CS 3160 – Game Programming

Project 5 – Mobile Controls and Profiling

Learning Objectives

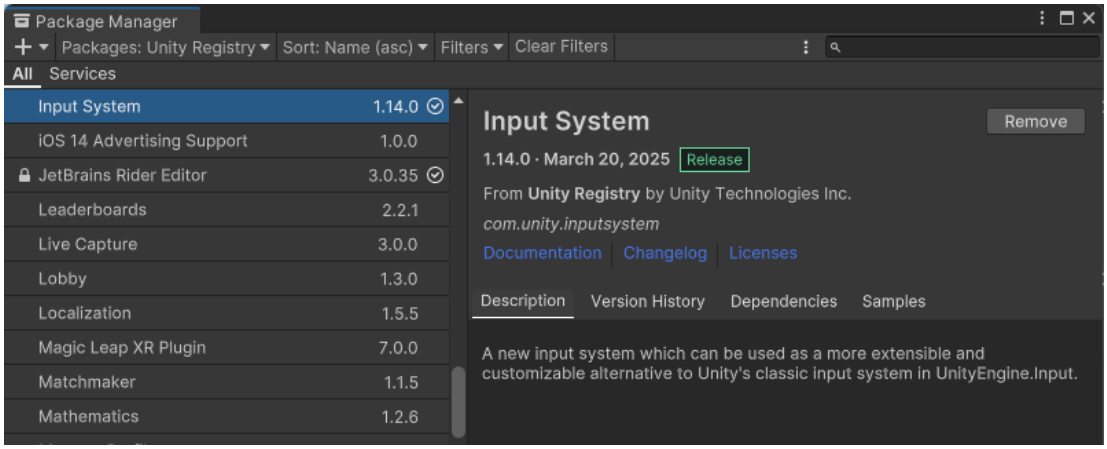
- Learn how to use the Unity profiler for optimizing code
- Learn how to implement mouse and keyboard, gamepad, and touch controls in Unity

Overview

In this project you will become familiar with using the Unity profiler to find optimization issues and implementing touch controls.

Sample Code

To complete this project some sample code is required. This is provided in the form of a Unity Package. Download the “SampleCode-Project5” Unity Package from Pilot. First, create an empty Unity project using the “3D (Built-In Render Pipeline)” template. Then, go to “Assets” > “Import Package” > “Custom Package…” then locate “SampleCode-Project5.unitypackage” downloaded from Pilot and import all files. After this, you must install the Input package provided by Unity here:



If importing was successful you should be able to load the “Project 5 - Level Hard” Scene in the “Scenes” folder and you can test the game.

NOTE: Importing a Unity Package does not import Layers. The layers used are listed above.

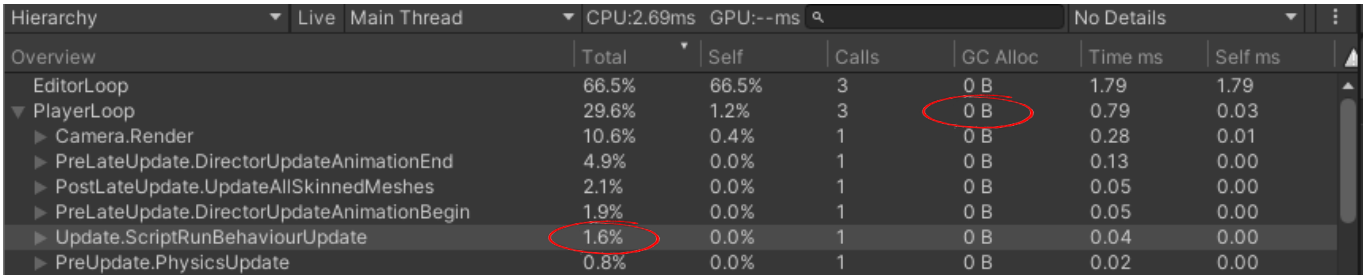
Performance Optimizations

Some optimization bugs have been introduced into a game by a new “intern” and you are tasked with finding them and solving them. This “intern” said they made some changes to the code from Project 4 and now the game runs very poorly when playing on the “Project 5 - Level Hard” scene and after killing the zombie in the “Project 5 - Level Easy” scene. Also, they say there are way too many garbage collection memory allocations.

