THE HONG KONG POLYTECHNIC UNIVERSITY
Department of Electrical and Electronic Engineering

EIE3360 Integrated Project
Laboratory 1B – Game Setup & Player

1. Objectives

In this laboratory exercise, you will use Unity to build a third-person stealth game for Android. You need to develop a game environment and player character for a stealth game,

1. **Game Setup and Lighting**
2. **Tag Management**
3. **Player Setup**
4. **Player Animator Controller**
5. **HashIDs**
6. **Player Movement**
7. **Player Health**

2. Preparation

- A PC with Microsoft Windows
- Unity 2022.3.16
- Unity Hub
- Android SDK
- Microsoft Visual Studio
- Basic knowledge in C# programming, object-oriented programming and 3D graphics

3. Developing a Unity Android Game

This laboratory exercise uses Unity to create a game called 'Stealth' for Android. In the game, the player sneaks around to avoid triggering the alarms and tries to escape from a maze where enemies are patrolling.

3.1 Game Setup and Lighting

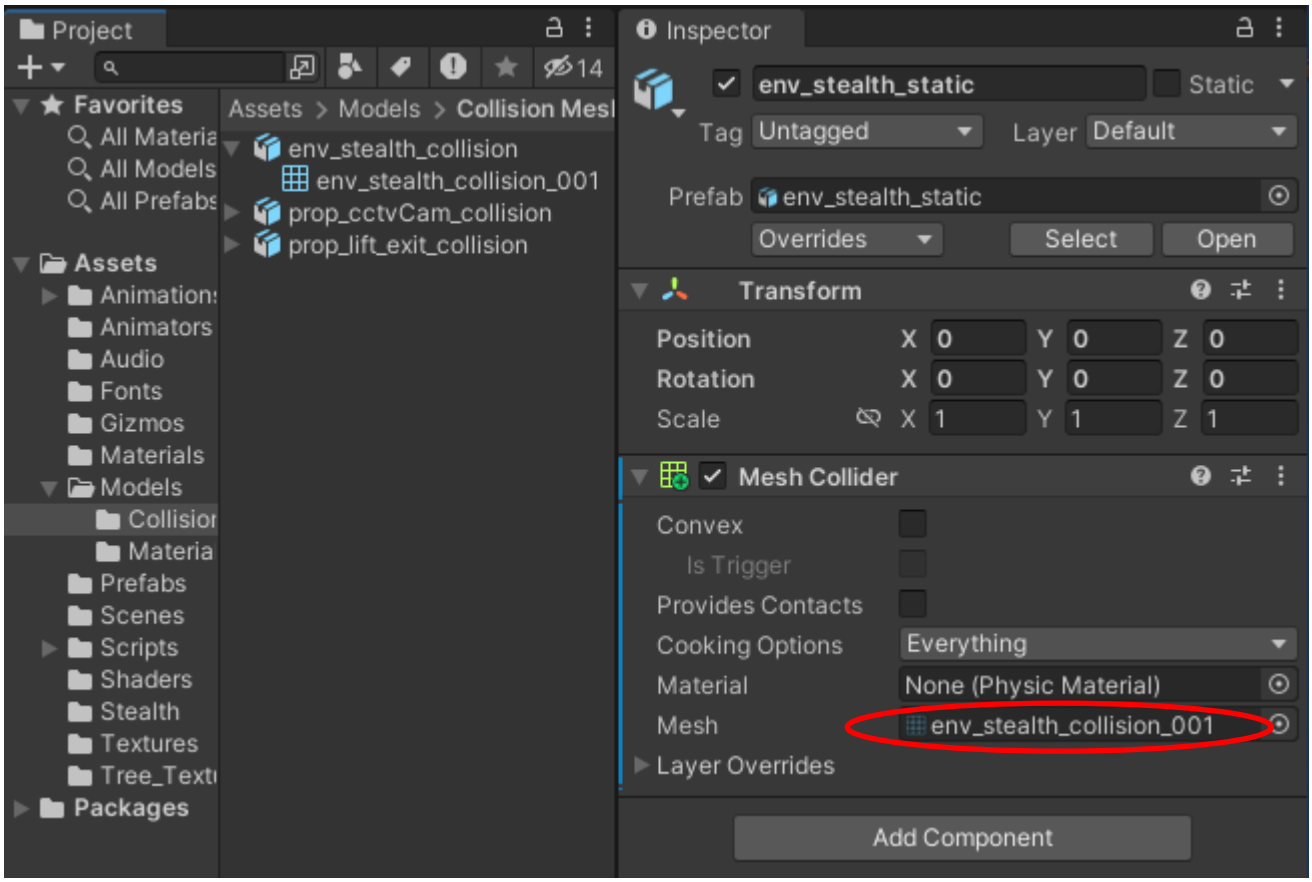
1. Create a new Unity Project

a. Create a Project named **EIE3360LabB**.
b. Download the "Stealth" package from Blackboard and import the package to the Assets folder.
c. Save the SampleScene as "Stealth_Lab1B" in the Assets folder.
d. Delete the Directional Light in Stealth_Lab1b.
e. Window -> Rendering -> Lighting -> Environment tab -> Skybox Material -> None.
f. Click File -> Build Settings, click "Add Open Scenes" to add the "Stealth_Lab1B" scenes to "Scenes In Build", and make sure the index is 0.
g. Save the scene and the project.

