miscG and miscB with the RGB values of your color (0 to 255).

```
AllowRandom = false
```

True or false. If true, you will be able to select Random Level order in the Option Menu.

```
AllowMap = false
MapStart = 0
```

AllowMap: true or false. If true, you will be able to select map order in the Option Menu. Map order uses a video map to advance through the adventure. It's an advanced feature and will require extra programming (check Sucker Punch or Chantze's Stone for examples).

```
LevelTrophy3 = 0
LevelTrophy4 = 0
```

The script manages a simple trophy system. The first 2 trophies are awarded automatically, for the last 2, you can define a level to complete without dying here. If you let 0, you won't use trophies in your game. If you put a value, you will need a single frame for trophies.

```
PenalNormal = 0
PenalHard = 2
PenalExtreme  = 4
```

In most cases you don't want to change this. These values are the number of input frames you remove for each difficulty. For example if a move is planned between frames 10 and 30, Easy and Normal difficulty won't be affected. On Hard difficulty the move will be planned between frame 12 and 30. On Extreme difficulty, between 14 and 30.

ADVANCED STUFF: TIERS

```
AllowTier = true
Tiers[0] = {3,2,9,9}
Tiers[1] = {1,2}
Tiers[2] = {3,4,5,6,7,8,9,10,11}
Tiers[3] = {12,13,14,15,16,17,18,19}
```

Tiered order is a kind of «semi-random» mode used in Dragon's Lair. It allows you play the levels randomly by groups. Tiers[0] is the configuration. The first number (3) is the number of Tiers, the following numbers (2, 9 and 9) are the size of each Tier. There are no limitations for the number of Tiers or the number of levels inside a tier.

Here the program will play both level 1 and 2 in a random order, then pass to group 2, and play levels from 3 to 11 randomly. Then it will go to group 3 and play from 12 to 19 randomly.

The last level of the game is always played last and is not defined in the Tiers. Here i've got in fact 20 levels. Level 20 will be played at the end and is not defined.

ADVANCED STUFF: BAT FILE

```
Start Singe.exe -k -z -d Blank\Index -v Blank\Movie\Blank.mp4 -w Blank\Blank.singe
```

In the bat file, you can add some options. Here they are:

```
-a, --aspect=N:D              force aspect ratio
-c, --showcalculated          show calculated framefile values for debugging
-d, --datadir=PATHNAME        alternate location for written files
-e, --volume_nonvldp=PERCENT  specify sound effects volume in percent
-f, --fullscreen              run in full screen mode
-h, --help                    this display
-k, --nologos                 kill the splash screens
-l, --volume_vldp=PERCENT     specify laserdisc volume in percent
-m, --nomouse                 disable mouse
-n, --nocrosshair             request game not display gun crosshairs
-s, --nosound, --mutesound    mutes all sound
-t, --trace                   trace script execution to screen and file
-u, --stretch                 use ugly stretched video
-v, --framefile=FILENAME      use an alternate video file
-w, --fullscreen_window       run in windowed full screen mode
-x, --xresolution=VALUE       specify horizontal resolution
-y, --yresolution=VALUE       specify vertical resolution
-z, --noconsole               zero console output
```

for example -e 50 will play the game sounds at 50% volume.

ADVANCED STUFF: DIFFICULTY VIDEO



frameEasy



frameNormal



frameHard



frameExtreme

Ok. Maybe not so «advanced». In order to create a «video» for choosing difficulty in-game you need to create four frames (one for each difficulty) and insert them in your menu movie.
Then you need to fill those lines in the addons.singe file:

```
IngameDiffchoice = true
frameEasy = offsetMenus +18
frameNormal = offsetMenus +21
frameDifficult = offsetMenus +24
frameExtreme = offsetMenus +27
```

Last thing: go to the Options Menu and for the second item (Difficulty), choose In-Game.

Now, when you will launch a game, the difficulty video will play and you can change it with RIGHT and LEFT and choose with ACTION.

ADVANCED STUFF: DEFAULT OPTIONS

In the Options menu there is a Default tab which will reset all the options to your preferred state.
You can change the default configuration very quickly:
- Go to the options Menu and set the configuration as you wish, then exit.
- Go to the Cfg directory and copy game.cfg.
- Paste it in the same directory.
- Rename it default.cfg and you're done!

ADD YOUR GAME TO THE MENU



Singe has now a front-end. It's a kind of directory from which you can launch all games of the library. Launch Frontend.bat if you want to try it (or Frontend43.bat for a 4/3 version).

