Bachelor's thesis

Information and Communications Technology

2024

Tuomas Vuorinen

Animated sequence rendering performance comparison: Unity and Unreal Engine

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2024 | 64 pages

Tuomas Vuorinen

Animated Sequence rendering performance comparison: Unity and Unreal Engine

This thesis compares two high-profile game engines, Unity and Unreal Engine, in terms of the impact their 3D rendering capabilities have on computer system performance and describes the procedures required to achieve fair testing. The aim was to establish which application is more suitable for efficiently recording and rendering an animated sequence in a pre-built environment, as well as to explore the differences in the overall workflow.

This aim was achieved through practical experimentation by rendering 18 individual 24-second animated sequences in real time using both Unity and Unreal Engine on three different computers, each running different hardware combinations. An equal starting point was ensured by recording all sequences in NVIDIA’s USD Attic sample scene.

Unreal Engine produced animated sequences of significantly higher quality than Unity, though this superiority came at the cost of higher computer resource usage and slower rendering times. Despite these shortcomings, the conclusion is that Unreal Engine is the winner, especially from a technological standpoint: its more sophisticated lighting features and its ability to support more detailed assets through add-ons makes it more suitable for this kind of rendering work. Still, Unity’s more modest system requirements make it a viable alternative, and if sufficient time is invested in refining the editor’s visual effects, its results can be brought on par with those of Unreal Engine.

Keywords:

3D rendering, animation, animated sequence, Unity, Unreal Engine

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2024 | 64 sivua

Tuomas Vuorinen

Animoidun kohtauksen kuvantamissuorituskyvyn vertailu Unitylla ja Unreal Enginellä

Opinnäytetyössä verrataan kahden suositun pelimoottorin, Unityn ja Unreal Enginen, 3D-kuvantamisominaisuuksien vaikutusta tietokoneiden suorituskykyyn sekä kuvataan tasapuolisessa testauksessa vaadittavia menettelyjä. Tarkoituksena oli selvittää, kumpi sovellus sopii paremmin animoidun kohtauksen tehokkaaseen tallennukseen ja kuvantamiseen valmiissa ympäristössä, sekä tutkia yleisen työnkulun eroja.

Tavoite saavutettiin käytännön testein kuvantamalla 18 yksittäistä animoitua 24 sekunnin kohtausta reaaliajassa sekä Unitylla että Unreal Enginellä. Kuvantamisessa käytettiin kolmea eri tietokonetta, joissa jokaisessa oli eri komponentit. Kaikissa kuvantamis- ja tallennusprosesseissa käytettiin tasa-arvoisena perustana NVIDIA:n USD Attic -mallia.

Unreal Enginen animaatioiden laatu oli selkeästi parempi kuin Unityn, vaikkakin tämä laadullinen ero vaati enemmän tietokoneiden resurssien käyttöä ja hidasti kuvantamista. Unreal Engine todettiinkin vertailun voittajaksi etenkin teknologiselta kannalta: sen kehittyneemmät valaistusominaisuudet ja -asetukset sekä kyky käsitellä yksityiskohtaisempia malleja editorin lisäosien kautta tekevät siitä paremman pelimoottorin tässä työssä tehdyn kuvantamisen kaltaiseen työhön. Unity on kuitenkin pienempien järjestelmävaatimustensa ansiosta varteenotettava vaihtoehto, ja jos enemmän aikaa käytetään valaistuksen parantamiseen, sillä saadut tulokset voidaan tuoda Unreal Enginen tasolle.

Asiasanat:

3D-kuvantaminen, animaatio, animoitu kohtaus, Unity, Unreal Engine

Contents

Figures

Tables

List of abbreviations

API	Application Programming Interface
AR	Augmented Reality
CAGR	Compound Annual Growth Rate
CPU	Central Processing Unit
FPS	Frames Per Second
GPU	Graphics Processing Unit
HDRP	High-Definition Render Pipeline
LTS	Long Term Support
RAM	Random-Access Memory
UI	User Interface
URP	Universal Render Pipeline
USD	Universal Scene Description
VFX	Visual Effects
VR	Virtual Reality

Glossary

Add-on	A third-party program or script that gives additional features to a program (Computer Hope, 2024.)
Child object	A digital object that is hierarchically dependent on another (see Parent)
Frame rate	The speed at which images (“frames”) are shown on a screen
GameObject	Fundamental objects in Unity that represent characters, props and scenery (Unity Technologies, 2017a.)
Graphics card	A hardware component responsible for rendering and displaying images, videos and animations on a computer monitor (Lenovo US, n.d.-b)
Inbetweening	Generating images between frames to make an animation flow more smoothly (Adobe, n.d.)
Plugin	A software add-on installed on a program to enhance its capabilities (Computer Hope, 2021.)
Parent object	A digital object that child objects inherit data from
Post-processing	The process of applying full-screen filters and effects onto a rendered image before it is displayed on a screen (Unity documentation, 2017b.)
Render Pipeline	A series of calculations that displays the contents of a scene on a screen (Unity documentation, 2017.)
Shader	A short program run by graphics hardware that renders graphics data

1 Introduction

The ongoing rapid growth of the global video game industry is projected to lead to a market size of $583.69 billion by 2030 (Grand View Research, 2023). The corresponding Compound Annual Growth Rate (CAGR) between 2022 and 2030 is estimated at 12.81%. This growth is helped by advancements in technology, such as the adoption of Augmented Reality (AR) and Virtual Reality (VR) in an increasing number of released products across the industry. A significant contributor to the recent development in the video game field has been the COVID-19 pandemic, which prompted an influx of new players as people began to seek new forms of entertainment during their time indoors.

As new games become more and more complex, so do their system requirements. More capable computers are thus needed to run increasingly impressive games for entertainment, educational and other applications. This contributes to a continuous cycle of innovation to cater for the expanding supply and demand. Such innovation is most often centred on new hardware developed by companies such as NVIDIA, Intel and AMD, but advancements in game development software have also played an important part in bringing new technologies to the mass market. In particular, Unity and Unreal Engine have risen to prominence as freely available game engines and development platforms.

In recent years, these two game engines have been used in the creation of an ever-increasing array of applications. According to a blog published by Cubix in January 2022, Unity Technologies claims that its cross-platform development framework forms the basis for over 50% of all mobile games currently on the market. At the same time, the approach chosen by Epic Games, the developer of Unreal Engine has led to its game engine being employed beyond its initial scope as a game development tool: notable examples of such use include the TV series *The Mandalorian* and *Westworld*. Similarly, Unity was first used in filmmaking with the 2016 release of *Adam*, a short film created by the company's demo team and rendered entirely in real time using the engine, as well as its two sequels. In 2017,

Unity introduced its Cinemachine tool which was used in the creation of the animated sequence analysed in this thesis.

In 2009, Unity became the first game engine company to offer a free licence to developers. This allowed emergent indie developers and hobbyists to use it to build applications for Apple's App Store and Google's Play Store. Subsequent monetisation schemes implemented by Unity Technologies in the wake of this widespread success resulted in the company's game engine capturing approximately 35 billion (25%) of the $137.9 billion game industry by 2018. (Young 2021; Wijman 2018.)

The ongoing Unity vs Unreal Engine user discussion covers every comparable technology offered by the two game engines. Due to the importance of high-fidelity graphics in modern video games and constant competition within the industry, the rendering characteristics of the two engines have become one of the most common topics in the debate. This thesis will focus on comparing the system performance impact of Unity and Unreal Engine when recording and rendering a pre-defined animated sequence in a static environment with lighting. In order to avoid an outcome influenced by the performance capabilities of an individual computer system, the testing was conducted on three different computers and in-application rendering setting pre-sets. Additionally, a pre-built Universal Scene Description (USD) asset was externally sourced to act as the space in which the animated sequences took place, ensuring that the testing environments in the game engines constituted a level playing field with as few deviations from one another as possible.

2 Literature review

2.1 Game engines

Šmíd (2017) describes game engines as complex, multipurpose tools for the creation of games and multimedia content. They are most commonly divided into several individual components, each of which provides unique sets of functionalities to the software. Alongside the game content itself, the engines also usually incorporate rendering, audio, physics, animation, scripting and artificial intelligence (AI) components. The render engine is the most important part of the ensemble as, apart from the gameplay, a game's graphics are typically under particular scrutiny by players and critics alike. From a technical standpoint, render engines thus need to be highly optimised as well as adaptable in order to provide efficient functionality both in anticipation of future requirements and to ensure that lower-end machines are still capable of running games developed with this technology.

2.2 Unity vs Unreal Engine

Extensive comparisons between nearly all aspects of Unity and Unreal Engine have been made ever since the former was first released in June 2005. A comparative analysis conducted by Christopoulou and Xinogalos (2017), which included Unity and Unreal Engine among other popular game engines, found that the capabilities of the two applications show considerable overlap. The study compared the engines with respect to their audiovisual and functional fidelity, composability, accessibility, developer toolkits, networking models, development features and deployment platforms. For the purposes of this thesis, more focus will be put on the visual features of both Unity and Unreal Engine as they influence the most relevant analysed metrics.

Research made in preparation for this thesis showed, however, that hardly any in-depth studies have been conducted to compare game engines such as Unity

and Unreal Engine in terms of their impact on specific system resources while rendering a predefined animated sequence.

A 2023 article written by Rahel Demant for XR Bootcamp lays out commonly considered similarities and differences between Unity and Unreal Engine. Reflecting the engines' status as current market leaders, special attention is given to scripting, rendering and visual effects (VFX) interfaces, target platforms, usage requirements, and developer support. Multiple examples of popular games of various genres developed with both engines are also provided. In the section detailing the applications' rendering systems, Unity is presented as being more modular than Unreal Engine thanks to its incorporation of both Shader Graph, a visual node-based editor, and the option to write shader code directly. Post-processing effect pre-sets are also readily available, reducing the time and effort needed to achieve the desired extent of VFX details in the final product. A limiting factor in Unity's rendering process is the need to choose a render pipeline with which visual effects are processed: the Universal Render Pipeline (URP) is more suited for cross-platform compatibility as it balances performance and quality, and the High-Definition Render Pipeline (HDRP) targets high-end hardware capable of handling higher-fidelity visuals (Sacco, 2023). Unreal Engine does not natively support writing shaders directly; instead, it employs a node-based material editor to provide a vast range of customisation with fewer technical hurdles. Post-processing in Unreal Engine offers a greater choice of options for each effect than provided by Unity without add-ons.

The general consensus in the industry appears to be that, in general terms, neither game engine is inherently better than the other. A 2024 blog post by Davoxel outlines that developers should consider the specific needs of their project when choosing between Unity and Unreal Engine, including targeted platforms, graphical requirements, and the team's programming expertise. For projects prioritising cross-platform compatibility and development speed, Unity is often the preferred choice. It is highly efficient for small to medium-scale projects where performance constraints are particularly significant; its ability to create and support lightweight applications ensures smooth performance across a wide

range of devices. In contrast, Unreal Engine is more suited for projects in which graphical fidelity is a priority, such as AAA titles, high-end simulations, and architectural visualisations. Its advanced rendering capabilities and ability to handle complex simulations mean that it is optimised for more demanding PC and console games, where delivering high-quality visuals without compromising performance is crucial.

As a result of the recent increased use of their game engines, Unity Technologies and Epic Games have taken the initiative to cater for a more simulation-based approach. In particular, Unreal Engine developers have been investing heavily in simulation to make the application easier to use, as well as to provide models and VFX to be used in simulation. Game engines provide a strong foundation for the development of new simulation systems, benefiting from decades of evolution, as they allow companies to make use of their technologies: rather than further developing pre-existing systems such as physics, AI and high-fidelity visual systems, time and money can be saved by utilising technologies already found in game engines. (Garratt et al., 2023.)

An example of simulation-oriented technology is the USD file format. Although it does not originate from a game engine, its rise to prominence in recent years as a multifunctional collaboration tool has prompted both Unity and Unreal Engine to develop add-ons to support it in their editors. The USD format is presented in a study authored by G.H. Nguyen et al. (2023) as a versatile layered data storage structure for 3D scenes. Originally invented by Pixar Animation Studios, USD incorporates a hierarchical system that allows for easy organisation of the scene and enables properties to be inherited from parent objects to their children. As reported in a 2023 article on Medium.com by Aaron Luk, Senior Engineering Manager of the USD ecosystem, the highly scalable format is available to the public as an open-source suite that uses Application Programming Interfaces (APIs) to allow users to create, edit, query, render, collaborate and simulate in virtual worlds. Notable use cases for USD qualities include digital twinning, where real-world areas such as factory floors are simulated accurately in large high-fidelity virtual scenes, and applications for autonomous driving.

3 3D Rendering

In its simplest form, **3D rendering** is defined as the production of a two-dimensional image based on three-dimensional data. This process almost always includes visual effects of varying complexity, resulting in photorealistic images complete with lighting and other ambient details. (Unity Technologies, n.d.)

Real-time rendering refers to the production of a scene by calculating 3D images at very high speeds before displaying them to create the impression that the environment surrounding the user is fully rendered regardless of the position it is being viewed from. This rendering technique incorporates the notion of interactivity, allowing for a more immersive experience if the operating system is capable of handling the constant high volume of background calculations. Real-time rendering is most often the technique of choice for video games, as it allows players to experience the full visual appeal of a 3D scene whilst also being able to interact with it. The main goal is to achieve the highest possible degree of photorealism at speeds that the human eye can perceive as natural movement. The minimum acceptable speed is usually 24 frames per second (FPS) and, the higher the frame rate goes, the more computer resources it requires to render at the desired speed. (XR Suite 2021; Unity Technologies, n.d.)

Offline or **pre-computed rendering** is the term used for high-quality rendering that does not rely on interactivity as a performance constraint. It allows for highly detailed and polished final images as a result of the process using more time to calculate a scene's lighting and shading. Compared to real-time rendering, where a frame is generally intended to be rendered in approximately 0.04 seconds (24 FPS) or faster, an individual frame calculated using offline rendering can take anywhere from minutes to even days to render depending on the complexity of the scene. Prominent adopters of offline rendering include the film and real estate industries, which use the technique for e.g., product presentation and as an alternative to replace live photoshoots. (3DHeven 2023; XR Suite 2021.)

As real-time rendering is the default rendering method offered by both Unity and Unreal Engine, offline rendering was not used as part of the practical implementation of the research described in this thesis.

4 Methodology

4.1 Aims and objectives

This thesis aims to establish concrete differences between Unity and Unreal Engine in terms of performance figures when performing real-time rendering tasks with different computer setups. Particular emphasis will be put on the behaviour of the two game engines during the rendering process in order to test the following hypotheses:

- Due to its level of detail and higher-quality results, Unreal Engine uses considerably more computer resources than Unity.
- Because of Unity's technological constraints, the computer performance level is a less significant factor for rendering.
- The rendering process is less taxing with each step up in computer system performance.
- USD assets are easier to work with in Unreal Engine than in Unity.

The first hypothesis is based on the fact that in various reviews, Unreal Engine has been praised for its ability to produce good-quality results with high levels of detail. This may be assumed to have a high cost in terms of computer resource usage.

Secondly, Unity is purportedly better optimised for lightweight tasks less heavy on computer systems. Hence the second hypothesis.

