Project In Intelligent Systems - 236754
3D SpaceSim Using Oculus Rift on Unity

Developer: Alex Nicola - (alexnicola07@gmail.com)

Supervisor - Professor Mirela Ben-Chen

Project Idea:

A 3d space simulation game that will use the Oculus Rift as monitor display and for looking mechanism. The game will be developed in Unity3d for Windows.

Link to executable:

https://www.dropbox.com/sh/09r6vfgb4esztn8/AAAz8mpfm4wj3Y4LJfeopOHha?dl=0

Link to project source:

TBD

YouTube Demo:

https://www.youtube.com/watch?v=YHK5eGBhTYU

Features Description

Flight Engine - The user uses the Keyboard and Mouse to navigate the Ship in 3d, the mouse controls the camera direction and the Keyboard controls the force added to navigating the ship in the right direction

Head Up Display(HUD) - Information is displayed on the screen including a visualization of the ship's Cockpit and panels. The information includes the ship's speed, Shields, Health, Ammunition and more.

First scene - A scene including some large ships, small fighters, asteroids, Space with stars, space dust for fast travel effect

Shooting mechanism - The user uses the left mouse button to fire the primary weapon of the ship. The projectile will fly straight forward and collide on impact, causing damage. The user uses the right mouse button to Fire a missile that will find the first nearing target.

Radar system - The user have the coordinates in space of all enemies. An indication of the enemies' coordinates is projected on a 2d radar map that is updated in real-time.

Oculus Rift Display - The view is displayed in 3d on the Oculus Rift

Oculus Rift Controls - The user controls the view direction according to the head direction. This is done with the Oculus Rift Gyros/Accelerometers

AI of enemies - The enemy ships try to destroy the player by chasing him, shooting him and trying to perform evasive maneuvers

Wave manager - The game spawns more waves of enemies to fight the player

How to Play?

1. Press W to give speed to the ship
2. Press Spacebar to hit the brakes
3. Use the mouse to navigate the ship direction
4. use Q and E to roll left and right
5. Use S and X to get the ship up and down in the Z Axis
6. Use Z and C to strafe the ship left and right in the X axis
7. Press the Left mouse button to shot lasers
8. Press the Right mouse button to shot Missiles
9. Press V to watch the ship in 3d person view
10. press B to look back

For oculus Rift:

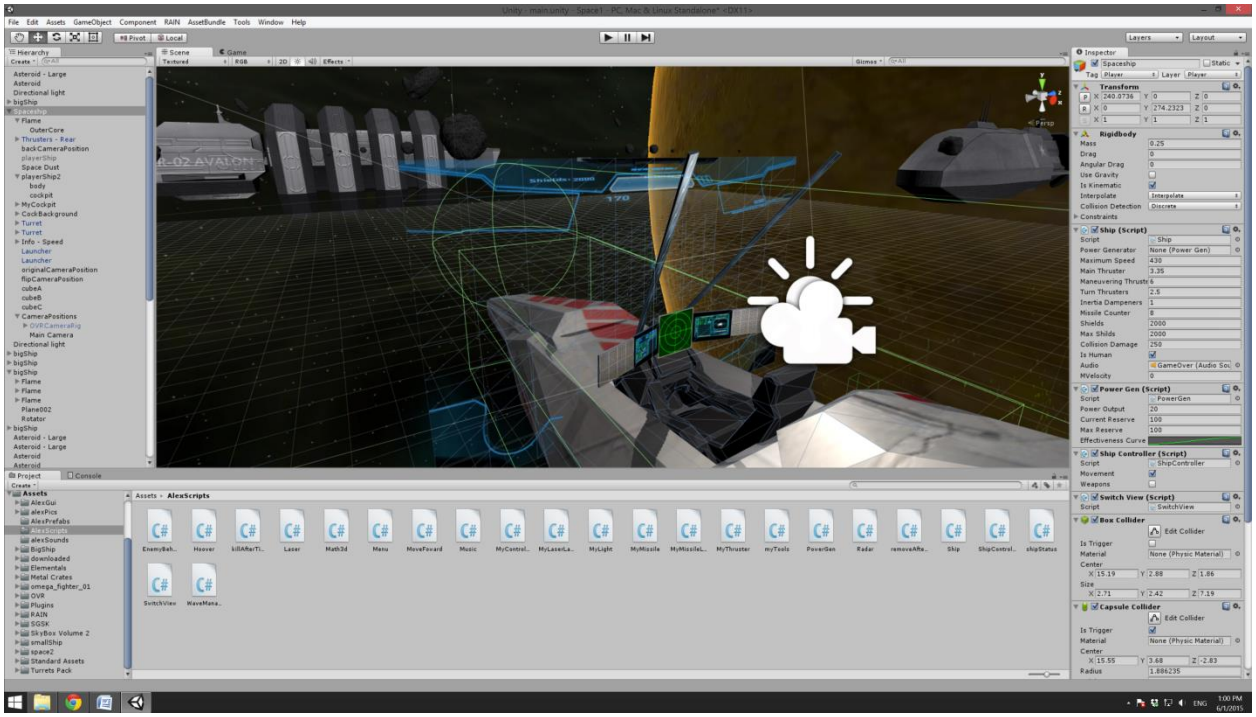
Make sure to install "oculus_runtime_rev_1_sdk_0.4.3_win"

it is provided in the game folder and also here

https://developer.oculus.com/downloads/

Make sure you plug the USB and the HDMI cable of the oculus and turning its power on.

