Google Cardboard & Virtual Reality

EEC 687

PROJECT PROGRESS REPORT 2

Jaimal Bhairavia | 2668554
Arpit Patel | 2657673

October 12th, 2016

Applications on Virtual reality:

Upon rummaging around in the app store we came across a few existing VR applications which were games, street views and video stories. VR games like the 'VR X-Racer' available in the iOS app store, are a great example of what can be achieved with the unity software. The game has a spacecraft in the first person view and one has to navigate it through obstacles and collect few bonus points. Naming other applications that were explored: google street view, Sisters, Spacy, Roller coaster. Google street view has a set of available views from which one can choose and experience everything in the scene pivoting on one position. Sisters is a video story in VR. Spacy is again a game, set in a sci-fi environment and one has to navigate through different scenarios. Roller coaster is a simple roller coaster ride in VR.

Essential Unity concepts [1]

Unity allows us to start with a blank canvas, infinite in nature. It offers a humungous range of 2D and 3D objects in its pallet which can be utilized to build up a scene. A scene is a series of objects in the workspace, each having their respective properties. The major components of a scene will be covered in the following section.

Following are the most important components of Unity:

- **Assets**
  Unity is using many different kind of assets for game developments and those are working as a backbone of game development. Assets could be a simple graphics created with some color, texture & image, it could be a 3D object having all sort of effects like moving, rotating, with some gravity or friction.
- **Scenes**
  Scenes contain multiple objects, graphics and functions. A game can comprise of a set of scenes. Hence, with increase in the number of scenes in a game, the complexity of the game increases.
- **Game Objects**
  The GameObjects are the most important type of object in Unity. Every object in a game is called GameObjects. It requires implementation of special characteristics before it becomes a character in game. The object can hold multiple objects under itself. All the objects have one attribute in common called Transform which contains information like position, rotation, scale and order.

- **Components**
  Components are used for applying behavioral actions to objects and specific object movement when a particular event occurs. Apart from behavior, we can also give record and modify the functionality conditionally when a set of events occur in order. There are inbuilt components in Unity like rigid body, cameras, coordinates, lights, effects, etc.
- **Scripts**
  Scripts are an essential part of development because all the properties of the objects cannot be assigned statically and some event occurrence depends on previous events. For e.g. in the X-Racer game, the animation effect of spaceship destruction on hitting an obstacle is a chain of events with dependencies. Here scripts come in to picture. Unity allows us to control behavior proceeding an event by writing a code in C# or Boo (language based on concept of Python lang.) Unity provides built in behavior class which is a set of scripting class which will automatically called upon some event. Unity has its own stand-alone script editor.

Exploring 3D with Unity [1]

- **Coordinates:**
  As we know that we will be using the 3D view for the screen which is just an addition of a z-axis. Z-axis is nothing but depth of the given object. In 3D application, every object is going to have a specific location on the surface and it is going to be assigned by coordinates in the (x, y, z) format.
- **Local Space Vs Global Space:**
  It is crucial to understand the difference between local space and global space, in 3D package we can work till an infinite point and so most of the time it becomes difficult to track all of the objects in our workspace. The position of all the object have relative location to world zero, which is usually called the local space of the object. Local space of object also has its own zero point in the terms of origin of three axes. For local object the origin is always a center of the object because it's become more feasible while calculating distance between object.
- **Vectors**
  3D vector is nothing but a line which have a direction and length. Vectors are relocatable in the world space but they remain constant. Vectors are mostly used in the game development as they allow us to calculate the distance and angle between two objects.
- **Cameras**
  Cameras act as a viewport for screen and they can be placed at any point in the world. All kind of effects like lighting, motion blur and animated effect can be created by varying the properties of the camera. Unity also allows different kind of cinematic effects that the human eye has never experienced before. A set of different

cameras when seen together can make a great visual which can be used in modern games these days. Cameras can also be used for render control of particular element.

