Bachelor of Science in Digital Game Development
June 2022

Visual and performance comparison: Physically based atmosphere method against Unity Universal Render Pipeline procedural skybox

Martin Höglund
Nikki Norberg

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Bachelor of Science in Digital Game Development. The thesis is equivalent to 10 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author(s):
Martin Höglund
E-mail: mahh18@student.bth.se

Nikki Norberg
E-mail: nino19@student.bth.se

University advisor:
Associate Professor Prashant Goswami
Department of Computer ScienceE-mail: prg@bth.se

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Background. In the pursuit of realism in rendering, a feature that may get overlooked is the sky and atmosphere—especially the consumer who might suspect that something is off without identifying it. A critical aspect of sky rendering is the color that should be dynamic with the time of day to be realistic. With multiple ways of rendering a sky and atmosphere, it is unclear which method best suits the circumstance.
Objectives. The objective of this study was to make that decision clearer with both the visual aspect and the realism. It was also crucial that performance was not affected too heavily by the techniques.
Methods. As the first objective can be subjective, we decided to ask several people through a survey to get a more objective answer. In the survey, we presented the participants with two images of identical scenes where the only difference between them was the method of rendering the sky. We then asked them which picture they thought was the most aesthetically pleasing and which they thought was the most realistic. The second part of the study was to test the performance of the techniques. We recorded the average frame time through a C# script over all of the scenes.
Results. When using the settings we established, the Physically-based atmosphere scored lower in both realism and aesthetics in the first visual test but scored higher in realism in all of the scenarios for the second visual test. Both the average frame rate and frame time did end up in favor of the Physically based atmosphere.
Conclusions. The outcome of this study was surprising, we had thought that the Physically-based atmosphere would score higher in both realism and aesthetics. The performance of the Physically based atmosphere method proved to be much more optimized than the Unity procedural skybox than we would have thought.

Keywords: Unity, Physically based atmosphere, aesthetics, realism.

Acknowledgments

We would like to thank all the participants that took their time to take part in answering our survey.

Contents

Glossary

2D Term used to describe a two-dimensional space. 1

3D Term used to describe a three-dimensional space. 10

CPU Hardware responsible for executing programs, stands for Central Processing Unit. 15

deferred rendering A method of rasterization where the lighting calculations are delayed or deferred until all geometry is rendered. 3

forward rendering A method of rasterization where every object is rendered in one or more passes depending on lights that affect that object. 3

GPU Hardware responsible for creating and displaying graphics, stands for Graphics Processing Unit. 1

path-tracing A form of ray-tracing where more rays are sent through every pixel to get a more accurate result. 3

physically based atmosphere A simulation of how light gets affected by the atmosphere. 3

rasterization A rendering technique where geometry is projected to a 2D plane and then determine if the given pixel lies within the geometry. 3

ray-tracing A rendering technique where rays are sent through every pixel to determine if any geometry was hit. 3

shader A programmable step in the graphics pipeline with instructions for the GPU. 3

skybox A cube that moves with the camera and displays the sky. 3

Chapter 1

Introduction

Realism in games regarding realistic lighting, physics, surfaces, and sounds has been a goal and something to strive for forever since the concept and creation of the first digital games. Every year new game-changing revolutions in-game creation surface, and new methods and techniques to render even more realistic results in an acceptable frame rate are developed. There have been some advances throughout the meaningful time regarding the relationship between surface and light rendering.

Today there are some free-to-use game engines like Unreal and Unity [5, 19]. The latter has some choices regarding rendering methods and pipelines. Unity's default (built-in) pipeline lets users choose between forward rendering and deferred rendering. In later versions of Unity, they provide their Universal Render Pipeline (URP) [16] which is a pre-built script-able alternative to the built-in one. Unity also provides the high definition render pipeline (HDRP) [17] which is a high-fidelity script-able pipeline that lets users create compute shaders. There are several methods one can use when rendering a frame. Some mentioned previously, like forward and deferred rendering, are examples of rasterization, most commonly used in real-time rendering. The most commonly used rendering method for offline rendering is ray-tracing or path-tracing which often is called unbiased rendering [1, 3, 10]. These rendering techniques provides a more realistic result, but the trade-off is that each frame takes longer to render. Recently there has been a push to use ray-tracing combined with rasterization in real-time rendering for specific cases, e.g., reflections.

One aspect of realistic rendering that's often overlooked is the rendering of the sky and atmosphere. One of the most used techniques for rendering a sky is the use of a skybox which can give a high definition rendering of the sky and does not impact the performance as heavily [4]. An alternative to a skybox is to do an atmosphere simulation. While impacting the performance more than a skybox, it may also provide a more realistic rendition of the sky. In their paper from 2020, Hillaire [6] suggested an implementation of an atmosphere rendering technique for real-time rendering that later was made part of the Unreal Engine 4. With Unity, an alternative option exists of using either a regular skybox or a physically based atmosphere simulation. This option is the procedural skybox that effectively works as a combination of the two previous techniques. The simulation is baked into a skybox that combats one of the drawbacks of using a skybox where it is static and does not allow for a dynamic time of day. Many of the methods used to render a physically accurate atmosphere and sky are adapted for offline ray-tracing or path-tracing rendering [7]. However, there exist some methods that are adapted to work with real-time rendering [21]. Those methods use approximations to achieve the

fast collation times that real-time rendering requires, but that also means that they will not achieve a level of physical accuracy that the offline rendering methods do. We chose the Hillaire 2020 method because it achieves a closer physical accuracy than other real-time rendering solutions while still using approximations to achieve a faster frame-time.

1.1 Aim and objectives

Multiple techniques to render a sky exist, and choosing which one to utilize is not always obvious. We hope to make the decision clearer when creating a realistic and dynamic world with this paper. However, this increase in realism can not come with too much cost to the performance. The scope of this project does not allow us to compare every method of sky rendering. Therefore only the procedural skybox and the technique presented by Hillaire will be used. Both of these methods are optimized to work with a wide variety of hardware. This means that the result of this comparison applies to a variety of projects.

1.2 Research questions

The following questions will be answered by the end of this paper:

- RQ 1: How does a physically based atmosphere simulation when implemented into Unity Universal Render Pipeline stand compared to the Unity procedural skybox in the same pipeline regarding aesthetics and realism?
- RQ 2: How does a physically based atmosphere simulation regarding frame rate and frame time performance compare to the frame rate and frame time performance of Unity procedural skybox?

1.3 Hypothesis and Expected Outcome

We reckon that since the method by Hillaire is in comparison to a traditional skybox a much more complicated algorithm the result of the performance tests will prove in favor of Unity's procedural skybox. The method by Hillaire is a physically-based atmosphere. We reckon that this will prove to give a much more realistic outcome when rendering scene images than Unity's skybox would. We also figure the aesthetic outcome to prove in favor of Hillaires method since beauty often comes with realism in terms of lighting, particles, and sky rendering.

1.4 Document layout

Related Work - chapter 2
The section on related works looks at several important methods for this study. We take a look at the different pipelines in Unity, their capabilities, and different methods related to the Hillaire method this study is oriented around.

Method - chapter 3
In the method section, how we conducted our survey for data collection of thoughts on realism and aesthetics of comparable Unity scenes using the Unity procedural skybox and Hillaires atmosphere method. We also look at the conduction of our performance tests.
Results - chapter 4
In this section, we present the data from the survey and the result of the performance tests.
Analysis - chapter 5
An objective analysis of given results from the survey and the performance tests, noting results that stick out.
Discussion - chapter 6
Here we present thoughts on the results and analysis and what may have affected the results to come out the way they did.
Conclusion - chapter 7
In this section, we answer the research questions and a conclusion of the whole project.
Future work - chapter 8
We give our thoughts on future works using the Hillaire method for Unity.

Chapter 2

Related work

In the Unity engine, there exist multiple ways of simulating a sky, primarily either the Physically Based Sky [13] that is included with the Sky and Fog volume, or you could use the Procedural Skybox [14] [4]. The Render Pipeline in use dictates which of these techniques can be used. The HDRP [17] is the only pipeline that can make use of the volume framework, which in turn makes HDRP the only supported pipeline for the Physically Based Sky. For the Procedural Sky, both the Built-in Render Pipeline and the URP [16] is supported, but the HDRP is unsupported. The Physically Based Sky is a practical implementation of the paper by Bruneton and Neyret [2].

Alternatively, to use one of the tools already made for Unity, one could implement their own, which Lague did in his video "Coding Adventure: Atmosphere" [8]. Lague later improved on this atmosphere in another video [9] with the ideas presented by Hillaire in his paper [6]. The paper from Bruneton and Neyret, and Hillaire proposes storing the pre-calculations in 4D tables. The calculations for atmosphere and sky rendering are rather computationally expensive for real-time rendering, as it usually uses some form of ray-marching. Therefore storing them in LUTs can give a performance increase. Because these look-up tables or LUTs are in four dimensions, they snowball in size with increased resolution. Therefore there needs to be a low resolution to conserve memory. As the sky changes color gradually, it can be effectively up-scaled, and the sun disk can be rendered on top as a separate object, making the small size of the LUTs, not an issue.

Chapter 3

Method

3.1 Implementation

The method by Hillaire can be found free for noncommercial use on GitHub and can therefore be cloned for us to test [11].

3.1.1 Plugin implementation and testing scene

We had initially planned on converting the method presented by Hillaire as a plugin for the Unity Engine to utilize it in future projects. Unity provides a sample project for creating plugins that we used. The plugin provides a setup for rendering with different APIs such as DirectX 11, DirectX 12, Open GL, Metal, and Vulkan. In our project, we would only use DirectX 11. The example plugin comes in different versions, so it is vital to use the latest one for most new and improved functions to work.

