University of South Wales

MSc Game Development

Academic Year: 2023-2024

CS4S765: Game Engine Optimisation

Practical Coursework – 1

<STUDENT ID>

Optimising Smugglers’ Bay

[1. Introduction 2](#_Toc1944041953)

[2. Initial Profile 4](#_Toc1555566926)

[3. Optimisation 5](#_Toc1836974478)

[3.1. Time-slicing 6](#_Toc1555606029)

[3.1.1. Theory/Background 6](#_Toc340505887)

[3.1.2. Implementation 6](#_Toc1270583068)

[3.1.3. Performance Analysis 6](#_Toc353917607)

[3.2. Discrete Level of Detail 6](#_Toc895084264)

[3.2.1. Theory/Background 7](#_Toc339435714)

[3.2.2. Implementation 7](#_Toc1047054980)

[3.2.3. Performance Analysis 7](#_Toc392225375)

[3.3. Object Pooling 7](#_Toc1209622798)

[3.3.1. Theory/Background 7](#_Toc1112818274)

[3.3.2. Implementation 7](#_Toc1884939274)

[3.3.3. Performance Analysis 7](#_Toc631210881)

[3.4. Sum-of-sines Approximation for Water Physics 7](#_Toc576397667)

[3.4.1. Theory/Background 7](#_Toc577844793)

[3.4.2. Implementation 7](#_Toc376485208)

[3.4.3. Performance Analysis 7](#_Toc1841548915)

[3.5. Audio Optimisations 7](#_Toc601198425)

[3.5.1. Theory/Background 7](#_Toc150182226)

[3.5.2. Implementation 7](#_Toc1445835064)

[3.5.3. Performance Analysis 7](#_Toc806925033)

[4. Conclusion 8](#_Toc1378939388)

[5. References 9](#_Toc1887522702)

# Introduction

This report details the theory behind, process, and result of implementing optimisation techniques inside the video game Smugglers’ Bay. Set at night-time during the medieval era, Smugglers’ Bay is a stealth game inspired by Thief: The Dark Project (1998) and involves looting from a castle surrounded by a moat, called Smugglers’ Bay, with the aim of breaking in and out unseen.

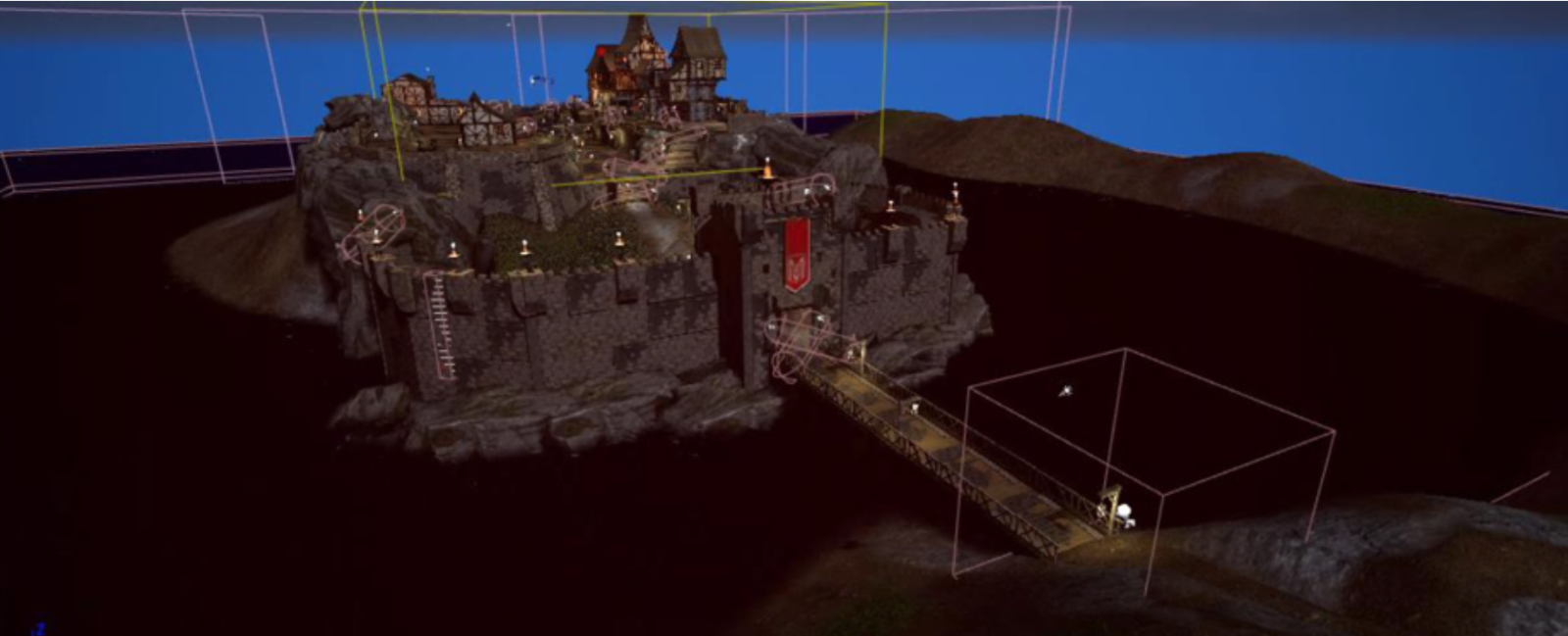
Developed using Unreal Engine 4, the gameplay includes:

* First and third-person perspectives, with the camera switching from third-person when outdoors to first-person when indoors to exacerbate the difficulties of traversing a space that cannot be scouted before entry.
* A lockpicking element, where the player must find the correct angle to move a pin to unlock doors to enter new areas.
* An occlusion-based line-of-sight cone system, like used in Metal Gear Solid (1998), implemented into rotating, flame-throwing gargoyle enemies that detect the player.
* A swimmable, water-filled moat implemented using Gerstner Waves simulation.
* A focus on audio, with many sound effects implemented to make audio, at times, a more important sense than visuals in order to make sneaking tense.

In this report, an initial profile of the game and an assessment is given to highlight potential areas for optimisation. The main section of the report is split into five sections covering optimising using the following techniques:

* Time-slicing
* Discrete Level of Detail
* Object Pooling
* Sum-of-sines Approximation for Water Physics
* Multiple Audio Optimisations

Finally, the report ends with a conclusion that gives a summary of the outcome of optimisation as well as general findings during the implementation process.



*Figure 1 Image of the Smugglers Bay During Development in UE4*

(Unity Technologies, 2022)

# Initial Profile

Pull data from profiler into profile analyzer to see average FPS.

Assess areas of significant frame drops.

Assess audio if optimising audio later.

Assess memory profiler – what's using a lot of memory?

Is there overdraw?

Is there significant garbage allocation?

...

There may be more

...

Remember to profile the build when it’s desirable to isolate from the editor.

Think about the marking scheme. This section is only really good for demonstrating your ability to use profiling tools.

<demo landscape image>

# **Optimisation**

## **Time-slicing**

### Theory/Background

### Implementation

### Performance Analysis

## **Discrete Level of Detail**

### **Theory/Background**

### **Implementation**

### Performance Analysis

## **Object Pooling**

### **Theory/Background**

### **Implementation**

### Performance Analysis

## **Sum-of-sines Approximation for Water Physics**

### **Theory/Background**

### **Implementation**

### Performance Analysis

## Audio Optimisations

### **Theory/Background**

Unity uses OGG Vorbis. It works by... <demo referencing>...

Streaming...

Nyquist-shannon Theroem...

### **Implementation**

...

...

...

### Performance Analysis

...

...

...

# **Conclusion**

What improvements were made with each optimisation technique? Consider adding some kind of graph here to visualise your results.

Which was least and which was most effective? Why do you think that is?

What was most challenging?

Compare your initial profile with your final profile. Whatever you assessed in your initial profile, you probably want to assess it again here:

* Frames per second (FPS)
* How much RAM used now?
* How many draw calls now?
* ...

Why did you choose to implement these techniques specifically?

Why did you decide against optimising other techniques?

Is there anything you would do different next time? (e.g there is a new paper for fluid simulation titled \_\_\_\_ that might be able to give greater results and is worth investigating but given the optimisation technique implemented for Smugglers’ Bay has been successfully used in games x, y, z (reference), it is more likely a technique that would also be successful in this context. But may be worth testing in the future.

# References

Unity Technologies. (2022). *Ultimate Guide to Profiling Unity Games.*