2. Create a "Game Scene."

a. Go to the "Models" folder in the Project window and drag and drop "env_stealth_static" into the Hierarchy.
b. Ensure the position is X = Y = Z = 0.
c. Drag and drop "prop_battleBus" into the Hierarchy.
d. Set the position to X = -11, Y = 0, Z = 17.5.
e. Rotate it to Y = 270.
f. Focus on the bus.
g. Select "env_stealth_static" and add a "Mesh Collider" in the Inspector.

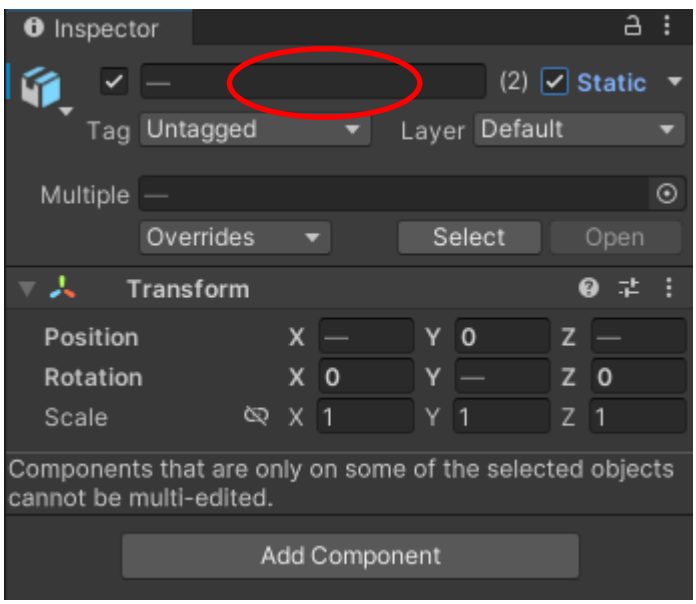
h. In the “Models\Collision Meshes” folder, expand “env_stealth_collision” and drag the “env_stealth_collision_001” to the “Mesh” properties of the “Mesh Collider”.



3. A mixture of baked dynamic light

Making the environment “static” allows game objects to be light mapped.

a. Select both “env_stealth_static” and “prop_battleBus”, and check the “Static” check box in the top-right of the Inspector.



b. Click “Yes, change children”.

c. Expand “env_stealth_static”, select every child object except “extents”, also select “prop_battleBus” and add a “Layer” and name it as “PlayArea” in the “User Layer 8”.

d. Click “Yes, change children”.

e. Select the “extents” game object in the Hierarchy and add a “Layer”, and name it “Extents”. Click “Yes, change children” again.
f. Save the scene and the project.

4. Set the camera

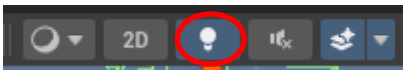
a. Select the “Main Camera” and rename it to “Camera_Main”.
b. Change the “Clear Flags” to “Solid Color” in the Inspector.
c. Change the “Rendering Path” to “Deferred”.
d. Set “Clipping Plan” Far to “100”
e. Click the color box of the “Background” and set the color to Black (R=G=B=0).

