UNIVERSITETI I EVROPËS JUGLINDORE
УНИВЕРЗИТЕТ НА ЈУГОИСТОЧНА ЕВРОПА
SOUTH EAST EUROPEAN UNIVERSITY

INTRODUCTION TO VR DEVELOPMENT WITH UNITY

EXPLORE THE WORLD OF VIRTUAL REALITY BY BUILDING SIMPLE VR PROJECTS

AUTHOR
XHEMAL ZENUNI

VIRTUAL TECHNOLOGIES
@WESTERN BALKAN UNIVERSITIES

Co-funded by the
Erasmus+ Programme
of the European Union

UNIVERSITETI I EVROPËS JUGLINDORE
УНИВЕРЗИТЕТ НА ЈУГОИСТОЧНА ЕВРОПА
SOUTH EAST EUROPEAN UNIVERSITY

INTRODUCTION TO VR DEVELOPMENT WITH UNITY

Explore the world of Virtual Reality by building simple VR projects

Author:
Xhemal Zenuni

Co-Authors:

Lejla Abazi Bexheti

Visar Shehu

Arbana Kadriu

Shqipe Salii

The textbook was prepared in the scope of the project:

ACCELERATING WESTERN BALKANS UNIVERSITY MODERNIZATION BY INCORPORATING VIRTUAL TECHNOLOGIES

(VTECH@WBUNI)

Published by:
South East European University, Tetovo

Editor: Burim Ismaili

Printed by: Arbëria Design, Tetovo

Distribution: 120 copies

Printed in Republic of North Macedonia

First Printing Edition, November 2022
ISBN: 978-608-248-052-7

CIP - Каталогизација во публикација
Национална и универзитетска библиотека "Св. Климент Охридски", Скопје

004.946:004.455-049.8

ZENULI, Xhemal
Introduction to VR development with unity : explore the world of virtual reality by building simple VR projects / Xhemal Zenuni ; co-authors Lejla Abazi Bexheti ... [и др.]. - Tetovo : South East European University, 2022. - 57 стр. : илустр. ; 24 см

Други ко-автори: Visar Shehu, Arbana Kadriu, Shqipe Salii. - Библиографија: стр. 57

ISBN 978-608-248-052-7

а) Виртуелна реалност -- Информатика -- Апликативен софтвер -- Развој

COBISS.MK-ID 58787333

Table of Contents

Introduction to VR Development with Unity

1
INTRODUCTION TO VIRTUAL REALITY

Virtal reality is a technology which allows a user to interact with computer-generated 3D worlds in completely new ways, whether that environment is simulation of the real or an imaginary world. By wearing a head-mounted display such as Oculus Rift, a user not only can view stereostopic 3D scenes, but it can interact with it in totaly new ways, such as look around the scene by moving the head, move around it by using different sensors and generally engage in immersive experiences that feel very real and can have profound impact on people.

Therefore, the VR technology is growing rapidely into new distinct discipline in world of the IT with wide applications in different fields, such as car design, construction, architecture or biology among others, thus putting VR skills developers in high demand. In this race, despite technological advancements at hardvare level supporting VR, building development skills for this new paradigm becomes important nowdays.

This manuscript is compiled for those who are interested in learning the most funadamental concepts and techniques in developing VR through Unity3D ecosystem.

VR Games or VR Applications

Gamers usually are early adopters of high tech graphics technologies and experiences, including highly interactive 3D environments such as different virtual reality based games. Therefore, it is not a surprise that initial products from VR emerged from the gaming industry, and majority of demos from producers of VR hardware such as Oculus Share are VR based games, as they are most enthusiastic users and advocaters of this technology.

However, virtual reality technogoly gains serious ground outside gaming industry as well. In non-gaming VR applicartion the focus is more on the interactive experience or to the application goal and less about winning. Some serious industries, such as healthcare, mechanical engineering, architecture and civil engineering have adopted this technology to offer controlled environments for mimicking real life scenarios with little risk or no risk at all for the person in it and are enjoying the benefits that VR technology provides.

One can think of several ways how VR technology can be used. It can be used for entertainment or gaming purposes, but the same paradigm can have also the potential in the industry apps with great potential to boost bussines growth.

Types of VR Experiences

There are many types of virtual reality experiences, which can be broadly divided into non-immersive, semi – immersive and fully – immersive simulations. Non-immersive simulations are one of the oldest, and they are often not considered as of type VR, because they are common in our everyday life. In this kind of interaction, the user is in its physical space interacting with a virtual one. Most of the classical games and apps fall within this category.

On the other hand, semi – immersive experiences give the users the perception of being in a different reality and they provide users with a partially virtual environment to interact with. A virtual tour or the simulation to flight an airplane can be of this category as the instruments of the pilot are real and the screens in front can display virtual content. This type of virtual environment provides a visual experience, but has no physical sensations to enhance the experience.

Fully – immersive experience ensures a realistic virtual experience, not only related to the physical environment but also to the interaction itself. In order to provide a

fully – immersive experience, special equipment such as VR headsets, different sensors such as body detectors, gloves and other advanced interaction technologies are required.

What Skills Are Relevant in VR Development?

In order to build virtual reality games or appliactions, a number of skills and concepts are ver important, and those include:

- 3D modeling: when building for a VR experience, creation and the work with 3D space and scale is essential.
- First-person controls: there are various approaches and techniques that can be used to control the movement of the main character in the game or app, such as game controllers, head movements or gaze-based selection.
- UI controls: user interface elements such as texts, buttons, panels and so on are important in games and VR apps as well to design, to interact with them and make selections.
- Animations: can enhance the experiences in VR worlds and several techniques can be used to move and animate different objects, either automatic or those that are dependent to events in the environment.
- Physics: crticial to VR apps is to simulate real world environment where we have different forces, such as gravity or friction actiong on different objects. Users have to sense these forces in VR environments as well and make the experience more immense.
- Multiuser services: some experiences need multiuser involvement, and real – time networking and interactions can be a daunting task to implement.
- Programming: is the core glue in VR development and knowledge in software engineering and programming in C# or any other programming language is considered as an important prerequisite.
- Other technical skills: such as sound design or video editing can be a benificary skill for specific games or apps.

Summary

Virtual reality is an emmerging technology that can mean different things to different people. In general,VR is not only about games but it can have wide adoption in industry and academia for different purposes, from learning and training purposes up to including people in very interesting immersing VR experiences in different fields, such as architecture and civil engineering, automotive industry, medicine and so on.

Building a successful VR game or app requires some fundamental skills, such as 3D design or programming skills that are important into building VR prototypes. Therefore this script aims to provide some skillsets to create build and run different VR apps or games, and try to discuss how virtual reality really works.

2
GETTING SET UP FOR UNITY VR DEVELOPMENT

Before doing any VR development, we need to install not only the right version of Unity, but also a VR device that might need to be configured. As there are many VR devices and brands with their unique controllers, depending on VR device intended to be used, additional modules might be needed to be installed and configured as well.

Installing SteamVR

The SteamVR offers cross-platform compatibility for different devices and it is considered as ultimate tool to access and play VR games and content, and it provides interaction system that is used throughout the text.

To get SteamVR, we first need the Steam client software. Athogh traditionally is used to purchse and and installing games, it is grown to provide all kinds of VR libraries, apps amd software required for VR experiences and developments as well. The Steam can be downloaded from https://store.steampowered.com/app/250820/SteamVR/ and the installation process is straightforward using a Windows installer.

Everything is free, but an account is needed as pre-requisite to install and configure the SteamVR. The creation and the process of login are straightforward. Once the process is finished successfully, the SteamVR can be started with a double-click to the icon pointing to the client and SteamVR window

Installing Unity Hub and Editor

Unity can be installed on Windows and MacOs computers, and there is a limited support for Linux at the moment. Before installing Unity Hub and other related software, we need to make sure that our computer meets the minimum system requierements for Unity (https://docs.unity3d.com/Manual/system-requirements.html).

The installation process is rather straightforward and can be accomplished in few steps:

1. Downoad Unity Hub and the right version of Unity from the official web site https://unity3d.com/get-unity/download.

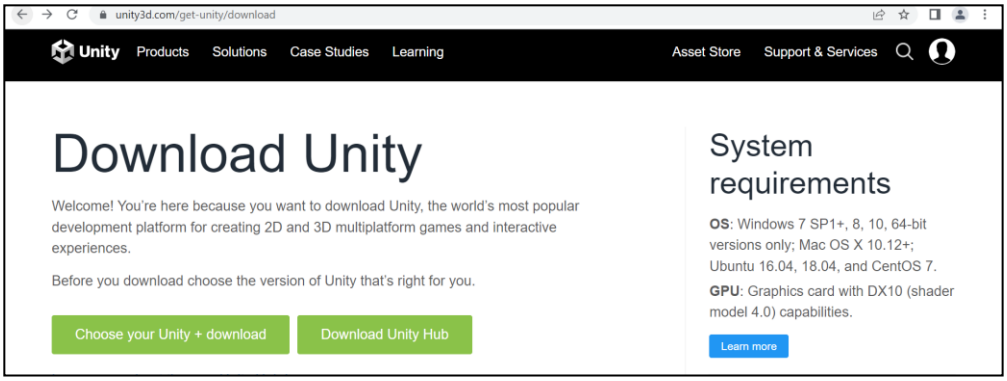

Figure 1 Unity3D Download Page

2. Select the right edition of Unity and the version for your operating system. The Unity Hub installation wizard will guide you to install the right version of Unity Editor.
3. Once the Unity Hub and Editor is properly installed, we can create new projects using different pre-defined templates for various types of games or apps, including VR.

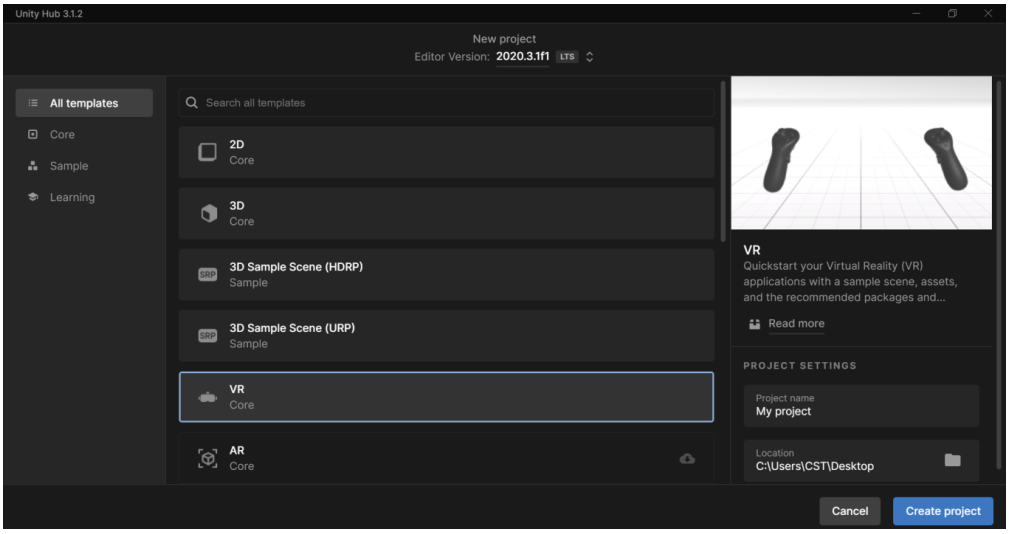

Figure 2 Unity Hub New Project Startup Page

Exploring Unity Interface

The Unity interface is made of five main windows.

1. The Scene View is the central part in Unity and offers an interactive world to visually build the game or the app. It is the place to work with different game or app objects, and do basic operations with them such as select, manipulate or modify them. So, it requires some fundamental skills that a developer needs to master when working with Unity as it includes some frequent and basic operations, such as how to position an object in space, select, scale and rotate them, and so on.

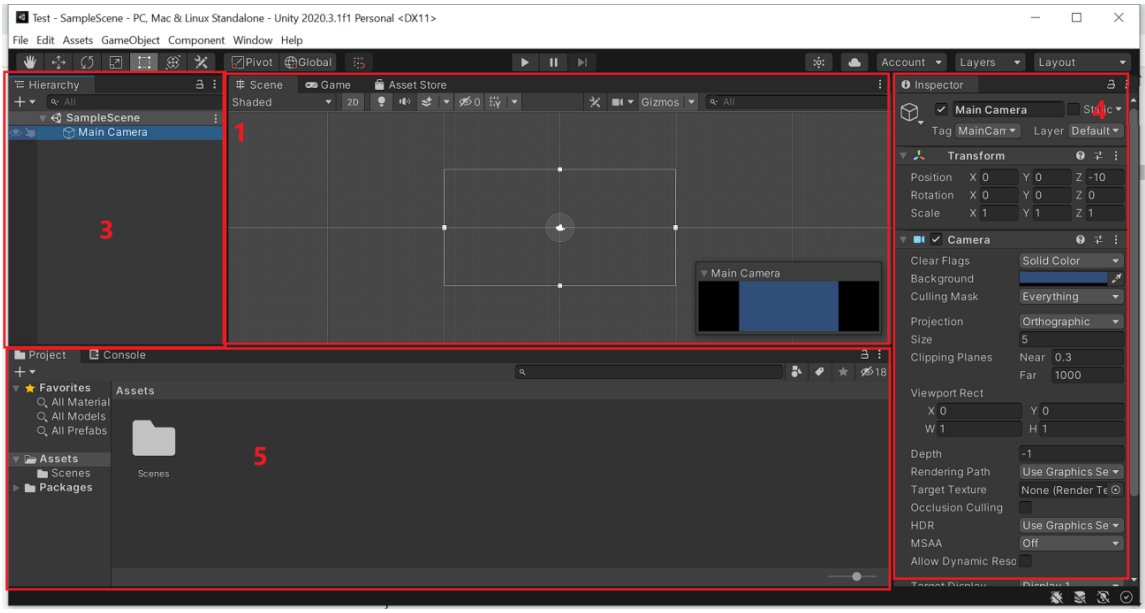

Figure 3 Unity Interface Main Windows

2. The Game View shows how the final or published product will look like. One can play and test the app or game at any point using the play button in the toolbar positioned immediately above the scene or game view.

