THE HONG KONG POLYTECHNIC UNIVERSITY
Department of Electrical and Electronic Engineering

EIE3360 Integrated Project
Tutorial 1 – A Simple Unity Game

1. Objective

In this tutorial exercise, you will use Unity to build a simple game for Android.

2. Preparation

- A PC with Microsoft Windows
- Unity 2022.3.x
- Microsoft Visual Studio
- Android SDK
- Basic knowledge of C# programming, object-oriented programming and 3D graphics

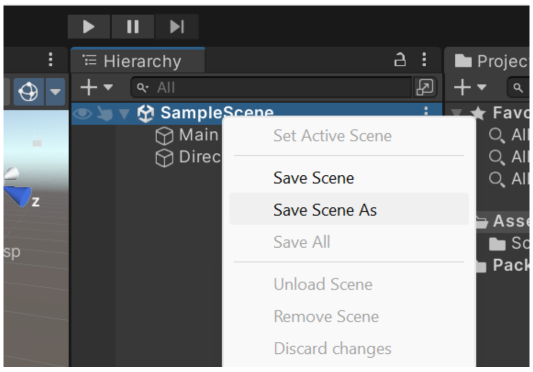
3. Developing a Unity Android Game

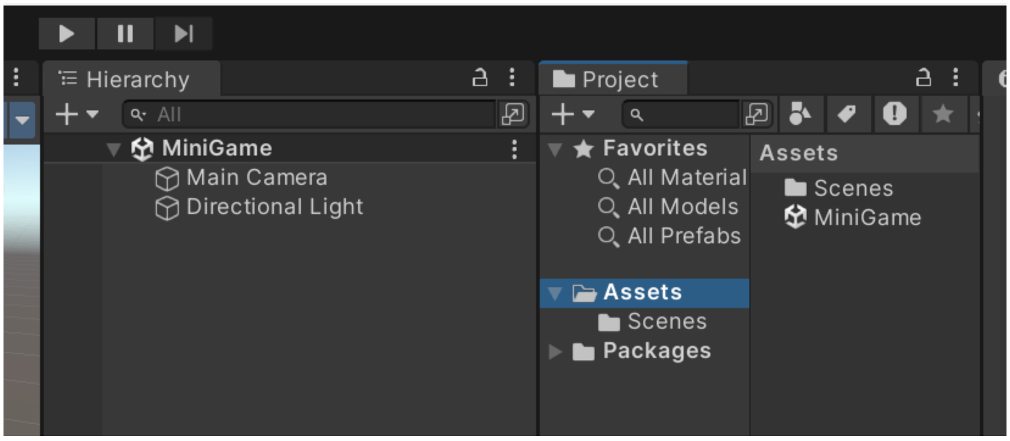
Unity is a game development ecosystem that provides a powerful rendering engine fully integrated with a complete set of intuitive tools and rapid workflows to create interactive 3D and 2D content. With Unity, the developer can publish their game to multiple platforms such as Windows, Mac, iOS, Android, PlayStation, XBOX, etc.
This tutorial exercise uses Unity to create a simple Roll-a-Ball game for Android. In the game, you will create several game objects. The player will control a ball rolling around the game board and move the ball using physics and force. You will also detect contacts between the ball and the pick-up game objects.

3.1 Setting up the Unity Game

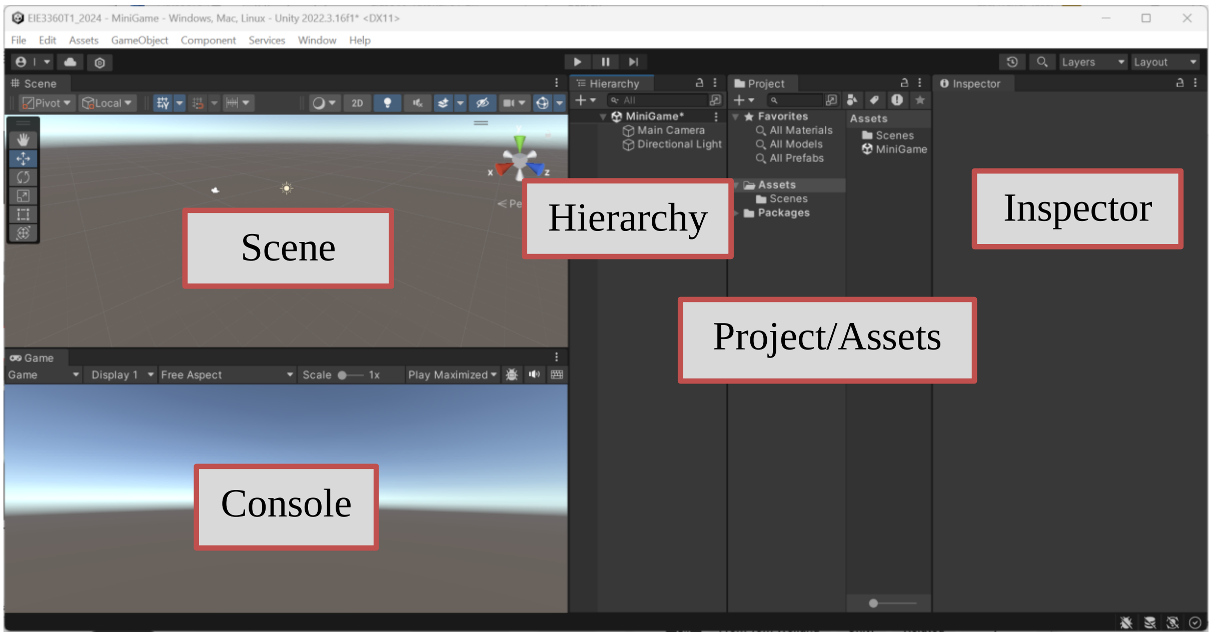
1. **Create a new Unity Project.**
   a. Click the Unity hub and sign in with your account.
   b. Get a free license.
   c. Click 'New project' → select 3D → Input the project name "**EIE3360Tut1**".