Some cool screenshots



Shields: 2000
Missiles: 4
83

Shields: 871
Missiles: 4
145

Implementation

Missile behavior

Missile are launched from generic launcher class that instantiate a projectile (laser or missile) that has its own behavior. Therefore you can put launchers anyway you want on your ship and on the enemy's ship.

After a short delay, the missile acquire a target by calculating the shortest distance of all enemy ships in a pre acquired list

Then the missile change its rotation to look at the enemy ship and translate forward

This is not the most effective way. There should be an interception algorithm that gets a point that is calculated according to the velocity vector of the enemy and the missile.

Another problem was the missile trail. After the missile explodes, its instance is destroyed. The missile trail was its child, so trails immediately disappeared.

The solution was to set a new parent to the trail. The parent should destroy the trails after a given time.

Enemy behavior

public enum EnemyState{ FollowPlayer, AttackPlayer, Idle, Evasive}

A state machine was created for the Enemy Artificial Intelligent (AI)

The state would than translate to the correct state according to some conditions.

The default behavior is to follow the player.

A random number is generated. If the number is high enough, the enemy shots the player for a few seconds

The Evasive state is triggered by distance from the player. If the player gets too close, the enemy generates 3-4 new points in space and flies to them sequentially.

Radar behavior

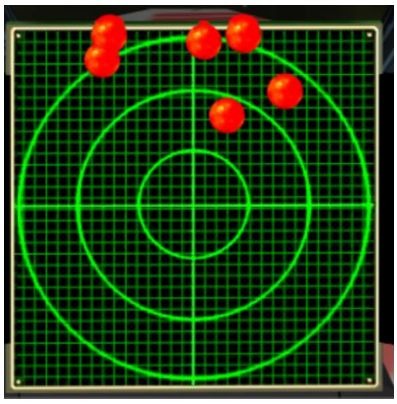
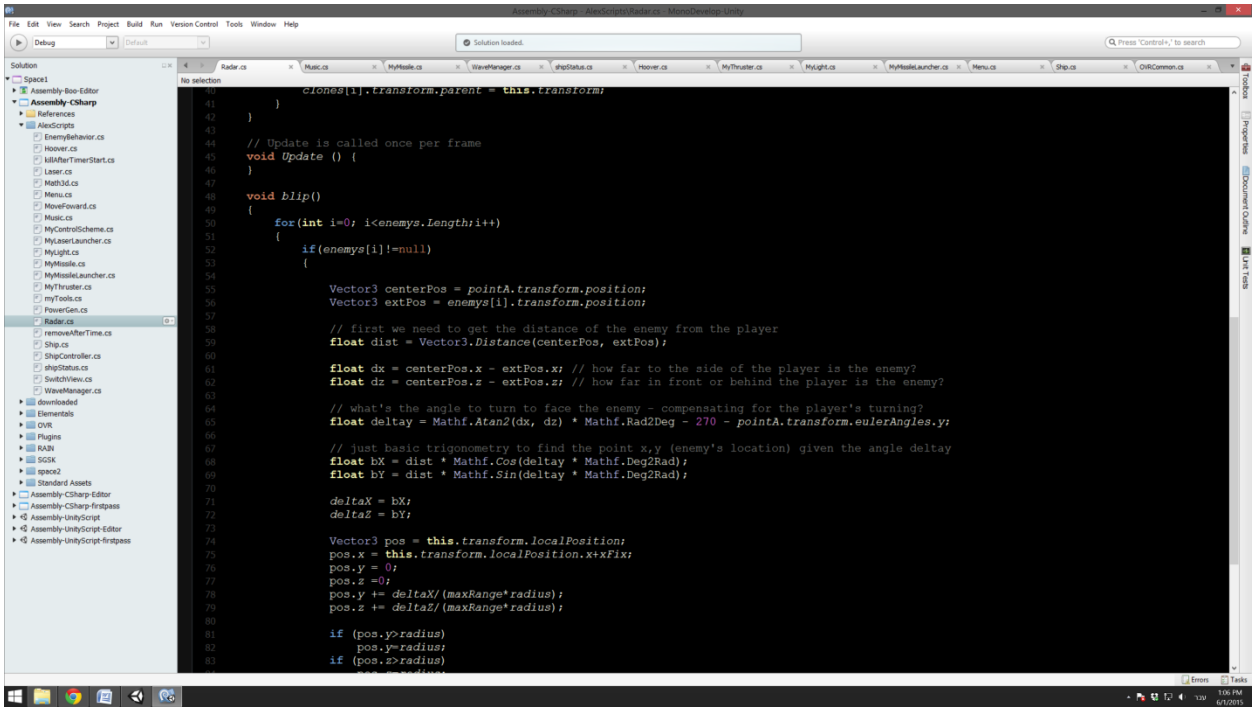
The radar acquires targets every second.

It takes every enemy 3d point and projects it on the player plane.

The player plane is calculated by taking a predefined point and a normal.

Afterword it takes the distance in X axis and Z axis and projects it to X and Y axis in 2d space.

The Radar compensate the rotation of the player by doing some basic trigonometry on a triangle and getting new X,Y points for the enemy.

Also the distance was calculated according to Range and radar width ratio.

Finally, if the range was too high, the point is displayed on the borders of the radar