Wadia College of Engineering

Department of Computer Engineering

Name of Student:	**Class:**
Semester/Year:	**Roll No:**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part II - 1**

LP II – PART II Augmented & Virtual Reality

ASSIGNMENT NO: 01

AIM:

Installation of Unity and Visual Studio, setting up Unity for VR development, understanding documentation of the same.

OBJECTIVES:

- Install the Unity Hub and Editor.
- Identify and use essential features of the Unity Editor.
- Navigate in 3D space in the Scene view.
- Create and manage projects in the Unity Hub.
- Identify the key elements of the Unity Learning Ecosystem, and their purpose.

APPARATUS:

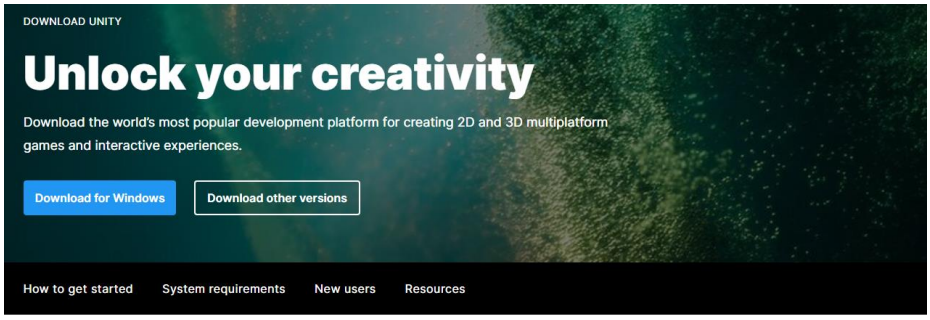
Minimum requirements	**Windows**	**Linux (Support in Preview)**
Operating system version	Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only.	Ubuntu 20.04, Ubuntu 18.04, and CentOS 7
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-capable GPUs	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs.

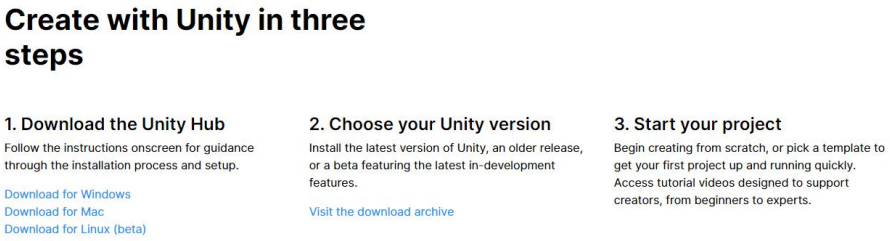
THEORY:

Before you begin this assignment, check that your computer meets the requirements for Unity. Unity can be installed on Windows and macOS. Support for Linux is in preview, which means that it will work but you might encounter some bugs.

Download the setup wizard

1. Go to the Unity download page. **(https://unity.com/download?_ga=2.87821008.93811714.1646883679-485231182.1644381162)**





2. Under **1. Download the Unity Hub**, select the option for your operating system. A file will download named **UnityHubSetup**. This file name might vary depending on the Hub version and your operating system.

Install and launch the Unity Hub

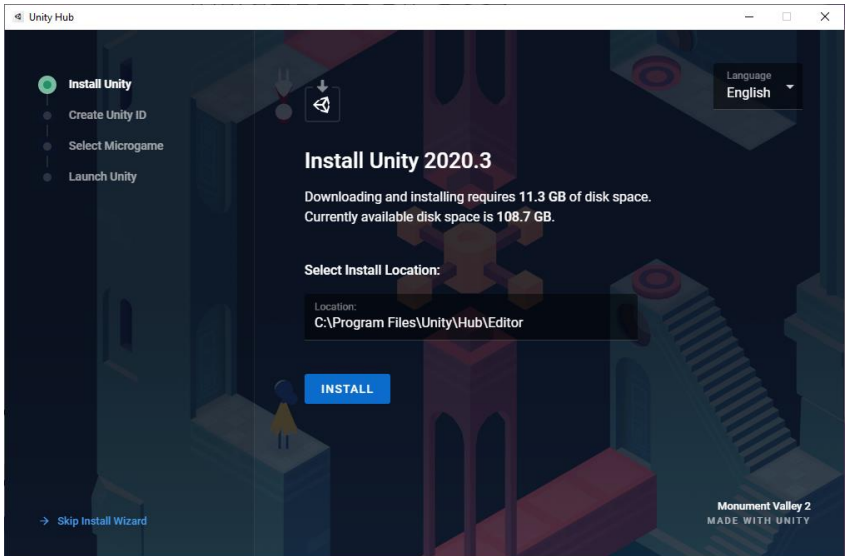
1. Locate the UnityHubSetup.exe file you downloaded in the previous step.

2. Launch the **UnityHubSetup** installer.

3. Follow your platform instructions to install the Unity Hub.

4. Open the Unity Hub application.

5. Choose an install location then select **Install** to install the Unity Editor.



The Unity Hub will guide you from here. The Unity Hub is a standalone application that streamlines the way you navigate, download, and manage your Unity projects and installations.

