# OCULUS RIFT
# AND
# LEAP MOTION
# FOR VIRTUAL REALITY

CGGC PROJECT 236629

## DOCUMENTATION


**ADVISOR**

ASST. PROF. MIRI BEN-HEN

**CREATORS**

KATYA SAPOZHNIKOV

ZIV SHAHAF

# TABLE OF CONTENTS

## INTRODUCTION

The **Leap Motion controller** is a small USB peripheral device which is designed to be placed on a physical desktop, facing upward. Using two monochromatic IR cameras and three infrared LEDs, the device observes a roughly hemispherical area, to a distance of about 1 meter (3 feet). The LEDs generate a 3D pattern of dots of IR light and the cameras generate almost 300 frames per second of reflected data, which is then sent through a USB cable to the host computer, where it is analyzed by the Leap Motion controller software using "complex maths" in a way that has not been disclosed by the company, in some way synthesizing 3D position data by comparing the 2D frames generated by the two cameras.

The smaller observation area and higher resolution of the device differentiates the product from the Kinect, which is more suitable for whole-body tracking in a space the size of a living room. In a demonstration to CNET, The Leap was shown to perform tasks such as navigating a website, using pinch-to-zoom gestures on maps, high-precision drawing, and manipulating complex 3D data visualizations.

The **Oculus** Rift uses state of the art displays and optics designed specifically for VR. Its high refresh rate and low-persistence display work together with its custom optics system to provide incredible visual fidelity and an immersive, wide field of view. The Rift's advanced display technology combined with its precise, low-latency constellation tracking system enables the sensation of presence – the feeling as though you're actually there.

Integrating with Leap Motion we enable using our hands in a virtual world just like in real world.
With wide-angled lenses that extend beyond the latest VR headset displays, the Leap Motion Controller can see your hands before you do.

**Unity** is a cross-platform game creation system developed by Unity Technologies, including a game engine and integrated development environment (IDE). It is used to develop video games for web sites, desktop platforms, consoles, and mobile devices.

Unity allows specification of texture compression and resolution settings for each platform the game supports, and provides support for bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion (SSAO), dynamic shadows using shadow maps, render-to-texture and full-screen post-processing effects. Unity's graphics engine's platform diversity can provide a shader with multiple variants and a declarative fallback specification, allowing Unity to detect the best variant for the current video hardware; and if none are compatible, fall back to an alternative shader that may sacrifice features for performance.

The game engine's scripting is developed in Visual Studio.
Programming language is C#.

## OVERVIEW


***Arm-Ageddon*** is a virtual reality game where player is located at another planet and battling the local creatures with fire effects shooting from his hands.

Using ***Oculus Rift*** and ***Leap Motion*** technologies and ***Unity's*** game engine we've developed a virtual world in which you can see and use your hands.

The Oculus Rift provide us to create our virtual world, and travel around it. Adding creatures that their main target is to attack us, while we try to eliminate all with our powers.

The Leap Motion Controller senses how you naturally move your hands and lets you use your computer in a whole new way. It senses almost every little move of hands and fingers.

## APPLICATION REQUIREMENTS

In order to use the app you must have the following:

- A Leap Motion Camera (buy here)
- Leap Motion driver (download here)
- Oculus Rift (buy here)
- Oculus Runtime and SDK (download here)
- Windows 8 and up
- A powerful PC:
    - AMD PhenomTM II or Intel® CoreTM i3 / i5 / i7 Processor
    - 2 GB RAM
    - USB 2.0 port
    - The better the Graphical card is the better the quality & performance will be
- At least 1 human hand

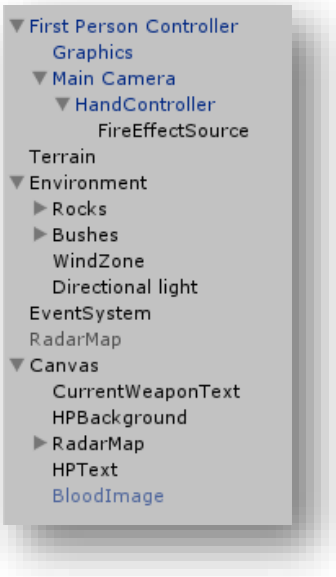
# PROJECT INFRASTRUCTURE & DESIGN

## UNITY ENVIRONMENT

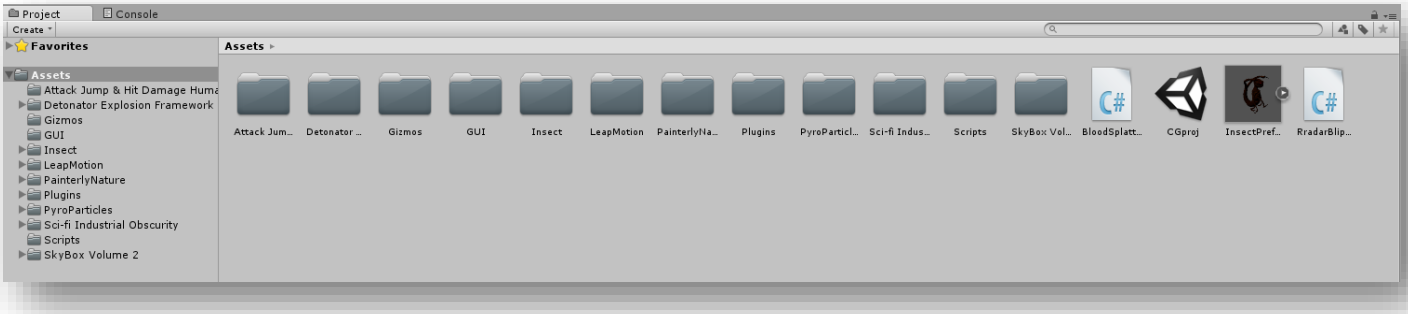
We used *Unity's* game engine to create a virtual world in which we could use Leap Motion's Hand Controller and Oculus Rift display.

The application in Unity is saved as a ***scene***.

Unity's IDE holds a hierarchy of game objects (as called by Unity's lingo) for every scene, in our application we have the following:




The scene also contians all the ***assests*** of the project which consists of every resource the application\editor\scence uses. Our project contains the following assets which are mostly self explantory:



## [GAME OBJECT] FIRST PERSON CONTROLLER

This is the main game object which represents the player. It contain a camera that is the OVR camera which rotates according to the oculus. Rotation of the camera also rotates the player, which allows him to walk in a different direction allowing the user the experience of walking in a virtual world. The Camera game object also has the hand controller game object that enables player to see his hands. When the player moves around the world his hands rotates with him and allow him to shoot in the forward direction.

First person controller behavior and properties are described in the FpsBehaviour.cs script. It also has a box collider which enables the creatures to attack and hit us, and it has the OculusLook.cs script that rotates the player according to oculus rotation.

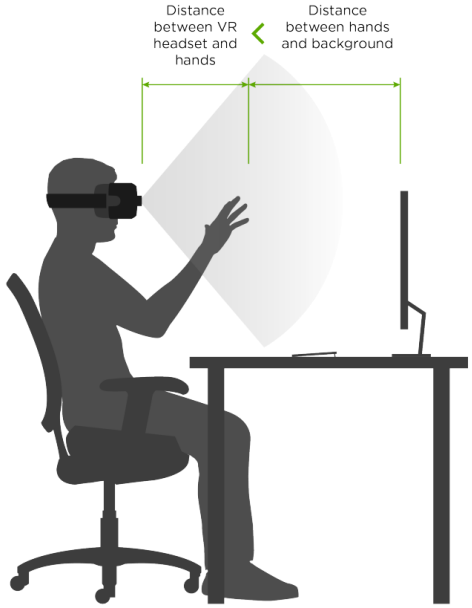
## [GAME OBJECT] HAND CONTROLLER

This game object represents all modules of the Leap Motion.

The hand controller has a Human Hands texture to give it a real look and feel.

The HandController is actually the *Interaction box* of the Leap Motion Controller (camera).

The Interaction Box is the area in which the user can controll the HandController using the Leap Motion camera. This is what it looks like:




---

## [GAME OBJECT] ENVIRNOMENT

This game object defines the virtual world we wander in. it defines all objects like trees, rocks, sky etc. The ground is a terrain that allows creatures and player to walk on and don’t fall down. Walking up a mountain as default doesn’t work because players and creatures have gravity that pulls them down. The solution was to ray cast to the ground and rotate the objects accordingly.

## [ASSET] SCRIPTS

All scripts are written in C# under the Unity namespace, mostly deriving from the *MonoBehaviour* class to allow the usage of the *Update* method and other related methods.

### [SCRIPT] CREATURE MOVEMENT

This script determines creature movement. It randomly rotates movement direction of the creatures around the world, while they always move forward. When a creature is close enough to the player, it will walk towards him and will start to attack him. When a creature gets hit by a fireball it will die, explode and disappear from the world. We use animations of the alien for walking, attacking and dying.