The logic of the third hypothesis is that computers with more processing power should be able to handle rendering tasks more easily regardless of the application used.

The final hypothesis has to do with the assumably better suitability of Unreal Engine for simulation-related work. USD assets can have extremely high levels of detail, which should make Unreal Engine the better option for handling them.

4.2 Testing setup and add-ons

Figure 1 summarises the elements of the overall testing setup. To ensure the best possible system performance for the creation of the final animated track as well as the subsequent rendering stage, version 5.1.1 of Unreal Engine was chosen due to its status as the most powerful iteration of the software at the time the practical testing was conducted. For the research carried out in Unity, version 2021.3.3f1 of the editor was used because of its stability as a Long-term Support (LTS) platform.

Figure 1 also shows the metrics chosen for this study. Each of them is relevant to the research outcome, as they serve to establish visible differences between the three computers used at the rendering stage.

Physical memory usage refers to the size of the data stored in the random-access memory (RAM) module of a computer at any given time. When a program is started, it is loaded from long-term storage into physical memory. If the specified file size limit of the physical memory is exceeded, a portion of the computer's storage is used to handle the excess data, potentially hindering system performance. (Lenovo US, n.d.-a)

Total Central Processing Unit (CPU) usage is defined as the percentage of processing power exhausted to process data and run various programs on a device. It is influenced by e.g. running background processes, malware and outdated drivers. (SolarWinds, n.d.)

In the context of this testing process, **Graphics Processing Unit (GPU) core load** is the percentage representation of the graphics card's processing power used at a particular time. Unlike with metrics such as physical memory usage, 100% GPU load does not mean that a GPU module is being overloaded; it signifies that the system is making full use of the module and that the component is performing normally. (MiniTool, 2022; Run:ai, n.d.)

Additional points of comparison are provided in the form of **overall rendering time** and **frame creation rate**, in particular to obtain data on the predicted gap

between Unity and Unreal Engine in terms of their handling of each individual rendering task.

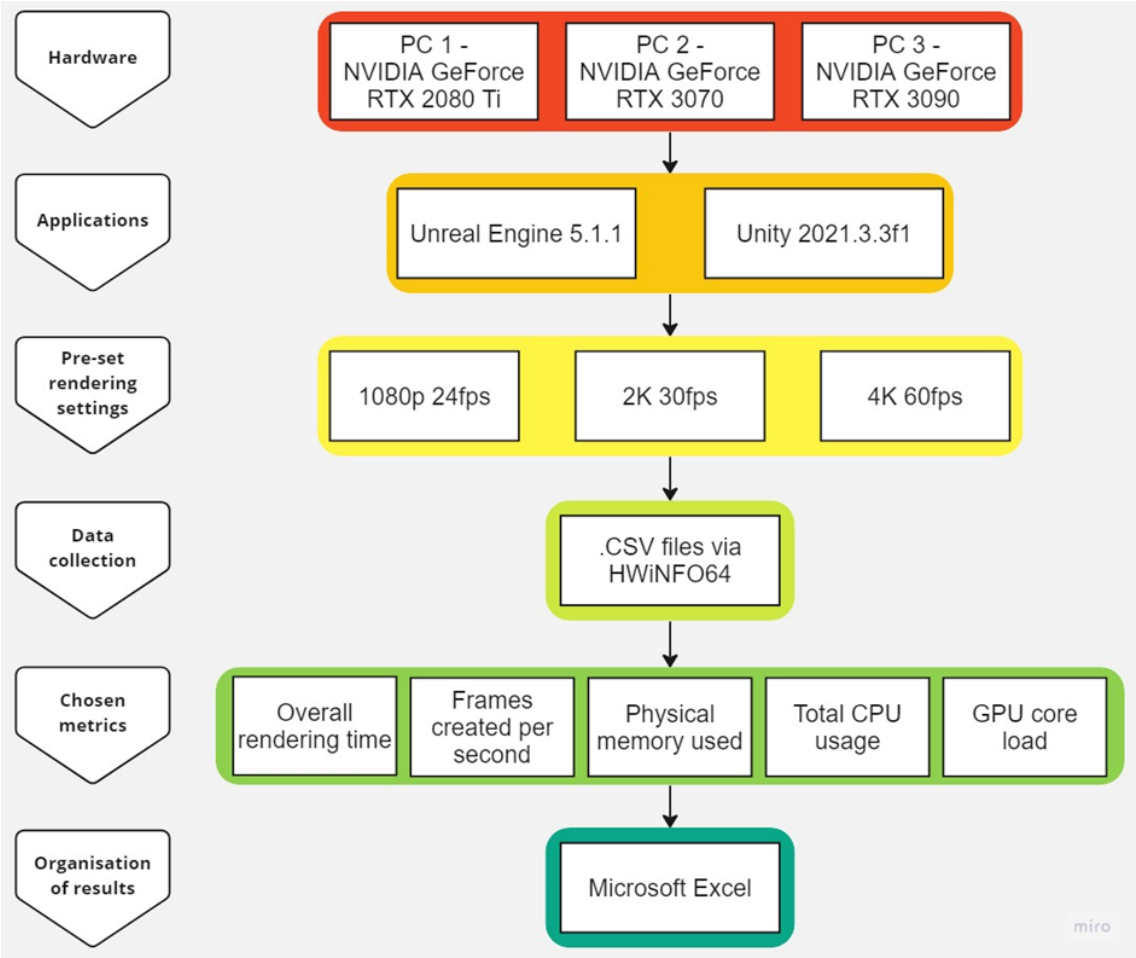

Figure 1. Breakdown of the testing setup.

To make the recording and rendering of a cinematic sequence possible, relevant add-ons were installed in both applications. While Unreal Engine already comes with a Sequencer interface as a standard feature, animations built with it required the addition of the Movie Render Queue tool to make it possible to render and export them as well as to add optional post-processing effects. Stitching the rendered frames into a playable .MP4 video also necessitated the installation of FFmpeg, a third-party software suite for recording, converting and streaming a wide variety of audio and video formats. It should be noted that almost any

general-purpose video editing software can also be used to combine the individual frames into a full animated sequence.

HWiNFO, a third-party system monitoring and diagnostics software, was used to record system performance during the rendering. This application records statistics such as memory availability, temperatures and power usage across all of the computer's different parts. It also produces detailed system information, such as the items in Table 1.

4.3 Testing environment

Table 1 presents the detailed specifications of the computers used in the testing process. From the beginning, it was understood that rendering the animated sequences on one computer only would lead to results inherently aligned with that specific system's strengths and weaknesses. To eliminate any such biases, three different desktop computers were chosen for the testing process in order to minimise potentially skewed data influenced by an individual system's performance.

Table 1. Computers used in the testing process: Detailed specifications.

	PC1	**PC2**	**PC3**
CPU	Intel Core i9-9900 @3.10 GHz	Intel Core i7-11700F @2.50 GHz	AMD Ryzen 5900
Memory size	64 GB	16 GB	64 GB
Memory type	DDR4 SDRAM	DDR4 SDRAM	DDR4 SDRAM
Memory clock	1333 MHz	1600 MHz	1800 MHz
Motherboard chipset	Intel Z370 (Kaby Lake)	Intel B560 (Rocket Lake PCH-H)	AMD X570 (Bixby)
GPU	NVIDIA GeForce RTX 2080 Ti	NVIDIA GeForce RTX 3070	NVIDIA GeForce RTX 3090
GPU memory size	11 GB	8 GB	24 GB
GPU memory type	GDDR6 SDRAM	GDDR6 SDRAM	GDDR6X SDRAM
GPU clock	300 MHz	210 MHz	1925 MHz
Hard drive type	NVMe 4x 8.0 GT/s	SATA 6Gb/s	NVMe 4x 8.0 GT/s

All testing was conducted using computers running Windows 10 and 11 operating systems. The machines shown in Table 1 were chosen on the basis of the GPUs fitted into them so that three levels of graphics processing power would be represented in the tests. To bridge the gap between the workflow differences of Unity and Unreal Engine, a ready-made environment was procured from NVIDIA's USD sample assets. At the time of writing, neither Unity nor Unreal Engine natively supported importing USD assets into their base editor environments. For the usage of USD assets, both applications provide their own solutions for USD handling via their respective add-on interfaces: an experimental

USD package can be imported from the Unity Package Manager, and Unreal Engine's plugins include the USD Stage interface. The Unity package converts USD assets into formats supported by the application while the Unreal Engine plugin works with USD file formats in their unaltered form. Hence, they feature two different approaches to the workflow.

A common workspace for all measurements done in the context of the study was needed to ensure a high degree of uniformity in the measurement phase. NVIDIA's USD Attic sample scene was chosen as testing environment for both applications in order to provide identical starting points and to keep avoidable deviations in system performance to a minimum.

4.4 Workflow

A causal-comparative research approach was deemed necessary for the project, as the majority of the work involved analysing and drawing conclusions from measured data pools. The fundamental differences between the game engines' layouts and integrated technologies meant that, instead of building identical scenes in both, the static USD environment mentioned in the previous section first had to be set up in an Unreal Engine workspace so that the animation sequence could be built into it.

The first step was to import the USD Attic sample scene into the editor. This necessitated the installation of the USD Stage plugin which allows the user to view and modify USD assets without changing their file format (see Figure 2).

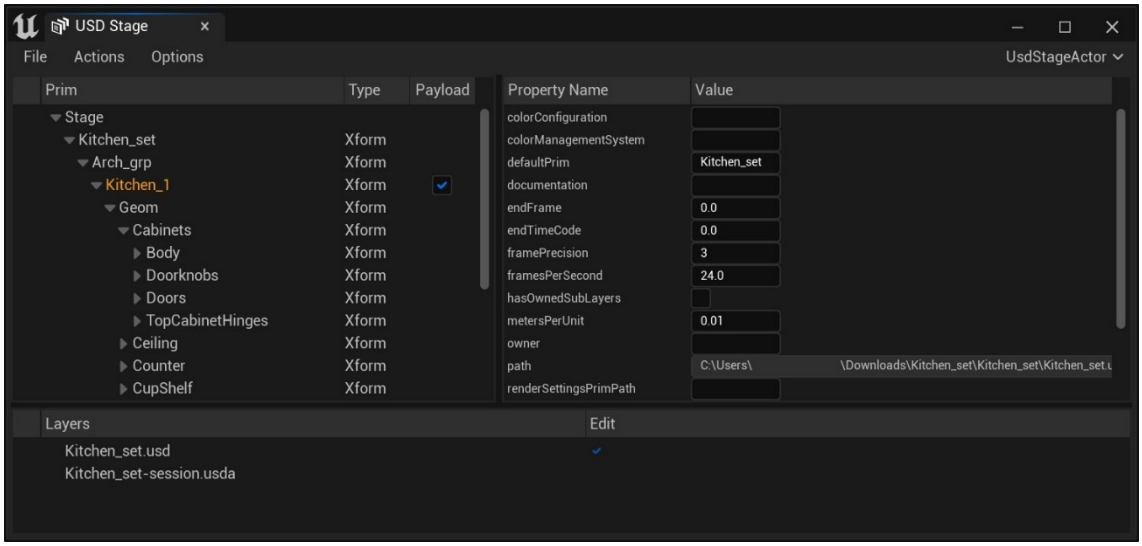

Figure 2. The USD Stage interface in Unreal Engine 5. (Unreal Engine 5.1 Documentation 2022)

Once the USD asset is loaded in via the USD Stage plugin, it is ready to be used and modified within the application as the user sees fit. For this workflow, the Attic sample scene was not modified in any way beyond disabling its built-in camera component and replacing it with Unreal Engine's built-in Cinematic Camera functionality. This camera system is automatically associated with the application's Sequencer tool with which the user is able to record a customised movement path. Changes in the asset position, rotation and scale values can be recorded with the Sequencer's timeline, and transitions and inbetweening are then applied automatically to create a smoothly flowing animation. The final camera path was laid out in a roughly elliptical shape which featured movement on all three axes as well as constant rotation.

Lighting in the scene was set up so that the attic scene would be lit by bright sunlight shining into the environment directly from the windows visible in Figure 3. Post-processing effects were enabled and adjusted to conceal the otherwise empty workspace outside of the windows in the scene and to make the lighting as realistic as possible.

Figure 3. The camera path overlaid onto a 3D view of the attic space.

After the animated sequence is completed, it can be rendered via the Movie Render Queue plugin (see Figure 4). The user is presented with an array of settings to choose from to determine the final quality and technical parameters of the output. For the purposes of this thesis, one setting pre-set was created for each resolution/frame rate pairing:

- 1080p/24 FPS
- 2K/30 FPS
- 4K/60 FPS

The Movie Render Queue's default behaviour is to export each rendered frame individually into the desired destination file folder. Therefore, installing FFmpeg and linking it to Unreal Engine was a necessary step to output a fully rendered video clip in the desired file format. .MP4 was chosen as the final format due to its versatility and ability to be streamed on nearly all devices capable of video playback. Sufficient quality for all of the chosen output settings was determined with test runs on each computer, after which measurements were collected and reformatted. The rendering process itself was completed with only the game

engine and HWiNFO running so that all available computing resources were directed to simultaneously creating the fully rendered video clip frames and recording the data that would eventually be used in the analysis phase.

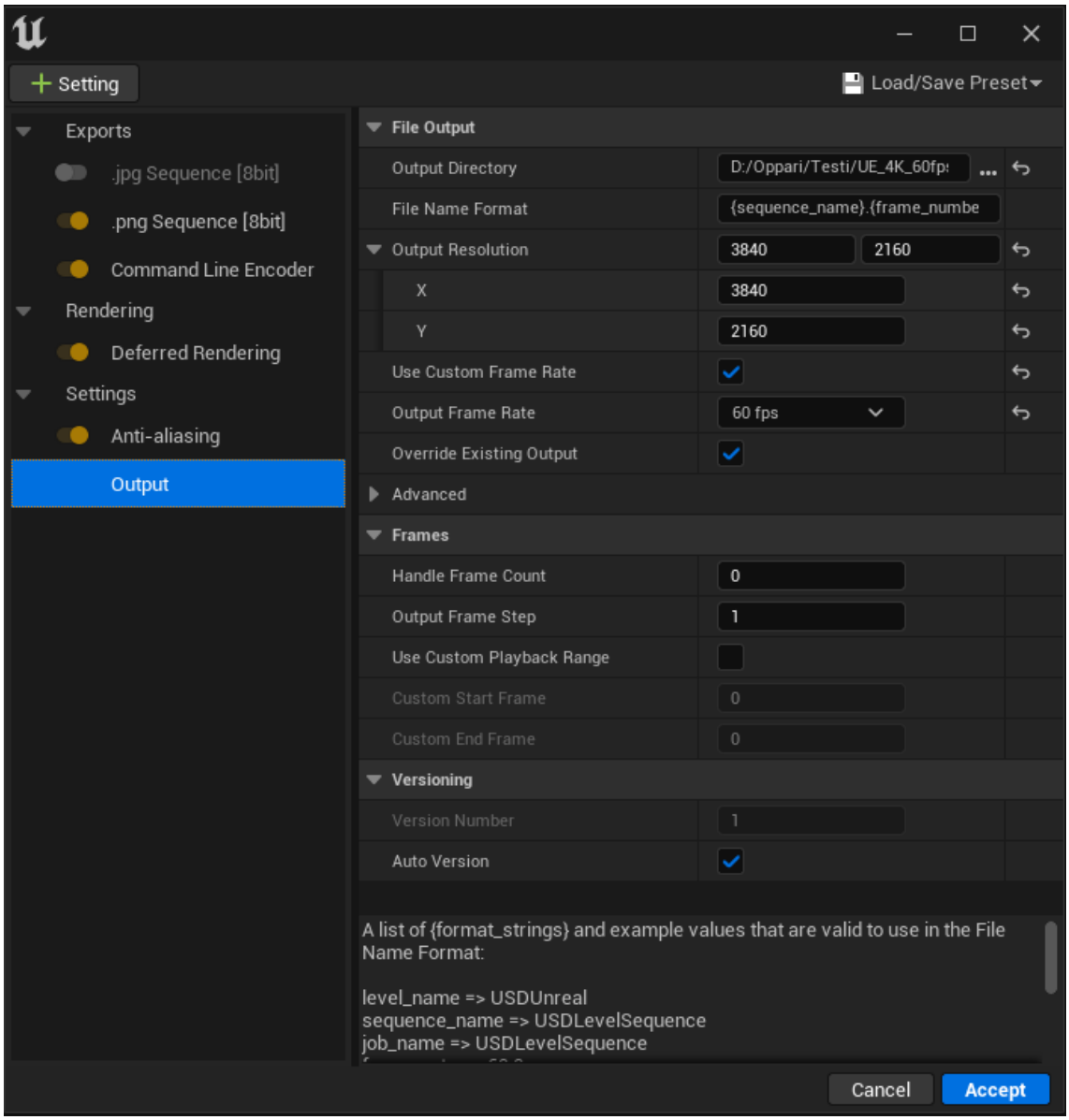

Figure 4. Detailed output settings for the 4K render in Unreal Engine's Movie Render Queue.