You can use the Hub to:

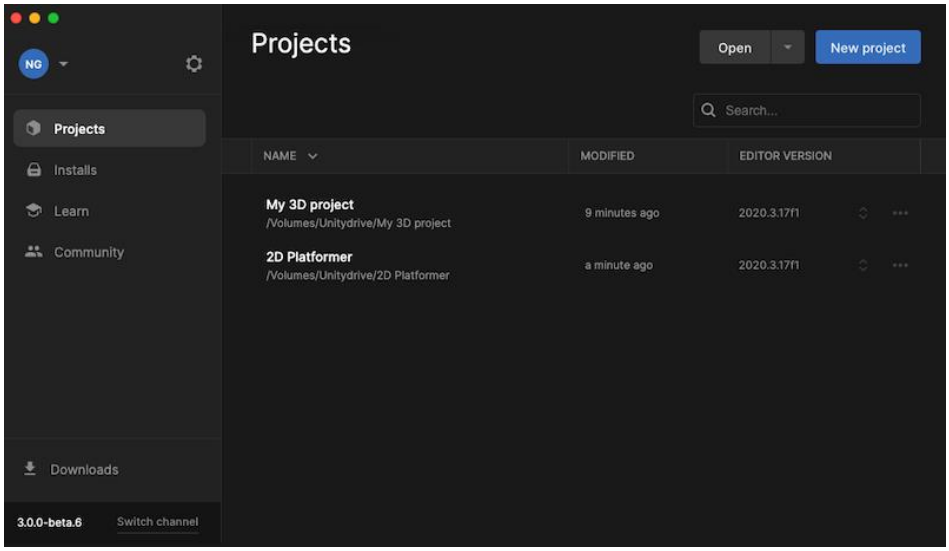
- Manage, download, and install modules and versions of the Unity Editor.
- Create and manage your Unity projects.
- Explore templates, sample projects, and learning materials for different skill levels.
- Manage your profile, preferences, and Unity licenses.
- Send feedback and get help from Unity.

Projects

In the Unity Hub, the Projects page displays your Unity projects. You can use the Projects page to create a new project, manage your existing projects, or open a project in the Unity Editor.

View your projects

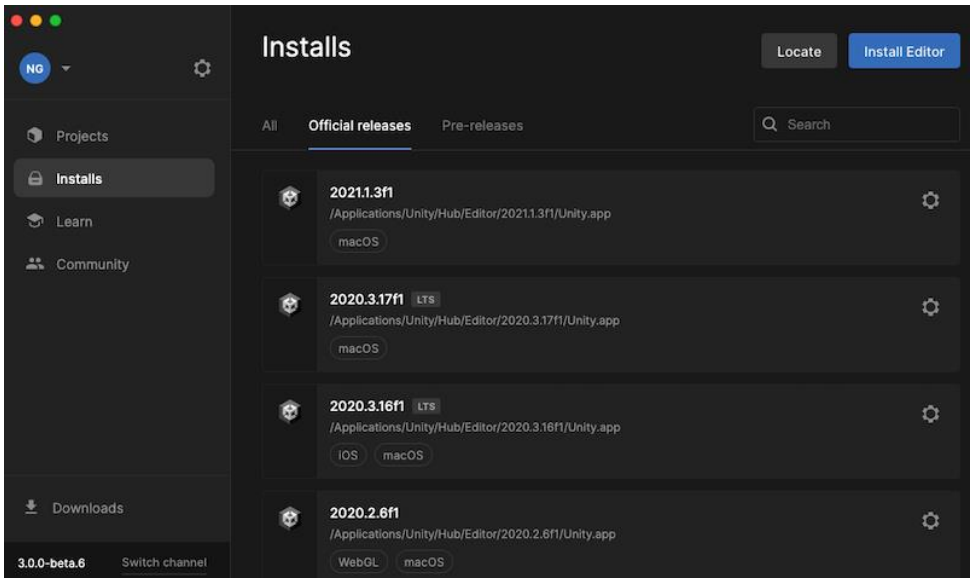
To view your Unity projects in the Hub, select the **Projects** tab. This displays your project names, when they were last modified, and the Editor version you made them with.



Downloading and installing Editors and modules with the Unity Hub

You can use the Hub to manage multiple installations of the Unity Editor along with their associated components and modules.

To view the Editor versions that you have added to the Unity Hub, select the **Installs* tab on the left panel of the Hub. This shows all of the Editor versions and modules that you can currently open via the Hub.

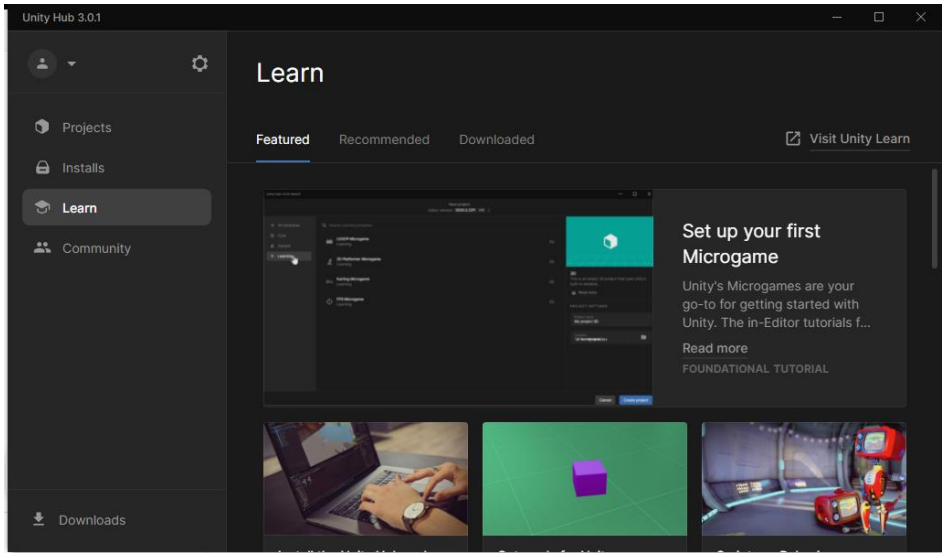


You can view either official releases or pre-releases by switching between the tabs at the top of the page. The Hub displays the targeted platforms of each Editor in the tags below the name.

Learn

Unity Learn helps you develop and learn how to use Unity for your projects. Unity Learn provides free and on-demand learning content, including tutorials and pre-made projects for different game types so you can build on them and customize assets and the behavior of GameObjects.

To view the Learn window, select **Learn** from the left panel.



Community

The community tab provides a variety of ways in which you can interact with other Unity users. You can also use the linked sites to communicate with Unity staff, to give feedback or get help.

The resources provided are:

- Unity Blog
- Answers
- Forums
- Live Help
- Unity Play
- Unity Pulse

Installing Unity

The Hub is the primary way to install the Unity Editor, create projects, and manage your Unity experience. It provides a central location to manage your Editor installations, Accounts and Licenses, and Projects.

The Unity User Manual 2020.3 (LTS) is available at

https://docs.unity3d.com/Manual/index.html

CONCLUSION:

We have successfully installed Unity Hub and Unity Editor version 2020.3.30f(LTS)

QUESTIONS:

1. What are the minimum system requirements (hardware) for installing Unity Game Engine?
2. What is the Unity Hub?
3. How is the Unity Hub helful?
4. List the various versions of Unity Game Engine. Which is the most recent LTS version?
5. What are the other game engines? Compare them with Unity.

Wadia College of Engineering

Department of Computer Engineering

Name of Student:	**Class:**
Semester/Year:	**Roll No:**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part II - 2**

LP II – PART II Augmented & Virtual Reality

ASSIGNMENT NO: 02

AIM: To demonstrate the working of HTC Vive, Google Daydream or Samsung gear VR.

OBJECTIVES: To understand the working of any one Virtual Reality Device.

APPARATUS: HTC Vive, Google Daydream or Samsung gear VR

THEORY:

HTC VIVE

The HTC Vive implements "room-scale" virtual reality, whereby a user can walk freely around a play area rather than be constrained to a stationary position. The controllers and headset use a positional tracking system known as "Lighthouse"; multiple external base station units (also referred to as "lighthouses") are installed in the play area, which each contain an array of LED lights, and two infrared lasers. The lasers are attached to rotating spinners which sweep the play area vertically and horizontally with timed pulses. The headset and controllers contain photosensors which detect the LED lights from the base stations, and then compare them with the timing of the laser sweeps in order to determine their relative position in 3D space.[28][29]

The Vive is bundled with two wand-like motion controllers; they feature circular trackpads similar to the Steam Controller. The headset contains two OLED display panels with a resolution of 1080×1200 per-eye, with a refresh rate of 90 Hz and a 110 degree field of view. It contains a accelerometer, gyroscope and proximity sensor, and a front-facing camera used for the "Chaperone" feature, which can display the boundaries of the user's chosen perimeter or the view of the camera in order to help guide the user away from objects and walls in their play area.

The headset must be connected to a supported PC using a "link box", which contains USB 3.0, HDMI, and power connectors. The Vive initially required computers running Microsoft Windows.

HTC Vive headsets support a number of accessories

Vive Controllers: The controllers have multiple input methods including a track pad, grip buttons, and a dual-stage trigger and a use per charge of about 6 hours.

Vive Tracker: A motion tracking accessory; it is designed to be attached to physical accessories and controllers, so that they can be tracked via the Lighthouse system.

Deluxe Audio Strap: In June 2017, HTC released the Deluxe Audio Strap, a replacement head strap for Vive headsets which includes integrated over-ear speakers.

Vive Wireless Adapter: The Vive Wireless Adapter was launched as an accessory in September 2018 for the original Vive and Vive Pro, which allows the headset to be operated wirelessly with a battery pack and transmitter, using V band WiGig technology.

Vive Facial Tracker: In March 2021, HTC announced and released the Vive Facial Tracker, an accessory attached to the headset containing infrared-illuminated cameras for facial motion capture.

The **aGlass** lenses are alternate eyepieces developed by 7invensun as part of the Vive X program, which add eye tracking support to the headset.

GOOGLE DAYDREAM



The first-generation Google Daydream View was announced on October 4, 2016. Daydream-ready smartphones can be placed in the front compartment of the Daydream View and then viewed in VR through the headset's two lenses. Daydream also included a new head tracking algorithm that combined the input from various device sensors, as well as integration of system notifications into the VR user interface.[8]

Daydream allows users to interact with VR-enabled apps, including YouTube, Google Maps Street View, Google Play Movies & TV, and Google Photos in an immersive view. Google recruited media companies like Netflix and Ubisoft for entertainment apps. Xiaimi 12 ultra

Google Daydream headsets are packaged with a wireless controller. This controller can be used for interacting with the virtual world through button presses or through waving the device. On-board sensors are used to track the orientation of the controller and approximate the position of the user's hand. The Daydream View's controller can be stored inside the headset while not in use.[11] The controller has a touch pad, two circular buttons (one functioning as a home button and one functioning as an app-specific button), and two volume buttons, along with a status light. The controller is rechargeable and charges via USB-C. The second-generation Daydream View was unveiled during the Made by Google 2017 event.

SAMSUNG GEAR VR

The Samsung Gear VR is a virtual reality headset developed by Samsung Electronics, in collaboration with Oculus VR, and manufactured by Samsung. The headset was released on November 27, 2015.

When in use, a compatible Samsung Galaxy device acts as the headset's display and processor, while the Gear VR unit itself acts as the controller, which contains the field of view, as well as a custom inertial measurement unit, or IMU, for rotational tracking, which connects to the smartphone via USB-C or micro-USB. The Gear VR headset also includes a touchpad and back button on the side, as well as a proximity sensor to detect when the headset is on.

The Samsung Gear VR is designed to work with Samsung's flagship smartphones. Currently supported are Galaxy S6, Galaxy S6 Edge, Galaxy S6 Edge+, Samsung Galaxy Note 5, Galaxy S7, Galaxy S7 Edge, Galaxy S8, Galaxy S8+, Samsung Galaxy Note Fan Edition, Samsung Galaxy Note 8, Samsung Galaxy A8/A8+ (2018) and Samsung Galaxy S9/Galaxy S9+. The focus can be adjusted using the wheel at the top of the headset. A trackpad is located on the right of the device, home and back buttons are located just above it. Volume can be adjusted through the volume rockers also found on the right-hand side. A USB-C port is located on the bottom of the headset.

IMPLEMENTATION: Demo

CONCLUSION: The working of the VR device was demonstrated successfully.

QUESTIONS:

1. What devices can be used with a VR system?
2. What types of mainstream head-mounted displays are available?
3. In which industries is VR mostly applied?
4. What are the deciding factors to buy a VR GEAR?
5. Compare the above VR devices w.r.t. use and features.

Wadia College of Engineering

Department of Computer Engineering

Name of Student:	**Class:**
Semester/Year:	**Roll No:**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part II - 3**

LP II – PART II Augmented & Virtual Reality

ASSIGNMENT NO: 03

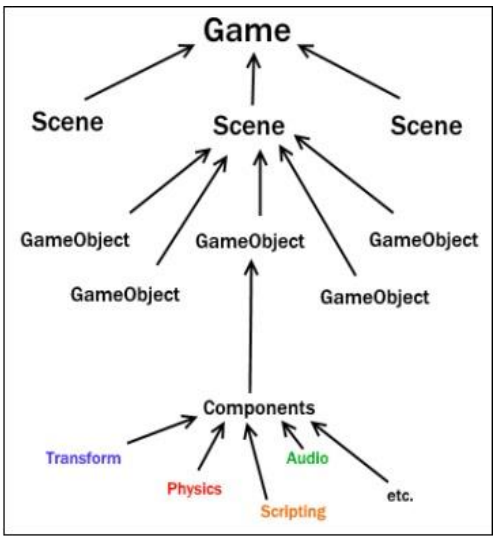
AIM: Develop a scene in Unity that includes:

1. A cube, plane and sphere, apply transformations on the 3 game objects.
2. Add a video and audio source.

OBJECTIVES: To be able to implement game scenes in Unity, add various game objects and apply transformations on the game objects.

THEORY:

- In Unity, all gameplay takes place in scenes.
- By default, a new Scene in Unity will have a Camera object in the scene called the Main Camera.
- The main camera renders everything that it sees or "captures" in a specific region called the viewport.
- A scene itself is made out of objects, called GameObjects.
- GameObjects have a set of components attached to them, which describe how they behave in the scene, as well as how they react to others in the scene.
- The **GameObject** is the most important concept in the Unity Editor.
- Every object in your game is a **GameObject**, from characters and collectible items to lights, **cameras** and special effects.

- 

IMPLEMENTATION:

(A) Adding Game Objects and Transformation on game objects

1. Open unity hub and create a new project. Type the necessary details like the project name and make sure that you select 3D.

2. After you create a new project, a window will pop up which has nothing except a main camera which is basically the viewport of what the user is going to see.

3. Firstly go to Game Object → 3D Object → Select plane.

4. A plane gets inserted on the floor.

5. Click on the plane gameobject, you will find a window with name inspector.

6. Click on the tab and change the values in the rotation, transformation and in scaling section to rotate, transform and scale the game object.

(B) To develop a 3D scene in Unity with Audio and Video Source

1. Open unity hub.

2. Create a menu project.

3. Select 3D core.

4. Click Create Project.

5. Wait for Project to load.

6. A new window will appear with an empty plane.

7. Click on the plus (+) icon at top left of the screen.

8. Select 3D objects such as cube, plane and sphere.

9. Select any 3D object on the Scene Window.

10. Now to the right side of the screen, a new window with multiple options will pop.

11. Select Transform Option and do changes to the object according to your liking.
12. Select the object again from the scene window and at the bottom right side of the windows, an option Add Component will appear.
13. Select that option and search for a video player and select it.
14. Now at the bottom right side, a new component will be there called video player.
15. Add a video clip to the component.
16. The 3D object will become a video player.
17. Now to add an audio source clip on the add component and search for the audio source.
18. Now again at the bottom right side an audio component window will appear, select the desired audio and the 3D object will become an audio source.

CONCLUSION: The 3 game objects were created successfully, transformations have been applied on them and audio/video source has been added.

QUESTIONS:

1. Does every GameObject in Unity has a transform?
2. How do you transform a GameObject in Unity?
3. How do you move an object from one position to another in Unity?
4. Which component do you use to add a sound file to Unity?
5. How do you add an audio and video source to an object in Unity?

Wadia College of Engineering

Department of Computer Engineering

Name of Student:	**Class:**
Semester/Year:	**Roll No:**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part II - 4**

LP II – PART II Augmented & Virtual Reality

ASSIGNMENT NO: 04

AIM: Develop a scene in unity that includes a cube, plane and sphere. Create a new material and texture separately for three Game objects. Change the color, material and texture of each Game object separately in the scene. Write a C# program in visual studio to change color and material texture of the game objects dynamically on button click.

OBJECTIVES:

THEORY:

Using materials with C# scripts

All the parameters of a material asset that you see in the **Inspector** window are accessible via script, giving you the power to change or animate how a material works at runtime.

This allows you to modify numeric values on the material, change colours, and swap textures dynamically during gameplay. Some of the most commonly used functions to do this are:

Function Name	Use
SetColor	Change the color of a material (Eg. the albedo tint color)
SetFloat	Set a floating point value (Eg. the normal map multiplier)
SetInt	Set an integer value in the material
SetTexture	Assign a new texture to the material

The full set of functions available for manipulating materials via script can be found on the Material class scripting reference.

One important note is that these functions **only set properties that are available for the current Shader object** on the material. This means that if you have a shader that doesn't use any textures, or if you have no shader bound at all, calling SetTexture will have no effect. This is true even if you later set a shader that needs the texture. For this reason it is recommended to set the shader you want before setting any properties, however once you've done that you can switch from one shader to another that use the same textures or properties and values will be preserved.

These functions work as you would expect for all *simple* shaders such as the legacy shaders, and the built-in shaders *other than the Standard Shader* (for example, the particle, sprite, UI and unlit shaders). For a material using the *Standard Shader* however, there are some further requirements which you must be aware of before being able to fully modify the material.

IMPLEMENTATION:

1. Open Unity Hub.
2. Click on new.
3. Select 3D core.
4. Click Create Project.
5. Wait for project to load.
6. A new window will appear with an empty plane.
7. Click on the plus (+) icon at top left of the screen.
8. Select 3D objects.
9. Select cube, plane and sphere.
10. Select any 3D object on the scene window.
11. Now click on the plus (+) icon at the bottom left of the screen.
12. Select Material.
13. A new file named material will appear in the asset window.
14. Click on that file.
15. A new window will appear on the right side of the screen.
16. On the new window, we can choose the color, texture of the material.

    We can choose how much metallic, smoothness, reflection, refraction of light there should be. We can create as many materials as we want.

17. For textures, we need to create those files in image format and import that in unity and apply them to our 3D objects.
18. For scripting, right click on the project window.
19. Select scripts and rename it.
20. Write your script in your .cs file in vs code.
21. Select the game object
22. Drag and drop the script on the game object to work.

C# script that changes colors

```
public class ColorSolid : MonoBehaviour
{
public Color ObjectColor;

private Color currentColor;
private Material materialColored;

void Update()
{
if (ObjectColor != currentColor)
{
//helps stop memory leaks
if (materialColored != null)
UnityEditor.AssetDatabase.DeleteAsset(UnityEditor.AssetDatabase.GetAsset
Path(materialColored));

//create a new material
materialColored = new Material(Shader.Find("Diffuse"));
materialColored.color = currentColor = ObjectColor;
this.GetComponent<Renderer>().material = materialColored;
}
}
}
```

C# script that changes colors on keypress.

```
function Update ()
{
    if(Input.GetKeyDown(KeyCode.R))
    {
        gameObject.renderer.material.color = Color.red;
    }
    if(Input.GetKeyDown(KeyCode.B))
    {
```

```
            gameObject.renderer.material.color = Color.blue;
        }
        if (Input.GetKeyDown (KeyCode.T))
        {
            gameObject.renderer.material.color = Color.white;
        }
    }
```

CONCLUSION:

3 game objects have been created successfully and their colour and texture changes on mouse click.

QUESTIONS:

1. How to change material color in Unity?
2. How to change the texture of a 3D object in Unity?
3. How to change the color of my textures in Unity?
4. Why do we use C# in Unity?
5. Which is better, C# or C++ in Unity?

Wadia College of Engineering

Department of Computer Engineering

Name of Student:	**Class:**
Semester/Year:	**Roll No:**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part II - 5**

LP II – PART II Augmented & Virtual Reality

ASSIGNMENT NO: 05

AIM: Develop and deploy an AR app, implement the following using Vuforia Engine developer portal

1. Plane detection
2. Marker based tracking (Create a database of objects to be tracked in Vuforia)

OBJECTIVES: Learn to implement AR using Vuforia and Unity.

THEORY:

Working with Vuforia

1. Open browser and visit Vuforia website.
2. Create an account to work with database
3. Login and click on the target manager
4. Click on Add Database.
5. Upload all your markers in the database
6. Click on download to download the database for unity editor
7. Now click on License Manager.
8. Click on your project's database name.
9. You will receive an License key which we will using later on.
10. Again click on download section which is present in the website's navigation bar.
11. Select and download the appropriate version of Vuforia package.

Working with Vuforia in Unity

1. Open unity hub.
2. Create new project
3. Select 3D core.
4. Click on create project.
5. Wait for the project to load.
6. A new window will appear with an empty plane.
7. Create a new folder in project window and place your vuforia package and your database
8. Go to window and click on import packages.
9. And import your vuforia package and your database.
10. It would take some time to import.
11. Delete the default main camera and add an AR Camera game Object by expanding the Game Object menu dropdown.
12. Select Vuforia Engine – AR Camera and open the vuforia Engine configuration and add the license Key.
13. In the same way as in step #23, go to game Object and add Vuforia Engine- Ground Plane-Ground plane stage.
14. The groundplane stage GameObject serves as a parent GameObject Your contact should be made a child of this component.
15. Select the object target and in its inspector select the database name and object Target name that you want to associate with the object Target instance.
16. If there are no object target databases in your project, there will be a button to import a default Object Target.
17. Add your augmented content as a child to the Object Target Game Object. Then test your scene in Play mode by pressing the play button.

Android setup for Unity:

1. Install Android Build support from Unity Hub
2. For Vuforia: Vuforia build support from Unity Hub
3. For ARFoundation: Windows->Package Manager-> ARFoundation & ARCore XR Plugin

Turning ON Developer mode

1. Turn on Developer mode on android phone by going to settings->about phone->software information->click on build number 7 times.
2. Developer mode turns on, then enable USB debugging in settings->developer options

IMPLEMENTATION:

Augmented Image Controller

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

public class AugmentedImageController : MonoBehaviour
{
    [SerializeField] private AugmentedImageVisualizer
_augmentedImageVisualizer;
    private readonly Dictionary<int, AugmentedImageVisualizer>
_visualizers = new Dictionary<int, AugmentedImageVisualizer>();

    private readonly List<AugmentedImage> _images = new
List<AugmentedImage>();

    // Update is called once per frame
    void Update()
    {
        if(Session.Status != SessionStatus.Tracking)
        {
            return;
        }
        Session.GetTrackables(_images,
TrackableQueryFilter.Updated);
        VisualizeTrackables();
    }

    private void VisualizeTrackables()
    {
        foreach(var image in _images)
        {
            var visualizer = GetVisualizer(image);

            if (image.TrackingState == TrackingState.Tracking &&
visualizer == null && image.TrackingMethod ==
AugmentedImageTrackingMethod.FullTracking)
            {
                AddVisualizer(image);
            }
            else if(image.TrackingMethod ==
AugmentedImageTrackingMethod.LastKnownPose && visualizer != null)
            {
                RemoveVisualizer(image, visualizer);
            }
        }
    }

    private void RemoveVisualizer(AugmentedImage image,
AugmentedImageVisualizer visualizer)
    {
        _visualizers.Remove(image.DatabaseIndex);
        Destroy(visualizer.gameObject);
```

```
        }

        private void AddVisualizer(AugmentedImage image)
        {
            var anchor = image.CreateAnchor(image.CenterPose);
            var visualizer = Instantiate(_augmentedImageVisualizer,
    anchor.transform);
            visualizer.Image = image;
            _visualizers.Add(image.DatabaseIndex, visualizer);
        }

        private AugmentedImageVisualizer GetVisualizer(AugmentedImage
    image)
        {
            AugmentedImageVisualizer visualizer;
            _visualizers.TryGetValue(image.DatabaseIndex, out
    visualizer);
            return visualizer;
        }
    }
```

Augmented Image Visualizer

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

    public class AugmentedImageVisualizer : MonoBehaviour
    {
        [SerializeField] private GameObject[] placeablePrefabs;
        public AugmentedImage Image;
        //private VideoPlayer _videoPlayer;

        // Start is called before the first frame update
        void Start()
        {
          // _videoPlayer = GetComponent<VideoPlayer>();
            //_videoPlayer.loopPointReached += OnStop;
            foreach(GameObject prefab in placeablePrefabs)
            {
                GameObject newPrefab = Instantiate(prefab,
    Vector3.zero, Quaternion.identity);
                newPrefab.SetActive(false);
            }

        }

        /*private void OnStop(VideoPlayer source)
        {
            gameObject.SetActive(false);
        }*/

        private void Update()
        {
            if (Image == null || Image.TrackingState !=
    TrackingState.Tracking)
            {
```

```
                return;
            }

            if
    (!placeablePrefabs[Image.DatabaseIndex].activeInHierarchy)
            {
                //_videoPlayer.clip =
    _videoClips[Image.DatabaseIndex];
                //_videoPlayer.Play();

    placeablePrefabs[Image.DatabaseIndex].SetActive(true);
            }

            transform.localScale = new Vector3(Image.ExtentX,
    Image.ExtentZ, 1f);

        }
    }
```

C# Script for Image Tracking

```
using GoogleARCore;
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Video;
using UnityEngine.XR;
using UnityEngine.XR.ARFoundation;


public class ImageTracking : MonoBehaviour
{
    [SerializeField] private GameObject[] placeablePrefabs;
    private readonly Dictionary<string, GameObject> spawnedPrefabs =
new Dictionary<string, GameObject>();

    //private readonly List<AugmentedImage> _images = new
List<AugmentedImage>();
    private ARTrackedImageManager trackedImageManager;

    public void Awake()
    {
        trackedImageManager =
FindObjectOfType<ARTrackedImageManager>();

        foreach(GameObject prefab in placeablePrefabs)
        {
            GameObject newPrefab = Instantiate(prefab, Vector3.zero,
Quaternion.identity);
            newPrefab.name = prefab.name;
            spawnedPrefabs.Add(prefab.name, newPrefab);
        }
    }

    private void OnEnable()
    {
        trackedImageManager.trackedImagesChanged += ImageChanged;
```

```
    }

    private void OnDisable()
    {
        trackedImageManager.trackedImagesChanged -= ImageChanged;
    }

    private void ImageChanged(ARTrackedImagesChangedEventArgs
eventArgs)
    {
        foreach(ARTrackedImage trackedImage in eventArgs.added)
        {
            UpdateImage(trackedImage);
        }

        foreach (ARTrackedImage trackedImage in eventArgs.updated)
        {
            UpdateImage(trackedImage);
        }

        foreach (ARTrackedImage trackedImage in eventArgs.removed)
        {
            spawnedPrefabs[trackedImage.name].SetActive(false);
        }
    }

    private void UpdateImage(ARTrackedImage trackedImage)
    {
        string name = trackedImage.referenceImage.name;
        Vector3 position = trackedImage.transform.position;

        GameObject prefab = spawnedPrefabs[name];
        prefab.transform.position = position;
        prefab.SetActive(true);

        foreach(GameObject go in spawnedPrefabs.Values)
        {
            if(go.name != name)
            {
                go.SetActive(false);
            }
        }
    }
}
```

CONCLUSION:

Plane detection and marker detection has been implemented successfully.

QUESTIONS:

1. How to add Vuforia in Unity?
2. How to add AR camera in Unity 2020?
3. How to make an AR app in Unity 2020?
4. Which is better AR Foundation or Vuforia?
5. How can AR be implemented on Android and iOS?

Wadia College of Engineering

Department of Computer Engineering

Name of Student:	**Class:**
Semester/Year:	**Roll No:**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part II - 6**

LP II – PART II Augmented & Virtual Reality

ASSIGNMENT NO: 06

AIM: **Mini-Projects/ Case Study**

Create a multiplayer VR game (battlefield game). The game should keep track of score, no. of chances/lives, levels (created using different scenes), involve interaction, animation and immersive environment.

OR

Create a treasure hunt AR application which should have the following features: A help button for instruction box to appear. A series of markers which would give hints on being scanned. Involve interaction, sound, and good UI.

OBJECTIVES: To implement an AR or VR game.

THEORY:

IMPLEMENTATION:

GETTING STARTED WITH A SIMPLE UNITY3D PROJECT

- Start Unity3D
- New Project (3D or 2D)
- File->Save Scene

Font size too small?

1. Right-click on the desktop
2. Select “Display settings”
3. “Change the size of text”
4. Default to 125%, change to 175%

- for Unity IDE presentations

Getting started with a simple Unity3D project

- Assets->Import Packages

  - Characters

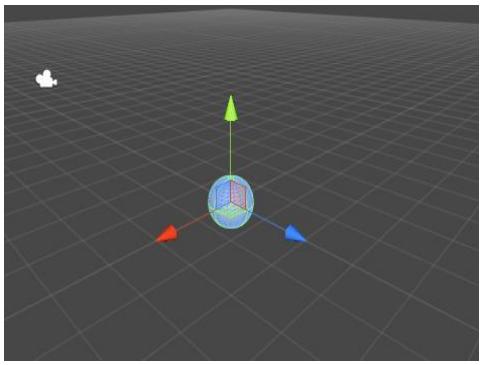
  - Particle Systems

  - Vehicles

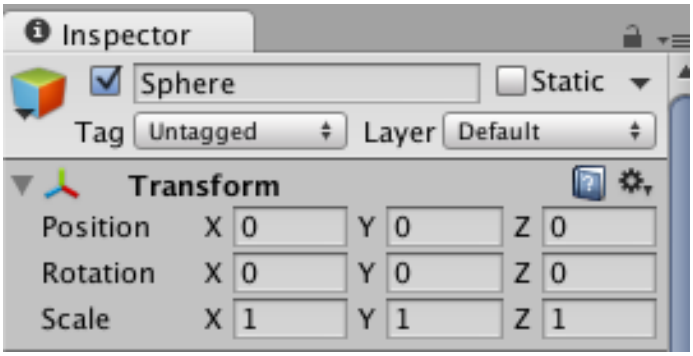
- Select components before importing each package.

Object Creation

- GameObject->3D object->sphere
- Edit->Frame Select (to show the created object)
- Hold onto the arrows to move the sphere or change the position in the Inspector.
- Note: Y is up.



- Inspector and transform panel



- Moving, translation, rotation and scaling tools

Unity 3D: Component

Select Created Game Object->Component->Physics->Rigidbody

- Rigid Objects: non-deformable with physical properties (gravity, inertial).
- Non-rigid Objects:
  - Deformable: changeable geometry

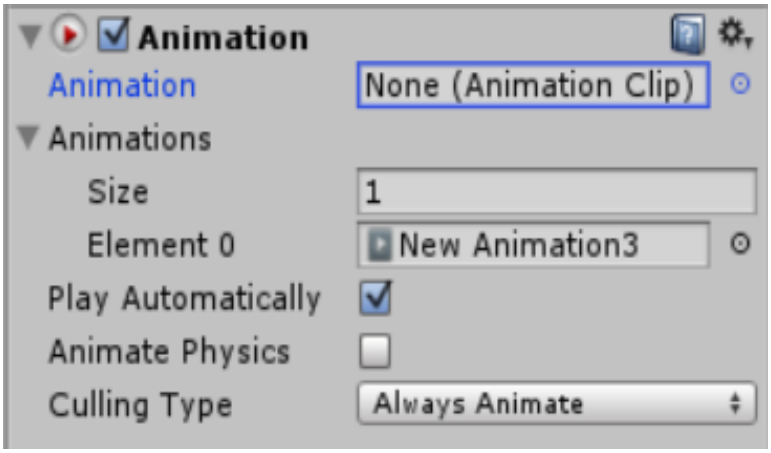
- Breakable: changeable topology.

- Intangible Objects: No predefined shape.

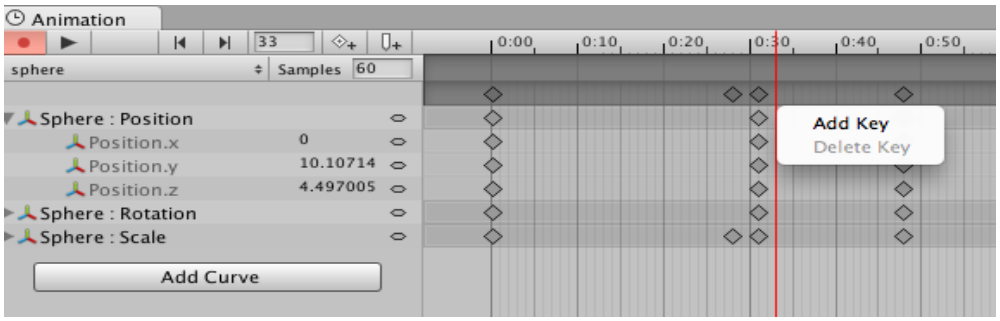
  fire, clouds, …

Animation

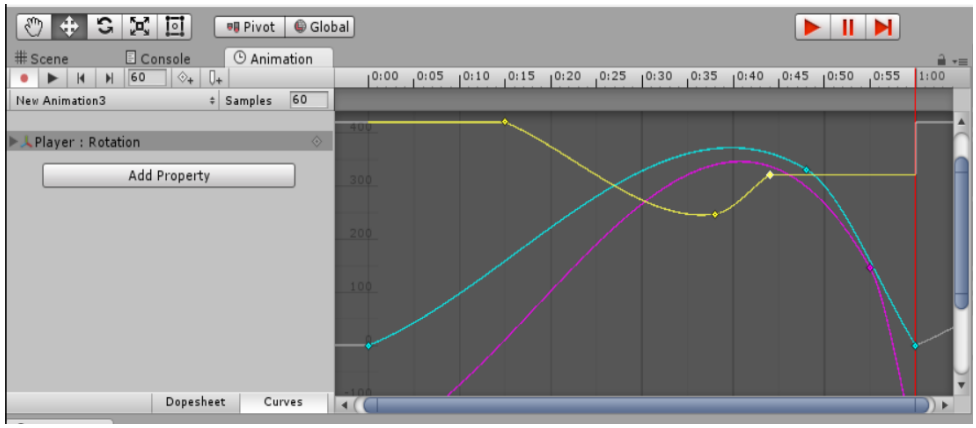
- Window->Animation
- Click on the object to be animated.
- Component->Miscellaneous->Animation
- Create New Animation Clip
- Click on the record button (red) at the top-left.
- Save the new animation clip file.
- Click on Curves.
- Select Add Property->transform → position, rotation or scaling



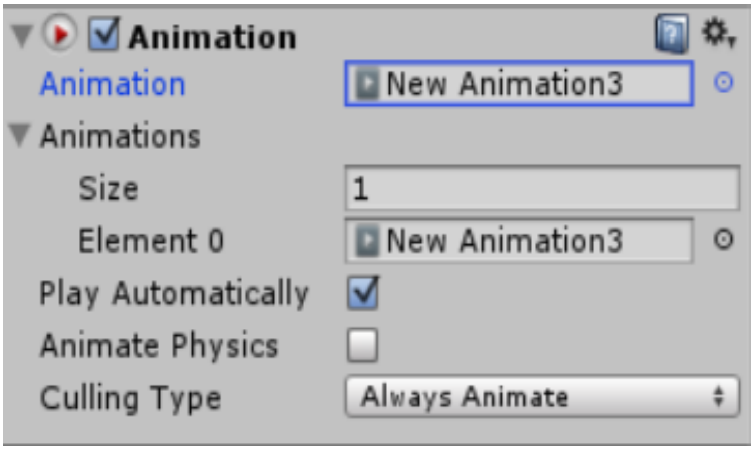
- Add key frame on the timeline as many as you want.



- Click on the red button again to finish making the animation clip.

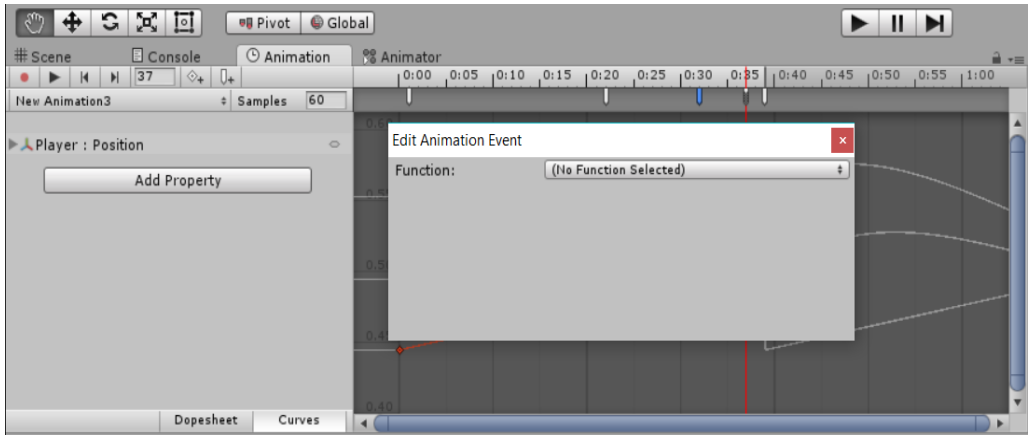


- Go back to Unity window.
- Under Inspector → Animation , change the name of animation clip from none to the one that you have made.



Animation Event

- Allows you to call functions in the object's script at specified points in the timeline.
- Add a new Animation Event by double-clicking the Event Line or by using the Event button.
- When you add an event, a dialog box will appear to prompt you for the name of the function and the value of the parameter you want to pass to it.



Light

- Game Object ->Light -> Directional Light (Default)
- Move and rotate just like any other object.

Terrain

- GameObject → 3D Object → Terrain
- In the hierarchy panel, select Terrain.
- In the Inspector: set x = -5, y = 0, z = -5.
- Click on one of the action icons in Terrain (Script) to raise/lower terrain, paint height, smooth height, paint texture, place trees, paint details, terrain setting. Adjust brush size to 1 before performing the operations.
- Paint Texture: Edit Texture -> Select
- To place trees, you have to build tree first (Game Object->3D object ->Tree), then choose"Edit Trees -> Add Trees" first to add different types of trees.
- In the Add Trees popup window, you need click on the little circle at the right-most of the "Tree" row.
- Select, say, Palm and then click on "Add" in the "Add Tree" window.
- Go back to the Inspector, select "Mass Place Trees" from available "Trees" to add.

Physics

- Unity3D provide Physics library.

  - Rigidbody, collider, joint, force and etc.
- Rigidbody component : gravity automatically added
- Collision detection : box, sphere, capsule , mesh, whelle and terrain.
- Collision call back function : OnCollisionEnter, OnCollision and OnCollisionExit.

Player (for a third-person game)

- Make sure to save the scene "File->Save Scene" (Ctrl S") and save the project "File->Save Project"

- Next, we need add the player.
- In the Project window, drag“Standard Assets->Character Controllers->3rd Person Controller”
  to the Hierarchy window.
- In the Hierarchy window, double-click on the 3rd Person Controller.
- Click on the “Move selected object” icon. Then move the controller to the top of the terrain. You may have to adjust your view angle by clicking on the ‘xyz” icon to see the position. Click the middle of the icon to get the perspective view.
- Now click on the “Play” icon and use the arrow keys to controll the player.
- You should see the player running around and make sure he does not run off the edge.

CONCLUSION: An Augmented Reality or Virtual Reality game has been implemented successfully.