- **Polygons, Edged, vertices and meshes**
  Polygons are nothing but individually created 2D objects those can later be combined to create a 3D object. When we import the modeling object, Unity converts all polygon to polygon triangle which has three vertices that are connected together by point on space. Game engine makes calculations about points of impact, using local space of each object that can be later used in the collision detection event. Whenever we are using many linked objects, game engine allows us to create complex shapes like meshes. Developers need to calculate the polygon count because as number of polygon increase it also increases the computation time and rendering calculation as well.
- **Materials, textures and shades**
  Materials are the visual appearance of objects that can be varied from simple color to complex surface or reflective image. Combination of one more images is called a texture, including shades which states the style of rendering object. Unity provides access to an exclusive app store kind of market where in one can download desired materials, skyboxes and other materials and aesthetic detailing components. We can also create our materials, textures and shades from a simple image or using the predefined assets available in Unity.

  Rigid body physics
  As objects in a game are strongly connected with real world objects in terms of depicting them, Unity uses Nvidia's PhyX engine. The game objects which are directly connected with physics requires high computation power and processing speed. There is separate physics engine which controls all rigid body components which may have some properties like Mass, Gravity, Velocity and Friction.
- **Collision detection**
  Collision of object become mandatory when we talk about game because there are moving objects in world space. In Unity we can assign Collider component which has invisible net on edges and reports if any collision occurs. For e.g. in the X-Racer application we used, whenever the spacecraft collides with an obstacle, the collider component of that wall reports a collision and further action is taken on the event.

Interface [1]

Unity provides us a customizable working environment that can be altered according to developers' flexibility including drag and drop operational ease of components. The interface is visible in the following image

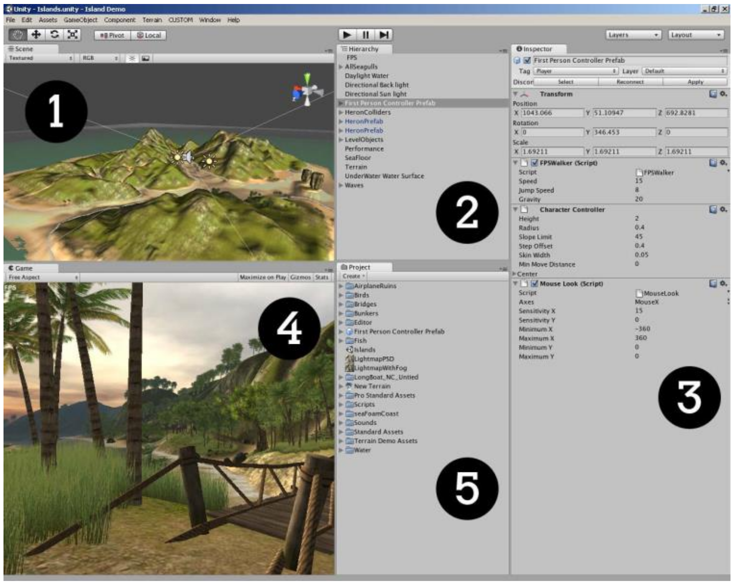
1) Scene

2) Hierarchy

3) Inspector

4) Game

5) Project.



FPS
AllSeagulls
Daylight Water
Directional Back light
Directional Sun light
First Person Controller Prefab
HeronColliders
HeronPrefab
HeronPrefab
LevelObjects
Performance
SeaFloor
Terrain
UnderWater Water Surface
Waves

☑ First Person Controller Prefab
Tag Player Layer Default
Discon Select Reconnect Apply

Transform
Position
X 1043.066 Y 51.10947 Z 692.8281
Rotation
X 0 Y 346.453 Z 0
Scale
X 1.69211 Y 1.69211 Z 1.69211

☑ FPSWalker (Script)
Script FPSWalker
Speed 15
Jump Speed 8
Gravity 20

Character Controller
Height 2
Radius 0.4
Slope Limit 45
Step Offset 0.4
Skin Width 0.05
Min Move Distance 0
Center