The workflow in Unity introduced some additional challenges to the process, starting with importing the USD scene into the editor. Unlike in Unreal Engine, the plugin needed for bringing the attic scene into the workspace was in an experimental state, and very little useful documentation could be found from official sources. Another difference was the way Unity's USD importing plugin handled USD assets: rather than bringing an asset directly into the scene, it first had to be converted into a standard GameObject before it could be handled within the workspace. When recreating the camera motion path, the position and rotation values on the editor's X and Z axes had to be switched around because Unity and Unreal Engine had different geometrical axis layouts. In addition, the transitioning motion between the camera's successive positions could not be recreated precisely: instead of smooth, non-stop inbetweening, the camera transitions required all movement to stop at each pre-defined point before it could be resumed towards the next position. These transitions can be seen in Figure 5, where each one stops upon reaching the next camera position.

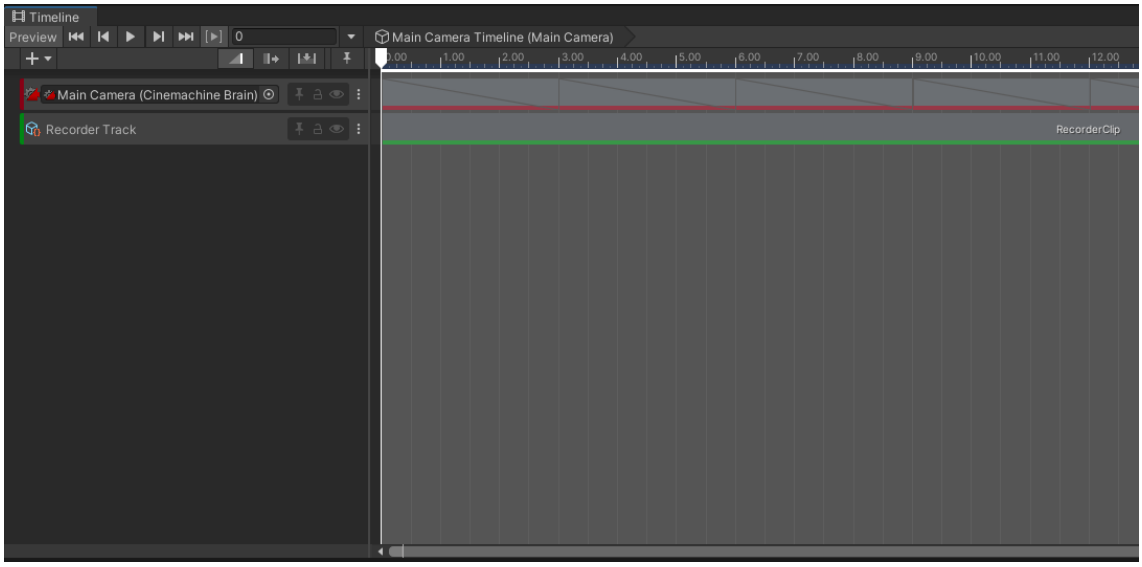

Figure 5. The Timeline window in Unity, displaying timed transitions between individual points on the camera's motion path.

In addition, Unity's lighting settings differed considerably from those of Unreal Engine, requiring further adjustment of the lighting conditions to reach an

approximately similar appearance. Yet, the results were significantly different, as can be seen in Figure 6.

Figure 6. Comparison between the first frames of the animated sequence in the two game engines.

Once each rendering task had been completed successfully, the contents of each .CSV file were reformatted into a readable form in Microsoft Excel and reviewed to obtain the necessary data and draw conclusions about the overall performance of each test computer. The most relevant data points were then isolated from the raw information and compiled into tables and charts for easier viewing and analysis.

5 Results

5.1 Metrics

Due to the abundance of data provided by HWiNFO64, narrowing down the results was necessary in order to locate and collect useful data points. On average, a successfully recorded data set contained approximately 345 columns of raw, unsorted data points, of which the majority held no useful information for the purposes of this thesis. As explained in section 4.2, the chosen metrics were the following:

- Overall rendering time
- Frames created per second
- Physical memory used
- Total CPU usage
- GPU core load

The data was collected with only the tested game engine and HWiNFO64 running during rendering processes. This ensured that the results represented the computing power of the systems being used solely to render the animated sequences instead of being distributed between other active programs.

5.2 Rendering results

The final data collection process involved little active work, as the majority of the time taken to obtain adequate results was spent waiting for the test computers to complete the rendering tasks and export the finished video clips. Once the relevant information had been isolated from the raw data, it could be compiled and organised into easily readable charts from which conclusions could be drawn. For brevity, the evaluated computers will be referred to as PC1, PC2 and PC3 in the order of their predicted performance figures from least to most powerful respectively.

5.2.1 General measurements

The final overall length of each recorded animated sequence was 24 seconds at varying frame rates. As seen in Figure 7, the rendering process in Unreal Engine took considerably longer to be completed on all testing platforms than in Unity. It can also be noted that the average time needed to run through each rendering task diminished with each step up in system performance.

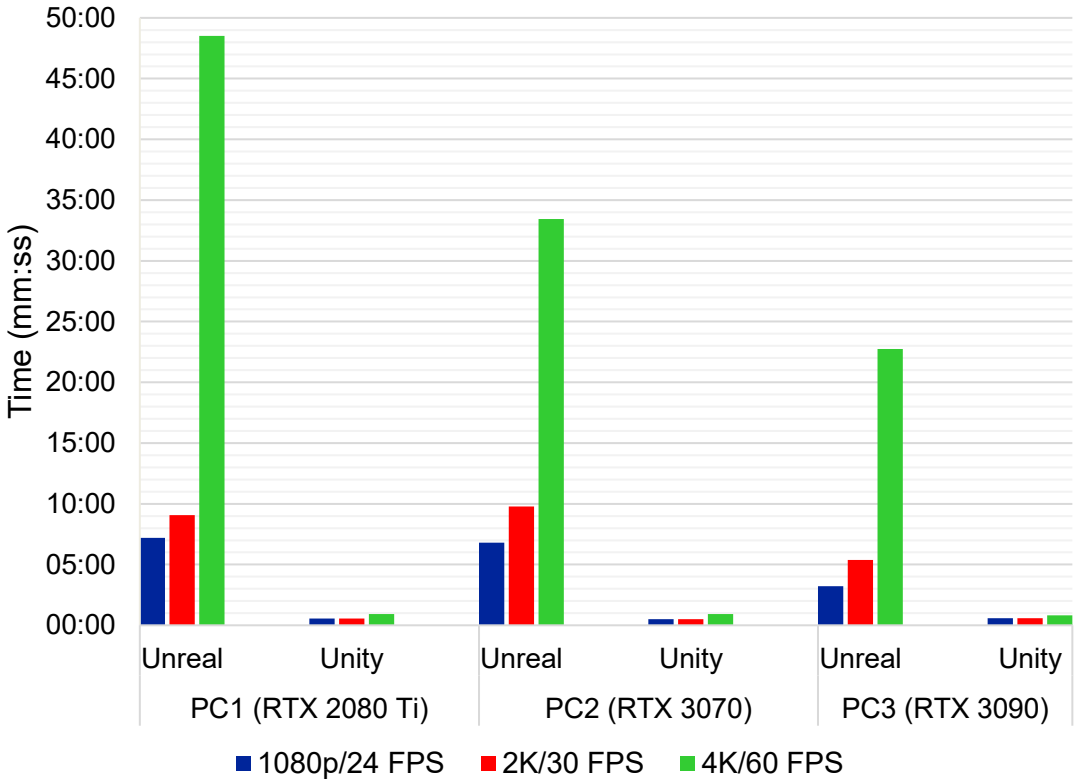

Figure 7. Time taken to finish each rendering task.

Figure 8 shows that the rates at which frames were created were much higher in Unity than in Unreal Engine. This corroborates the data in Figure 7, as less time required to finish the rendering tasks translates directly into higher frame creation rates. The rendering pre-sets produced 576 frames at 24 FPS, 720 frames at 30 FPS and 1440 frames at 60 FPS. These were calculated by multiplying the length of the animated sequences (24 seconds) by the frame rate they were rendered at.

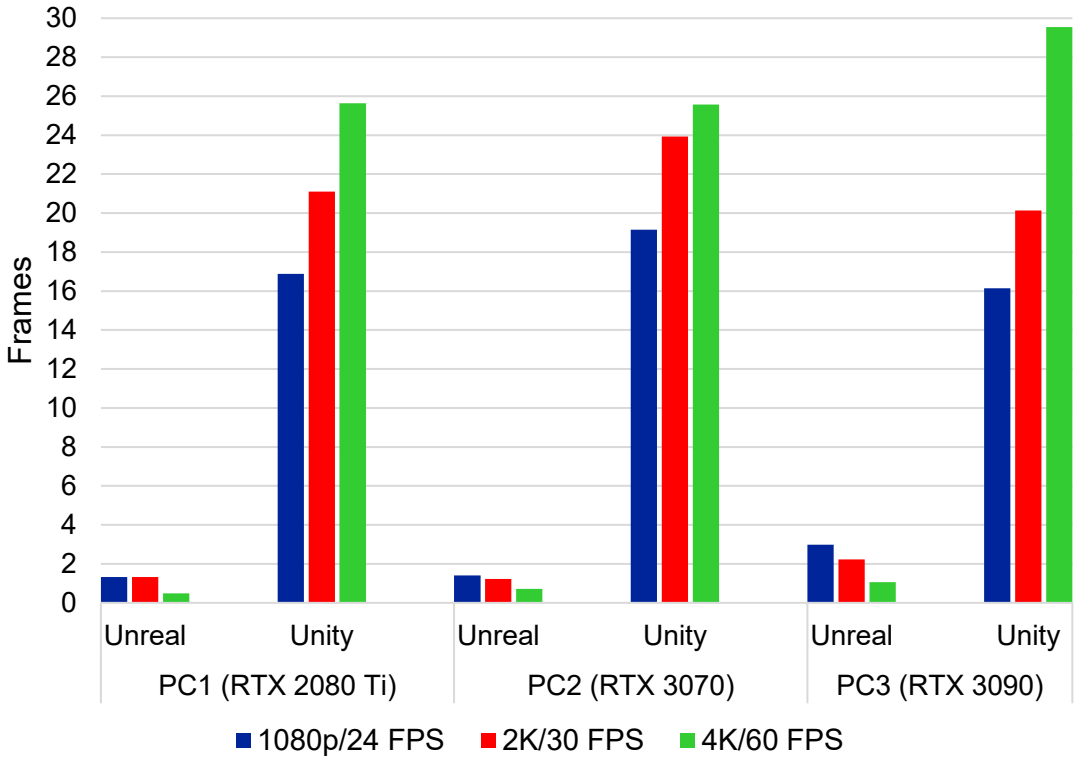

Figure 8. Average number of frames created per second of rendering time.

5.2.2 PC1 results

Table 2 summarises the rendering times and their differences on the PC1 system with each setting pre-set. All Unity renders on the PC1 system were completed in less than one minute, while the fastest Unreal Engine render took more than 7 minutes. On average, completing rendering tasks using Unreal Engine on the PC1 system took approximately 31 times longer than in Unity. The duration of the 4K render was as much as some 52 times longer.

Table 2. PC1: Rendering time statistics per each setting pre-set.

	Unreal Engine 5.1.1	**Time difference (%)**	**Time difference (mm:ss)**	**Unity 2021.3.3f1**
1080p/24 FPS	07:13	1274%	06:39	00:34
2K/30 FPS	09:04	1600%	08:30	00:34
4K/60 FPS	48:30	5196%	47:34	00:56
Combined average	21:35,6	3122%	20:54,1	00:41,5

As shown in Figures 9, 10 and 11 as well as Table 3, average physical memory usage of the PC1 system in Unreal Engine amounted to approximately 30,800 MB and never exceeded 45,000 MB. Unity processes generally settled on memory usage between 15,000 and 17,000 MB, with the 4K render exhibiting the highest consumption. With Unreal Engine, the highest resolution exhibited the lowest memory usage out of all three test cases. As such, physical memory usage at any given time remained well below the established maximum of 64 GB (see Table 1) throughout the rendering tasks even after brief elevated rates at the beginning of each process. On average, the physical memory usage of Unreal Engine render processes was approximately twice that of Unity.