In order to add your game to this front-end you need a few files:



a marquee (size 827x229)



a box (445 max height)



a short presentation video (preferably 4:3, like 320x240) no need to make it too big or long

Some mock ups for those files are provided in the Blank directory. Look into the Menu sub directory and you'll find all the files you need. The easiest way is to overwrite those files with your content. Then you just need to edit the file frontend.dat. This file contains the informations about your game for the front-end.

```
GAMES = {
        {
                TITLE       = "Title or your Game",
                SCRIPT      = "Blank/myscript.singe",
                VIDEO       = "Blank/Movie/myvideo.mp4",
                SCRIPT2     = "",
                VIDEO2      = "",
                NAME        = "",
                NAME2       = "",
                DATA        = "Name of the data directory",          --
        STRETCH     = false,
START = 1,
                ATTRACT_END   = 500,
                SKIN          = "BLUE",
                YEAR        = "1983/2020",
                DIFFICULTY    = "4/5",
                DEVELOPER     = "You",
                PUBLISHER     = "You",
                GENRE         = "QTE",
                RESOLUTION    = "1920x1080",
                DESCRIPTION   = "Description of your game"
        }
}
```

-TITLE is used for sorting the games in alphabetical order.
- Leave STRETCH and SINDEN_GUN like this.
-Set NO_MOUSE to false for gun games.
- RESOLUTION will define the size of the video if the menu is in Windowed mode.
- POSTER MARQUEE and ATTRACT show the path to the files needed by the front-end.
- ATTRACT_START and ATTRACT_END are the frames of your video. Here the video menu-attract.mp4 will be played from frame 1 to frame 500 and then loop.
- SKIN is the color of the arcade ("GREEN", "RED", "BLUE", "YELLOW", "PURPLE", "ORANGE", "BLACK").
- GENRE is just an info, but if you name is "SHOOTER", a gun will appear in the menu.
- The remaining lines are just textual informations about your game. Description can be quite long. If it's too long (more than 7 lines on screen), the front end will cut it, and you will have to go down with the stick to see the following lines.
Sometimes, you'll need the menu to launch 2 different scripts (2 different versions of the game) or 2 different video (for example video in 4/3 and 16/9). In that case you need to fill SCRIPT2 or VIDEO2 and always NAME and NAME2 (a title for each version). If you do so, both versions will appear on screen with the possibility to push button 1 or button 2 to launch them.

GAMES



FIRE & ICE

First release with v1.0 (2019)
From the Bashki/Frazetta masterpiece.
2 roms available: Original an Enhanced (all the new moves).
Favorite scene: swordfight between Larn and Nekron.



TITAN A.E.

Second Game (2019)
Homage to the legendary Don Bluth .
Introduces diagonal moves, map and saves to the code.
Favorite scene: like them all maybe level 6 ? Or 2 ?



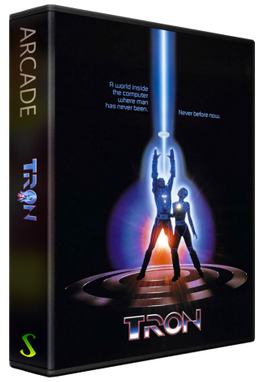
CLIFF HANGER HD

First Singe remaster (2019) from the Stern Arcade (1983)
Introduces button 2, hints and Cliff style moves.
With 2 extra levels explaining the story a lot better.
Favorite scene: Scuba ?



TIMEGAL HD

Heavily remastered (2020) from the Taito arcade (1985)
Introduces mirror for scenes and death, CHOOSE, Get Ready, Extra death, Tiers...
Favorite scene: Too short to choose !



TRON

Fifth Game (2021)
The iconic movie about video games turned into a video game.
Introduces Lifebar mode.
Favorite scene: Light Cycles and Disc fights (of course).



DL2e: TIMEWARP

Full HD port of the enhanced Rom with clever objects management (2021).
DL3 included as a secret level.

Favorite scene: Wonderland



DRAGON'S LAIR ENHANCED

Full HD port of the enhanced Rom of the godfather (2021).
The best way to play the game as originally intended.
Introduces TIMED, WAY, WAY-OUT.
Favorite scene: Whole Game !



SPACE ACE ENHANCED

Full HD port of the enhanced Rom (2021) with mirror, alt moves and clever difficulty levels.
Introduces PATH and YESNO.

Favorite scene: Whole Game !



PRINCESS MONONOKE

Third Game (2020)
From the Miyazaki masterpiece.
Introduces HOLD and LETGO moves.
Favorite scene: everything with archery.



ASTERIX

port of a Singe 1 game by Nado74 (2020)

Classical French Hero.

Favorite scene: Camp (level2)



SUCKER PUNCH

Fourth Game (2020)
Zak Snider goes wild !
Introduces DOUBLE, Percent, swapscene (difficulty).
Available in 16:9 and 2.35.
Favorite scene: Temple and Train !



