Master Specifications Document

Project: JarsTiles

# Overview

This document contains all of the features necessary for the game development. When you feel like taking up a certain feature of the game, just add your name to the right of the feature in brackets, include a starting date, and highlight the field blue.

For example:

* Collision Detection (Richard Peng- 2/22/11)

**Note: Before you decide to take on a feature, make sure you have the latest specifications document! You do not want to overwrite the Master copy before you merge the changes with your local copy!**

Each feature will be colour coded as follows:

|  |  |
| --- | --- |
| **Colour** | **Assignee** |
| [Yellow] | Feature available for implementing |
| [Red] | Bug or design problem |
| [Green] | Feature complete |
| [Blue] | Implementation in progress |

# Main Project

## V 0.1

* Flex compiler (mxmlc) build script
  + Need a script that can compile a swf file for the project, for people that don’t have flash
  + Idea: <http://www.senocular.com/flash/tutorials/as3withmxmlc/>

# Main Program

The main program bootstraps the engine, and initializes everything required for this to run. This is the program entry point (document class)

## V 0.1

* Main class (jarsTiles.as)
  + Basic bootstrapping

# Player

The Player class has all the attributes of a normal player. This must include position, status, and score.

## V 0.1

* Player Class (player.as) (rpeng- 2/23/11)
  + Variables to track position, status, score
  + Getter/ Setter methods, etc

# Tile

The tile class represents a single tile in the arena.

## V 0.1

* Tile Class (tile.as) (rpeng-2/23/11)
  + Basic status- such as if tile is still alive, and which players reside on the tile
  + Methods to drop tile

# Engine

The game engine contains all of the logic needed for the game to operate. The engine must be able to handle input, process game objects (such as players and tiles), and handle all of the logic (collisions, scores, etc.)

## V 0.1

* Engine Core (engine.as)
  + The engine core handles game objects and logic. In-game, the engine needs to keep track of the positions and status of all the players, as well as the status of all the tiles. It will also need to handle user input that will control each character.

# Painter

The painter module generates the terrain, and paints the graphics based on the data from the engine. It will also handle “skinning” of characters, so that players can create their own unique avatars.

## V0.1

* Painter core (painter.as)
  + Ability to generate a 2D basic terrain with markers to indicate characters