P4220933	Johann Ostero	08/01 - 2016

CEG – Milestone 5th

Introduction

The game is a 2D Horizontal plane shooter. It is going to be a semi-physics based game, where gravity is a factor. This means that the plane can fall down to the ground, crash into various other objects and based on the angle or speed will eliminate the plane.

As it is a plane shooter, the game will include shooting various projectiles, such as rockets and regular machine gun bullets. The game is heavily inspired by **Triplane Turmoil**[1]

As of now, the game will only have versus mode.

Milestone 6 – World Model implementation

The task is to create a world model as well as Objects that are included in the game, such as player, enemy, scenery, etc. All the object types will inherit from the a core Object and this core object will contain all the properties that every single object will need, such as Sprite Id, Transformations, is active, etc., as well as actions such as Update and Render.

Everything will be moved in to the world class, this including *HAPI init* and *HAPI Update*. Every object will be stored in a std::vector, and treated polymorphically.

The way to test it is to run 3 maps with each their own objects with different property values, such as scale, rotation and position, as well as various sprites.

Milestone 7 Input & Directional Movement

Input will be added to handle custom user input, meaning that the user can decide which keys/controller input should be used for what controls. This will also include hooking up controllers to add or take over current controls. This will be part of the player object.

Directional Movement, is to implement a direction vector that allows the user to direct which way something should move. This will be very handy for when implementing Physics (i.e. Force) and projectiles. When it comes to force, it will play a key role in allowing the user to determine which way the plane should move. This will be implemented into the already setup 2D Vector class. This will be tested by making a plane do a loop.

Milestone 8 – Collision Detection

This task is to check if two objects have collided in any form, however, not all objects will reflect on the collision. It depends on the “side”, meaning that bullets will not hit bullets, but bullets will check for collision on enemies or players (depending on who shot it).

A method needs to be added to the base object class that checks for collision, since every object will have a potential to hit objects. The “sides” will be added to the base object class as well, this will be setup as enums.

This will be tested by making two objects with the same side and then two objects with opposite sides and print out the output.

[1] Triplane Turmoil information: https://en.wikipedia.org/wiki/Triplane_Turmoil_series

Milestone 9 – Physics

A basic physics engine will be implemented. This is to add gravity as well as force, making the plane explode upon collision, depending on its velocity. This will also add things such the plane falling down if its velocity goes too far down. This adds an additional feature that will prevent two planes exploding when colliding if the velocity is too low, however it will add the other object's force to itself and vice versa and take a small amount of damage. Essentially the amount of damage upon collision will be determined by the planes attack damage and velocity.

This will require a Rigidbody class. This class will not be added to the base object class, because not all objects are to be affected by gravity and forces, such as static objects.

This will be tested by trying to take off with a plane as well as crash land. Also, try to collide into other objects (other planes included) in various speeds.

Milestone 10 – Bullets & Projectiles

Projectiles, will be spawned from the plane and "shot" in the direction that the plane is facing. These projectiles will rely on collision detection to determine if they've hit something. When a projectile goes off the screen it disappears. There will be two types of projectiles: *Regular bullets* and *Rockets*

Both of these projectiles will behave in the same way, but with different property values, such as speed, rate of fire and damage upon impact.

This will require additional object child classes. There will only be added a fixed amount of bullets to game, meaning that none of the bullets will be deleted nor added when the game runs. When a bullet has been used it will become inactive, hiding it from the world. The amount of bullets will be determined by how many players there are. When firing, the bullets will be pulled from a projectile pool, by checking if the bullet is active; if not then set it to active and move it into position and shoot it.

This will be tested by shooting at objects and out of the map using both projectiles and see the results, depending on if the object will take damage or not.

Milestone 11 – Basic AI

The AI will be used to control the enemy. The enemy will be very basic and will simply follow the player and try to shoot them. Possible additional features could be to tell them to pick up weapons and use them against you.

The AI will be implemented into the Enemy object class and possibly having a separate AI state machine class, depending on how intelligent the AI becomes.

This will be tested by attempting to play against an AI in various scenarios as the AI will have to know how to avoid obstacles in its way.

Milestone 12 – Setup Game

Now base material has been implemented, only thing left is to add a scoring system, a UI to display the player health and ammo, also a menu screen that displays how many players are to be added to the game, setting up custom controls and to play the game.

Also determine the winner when there is only one plane left. This will be tested by allowing other people to play the game and receive feedback.