By the time we attempted to create the plugin for the first time, we had compiled shaders into bit shader code which created bit arrays in a simple header file to represent the shaders. These header files would allow us to import the shaders into Unity for later use. The problem with creating bit arrays for arrays is the potential size of the file. A small shader may render an acceptable bit array size. However, since bit arrays are much more significant in terms of the number of signs needed to present the same information as a standard shader would, the array grows substantively with every line of code. While the files created would be huge, the files would also be permanently taking space in the .dll file, always to be accessed. The fact that the file is baked into the .dll file and can always be accessed can be very convenient, but it does create an overflow in memory usage as the shader file does not always need to be accessed, only sometimes. A much more memory-friendly solution is using .cso files to represent the shaders in Unity. The renderer will only read these files when they are needed. The use of .cso files would ramp down the memory usage and compiling time. However, as these files .cso files can not be baked into the .dll file, they would have to be moved to a specified location to work. A workaround for this problem is to compile the shaders in run-time. That way, they can be baked into the .dll file.

When we tried to import our plugin into Unity, some unknown errors occurred, which we suspect were due to the shader compilation. Since the time was quite limited for this project, we did not look into it any further and abandoned the idea of implementing a plugin. After some research, we found that implementation of the

method had already been done for Unity. Laque created this implementation in one of his videos [9] and also distributed the Unity project. After some modifications to the scene used by Laque in his video, leaving us with only the atmosphere, we had a scene where we could import the test assets.

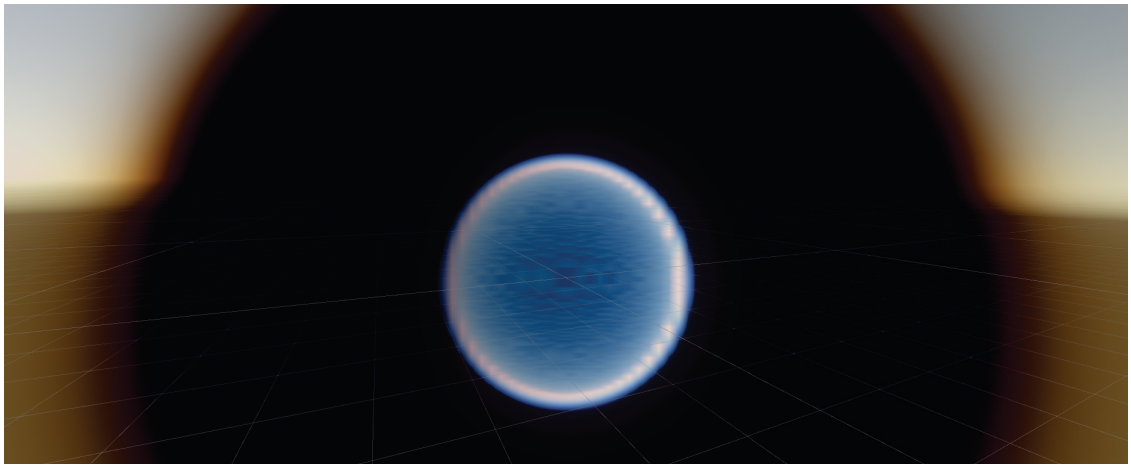


Figure 3.1: Hillaire method, Atmosphere as seen from far away

3.2 Survey

The method we decided to use for the first research question "How does a physically based atmosphere simulation when implemented into Unity Universal Render Pipeline stand compared to the Unity procedural skybox in the same pipeline regarding aesthetics and realism?" was a user study with a survey, gathering viewers' information about their thoughts on aesthetics and realism regarding the two methods mentioned in the questions. We chose this method to create an as isolated environment as one could in an online test given at a distance. This method of giving out a survey online often generates more answers than if the survey was given in a set place. The alternative was a focus group in which a group of participants would sit together and pictures would be shown to them, much like we did in our chosen method for them to then discuss. This method, however, would maybe prove to participants being influenced by each other and not giving sincere answers.

The methods by Hillaire create an atmosphere around the scene, and Unity's procedural skybox is generated from given parameters. The survey contains two pictures per question which show two identical 3D scenes set up in Unity where the only difference was the sky rendering method. The position of the sun was identical as well as the camera position. We chose to render three different pictures per scene at three different times per picture. Giving each Hillaire rendered picture a corresponding Unity skybox rendered picture with the same sun position (time of say) to be compared with. We ended up with twelve pictures, with six comparisons done in the survey. Each picture in each comparison was given a number, 1 and 2, for the participants to name in the questions below. The questions asked under each comparison were:

- Which picture is more aesthetically pleasing?
- Which picture do you feel is more realistically accurate?

Each question was followed by the name of the scene and the time of day evaluated by the position of the sun. The participants were not informed about what picture was rendered using what method. The pictures were set to come one after another in a stacked order followed by the questioners underneath them. The scenes where Hillaires method was used were always shown before the method where Unity's method was used.

3.2.1 Participants

The participants are of any age, background, and sex. We chose not to aim our survey toward any particular group of people because of the nature of the questions. To answer the questions, one does not need to have any experience in any specified kind of field. The survey was conducted online through a Google Form, so the participants could be located anywhere as we did not supervise the participants. We aimed to get at least twenty participants to get a variety in who answered and get an objective answer. We distributed the survey through various Discord servers, which netted twenty-one participants for the survey.

All information collected was anonymous and could not be connected in any way to the participant. The data collected were not of sensitive nature, therefore we found the free choice of the participant answering and being able to leave the survey without sending it in or giving a reason, consent enough.

3.2.2 Scenes

The scenes needed to show outside scenarios as the methods are the most visible in the sky and how the light falls on faraway objects like trees and maintain objects closer to the camera. As the sun plays a significant role in the setting of the scenes making the sky different colors at different times of the day, we found this effect would be most visible without roofs or other objects to cover it. We chose to use forest scenes for our experiments where trees, grass, and bushes are the leading contenders for both scenes. We came to call them the Orange Forest and the Green Forest, as they are visually set at different times of the year, one in fall and one in spring. The Orange Forest has a lake running through it with some hills in the background. The Green Forest takes place on a hill, and a path can be seen laid out up the hill with a fence running along the path. The scenes are relatively simple regarding polygon usage. Most trees in the Orange Forest are mathematically generated from a script, while the other trees use minimal polygons, as trees in a scene would in a game. A terrain object was used to draw trees, grass, and bushes on which uses instancing, further minimizing the performance cost. All the objects in the scenes were found on the Unity asset store, free to use [18].

3.2.3 The sun

The sun is a directional light with a sun disc connected to it. The disc does move around in the sky as the directional light rotation is changed and so does the color of the sky for both methods. The same directional light was used for both methods, and so was the direction.

Time of day	**Green Forest**	**Orange Forest**
Daytime	90, 0, 0	90, 230, 0
Afternoon	150, 0, 0	150, 230, 0
Evening	170, 0, 0	170, 230, 0

Table 3.1: Sun rotation for each scene

The rotations used were chosen purely due to the effect they had on the scene. We found these rotations to give the most variation in scene lighting.

3.2.4 Settings

The two methods use different techniques and therefore use different algorithms and variables that are used in different ways. Two variables that might have the same name for the two methods do not necessarily use the variables the same way in code.

Variable / Category	**Variable**	**Value**
Body Radius		255
Atmosphere Thickness		110
Rayleigh Scattering	Wavelengths RGB: Wavelength Scale: Rayleigh Sendity Avg:	639.5, 526, 441.8 748.5 0.086
Mei Scattering	Mei Density Avg: Mei Coefficient: Mei Absorption:	0.08 0.38 0.11
Ozone	Ozone Peak Density Altitude: Ozone Density Falloff: Ozone Strength: Ozone Absorption:	0.12 3.7 0.4 -3, 3.12, 0.02
Sun Disc	Sun Disc Size: Sun Disc Blur A: Sun Disc Blur B:	0.8 32.95 15
Transmittance LUT Size		32
Areal Perspective Strength:		0.529
Sky Texture	Num Sky Scattering Steps: Sky Render Size: Sky Transmittance Weight:	256 128, 256 0.429
Tone Mapping	Intensity: Contrast: White Point:	1.45 1.45 1.45

Table 3.2: Hillair Method settings used for renders used in survey.

These values are the values used originally by the unity implementation and were not changed.

Variable / Category	**Value**
Sun Size	0.04
Atmosphere Thickness	1
Exposure	1.3
Sky Tint	#808080
Ground	#5E5957

Table 3.3: The default Unity Procedural Skybox settings

These values are the default values that can be found on the default procedural skybox used in the Unity Universal Render Pipeline.

3.3 Performance tests

For this test, we found the most accurate results would come from data collection of precise information given by the Unity scene. No other research method would make sense for this test.

This test was used to answer the section research question "How does a physically based atmosphere simulation regarding frame rate and frame time performance compare to the frame rate and frame time performance of Unity procedural skybox".

We created four different scenes in Unity for frame rate and frame time tests—two scenes with the Orange Forest assets and two scenes with the Green Forest assets. We created two scenes with the same assets to create an isolated environment for the methods to be tested in as we could but fully separate the methods to their scenes. A C# script was created and used to measure frame rate and frame time. The script measures performance over a set time and gives results for the lowest recorded frames per second, highest recorded frames per second, and the average frames per second over this set time. It also records delta time, meaning the time it takes to draw a single frame [20]. This was recorded much like the frames per second: lowest recorded delta time, highest recorded delta time, as well as the average delta time. It is known that spikes may occur in rendering, resulting in meager frame rates and frame times and very high rates of frame rates and frame times. These things are usually not noticeable to the user. The performance tests were conducted on an AMD Ryzen 9 5950x CPU and an Nvidia RTX 3080Ti GPU. No additional culling was used other than the default frustum culling present in Unity and the rendering technique used was forward rendering.

3.3.1 Camera movement

A camera path was created for the camera to travel while conducting the tests, as this would represent more realistic results than if the camera had been stationary for the whole test. We created the path by using a tool that can be found in the Unity asset store [12]. The tool can easily be used by placing waypoints that create paths between the points. The paths turns can be manipulated using the beziér handles attached to each waypoint. We could choose how fast the camera would travel down the camera path and how long the camera would stay on the waypoints. As calculations including camera movement, where how the light falls against objects in relation to the camera position and direction are a big part of game frame rendering, we thought this to be an essential aspect of the testing. We chose a camera path over us moving around the camera in the scene using a character controller to get an accurate representation of the two performances. Using a controller instead of a predetermined path could not guarantee the same movement for both scenes, which would generate unfair results.