☑ Mouse Look (Script)
Script MouseLook
Axes MouseX
Sensitivity X 15
Sensitivity Y 0
Minimum X -360
Maximum X 360
Minimum Y 0
Maximum Y 0

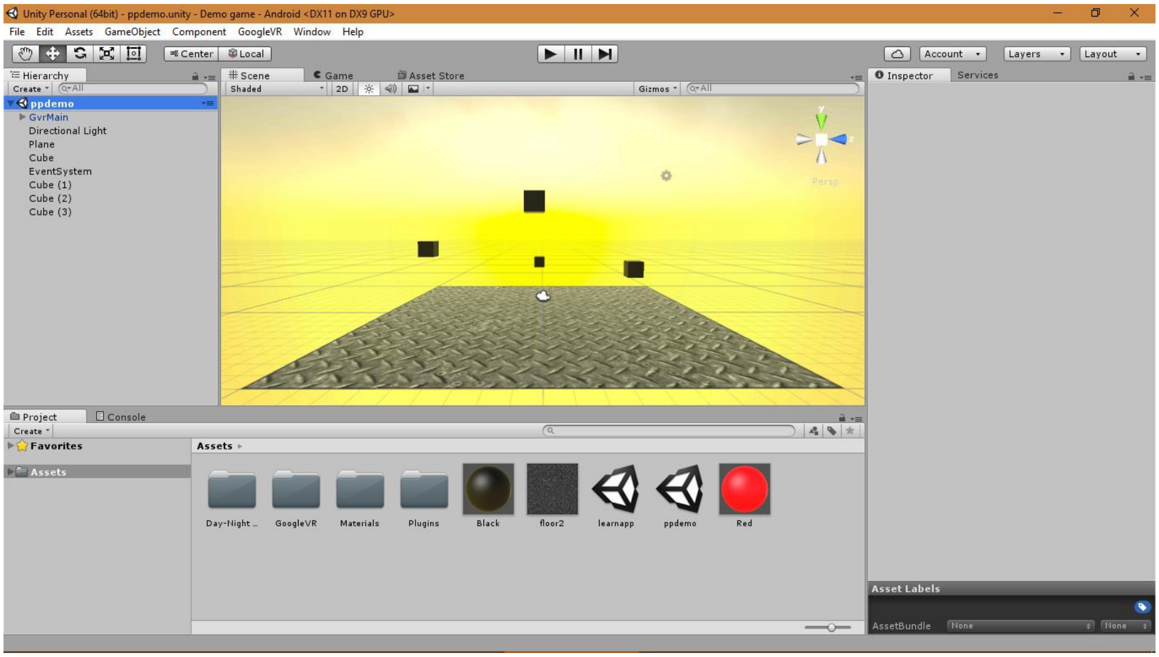
AirplaneRuins
Birds
Bridges
Bunkers
Editor
First Person Controller Prefab
Fish
Islands
LightmapPSD
LightmapWithFog
LongBoat_NC_Untied
New Terrain
Pro Standard Assets
Scripts
seaFoamCoast
Sounds
Standard Assets
Terrain Demo Assets
Water

- **The scene window and hierarchy**
  The scene window is a place where we start building our project which is tied up with the hierarchy. Hierarchy shows all currently open scene in alphabetical order. This has four control buttons, hand tool which allows navigation of scene window including drag and drop of panel from one place to another place. Second is translate tool which shows active selection in current scene window. Third is rotate tool which works as same as translate tool works and scale tool used for adjusting the size and scale of the object.
- **The Inspector**
  It's a tool for inspecting the properties of the elements like input box, slider scale, buttons and drop down menus and some of them are available from the Unity's drag and drop functionality. It also allows edit option for accessing some properties and preferences. The object features two components, transform and animation
- **The Project Window**
  Project window contains the directory structure of project it also has parent and child folder view. Every Unity project has three subfolders named Assets, Library and temp folder which only activates when Unity editor is running. If we add some assets in the assets folder then we will able to see in main window.
- **The Game Window**
  This window will activate when we click play button and it will run the project. It allows us to adjust the size of the window and let us experience how the game would look like and how it would work on the hardware it is supposed to be executed on.