   Note: at this time, you NO need to import any packages.
   d. Right-click 'SampleScene' → Save Scene As 'MiniGame' → Save Scene under the Assets folder.





e. Change the layout to 2 by 3 at the upper right corner.

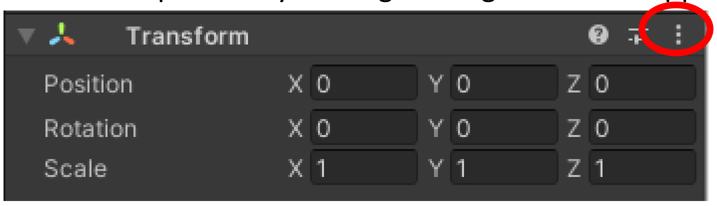


1. Create a plane for the game from the menu bar

a. Click GameObject → 3D Object → Plane.

b. Rename the plane to ‘Ground’.

   c. 'Reset' the transform component by clicking the cog icon in the upper right corner.



Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

   d. Double click the name "Ground" to focus on the ground plane.
   e. Click on the plane and set the **Scale** to X = 2, Y = 1, Z = 2.

2. **Create a sphere from the 'Hierarchy' list on the left**
   a. Right click "Hierarchy" list → 3D Object → Sphere.
   b. Rename the sphere to 'Player'.
   c. Reset the transform component.
   d. Double click the name "Player" to focus on the sphere.
   e. Set the **Position** to X = 0, Y = 0.5, Z = 0.

3. **Create another directional light from the 'Hierarchy' list on the left**
   a. Create a new directional light by select GameObject → Light → Directional Light
   b. Rename the directional light to 'Main Light'.
   c. Reset the transform component.
   d. Set the **Rotation** to X = 30, Y = 60, Z = 0.
   e. Set the Shadow Type → Soft Shadows.

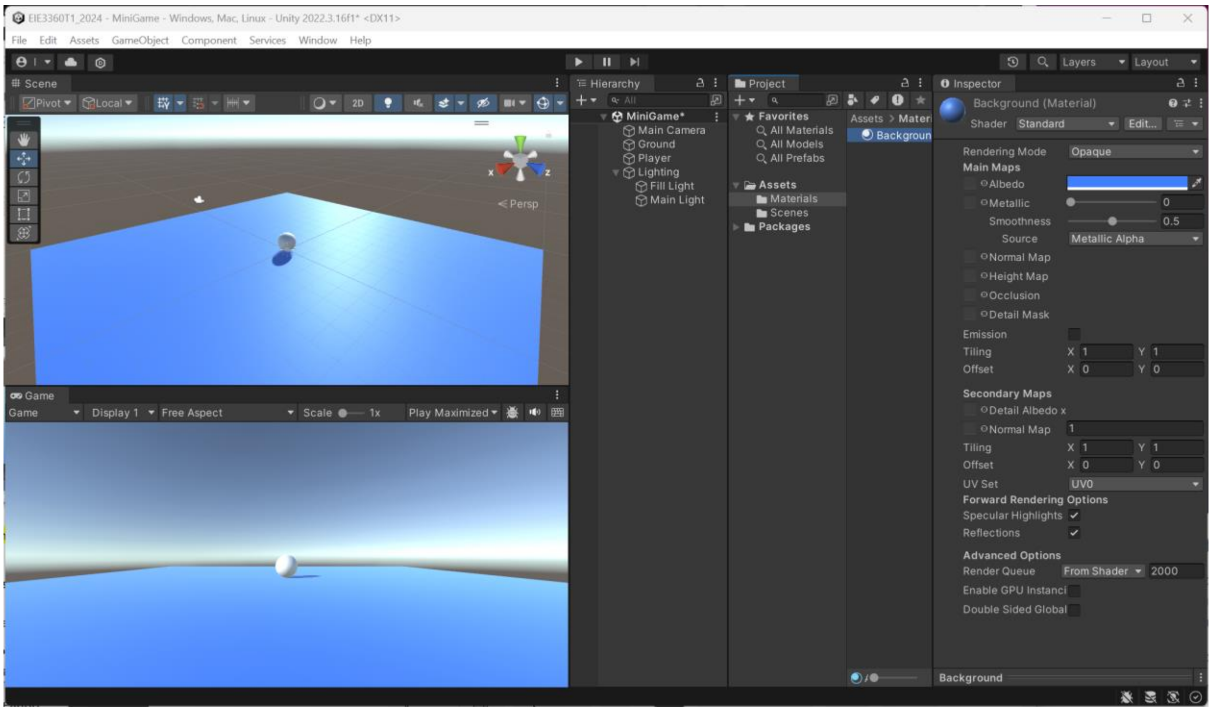
4. **Create a fill light**
   a. Click Edit → Duplicate, duplicate the "Main Light" and rename it to 'Fill Light'.
   b. Revise the **Rotation** to X = -30, Y = -60.
   c. Change the color value to R = 173, G = 255, B = 255, A = 255.
   d. Lower the intensity to 0.1.
   e. Turn off Shadow (Shadow Type → No Shadows).