5. Add the lights

a. Go to the Prefabs folder in the Project windows, drag and drop the “lights_baked” game object to the Hierarchy and reset the position to X = Y = Z = 0.
b. Create a “Point Light” in the Hierarchy and rename it to “light_playArea_point”. Set the position to X = -3, Y= 1, Z = 10.
c. Adjust the light color to R = 255, G = 245, B = 150, A= 255 and reduce the “Range” to 8.
d. Change the “mode” from “Realtime” to “Baked”.

Note: This can avoid rendering in real time to affect performance

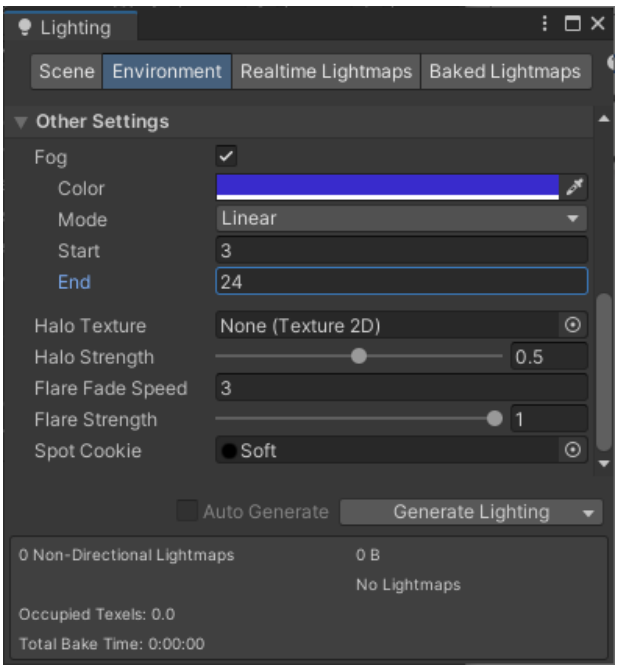
e. Turn on and off the lighting by clicking the “Light bulb” icon at the top of the Scene windows if needed.



f. Duplicate the “point” light two times and move one to the center of the room and the other one to the end of the room.
g. Drag the three “point” lights to the “lights_baked” game object to keep them organized in the Hierarchy.
h. Save the scene and the project.

6. Change the render setting

a. Click Windows -> Rendering -> Lighting Setting and go to the “Environment” tab.
b. Go to Other Settings -> check the “Fog” to enable it.
c. Set the “Fog” Color to R = 57, G = 43, B = 204, A = 255.
d. Set the Mode to “Linear”.
e. Set the Linear Fog “Start” to 3, and “End” to 24.

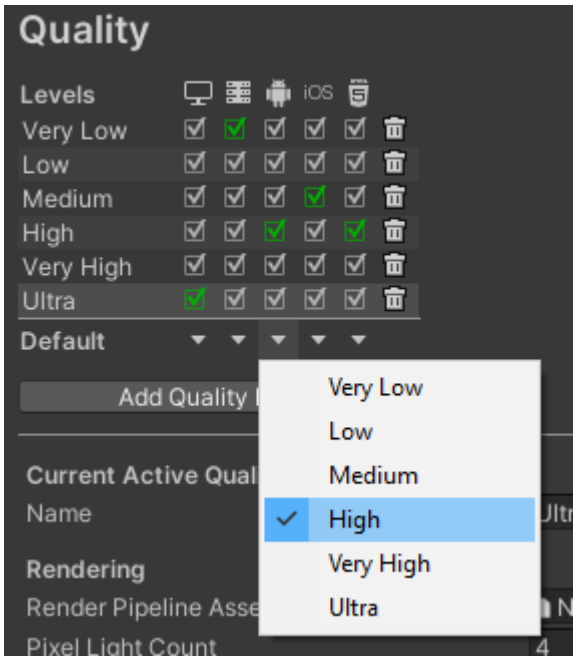


f. You may turn off the fog in the Scene to have a better view during development.

7. Change the quality setting and baked light

a. Click Edit -> Project Setting -> Quality.

b. Click on the arrow under the "Default" of the Android platform and make it "High".