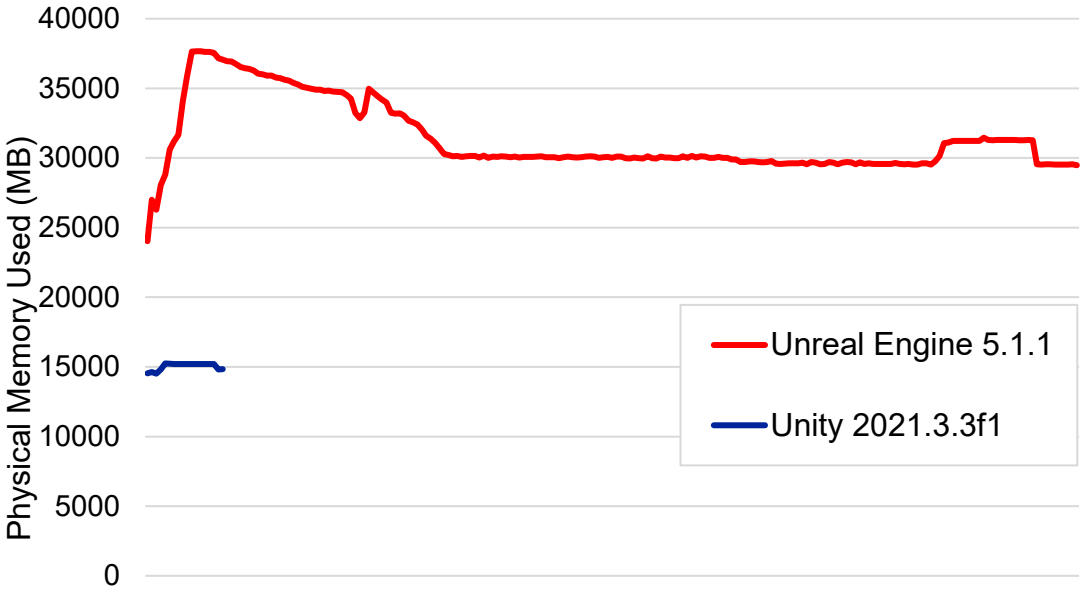

Figure 9. PC1: Physical memory usage over the duration of the 1080p/24 FPS rendering task.

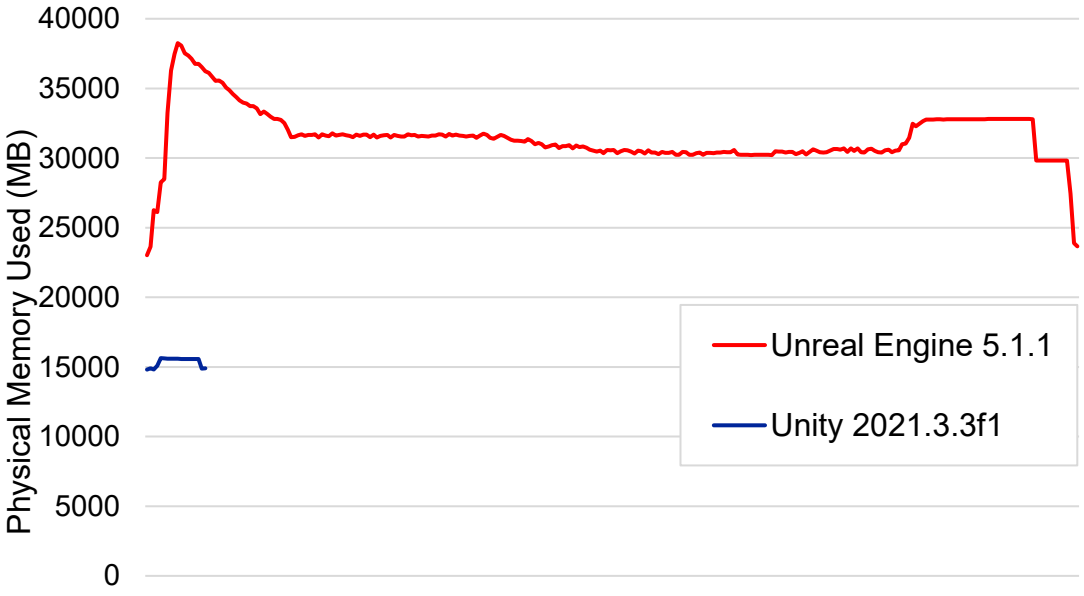

Figure 10. PC1: Physical memory usage over the duration of the 2K/30 FPS rendering task.

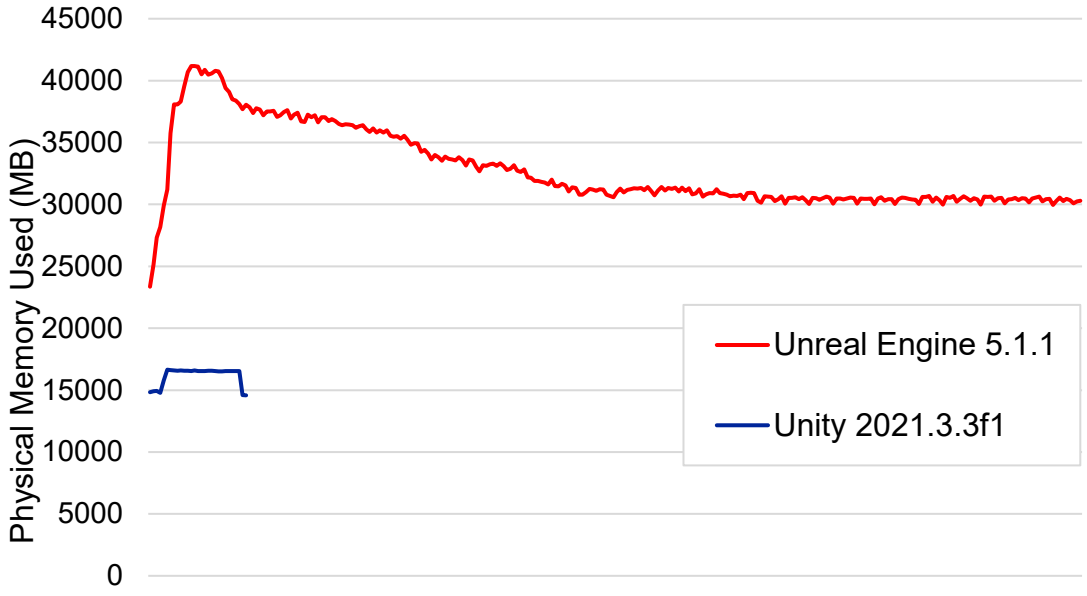

Figure 11. PC1: Physical memory usage over the duration of the 4K/60 FPS rendering task.

Table 3. PC1: Average physical memory usage in all test cases.

	Unreal Engine 5.1.1	**Unity 2021.3.3f1**
1080p/24 FPS	31360.41 MB	15038.17 MB
2K/30 FPS	31483.97 MB	15360.11 MB
4K/60 FPS	29648.80 MB	16163.38 MB
Combined average	30831.06 MB	15520.55 MB

The total CPU usage figures displayed notable fluctuation in all test cases from start to finish (see Figures 12, 13 and 14). The most significant example can be seen in Figure 12, showing the CPU running at 100% capacity during 1080p rendering in Unreal Engine before returning to sub-40% usage. No other instances of abnormally high usage were recorded, apart from a few momentary spikes, also in Unreal Engine, that went up to a maximum of approximately 90%.

Table 4 shows that average CPU usages during 1080p and 2K Unity renders were some 25 to 19 percentage points lower than during the 4K render. These

usage values were reversed in Unreal Engine, with somewhat less drastic differences.

CPU usage during Unity processes remained below 50% with the 4K render using the most processing power. The 1080p and 2K renders displayed a spike at the beginning, after which the numbers stabilised at around 12% and 20% respectively.

A slight rise in CPU usage was seen towards the end of all Unreal Engine test cases. Figure 14 illustrates this trend, with average usage ending up between 20 and 30% prior to a final steep spike.

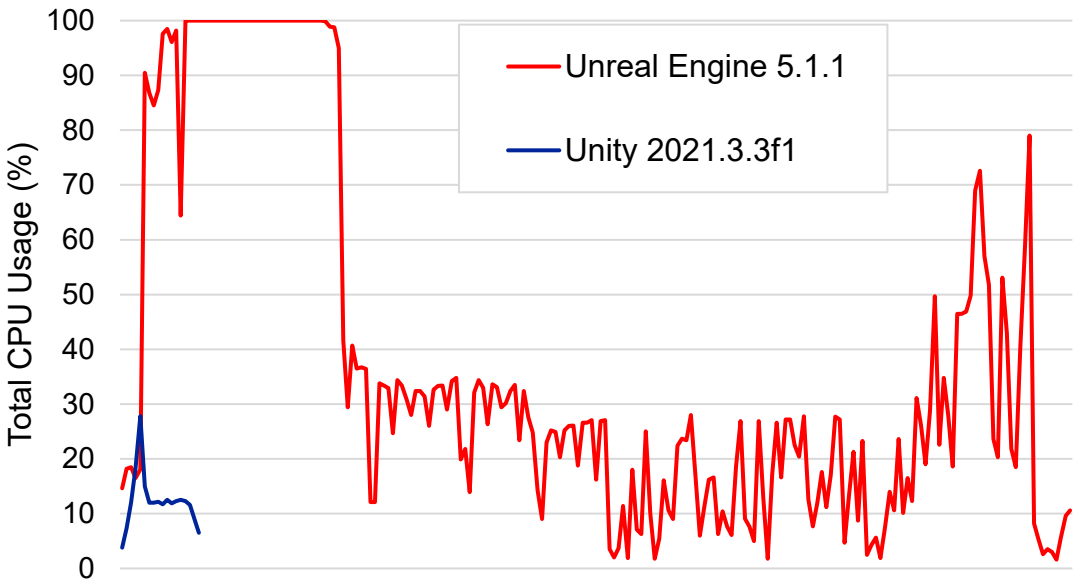

Figure 12. PC1: Total CPU usage over the duration of the 1080p/24 FPS rendering task.

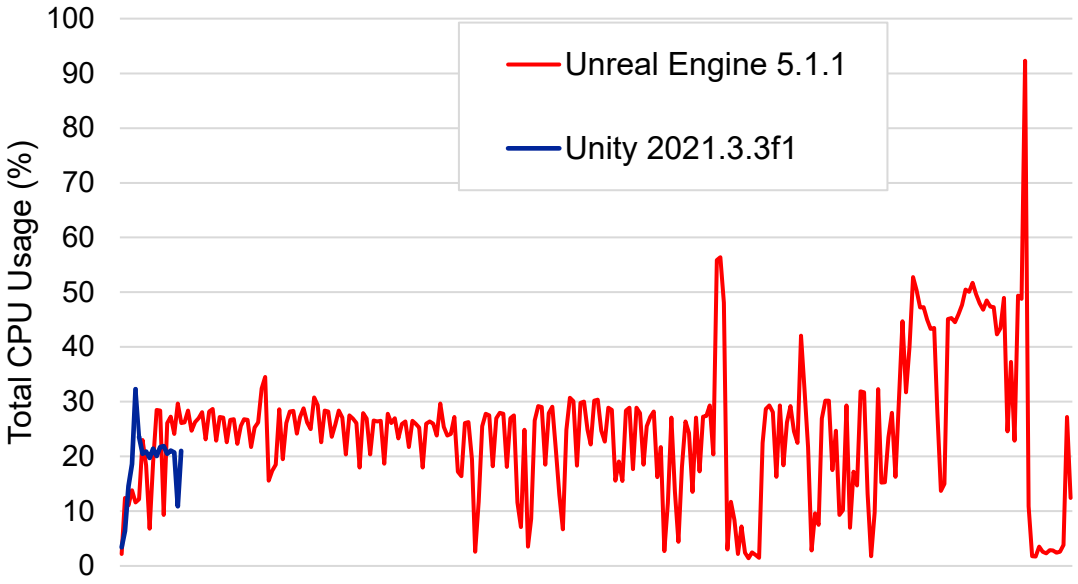

Figure 13. PC1: Total CPU usage over the duration of the 2K/30 FPS rendering task.

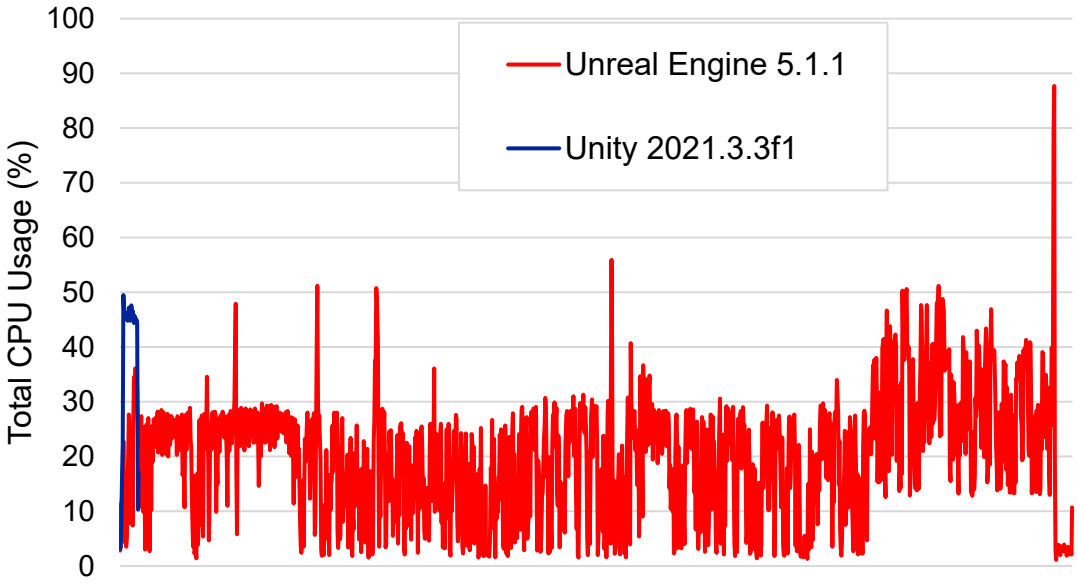

Figure 14. PC1: Total CPU usage over the duration of the 4K/60 FPS rendering task.

Table 4. PC1: Average total CPU usage in all test cases.

	Unreal Engine 5.1.1	**Unity 2021.3.3f1**
1080p/24 FPS	38.32%	12.26%
2K/30 FPS	24.48%	18.85%
4K/60 FPS	20.10%	37.66%
Combined average	27.63%	22.92%

As illustrated in Figures 15, 16 and 17, GPU core load percentages showed fluctuating rates that alternated between near-maximum and minimum throughout the test cases. While the highest loads measured during Unreal Engine rendering were close to 100%, the momentary dips to sub-50% utilisation meant that on average, only approximately 55% of the GPU processing power was reached (see Table 5). In Unreal Engine, the core load during the 1080p and 2K processes did not reach full capacity, but the 4K render touches 100% processing power several times over its duration. In contrast, as shown in Figures 15 and 16, Unity processes remained below 40% in the first two test cases, and the 4K render reached around 75% at its peak.

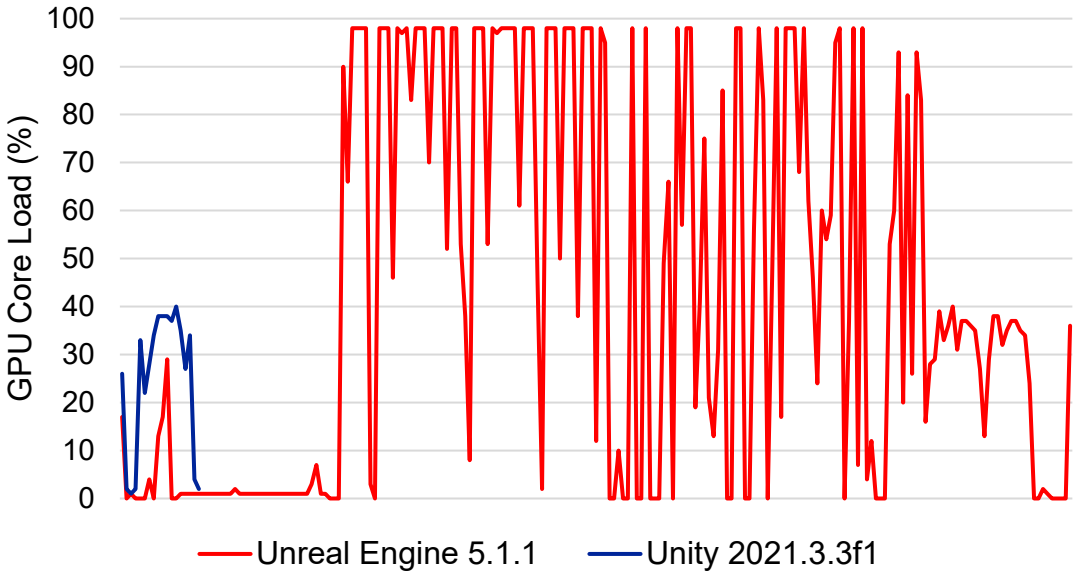

Figure 15. PC1: GPU core load over the duration of the 1080p/24 FPS rendering task.