5. **Delete the default Directional Light by selecting it and pressing the 'Delete' key.**

6. **Organise the hierarchy by using empty game objects**
   a. Click GameObject → Create Empty (to create an empty game object).
   b. Rename it to 'Lighting'.
   c. Reset the transform component.
   d. Drag both the Fill Light and Main Light to the Lighting object.

7. **Add a texture to the ground**
   a. Create a folder called 'Materials' under Assets in the Project view.
   b. Open the 'Materials" folder and right-click Create → Material (to create a material).
   c. Rename it to 'Background'.

d. Select the material, and set the Albedo's colour to R = 64, G = 128, and B = 255.
e. Drag the material onto the plane to apply the texture.
f. Save the scene and save the project.



Background (Material)
Shader: Standard
Rendering Mode: Opaque
Main Maps
Albedo
Metallic: 0
Smoothness: 0.5
Source: Metallic Alpha
Normal Map
Height Map
Occlusion
Detail Mask
Emission
Tiling: X 1, Y 1
Offset: X 0, Y 0
Secondary Maps
Detail Albedo x
Normal Map: 1
Tiling: X 1, Y 1
Offset: X 0, Y 0
UV Set: UV0
Forward Rendering Options
Specular Highlights ✓
Reflections ✓
Advanced Options
Render Queue: From Shader 2000
Enable GPU Instanci
Double Sided Global

Checkpoint 1: Demo to the instructor or teaching assistant

3.2 Moving the Player

1. **The game object needs a 'Rigidbody' component to move the ball.**
   a. Select a Player.
   b. Click 'Add Component' in the Inspector window → Rigidbody.

The purpose is to let the 'Player' object move under our control and use a script to get input from the keyboard and apply it to the Player object.

   c. In the Project view at the bottom, select the Assets folder.
   d. Right-click the Assets folder → select Create → Folder and name it 'Scripts'.
   e. Right-click the Scripts folder → select Create → C# Script and name it "PlayerController".
   f. In the Hierarchy view, select the Player.
   g. Click the 'Add Component' button in the Inspector window on the right.
   h. Search 'Player Controller' and add the Player.
   i. Go to the 'Script' folder and double-click the script 'PlayerController' to launch Visual Studio to edit the script.
   j. Create a 'FixedUpdate' function to listen to the keys A, W, S and D from the keyboard to control the sphere.
   k. **Edit the codes** as follows:

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

public class PlayerController : MonoBehaviour
{
    private Rigidbody rb;

    // Use this for initialization
    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }
    void FixedUpdate ()

//Update() runs once per frame. FixedUpdate() can run once, zero, or several
times per frame, depending on how many physics frames per second are set in the
time settings and how fast/slow the framerate is.
    {
        float moveHorizontal = 0;
        float moveVertical = 0;

        if (Input.GetKey(KeyCode.A))
            moveHorizontal = -1;
        if (Input.GetKey(KeyCode.D))
            moveHorizontal = 1;
        if (Input.GetKey(KeyCode.W))
            moveVertical = 1;
        if (Input.GetKey(KeyCode.S))
            moveVertical = -1;


       Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);
        rb.AddForce(movement * 10);
    }
}
```

l. Save the code and close Visual Studio.

m. In Unity, press the 'Play' button to run the game.

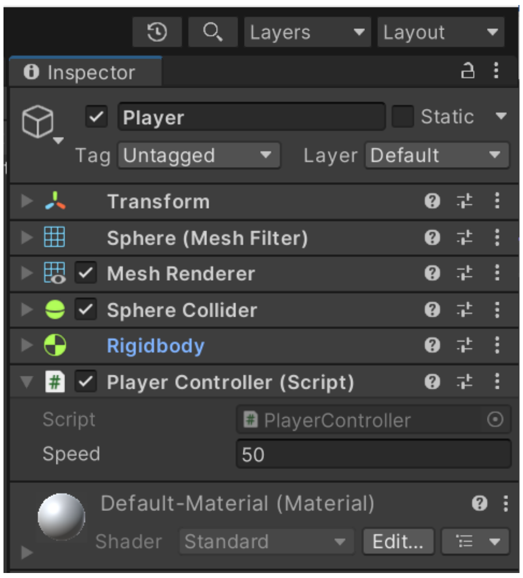
n. Try to press the keys to move the sphere. (if you want to stop the game, by pressing the Play button again).

o. Create a public float member variable 'speed' in the script and change the 'AddForce' value to **(movement * speed)**.