8. Create the directional light and alarm light

a. Create a "Directional Light" from the Hierarchy and rename it to "light_main_directional".

b. Add a tag and name it "MainLight" in the Inspector and assign the "MainLight" tag to "light_main_directional".

c. Change the light color to R = 33, G = 45, B = 48.

d. Set the "Shadow Type" to "Soft Shadows".

e. Set the "Mode" to "Baked"

f. Click the "Culling Mask" and de-select "Extents".

g. Set the Rotation to X = 45, Y = 305, Z = 0.

h. Duplicate the "light_main_directional" and rename it to "light_alarm_directional".

i. Add a tag, name it to "AlarmLight" in the Inspector, and assign the "AlarmLight" tag to light_alarm_directional.

j. Set the color to R = 70, G = 0, B = 0.

k. Set the "intensity" to 0.

l. Click the "Culling Mask" and select 'Everything'.

m. Save the scene and the project.

3.2 Tag Management

1. The tag in Unity is for identifying references to game objects. We will apply a script to the variables for convenience.

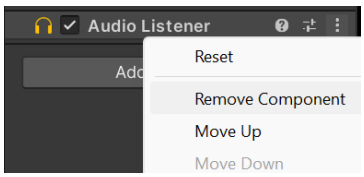
a. The following script contents are for your reference only.

```
using UnityEngine;
using System.Collections;

public class Tags : MonoBehaviour
{
    // A list of tag strings.
    public const string player = "Player";
    public const string alarmLight = "AlarmLight";
    public const string siren = "Siren";
    public const string gameController = "GameController";
    public const string mainLight = "MainLight";
    public const string enemy = "Enemy";
}
```

3.3 Player Setup

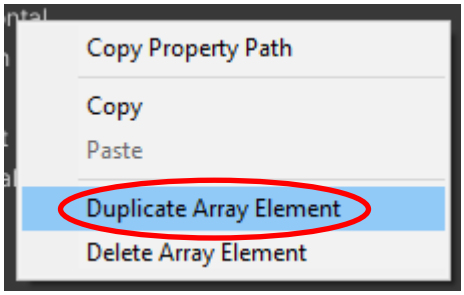
1. **Create the player for the game**
   a. In the “Models” folder, drag and drop the “char_ethan” game object into the Hierarchy.
   b. Set the position to X = -2.5, Y = 0, Z = 0.
   c. Double-click the name to focus on it.
   d. Set the Tag to Player in the Inspector.
   e. Click Add Component -> Capsule Collider.
   f. Set the “Center” to X = 0, Y = 1, Z = 0.
   g. Change the “Radius” to 0.4, and set the “Height” to 2.
   h. Click Add Component -> Rigidbody.
   i. Expand “Constraints” and check Y of “Freeze Position” and X and Z of “Freeze Rotation”.
   j. Click Add Component -> Audio Source, uncheck “Play On Awake”, and check “Loop”.
   k. Set the “Volume” to 0.1.
   l. In the “Audio” folder in the project window, drag and drop “player_footsteps” to the “Audio Clip” in the inspector.
2. **Change the camera listener**
   a. Select “Camera_Main” in the Hierarchy.
   b. Remove the “Audio Listener” component.



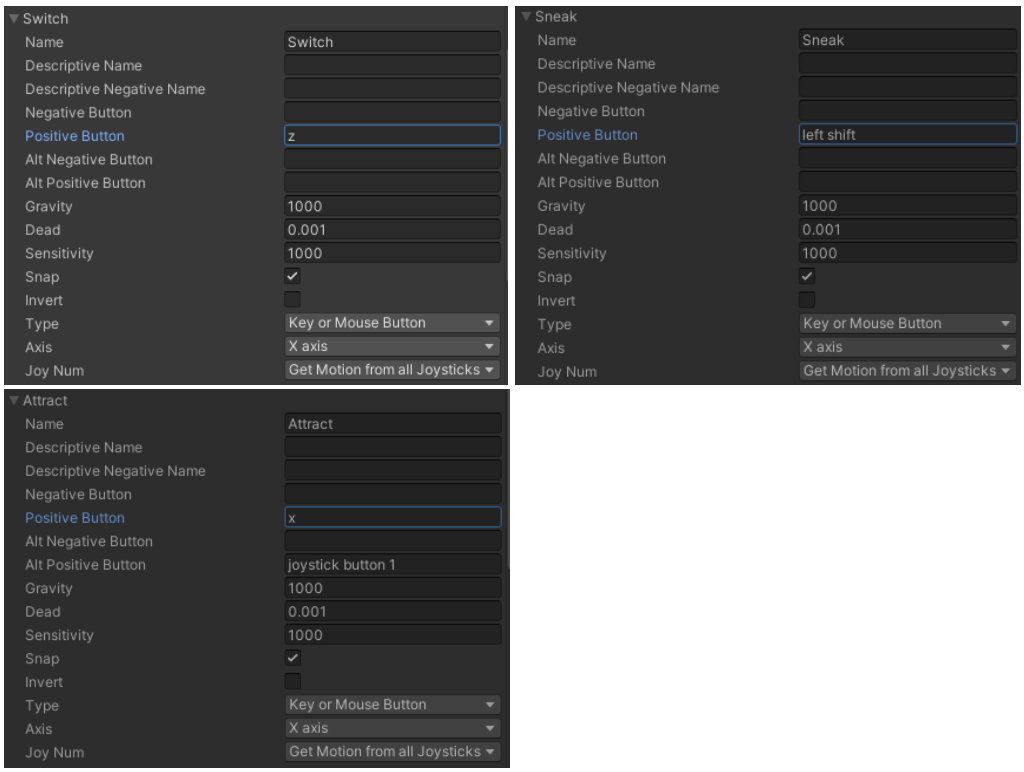
   c. Select “char_ethan”, click Add Component -> Audio Listener.
   d. File -> Save.
3. **Change the Input Manager**

   a. Edit -> Project Settings -> Input Manager -> Expand “Axes”.

   - Uncheck Snap in both Horizontal and Vertical.
   - Right-click one of the array elements and select duplicate.
   - Duplicate THREE times and name them “Switch”, “Sneak”, and “Attract”.



- Make sure the “Type” is “Key or Mouse Button”.
- Z key for “Switch” (by expanding “Switch” → Positive Button = z)
- Left Shift key for “Sneak” (by expanding “Sneak” → Positive Button = left shift)
- x key for attracting enemy (by expanding “Attract” → Positive Button = x)



1.4 Player Animator Controller

1. Make the player move to set up an animator controller

a. Select “char_ethan” and go to the “Animator” folder in the project panel. Then, drag and drop the “PlayerAnimator” to the “Controller” properties of the “Animator” in the Inspector.

b. Press the “Play” button and switch to the “Scene” panel; the animation is applied to the character.

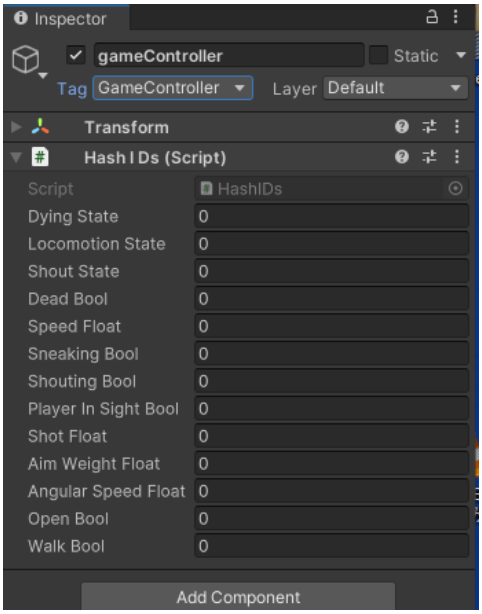
3.5 HashIDs

When using the animator, we can reference the states and parameters by a name as a string or hash stored as an integer. In this session, the script “HashIDs” stores the hashes so that we can reference the class variables.

a. Read the HashID script to understand the hash tags for various strings used in our animators

b. Create an empty game object in the hierarchy called “gameController.”

c. Tag the “GameController” in the Inspector and drag the “HashID” script from Scripts/Unapplied in the project windows to the gameController object.



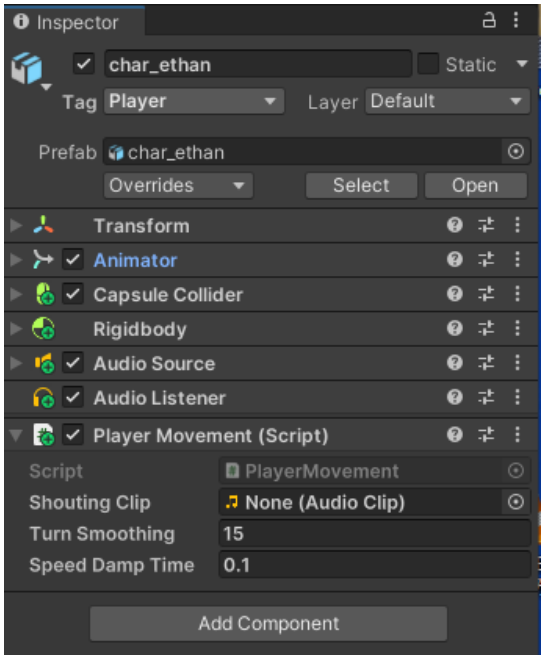

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class HashIDs : MonoBehaviour
{
    // Here, we store the hash tags for various strings used in our animators.
    public int dyingState;
    public int locomotionState;
    public int shoutState;
    public int deadBool;
    public int speedFloat;
    public int sneakingBool;
    public int shoutingBool;
    public int playerInSightBool;
    public int shotFloat;
    public int aimWeightFloat;
    public int angularSpeedFloat;
    public int openBool;


    void Awake()
    {
        dyingState = Animator.StringToHash("Base Layer.Dying");
        locomotionState = Animator.StringToHash("Base Layer.Locomotion");
        shoutState = Animator.StringToHash("Shouting.Shout");
        deadBool = Animator.StringToHash("Dead");
        speedFloat = Animator.StringToHash("Speed");
        sneakingBool = Animator.StringToHash("Sneaking");
        shoutingBool = Animator.StringToHash("Shouting");
        playerInSightBool = Animator.StringToHash("PlayerInSight");
        shotFloat = Animator.StringToHash("Shot");
        aimWeightFloat = Animator.StringToHash("AimWeight");
        angularSpeedFloat = Animator.StringToHash("AngularSpeed");
        openBool = Animator.StringToHash("Open");
    }
}
```

3.6 Player Movement

1. **Apply the "PlayerMovement" script to control the player's movement**
   a. Select the "char_ethan" object
   b. Go to Scripts\Player in the project windows, drag and drop the "PlayerMovement" script to apply to the "char_ethan" object in the hierarchy.



2. **Apply the audio to the player.**
   a. Select the player “char_ethan” object.
   b. In the “Audio” folder in the project window, drag and drop “player_attractAttention” to the “Shouting Clip” variable.
   c. Save the scene and the project.
   d. Go to the “Scene”, select “Camera_Main” and go to GameObject on the menu → “Align With View” to set the camera tracking the player.

   e. Click the “Play” button to play the game. The player can move in the Game mode.
      - **arrow** keys or **W, A, S and D** keys to control, hold the Left shift key, move the arrow keys to sneak, and press **x** to shout.

Checkpoint 1: Demo to the instructor or teaching assistant

3.7 Player Health

1. **Apply a script to handle the player's health**
   a. Select the player "char_ethan."
   b. Go to Scripts\Player in the project window and drag the "PlayerHealth" script to the "char_ethan" object.
2. **Apply the audio**
   a. Select the player "char_ethan".
   b. In the "Audio" folder in the project window and drag and drop "endgame" to the "Death Clip" variable.
   c. Save the project.

4. Exercises

Modify the game to fulfil the following requirements:

1. The camera follows the player whenever it moves.
2. Create a servant character called "Mimi" (download "Mini.fbx" from Blackboard) who follows the player.
3. When the distance between the player and the servant is greater than a certain threshold (e.g., more than 1 unit), the servant moves towards the player.

Checkpoint 2: Demo to the instructor or teaching assistant/take the video with explanations and upload it to Blackboard

Describe and explain the methods you used to achieve the results in your lab report. Any necessary scripts you have added or modified should include.

5. References

[1] Unity - Game Engine, http://unity3d.com/

[2] Unity - Stealth tutorial, https://www.youtube.com/playlist?list=PLX2vGYjWbI0QGyfO8PKY1pC8xcRb0X-nP

[3] Unity Manual, http://docs.unity3d.com/Manual/index.html

[4] Unity - Script API, http://docs.unity3d.com/ScriptReference/index.html