|  |
| --- |
|  |
| Milestone Document |
| Games Engine Construction |

|  |
| --- |
| Ben Ritson  t7093422 |

# Game Demo Summary

To demonstrate the functionality of my Game Engine, I intend to create a top-down, vertical, 2D shooter game. The goal of the player is to avoid incoming projectiles fired from enemy ships while attempting to hit and destroy enemy ships with their own projectiles, increasing their score along the way. In order to create the engine and game demo, I have set out some milestones to achieve throughout the development process.

# Milestone 1

The requirements for this milestone are to create functionality that:

• Can fire projectiles from the player sprite

• Can instantiate objects, and delete them, outside of the game loop.

As with all milestones the code should be bug free, well commented, maintainable and

contain error checking and work in both debug and release modes.

In order to test that the milestone requirements have been met you need to show:

• That a projectile can be fired from the Player sprite and will continuously move until it leaves the screen, it should then delete itself.

• That your objects, such as the player and projectiles, should be objects instantiated from their own class instead of a collection of variables stored in the main function.

Code Structure

• You should have a fully encapsulated Entity class to hold entity data and private functions.

• You should have at least two other classes, both of which should be child classes of the Entity class, to create objects from. E.g. PlayerEntity, ProjectileEntity.

• The projectiles must be self-contained. Once a projectile is fired from a PlayerEntity, the PlayerEntity shouldn’t need to interact with it anymore. For example, once instantiated, the bullet should propel itself without communication with the PlayerEntity.

• You should have a World Model system to hold a collection of entities and public functions to access the entity functionality.

•You should avoid allocating or deallocating memory within the game loop to increase optimisation.

# Milestone 2

The requirements for this milestone are to create functionality that:

• Can draw multiple kinds of enemy ships (or the equivalent) to the screen through instantiated objects similar to the player and projectiles.

• Can move the enemy ships.

• Can detect bounding rectangle collisions.

• Can store an HP variable for the enemy ships.

As with all milestones the code should be bug free, well commented, maintainable and

contain error checking and work in both debug and release modes.

In order to test that the milestone requirements have been met you need to show:

• That an enemy ship object can be instantiated and drawn to the screen following a pre-programmed path.

• That more than one enemy ship can be on the screen at once, each with different variable values. One may have more HP, for example.

• That the projectiles can collide with other ships and lower their HP.

• That enemy ships can be destroyed when their HP reaches 0.

Code Structure

• The enemy ships should be instantiated from a class which inherits the Entity class, like all other objects, and stored in a collection (such as a vector or map).

# Milestone 3

The requirements for this milestone are to create functionality that:

• Can make object movement smoother by using interpolation.

• Can have enemy ships fire projectiles using basic AI routines.

• Can tally up a score variable and display it to the screen.

• Can store an HP variable for the player.

• Can destroy the player.

As with all milestones the code should be bug free, well commented, maintainable and

contain error checking and work in both debug and release modes.

In order to test that the milestone requirements have been met you need to show:

• That an enemy ship object can intelligently fire projectiles at the player which can cause damage to the player’s health.

• That when the player’s health reaches 0, the player is destroyed, a dialogue box will appear, and then the game will close.

• That the score variable is displayed on the screen and increases when an enemy ship is destroyed.

• That the player’s health value is drawn to the screen.

Code Structure

• You should have an AI system in place to hold functionality for enemy ship AI.

• You should have a UI system in place to hold functionality for things such as player input and drawing the score and player HP to the screen.

# Milestone 4

The requirements for this milestone are to create functionality that:

• Can play sound effects and music.

• Can load level data from a file.

• Can save score data to a file.

• Can restart the game when the player dies.

As with all milestones the code should be bug free, well commented, maintainable and

contain error checking and work in both debug and release modes.

In order to test that the milestone requirements have been met you need to show:

• That sound effects are played at appropriate times, such as when a projectile is fired or a ship is destroyed.

• That background music will play during the game and loop upon completion.

• That a player can progress through a level, defeat a boss ship, and then proceed to a different level.

• That when the player dies, the option to restart the game appears, instead of closing the game.

• That when the final level is completed, a victory notice appears and the player is given the option to restart.

• That once a player dies or is victorious, their score will be saved to a file.

Code Structure

• You should have an Audio system in place to handle functionality for playing sounds and music.

• Your World Model system should provide the functionality to load and save the level and score data to files.