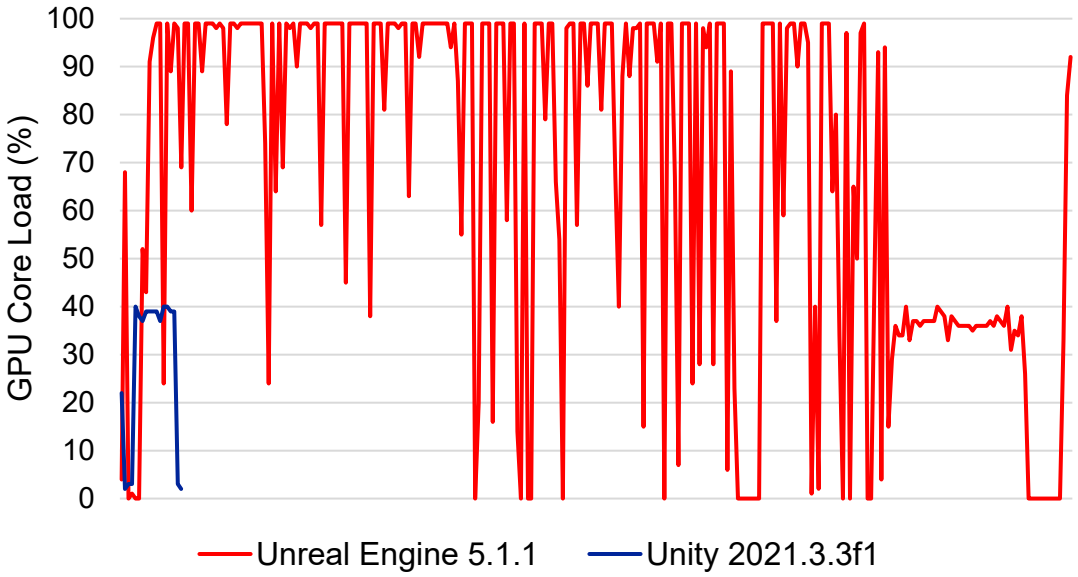

Figure 16. PC1: GPU core load over the duration of the 2K/30 FPS rendering task.

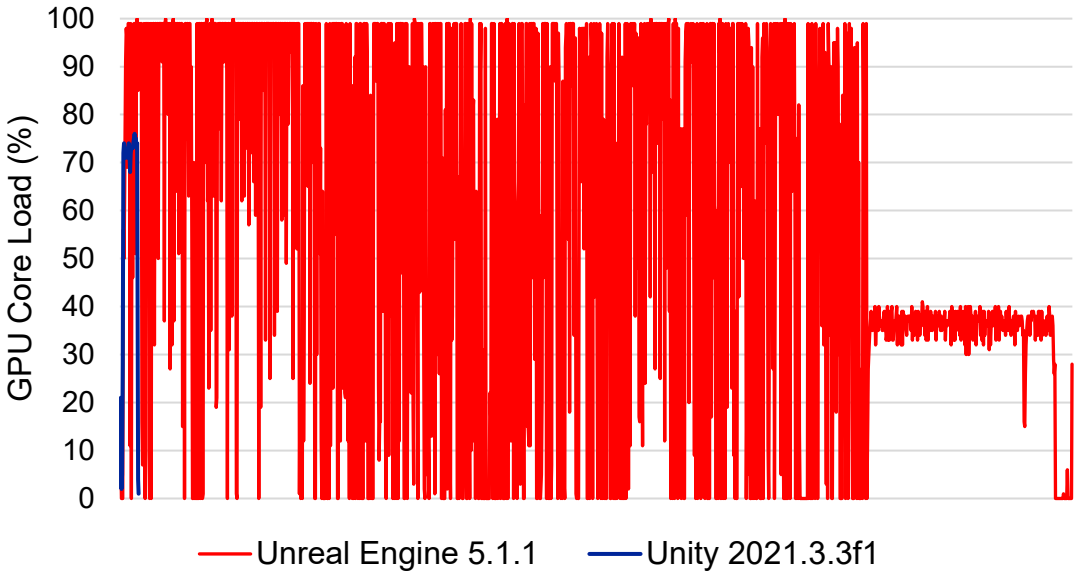

Figure 17. PC1: GPU core load over the duration of the 4K/60 FPS rendering task.

Table 5. PC1: Average GPU core load in all test cases.

	Unreal Engine 5.1.1	**Unity 2021.3.3f1**
1080p/24 FPS	42.65%	24.50%
2K/30 FPS	67.71%	27.83%
4K/60 FPS	54.36%	57.52%
Combined average	54.90%	36.62%

5.2.3 PC2 results

Table 6 summarises the rendering times and their differences on the PC2 system with each setting pre-set. With the PC2 system, the rendering tasks took, on average, approximately 26 times longer to complete in Unreal Engine than in Unity. Similar to the PC1 measurements, the 4K render had the largest time difference, amounting to 32 minutes and 31 seconds (roughly 36 times longer than in Unity). Overall, every rendering process, except for the 2K task in Unreal

Engine and the 4K task in Unity, took less time to finish than the same processes carried out using the PC1 system.

Table 6. PC2: Rendering time statistics per each setting pre-set.

	Unreal Engine 5.1.1	**Time difference (%)**	**Time difference (mm:ss)**	**Unity 2021.3.3f1**
1080p/24 FPS	06:48	1360%	06:18	00:30
2K/30 FPS	09:46	1953%	09:16	00:30
4K/60 FPS	33:27	3584%	32:31	00:56
Combined average	16:40,3	2578%	16:01,5	00:38,8

In all test cases, the PC2 system used less memory on average than the PC1 system to complete the rendering processes. Unity processes exhibited a pattern similar to tests on the PC1 system, with memory usage rising from around 10,000 MB to between 11,000 and 12,000 MB at the highest. The average usage displayed in Table 7 reflects this, with both 1080p and 2K renders using around 10,600 MB of memory, while usage during the 4K render was approximately 1,000 MB higher.

Unreal Engine's average physical memory usage was roughly 500 MB lower than Unity's, with the combined average amount equating to roughly one third of the PC1 average. In all three test cases, the chart lines illustrated in Figures 18, 19 and 20 displayed similar usage statistics at set points throughout the testing processes: At the beginning, two peaks could be observed at which memory usage momentarily climbed to approximately 16,000 MB and then fell and stabilised before rising again prior to the endpoint of the recorded data set. Some differences were observed in the 4K render chart where physical memory usage decreased further halfway through the rendering process before jumping up considerably and reaching between 13,000 and 14,000 MB at the end, higher than in the other two cases.

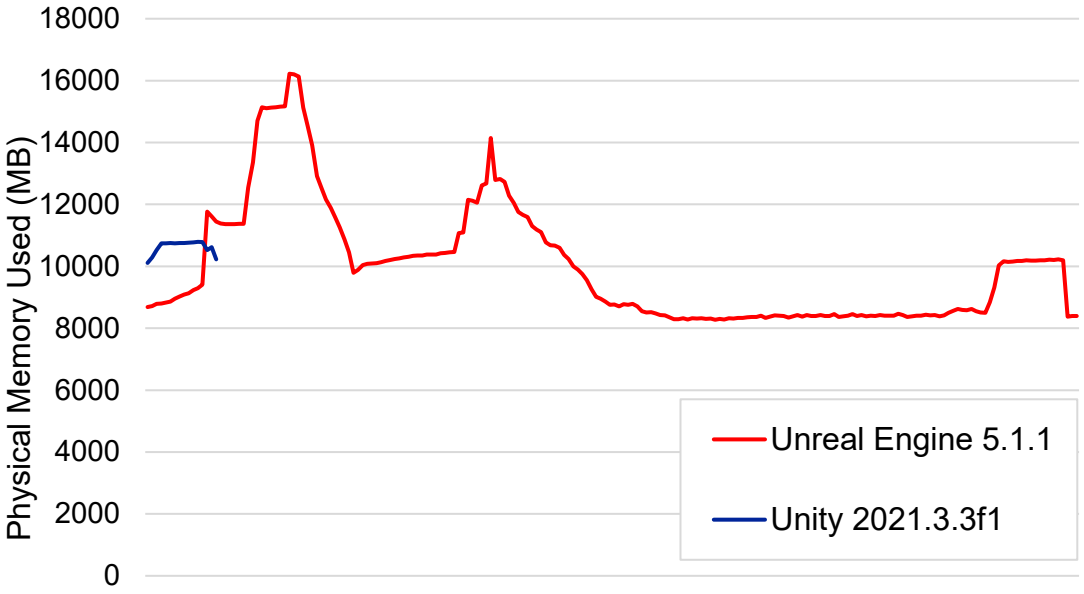

Figure 18. PC2: Physical memory usage over the duration of the 1080p/24 FPS rendering task.

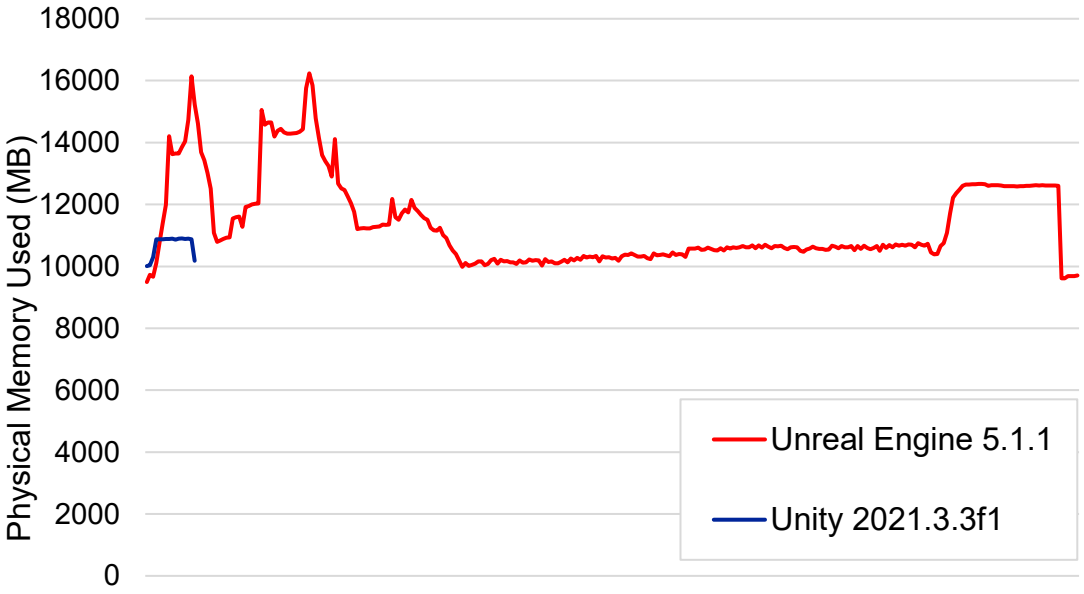

Figure 19. PC2: Physical memory usage over the duration of the 2K/30 FPS rendering task.

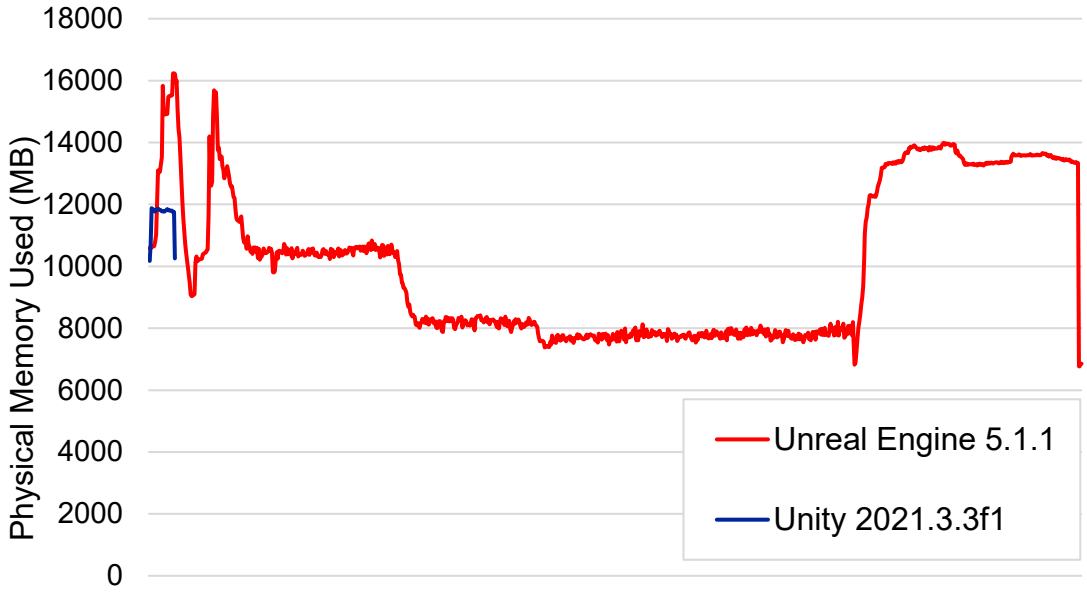

Figure 20. PC2: Physical memory usage over the duration of the 4K/60 FPS rendering task.

Table 7. PC2: Average physical memory usage in all test cases.

	Unreal Engine 5.1.1	**Unity 2021.3.3f1**
1080p/24 FPS	9982.37 MB	10618.50 MB
2K/30 FPS	11362.05 MB	10696.13 MB
4K/60 FPS	10071.19 MB	11604.72 MB
Combined average	10471.87 MB	10973.12 MB

The total CPU usage of the PC2 system was found to be consistently volatile throughout all test cases, as shown in Figures 21, 22 and 23. This is most prominently visible in Figure 23, which shows the computing power used by Unity briefly rising to approximately 80% before returning to around 50% at the end of the recording. A similar spike can be seen in Figure 22, where CPU usage jumps to nearly 70% prior to settling at slightly above 30%.

The Unreal Engine results indicate a lower net CPU usage at the beginning of each rendering process before stabilising between approximately 35% and 50%

and a new increase to 60-70% towards the end. In the 4K render, this final jump was less pronounced, only reaching around 55%, except for the very end, where usage momentarily increased to over 95%. Average CPU usage remained below 30% for both Unity and Unreal Engine rendering tasks (see Table 8).