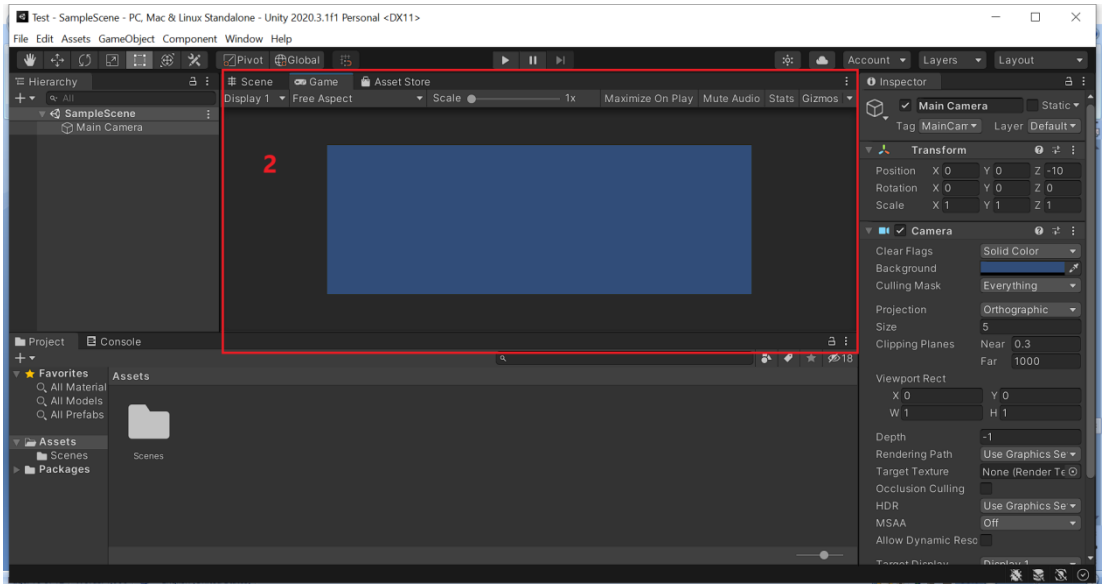

Figure 4 The Game View Window in Unity

3. The Hierarchy Window displays the list of all objects in the current sene. As objects will be added or deleted in the scene, the list is updated accordingly. They can also be organized in parent/child hierarchy. Objects can be re-organized and their hierarchical relationship can be changed.
4. The Inspector Window shows the properties of the selected object. Unity uses a Component-Based Object system, much like other Game Engines. Everything is just an empty Object first. Then things are added to it, for example a position, a rigidbody component, a script and so on. The inspector is context sensitive as properties are changed based on the selected object.
5. The project Window stores all the files that are or can be used to build the projects. This files can be of different types and can be used to build different GameObjecst out of which as commonly used formats are images, sounds, prefabs, scripts, different models, and so on. The project window allows to organize and move them around different folders, so when a project becomes larger, this part can be used to keep the project organized.

There are also few other general – use parts in the Unity environment, eapecially on the top part where the main toolbar can be found as shown in Figure 5. It is made of several tools which are useful for different purposes, out of which transform tools are essential. These tools can be used to select and transform GameObjects such as move around the scene, scale or rotate them. The hand is used to move camera and the little cross icon is used to move GameObjects.

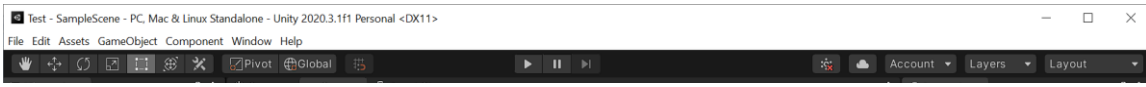

Figure 5 The Toolbar in Unity

Unity Terminology

The following is a list of words and terminology commionly used when developing games or apps in Unity environment.

GameObject are fundamental objects in Unity and they act as containers for Components such as rendering, position, scripts and other elements which implement real functionalities. By bulding GameObjects they become the most fundamental building blocks in Unity and they become camera, light, players, enemies and worlds in Scenes.

Components are elements that are added to GameObjects and they provide behavior to them such as sound or scripts and so on.

Prefabs are tpre-configured GameObjects at run time and act as templates from which we can create instances such as bullets, coins, stars or other fundamental objects important in game development.

Tranform is a component automatically applied to every GameObject. It is used to store and manipulate the information about position, rotation and scale. This component is used whenever we need to move, rotate or change the size of an object.

Tag is a label assigned to one or more GameObjects and they can be used to identify and organize GameObjects.

Layers are a tool that can be used to create a group of objects that share particular characteristics and to restrict operations such as raycasting or rendering.

Triggers are used for collision check and detection. The unity will register the collision between two GameObjects but will not do anything physically to resolve it. These events can be captured from script, so you can know when a collision happens, and react to it in your own way.

Creating a New Project in Unity

To create a new Unity project, launch the Unity Hub where a number of different options appear, including a button that allows a new project dialog box to be activated as shown in the following screenshot.

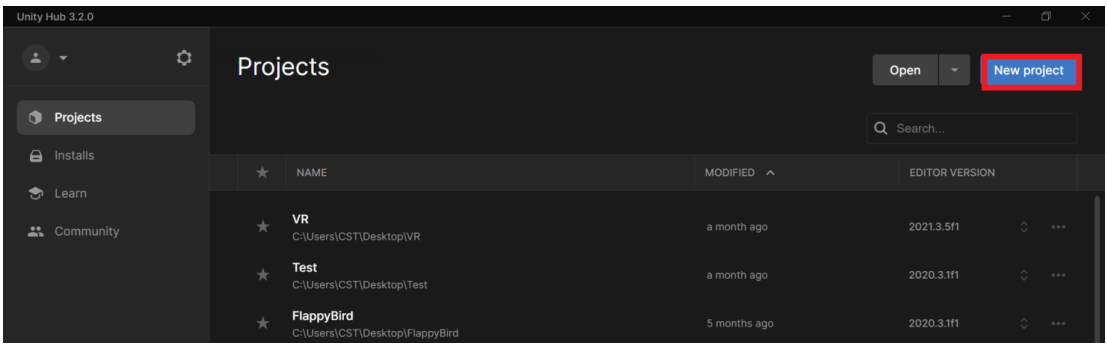

Figure 6 Unity Hub startup page

Once the button **New Project** is clicked, a new dialog box appears out which you can select a number of options and features for the new project, out which the project name and the folder location are as the most basic and common ones.

The dialog box enables a VR template based project to be selected as well and any other asset packages at this time as well, but a simple and first projects can be created using the 3D (Core) template, and additional packages and assets can be included in a later time if they are needed.

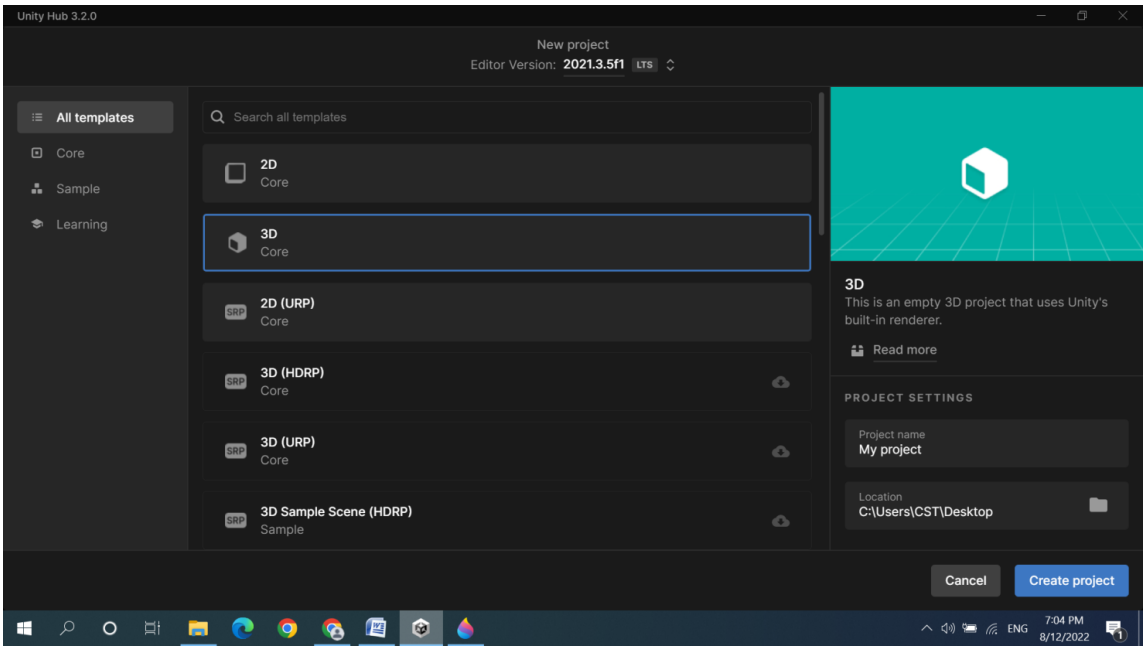

Figure 7 The create new project dialog box

A default Unity 3D scene includes a **Sample Scene** object which has a child the **Main Camera** and a single **Directional Light** component. The scene object has a reference empty ground plane grid which spans across three main space dimensions, namely across the *x, y* and *z* axes. The y axis represented with the green color is directed up as depicted in the figure 8.

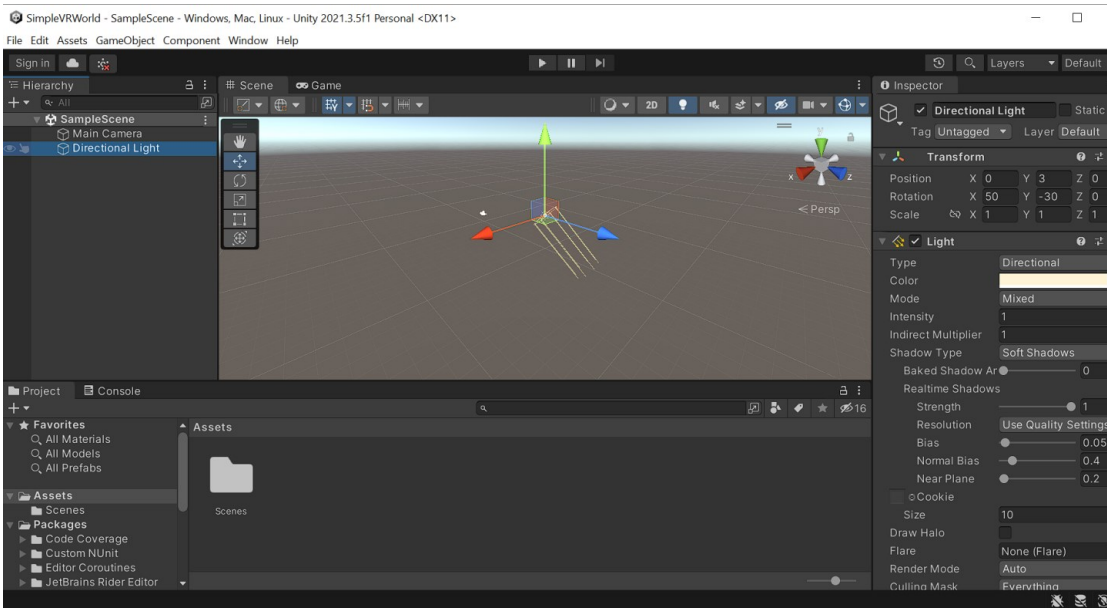

Figure 8 The representation of space axes in Unity scene

Unity is not specifically designed for 3D design, as there are other applications designed for this purpose, yet is supports the creation of some basic 3D shapes such as the cube, the sphere, the plain and so on. This enables users to create some basic objects in scene, and consequently the development of simple applications and games in 3D environments, which can be treated as the foundations of VR worlds as well.

Creating a Simple 3D World in Unity

In the following section we will create a simple 3D world in Unity, using the built-in capabilities and by adding some objects and setup the environment, such as a plane, a sphere and so on. We will add some basic UI elements as well and thus try to improve the user experience. Finally some scripts will be written to show the basics of programming in Unity to create the game functionality, see how to check for the input from the keyboard and use it for adding interatction with the users. The project is inspired from Unity learn academy (https://learn.unity.com/project/roll-a-ball), with slight modifications.

The first step will be to add a ground plane into the scene. In order to create the plane which will be used as a ground to an application or to a game, we need to use the Hierarchy panel and click on the plus sign which is located on the top-left corner. Several options will appear, out of which 3D Object category is where the basic 3D shapes can be found. By selecting Plane, a game object of type plane is added in the scene.

The plane by default is white and it is positioned in the center, that is at position (0,0,0) known as the origin of the world as shown in Figure 9.

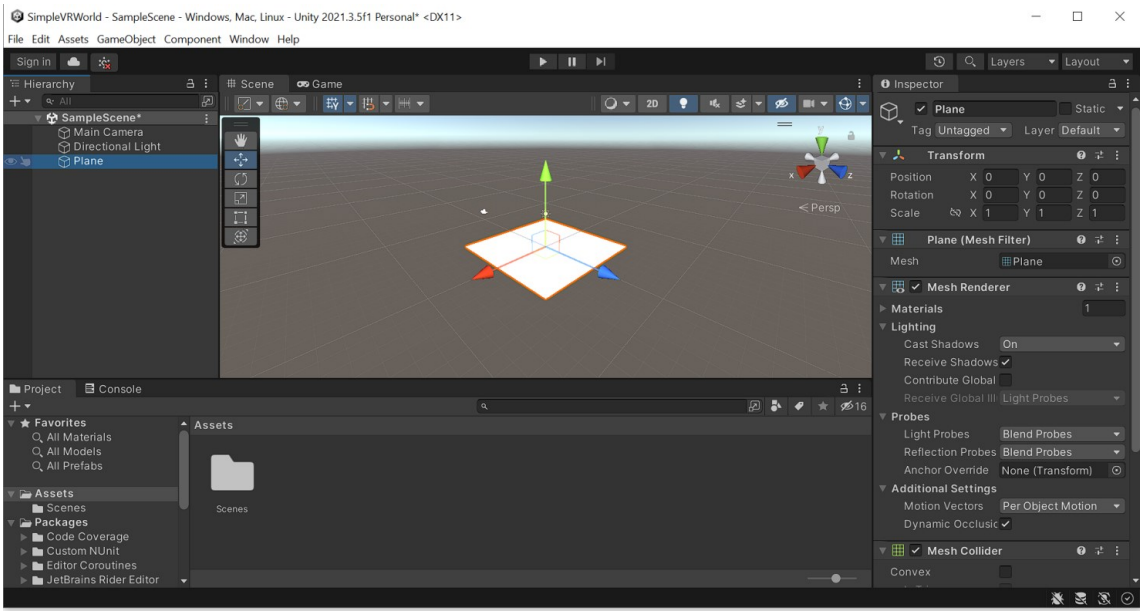

Figure 9 The presentation of plane object in scene

The plane can be scaled and transformed using the Transfrom component which can be seen at the Inspector object on the right side of the Unity environment. Since the Plane has no volume (although is considered a 3D object), so scaling along Y axis will have no visual impact (unless the value is changed to a negative number). However to have a better and more comfort working space, the values for plane scaling according to X and Z axis are changed to two.

The next step will be to create a simple sphere primitive object to the scene. To create a sphere, in the Hierarchy select the Create menu, 3D Object and then select Sphere as shown in Figure 10.

Like the plane and any other 3D object, the sphere has a radius one and with its origin at the center. At the beginning it looks like the sphere and the plane are merged together, and somehow the sphere is buried in the plane. In order to be perfectly on top of the plane, the sphere needs to be raised by 0.5 units. This time we can use the Transform component and set the y position to 0.5. We also can change the name of the plane and sphere to **Playground** and **Ball** respectively for better reference.