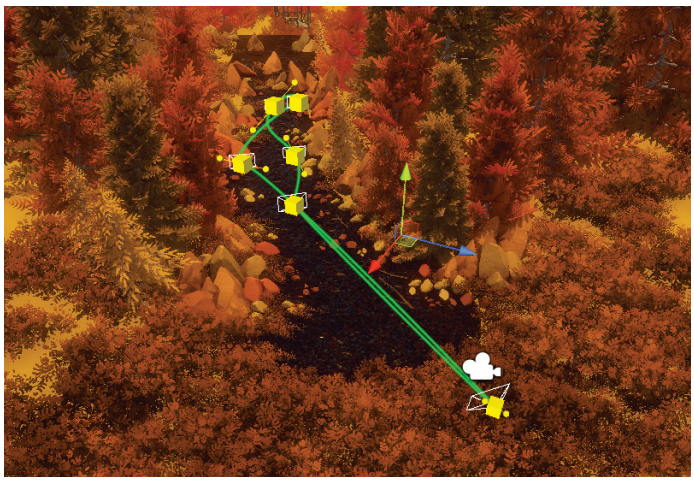


Figure 3.2: Camera path, Orange Forest

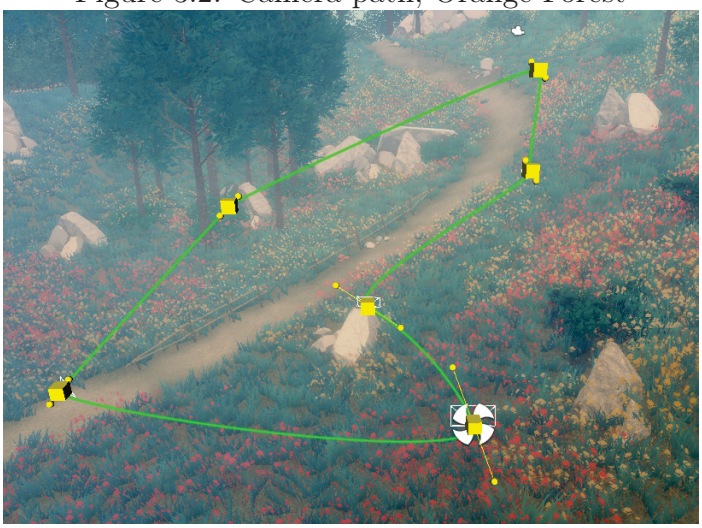


Figure 3.3: Camera path, Green Forest

3.3.2 Visuals

Games rarely only display plain geometry with textures on them. Therefore we chose to implement other techniques to make the scene more realistically accurate to a typical game scene. Unity does give users the option of using wind zones. These zones made the trees, grass, and bushes sway in the wind, something often seen in outdoor game scenes. For the water in the Orange Forest, we created a simple UV panning script to make the water's normal map scroll over the surface of the water, an easy technique often used for simple water in-game scenes. We also added a reflection probe to make the water reflect its surroundings. A reflection probe is a volume one can place around or place objects in to make them generate a cube map for reflection [15]. We added a post-processing effect script for a more pleasing visual outcome, much like any game designer would.

Chapter 4

Results

4.1 Result from survey

As mentioned, the participants who took part in the survey were given two pictures to compare each other featuring identical scenes, camera angles, and time of day. The only difference is the sky rendering method. Participants then got two questions related to the pictures to answer.

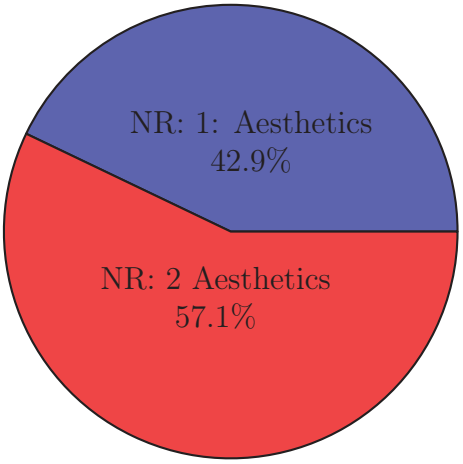
- Which picture is more aesthetically pleasing?
- Which picture do you feel is more realistically accurate?

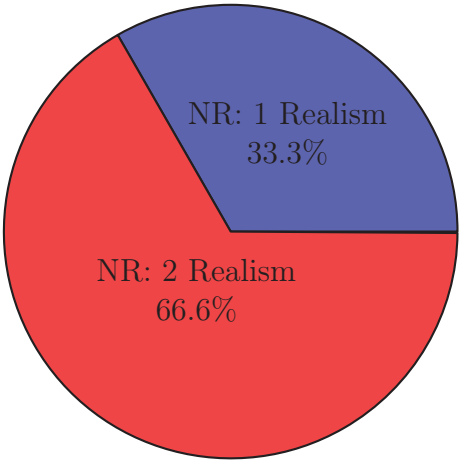
We gathered twenty-one participants to answer our questions. Below we display each comparison along with a description of each picture with the picture number and the results from the survey as pie charts.