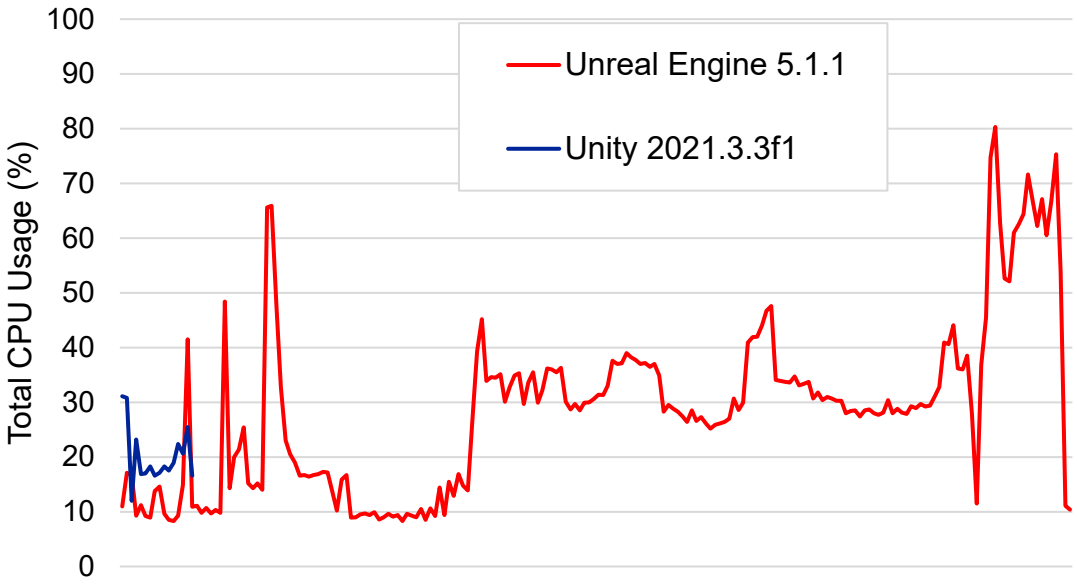

Figure 21. PC2: Total CPU usage over the duration of the 1080p/24 FPS rendering task.

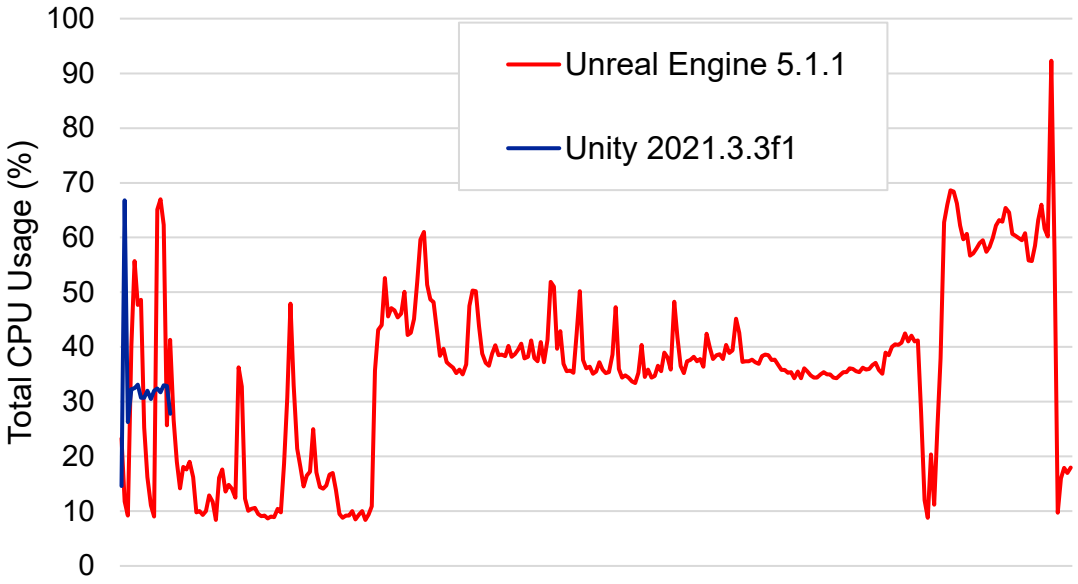

Figure 22. PC2: Total CPU usage over the duration of the 2K/30 FPS rendering task.

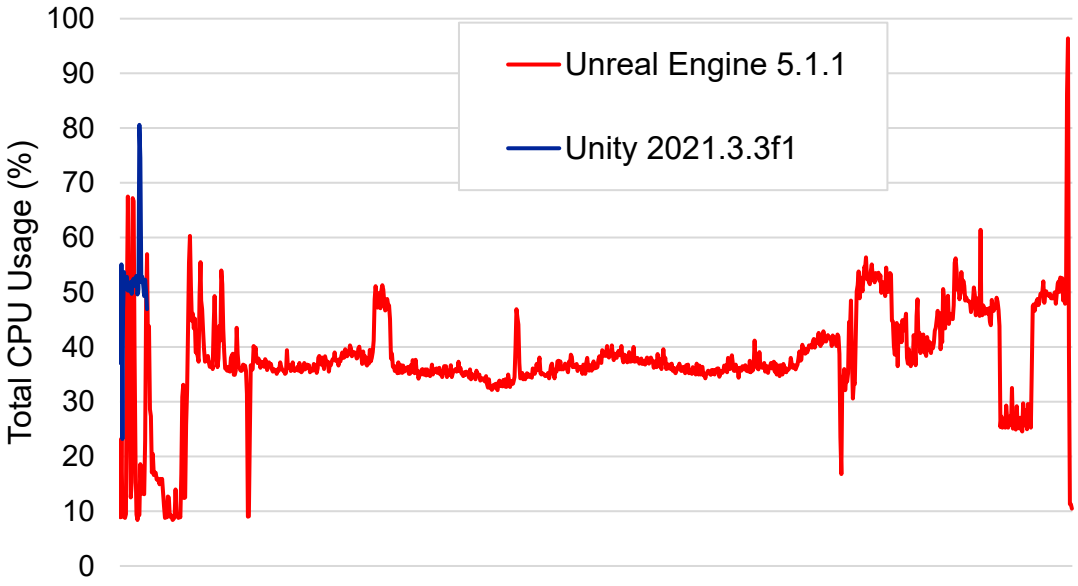

Figure 23. PC2: Total CPU usage over the duration of the 4K/60 FPS rendering task.

Table 8. PC2: Average total CPU usage in all test cases.

	Unreal Engine 5.1.1	**Unity 2021.3.3f1**
1080p/24 FPS	38.32%	12.26%
2K/30 FPS	24.48%	18.85%
4K/60 FPS	20.10%	37.66%
Combined average	27.63%	22.92%

Stark differences between Unity and Unreal Engine's handling of GPU core load are immediately apparent in Figures 24, 25 and 26. As shown in Table 9, the tests performed in Unity resulted in an average core load of some 38%, while Unreal Engine averaged slightly over 65%. In Unity, a momentary maximum between 60% and 70% was visible in the 2K and 4K render graphs. The 1080p rendering process reached a maximum of approximately 40% at its peak.

By comparison, test results from Unreal Engine show long periods during which the GPU ran at 100% capacity with little to no deviations. Noteworthy instability is present at both ends of all three Unreal Engine graphs, as shown especially in Figures 24 and 25 where the core load drops down to 0% on multiple occasions. Between these two behaviours, the core load exhibited fluctuation of varying intensity. Towards the end of each Unreal Engine test case, the GPU core load only rose to an approximate maximum of 40%, the 1080p rendering task displaying a momentary jump to almost 60% at the very end of the task.

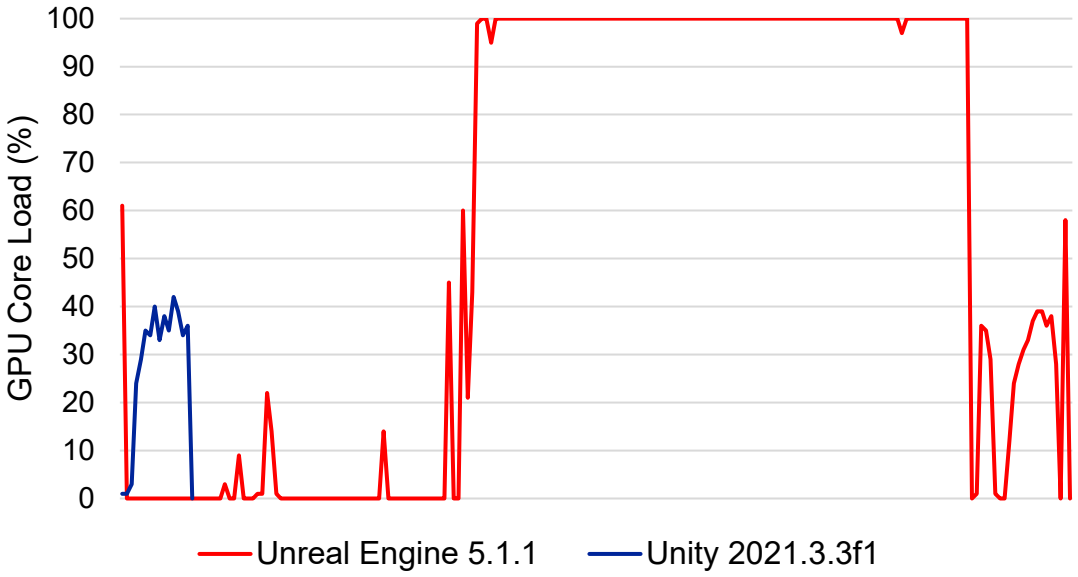

Figure 24. PC2: GPU core load over the duration of the 1080p/24 FPS rendering task.

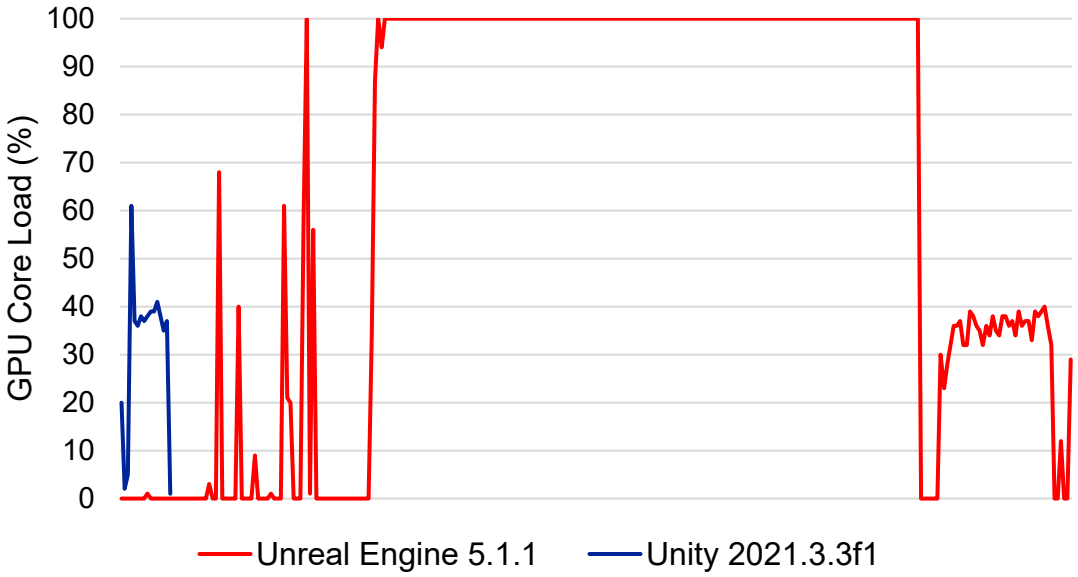

Figure 25. PC2: GPU core load over the duration of the 2K/30 FPS rendering task.

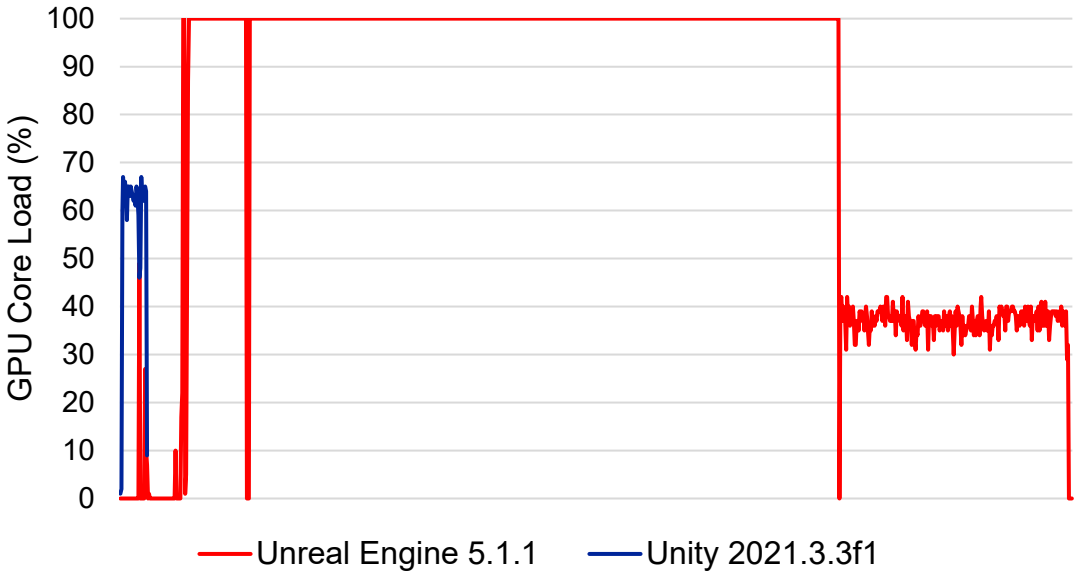

Figure 26. PC2: GPU core load over the duration of the 4K/60 FPS rendering task.

Table 9. PC2: Average GPU core load in all test cases.

	Unreal Engine 5.1.1	**Unity 2021.3.3f1**
1080p/24 FPS	55.84%	26.50%
2K/30 FPS	63.23%	31.50%
4K/60 FPS	77.40%	56.07%
Combined average	65.49%	38.02%

5.2.4 PC3 results

Table 10 summarises the rendering times and their differences on the PC3 system with each setting pre-set. On this computer rendering times follow largely the same pattern as the times recorded with the PC1 and PC2 systems. On average, Unreal Engine renders took approximately 16 times longer to complete than their Unity counterparts. As was the case with the other two computers, the

4K render presented the most significant difference, with Unreal Engine using nearly 28 times more time than Unity.

Physical memory measurements were not available for the PC3 system as the .CSV files containing the data collected from all test cases in both Unity and Unreal Engine were left incomplete due to unknown reasons. Total CPU usage and GPU core load statistics were extractable, and are displayed in this section.

Table 10. PC3: Rendering time statistics per each setting pre-set.

