|  |  |
| --- | --- |
|  | **Plentoon Games** |

**Software Design Document**

**All work Copyright ©2017 by Plentoon Games**

**Written by Chengyu Yang, Kin Pan Mak, Liam Goerke and Jayden Grant**

**Version # 1.0**

**March 31, 2017**

# Game Engine Architecture

* Player movement
* Particle effects
* Physics engine/ collision detection
* Graphics engine
* Model loader

Files:

* Camera
* Control
* GameMathHelp
* GameTime
* Graphics
* Matrix
* ParticleEmmitter
* Vector3
* Collusion
* World
* GameObject
* GameLoader

Explanation:

Main.cpp calls the game entry class World to initialize and create game rendering system, object loading, and then enter the main game loop (loop through the game components)

World class loads up all the game components (game assets, rendering system, camera) and keep looping through the game loop (updates the camera and user interaction)

**GAME COMPONENTS**

Graphics file stores the base class Graphics for general graphics API functions and variables. OpenGL inheritance from base class that using glut render system. DirectX has not yet been developed. GraphicFactory class creates the wanted API child object.

Camera class stores the camera position and look at position, and running the movement and rotation calculation. (not affected by different input device)

Control class undertakes the control function (currently for glut) to handle the user interaction

GameTime class runs in the main game loop and tracks the game refresh rate. It will be used for balancing different computer speed caused objects in different update.

ParticleEmmitter runs the particle effects. Currently is only flying with patterns. Will be changed to something meaningful in next version.

Collusion works for collusion detection between player and other objects in the game world.

GameObject and GameLoader work for models and store as an object to be displayed.

GameMathHelp file stores 3 classes, MathHelp (general math help for the game), Vector3 and Matrix. They are mainly for camera calculation and object loading.

# External file formats

Most of the 3d model files are used to load objects into the world using the model loader

Load textures(not for 3d objects) and game objects in Initialize() function in World class using GameObject and GameLoader functions.