Technical Specification Document

[Coding Guidelines](#_fxvukeu9k17x)

[Scripts And Events](#_hack7kfcd2ch)

[Variables](#_roie85905go7)

[Formatting](#_c7v9zjlmbqtz)

[Misc](#_34eyf11drlfy)

[Version Control](#_c35pipn1vpdd)

[Folder Structure](#_losr7ih79ymj)

[Branches](#_94nhgj5t42yf)

[Pull Requests](#_ltqs3vplavv5)

[Commits](#_qd2r0ea43gm)

[Tags](#_z3rcxu524n52)

[File Formats](#_lz9dcjuknuec)

[Sprites](#_zi5izjmjgxyb)

[Animations](#_u9hfl2wqta1q)

[TileSets](#_g908sebadcii)

[Audio](#_l8498pye06v2)

[Documents](#_gtriweyifpt7)

[Game Maker](#_uth5b9uk791e)

[Assets](#_jbzmnhqrj1up)

[Scripts](#_h3jrsdmu5y10)

[Folders](#_jycq7r56flpw)

[Target Platform](#_2b74w3uqqidj)

[Workflows](#_ga0sqxkvq4jb)

[Assets](#_1na7hsq8hdk8)

[Objects](#_fuvhkndx7vos)

[Rooms](#_i8nhm0obw0y7)

[Communication](#_c4m3ykdbcsuq)

[Slack](#_e4qav4u4o1kk)

[Google Drive](#_53iufiqdp4wn)

# 

# 

# Coding Guidelines

## Scripts And Events

All scripts should follow snake casing naming conventions. Scripts that are related should share a common prefix. This will make it easy to identify related scripts as well as ensure that the built in intellisense will offer relevant options.

Constructor scripts should contain the postfix init. Similarly destructor scripts should contain the postfix destroy. This is the practice that internal GameMaker scripts follow and will avoid confusion.

|  |
| --- |
| //grouped scripts  entity\_init();  entity\_move();  entity\_attack();  entity\_destroy(); |

## Variables

All variable names should be descriptive and indicative of their purpose. Exceptions can be made for index variables as long as they follow the convention of being labeled i,j,k. All variables should be declared using the most local scope possible. (local -> instance -> global)

Global variables should never be used. In the case that a global variable is needed, use a persistent object and a instance variable. This is because the existence of a global variable is not easy to verify within GameMaker, and accessing a global variable before it has been declared will crash the application.

|  |
| --- |
| global.var foo = 109; |

Instance variables should be named using snake casing. They should always be declared during the creation event of an object. This is to prevent any errors that may occur when accessing non existent instance variables.

|  |
| --- |
| foo\_speed = 42; |

Local variables should be named using snake case. All script internal variables should always be stored as local variables.

|  |
| --- |
| var instance\_variabel = 22079460347; |

Macros should be named using capital snake case as to distinguish them from other forms of variable. They should be used for any form of global constant. Even though GML macros support expressions as values, this should be avoided and scripts should be used to contain these expressions instead.

|  |
| --- |
| var t = MIN\_TIMEOUT; |

## 

## Formatting

All code blocks should be indented with four spaces. There should be one blank space separating relevant code blocks. All statements should be completed with a semicolon.

|  |
| --- |
| var a = 4;  var b = 5;  var c;  c = a \* b;  c = abs(c, 10);  if(a > b)  {  c = 8;  a = -1;  } |

All control flow statements (if, while, for) should be followed by open and close curly brackets. The brackets should be located on the first line following the statement. This is to increase readability and make it easy to edit in the future.

|  |
| --- |
| //good  if(a < b)  {  //code here  }  //bad  if(a < b)  //code here |

The first line of every script should be a comment beginning with three forward slashes and then the method signature of the script. This allows game maker to create accurate intellisense for the script.

All arguments used by scripts should be set as local variables at the beginning of the file. This will make the scripts more readable and easier to develop.

|  |
| --- |
| ///script\_name(input, args, here)  var input = argument0;  var args = argument1;  var here = argument2; |

## Misc

* Always use functions built into Game Maker Studio rather than implementing new ones. Game Maker Studio functions are written in C and already optimized.
* Use custom events to trigger actions that could be changed later in development
* Avoid creating dependencies between scripts.

# Version Control

## Folder Structure

All assets should always be located within the repo. All asset types should be stored within separate folders within the repo. Some assets will have two or more forms during their creation. In this case, a sub folder titled Final can be used.

The main game maker project is stored within the Game folder.

