UMEÅ UNIVERSITY

AN EVALUATION OF REAL-TIME GLOBAL ILLUMINATION TECHNIQUES

Johan Sundkvist

Bachelor Thesis, 15 credits
Education program

2019

Abstract

Real-time global illumination techniques are becoming more and more realistic as the years pass and new technologies are developed. This report therefore set out to evaluate different implementations of real-time global illumination. The three implementations that are looked at are Unity, Unreal and a Voxel Cone Tracer. For Unity and Unreal we also look at their three main settings for global illumination, in order to look at the differences between methods using pre-computation and those that are run-time focused.

The evaluation focuses on visual quality and render times. In order to compare render times we look at how the framerate is effected by both scaling up the number of lights and then adding motion to the light sources. When comparing the results from a visual quality perspective we made use of the perceptual quality metrices SSIM and HDR-VDP-2 with reference images rendered in Blender. For these tests we use three different scenes with different lighting conditions.

We conclude that while the Voxel Cone Tracer faced large framerate problems when lighting was scaled up, the higher end implementations in the form of Unity and Unreal, neither had any problem keeping a high frame rate when the same tests were applied to them. The visual quality tests faced some problems as the tests using HDR-VDP-2 had problems getting any good results. The results from SSIM were more useful, showing that there still is progress to be made for all the implementations and that the ones using more pre-computations generally perform better.

Contents

1 Introduction

This work is an evaluation of multiple state of the art implementations of real-time global illumination. The goal of this is to analyze the accuracy and efficiency of these implementations in both simple and complex light transport conditions. Focusing on both visual quality and speed.

1.1 Background

In the field of computer graphics, realistic illumination has always been a major hurdle. Calculating the direct illumination caused on an object in the immediate view of a light source is no longer a problem; the same can however not be said about indirect illumination.

Correctly calculating how the light hitting one object will effect those surrounding it has always been very difficult due to the amount of computation required. The reason that so many calculations are required is pretty simple. In order to perfectly emulate the effect of a light source one would have to send out practically infinite rays from it, which then hit an object and create new rays bouncing in practically infinite directions and so on. Since we cannot peform infinite calculations in an instant we can only create an approximation of realistic lighting, and the more time we have the closer can bring our approximation to reality. This is why high level global illumination generally has been reserved for non real time applications such as movies where companies like Pixar can afford to let a movie render for 100 million CPU hours[8].

Recently improvements have been made as modern hardware improve computation speed, and clever shortcuts reduce the total number of computations required. This has been pushed by technological innovation over the past 10 years; NVIDIAs RTX framework allowing for ray-tracing in the real-time graphics pipeline with the OptiX API is an example of this[11].

All of this activity means that there are many different approaches to real-time global illumination; methods making use of ray-tracing, photon mapping and other techniques to achieve a realistic look under massive time constraints. All meant to handle the same problem but in different ways. Since the subject of real-time global illumination appears to become more and more relevant, it is important to look closer at these various implementations and examine their strengths and weaknesses; whether it is hardware scaling or difficulties in specific lighting and environment conditions.

1.2 Purpose and Research Question

The purpose of this study is to evaluate how different implementations of real-time global illumination, perform in scenes with either varying lighting complexity or a focus on specific aspects of global illumination (such as specular reflection). The evaluation is done from two perspectives: render-time, based on scalability and handling of dynamic lighting; and visual quality, tested by using the perceptual quality metrices SSIM and HDR-VDP-2 to compare the implementations output to that rendered in Blender. We also use the results from this to answer a secondary question. How useful are the perceptual quality metrices when applied to real-time implementations of global illumination?

2 Related work

A general problem that can be seen in papers discussing implementations of global illumination is that every paper uses different methods and metrics to evaluate their implementations. Sun and Agu[7] for example shows render times for their implementation on a couple of scenes, but does not perform any comparison with other implementations. Yao et al.[16] also shows render times for various scenes but does include a comparison with other implementations. These comparisons are entirely visual however so they are subjective comparison.