```
public class PlayerController : MonoBehaviour
{
    private Rigidbody rb;
    public float speed;
…
…
…
        Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);
        rb.AddForce(movement * speed);
    }
}
```

p. Build the project and make sure there is **no compilation error**.

q. Modify the speed to 50 in the Inspector window.



Layers
Layout
Inspector
Player
Static
Tag Untagged
Layer Default
Transform
Sphere (Mesh Filter)
Mesh Renderer
Sphere Collider
Rigidbody
Player Controller (Script)
Script PlayerController
Speed 50
Default-Material (Material)
Shader Standard
Edit...

r. Run the game again.

s. Save the scene and the project.

Note:
You can refer to the Unity Script Reference from https://docs.unity3d.com/ScriptReference/Input.html to search the Interface into the input system. Read the Input class description, its variables and functions.

Checkpoint 2: Demo to the instructor or teaching assistant

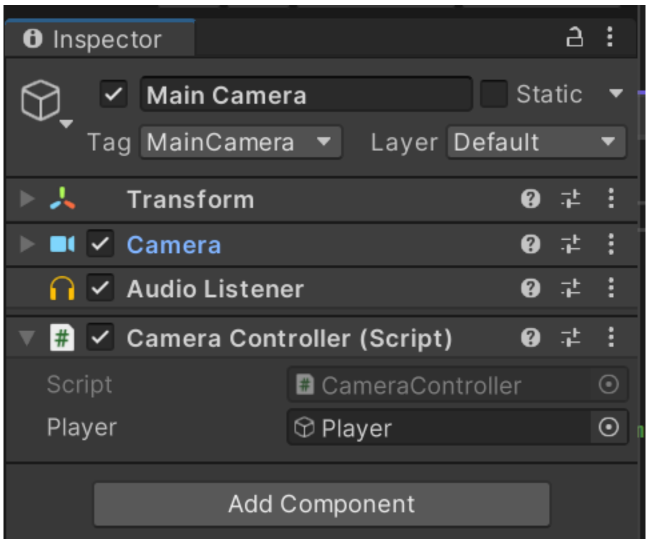
3.3 Moving the Camera

1. **Create a script to control the camera to follow the player.**
    a. Set the 'Main Camera' position to (0, 10, -10) and the rotation to (45, 0, 0).
    b. Create a new C# script inside the Scripts folder for the camera and name it 'CameraController'.
    c. Double-click the script to open Visual Studio to edit it as below.

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

public class CameraController : MonoBehaviour
{
    public GameObject player;
    private Vector3 offset;

//Sets the depth bias on the GPU. Depth bias, also called depth offset, is a
setting on the GPU that determines the depth at which it draws geometry. Adjust
the depth bias to force the GPU to draw geometry on top of other geometry at the
same depth.
```

```
    void Start()
    {
        offset = transform.position;
    }

    void LateUpdate()
//every frame but after Update() method is executed. Therefore, if you want to
check and control something after the Update() method finishes its job, you need
to use LateUpdate() method
    {
        transform.position = player.transform.position + offset;
    }
}
```

d. Save the file.
e. Add the Camera Controller script to the Main Camera.
f. Drag the Player game object into the 'Player' slot in the Camera Controller (Script) component in the Inspector window.



Script: CameraController
Player: Player

g. Enter game mode, and you will see the camera following the ball.

3.4 Creating the Pick-up objects

To avoid the ball rolling outside the ground, we need to place the walls around the edges and create a set of collectable objects for the player to collect.

1. To create walls,

a. Create an empty game object named Walls in the Hierarchy window.
b. Change the Position to X=0, Y=0.5, Z=0.
c. Change the Scale to X=1, Y=1, Z=1.
d. Create a 3D object, select 'Cube', name it 'West Wall', and drag it to the Walls object.
e. Change the Position to X = -10, Y = 0, Z = 0.
f. Change the Scale to X = 0.5, Y = 2, Z = 20.5.
g. Duplicate the 'West Wall' and make the other 3 walls: East Wall, North Wall, and South Wall.

East Wall:	Position X = 10, Y = 0, Z = 0	Scale to X = 0.5, Y = 2, Z = 20.5
North Wall:	Position X = 0, Y = 0, Z = 10	Scale to X = 20.5, Y = 2, Z = 0.5
South Wall:	Position X = 0, Y = 0, Z = -10	Scale to X = 20.5, Y = 2, Z = 0.5

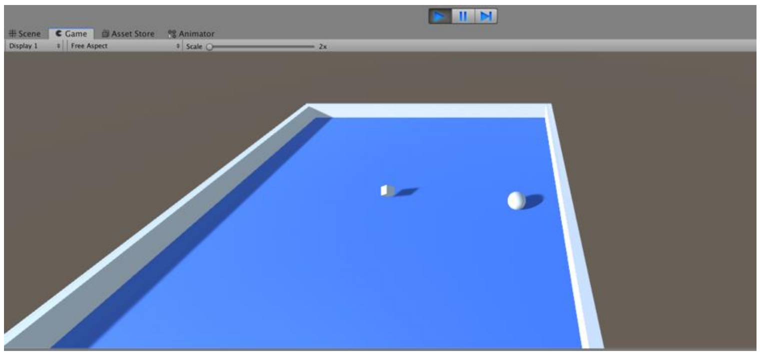
h. Enter game mode and test it.

2. Create the collectable objects

a. Create a cube and rename it to ‘PickUp’.
b. Double-click to focus on the ‘PickUp’ Object.
c. Set the Position to X = 0, Y = 0.5, Z = 0.
d. Set the Rotation to X = 45, Y = 45, Z = 45.
e. Scale it to X = 0.5, Y = 0.5, Z = 0.5.
f. Create a C# script and name it ‘Rotator’ in the Scripts folder in the Project window.
g. Drag the ‘Rotator’ script to the ‘PickUp’ object.
h. Edit the ‘Rotator’ script as below.

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

public class Rotator : MonoBehaviour
{
    // Update is called once per frame
    void Update()
    {
        transform.Rotate(new Vector3(15, 30, 45) * Time.deltaTime);
    }