4.1.1 Green forest, daytime comparison

Figure 4.1: Hillaire method, Green Forest, Daytime, NR: 1

Figure 4.2: Unity method, Green Forest, Daytime, NR: 2

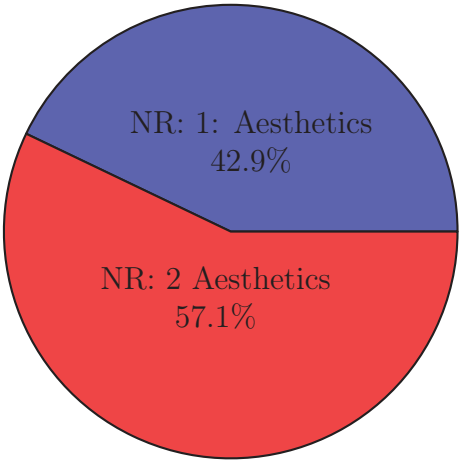


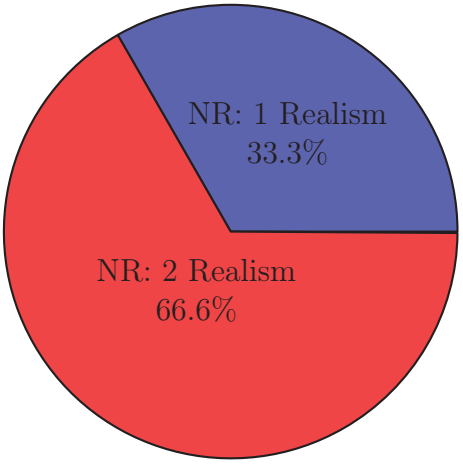


4.1.2 Green forest, Afternoon comparison

Figure 4.3: Hillaire method, Green Forest, Afternoon, NR: 1

Figure 4.4: Unity method, Green Forest, Afternoon, NR: 2

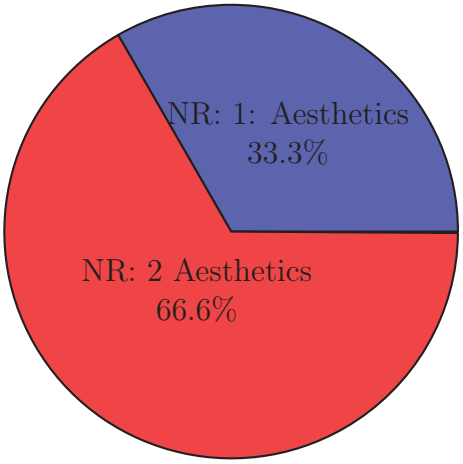


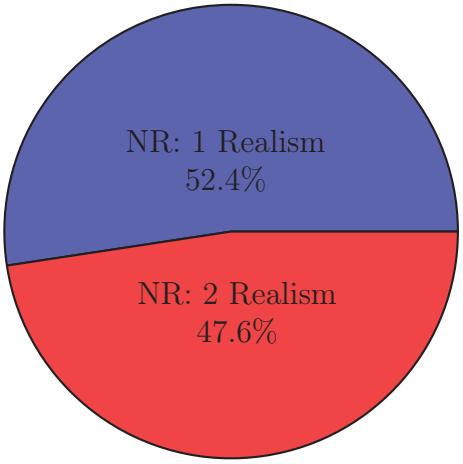


4.1.3 Green forest, Evening comparison

Figure 4.5: Hillaire method, Green Forest, Evening, NR: 1

Figure 4.6: Unity method, Green Forest, Evening, NR: 2





4.1.4 Orange Forest, Daytime comparison

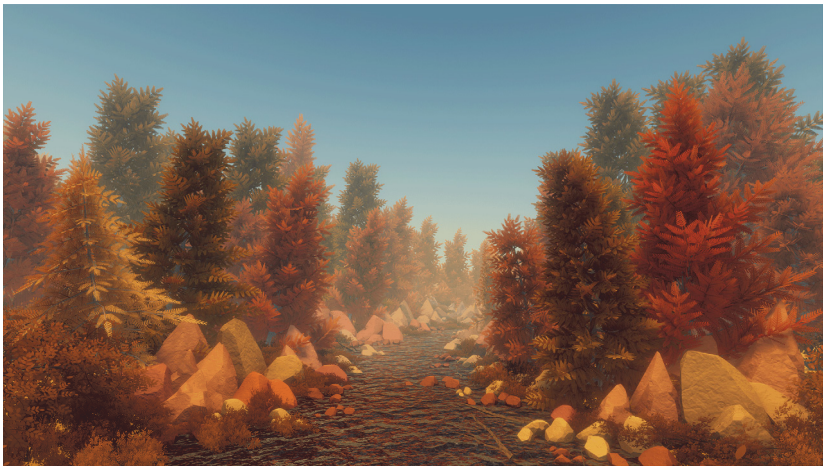


Figure 4.7: Hillaire method, Orange Forest, Daytime, NR: 1

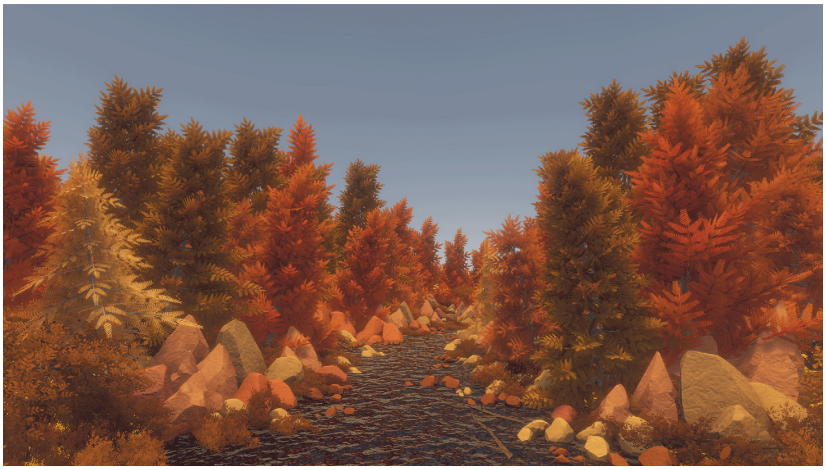
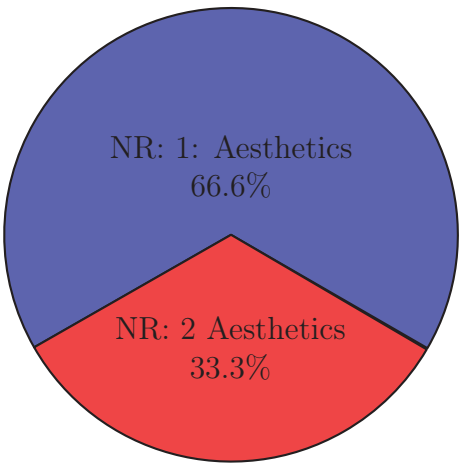
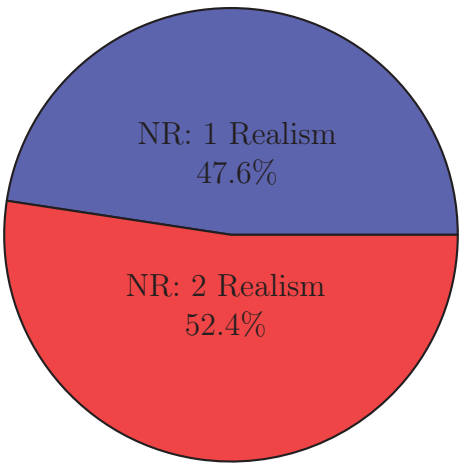