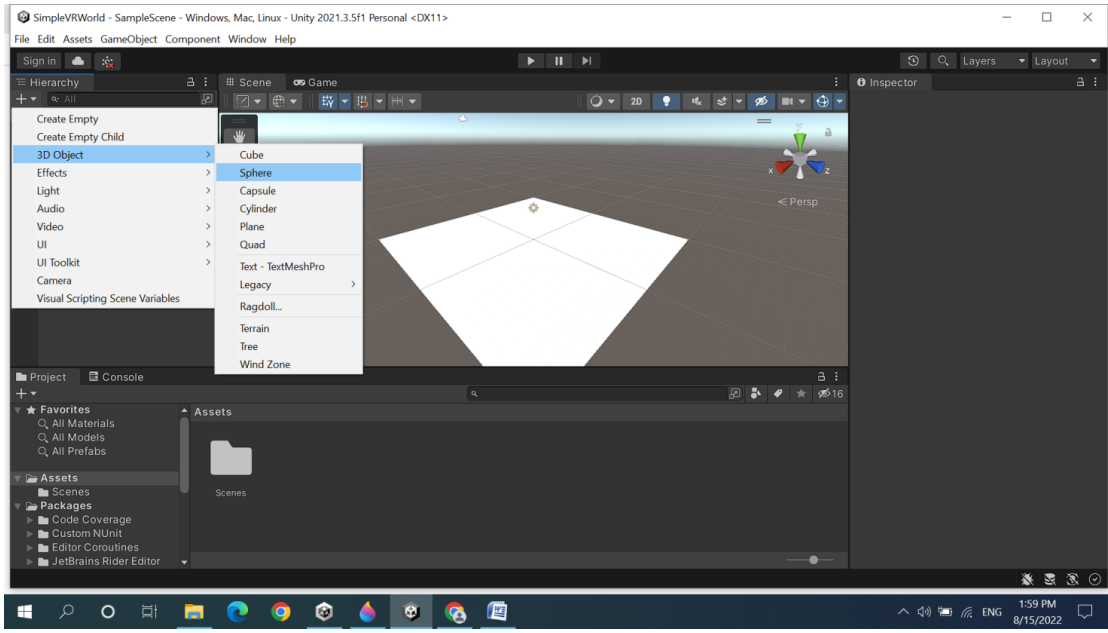

Figure 10 The creation of primitive sphere object

As primitive objects, such as plane and sphere are in white color by default, some contrast colors between them will be added. To add color or a texture to a model, one needs to use materials and apply them to objects. The following steps will be followed:

1. In the Project window, select the top-level Assets folder, select the Create menu and then folder. Rename the new folder to Materials.

2. With the Materials folder selected, use the Create menu and this time select Material and rename it to Playground.
3. In the Inspector panel, click the white rectangle to the right of Albedo which opens the Color Panel and set the RGB values to 20, 80 and 140, which is a nice blue color.
4. Repeat the precedings steps 2 and 3 to create a yellow material named as Ball and with RGB values set to 240, 240 and 0.
5. Select and drag the Playground material from the Project panel into the Playground object in the scene.
6. Select and drag the Ball material from the Project panel into the Playground object in the scene.

Once the above steps are completed, the scene looks like in Figure 11.

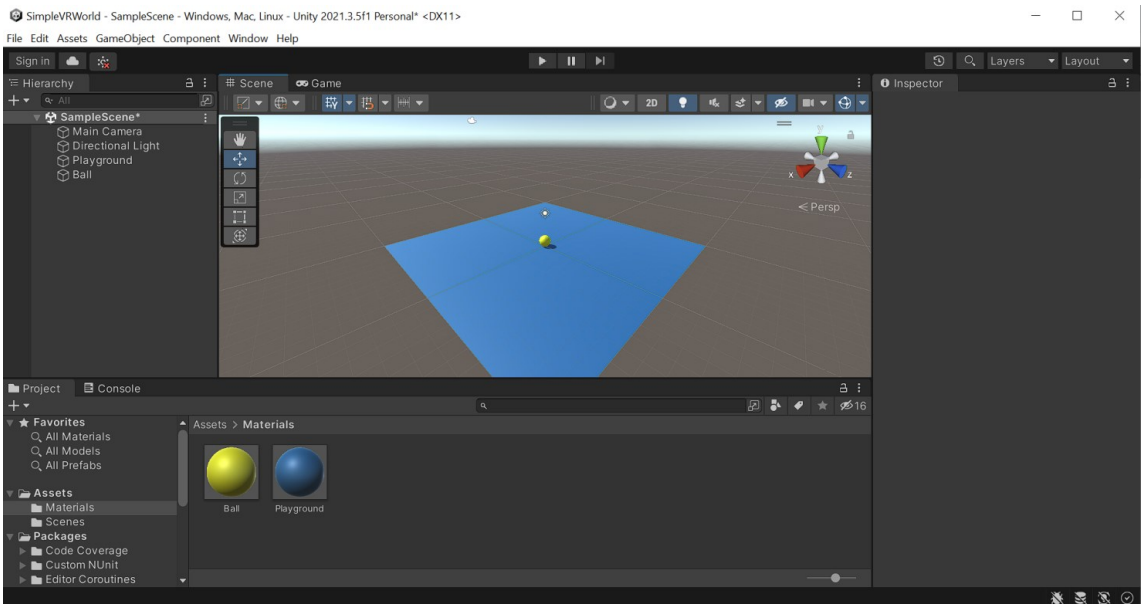

Figure 11 The materials added to primitive 3D objects

Building the Walls

In the playing area, we will place walls around the edges to keep the Ball object from falling off, and some of them as obstacles within the playground.

The creation of walls and obstacles can be created in the following order:

- To get organized, at the very beginning we will create an empy object. This object has no visual effect in the game, as it serves for organization purpose onky. In Hierarchy, we will select Create->Create Empty and rename it to Walls.
- To create the first wall, in Hierarchy we will select Create->3D Object->Cube. We will rename the object to WestWall. We will select the object and in Transform component we will reset the values to its default values. In the Inspector, we will change the cube's Transform scale x value to 0.5, y value to 2, and z value to 20.5. Finally, we will set the transform's position x value to minus 10.
- We will select the WestWall, right click with the mouse and select the option Duplicate. We will rename the object to EastWall and in we will chane only the transform's position x value to 10.
- We will repeat the duplication process again and rename the object to NorthWall. In the Inspector, we will change the cube's Transform scale x value to 20.5, y value to 2, and z value to 0.5. We will change the position as well and set the value of x and y to zero, and the value of z to 10.
- We will duplicate the NorthWall, rename it to SouthWall and change the z position from 10 to minus 10. This will be sufficient to create the surrounding walls of the playground. We will add two obstacle objects as well.
- We will duplicate the SouthWall object and rename the new object to Obstacle1. We will set scale (x,y,z) values to (10,2,0.5) and position (x,y,z) values to (0,0,5).
- The last step will be to duplicate the Obstacle1 and rename the new object to Obstacle2 and change the position value z from 5 to minus 5.

After the completion of the above steps, the playing scene looks like in the following figure.

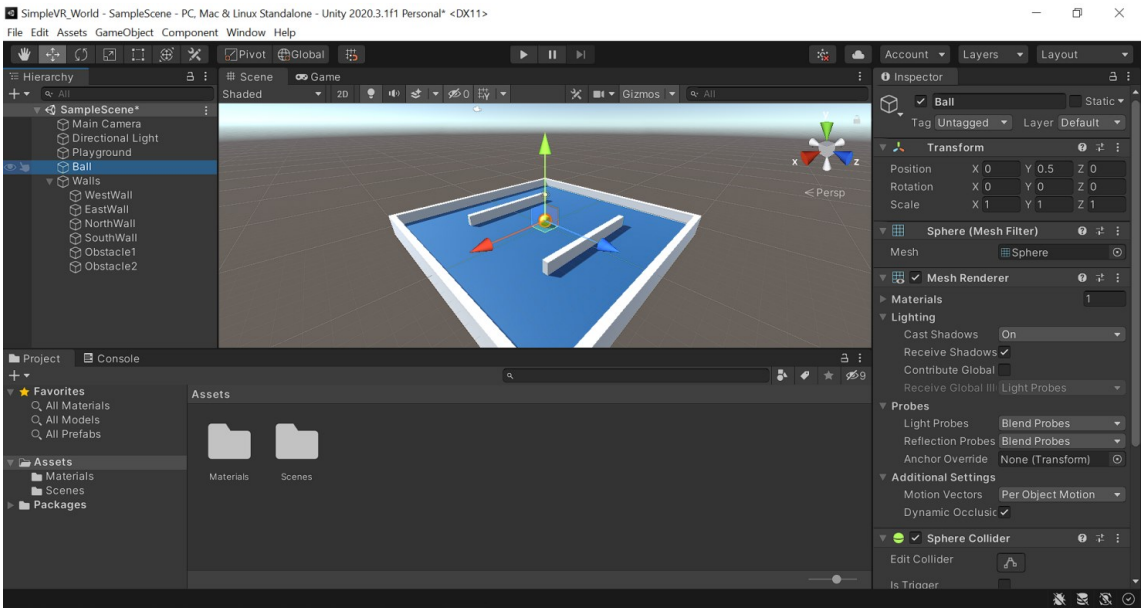

Figure 12 Adding walls and obstacles in game scene

Creating Collectibles

We will create collectible elements in scene. The idea is that the Ball object will be moved around the playground and the motion will be controlled by keyboard inputs, and it will collide and collect the collectible elements.

To create the collectible elements, we will:

- Create a cube object by going to Hierarchy and select the option 3D Object->Cube.
- The object will be named Collectible.
- In Transform component, we will set its position to (0,0.5,0), the rotation to (45,45,45) and the scaling to (0.5,0.5,0.5).
- We will create a specific Material for this object, and set its Albedo RGB color values to (0,245,1) and attach to the Collectibe element.
- In Assets window, we will create a new folder and name it Prefabs.
- We will drag the Collectible element from Hierarchy to Prefabs folder, and thus make Collectible a prefab.

- Next, we will delete the Collectible from Hierarchy and there we will add new empty object Collectibles.
- We will drag the Collectible prefab above the Collectibles object to create an Collectible instance in scene and make it as child of Collectibles. We will change the position of the Collectible element in arbitrary form, and make sure that the y position value remains 0.5, while we ensure that we put x and y values that will position this element inside the playground and with some offset to walls and obstacles.
- We repeat the above step for several times, and the final scene looks like in the figure below.

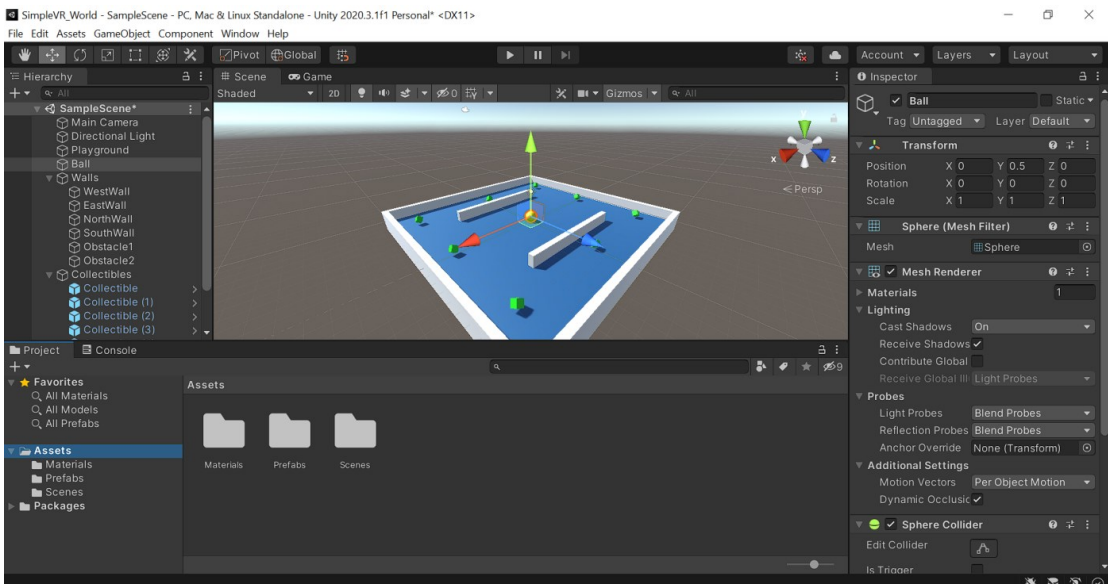

Figure 13 Adding collectible objects in the project

Working with UI Elements

Unity offers support to work with different UI (user interface) elements or objects. They are two dimensional graphics which overlay the main gameplay and present information to end users. Common UI elements include text, buttons, panels, sliders and so on. In Unity, this type of objects resides on a Canvas.

In this game, we will create a text element that will show to the user how many Collectible elements have collected. To do this:

- In Hierarchy we will select the Add button and go to UI -> Text.
- We will rename the object to CountText, and change the value of text in Text component to "Count: 0".
- In Text component we will change the font size to 24, the font color to white and set the value horizontal and vertical overflow to Overflow.
- In Rect Transform component we will select the icon will open the anchors and presets menu. By holding Shift and Alt or option and select the top left anchor point. This will position the text in that up corner.
- In Rect Transform component we change PosX to 90 and PosY to -20, and the result is better viewed if it is opened in play mode and it loos like in the following figure.

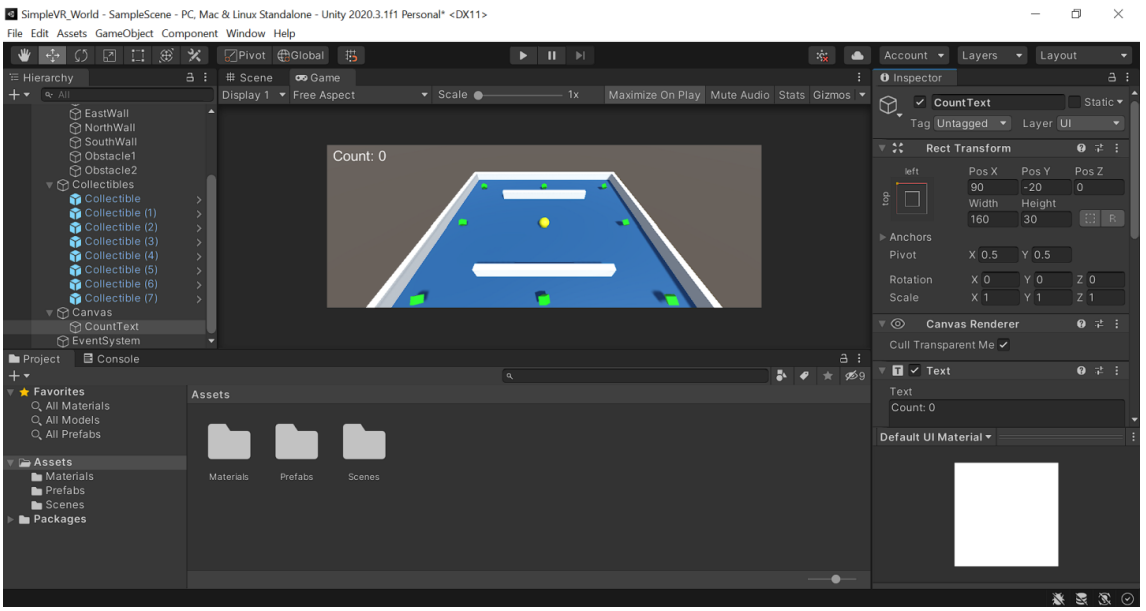

Figure 14 Adding UI Text element to keep the counting score

Moving Objects and Scripting in Unity

Objects in Unity can be moved using two main approaches.The first one is by by altering the values x, y and z in Transform component over the time. The other approach is to Unity physics components.

Unity includes a build-in physics engine that handles the physics for object interactions and effects, like the gravity, acceleration, collision detection, object fraction and so on. The items that play into physics include the following:

- The Rigidbody component
- The Collider component
- The Physic matrial and
- The project Physics Manager

Rigidbody is the key component to physics that can be added to objects and it defines some key physic parameters such gravity, mass and drag among others. Rigidbody automatically reacts to the gravity and collisions, calculates the momentum and updates object position and rotation accordingly.

