

# *Jumping Up A Level*

## *GPG 214.1*

### *Optimisation Reports*

#### **Team Name**

James Kilpatrick

---

## [Optimisation Testing Report](#)

[Introduction](#)

[Testing Methodology](#)

[Test Cases](#)

[Profiling Results](#)

[Optimisation Strategies](#)

[Performance Improvement](#)

[Challenges and Lessons Learned](#)

[Conclusion](#)

# Optimisation Testing Report

## Introduction

The purpose of this testing report is to analyze the amount of calls being sent during a normal gameplay testing session. This is for the client to show them the progress of the work being done and that key tasks are being met.

## Testing Methodology

The code is being tested on Unity version 2022.3.26f1 on a windows 10 PC. The testing procedures involved will be a regular play through test of the mechanics. Then there will be a test of each mechanic being tested multiple times such as the saving and loading and jumping mechanics

## Test Cases

Player Data Saving and Testing Case:

The testing will cover the player data which involves the players name and color being saved as well as how many times the player has jumped or died in this session.

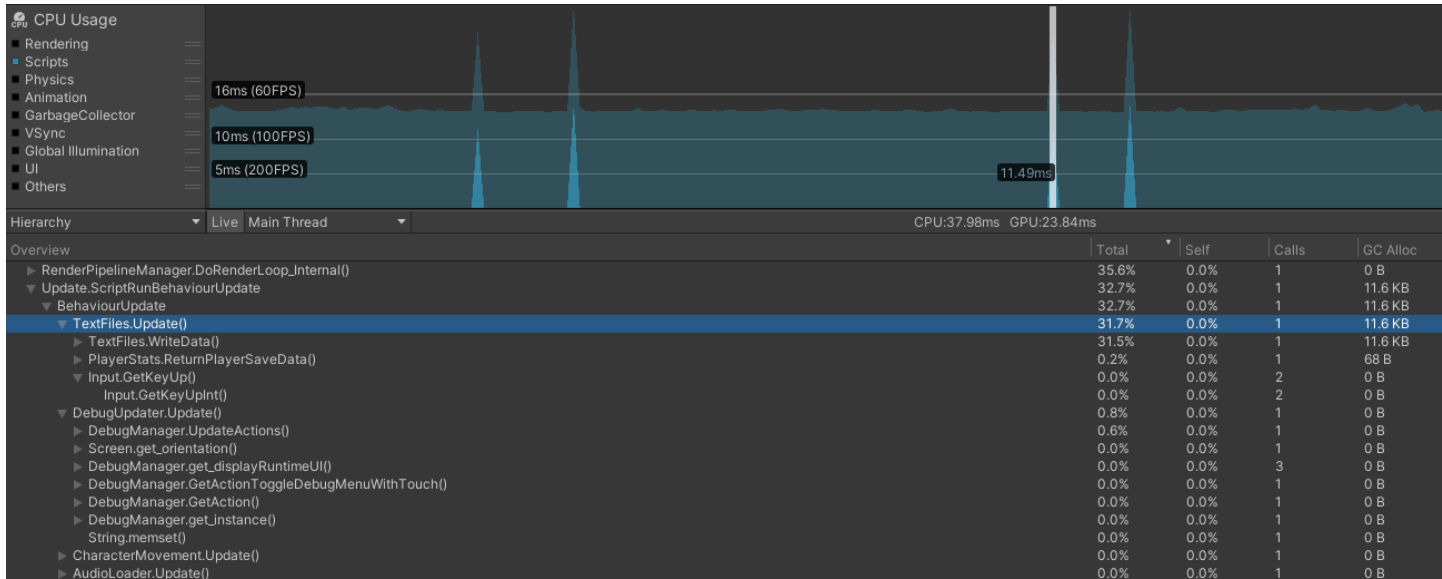
Jumping/Moving Mechanic Testing Case:

The testing will cover the player movement to see how many calls are being made for specific actions such as jumping or moving.

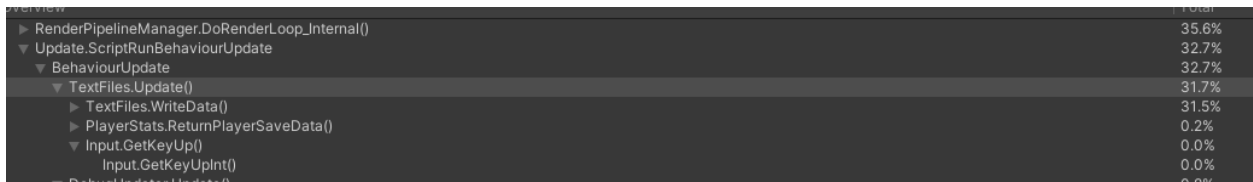
# Profiling Results

## Player Data Saving and Loading Test Results:

Unity profile data:

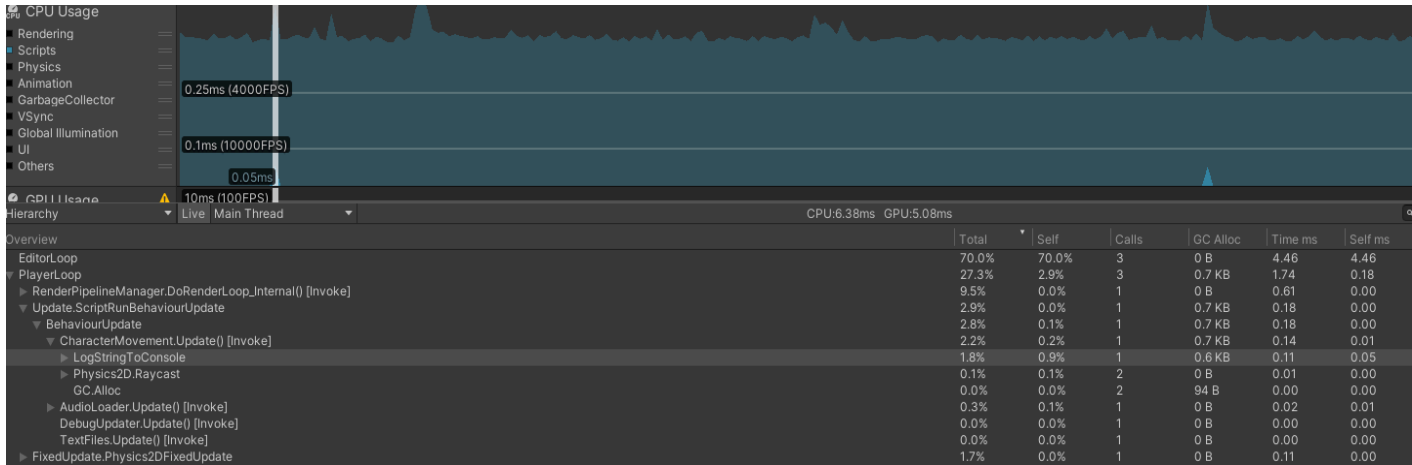


Each of the spikes here are when the data is being saved and then loaded. The first spikes are for the saving seen below:



## Player Jumping/Moving Mechanic Testing Results:

Unity Profiler:



The two small spikes 1 being covered by the white line are when the player is jumping. The one being covered in the white line shows these results.

▼ PlayerLoop	27.3%	2.9%	3
▶ RenderPipelineManager.DoRenderLoop_Internal() [Invoke]	9.5%	0.0%	1
▼ Update.ScriptRunBehaviourUpdate	2.9%	0.0%	1
▼ BehaviourUpdate	2.8%	0.1%	1
▼ CharacterMovement.Update() [Invoke]	2.2%	0.2%	1
▶ LogStringToConsole	1.8%	0.9%	1
▶ Physics2D.Raycast	0.1%	0.1%	2
GC.Alloc	0.0%	0.0%	2
▶ AudioListener.Update() [Invoke]	0.3%	0.1%	1
DebugUpdater.Update() [Invoke]	0.0%	0.0%	1
TextFiles.Update() [Invoke]	0.0%	0.0%	1

## Optimisation Strategies

- Description of the optimisation techniques applied to address identified bottlenecks.
- Explanation of the rationale behind each optimisation strategy.
- Documentation of the implementation details and changes made to the code, assets, or settings.

For the saving and loading to improve on the bottlenecks appearing we can reduce the amount of debug logs that are being created. Currently we are debugging out the whole text file document as well as the saved elements such as the name and color. Doing this would reduce the amount of calls needed saving on resources

For the Jumping bottleneck identified we can also reduce the debug logs being created as it currently debugs out that the code is working and the character has jumped. We do not need this code as we can see the character jumping on the screen.

## Performance Improvement

### Removed Debug Log in Jumping Code



After removing the debug log in the script there were noticeably less spikes in the unity profiler. This is most likely due to the less calls being made in the update due to it only needing to move the character and not also updating and creating a debug log

## Challenges and Lessons Learned

As this was my first time using the unity profiler I believe I could be more efficient with it in the future. It took me a while to understand what I was looking at and how stuff was affecting the resources used. For future uses I would like to work on bigger projects where I am able to more easily understand the areas I can improve as currently this project is very resource light and it was hard finding any areas I could improve on with my lack of knowledge.

## Conclusion

Overall I was able to improve the optimisation via removing an unneeded debug log and was able to learn and understand what needed to be improved upon in future versions. I believe I could have done better optimisation with a better understanding but am still proud of how I managed to understand and improve something for my first time.