// deltaTime is the time in seconds between the last and current frames. Since
Update is called once per frame, Time. deltaTime can be used to make something
happen at a constant rate regardless of the framerate.
}
```

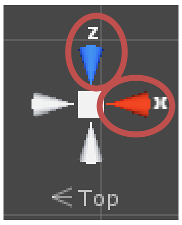
i. Save the code and test it in the game mode.



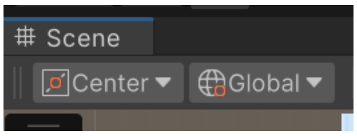
3. To create a Prefab

A Prefab is an asset that contains a template for the game object and places it around the game area. With the “PickUp” Prefab, you can conveniently make changes to a single instance or for all the ‘PickUp’ objects.

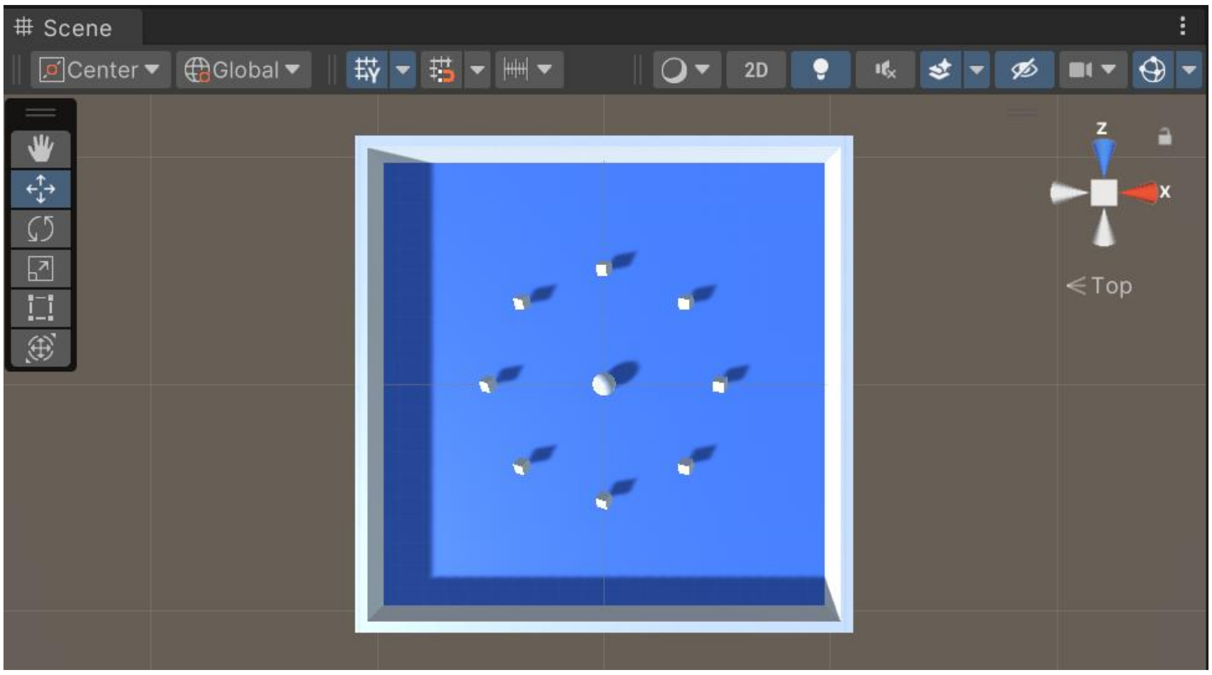
1. **Making the Prefabs**
   a. Create a folder called ‘Prefabs’ under Assets.
   b. Drag the ‘PickUp’ game object from the Hierarchy window into the Prefabs folder.
   c. Create a new empty game object called ‘**PickUps**’ to hold the PickUp objects.
   d. Set the transform position to the origin and drag the ‘PickUp’ game object into the ‘**PickUps**’ object.
   e. Move to the top-down view by clicking on the ‘gizmo’ in the upper right of the scene view.



   f. Change the editor from ‘Local’ to ‘Global’ mode so that the orientation of the gizmo changes.
   g. Drag game objects around relative to the world’s global axes.



   h. Move the first ‘PickUp’ object to the upper part of the play area, then Press **Ctrl + D** on PC (cmd + D on Mac) to duplicate 7 more ‘PickUp’ game objects
   i. Place them around the play area in a circle.
   j. Save the scene and the project.

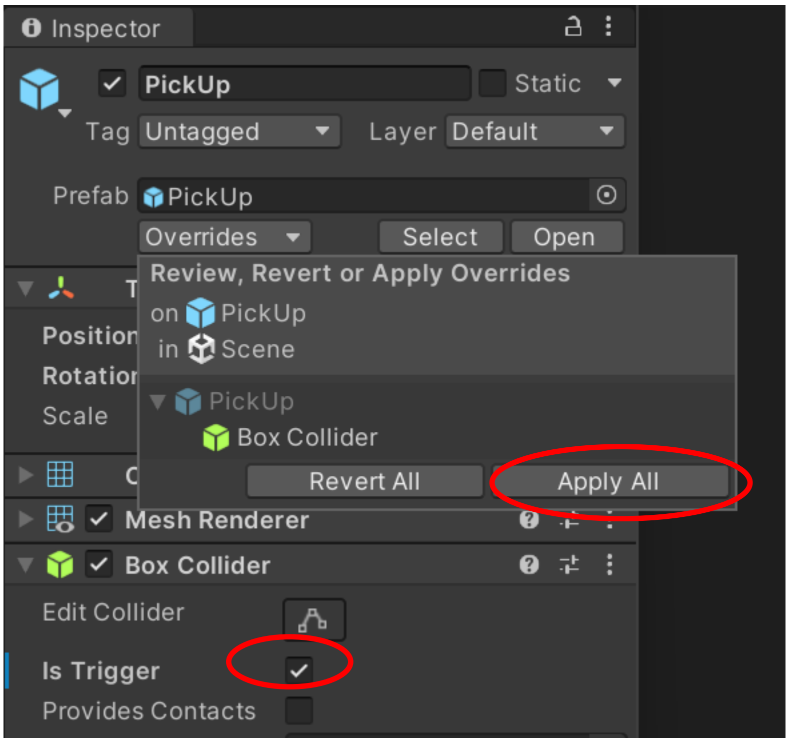


3.5 Collecting the Pick-up Objects

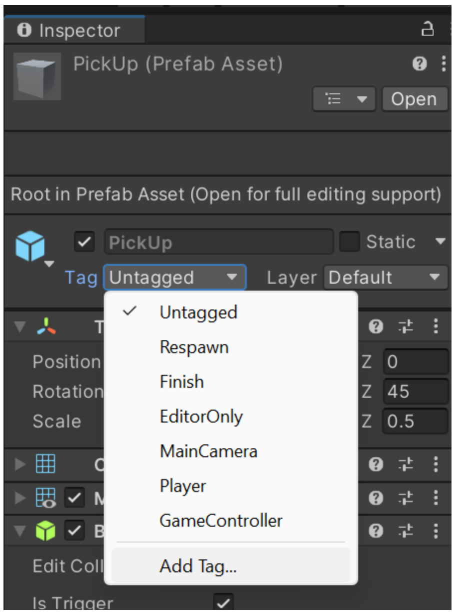
1. **To collect the PickUp game objects, you need to write a script to detect the collision between the Player game object and the PickUp game objects.**
   a. Edit the 'PlayerController' script to add an 'OnTriggerEnter()' function.
   b. OnTriggerEnter() will be called by Unity when the Player game object first touches a 'trigger collider'.
   c. A reference to the trigger collider 'other' is given when it touches the Player game object
   d. Deactivate the game object if it is a 'PickUp' game object.

```
void OnTriggerEnter(Collider other)
{
    if (other.gameObject.CompareTag ("PickUp"))
    {
        other.gameObject.SetActive(false);
    }
}
```

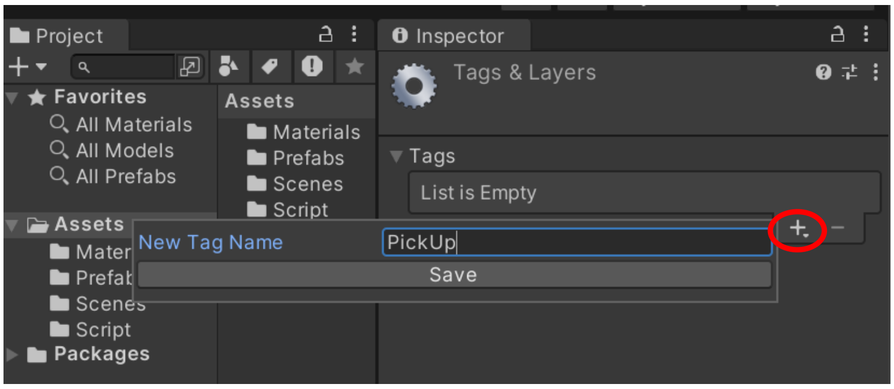
   e. Select a 'PickUp' object in the Box Collider in the Inspector window, check 'Is Trigger'.
   f. Click 'Overrides' -> 'Apply All' at the top to apply to the Prefab.



2. **Set up the tag value for the PickUp game object**
   a. Select the Prefab asset for the PickUp object.
   b. Click ‘Add Tag…’ in the Tag drop down list and click ‘+’ to add a new tag.



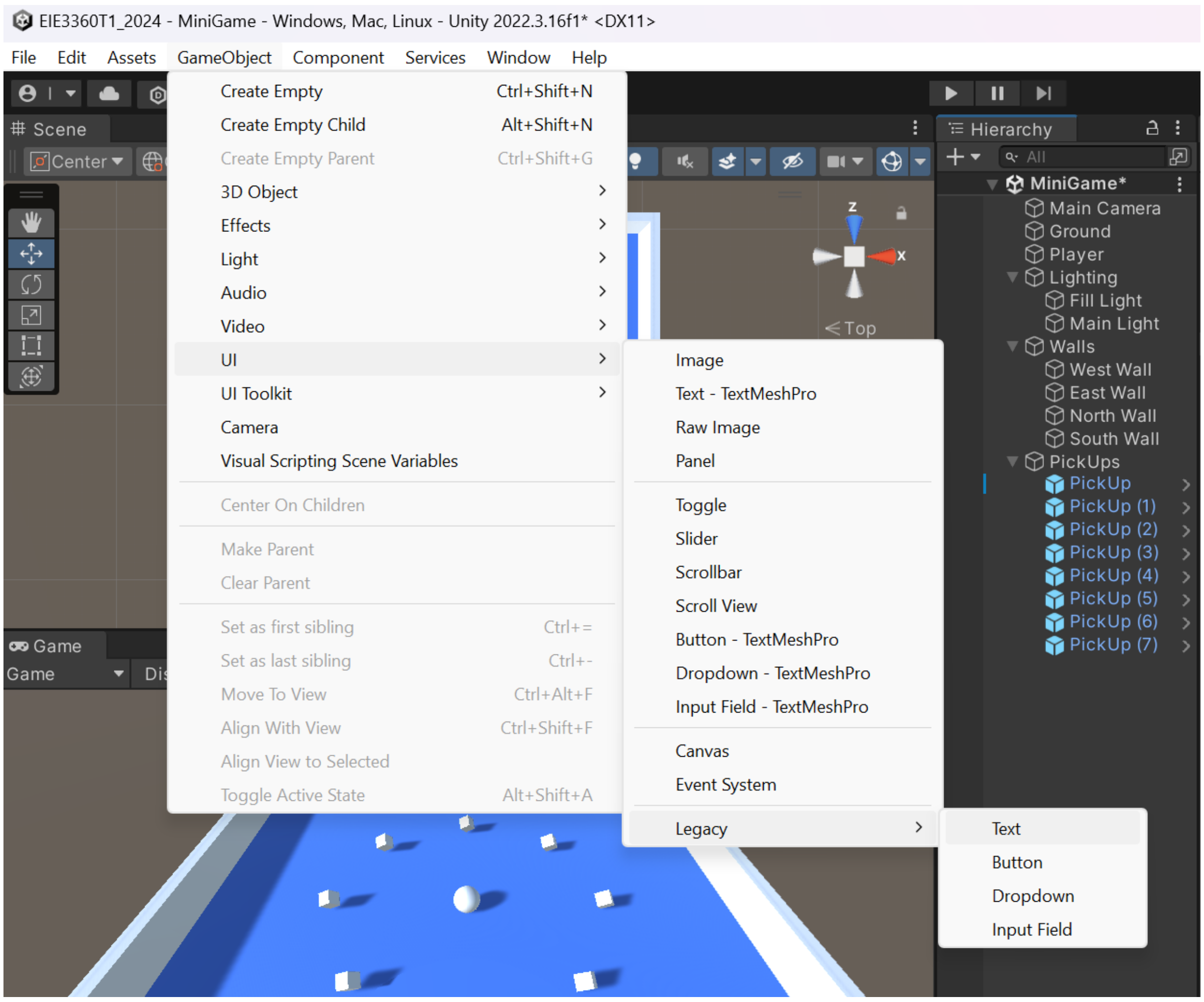
   c. Type ‘PickUp’ for the New Tag Name (Tag 0).
   d. Press ‘Save’ to confirm the tag.



   e. Back the Prefab asset, select ‘PickUp’ in the Tag drop down list.
   f. All ‘PickUp’ game objects should now be assigned the tag ‘PickUp’.
   g. To optimise the performance, add a ‘Rigidbody’ to the Prefab asset for the ‘PickUp’ object and select ‘Is Kinematic’.
   h. Enter game mode and test your game.

3.6 Displaying the text

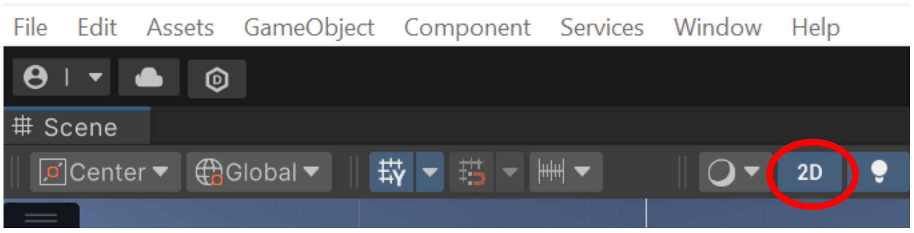
1. **Display some information on the screen**
   a. In the Hierarchy, select GameObject → UI → Legacy → Text.



b. A 'Text (Legacy)' GameObject is created under Canvas (if you cannot find Canvas, please search 'Canvas Group' directly when adding a component).

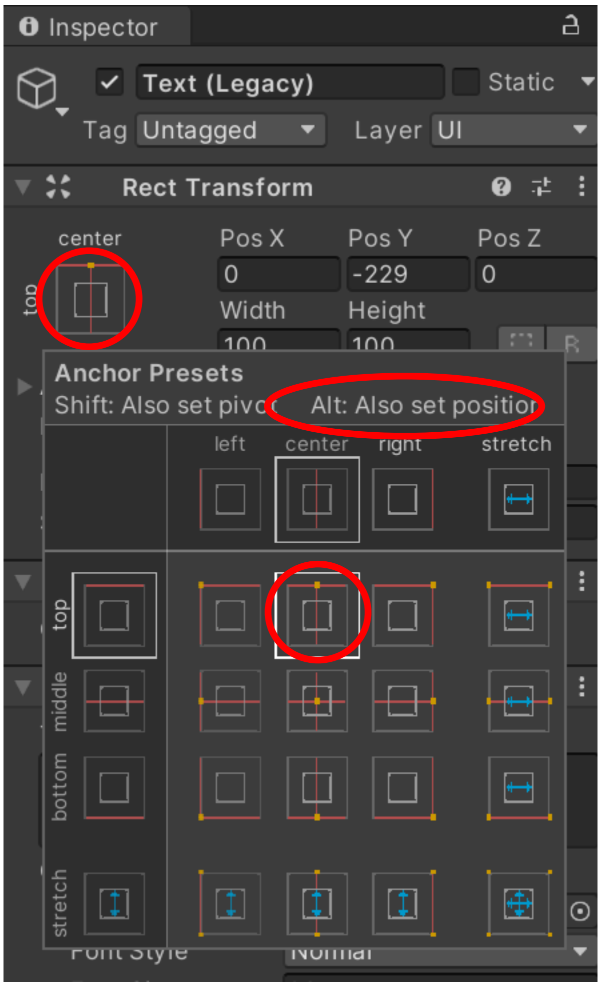
c. Reset the transform to the origin.

d. Click the 2D button to switch to the 2D view.



e. Double-click the name 'Text (Legacy)' to focus on the text box.

f. Alt Set the Anchor to the top center.



g. Set the position to X = 0, Y = -80, Z = 0.
h. Set the Width = 800, Height = 80.
i. Set the 'Text' to your name and the Font Size to 30.
j. Set the paragraph alignment to 'center'.
k. Save the scene and project.

3.7 Publishing the Game

1. To build the Android game

a. Confirm the Android SDK path.
   - Click Edit → Preferences... → External Tools → Android.
   - Check the path of Android JDK, SDK…

b. Click File → Build Settings, choose Android for the platform and click "Switch Platform".

c. Click 'Player Settings…' to set the followings:

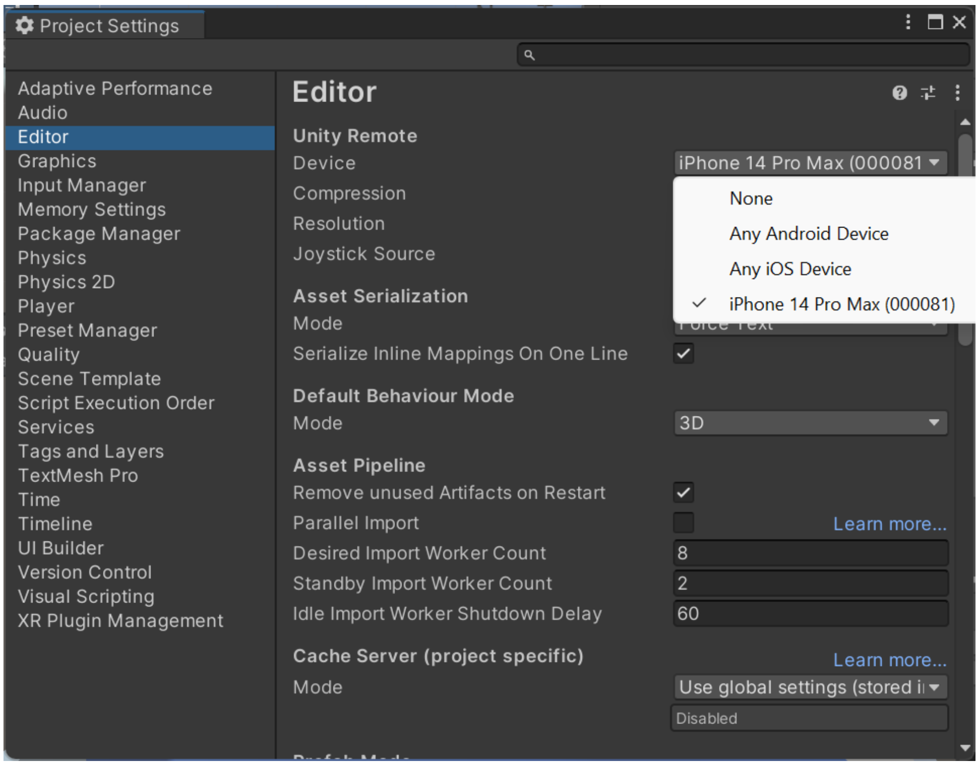
Resolution and Presentation

Default Orientation = **Landscape Left**

For testing purpose (Android and iPhone):

1. For iOS, iTunes is required on Windows.

2. For Android devices, enable Developer Mode and turn on USB Debugging. Choose "Transfer Photos" or "PTP" in USB Option.
3. Trust the PC and allow access to Photos.
4. Download Unity Remote 5 from Goggle Play/App Store.
5. Connect your device to the PC.
6. Project Settings -> Editor, Unity Remote->Device, choose your device



Project Settings

Adaptive Performance
Audio
Editor
Graphics
Input Manager
Memory Settings
Package Manager
Physics
Physics 2D
Player
Preset Manager
Quality
Scene Template
Script Execution Order
Services
Tags and Layers
TextMesh Pro
Time
Timeline
UI Builder
Version Control
Visual Scripting
XR Plugin Management

Editor

Unity Remote
Device: iPhone 14 Pro Max (000081
Compression
Resolution
Joystick Source

None
Any Android Device
Any iOS Device
✓ iPhone 14 Pro Max (000081)

Asset Serialization
Mode
Serialize Inline Mappings On One Line ✓

Default Behaviour Mode