Colliders are used to detect if two objects collide. In order to do this, both colliding objects must have a Collider component. There are colliders of different geometric shapes, and they are also separate categories for 2D and 3D games or worlds.

The Phisics Materials can be assigned to colliders, and they can be used to adjust the friction and the bounciness effect of colliding objects.

Unity has the Physics Manager that can be activated by navigating Edit->Project Settings->Physics. It defines some global and default values for different important physic parameters, such as gravity, bounce threshold and so on.

In the developed scene, we will define two types of motion. One will be applied to collectible elements and it will not be based on physics but rather on position, more specifically the rotation parameters to achieve a simple animation effect. And, the second type of motion will be defined for ball object and it will be based on physics and dependent on user input.

To achieve this, we will have to write scripts for both implementations. The language that is used in Unity is C#. In oreder to be called and executed, the script must be attached to a GameObject in the scene.

To automatically rotate the collectable elements, first we will position to the Prefabs folder in Assets window, and select the Collectible prefab. On Inspector we will click on Add Component button found at the bottom and then select New Script and give a name to the new script file. In this case CollectibleController was used to name the script.

The script is automatically created on Assets window and it is attached to the Collectible prefab. For better organization, we will create a new folder Scripts in Assets, and move the just created file there. Finally, we with double click on the file, the same can be opened for editing and the rotator code is as in the following listing:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CollectibleScript : MonoBehaviour
{
       // Update is called once per frame
    void Update()
    {
        // Rotate the game object that this script is attached to
by 15 in the X axis,
        // 30 in the Y axis and 45 in the Z axis, multiplied by
deltaTime in order to make it per second
        // rather than per frame.
        transform.Rotate (new Vector3 (15, 30, 45) *
Time.deltaTime);
    }
}
```

Listing 1 The rotator script

Some notes on the structure of code:

- Unity provides a very powerfull library that combined with standard C# libraries it allows fast and easy game or app development. UnityEngine is the fundamental library that has some very fundamental classes defined, such as MonoBehavior, GameObject, Vectors, Time among others.
- It has some important event functions, such as Start( ) or Update( ). Start( ) will be executed at the beginning and it is used for different initialization. On the other hand Update( ) basically is an infinite loop that is executed once per frame.
- So basically, in Update( ) event we change the rotation x,y and z values in object transform component, an this gives the effect of rotating the object over the time.

On the other hand, the motion of the Ball object will be based on physics and dependent on user input. To achieve this:

- We will select the Ball object in Hierarchy, then position to Inspector and click on the Add Component button. We will navigate to Physics->Rigidbody as in the figure.

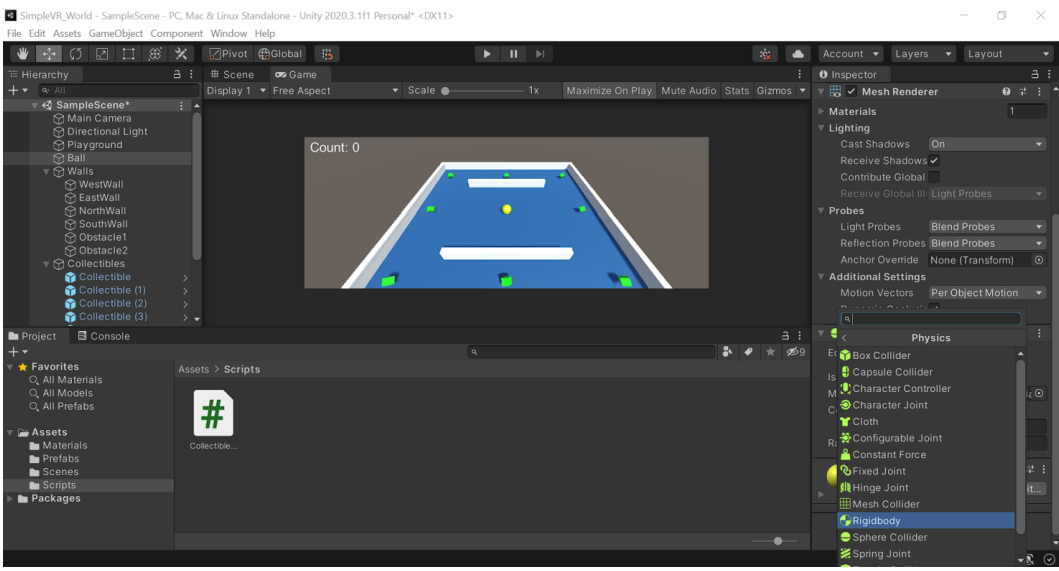

Figure 15 Adding the Rigidbody component in the game object

- We will add or attach a new script similar to the Collectible object and name it as BallController. We will move it to the Scripts folder and with double click we will open and edit the code as in the Listing 2.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BallController : MonoBehaviour
{
    Rigidbody rb;
    public float speed;
    // Start is called before the first frame update
    void Start()
    {
        rb=GetComponent<Rigidbody>();
    }

    // Update is called once per frame
    void Update()
    {
        float moveX=Input.GetAxis("Horizontal");
        float moveY=Input.GetAxis("Vertical");

        Vector3 dir=new Vector3(moveX,0.0f,moveY);
        rb.AddForce(dir*speed);
    }
}
```

Listing 2 Ball controller script

Some important notes on the script:

- A public variable speed is defined through which we can control how much speed to add to the Ball object. Since it is public, this value can be set from Unity IDE.
- The rb is used to link with Rigidbody component of the Ball object. This connection is set on Start( ) event.
- On Update( ) event we check all the time if there is any input in the keboard from the user. The Input object has a usefull method GetAxis which requires a string input, with possible values "Horizontal" and "Vertical". If the first string value is used, than the Left and Right arrow from keyboard will be checked. On the second string, the game will check if Up or Down arrow from keyboard are pressed. GetAxis method returns -1, 0 or 1 as a value depending if respective keys are pressed in the keyboard.
- moveX and moveY are used to define the direction vector of the pushing force in the Ball object. Once the direction vector is defined, AddForce method from Rigidbody component is used to define the pushing force on the Ball. In this way, the motion of Ball object is determined by user input.

Controlling the Camera

By default, the camera is fixed and it will not move from its current position. To make the camera follow the Ball object, we will need to write a script for this purpose. But, before we do this, the camera position and rotation angle need to be adjusted. The camera will be lifted by 10 and tilted down for 45 degrees as shown in the figure.

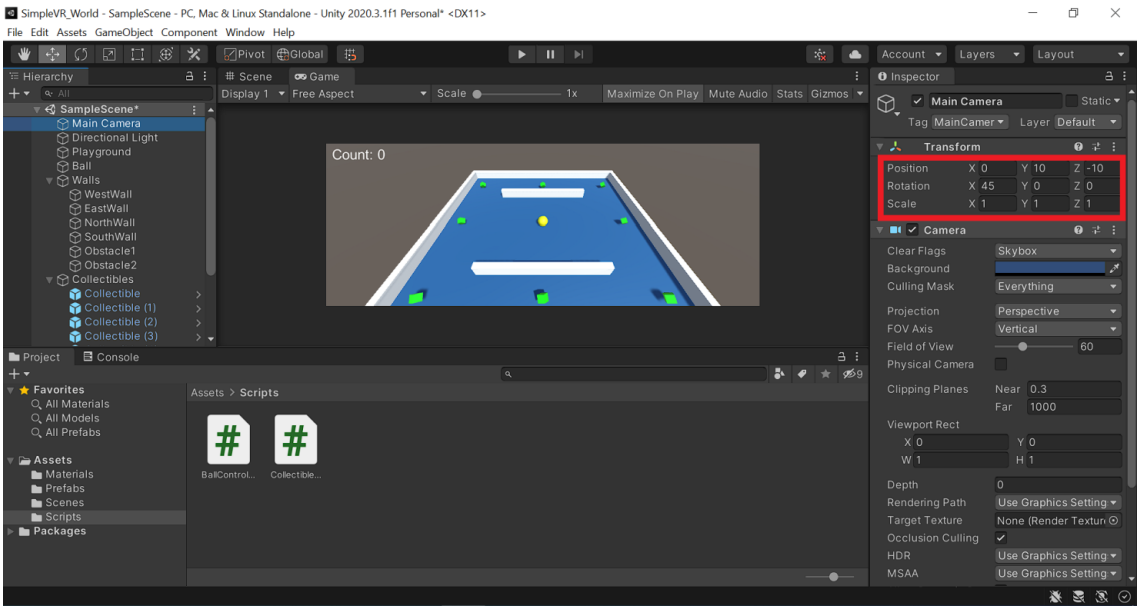

Figure 16 Setting the camera in the project

In Main camera we will add the CameraController script and we will update it as in Listing 3.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraController : MonoBehaviour
{
    Vector3 offset;
    public GameObject player;
    // Start is called before the first frame update
    void Start()
    {
        offset=transform.position-player.transform.position;
    }

    // Update is called once per frame
```

```
    void LateUpdate()
    {
        transform.position=player.transform.position+offset;
    }
}
```

Listing 3 The main camera controller script

An offset between the Ball object is defined and the same offset is preserved over the time. So the camera position is updated all the time as the Ball position plus the initial offset.

In script we have a public GameObject named as player. Since it is a public, in Unity IDE we will have to make the link with the actuall Ball object and the player variable in script. This is done by dragging the Ball object from Hierarchy and dropping in the public component on selected camera script as shown in figure.

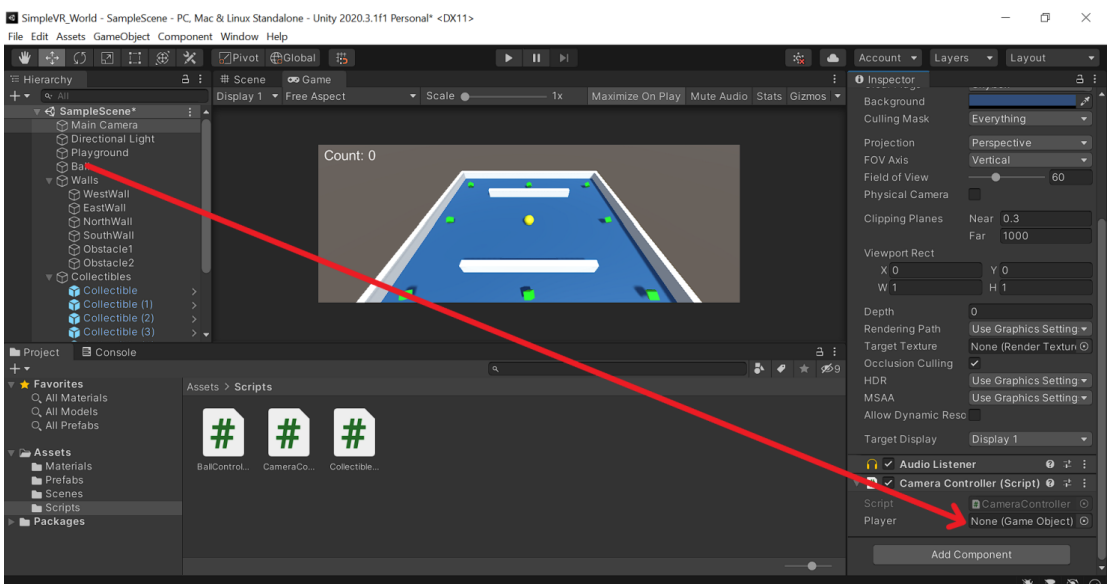

Figure 17 Linking the Ball with player object from camera script

Collision Detection

In the game, the Ball object should be capable of collecting the Collectible GameObjects and show in UI text how many of them currently has collected. For this purpose, we need to do the following settings:

- Select the Collectible prefab from Prefanbs folder, and on Inspector select Tag component and Add Tag option as in figure. Define CollectibleTag and set this value to the Tag component. Tags are very useful to identify an object or a group of objects.

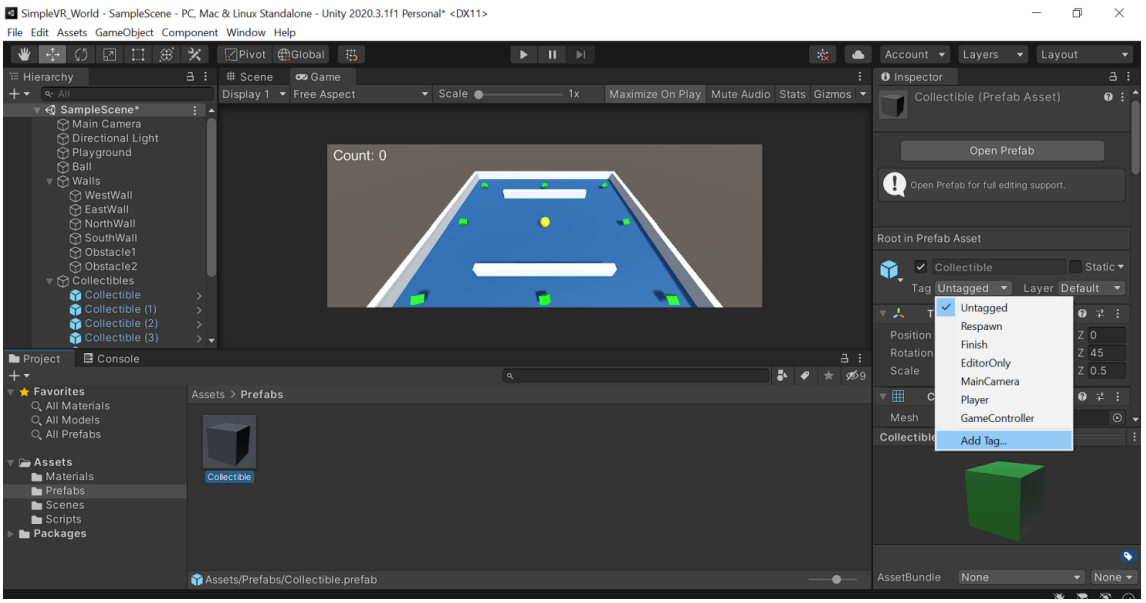

Figure 18 Defining tag for the Collectibe prefab object

- Update and extend the code in BallController script as in the listing below.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class BallController : MonoBehaviour
```

```
{
    public Text scoreText;
    int count;
    Rigidbody rb;
    public float speed;
    // Start is called before the first frame update
    void Start()
    {
        rb=GetComponent<Rigidbody>();
        count=0;
    }

    // Update is called once per frame
    void Update()
    {
        float moveX=Input.GetAxis("Horizontal");
        float moveY=Input.GetAxis("Vertical");

        Vector3 dir=new Vector3(moveX,0.0f,moveY);
        rb.AddForce(dir*speed);
    }

    void OnTriggerEnter(Collider other)
    {
        if(other.gameObject.CompareTag("CollectibleTag"))
        {
            other.gameObject.SetActive(false);
            count++;
            scoreText.text="Score: "+count.ToString();
        }
    }
}
```

Listing 3 The complete code in BallController script

- Since we will use UI element in the script, then UnityEngine.UI library needs to be included in the script.
- A public Text variable named scoreText and a counter has been defined. The first one will be linked with the UI text defined earlier, and the second one will count how many collectible elements have been collected. The link between scoreText and the UI CountText will be done as in the case of camera and the Ball object.
- OnTriggerEnter will check and this event will be triggered if there is any collision between the Ball object and any other object that has collider and has checked IsTriger property. If the object with who collides has the tag "CollectibleTag" then hide that object, and update the score appropriately.

Once the final steps have been completed, we need to run and test the project. We see that the collectible elements rotate all the time, the Ball object moves and it is controlled by keyboard inputs, the camera follows the intended object, collision check work correctly and the score is updated as expected. Figure below shows the project or the game in playing mode.

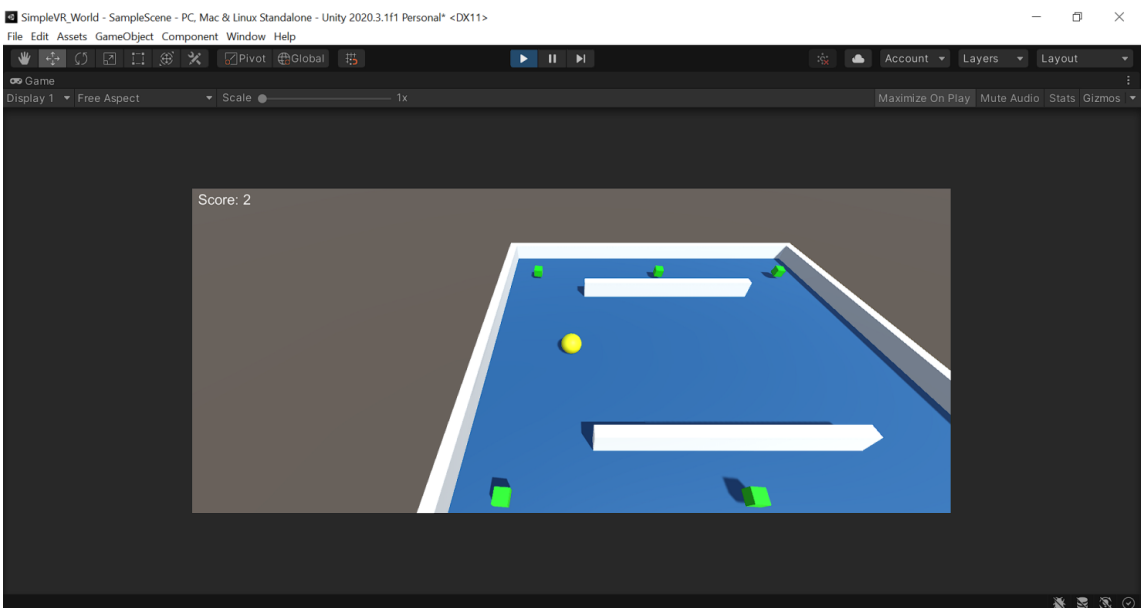

Figure 19 The project in playing mode

Building the Game

One of the great things in Unity is that it allows deploying the project to different platforms, including mobile platforms and game consoles. To create a build of the game, in top menu, we have to select File and Build Settings. A window as shown in the figure will appear.

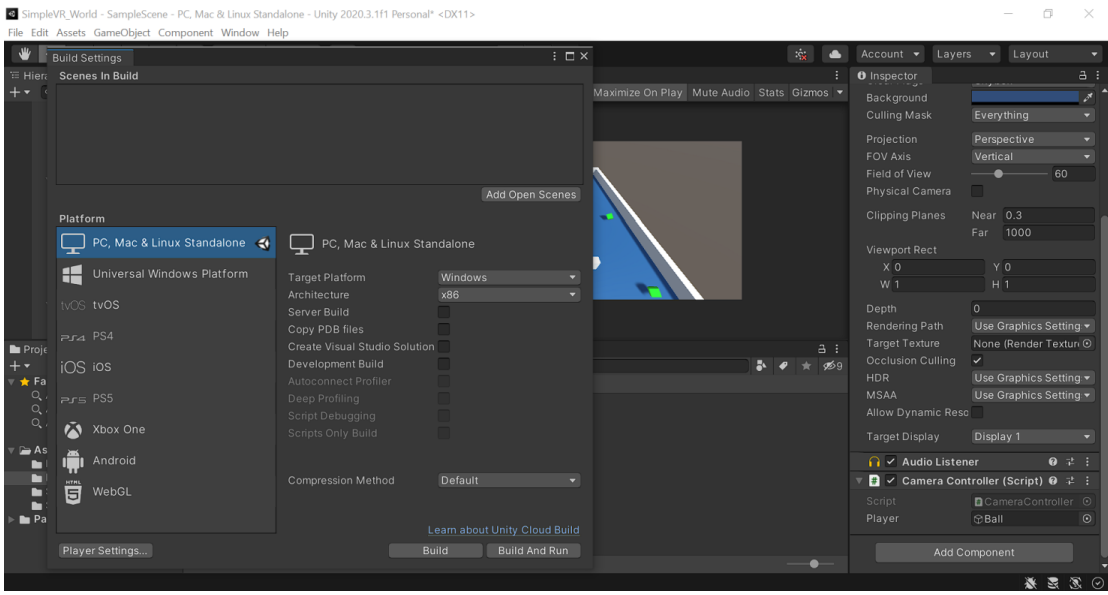

Figure 20 Build and run settings in Unity

We will add the open scene to be build and select all relevant parameters and click the button Build And Rund. This will basically create and build a simple game in Unity.

Unity support audio, animations, and a lot of other cool features, however they have not included within the scope of this text.

External 3D Modelling Software and Tools

Unity allows to build simple 3D worlds as demonstrated earlier, but it has to be underlined that Unity has limited capabilities in generating more complex 3D models that form the basis of VR development. In many situations, Unity engine

can be considered as an intergral part of 3D content creation, yet it is not optimized for the 3D modeling from scratch. Unity mainly supports the box modeling approach or technique where the designer takes basic shapes such as the box, sphere or the cylinder as a starting point and works along the process until the desired model is complete. The process is straightforward and efficient, yet it is difficult or impossible to achieve top-notch quality in this regard.

Therefore, when it comes to complex 3D modeling, a third – party modeling both paid or free need to be considered. In general, currently Unity supports mashes and animations from two different types of files:

- Files with extensions .obj or .fbx which are considered as generic exported file format. These types of files are usually smaller that the equivalent proprietary files and allow to be imported only part of the model, instead of importing the whole model in Unity. They also encourage a modular approach as it allows to use different components for collision or interactivity.
- Proprietary files such as .max or .blend files. Besides the cost, such files are generally not directly editable as well. These files can be edited, if necessary only by the software that originally created them, such as Autodesk 3ds Max or Blender.

Some of the best 3D modeling softwares around that can be used today are:

- Blender (https://www.blender.org/): it is a free and open-source 3D modeling software that has been around for a while. It has a large community of artists and other enthusiasts that supported its continous development. It has become a software of choice for creation of virtual reality content, interactive 3D content, animations, animated films and so on.

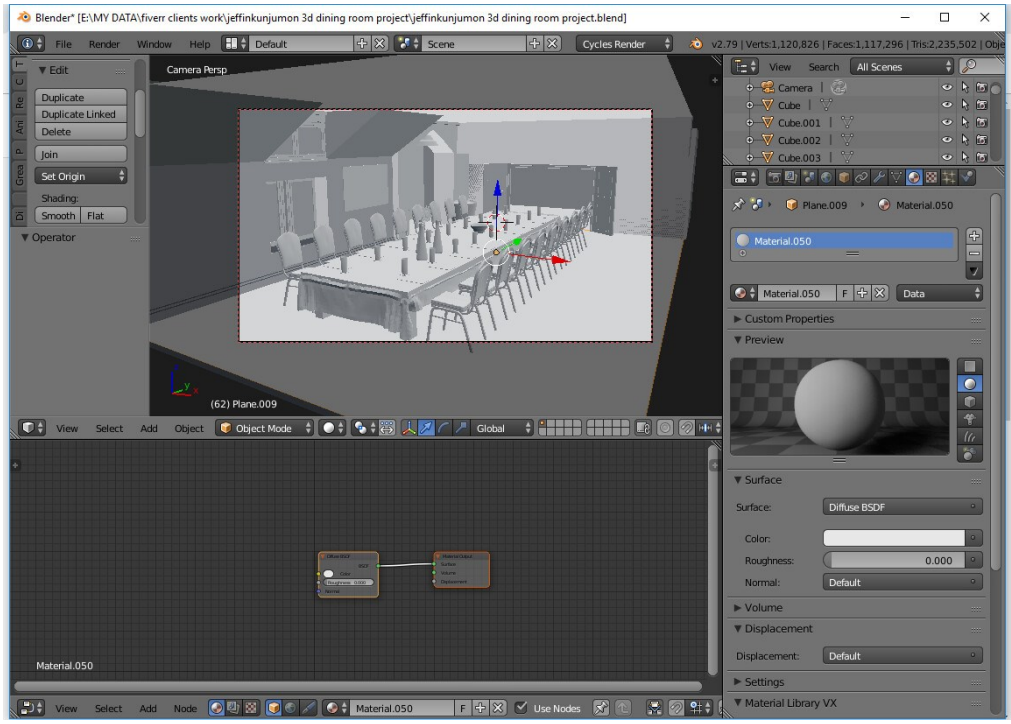

Figure 21 A sample 3D modeling project created with Blender (https://jakkijiji.artstation.com/projects/xzxzn1)

- Sketchup Free (https://www.sketchup.com/plans-and-pricing/sketchup-free) advertises itself as a very simple 3D modeling tool with no strings attached. Scetchup runs directly on web, has interoperability with different image file formats such as JPG or PNG, and currently it offers 10 GB storage for user projects and access to a numerous user-generated and manufacturer-produced 3D models.

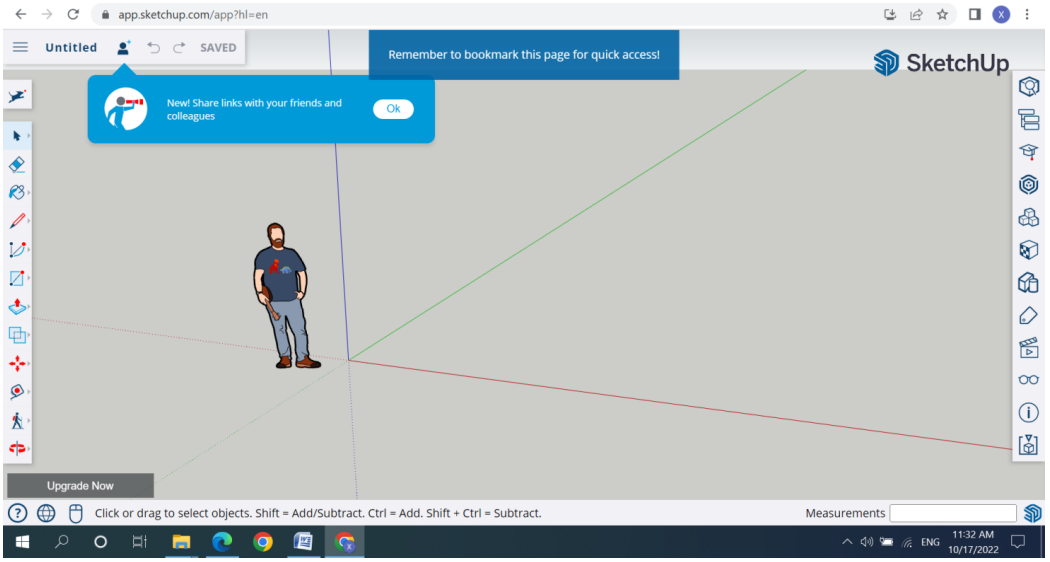

Figure 22 The web interface of free sketchup

- Wings 3D (http://www.wings3d.com/) is a free, open source and cross platform modeling tool that has been developed since 2001. The interface may look a bit unusual, but it is user friendly and once accustomed, the process of building models becomes straightforward based on polygons and the smoothing technique adopted by the tool. Wings 3D offers a wide range of modeling tools, supports lights and materials, however in current version there is no support for animations.

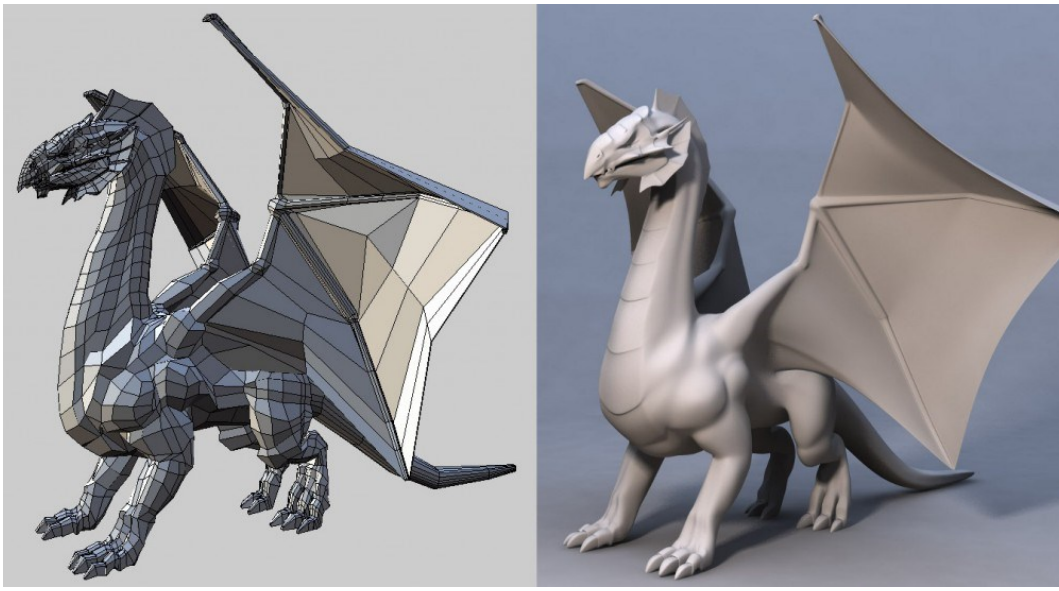

Figure 23 Wings 3D a polygon modeler

Unity suppors proprietary 3D application files as well. Althoguh a licensed copy of the software used must be installed on each machine that uses the Unity Project, which consequently creates addional cost, some individuals and enterpresises may consider such tools for creation of rich 3D models. Today, some of the best 3D modeling paid-option softwares are:

- Autodesk Maya (https://www.autodesk.com/products/maya/overview) is professional software and a standard for computer graphics disciplines, which offer to artists unrivalled set of features and tools to deliver stunning visuals. Although it is expensive and not easy to learn, a lot of artists rely on Maya to create complex characters and worlds, and dazzling effects. Maya offers vast features such as character animation, physics, realistic effects and simulations ranging from explosions to cloth simulations.

Figure 24 Overview of the Maya development environment

- Autodesk 3ds Max is another professional 3D modeling software that is used to create premium designs. The tool is considered as easier to learn compared to its sister software Maya, however it offers a wide range same of capabilities, ranging from fluid simulations to character ragging and animation.
- Cheetah 3D (https://www.cheetah3d.com/) is aimed for beginners or medium artists designed for Mac platforms. It is easy to learn tool and it offers an intuitive user interface and a wide range of tools for modeling, rendering and animations.

The list presented in this section is far from definitive, and new tools may emerge in near future. However, the foundation of every VR development relies on modeling, especially on the 3D characters and worlds.

Asset Stores for Virtual Reality Development

Creation of 3D models and worlds for VR developments can be a daunting and time consuming task. There is also a huge community that creates and shares game and other type of assets which can be used to reduce costs, complexity and the development time of virtual reality development.

Unity has its own asset store where large communities of people create and publish various types of assets ranging from simple textures, models, animations, tutorials up to complete projects. Developers have a huge catalog of assets at their disposal to speed up the application or game development.

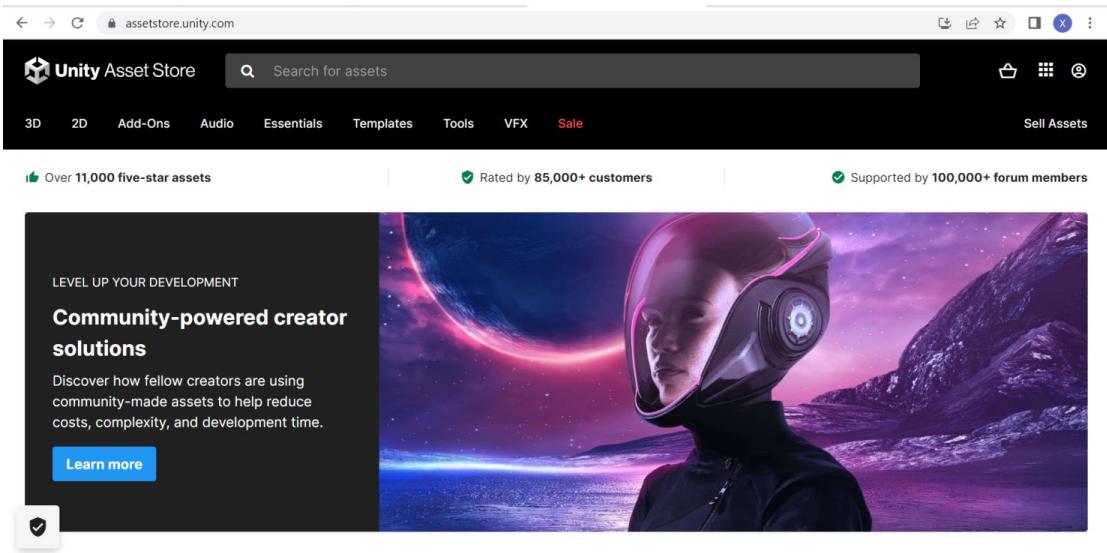

Figure 25 The Unity Asset Store

As the number of assets for VR development exceeds 500 in unity assetstore, some of cool staff here that can be baught or find for free include VR hands models mega pack, auto hand, VRIF (VR Interaction Framework) and many more.

Summary

This chapter introduces Unity as an effective environment to create virtual reality applicatons or games. It explains the steps involved in setting up the development

environment and it explores the Unity interface through 5 main windows: the scene view, the project window, the hierarchy window, the inspector window and the game view.

VR mainly consists of the 3D objects and worlds, and the user engagement with those elements through some natural way of interatctions. Unity offers some basic capabilities of building 3D objects, such as planes, spheres, cubes and other regular geometric shapes, yet it lacks more advanced capabilities to create more complex worlds and dazzling effects.

Users seeking for professional 3D modeling capabilities, such as creating realistic characters and effects, such as explosions or clothe simulation have to consider external modeling softwares, both free or paid, such as Blender, Maya, Sketchup and so on. A quick listing and overview of such tools is provided.

Finally, in certain situations, when we want to speed up the development process or reduce the cost, stores such as Unity asset store may be considered to gain fast access to a large catalog of simple textures, models, animations, tutorials and up to complete projects. Once the VR world is created, the next important task is to deal with the interaction of users in such context.

3
VR INTERACTIONS

VR development requires a user – centric approach to ensure enjoyable experiences through interactions with the VR world in a natural way or close to it as much as possible. As virtual objects and worlds are specific, a numerous specific interaction techniques for VR have been developed.

Although there is no standard classification of interaction techniques for VR, they broadly can support one of the three main actions:

- Selection
- Locomotion
- Manipulation

Interaction Techniques for Selection

In simple terms, selection enables the user to tell the system which object or user-interface element wants to interact with. Once the object is confirmed to be selected, it becomes the focus for further interactions.

Raycasting is the most common interaction technique used for selection of targets at the distance. The user points a ray of light at the target and confirms its selection with a button click, a motion guesture or a voice command. There are two challenges with raycasting: accidental selection and occlusion. The first one occurs due to intersection with other object while the pointer is moving toward the target object, and the later one occurs when the target object is hidden behind another object.

Gaze-based selection is another selection technique. The selection starts by looking into the object and the selection is confirmed by an external controller input or by dwelling into the object for some defined period of time. This approach can be expensive and difficult to implement as will need eye tracking in a head-mounted virual reality displays.

Gesture selection is another natural and efficient interaction method in the virtual environment which can effectively express user demands for object selection. An appropriate gesture interaction device such as data gloves will be required. Nearby

objects can be grabbed naturaly and user interaction with objects in virtual world is more realistic.

Interaction Techniques for Manipulation

Once the virtual object is selected, the user may want to manipulate it. Some of common manipulation action can include scaling, rotation or translation of these objects. The choice depends on the available input controller's capabilities. These can range from simple scroll – wheel input controllers up to interaction systems that are able to recognize more natural uestures such as pinch or stretch.

Interaction Techniques for Locomotion

Locomotion involves the use of controllers to enable the movement from one place to another within a virtual reality environment. One problem when a user performs locomotion in virtual world while being still in the real world is that it tends to cause VR sickness.

There are different ways how to walk and run in VR worlds, yet on rails, gaze-directed staring, teleport and real movement are more commonly used. On rails the user movement is controlled by the system as like in a roll a coaster simulator. However, this movement can be extended by allowing the user to look and determine the direction they want to move through gaze directed steering. Gaze-bazed or raycasting can be used also to teleport from one place to another one as well.

Modern VR headsets can enable more intuitive locomotion such as real movement as well, by simply walking around. This reduces vection, but yet it can increase the risk of collision with real-world objects.

Implementation of specific interaction models is largely determined by the capabilities offered by the available VR devices and the underlying development software infrastructure. In the following sections, we will overview XR Toolkit which is establishing as a new standard and Unity implements it as an input and interaction software.

What is XR?

XR is an umbrealla term used to describe virtual reality, mixed reality and augmeneted reality types of application. XR extends across these and future immersive technologies to enable people not only to visit virtual environments, but to engage in immersive experiences and to interact realistically with virtual entites as in real life.

Unity works closely with different partners involved in XR development, to ensure that developers will have the necessary tools to develop XR content that will be supported by all platforms, such as Oculus, Play Station, Microsoft HoloLens and other major players in the field.

Unity has developed a new XR plugin to integrate Unity engine to create XR content that will support XR devices without having to modify the core engine.

XR Setting Configuration

The very first step in Unity projects that are enabled to use different XR interactions is to install and configure the XR interaction toolkit. We need to use a software package that provides simple managemtn of XR-plugins called XR Plugin Management.

The installation can be initialized by going to Edit->Project Settings in Unity environment as shown in the following figure.

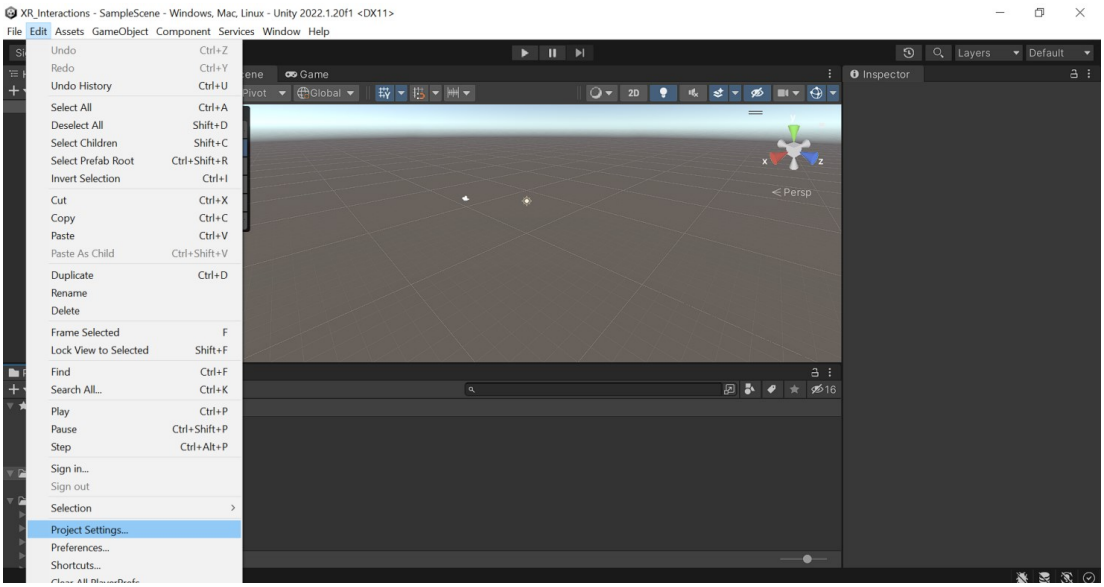

Figure 26 Project Setting Options in Unity

The next step will be to select XR Plugin Management option located on the bottom left of the new window, and click the button Install XR Plugin Management.

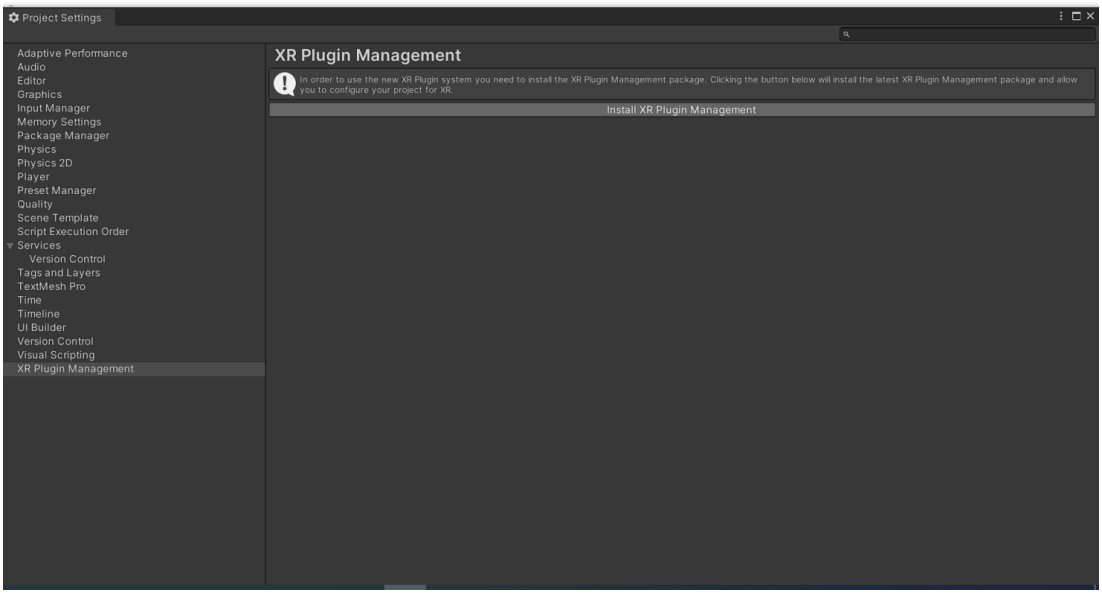

Figure 27 The Unity interface to install XR Plugin Management

After the installation process, the XR Plugin Management is changed to the options provided below. Out of other options, the Open XR is selected.

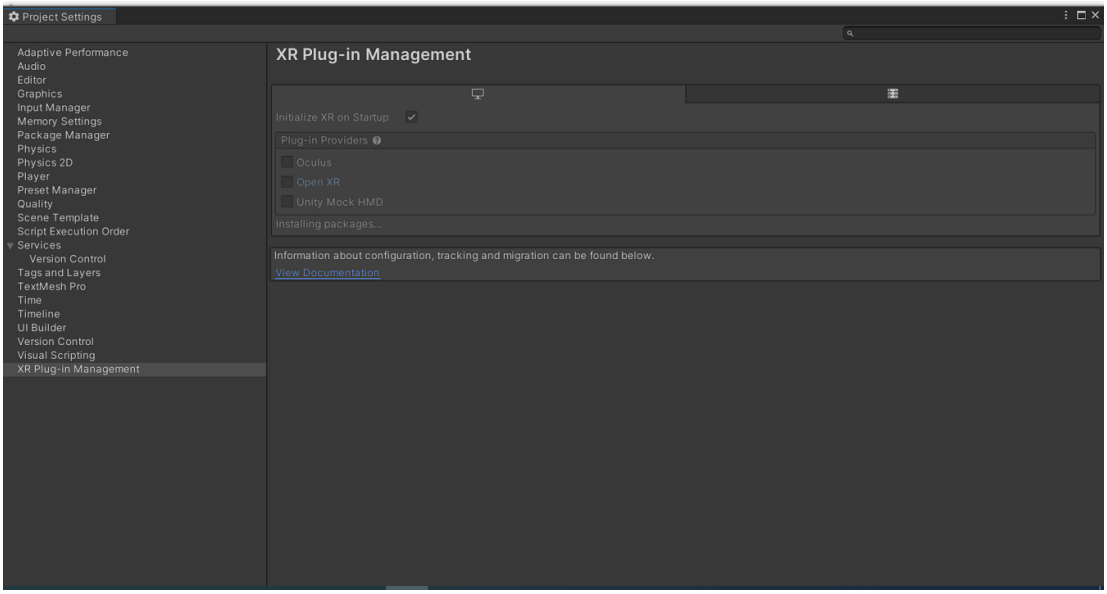

Figure 28 The installation and configuration of Open XR plugin.

At the end of the process, a warning message similar to the one displayed down may appear, and it will require to restart the Unity editor.

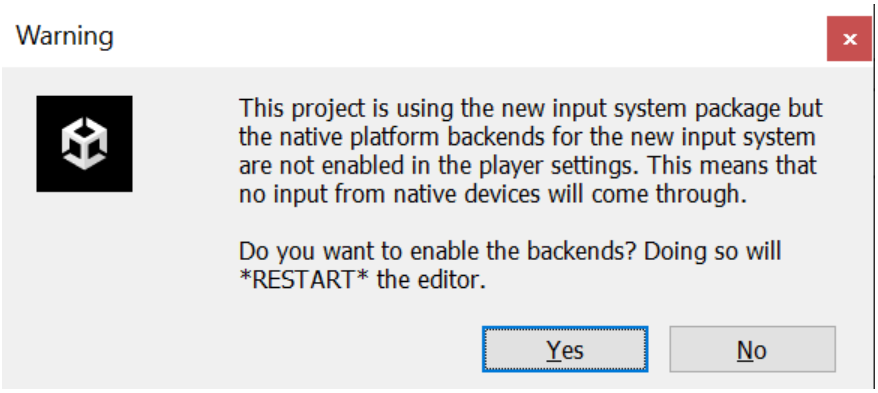

Figure 29 The warning window at the end of XR Plugin installation process

Open XR is an abstraction layer enabling high performance and open standard access to different XR platforms and devices. This is essential to speed up the development process as now developers can create cross-platform XR experiences in a unified manner.

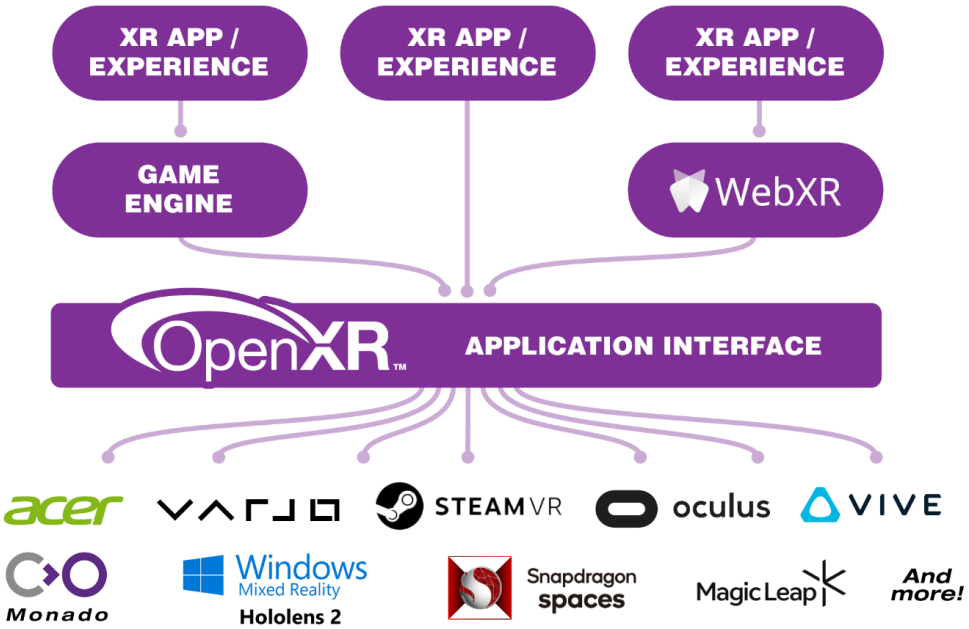

Figure 30 Open XR as an abstraction layer to cross-platform development (https://www.khronos.org/openxr/)

XR Plug-in managaer will require to define an interaction profile as well, that is to determine the type of headset you'll be interacting with. In our case, we will select Oculus Touch Controller Profile as shown in figure below.

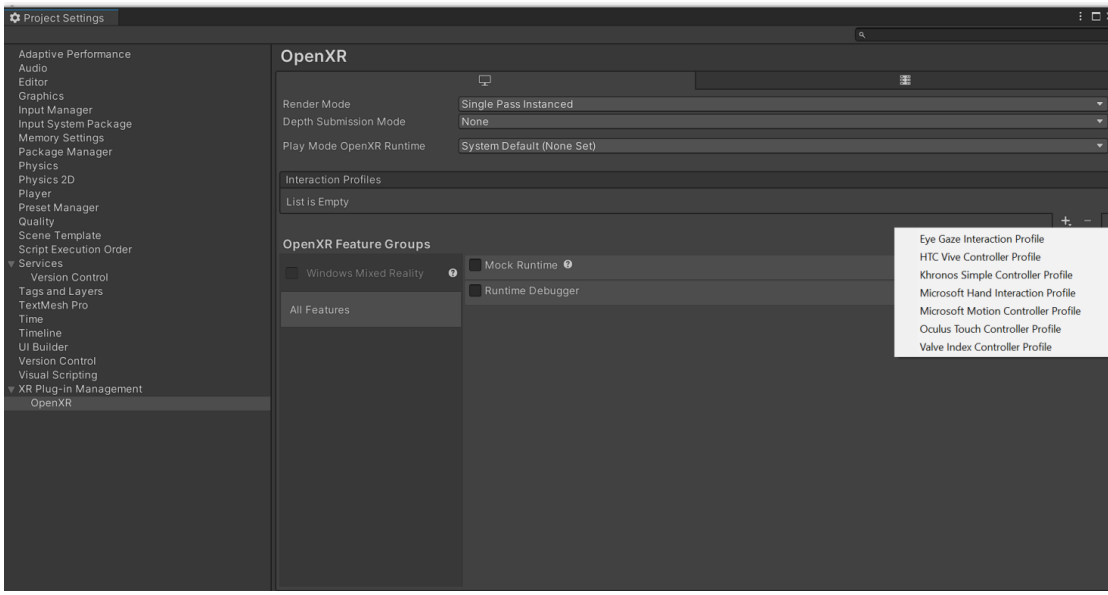

Figure 31 Setting up the interaction profile in Open XR

On the same window, the Renderer mode is changed from Single Pass to the preferred Multi-pass options, which will renderer the Scene into two images shown separately for each eye.

Installing XR Interaction Toolkit

The next step is to install the XR Interaction Toolkit, as it provides a framework that makes 3D and UI interactions available from Unity input events. The installation can be completed from the Package Manager from the Unity Window main menu.

The following steps should complete this process:

- Go to Window->Package Manager option
- Click on + sign located on the top – left corner of the window
- Select “Add package from git URL” option
- Write “com.unity.xr.interaction.toolkit” and click on Add button

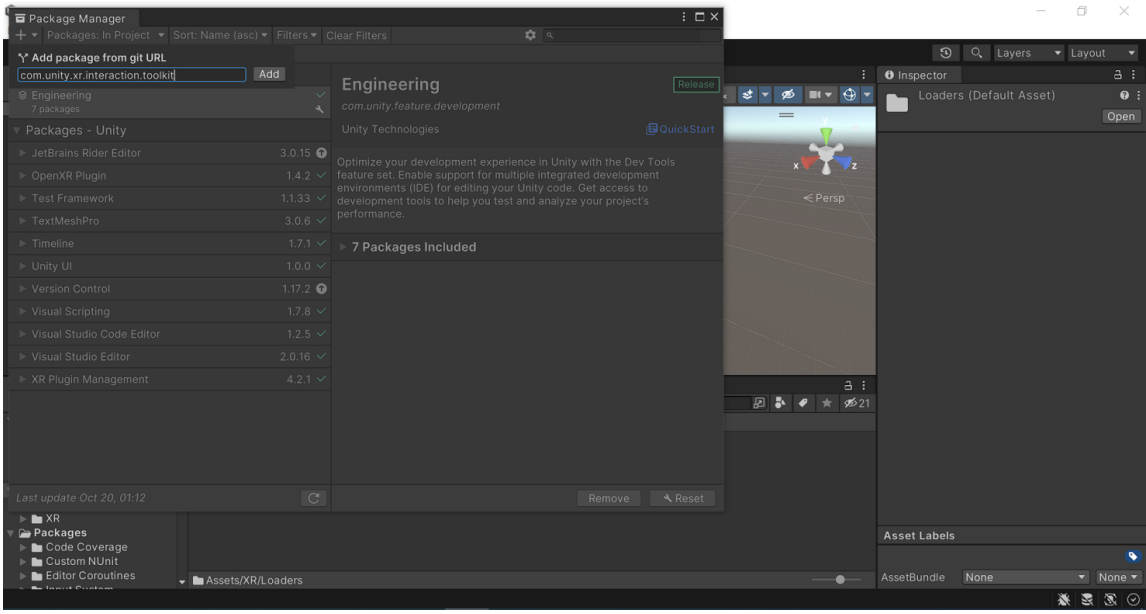

Figure 32 Installing XR Interaction Toolkit

A warning window will appear and informe you that if you have an older XR Interaction Toolkit version you should make a backup before updating to this newer one. Click on “I Made a Backup” button of the window if you have a new fresh project.

After installed, select the package and import the Default Input Actions as shown below.

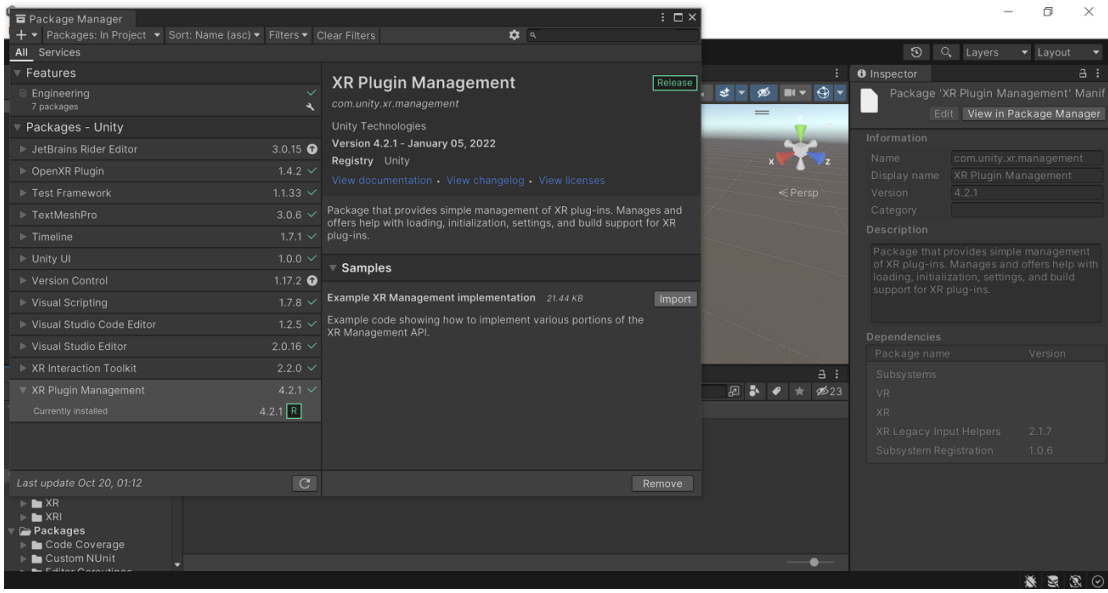

Figure 33 Importing Input Actions in XR Plugin Management

The Assets folder in the project tab should have additinal folders, such Samples, XR and XRI.

In general, this is the daunting task that needs to be completed at the beginning, so we have XR enabled project in Unity. In the following section, we will explain through a simple example, how to create a simple 3D world by defining XR origin, locomotion system and a grabbable object.

Scene Configuration

Before we do any VR development and XR interactions, we will build a kind of a floor and see what is going on the scene, and later we can setup the camera rig and develop the rest of the elements.

We will use the plane 3D object to create the floor. By clicking on + sign on Hierarchy, we can easily select 3D object category and then finally select Plane option.

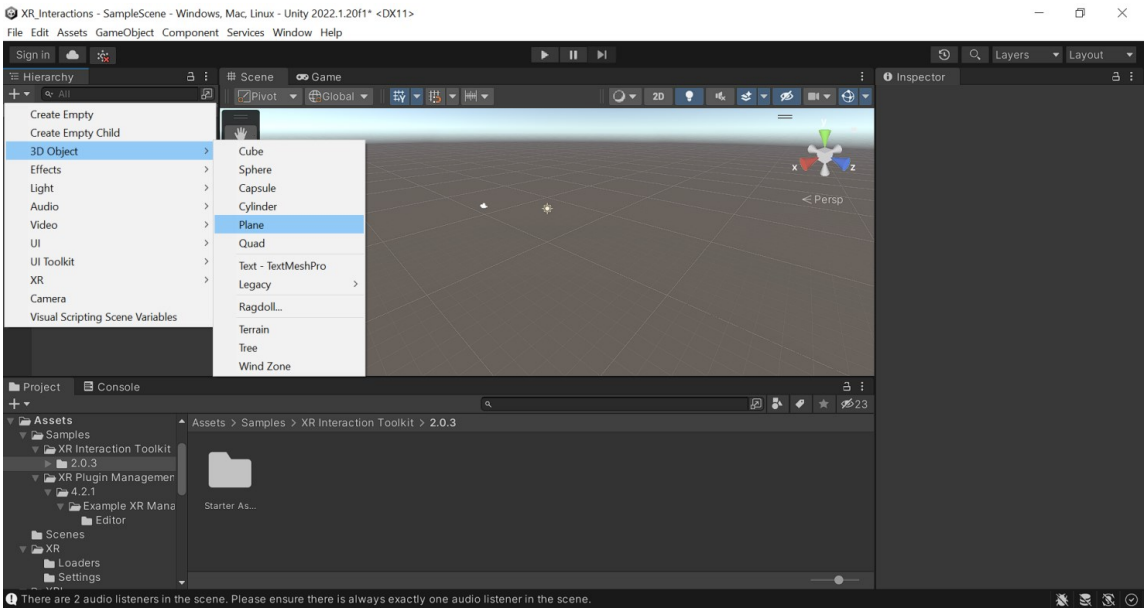

Figure 34 Creating the game floor

Now, we can setup the XR origin. The aim of the VR is to create immersive experiences, and all the visual and audio experience, and the interactions happen inside the VR device, such as Oculust Quest or Meta Quest.

In Unity, the camera and sound are attached to the XR origin. To create the XR origin in Unity, we need to click on the empty space in hierarchy window and create XR origin (action-based) game object.

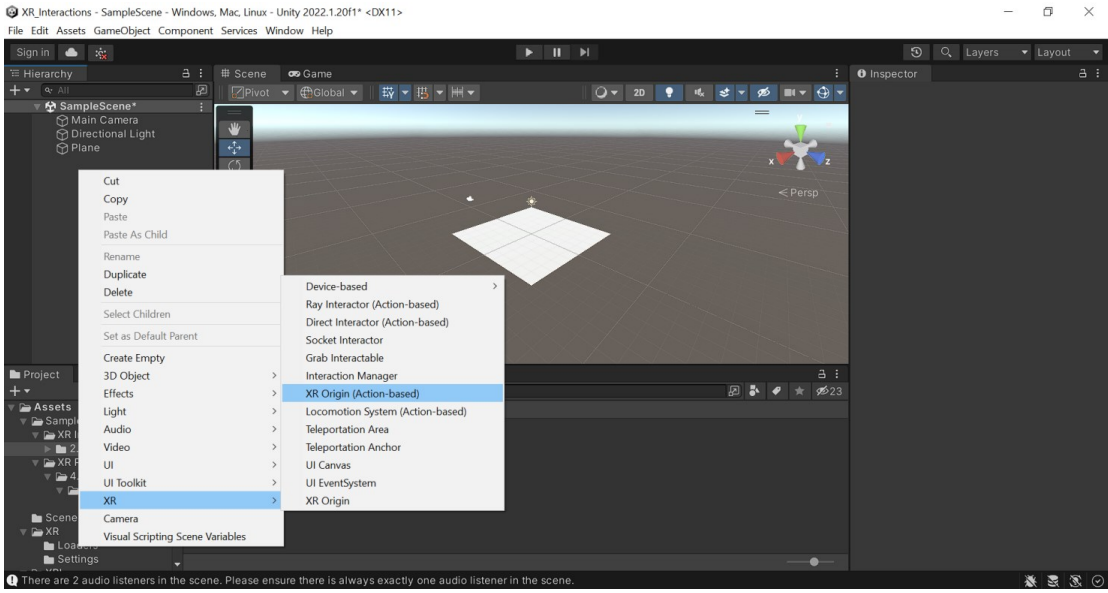

Figure 35 Creating the XR origin

This step will not setup the head camera, but it will automatically add the controllers. Both controllers have different actions, such as tracking, position action and rotation action among other already configured. The XR Interaction Manager game object that contains the script with the same name is automatically imported in hierarchy as well.