Figure 4.8: Unity method, Orange Forest, Daytime, NR: 2

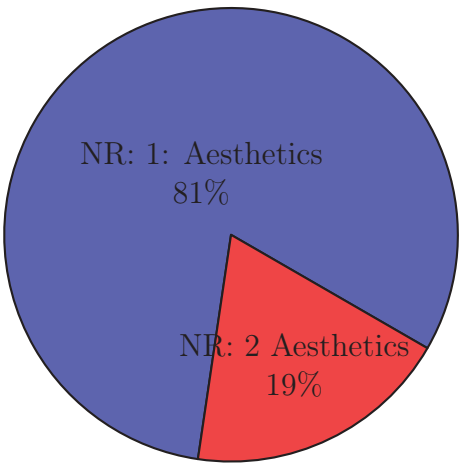


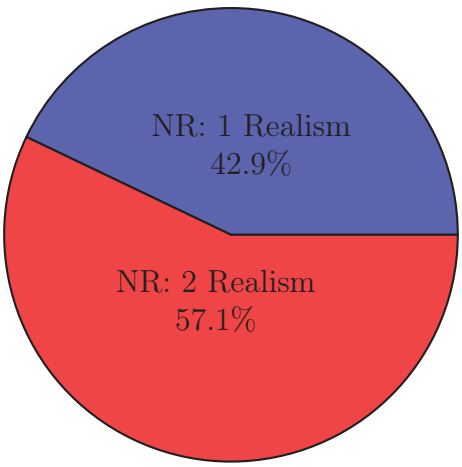


4.1.5 Orange Forest, Afternoon comparison

Figure 4.9: Hillaire method, Orange Forest, Afternoon, NR: 1

Figure 4.10: Unity method, Orange Forest, Afternoon, NR: 2

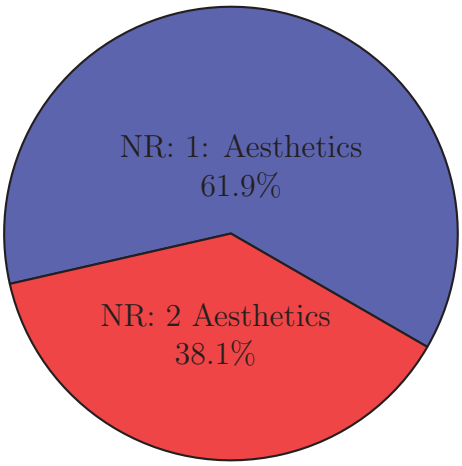


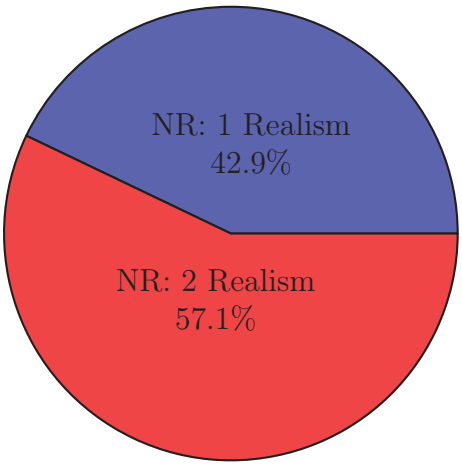


4.1.6 Orange Forest, Evening comparison

Figure 4.11: Hillaire method, Orange Forest, Evening, NR: 1

Figure 4.12: Unity method, Orange Forest, Evening, NR: 2

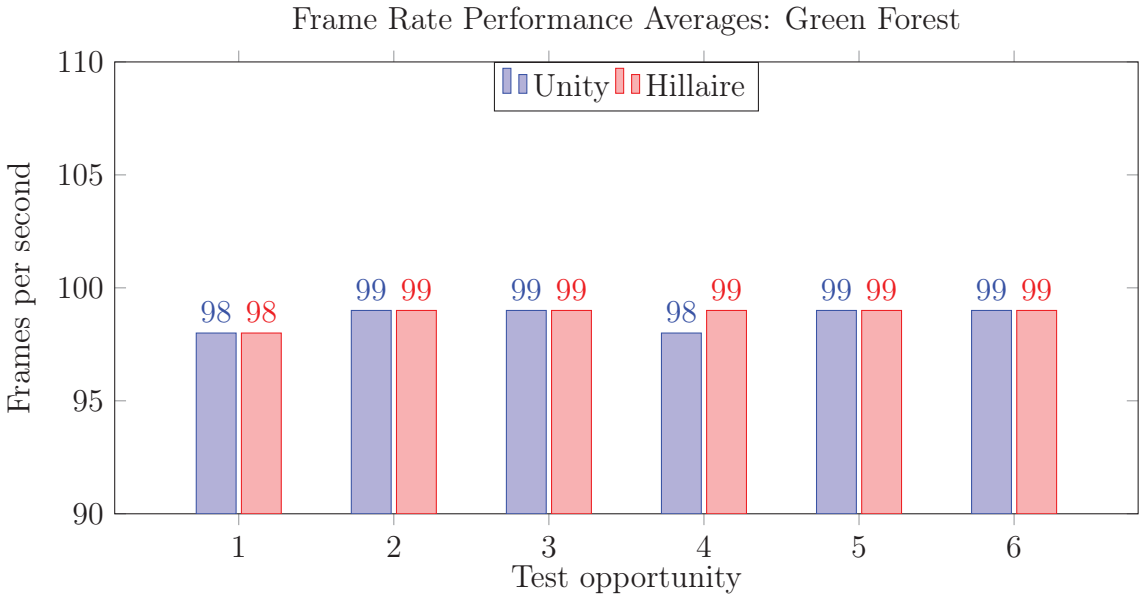


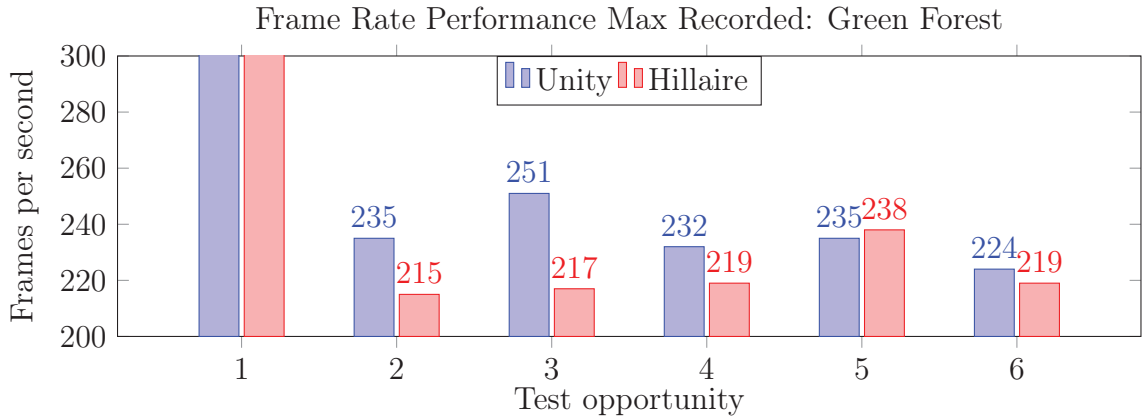


4.2 Performance results

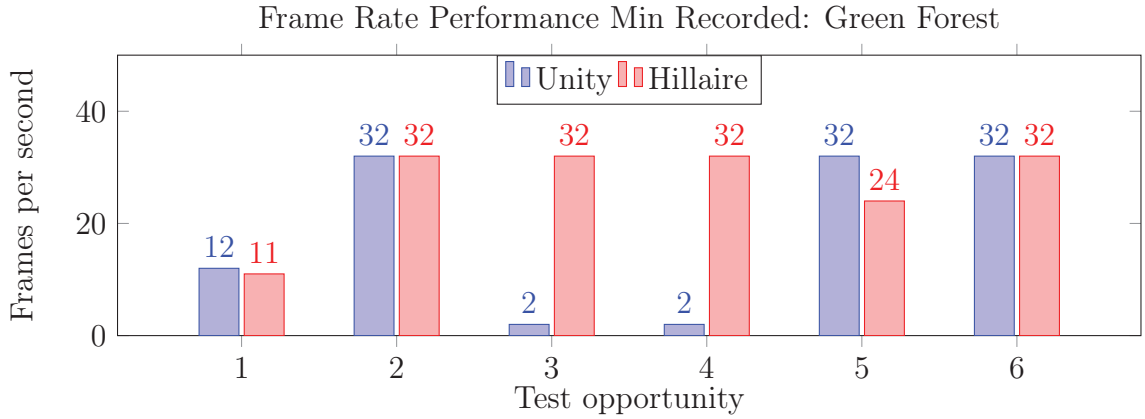
The frame rate and frame time performance were recorded for both scenes using both methods, one at a time. We chose to record the tests over five minutes. Each test for each scene and method was done six times to rule out abnormalities. The tests were conducted on an AMD Ryzen 9 5950x CPU and an Nvidia RTX 3080Ti GPU. In the Unity game engine, we completed the tests on both the "Every V Blank" VSync Count and the "Don't Sync" VSync Count settings, which gave the same results.

4.2.1 Green Forest Performance tests

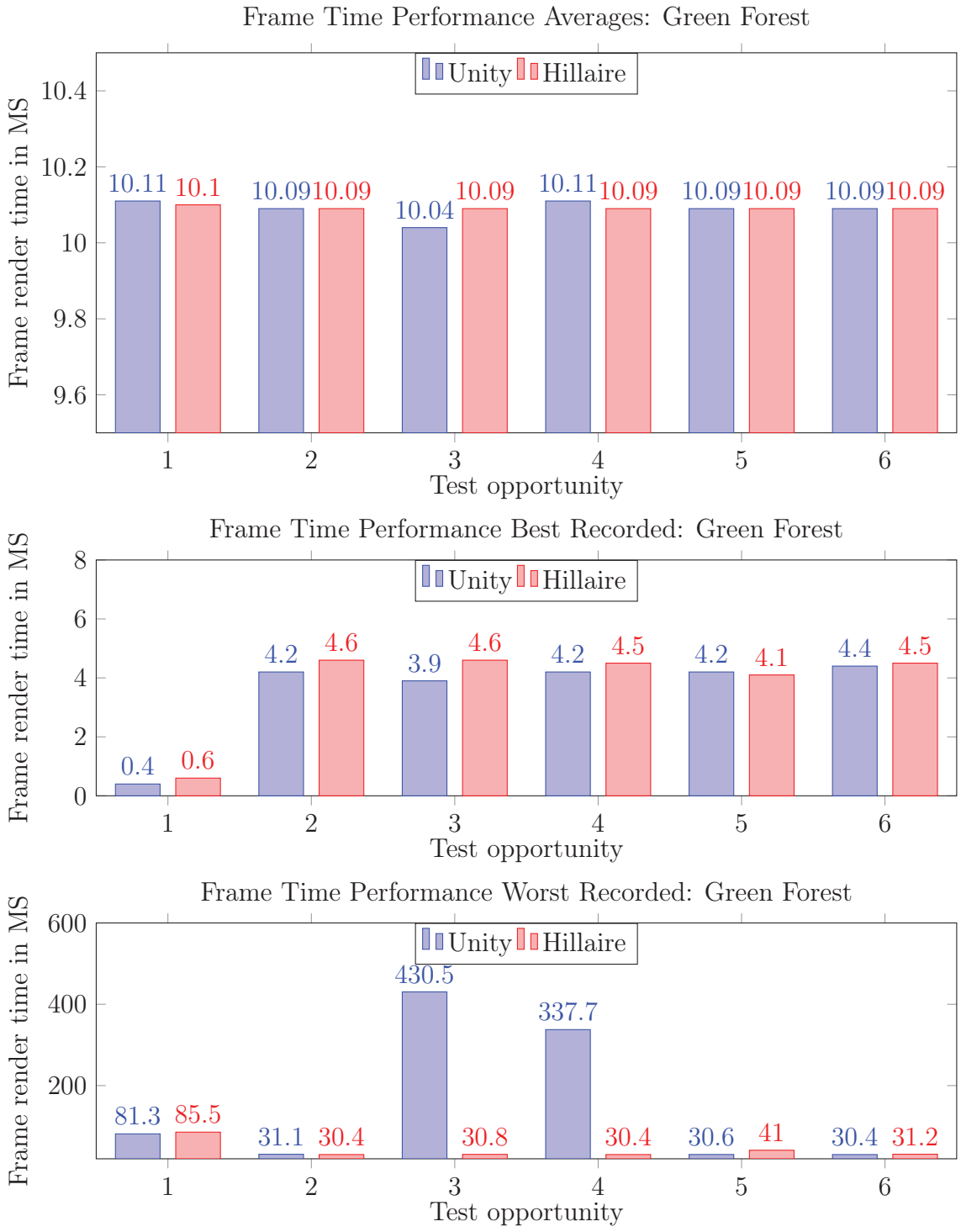




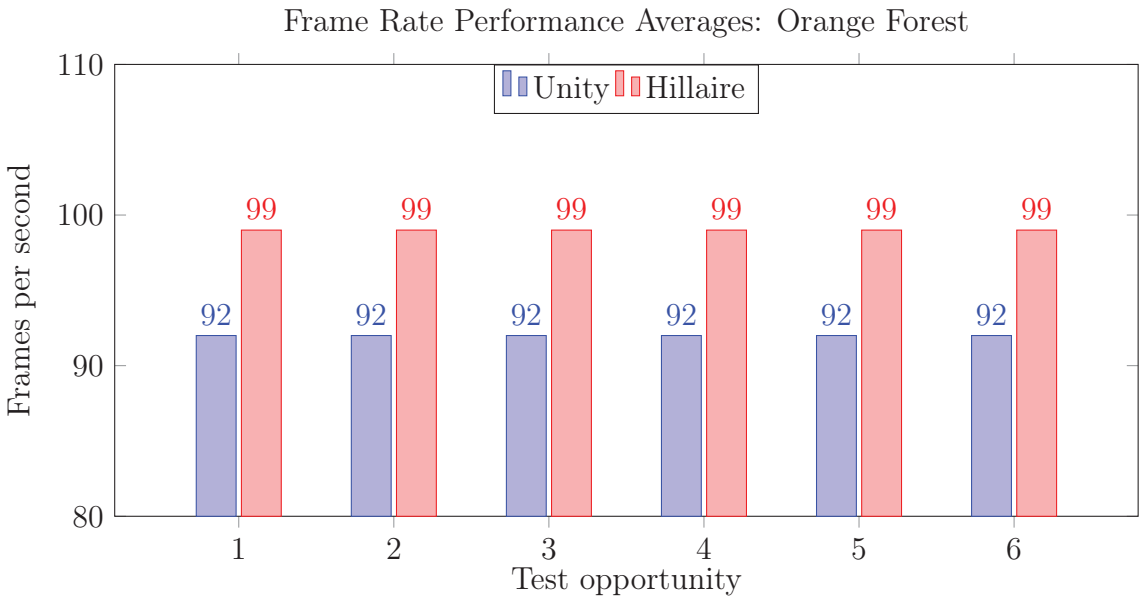
[1]

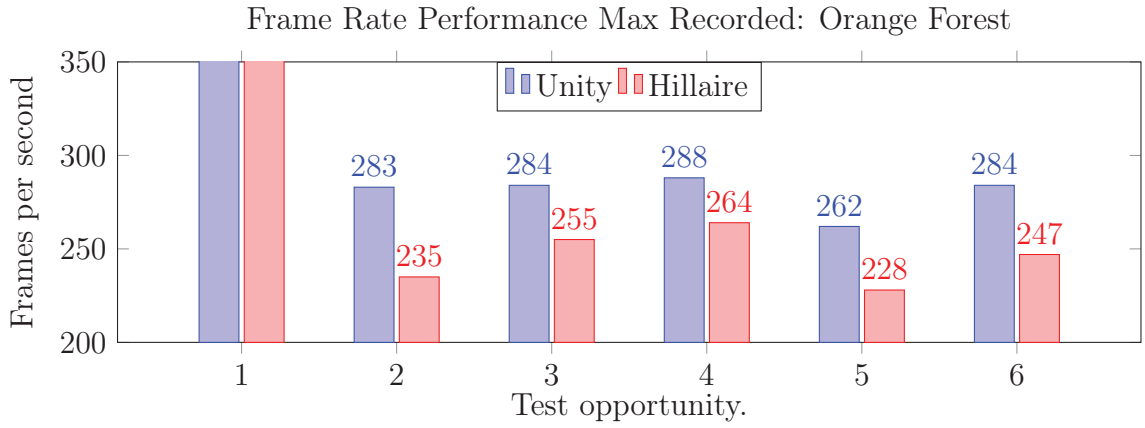


[1]The Unity method of the first max frame rate test had a max recorded frame rate of 2464 and the Hillaire method a frame rate of 1575.