ALTERED CARBON RESLEEVED

Collaboration with Mazinger-4Life (2021)

A lot, lot of fun action !

Favorite scene: Ninjas ! (level2)



CHANTZE'S STONE HD

Full port of the long lost game based on the work of Poiu (2021).
Includes 2 roms: Chantze and Triad Stone.
Favorite scene: Tower of Napier



NINJA HAYATE HD

Full restoration of the Taito arcade based on the work of Poiu (2021).
Includes 2 roms: Arcade and PSX.
Introduces Survivor Game.
Favorite scene: Final battle.



THE SECRET OF NIMH

Another Don Bluth masterpiece turned into game (2023).

Introduces MULTI and LOOP

Favorite scene: Sword fight



BRAIN DEAD 13

A Die & Retry Game which is a real test for the PATH action !
Released along with v4.

Introduces all the v4 novelties.

Favorite Scene : Vivi of course.

MOVES

Moves have a fixed structure with 4 mandatory informations: Start Frame of the move, End Frame of the move, TYPE of the move and Death played in case of error. There are 2 more spaces for Options. Depending on the TYPE of the move these options might be mandatory or optional. The table below is a summary of the Moves Appendix for quick reference.

move[number] = {Start Frame, End Frame, TYPE, Death, Option#1, Option#2}

TYPE	Description	Option#1 (position 5)	Option#2 (position 6) or Note
UP, DOWN, LEFT, RIGHT	Basic Moves (directional)	Any other Basic Move accepted as a good move.	For the lazy: **U, D, L, R also works** for type and options.
BUTTON1, BUTTON2, BUTTON3	Basic Move (buttons)	Any other Basic Move accepted as a good move.	For the lazy: **B1, B2, B3 also works** for type and options.
UPLEFT, UPRIGHT, DOWNLEFT, DOWNRIGHT	Basic Move (diagonals)	Any other Basic Move accepted as a good move.	For the lazy: **UL, UR, DL, DR** also works for type and options.
ANYTHING, NOTHING	Do Any Action / No action	**ANYTHING** will be a success if you use the stick or any button.	**NOTHING** will be a success if you don't touch the stick or any button.
COMBO	Any 2 Basic Moves at the same time	Basic Move #1	Basic Move #2
MASH	Fill the gauge by mashing any Basic Move.	Basic Move (if nothing BUTTON1 as default) **or** LR, UD, B1B2 for a double MASH	Any number to increase or decrease (-) the resistance. -3/3 suggested max.
MULTI	Any number of Basic Moves in sequence	A string ("") with U, L, D, R, B1, B2, B3 UL, UR, DL, DR (separated by , no space)	A number of frames (memory). Will show ??? after those frames are played.
LOOP	Any circle or Half Circle with the stick (any direction)	A string ("") with: Start (U/L/D/R), Dir(+/-), # of turns (Q,H,#), Button.	A number of frames (memory). Will show ??? after those frames are played.
HOLD	Any BM. Hold until you fill the gauge	Basic move	Move fails if started before overlay.
LETGO	Release the Hold at a precise moment.	No options. Used in Monoke to release arrows (Hold is the Bow).	**Always after HOLD**. Will fail if nothing is held when LETGO begins.
PATH	Select a direction	A string ("") with U, L, D, R, B1, B2, B3 (separated by , no space)	A move where the Path ends or nothing to jump between scenes/levels.
YESNO	Push Button 1 or not with one path for each	A string ("") with 2 moves for pushing the button or not (separated by , no space)	A move where the Path ends or nothing to jump between scenes/levels.
CHOOSE	Choose an answer (over 4) to a question.	Used in Timegal	Note: you must create a **choice table** with all the choices.
TIMED	Veru short action timed within a move.	Check the manual for options.	Note: you must create a **timed table** with the good timings.
WAY	Placeholder for short video sequence	**Death Value is the Option**. A death value of 0 means next Move	If any other number then **jump directly to that Move** in the scene.
TOSCENE	Placeholder for short video (then jumps to Scene)	**Death Value is the Option**. A death value of 0 means next Scene	If any other number then **jump directly to that Scene**
TOLEVEL	Placeholder for short video (then jumps to Level)	**Death value is the Option**. A death value of 0 means next Level	If any other number then **jump directly to that Level**
SKIP	Skip a part of the video	An extra text (between "") will be printed on screen	Death has no effect.

Mandatory Optional Special (RTFM!) Note

Note: for any move except SKIP, WAY, WAYOUT and TOLEVEL if Death is set to 0 then Death movie is random. If Death is set to -1, the move is optional and will not score. If Death is set to -2, then the move is optional and will score anyway.