	Unreal Engine 5.1.1	**Time difference (%)**	**Time difference (mm:ss)**	**Unity 2021.3.3f1**
1080p/24 FPS	03:13	536%	02:37	00:36
2K/30 FPS	05:22	894%	04:46	00:36
4K/60 FPS	22:45	2785%	21:56	00:49
Combined average	10:26,9	1563%	09:46,8	00:40,1

The measured CPU usage in all tests conducted on the PC3 system generally remained below 50% for their entire durations, with some erratic spikes at the beginning and the end of each data set as notable exceptions. The results displayed a significant degree of instability with a tendency to remain within certain percentage ranges: For example, as seen in Figures 27 and 28, the total CPU usage fluctuated between near-zero and 30% for much of the Unreal Engine test duration. The 4K render also showed similar results, albeit with an upper limit of approximately 35-50% (see Figure 29). Table 11 shows that the average CPU usage of Unity rendering processes sat below 20%, while Unreal Engine processes used approximately 6% more.

As illustrated in Figures 27 and 28, the computing power used for the Unity 1080p and 2K renderings remained below 20% except for a single brief jump during the 2K render. For the 4K render, Unity's CPU usage was noticeably higher,

averaging approximately 33% with a momentary maximum of over 50% on several occasions. For Unreal Engine, CPU usage in the 4K render varied between 10 and 50%, with short but frequent drops to 0%.

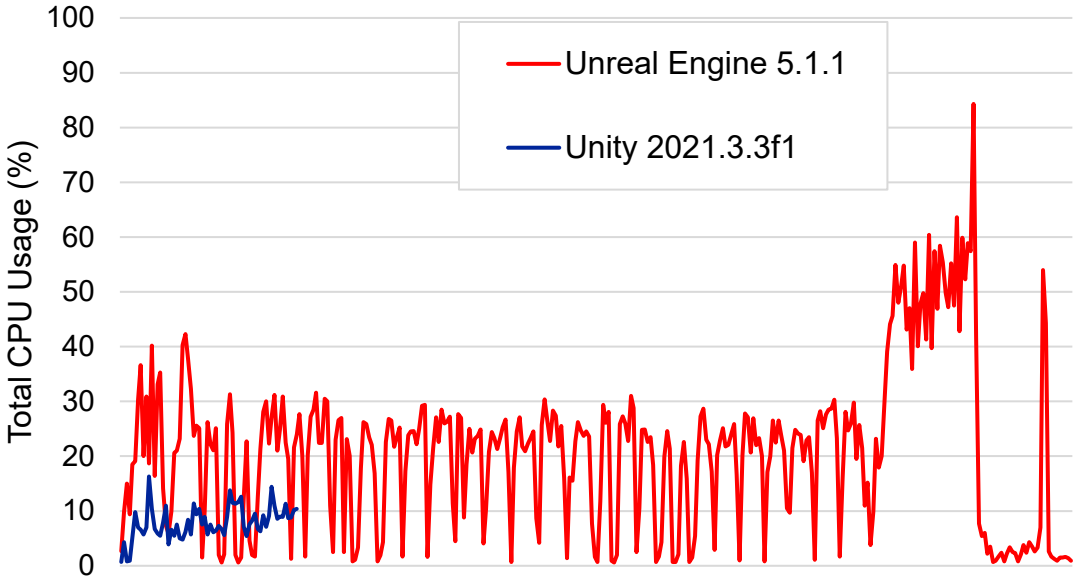

Figure 27. PC3: Total CPU usage over the duration of the 1080p/24 FPS rendering task.

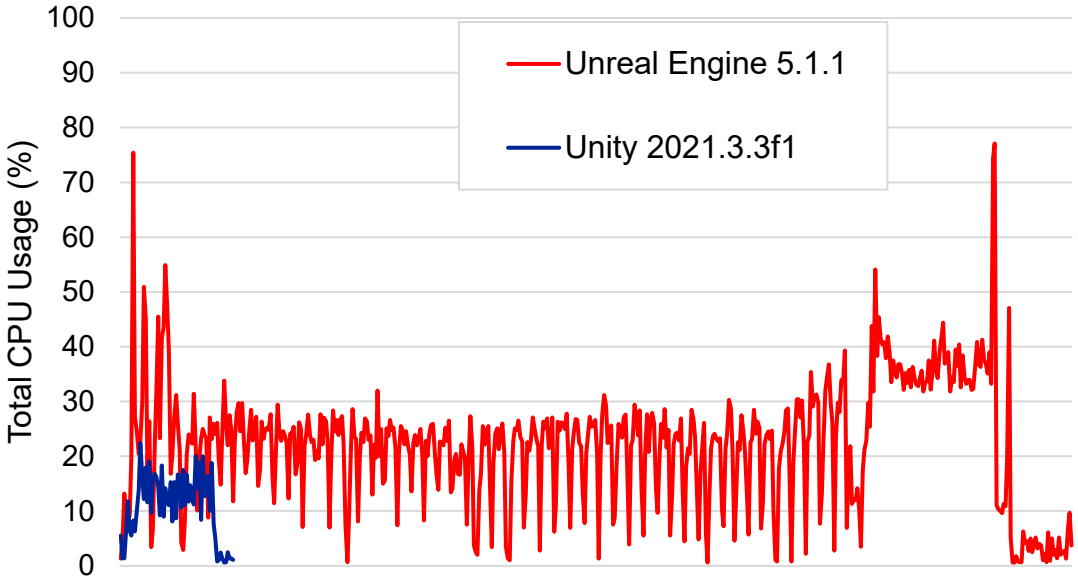

Figure 28. PC3: Total CPU usage over the duration of the 2K/30 FPS rendering task.

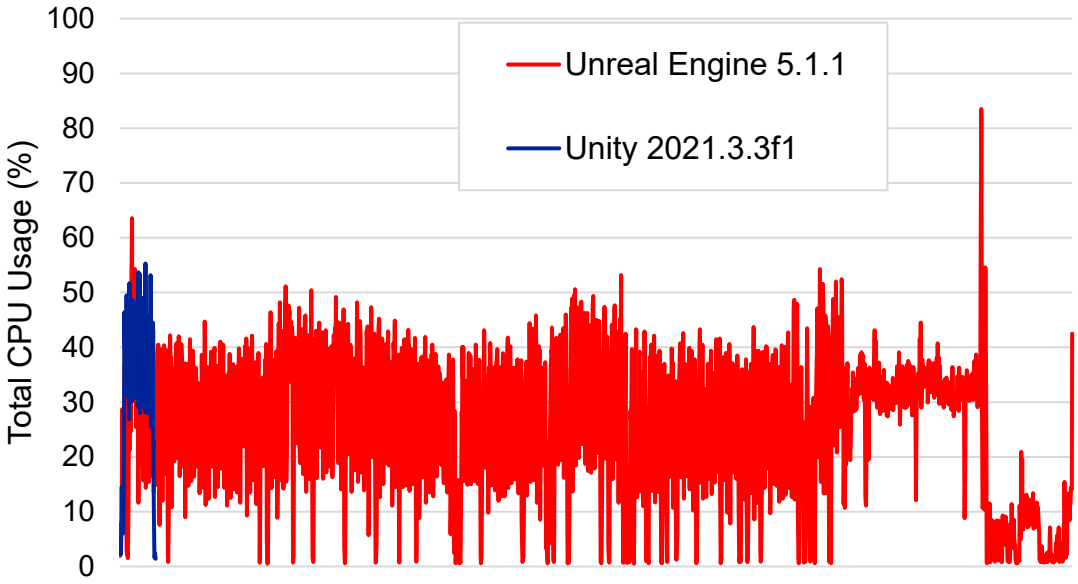

Figure 29. PC3: Total CPU usage over the duration of the 4K/60 FPS rendering task.

Table 11. PC3: Average total CPU usage in all test cases.

	Unreal Engine 5.1.1	**Unity 2021.3.3f1**
1080p/24 FPS	20.89%	7.90%
2K/30 FPS	22.16%	10.78%
4K/60 FPS	26.78%	33.36%
Combined average	23.28%	17.34%

As seen in Figures 30, 31 and 32, GPU core load figures for the PC3 system display a high degree of instability throughout all test cases. While percentages above 95% were consistently reached in all Unreal Engine tests, the end result only amounted to a usage average of approximately 63%. Figures 30, 31 and 32 show near-constant fluctuation between optimal (100%) and minimum (0%) GPU core load in Unreal Engine, with only the 4K render reaching a full 100%. All three cases exhibit the same behaviour, with a brief low core load at first followed by hefty jumping from one extreme to the other for the majority of the rendering process. Towards the end, Unreal Engine's GPU core load settles between

approximately 30 and 40% and then has a short sequence of fluctuations before the end of the data sets.

Unity data sets show more moderate GPU core loads with momentary percentages reaching approximately 95% during the 1080p and 2K renders (see Figures 30 and 31). The 4K render produces more consistent variation between around 40 and 55% over most of its duration. On average, Unreal Engine renders exhibited an approximately 28% higher GPU core load than their Unity counterparts (see Table 12).

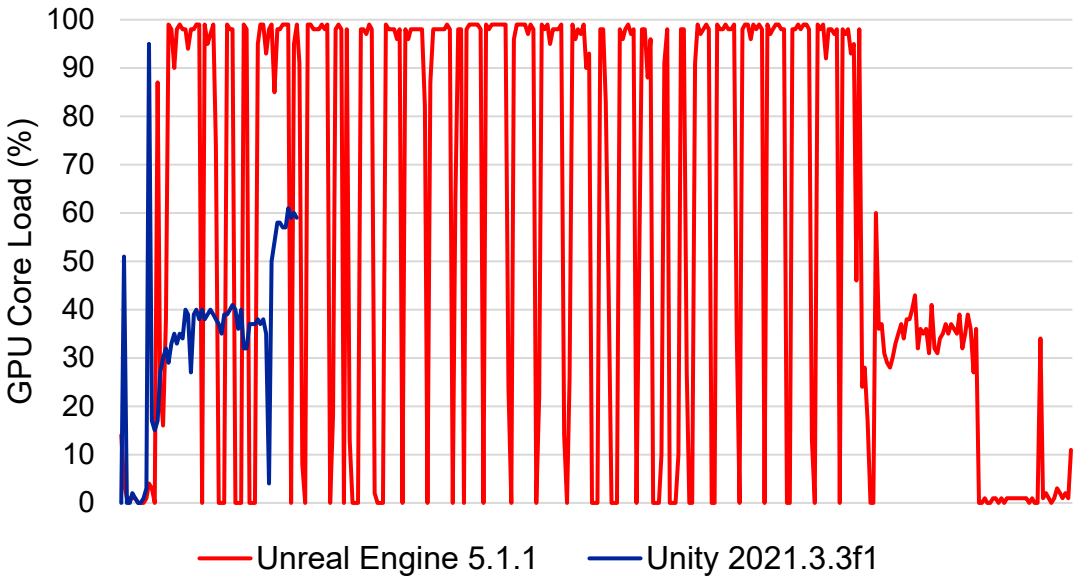

Figure 30. PC3: GPU core load over the duration of the 1080p/24 FPS rendering task.

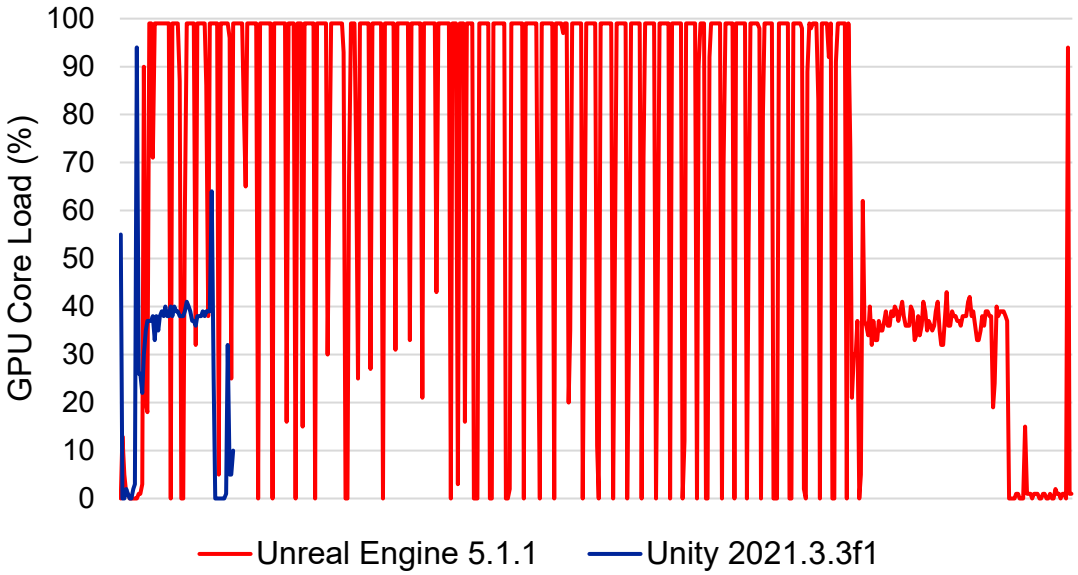

Figure 31. PC3: GPU core load over the duration of the 2K/30 FPS rendering task.

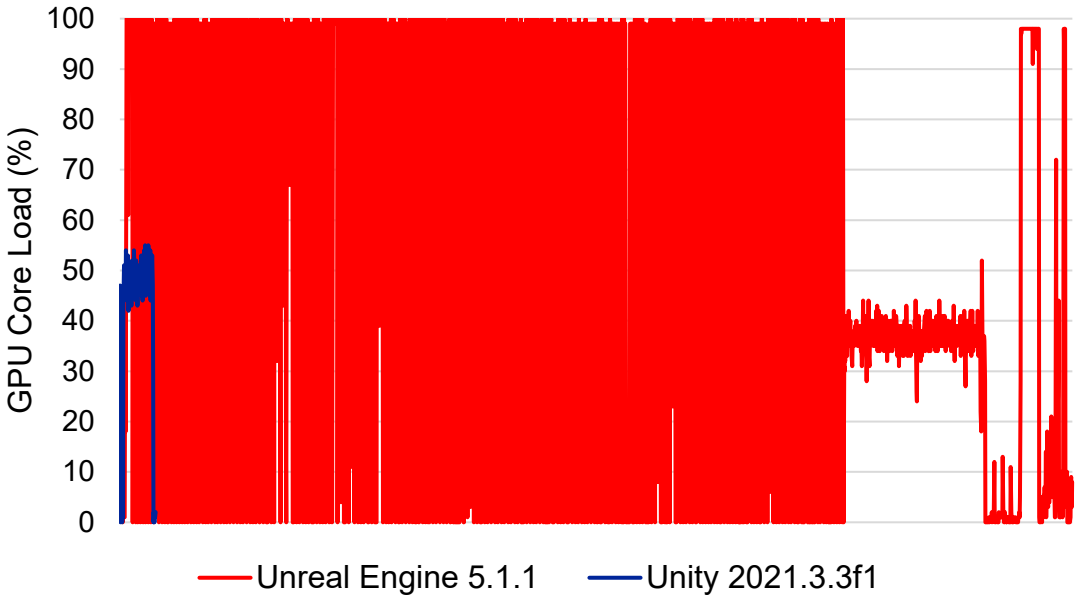

Figure 32. PC3: GPU core load over the duration of the 4K/60 FPS rendering task.

Table 12. PC3: Average GPU core load in all test cases.