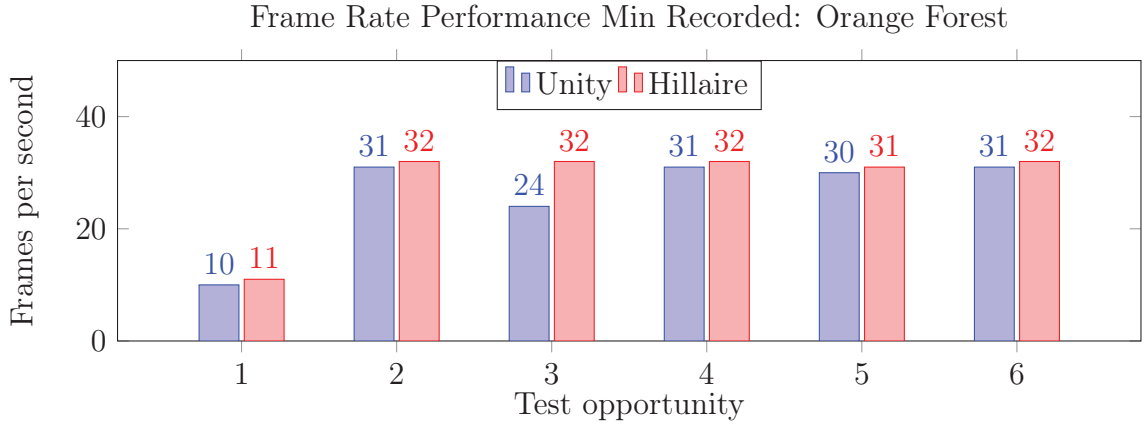

Frame Time Performance Averages: Green Forest
Unity
Hillaire
Frame render time in MS
10.4
10.2
10
9.8
9.6
10.11 10.1
10.09 10.09
10.04 10.09
10.11 10.09
10.09 10.09
10.09 10.09
1
2
3
4
5
6
Test opportunity
Frame Time Performance Best Recorded: Green Forest
Unity
Hillaire
Frame render time in MS
8
6
4
2
0
0.4 0.6
4.2 4.6
3.9 4.6
4.2 4.5
4.2 4.1
4.4 4.5
1
2
3
4
5
6
Test opportunity
Frame Time Performance Worst Recorded: Green Forest
Unity
Hillaire
Frame render time in MS
600
400
200
81.3 85.5
31.1 30.4
430.5 30.8
337.7 30.4
30.6 41
30.4 31.2
1
2
3
4
5
6
Test opportunity

4.2.2 Orange Forest Performance tests

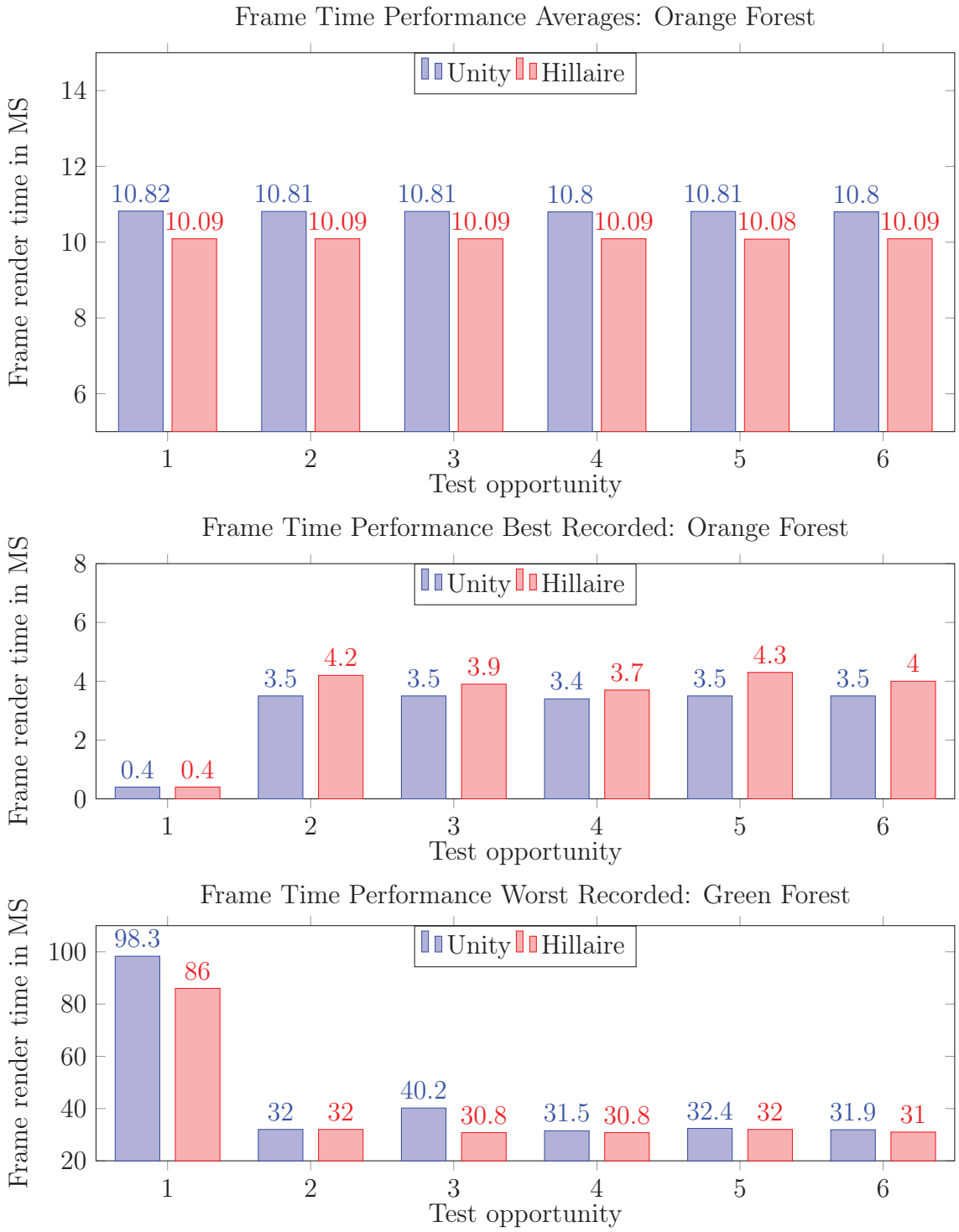




[2]



[2]The Unity method of the first max frame rate test had a max recorded frame rate of 2212 and the Hillaire method a frame rate of 2069.


Frame Time Performance Averages: Orange Forest
Unity
Hillaire
Frame render time in MS
14
12
10
8
6
10.82
10.09
10.81
10.09
10.81
10.09
10.8
10.09
10.81
10.08
10.8
10.09
1
2
3
4
5
6
Test opportunity
Frame Time Performance Best Recorded: Orange Forest
Unity
Hillaire
Frame render time in MS
8
6
4
2
0
0.4
0.4
3.5
4.2
3.5
3.9
3.4
3.7
3.5
4.3
3.5
4
1
2
3
4
5
6
Test opportunity
Frame Time Performance Worst Recorded: Green Forest
Unity
Hillaire
Frame render time in MS
100
80
60
40
20
98.3
86
32
32
40.2
30.8
31.5
30.8
32.4
32
31.9
31
1
2
3
4
5
6
Test opportunity

Chapter 5

Analysis

5.1 Survey results: Green Forest

The score for realism and aesthetics came out lower for the Hillaire method with about 30 - 40 % against Unity's 60 - 70 %. This is true for all Green Forest scenes except the Evening scene, where realism scored 52.4%, scoring higher than the Unity method for the first time in the Green Forest comparisons.

5.2 Survey results: Orange Forest

In the Orange Forest scene, the Hillaire method scored higher in aesthetics for all time of day comparisons by 60 - 80 % but scored lower in realism for all comparisons. With about 40% against 60%, the closest being 47.6 % against 53.4%.

5.3 Performance tests: Green Forest

Regarding the performance tests, the average frame rate is evenly matched when testing the methods against each other in the Green Forest scene. The Unity method recorded higher results four out of six times in the max recorded frame rates. Both methods are almost evenly matched in the minimum recorded frame rates except for two times. The Unity method plummets to two frames a second against the usual thirty-two.

The frame time comparisons are as expected when looking at the frame rate comparisons. The average is evenly matched with the best recorded giving the Unity method advantage, and the worst recorded, showing two plummeting values for the Unity method.

5.4 Performance tests: Orange Forest

The Orange Forests tests show advantages in average frame rates for the Hillaire method with a general 99 FPS against Unity's 92. The max recorded FPS again shows an advantage to the Unity method and, this time, a slight advantage to the Hillaire method when looking at the minimum recorded FPS.

Again the frame time average matches the FPS average and gives the Hillaire method an advantage. The best-recorded frame time shows Unity recorded the best

frame time, but the worst recorded frame time shows the Hillaire method scoring better worst frame times.

Chapter 6

Discussion

6.1 Survey results

As we were expecting the method by Hillaire to come out as more realistically accurate and more ascetically pleasing than Unity's method, the result was quite surprising. The performance results also came out as a surprise to us as we were expecting the Hillaire method to be little to moderately more performance demanding than a skybox.

6.1.1 Survey results: Green forest

The green forest results scored in Unity's favor, with almost all tests but one rating Unity's procedural sky-box more aesthetically pleasing and realistically accurate than the Hillaire atmosphere simulation. In these scenes, the fog is way more visible and prominent than in the Orange forest scenes and looks like a morning fog kind of deal. This kind of hides the depth of shadows in the trees and the shadow that falls on the ground below them. The fact that the mist/fog spreads all over the scene and is this thick, we think, is a leading factor in why the participants voted against this scene in favor of Unity's procedural skybox. Though the sky looks pretty with a nice gradient, the depth of the rest of the scene is obscured by the heavy fog that is so pleasing in the Unity scenes.

Figure 6.1: Hillaire method, Green Forest, Daytime

Figure 6.2: Unity method, Green Forest, Daytime

As for the Evening, the Hillaire scene was the only one that scored a higher score than the Unity scene regarding realism, though it was a close call. The significant difference in these scenes is the color of the light and sky. The Unity scene shows a relatively light blue sky, barely changing in terms of light color compared to the Hillaire scene, which shows a drastic change between the afternoon and evening. The Hillaire scene is orange in color with a dark blue/green sky. The scene is dark and tinted with an orange fog. Generally, a lovely orange/pink sky is what we like to see in the evening hours, and the Hillaire scene, unlike the Unity scene, displays that orange light and sun we are used to seeing. The Unity scene's sky does not change in color at all.