### [SCRIPT] FPS BEHAVIOUR

This script controls the player. It rotates him according to camera rotation and walking direction (forward or backward) determines by hand gestures. Also when colliding with a creature it will be injured. After several injuries the player will die.

### [SCRIPT] OCULUS LOOK

This script is used for updating the game camera's rotation according to the actual oculus device rotation.

### [SCRIPT] WORLD MANAGER

This script manages the game. It creates creatures in the world when game starts, by duplicating creatures and spread them over the map. When the game is over it will restart the game.

### [SCRIPT] BLOOD SPLATTER SCRIPT

This script creates a random blood splatter on the screen when an insect attacks the player. The script uses a quaternion interpolation to give the fading effect.

### [SCRIPT] EFFECTS CONTROL

This script contains 2 main classes:

#### HAND-GESTURES

This class interacts with the leap motion hand controller to hold the current gesture the player uses. The gestures we track in this project are:

- **Palm direction (facing the player or not)** – We use the palm normal and calculate the dot product between it and the camera's forward vector in order to decide the direction of the palm. We use a hardcoded threshold for the angle to decide whether to shoot or not.
- **Arm Grabbing** – We use leap motion's API to get an indication of the palm's grabbing strength. When the strength is 1 the palm is closed in a fist gesture.

### EFFECTS-CONTROL

This class controls the fireball shooting effect in the scene. We use the oculus camera's forward direction as the direction to which we shoot the fireballs. The fireballs are a Unity Prefab which is instantiated at runtime. The fireballs start from the palm position and advance forward until they hit another object or travel too far away.

## PROJECT DIFICULTIES

- **Be able to wander in the VR while oculus is your eyes**
  We needed to synchronize the camera to rotate according to oculus rotation. At first there was an offset and rotation wasn't smooth. The solution was to take oculus rotation, perform interpolation to cameras rotation any apply each frame. Also for the person to move in the forward direction of the oculus, we also applied the same interpolation to First Person Controller, which is the object of the person in the game. The trick is to work with inherited objects correctly in unity.
- **Be able to see hands controller in front of eyes**
  We spent a lot of time to position the hands in front of the oculus camera to make it feel real, and also comfortable. Leap's field of view is very small, and also hand have to be in front of the camera to work properly, but this way hands get tired so we were able to make the best out of it and find the perfect angles to work with.
- **Make creature walk around the terrain randomly**
  Creatures are created randomly when game starts. The problem is that we wanted to make them wonder in many directions around the world. So we made each creature rotate in a random angle every 5 seconds, and then go straight for 10 seconds in the direction, this way they are spread around the world. We had a problem when they wanted to go up a hill, they were falling down because they have gravity. The solution we applied for this problem is to update their up direction according to the up direction of the terrain, so they could walk up a hill, and won't roll down.
- **Use hands to create fireballs**
  We wanted to create the affect that fireball is coming from the hands, because Leap is attached to the oculus moving the head will move Leap also, so field of view is always moving, so what we did is to fire according to the forward direction we look, and move hand position according to oculus direction created the effect the fireball comes out of the hands.

## KNOWN ISSUES

- Fireball doesn't come out of the hands exactly so there is a slight offset, this is why if we look at a certain direction and hands directed to another the shooting will be in the oculus direction.
- Creatures may sometimes get stuck in the ground.

## WHAT'S NEXT?

The main goal of this project was to create something for a VR environment. We used Oculus Rift1 and LeapMotion and Unity for this purpose. During the process we've learned the capabilities of each of these component and are aware of their limitations. With this we have thought of a few ideas on what to do next with this project:

1. Extend the gaming options
    a. Add GUI to Oculus screen
    b. More weapons
    c. Add hand Menu like this
    d. Distribute "Super-Power-Boxes" around the world so that when the player touchs it he will get super powers for a limited time
2. Real VR

    Add a second camera to the head mount to stream color video.

    Use the color video as the virtual reality world, i.e. the player sees the real room he is standing in. Use Oculus to provide the player with augmented reality.

    So - The leap motion will provide the virtual hand, the web-cam provides the world, The oculus provides player's view orientation and the VR world.

    For example: Player puts on the oculus – sees the room and then enemies appear in the room and attack him (Just like in the TV series 'Mesudarim' - Video (start at 1:20).

3. Use a 'user-facing' camera with hands recognition \ face tracking like Intel's RealSense SR300 and create a race car game or something similar