	Unreal Engine 5.1.1	**Unity 2021.3.3f1**
1080p/24 FPS	57.58%	34.28%
2K/30 FPS	66.51%	28.30%
4K/60 FPS	63.94%	41.31%
Combined average	62.67%	34.63%

6 Discussion

6.1 Relevance of results

As noted in Šmíd's comparison between Unity and Unreal Engine (2017), the render engine is arguably the most important part of a game engine. Good graphical quality continues to be an important selling point for games, as one of the most commonly rated aspects of any game is its presentation. It is not unusual that over 90% of calculations done during gameplay are related to rendering only, while the remaining 10% is shared by game logic, scripts, physics and audio. Consumers' expectations for better visual quality continue to grow year by year, and catering for this demand requires developers to take also lower-end machines into consideration when choosing a game engine for their projects. Factors such as these contextualise the research done in this thesis, as its aim is to compare Unity and Unreal Engine in terms of their performance when rendering animated sequences at various frame rates and resolutions.

6.2 General findings

The overall results of the testing process were largely consistent with the unique specifications of the different computers used in the practical phase of the project. In some cases, the features of each system could be quickly recognised from the numerical data they provided. However, patterns were only found in individual data sets rather than the assessed metrics as a whole, which meant that they had to be analysed one by one and compared only to other measurements of the same kind.

Workflow-wise, neither Unity nor Unreal Engine presented significant challenges in getting the USD attic scene to show up properly in their workspaces, though the experimental state of Unity's USD handling add-on and the lack of official documentation meant that successful installation of the ready-made environment into the editor was achieved largely by trial and error. As seen in Figure 2 in

section 4.4, Unreal Engine provided more tools for the importing process, but these were ultimately not used in this research project because of the need to have identical backdrops for recording in both applications. Position and rotation values of the camera path's intermediate locations were easier to set and modify in Unreal Engine thanks to the intuitive design of the Sequencer tool interface. Moreover, even though Unreal Engine had more user interface (UI) elements to navigate through in order to produce the animated sequence, the process in its entirety was less vague and confusing than in Unity.

6.3 Rendering time and frame creation rate

Out of all unique data sets, rendering times and average frame creation rates were the most readily available since they did not depend on the performance of each evaluated computer. As seen in Figure 7 in section 5.2.1, the time taken to finish the rendering task in Unreal Engine diminished considerably with each upward step in system computing power. Meanwhile, rendering times in Unity did not exhibit such drastic changes: Tables 2, 6 and 10 show that none of the rendering tasks performed in that application ever took more than 56 seconds to finish. This points to much greater consistency in different hardware environments.

A recurring trend could be distinguished in that Unity required less time to produce more frames of the entire animated sequence per second with all individual setting pre-sets. It is likely that the difference is caused by several factors, including:

- the quality and exact parameters of the lighting settings in the workspace
- the way the rendering task is handled by the engines' recording tools
- the system performance requirements of the engine itself.

6.4 Physical memory usage

Physical memory usage trends were the least volatile out of all collected data sets. Not only did they exhibit very little instability but they were also the easiest to read and draw conclusions from. The usage data was only available for the PC1 and PC2 systems, as the .CSV files compiled as part of the rendering processes on the PC3 system only turned out 11 of the expected average of 345 columns of raw data. The available columns did not include physical memory usage measurements, which meant that they could not be analysed.

The data that was available from the other two computers showed that the physical memory usage of the PC1 system during Unreal Engine renders remained well below the maximum of 64 GB. However, the PC2 system exceeded its own maximum of 16 GB on multiple occasions, which assumably resulted in some of the computer's internal storage being harnessed as makeshift memory until the usage levels went down again. In the end, such memory overflows only occurred four times in total: once during the 1080p and 4K renders, and twice during the 2K render. Memory usage in Unity was both less significant and more stable, reflecting the application's more modest resource expenditure.

6.5 Total CPU usage

As no other programs besides Unity and Unreal Engine themselves and HWiNFO were left running during rendering, the expectation was that the resulting data would reflect the CPU usage of the game engines as purely as possible. While this was achieved for the most part, some unexpected behaviour was recorded, particularly during the 1080p rendering task on the PC1 system: Figure 12 (see section 5.2.2) shows 100% CPU usage during the initial stages of the Unreal Engine process. Reasons for this unforeseen reading are not known, as no other rendering tasks displayed similar occurrences.

Based on the analysis of the total CPU usage data, it can be concluded that the Unity rendering processes were less predictable than their Unreal Engine

counterparts. While they were completed much more quickly, their resource usage generally showed more uneven patterns than the same tests performed on Unreal Engine. Examples of this can be seen in Figures 27, 28 and 29 (see section 5.2.4): Unreal Engine results show a consistent pattern of low to medium CPU usage across all three test cases, whereas Unity results start from around 10% usage during the 1080p render and climb to between 10% and 20% in the 2K render and ultimately to 30-50% during the 4K render.

6.6 GPU core load

As a particular effort was made to dedicate as much as possible of the computing power of the test computers to the rendering tasks, the GPU core load percentages were expected to reach optimum levels above 90% either with few deviations or without them altogether. However, as seen in Tables 5, 9 and 12 (in sections 5.2.2, 5.2.3 and 5.2.4), the results were not as uniform as initially anticipated. On the contrary, they displayed considerable instability particularly throughout the Unreal Engine rendering processes run on the PC1 and PC3 systems. The results on the PC2 system were best in line with the original expectations, though they, too, showed some fluctuation.

The results obtained from the Unity rendering processes were not as surprising, given that the entire task was finished in under one minute in all cases. As could be expected, the combination of fast rendering and the lower output quality result in overall GPU core loads that did not reach the same heights as the Unreal Engine equivalent.

6.7 Anomalies

Several oddities were detected in the charts. For Unreal Engine, the PC2 system took more time to complete the 2K rendering task than the PC1 system even though performance and efficiency gains were to be expected with each step up when moving on to more powerful computers. This unexpected outcome may stem from the smaller size of the integrated RAM in the PC2 system, meaning that even if it used a newer and more powerful GPU, it was slowed down by the 16 GB of computer memory it could use to efficiently complete the rendering tasks as opposed to the 64 GB utilised by the other systems. This theory is supported by the fact that even with its less powerful GPU, the PC1 system presented no notable hardware-related issues during or after the rendering processes.

As mentioned in section 6.4, no physical memory usage statistics of the PC3 system were obtained. The reason for the significantly reduced amount of data that could be extracted from the rendering process could not be determined. The rendering time and frame creation rate measurements were not affected by this deficiency and could be used for acceptable analysis and conclusions.

6.8 Hypotheses

Figure 6 (see section 4.4) shows a clear difference in the environmental lighting applied to the USD attic scene in Unity and Unreal Engine. However, the stark difference in quality comes with the cost of significantly longer rendering times, as the average time taken to finish the tasks in Unreal Engine was up to 31 times longer than in Unity. Additionally, despite some individual deviations, Unreal Engine’s average computer resource usage was higher in all test cases, confirming the first of four hypotheses presented in section 4.1.

Both the observed consistency of rendering times in Unity and the generally lower average system resource usage in all test cases support the second hypothesis. This, combined with the lesser graphical detail of Unity results, implies that

rendering with a high frame rate and resolution has less net impact on system performance in Unity because of its lower hardware requirements.

The third hypothesis could not be confirmed with certainty. While Figures 7 and 8 point to the expected smaller average computer resource usage in higher-performance systems, other measurements were not as linear: For example, total CPU usage percentages on the PC2 system did not follow the aforementioned pattern, with both Unity and Unreal Engine averaging more than 32% usage as opposed to under 28% in all other cases on the PC1 and PC3 systems.

In section 6.2 above, mention is made of the difficulty with which the USD attic sample asset could be brought into working order in Unity. This, coupled with Unreal Engine's more coherent UI layout as well as a greater choice of options, confirms the final hypothesis that Unreal Engine is better than Unity in terms of USD asset handling. Moreover, the greater efficiency with which Unreal Engine is able to handle large and detailed objects, such as the USD scene used in this comparison, puts it in a more favourable position in this context.

7 Conclusions and recommendations

The purpose of this thesis was to compare the performance of two game engines, Unity and Unreal Engine, during pre-set rendering processes so as to determine which application is more suited for rendering short animated sequences in real time. This was achieved by measuring and recording five metrics on three different computer systems, ensuring that the analysed data was as neutral as possible. Unreal Engine was expected to emerge as the superior alternative, particularly in terms of the visual quality of the animated sequences it produced on each computer using three different setting pre-sets.

The data sets obtained from a total of 18 different rendering processes indicate that Unreal Engine is indeed more viable for real-time rendering work, though the higher-quality results come at the cost of elevated computer resource usage and considerable investment in time needed for completing the rendering tasks. It should be noted that Unity results are by no means unacceptable and that they provide a viable alternative to Unreal Engine especially for lower-end machines. If more work is put into refining the visual details of the recorded environment, as well as the post-processing effects, the quality of animations produced with Unity can be brought on par with Unreal Engine's equivalent. In other words, while Unreal Engine has a time cost at the rendering stage, with Unity it is incurred at post-processing.

Additionally, it is advisable to wait for USD asset management to be officially incorporated into both of the evaluated game engines: The highly experimental nature of Unity's USD importing package, as well as its apparent lack of official documentation, posed significant hindrances during the preparations for data gathering for this project. Unreal Engine's USD Stage plugin was much more user-friendly and coherent. Therefore, until Unity's USD import package is developed further, Unreal Engine's equivalent remains the better option.

Given that this research was conducted on only three computers and that the length of the animations created using them was only 24 seconds, more research

is still needed to provide more in-depth points of comparison between the rendering characteristics of Unity and Unreal Engine.

References

3DHeven. 2023. Real-time and Offline 3D rendering: What are the Differences? - 3DHEVEN. Accessed 15.5.2024. Available at: https://3dheven.com/real-time-and-offline-3d-rendering-what-are-the-differences/

Adobe. n.d. Creating animated action with tweening. Accessed 30.4.24. Available at: https://www.adobe.com/creativecloud/video/discover/tweening.html

Christopoulou, E., Xinogalos, S. 2017. Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices. Int. J. Serious Games 4. Accessed 29.4.2023. Available at: http://journal.seriousgamessociety.org/index.php/IJSG/article/view/194

Computer Hope. 2021. What is a Plugin? Accessed 30.4.24. Available at: https://www.computerhope.com/jargon/p/plugin.htm

Computer Hope. 2024. What is an Add-on? Accessed 30.4.2024. Available at: https://www.computerhope.com/jargon/a/addon.htm

Cubix, 2022. 50% of all Mobile Games are Developed on Cross-Platform Engine, Unity. Cubix Blogs. Accessed 16.3.2024. Available at: https://www.cubix.co/blog/50-of-all-mobile-games-are-developed-on-cross-platform-engine-unity/

Davoxel, n.d. Unity vs. Unreal Engine: Choosing the Right Platform for Your Game. Accessed 28.4.2024.
Available at: https://www.davoxel.com/blog/unity-vs-unreal-engine-comparison

Demant, R. 2023. Comparing Unity vs Unreal for VR, MR or AR Development Projects. XR Bootcamp. Accessed 11.4.2024. Available at: https://medium.com/xrbootcamp/comparing-unity-vs-unreal-for-vr-mr-or-ar-development-projects-edbc4f93b21c

Garratt, W., Rithviik, S., Wang, F. 2023. Achieving Interoperability Between Gaming Engines by Utilizing Open Simulation Standards. 2023 Simulation Innovation Workshop (SIW). Simulation Interoperability Standards Organization - SISO. Accessed 20.4.2024.
Available at: http://bura.brunel.ac.uk/handle/2438/26270

Grand View Research. 2023. Video Game Market Size To Reach $583.69 Billion By 2030. Accessed 16.3.2024. Available at: https://www.grandviewresearch.com/press-release/global-video-game-market

Lenovo US. n.d.-a. Unlocking the Hidden Power of Physical Memory. Accessed 17.4.2024.
Available at: https://www.lenovo.com/us/en/glossary/physical-memory/

Lenovo US. n.d.-b. What is a Graphics Card? and Why Do I Need One? Accessed 30.4.24.
Available at: https://www.lenovo.com/us/en/glossary/what-is-graphics-card/

Luk, A. 2023. What You Need to Know About Universal Scene Description — From One of Its Founding Developers. Medium.com. Accessed 9.4.2024.
Available at: https://medium.com/@nvidiaomniverse/what-you-need-to-know-about-universal-scene-description-from-one-of-its-founding-developers-12625e99389a

MiniTool. 2022. Is 100% GPU Usage Bad or Good? How to Fix 100% GPU When Idle? Accessed 18.4.2024. Available at: https://www.minitool.com/news/100-gpu-usage.html

Run:ai. n.d. 6 Reasons for Low GPU Utilization and How to Improve It. Accessed 18.4.2024.
Available at: https://www.run.ai/guides/multi-gpu/low-gpu-utilization

Sacco, M. 2023. Unity: Understanding URP, HDRP, and Built-In Render Pipeline. Accessed 11.4.2024. Available at: https://www.occasoftware.com/blog/unity-understanding-urp-hdrp-built-in

Šmíd, A. 2017. Comparison of Unity and Unreal Engine. ČVUT/DCGI. Accessed 23.4.2024. Available at: https://dcgi.fel.cvut.cz/en/theses/2017/smidanto/

SolarWinds. n.d. What is CPU Usage? - IT Glossary. Accessed 17.4.2024.
Available at: https://www.solarwinds.com/resources/it-glossary/what-is-cpu

Unity Technologies. 2017a. Unity - Manual: GameObject. Accessed 30.4.2024.
Available at: https://docs.unity3d.com/560/Documentation/Manual/class-GameObject.html

Unity Technologies. 2017b. Unity - Manual: Post-processing overview. Accessed 30.4.2024. Available at: https://docs.unity3d.com/560/Documentation/Manual/PostProcessingOverview.html

Unity Technologies. 2024. Unity - Manual: Render pipelines. Accessed 11.4.2024. Available at: https://docs.unity3d.com/Manual/render-pipelines.html

Unity Technologies. n.d. Understand Real-Time Rendering In Both 3D & 2D with Unity. Accessed 15.5.2024. Available at: https://unity.com/how-to/real-time-rendering-3d

Unreal Engine 5.1 Documentation. 2022. USD Stage panel. Accessed 29.4.2024. Available at: https://d1iv7db44yhgxn.cloudfront.net/documentation/images/f5b95820-5e29-4acc-8fad-f7b4768dae7a/5-0-010-usd-stage-panel.png

Wijman, T. 2018. Newzoo's 2018 Report: Insights Into the $137.9 Billion Global Games Market. Accessed 15.4.2024. Available at: https://newzoo.com/resources/blog/newzoos-2018-report-insights-into-the-137-9-billion-global-games-market

XR suite. 2021. Real-time and offline 3D rendering: what are the differences? XR suite - by L&S. Accessed 15.5.2024. Available at: https://www.xrsuite.fr/post/real-time-and-offline-3d-rendering-what-are-the-differences

Young, C. 2021. 7. Unity Production: Capturing the Everyday Game Maker Market. Sotamaa, O., Svelch, J. (Eds.), Game Production Studies. Amsterdam University Press, pp. 141–158. Accessed 15.4.2024. Available at: https://doi.org/10.1515/9789048551736-009