Figure 6.3: Hillaire method, Green Forest, Evening

Figure 6.4: Unity method, Green Forest, Evening

6.1.2 Green Forest settings

The green forest was a more extensive scene than the Orange Forest scene in terms of object sizes. As the scene was downloaded from the asset store and it used a terrain object, it would not scale down uniformly, forcing us to leave it as is in terms of size. As the atmosphere is created for a planet, the scenes in the out scenario have to be very small to create a realistic effect. As the scene was a bit bigger than we would have liked, the atmosphere appeared thicker than it should have, had the scene been scaled down somewhat.

6.1.3 Survey results: Orange forest

The results of the survey regarding the Orange Forest surprised us the most. Even though the results of the aesthetic aspect came out in favor of the Hillaire method, for all tests, the realism aspect did not. Once again, we think we have the fog to blame for this result. Even though the fog is way less intrusive in this comparison, it might introduce some hiding of depth details like in the Green forest scene. The sun of the Unity method is larger than the sun of the Hillaire method, which might be an essential factor. The overall look of the sun, from color to blur and light emission, seems different, which might give the Unity scene more realism than the Hillaire one.

Figure 6.5: Hillaire method, Orange Forest, Daytime

Figure 6.6: Unity method, Orange Forest, Daytime

6.1.4 Survey results: Final thoughts

The realism of the Orange forest when it comes to the Hillaire method might be affected by the color of the sky. As seen in the two Evening Orange forest comparisons, we find the Hillaire method scene's sky-tinted green, unlike the Unity method scene. This might be the result of the particular post-processing script we used for all scenes, but the effect might have been more assertive on the Hillaire scenes than the Unity ones. This can especially be seen in the Green Forest Evening scene as well. The sky in this scene appears dark turquoise instead of the typical orange, pink and blue we are used to seeing in the evening sky.

Another thing to consider is the fact that the participants did not see the pictures presented in the survey in a controlled environment. Variables like screen distance from eyes, the light level in the room, screen size as well as screen colors might have had an impact on the results. Another fact that could maybe have had an impact was the fact that the pictures were placed in a stacked fashion, always showing one method before the other method. This survey was also distributed without a consent form. This might have led to people under the age of 18 to participate as well. This however is highly unlikely as the form was only distributed in Discord servers where all participants are college students. If any of these variables had a significant impact on the results though is hard to say.

Figure 6.7: Hillaire method, Orange Forest, Evening

Figure 6.8: Unity method, Green Forest, Evening

We think the Hillaire method would get a better rating overall if the scenes had been a tiny bit smaller, especially the Green Forest one, to minimize the fog covering depth details in the scene assets. The green sky in all Hillaire scenes might have contributed to a minimized realism score and the size and blur of the sun. The Hillaire method does not present any pink being rendered in the sky at any time of

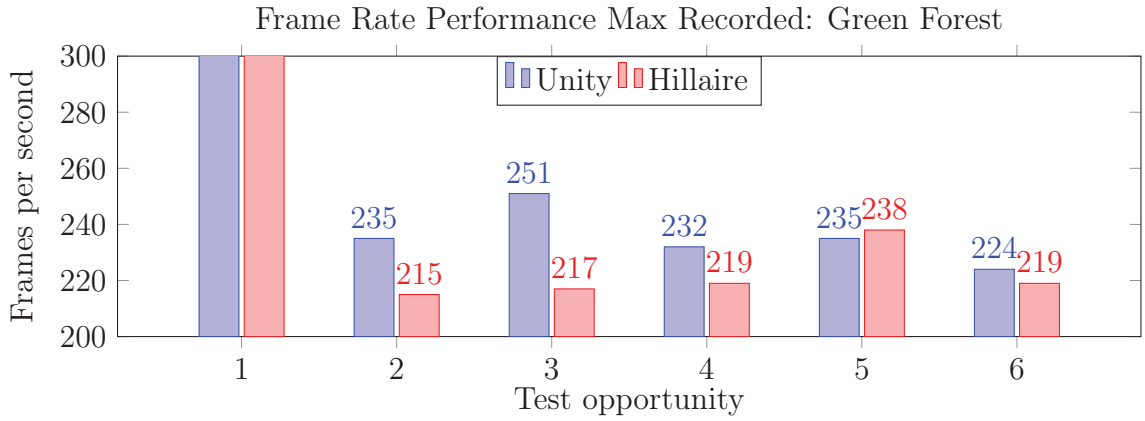
the day, which would give not only a more realistic sky but also a more beautiful one. The Unity scene would have gotten a better aesthetic score if the sky changed to a more orange one at later times of day, which it does not show signs of doing.

6.2 Performance results

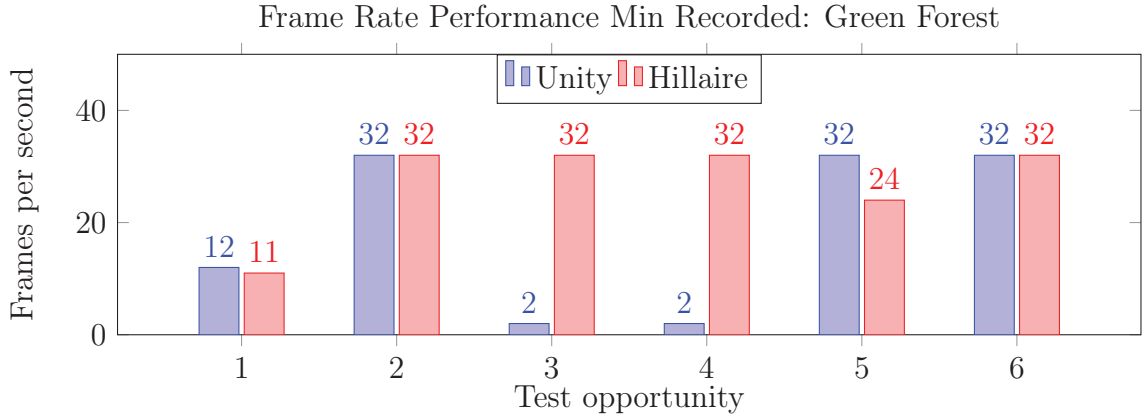
As mentioned, the results of the performance tests surprised us. At the beginning of this project, we thought the results of the Hillaire method would be a generally lower frame rate and a higher frame time which has not been the case.

6.2.1 Performance results: Green forest

The averages of the Green Forest Scene tests do not show any difference in performance except a very slight decrease for the Unity scene test in one case where it is not shown in the Hillaire Scene test. However, we find a more exciting result by looking at the Max recorded, and Min recorded frame rates. The Max recorded results show a generally higher frame rate for the Unity scene four times out of the six recorded. The first test shows a significant increase in the max recorded frame rate for both scenes, which is worth mentioning. The Min recorded frame rate also shows something interesting. The Unity method shows a plummet in frame rate two times out of six if we do not count the first tests that seem to be a weird fluke. This might show that the Hillaire method is slightly more stable than the Unity one, but it also might be a coincidence.



[1]