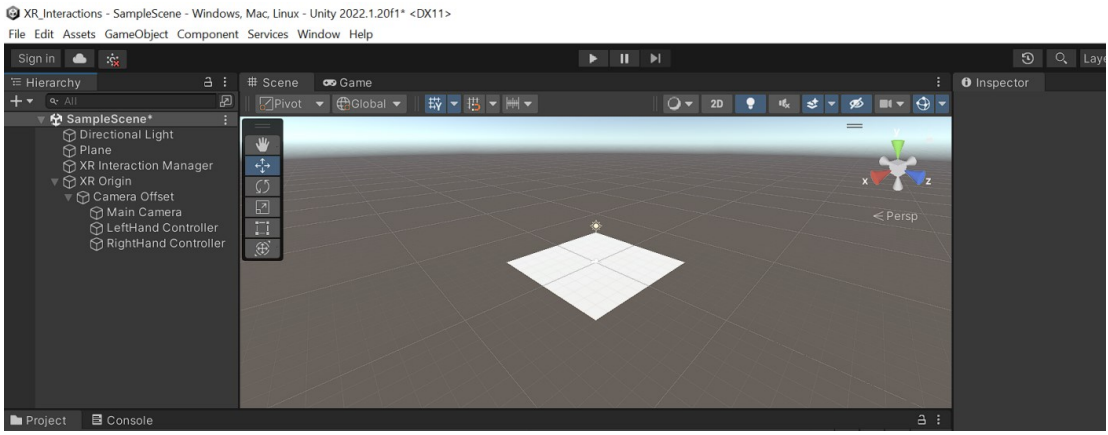

Figure 36 The XR Origin with left and right controller

Once the steps above completed, the Unity editor is ready for checking if our headset and hand controllers are identified by the Unity editor.

As we want to move around the scene, adding locomotion capabilities is easy. All we have to do is to go to Hierarchy, right click and add a Locomotion System (action-based).

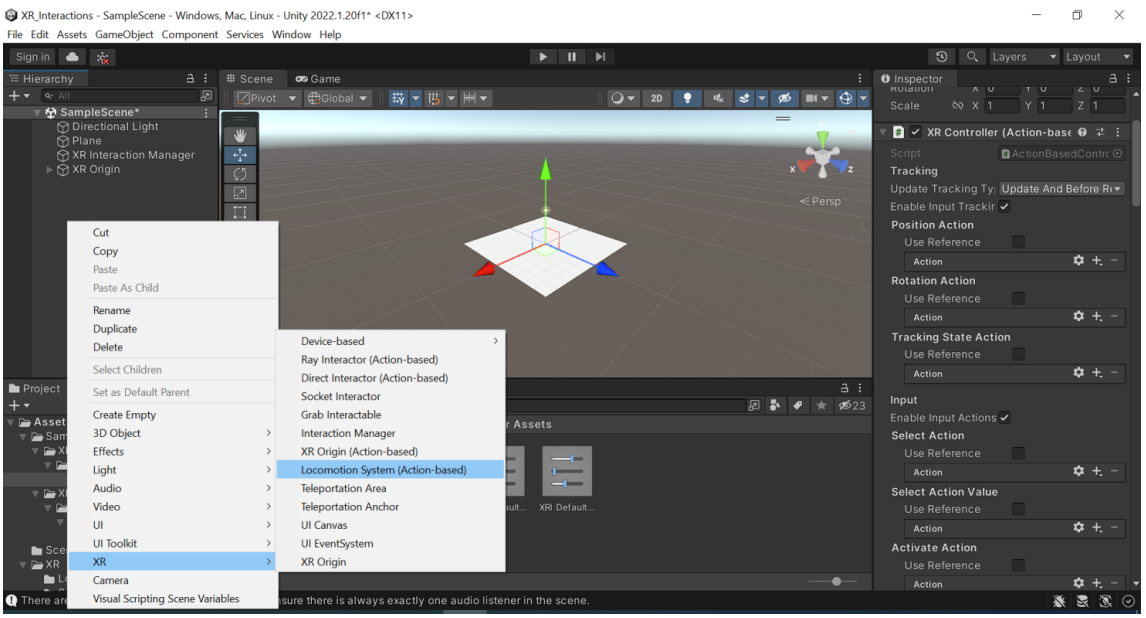

Figure 37 Adding the locomotion system in Unity

We will configure the locomotion system for teleportation purposes. The simplest solution will be to create teleportation area, and this area can be used to teleport around. If we don't define Teleportation Anchor, the Teleport Area will allow you to teleport to any point on the plane. To create a teleportation area, we have to right click on the empty part on the hirearchy, right click with the mouse and create XR object Teleportation Area. The teleportation area will be the same size and position as the plane, so basically the user can walk or move to any point of the floor.

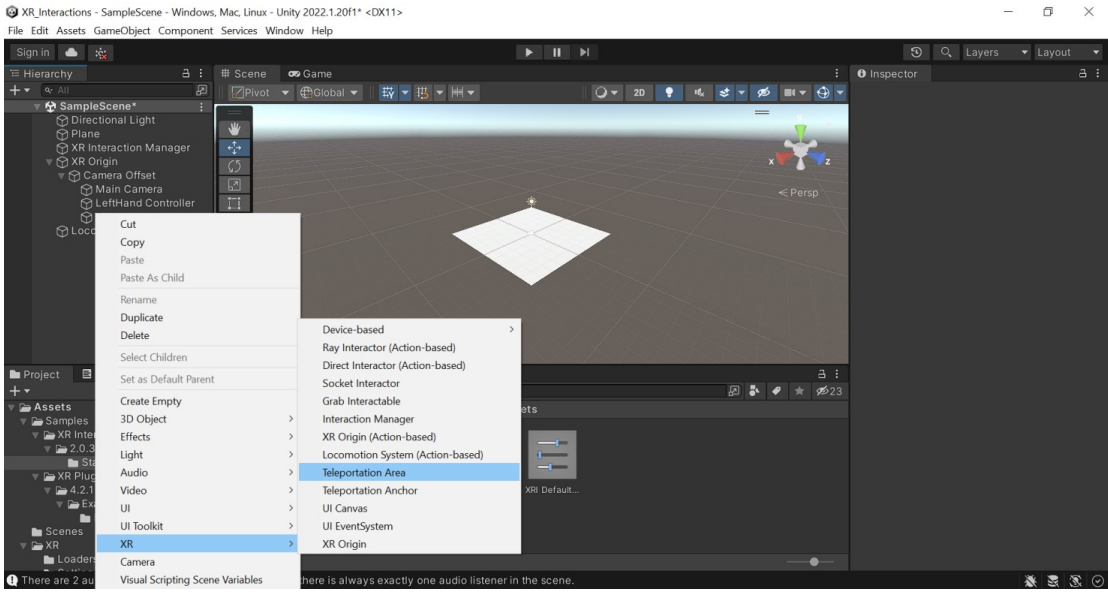

Figure 38 Creating a Teleportation Area in Unity

At the end we will create a grabbable object, namely a simple sphere object. The interaction with this object is different of the teleportation. It will enable the user to select and make the object interactable.

The following steps need to be followed:

- Create a Sphere object in hierarchy
- Change the Y position to 0.5
- Create a material in Asset folder and define the Albedo propery in red colour
- Drag the material above the sphere in Scene to apply the material
- While the sphere is select in hierarchy, add a XR Grab Interactable component in the inspector

When adding the XR Grab Interactable component to an object, it will automatically add the physic componenet Rigidbody, so that we can pick up the sphere object and interact with it in a physics-based way.

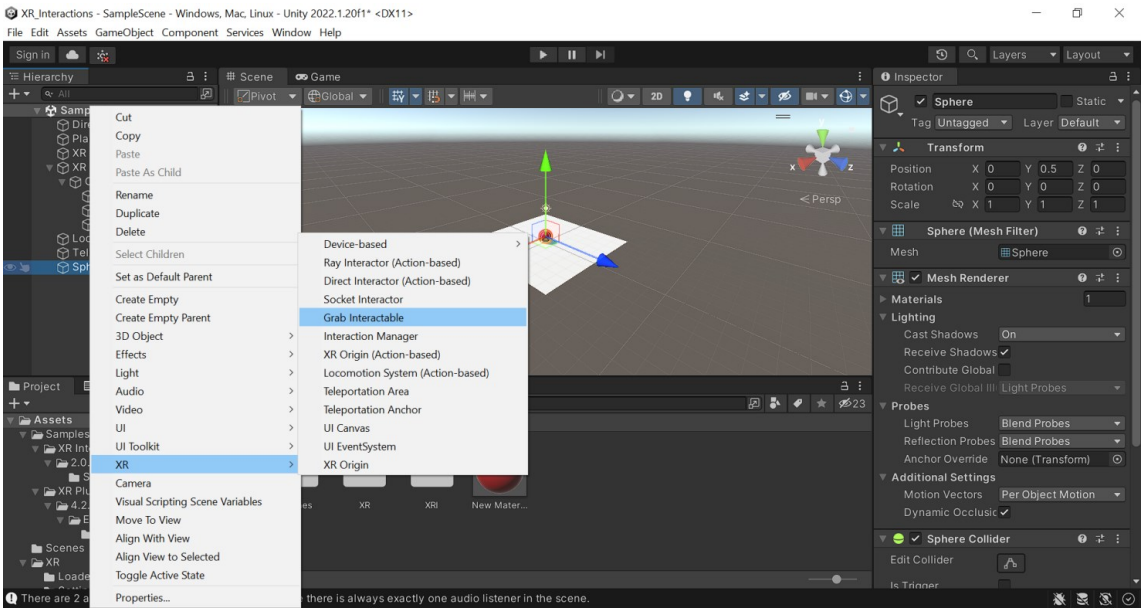

Figure 39 Adding the Grab Interactable component in the sphere object

For better performance, some parameter may be tunned. In this case, the Movement Type value has been changed to Velocity Tracking, the Enable Smooth Position and Rotation has been checked, and Collision Detection of Rigidbody component has been changed to Continuous Dynamic.

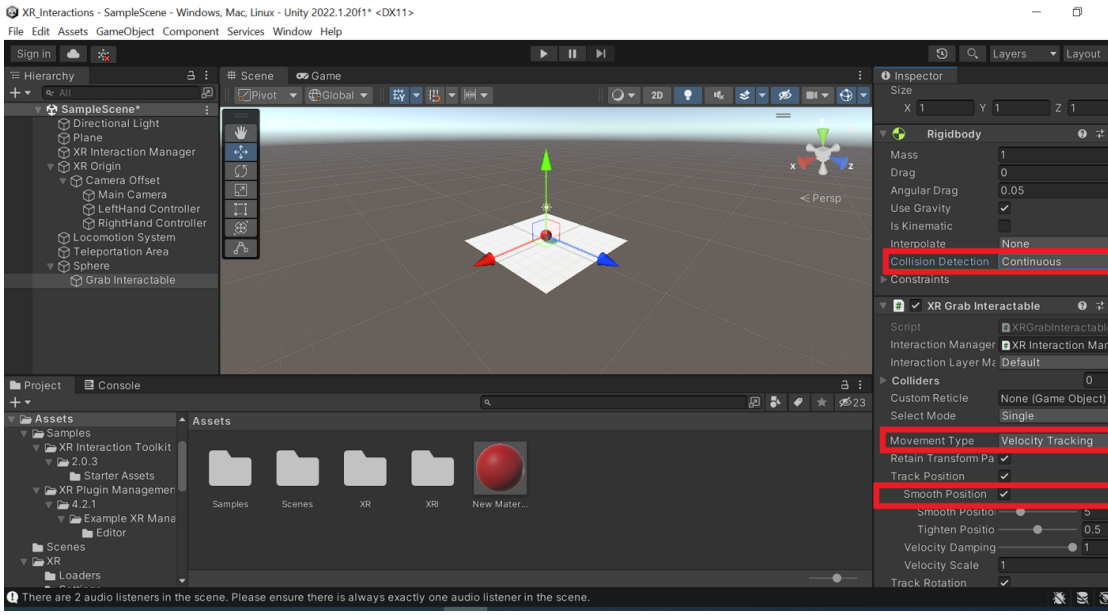

Figure 40 Tunning the object interaction parameters

This steps will be sufficient to create a demo for creating a simple VR world based on primitive Unity 3D objects with locomotion and object interactions.

Summary

Classical apps or games developed for PCs or tablets have a standardized set of inputs which typically includes the keyboard, the mouse or some type of joystick, and thus the ingteraction of users there is standardized as well. In contrast, VR requires rich interactions for selection, manipulation and locomotion. Their implementation is a key aspect to make VR compelling.

However, creating and tunning such interactions is a challenging and time consuming task for developers. Here, we have explored the XR Interaction Toolkit as a high – level, component based interaction system for creating VR experiences which aims to speed up the iteration process, bringin more interactivity and immersion to VR experiences.

Developers have to make critical decisions how the VR world is going to be accessed and what type of interactions for users need to be enabled in such world. But, such decisions are affected by the choice of hardware and the capabilities of the tool for VR interactions. Natural interactions are essential, and XR interaction toolkit forms a good basis for this purpose. Yet, other interaction tools are developed and new one will emerge in the future, and they need to be evaluated and maybe become a tool of choice in the development process.

4
WHAT'S NEXT?

The manuscript has been designed to offer an introduction to VR development using Unity. The intention has been mainly to attract the attention from interested parties not only to begin to understand the virtual reality, but to offer some basis to create their own VR content and experiences as well.

There are a lot of important issues when developing for VR, both technical and philosophical that need to be thought and worth considering. Some of them are:

1. Creation of immersive VR content can be daunting and time – consuming task. On the other hand, the developer tools and the supporting hardware may come with drawbacks and limitations. On top of this, as any other new or emerging technology, the developer tools and the supporting hardware infrastructure has high upfront cost. All these aspects may cause frustration at the beginning, yet as the development tools will improve over the time and the cost of the technology will decrease, we will see that the the VR technology will become more affrtable and more widely adopted in different sectors.

2. VR sickness among developers and users. Among VR users, a new term known as VR sickness has been coined. It is used to describe the negative physical effects that VR utilization has over the users. Common symptoms include but not limited to dizziness, disorientation, eye strain, and so on. Many hardware manufacturers will use the term Interpupillary Distance (IPR) that refers to the relationship between the distance of lenses on the headset and the distance between user's eyes. Headset manufacturers will usually include an IPR adjuster, as if IPR is not set correctly, it may cause eye strain and discomfort. Although there is no cure to VR sickness yet, there are some good practices and known techniques that can reduce the likelyhood or avoiding sickness triggers.

3. Cybersecurity and data privacy. As many of VR content is consumed online and through mobile devices as well, there is an increased concern about identity theft, password stealing, data leackage, location tracking and all other security issues. Developers must adhere to strong security policies and embed strong security measures when create their VR apps, and ensure that user is protected and their private data is secured. This will have also direct impact to increasing the adoption rate and success of the VR technologies in the competitive market.

4. The VR can have also a philosophical dimension as well. Is VR real and if yes, how real is it? The discussions and arguments usually are between two extreme views. On one side, we have the most common view that cyberspace and virtual reality are some type of consensual hallucination, and all what happens there is fictional and not real. On the other side, an opposite view is presented and defended. According to this view, virtual objects, avatars and virtual worlds are perceived and considered a sort of genuine reality and what happens there is truly real. No matter on the philosophical view, virtual reality is used as a catalyst for thought outside the technological aspects and specifics, like raising questions about what we are or what we choose to be.

REFERENCES

[1] Linowes, J. (2015). *Unity Virtual Projects*. Packt Publishing.

[2] Murray, J.W. (2020). *Building Virtual Reality with Unity and SteamVR.* CRC Press.

[3] Unity. Unity Real-Time Development Platform (https://unity.com/).

[4] Unity Asset Store. Unity Asset Store - The Best Assets for Game Making (https://assetstore.unity.com/)

[5] Home of the Blender (https://www.blender.org/)

[6] Sketrchup Free 3D modeling Software (https://www.sketchup.com/plans-and-pricing/sketchup-free)