COMP280 – Optimizing my Arduino Project

Thomas O’Leary 1903716

November 19, 2020

1 Introduction

For my end of year project in Level 4 (First Year), we were tasked to create a custom controller powered by an Arduino and a game for this controller to be used in.

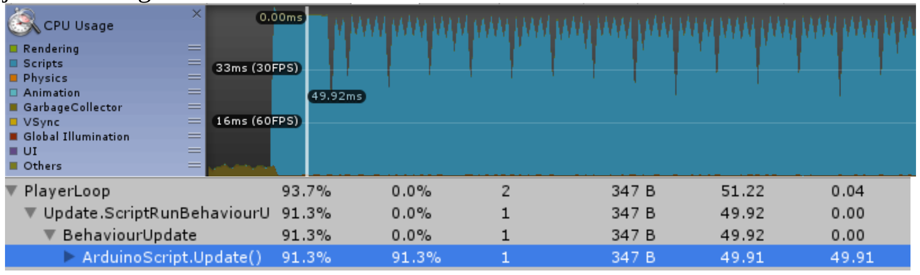
My idea for the game and controller was to create a Maze game where the Player had to navigate a ball through said maze by rotating/moving the controller around (to roll the ball through the maze). As well as this, when the ball collided with a wall, the player would receive haptic feedback from the direction in which it collided.

There were many complications that came along with completing this project, most notably the start of the pandemic. So when implementing the functionality into Unity, I could only get the Accelerometer to work properly.

https://github.com/thomasoleary/Comp140-Maze

2 Profiling

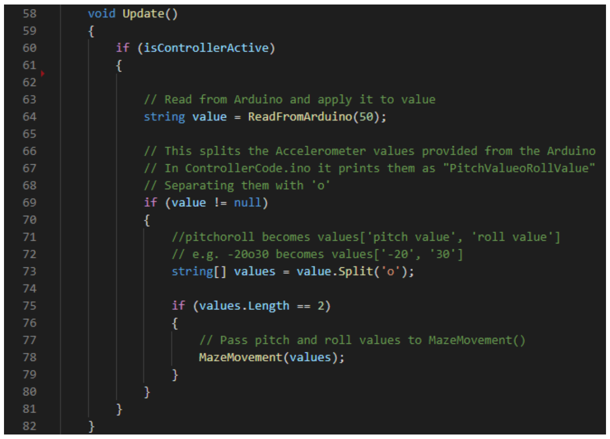
Using the default Unity Profiler, I was able to create a baseline check for my project – see Figure 1.



▼ PlayerLoop	93.7%	0.0%	2	347 B	51.22	0.04
▼ Update.ScriptRunBehaviourU	91.3%	0.0%	1	347 B	49.92	0.00
▼ BehaviourUpdate	91.3%	0.0%	1	347 B	49.92	0.00
▶ ArduinoScript.Update()	91.3%	91.3%	1	347 B	49.91	49.91

Figure 1, First Profiler results

Now from this first check, it is obvious to see that Update() within ArduinoScript is severely affecting this projects performance. It also seems that there is a pattern of 4/5 large spikes, then performance slightly increases – just for it to continue the spike pattern after. On average, the scene was running at 23 frames per second.



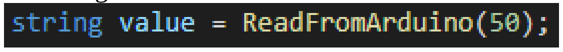
```
58      void Update()
59      {
60          if (isControllerActive)
61          {
62
63              // Read from Arduino and apply it to value
64              string value = ReadFromArduino(50);
65
66              // This splits the Accelerometer values provided from the Arduino
67              // In ControllerCode.ino it prints them as "PitchValueoRollValue"
68              // Separating them with 'o'
69              if (value != null)
70              {
71                  //pitchoroll becomes values['pitch value', 'roll value']
72                  // e.g. -20o30 becomes values['-20', '30']
73                  string[] values = value.Split('o');
74
75                  if (values.Length == 2)
76                  {
77                      // Pass pitch and roll values to MazeMovement()
78                      MazeMovement(values);
79                  }
80              }
81          }
82      }
```

Figure 2, Update() in ArduinoScript.cs

With a first glance to Update() – see Figure 2, its reading values that are being outputted by the Arduino to a specified serial port – then passing these values into the function MazeMovement().

3 Improvements

Using the data from my first profile, I can begin my attempts to improve the performance of this function. In this first iteration, I decided to focus on line 64 in ArduinoScript.cs – see Figure 3.

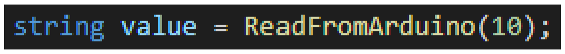


```
string value = ReadFromArduino(50);
```

Figure 3, Line 64 in ArduinoScript.cs

ReadFromArduino() is a simple function that runs a Try Catch exception that, you guessed it, reads the values from the Arduino. The *50* integer parameter is the timeout value (in the first case it had a 50ms timeout).

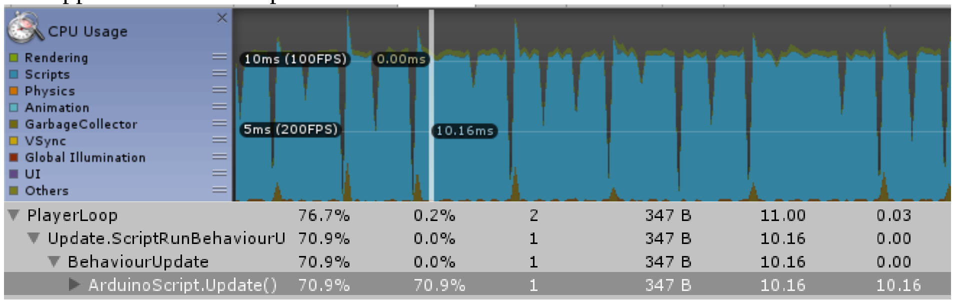
To check to see if this had a severe impact on Update(), I changed the value from *50* to *10* – see Figure 4.



```
string value = ReadFromArduino(10);
```

Figure 4, Changing the parameter value

Much to my surprise, this vastly increased the performance in the scene – see Figure 5. Changing the average FPS from ~22fps to ~93fps. And although there are still some spikes, which come in a predictable pattern, they are not as rapid as the ones that appeared in the first profile.



```
▼ PlayerLoop                   76.7%    0.2%    2    347 B    11.00    0.03
  ▼ Update.ScriptRunBehaviourU 70.9%    0.0%    1    347 B    10.16    0.00
    ▼ BehaviourUpdate          70.9%    0.0%    1    347 B    10.16    0.00
      ▶ ArduinoScript.Update() 70.9%    70.9%   1    347 B    10.16    10.16
```

Figure 5, Profiler results from the ReadFromArduino() parameter change

4 Conclusion

References

[1]

[2] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[3] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.