## Branches

No changes should ever be committed directly into the master branch. The master branch should be reserved for the current final iteration of the game. Only completed work should ever enter the master branch.

All ongoing work should be done on sub branches and then submitted to the master branch via a pull request. The branch name should be indicative of the work being done. Multiple developers may work on the same branch if needed.

## Pull Requests

When development on a branch is completed, a pull request should be made. The pull request format should follow the same rules as commit message formats. Anyone can accept a pull request, however this should not be done by the creator of the request unless it cannot be avoided.

If a pull request contains merge conflicts, it should be rejected immediately. It is the responsibility of the branch to fix any conflicts before merging to master.

## Commits

Commits should attempt follow the best practice of encapsulating one change at a time.

Commit message summaries should be concise and indicative of the task performed. They should be kept short, and should not reference specific tasks.

Commit message descriptions should contain a more detailed description of individual tasks performed, broken down into subtasks if required. These can be in any form as long as it is clear. If a commit is related to a task, then it should be shown within the commit message. To reference a task, the following format can be used:

|  |
| --- |
| @Task [TaskId] |

## Tags

Tags should be used to indicate any milestones within the master branch. At any release, the tagged version corresponding to that release will be used to generate the executable. Therefore the version must be accepted by all team members before a tag is assigned.

# File Formats

## Sprites

All finished sprites should be stored as PNG files. They should have both a width and a height that is a power of 2 in order to be consistent with world sizing. The majority of sprites should be 64x64 in order to align themselves correctly within the dungeon. Character and enemy sprites should be 64x128 pixels in size.

Sprites that are in development should be stored as PSD files. These original versions should be stored at all times, even after a sprite has been completed.

## Animations

Finished animations should be stored as single sprite sheet containing the full animation. Each frame must be smoothly separated in order to import correctly. The format of the sheet should be the same as regular sprite assets.

## TileSets

Finished tilesets should be stored as PNG files. All tiles on a tileset should be 64x64 pixels in size. There should be a 1 pixel separator between tiles. The tileset itself should have an equal height and width.

## Audio

Finished audio should be stored as OGG files. All audio should be trimmed so that there isn't any excess silence before or after the effect/soundtrack.

## Documents

All documents should be created and stored within Google Drive. They should all be well formatted and readable. Documents that are related to specific tasks or assets should be stored within well marked folders.

# Game Maker

## Assets

All game maker assets should be named using snake case, they will all contain a prefix to indicate the type of asset that they are. All assets that are related should be kept in folders within the asset tree.

The prefixes to use for each object types are as follows:

* Sprites -> spr
* Sounds -> snd
* Backgrounds -> bkg
* Paths -> pth
* Shaders -> sdr
* Fonts -> fnt
* Timelines -> tml
* Objects -> obj
* Rooms -> rm

## Scripts

All game maker scripts should be named exactly the same as their calling name. Every script asset should contain only a single script. Tabbed scripts should be avoided because it hides the existence of scripts from other developers.

## Folders

All folders should be named using camelcase and starting with a capital letter. Folders should be well organized and named descriptively.

## Target Platform

The target platforms for this project are standalone executables for Windows, Max OSX and Linux. The compiler used to build for the platform is preferably YYC, however this is only required for Windows.

Minimum Target Versions:

* Windows 7
* Ubuntu 14.04
* Mac OSX Yosemite

# Workflows

## Assets

Any created assets should be stored within the git repo at all times. The raw version can be stored directly in the corresponding asset folder.

Finalized assets should be stored within a folder labeled Final that can be found within the corresponding asset folder. These finalized asset files should be named using the same conventions as are used within Game Maker.

## Objects

Game objects should be built using object composition practices. Inheritance should be kept to a minimum, however it can be used in cases where Game Maker functions already support inheritance (collision).

Code located within game objects should be kept to a minimum. This code should, for the most part, be references to other scripts to execute.

## Rooms

Rooms should be set to run at 60FPS. They should be no larger than the scene contained within. All rooms that are to be used for the level generator must have odd multiples of 64 as both height and width.

# Communication

## Slack

All general information should be communicated via the Slack Team Chat.

Individual channels should be created and used to discuss any features. #General should not be used for any feature specific design decisions. All completed tasks should be announced via the #task channel. This way all members of the team can be notified when changes are completed.

The #general channel should be used for inter team discussion and information about the project. Non game related information should be shared using the #random channel.

## Google Drive

Any documents should be stored within the Google Drive folder. If there are notes regarding meetings or useful files that are not game assets, they should be stored here.

Any documents that are required for milestones should also be stored within the root folder within the drive.