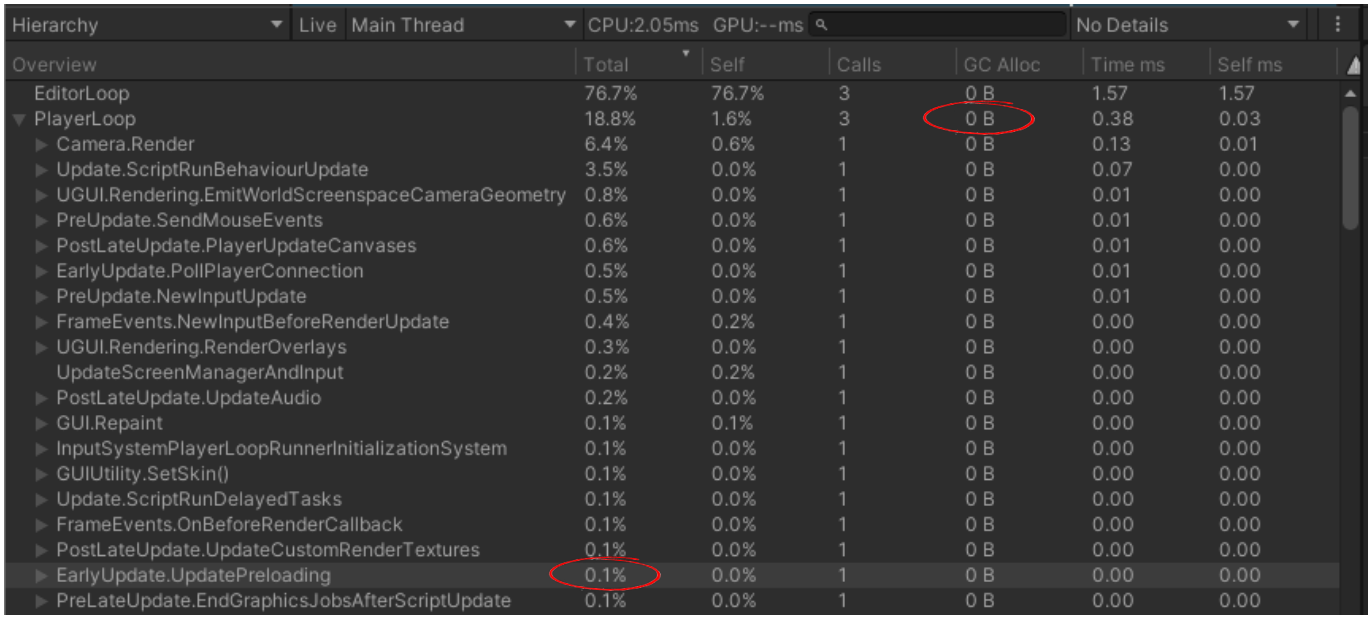
The “intern” admits they had help from AI and they actually have no idea why it’s so broken and poorly optimized but they left plenty of comments to help you clean up their mess. To fix these bugs you must use the Unity profiler to find why it's running so poorly. The Unity profiler can be found by going to:
Window -> Analysis -> Profiler

Once you are inside the profiler you must identify what portions of the game are causing the performance and memory allocation issues. Once identified, you must fix the issues. You must maintain the original intent of “intern”. Read the comments left by the “intern” to know how to fix the performance and memory issues while maintaining the desired functionality of “intern’s” code. Do not bring in any code you wrote for “Project 4”. The AI and UI implemented in Project 4 must not be implemented in this project.

The “Project 5 - Level Hard” code is fixed when you see 0 bytes of total “GC Alloc” for the “Player Loop” method and <5% “Total” time for the “Update.ScriptRunBehaviorUpdate” method inside of “PlayerLoop” when playing the “Project 5 - Level Hard” scene:



The “Project 5 - Level Easy” code is fixed when you see 0 bytes of total GC Alloc and <5% “Total” time for the “EarlyUpdate.UpdatePreloading” method inside of “PlayerLoop” after killing the zombie in the “Project 5 - Level Easy” scene and loading the “Project 5 - Win Screen”:



Touch Controls
After the optimization issues are fixed, you will have to implement controls using the new input system. 5 different controls are required:

- Movement joystick - controls moving the player
- Look joystick - controls rotating the player
- Jump button - makes the player jump when tapped
- Sprint button - makes the player sprint only when held down
- Attack button - makes the player attack when tapped (after the player has touched the red sphere)

To implement these touch controls you must first create a new Input Action Asset named "PlayerInputActions". Then, create a new Action Map and create 5 new actions:

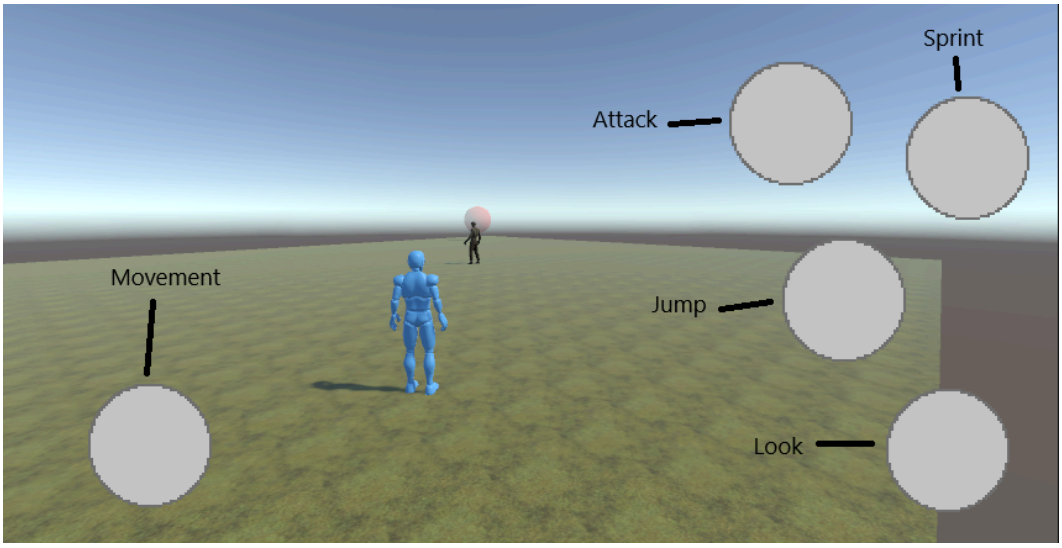
- Movement
    - Action type: Value
    - Control type: Vector 2
    - Controlled by: left stick (gamepad) and WASD (keyboard)
- Look
    - Action type: Value
    - Control type: Vector 2
    - Controlled by: right stick (gamepad) and delta (mouse)
- Jump
    - Action type: Button
    - Controlled by: button south (gamepad) and space (keyboard)
    - Interactions: tap interaction
- Sprint
    - Action type: Button
    - Controlled by: button east (gamepad) and shift (keyboard)
    - Interactions: hold interaction
- Attack
    - Action type: Button
    - Controlled by: right trigger (gamepad) and left button (mouse)
    - Interactions: tap interaction

You must modify Player.cs to remove the old input system controls and add the new input system controls: Movement, Jump, Sprint, and Attack. You must modify CameraFollow.cs to add the new input system controls: Look. For the button controls (Jump, Sprint, and Attack), you can use the methods WasPerformedThisFrame() and/or IsPressed() instead of ReadValue<>(). Also, use "OnDestory()" to disable the PlayerInputActions that was enabled in your scripts to prevent leaks and performance issues.

Once this is completed, add 2 UI joysticks and 3 UI buttons to a Canvas. The two joysticks should control the left and right stick, the 3 buttons should control the button south, button east, and right trigger. For the buttons and joysticks you can use the "UICircle" image included in the "Materials" folder. For the buttons, use "On-Screen Button" and for the joysticks use "On-Screen Stick" scripts.

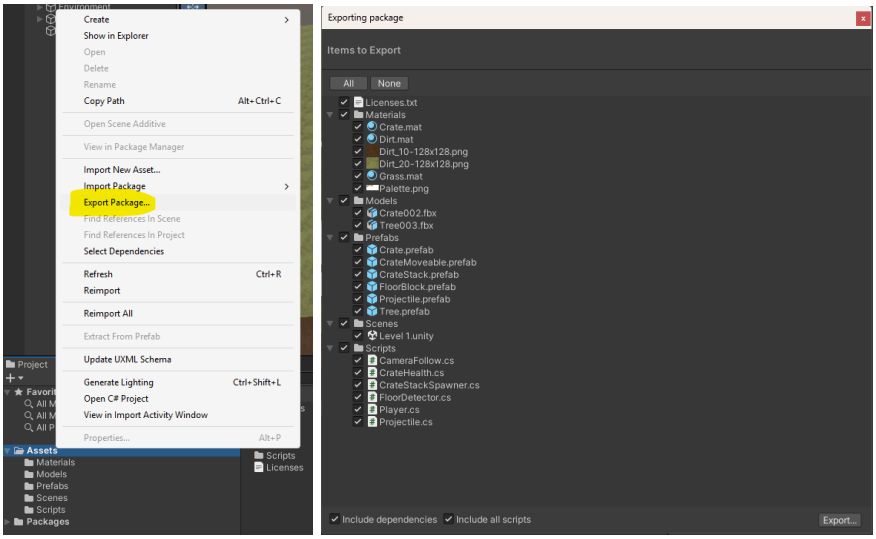
If implemented correctly, your game should allow for controlling using keyboard and mouse, gamepad, and touch controls. You do not have to test the gamepad controls with an actual gamepad, only keyboard and mouse and touch controls are required to work. To test the touch controls just right click and drag on the UI elements in the editor. You are not required to build and run on platforms with touch controls i.e. Android or iOS.

Here is an example layout of the UI sticks and buttons:



How to Submit

To submit your code, you must pack your project into a Unity Package. To create a Unity Package first right click on the “Assets” folder inside of the Unity Editor, select all items and hit Export. Lastly, name your file “YourLastName-Project5” where “YourLastName” is your actual last name, and submit this file to the Pilot dropbox:



Grading

This lab is worth 5.00 points, distributed as follows:

Task	Points
There are no build or runtime errors	1.00
GC Alloc is 0 B for “Hard” scene and after winning “Easy” scene	1.00
Total time is <5% for “Hard” scene and after winning “Easy” scene	1.00
Look and movement joysticks implemented	1.00
Jump, sprint, and attack buttons implemented	1.00
Total	5.00