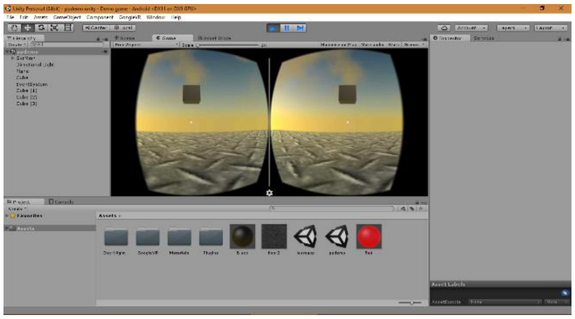
Work Completed:

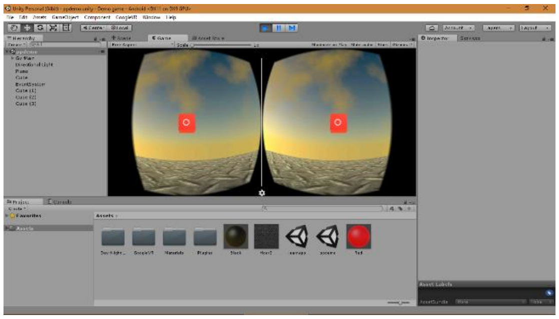
We were able to delve into the concepts of Unity and see for ourselves how things work in the software. The cardboard SDK was imported into unity and worked upon. The assets and prefabs of the SDK were used to form the base of the application. Further, components like the GVR Main and GVR Rectile of the SDK were used to implement main camera views and functions like the tracker in the center of the screen. All the attributes of the software described in the above section and the cardboard SDK components were used to create a demo application.

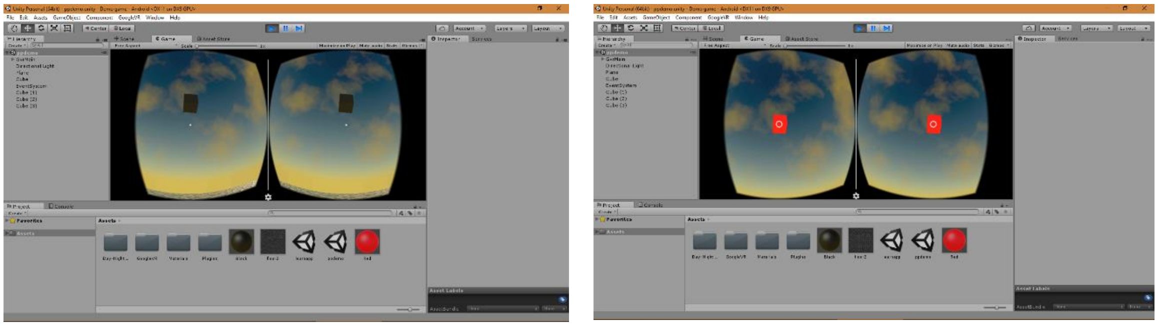
Following are the images from the demo application.



Project Window







Game Windows

We personally feel the most interesting attribute of the cardboard SDK is the rectile function and it can have a variety of implementations. We would be trying to use this function in the application we plan to develop.

We tried to export this demo application to an android mobile phone but couldn't, due to technical issues that we are trying to resolve.

Future Work:

(Week of 10/12 – 10/19)

- Solve the issue of exporting the application to the mobile phone
- Export the application to an iOS device
- Check for desired performance on the mobile device
- Evaluate possible application ideas

(Week of 10/19 – 10/26)

- Finalize the structure of the actual application to be developed
- Start building the application in unity
- Implement the basic attributes of cardboard SDK
- Analyze components of the application to be coded in C#

References:

1) Will Goldstone. 2009. Unity Game Development Essentials. Olton Birmingham, B27 6PA, UK