Mode: 3D

Asset Pipeline
Remove unused Artifacts on Restart ✓
Parallel Import — Learn more...
Desired Import Worker Count: 8
Standby Import Worker Count: 2
Idle Import Worker Shutdown Delay: 60

Cache Server (project specific) — Learn more...
Mode: Use global settings (stored i
Disabled

7. Joystick source -> Local
8. Click Play and run the simulation on the Android or iOS Device.

d. Installation to Android devices:
   Build Settings -> Build and Run -> Save the APK file on PC, and installation on Android device should start automatically.
   No installation option for iOS on Windows PC.

4. Exercise

1. **Modify the code so that.**
   1. Add another UI Text to show how many "Pick-up" objects have been collected.
   2. The player tilts the Android device to control the ball and collect the Pickup objects.
   3. Hint: Read the Input.acceleration and Text section.
   4. Set the sphere colour to RED and the PickUps to Yellow.
   5. Export a WebGL application to run on the GitHub.
      Follow this link to build a WebGL application.

Unity WebGL Build and Hosting on GitHub
https://youtu.be/LAawMvpxOOM?si=L8H7heYo8HaAUcjL
6. Modify the C# code so that the same code can build both desktop (keyboard control) and mobile (tilts control) app.

Checkpoint 3: Demo to the instructor or teaching assistant

5. References

[1] Unity - Game Engine, http://unity3d.com/
[2] Unity - Roll-a-ball tutorial, https://learn.unity.com/project/roll-a-ball-tutorial
[3] Unity Manual, http://docs.unity3d.com/Manual/index.html
[4] Unity - Script API, http://docs.unity3d.com/ScriptReference/index.html