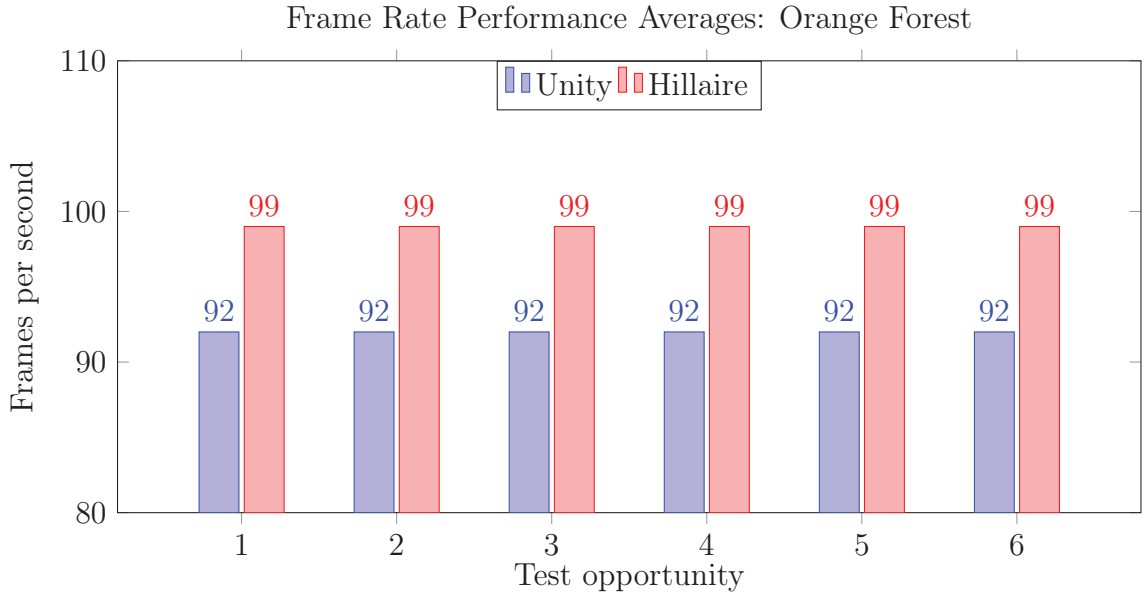


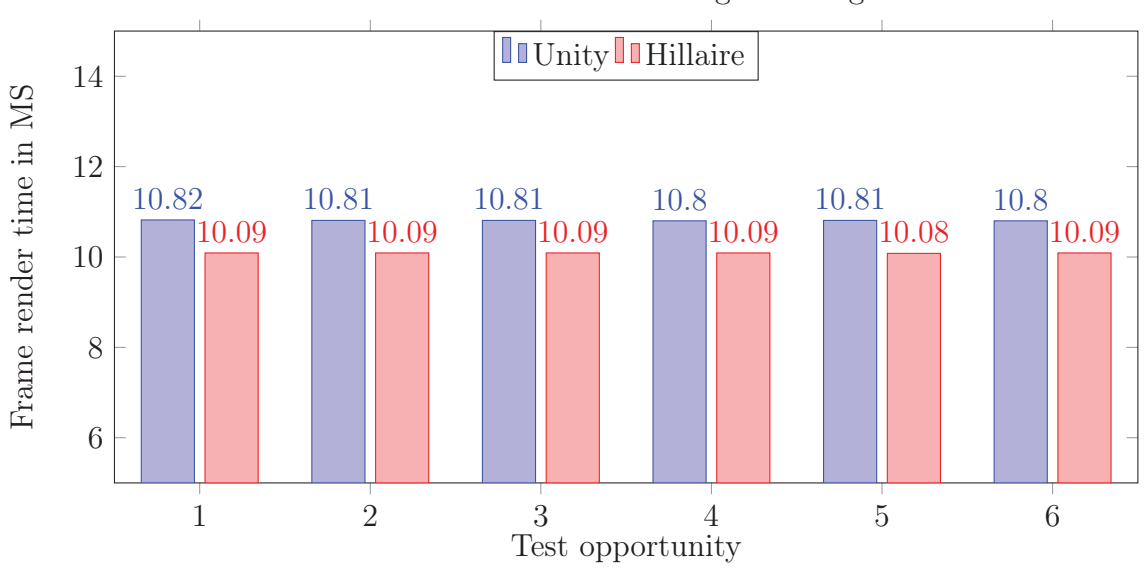
[1]The Unity method of the first max frame rate test had a max recorded frame rate of 2464 and the Hillaire method a frame rate of 1575.

6.2.2 Performance results: Orange forest

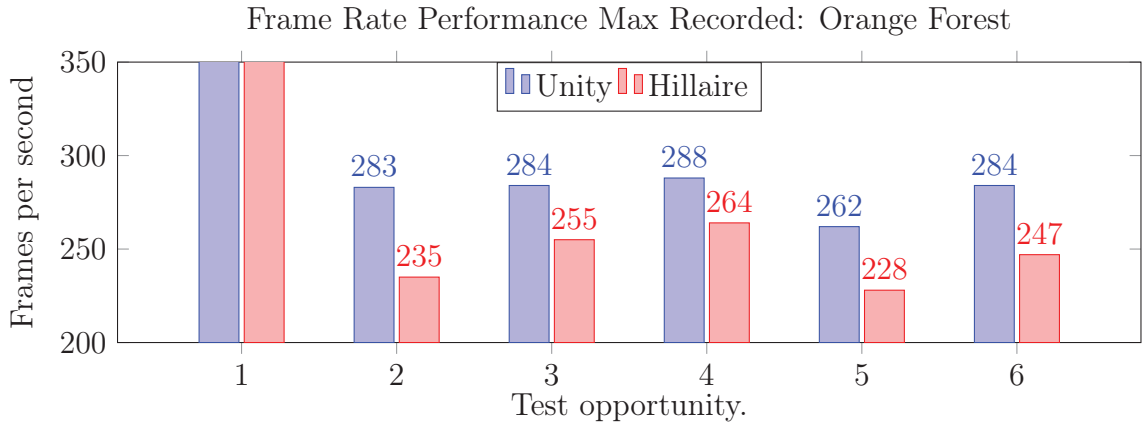
Something interesting happened to the average frame rate tests for the Orange Forest Scenes. Unlike the Green Forest Scene tests that showed a steady 99 frames per second for both methods, this is not the case for the Orange Forest Scene. The Hillaire method shows 99 steady frames per second as before, but the Unity method has dropped to 92 frames per second. The main difference between the Orange and the Green scene is using a reflection probe and water animation for the Orange scene, which the Green scene does not have. The settings for the Unity method were the same for both scenes, and so were the settings for the Hillaire method.

This result is somewhat odd given the circumstances unless the use of reflections would somehow make it harder to render a skybox? We find the relation to be unlikely.





Interestingly enough, the Max recorded frame rate for the Unity scene scored higher in all six tests even though the average recorded frame rate was noticeably lower. The Min recorded frame rate is about the same for both methods and does not show anything of use.



[2]

[2]The Unity method of the first max frame rate test had a max recorded frame rate of 2212 and the Hillaire method a frame rate of 2069.

Chapter 7

Conclusion

This project has taken some unexpected turns throughout the time conducting it. Unfortunately our project did not lead to a helpful plugin for future use as planned initially, but that was not the project's goal. The plugin part was a mere bonus. Not only did our plan for this project take a couple of turns, but so did the result. The tests have shown a much different outcome than we expected. The Hillaire method scored lower in realism in every case except one, but it also scored lower in aesthetics 50% of the tests. For example, this outcome might have resulted from several factors like the extra post-processing effect tinting an unnatural color on the sky. We did have much trouble getting the scenes to look natural in the Hillaire atmosphere, and getting the size of the scene right was a huge hassle since we used pre-made assets from the asset store, which made it much bigger than the atmosphere itself. Since the scenes would not scale uniformly, we had to settle for a larger scene than we would have recommended for others using this method in the future. When it came to the performance tests, we expected the much more complex method of Hillaire to be more performance demanding than the procedural skybox by Unity, which was clearly shown not to be the case. The method by Unity showed a higher max recorded frame time and rate, but the average is the main contender, which gave the Hillaire method an advantage. The method by Hillaire is much more optimized than we would have thought.

7.0.1 Research questions

- RQ1: How does the method by Hillaire when implemented into Unity Universal Render Pipeline stand compared to the Unity procedural skybox in the same pipeline regarding aesthetics and realism?

As concluded, the method by Hillaire does not stand firm against the Unity procedural skybox with the settings used in our study.

However, we believe the result could be much more significant in both realism and aesthetics if the settings were modified. This would give the sky a much more natural color, making the scene smaller to eliminate that heavy fog and giving the sun a less soft blur and a bigger size.

- RQ2: How does the method by Hillaire regarding frame rate and frame time performance compare to the frame rate and frame time performance of Unity procedural skybox?

Overall, the Hillaire method compares well to Unity's procedural skybox, with a steady average frame rate of about 99 for both methods. In some cases, the method by Hillaire may perform better than the method by Unity both in average frame rate as well as stability.

Chapter 8

Future work

It would be fun to see a plugin featuring this method by Hillaire for future implementations. The plugin could use variables like atmosphere radius and thickness to make it easier to use for all scene sizes and an atmosphere center point. There could be an easy way of changing the color gradient to give the sky a pink tint and orange to give a closer resemblance to the real-life evening sky. The plugin could also feature volumetric clouds with different weathers like snow or rain.

References

[1] Blender. (2022) Cycles introduction. Last accessed June 16, 2022. [Online]. Available: https://docs.blender.org/manual/en/latest/render/cycles/introduction.html

[2] E. Bruneton and F. Neyret, "Precomputed Atmospheric Scattering," *Computer Graphics Forum*, vol. 27, no. 4, pp. 1079–1086, Jun. 2008. [Online]. Available: https://hal.inria.fr/inria-00288758

[3] Chaos. (2022) 3d rendering software. Last accessed June 16, 2022. [Online]. Available: https://www.chaos.com/3d-rendering-software

[4] V. D. Community. (2021) Skybox (2d). Last accessed June 16, 2022. [Online]. Available: https://developer.valvesoftware.com/wiki/Skybox_(2D)

[5] U. Engine. (2022) Unreal engine. Last accessed June 16, 2022. [Online]. Available: https://www.unrealengine.com/en-US/)

[6] S. Hillaire, "A scalable and production ready sky and atmosphere rendering technique," *Eurographics Symposium on Rendering 2020*, 2020.

[7] H. W. Jensen, F. Durand, J. Dorsey, M. M. Stark, P. Shirley, and S. Premože, "A physically-based night sky model," pp. 399–408, 2001, cited By :12.

[8] S. Lague. Coding adventure: Atmosphere. Youtube. [Online]. Available: https://www.youtube.com/watch?v=DxfEbulyFcY

[9] ——. Trying to improve my geography game with more real-world data. Youtube. [Online]. Available: https://www.youtube.com/watch?v=UXD97l7ZT0w

[10] Otoy. (2020) About octanerender. Last accessed June 16, 2022. [Online]. Available: https://home.otoy.com/render/octane-render/

[11] Sebth. Unrealengineskyatmosphere. GitHub.com. [Online]. Available: https://github.com/sebh/UnrealEngineSkyAtmosphere

[12] U. Technologies. Camera path creator. Unity Technologies. [Online]. Available: https://assetstore.unity.com/packages/tools/camera/camera-path-creator-84074

[13] ——. Physically based sky. Unity Technologies. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@7.1/manual/Override-Physically-Based-Sky.html

[14] ——. Procedural skybox. Unity Technologies. [Online]. Available: https://docs.unity3d.com/Manual/shader-skybox-procedural.html

[15] ——. Reflection probe. Unity Technologies. [Online]. Available: https://docs.unity3d.com/Manual/class-ReflectionProbe.html

[16] ——. (2020) About the universal render pipeline. Last accessed June 16, 2022. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/manual/index.html

[17] ——. (2022) High definition render pipeline overview. Last accessed June 16, 2022. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@13.1/manual/index.html

[18] Unity, "Stylized forest environment," 2022, last accessed April 2022. [Online]. Available: https://assetstore.unity.com/packages/3d/environments/fantasy/stylized-forest-environment-81745

[19] ——. (2022) Unity. Last accessed June 16, 2022. [Online]. Available: https://unity.com/

[20] Wikipedia. Delta timing. Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Delta_timing

[21] A. Wilkie, P. Vevoda, T. Bashford-Rogers, L. Hošek, T. Iser, Koláář, M. ová, T. Rittig, and J. Křivánek, "A fitted radiance and attenuation model for realistic atmospheres," *ACM Transactions on Graphics*, vol. 40, no. 4, 2021, cited By :1.

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden