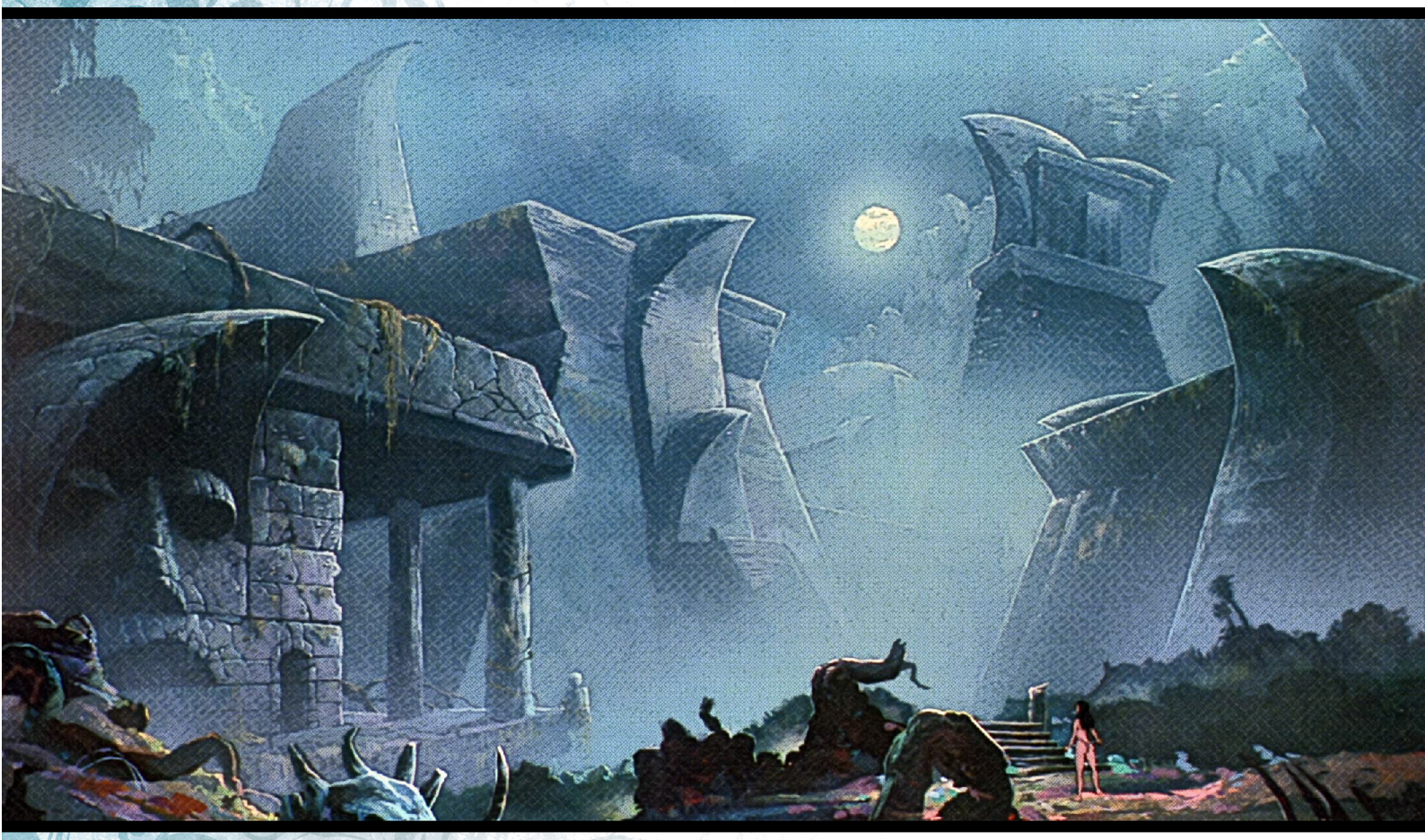


LUA SINGE ENGINE

CREATE YOUR OWN FMV GAME

KARIS 2021 - V3.32

Welcome!

This tutorial will show you how to create your own FMV game using SINGE 2.0. It is really simple and quick. If you've got a simple 10 minutes movie ready, 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 my SINGE script.

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.

In 2020, Scott decided to give a new life to Singe and created **SINGE 2.0**. This new installment is no more a dll, so it doesn't need the good old Daphne to run. SINGE 2.0 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...

LUA SINGE ENGINE

LUA SINGE 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 2.0 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 4 Action buttons,
- Combined actions for button 1 : you can ask the player to combine an arrow plus button 1
- Special actions MASH, HOLD, LETGO, DOUBLE, RUN
- 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 Double 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, Random, Tiers, Map, Percent, Life Bar)
- Selectable gaming 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
- Movie Mode : just watch the movie without playing
- 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!

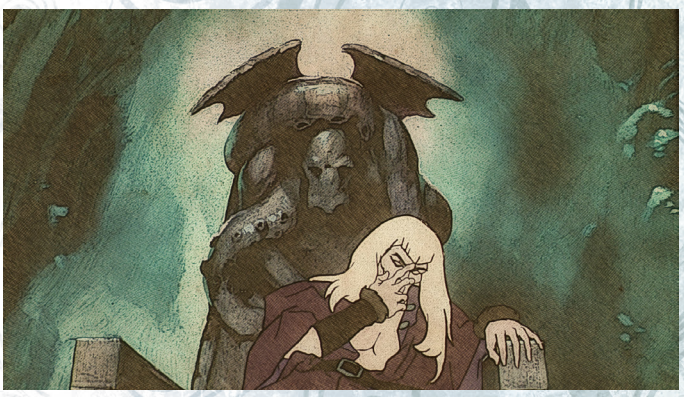
The creation of your game will be divided in 3 parts. Video editing, files preparation and SINGE editing. In this tutorial i will cover the video editing part very slightly (there are plenty of good tutorials for this all over the web).

Part 1: Edit the video file

For a complete game you'll need to gather a few 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.

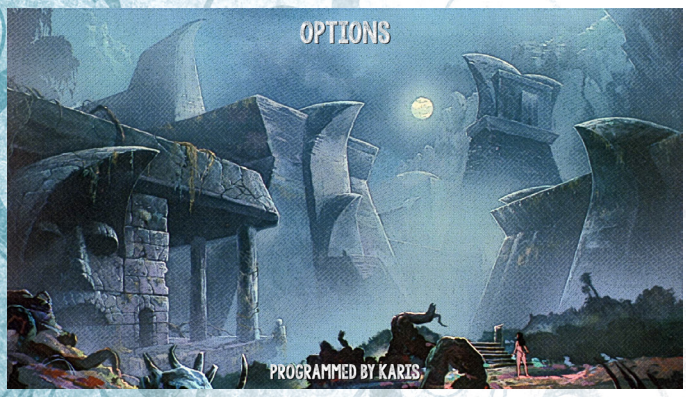
Attract: it is a longer video to present the story. It is longer because it will loop, waiting for the player to start the game. You can have up to 3 different Attract movies.



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, it will give the player enough time to breath between failures.

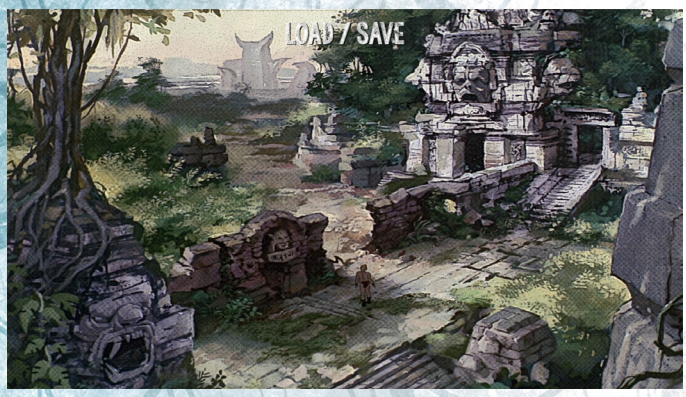
Levels: this is where the game is played. You need at least 1. In those movies the first part can be declared as an intro and skipped with the push of a button. It is useful to have an intro clip to present the action, but it's highly optional.

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



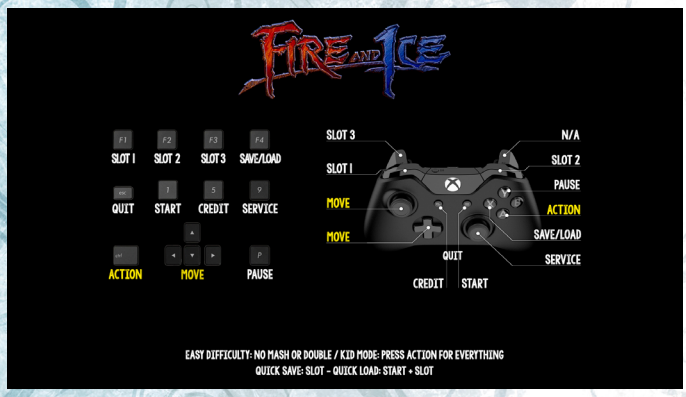
a **single frame** for the option menu

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

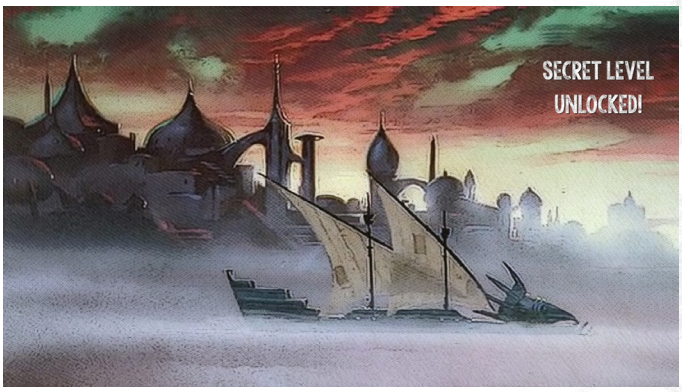
All those parts will be shown by the program during special moments :
- Option menu when you hit the service key (9)
- Victory when you finish the game
- Save menu when you enter this menu (F4)
- Rankings and commands, during the Attract (you can hit Ctrl to swap between things)
- Game over when you 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

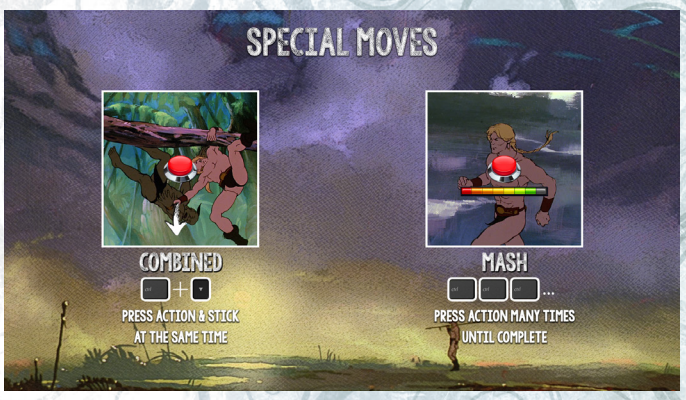
For those videos, you need to have the good length, because it will define the timing of the event (for example entering high score). So sometimes you don't need to be too long (Game Over is around 200 frames), and sometimes you need to be longer (High Score is around 700 frames).

Also note that a 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

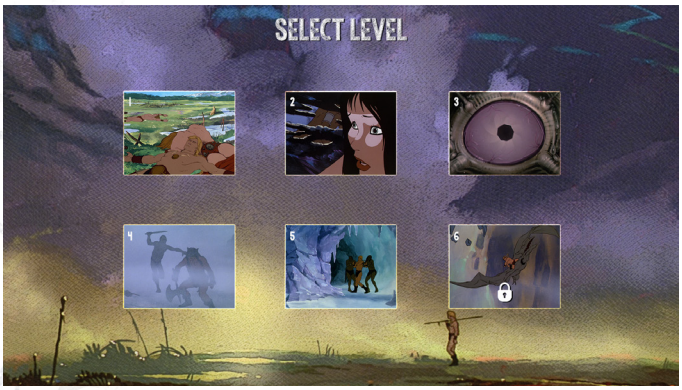
a **single frame** for the Trophies

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 where you found this tutorial
- 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

```
MYDIR = «Blank»
```

Here, once again replace Blank with the name of your game (between quotation marks).

```
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.

```
SCOREMOVE = 700
BUFFMOVE = 300
SCORESCENE = 10000
SCORELEVEL = 50000
SCOREGAME = 250000
DEATHPENALTY = 1000
PERFECTBONUS = 10000
EXTRALIFE = 0
SCORESECRET = 1
```

In most cases you don't want to change this. SCOREMOVE is the score for completing a move. BUFFMOVE is a bonus for each move depending on the difficulty (Easy: 0, Normal: 300, Hard: 600, Extreme: 900). SCORESCENE/LEVEL/GAME is the score for finishing a Scene/Level/Game. DEATHPENALTY is a penalty applied to SCORESCENE for each death during the Scene. PERFECTBONUS is a bonus for finishing a level with no death. EXTRALIFE is the score to reach to earn a new life (0 means no new life). SCORESECRET is a bonus for finishing the Secret Level. It may seem low, but any score finishing with 1 is very special!

```
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.

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.

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.

AllowTier = false

AllowTier: true or false. If true, you will be able to select tiered order in the Option Menu. Tiered order is a kind of «semi-random» mode used in Dragon's Lair. It's an advanced feature and in most cases you don't want to change this (see at the end of this manual for an explanation).

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 so in most cases you don't want to change this (see at the end of this manual for an explanation).

IngameDiffchoice = false

True or false. If true, you will be able to select difficulty through a video sequence at the beginning of the game. It's an advanced feature so in most cases you don't want to change this (see at the end of this manual for an explanation).

BarSize = 5
BarMin = 1
BarBonus = 4

Default options for the Life Bar Mode (see Appendix).

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.

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

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).

frameEasy = offsetMenus +18
frameNormal = offsetMenus +21
frameHard = offsetMenus +24
frameExtreme = offsetMenus +27

One frame for each difficulty level if IngameDiffchoice = true. If you don't plan to choose the difficulty in-game, leave it like this.

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

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.

```
offsetMap = 0
offsetMapend = 0
```

Video for the map. Once again, highly optional (see later).

```
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, 9849, 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, 9849, 2, 0, 0,,-1}
```

- Level[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 (9849) 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 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.

Here, 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.

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...

```
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}
```

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.

Alternate values for the Death

0 means random death. A random death will be played among those you defined earlier
-1 means No Death. The move is now optional, if you don't do it you won't die.
-2 means No Death with score. Same as the previous one, but scoring points

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.

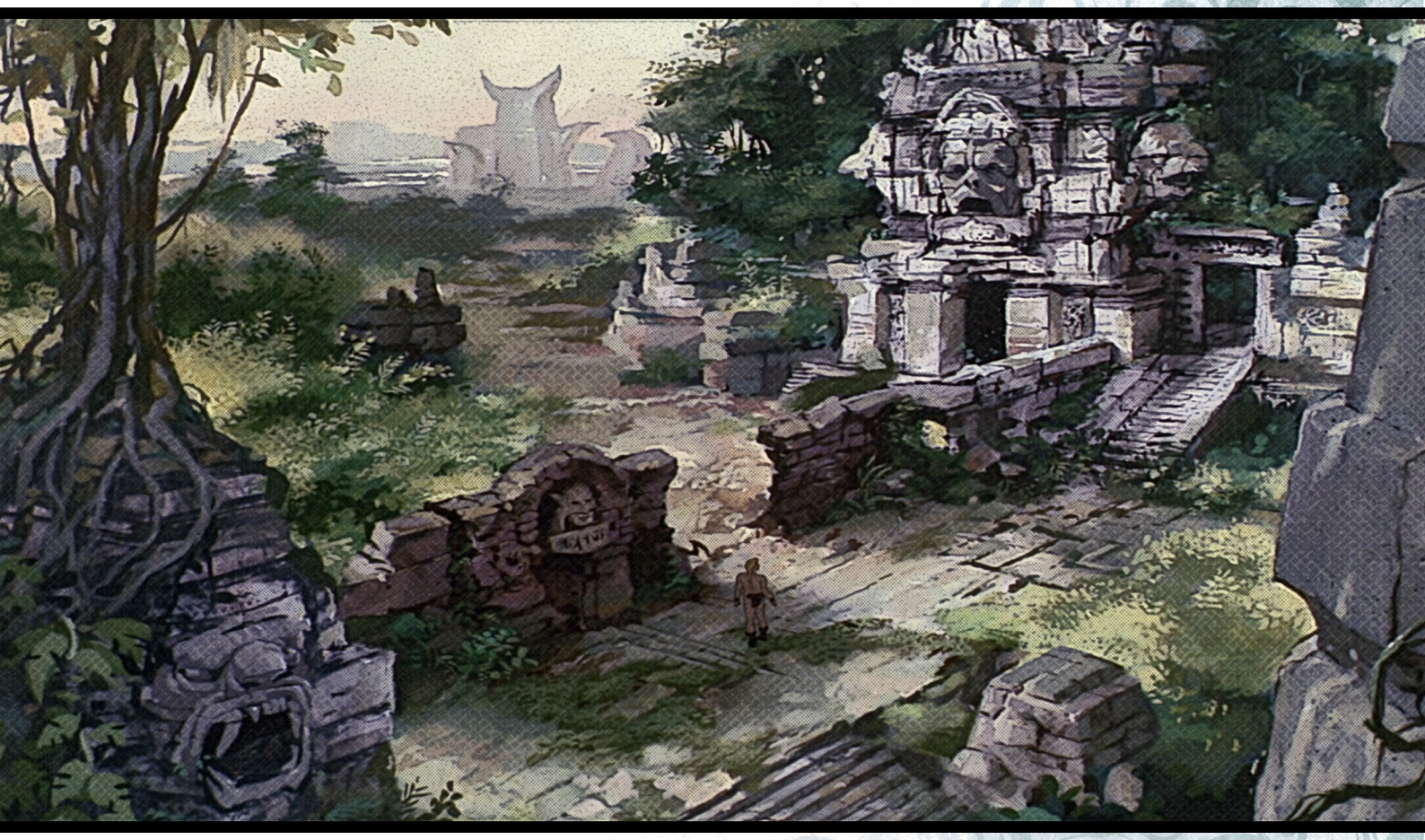
```
if thisLevel == 2 then

        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!



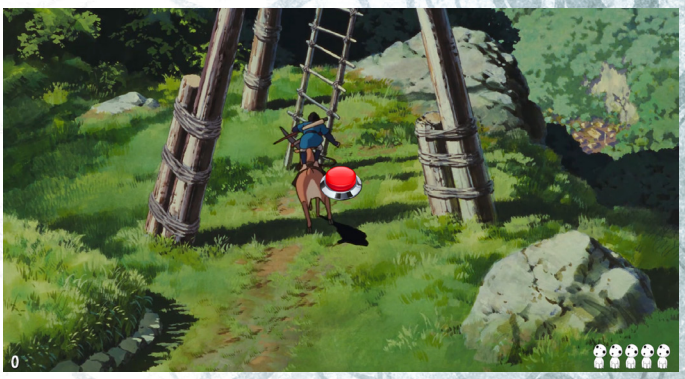
APPENDIX

MOVES

Here we will discuss about the possible moves you may ask.

A) Basic Moves

UP, DOWN, LEFT, RIGHT



BUTTON1, BUTTON2, BUTTON3, DOUBLE

Directional moves, and Action buttons. DOUBLE means you need to hit BUTTON1 twice.

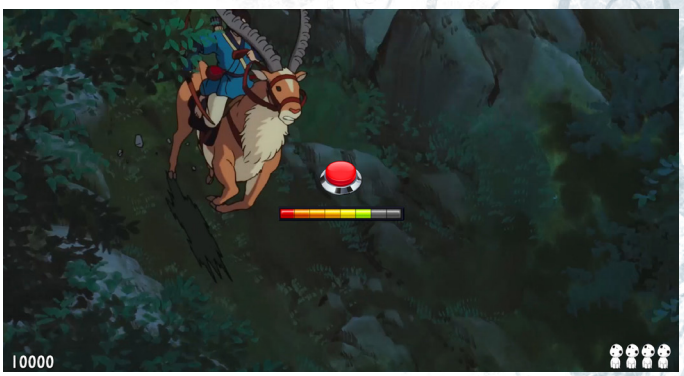
B) Combined Moves

UPLEFT, UPRIGHT, DOWNLEFT, DOWNRIGHT

ACTUP, ACTDOWN, ACTLEFT, ACTRIGHT

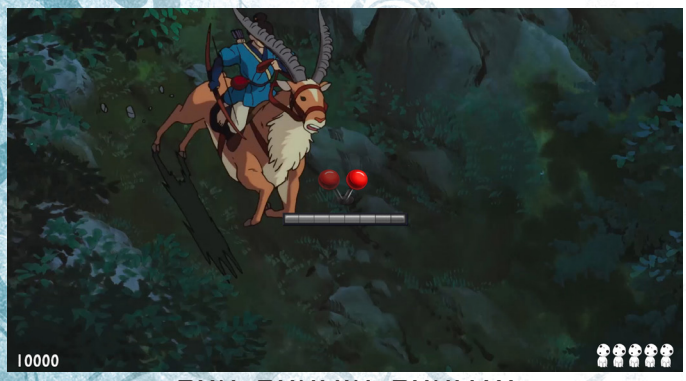
Diagonal moves and Action Button + Direction.

C) Advanced Moves

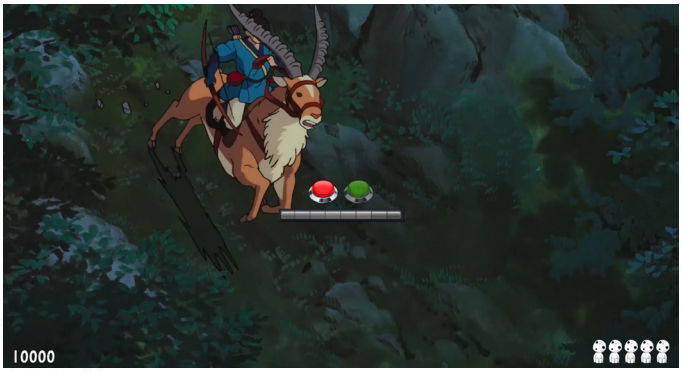


MASH, MASHMIN, MASHMAX

MASH will require to push the action button many times, fast enough to fill the gauge (which decrease between hits). MASH is adaptive, but it's better to have at least 20 frames (and up to 300). The decrease of the gauge depends on the difficulty. MASHMIN (and MASHMAX) are exactly like MASH but requires less (and more) moves.



RUN, RUNMIN, RUNMAX

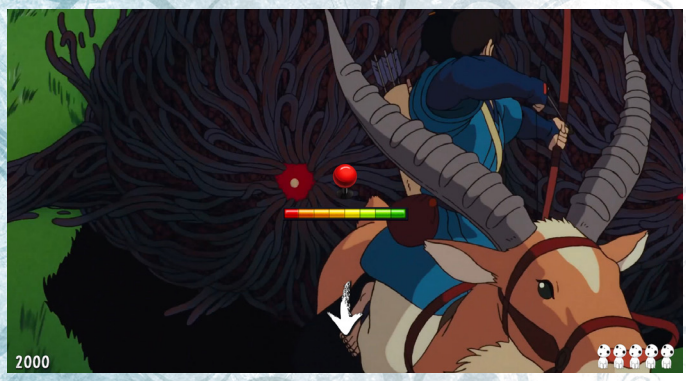


MASH2, MASH2MIN, MASH2MAX

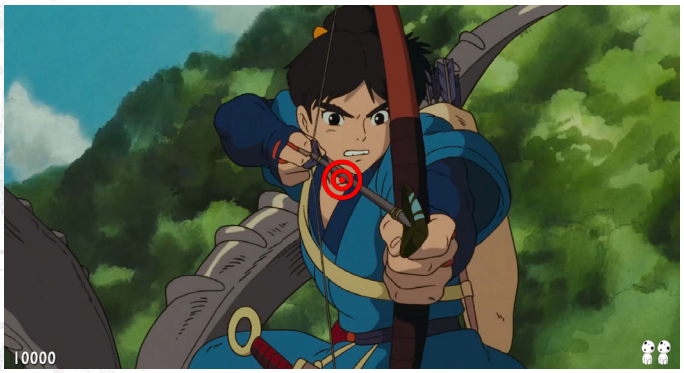
RUN will require to move the stick from left to right, fast enough to fill the gauge (which decrease between hits). RUN is adaptive, but it's better to have at least 20 frames (and up to 300). The decrease of the gauge depends on the difficulty. RUNMIN (and RUNMAX) are exactly like RUN but requires less (and more) moves.

MASH2 will require to push button 1 and button 2 alternatively, fast enough to fill the gauge (which decrease between hits). MASH2 is adaptive, but it's better to have at least 20 frames (and up to 300). The decrease of the gauge depends on the difficulty. MAH2MIN (and MASH-2MAX) are exactly like MASH2 but requires less (and more) moves.

In the Option Menu, you can decide how to use those actions. "Good Hand Stick" means RUN is made with your good hand. "Bad Hand Stick" is easier. "Two Buttons", mean that you don't use the stick, but MASH2 instead. "Mash Only" means those 2 actions are replaced with MASH.



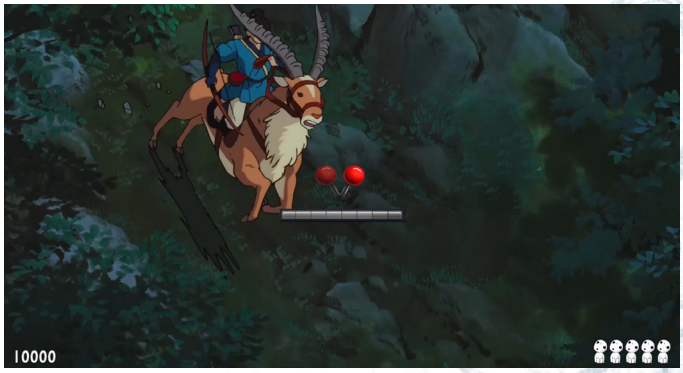
HOLDLEFT, HOLDRIGHT, HOLDLEFT, HOLDRIGHT
HOLDBUT



LETGO

HOLD will require to push a direction and stay until the gauge is full. Hold is adaptive but, due to the latency, you need at least 20 frames (and up to 300).
LETGO is alway after HOLD. It will require the player to release the HOLD key quickly. LETGO is generally short (10 frames).

Notes: HOLD can be standalone (without LETGO). Unlike MASH, HOLD will fail if you press the directional key before the instruction. HOLDBUT works with BUTTON1.
LETGO will fail if you release the key before or after the instruction.



MULTI

MULTI will require to repeat an action (button or arrow) several time. You have to define both the type of action and the number of repeats :

move[4] = {12350, 12414, MULTI, 81, UP, 4}

As you can see there are more items to fill for a MULTI move. The first 4 does not change but then you have to insert the type of move you want to repeat in position 5 and the number of repetitions in position 6. So in this example, you will ask player to go UP 4 times. Each time they go UP, one move will be removed from the screen (allowing a better count).

LOOPLEFT, LOOPRIGHT

Those moves will require the player to make a full circle with the stick either going to the left (LOOPLEFT) or to the right (LOOPRIGHT). To achieve the LOOPLEFT you need to pass through LEFT, DOWN, RIGHT, UP and LEFT in that order (LOOPRIGHT is, of course, the opposite).

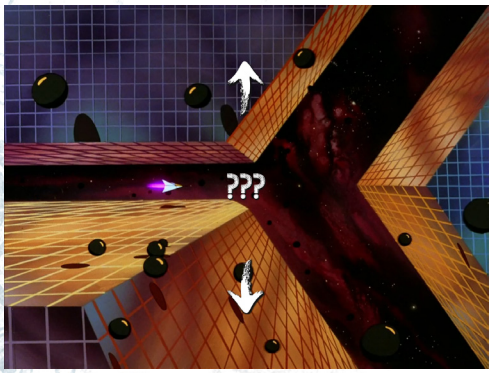
Note: the move is 4 directional (not counting diagonals) in order to be achieved easily on keyboard. But you can still replace it automatically in the option menu (Performance tab). In that case all LOOPLEFT and LOOPRIGHT moves will be replaced by HOLDLEFT and HOLRIGHT.

D) 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.

Note: this move doesn't grant points (if the skip is made) or provoke death (if ignored)

E) Complex Moves



PATH

PATH allow to choose between several moves/scenes/levels and jump to another location. PATH action is more complicated and require some special lines.

```
move[3] = {150, 170, RIGHT, 3}
move[4] = {200, 220, PATH, 3}
and so on...
path[4] = {UP, 5, DOWN, 7, 0, 0, 0, 0, 9}
```

As you can see there are 9 items inside a path table. But most of them will stay to 0.

Note that the value path[4] is matching the move where the PATH is called. So if you've got several path actions in the same scene, each one will have its own path table.

Values 1, 3, 5 and 7 are filled with up to 4 moves (BUTTON1 is possible for example). So in the above example the choice is between UP and DOWN, but if you replace the 0 (fifth value) with RIGHT, then 3 moves will be possible (7th value for another direction). 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. The 6th value indicates where to go if the third action is made (here it is 0 since there are only 2 choices). The 8th value is for the 4th action. The 9th value indicates where to jump when the defined path is finished.

So, in this example, 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.

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:

```
path[4] = {UP, 1025, DOWN, 7, 0, 0, 0, 0, 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:

path[1] = {UP, 2, DOWN, 3, RIGHT, 4, 0, 0, 0}

Let's say the PATH action is placed as last action of level 1. In that case, by going UP the player will go to level 2, DOWN to level 3 and RIGHT to level 4 (the jump value is not relevant in that case). It works the same with scenes, if PATH is the last action of a scene.

Last option: PATH can finish a level even it's not the last move of this level:

path[1] = {UP, 5, DOWN, 7, 0, 0, 0, 0, OUT}

If the Jump value (9th position) is OUT, then when path is finished, the level ends

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 action button.

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

path[4] = {5, 7, 9}

As you can see there are only 3 items in the path table. The first value is defines where to go when the button is pressed. The 2nd value where to go if nothing is done. The 3rd value is the jump value So, in this example, if the player press ACTION 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.

WAY, WAYOUT

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.

WAYOUT is exactly like WAY but will then Exit the level/scene.

CHOOSE

This move 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.

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. Let's look at the related timed table (like PATH, same number):

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]). In other terms, 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.

F) Alt Moves

There are cases where you would like to allow 2 possible moves. It will happen mostly for games where the Overlay does not show moves. For example, at the end of Space Ace (fight against Borf), the original program (depending on the rom) may allow several possibility with the staff. If Ace has to raise it to parry, is it BUTTON1 or UP ?

```
move[18] = {14777, 14789, BUTTON1, 80, UP}
```

As you can see, a fifth entry (UP) was added to the line. So when you're on move 18, you can hit BUTTON1 or UP and both will be accepted as a correct move.

Notes: if the overlay is showing moves, only button1 will appear on-screen. Also, this action is possible only for simple moves (you cannot use MASH, PATH or else).

G) 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.

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* (* 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)

- Coins per Credit: Freeplay/1/2/3. Just for fun, you don't need real coins.

Note : during the intro you can add coins with key 5

- 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.

- Run Action with: Good Hand Stick, Bad Hand Stick, Two Buttons, Mash (see RUN).

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

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:

- 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.

ADVANCED STUFF: 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, 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.

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 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!

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.

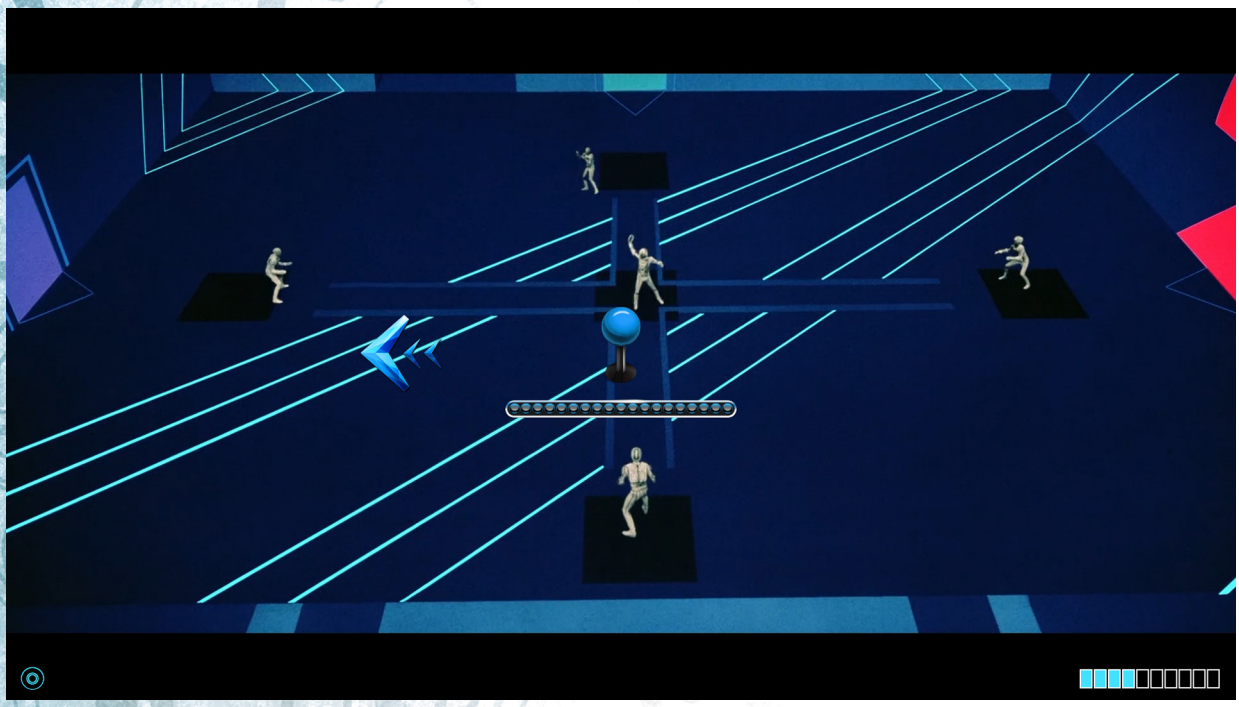
PERCENT MODE

Percent mode 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 (as defined in the Options menu).

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).

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

LIFE BAR MODE



Life Bar mode 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 default Life Bar settings are located in the Advanced Settings part of the Menu:

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.

SURVIVOR MODE

Survivor mode 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.

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

ADVANCED STUFF: 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 Menu.bat if you want to try it.

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 games.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,
                NO_MOUSE     = true,
                RESOLUTION_X = 1920,
                RESOLUTION_Y = 1080,                                  --
                SINDEN_GUN   = "",
                POSTER       = "Blank/Overlay/menu-poster.png",
                MARQUEE      = "Blank/Overlay/menu-marquee.jpg",
ATTRACT         = "Blank/Movie/menu-attract.mp4",
                ATTRACT_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.