A previous comparison between multiple modern solutions to real time global illumination that attempted to minimize the subjective aspects was conducted by Ritschel et al.[6]. In 2012 they compared a large number of implementations of different techniques for a large number of criteria such as speed, quality, dynamics, scalability, GPU(how well it maps to GPUs) and implementation. The problem that they admit themselves is that some of these criteria (quality and dynamics) are still very subjective. But as a general overview of the strengths and weaknesses of different techniques it offers a good guideline.

In 2015 Balen Meneghel and Lobo Netto attempted to build a comparison basis for analyzing global illumination methods based on methods using perceptual quality metrices[5]. The metrices used for this were Structural Similarity Index (SSIM), Visual Difference Predictor (VDP-HDR-2) and metrices which consider only numerical absolute differences between images, such as Root Mean Square Error (RMSE). Since the results of this study are achieved through comparison with reference images it allows a less subjective analysis of each implementations visual quality. The study was however not done on real time global illumination, but it should still be applicable.

3 Global Illumination and Related Terms

Global illumination (GI), or indirect illumination is the name given to the algorithms used in computer graphics with the purpose of calculating more realistic lighting. These algorithms take into account not only the light that comes directly from a given light source, but also how that light upon impact with a material continues to bounce around the given scene. This allows for (among other things) reflections, refractions and more complex shadows, although the most important element is probably simple indirect diffuse lighting. Ray-tracing and Photon mapping are two examples of algorithms in the global illumination category.

Ray tracing works by first sending out rays from the camera. When those rays then make impact on an object more rays are sent from the impact point in several directions: through the object, in order to determine refraction; the mirror-reflection direction, to see what will be seen in the objects reflection; and towards light sources, in order to check if there is anything blocking the light.[2] Photon mapping works by instead first tracing rays (also called photons) from the light source instead of the camera. It then stores the results of these rays impacts in so called photon maps. In step two a ray tracing algorithm that makes use of the photon maps renders the scene.[3]

One of the algorithms that is used by a program tested in this report is called Cone Tracing. Cone tracing is a modified version of ray tracing that extends the definition of a ray, which is traditionally infinitely small, into a cone. What this means is that it includes information on the spread angle and the virtual origin. This allows for a couple of improvements: better anti-aliasing, a way of calculating fuzzy shadows and dull reflections, a method of calculating the correct level of detail in a procedural model and texture map, and faster intersection calculation.[1]

All the previously mentioned algorithms can be quite computationally demanding as there theoretically can be innumerable light rays bouncing around thousands of times at any given instance. Due to the time constraints present in any application that is meant to be interactable in real-time, many implementations will perform a large amount of pre-computation in order to avoid having to perform the heavy calculations during run-time. This is often referred to as baking.

When emulating the way light works in the real world we can potentially use many different types of light sources, but in this report only two different types are used. The first of these is point lights, which is simply light emanating in all directions from a coordinate. Directional lights are the second type used and is light comping from a specific direction. For a real world comparison these light sources can be thought of as an infinitely small (but still glowing) light bulb, and the sun in the sky.

4 Choice of Global Illumination Methods

In this study the following methods were chosen. These specific implementations were chosen in order to allow testing different approaches of calculating the indirect lighting.

•

Voxel Cone Tracer: A global illumination approach that uses a combination of cone tracing and voxels. First it discretizes the scenes geometry using voxels. This is followed by three steps during which the lighting is computed based on this discretized representation of the scene. Some problems are apparant due to the approach only being an approximation (like all GI algorithms). For this engine they include light leaking, caused by the geometry simplification; and color banding, caused by the sampling steps of the voxel structure during cone tracing. [14]

Unity: The global illumination engine used by Unity supports two different methods of computing global illumination: baked GI, which works by precomputing all indirect lighting and storing it for later use. The second method added in Unity 5.0 is called Precomputed Realtime GI. As the name implies it still requires a precomputation phase. But it does not just calculate how light will bounce of an object, but all possible ways it can bounce of it. This means that the number of lights, their direction, position, and other properties can all be modified and the lighting will update accordingly. This is not possible with baked GI. Both of these methods share one problem however: they both only calculate the indirect lighting from static objects. This means any dynamic object in the scene can be impacted by the indirect lighting (if you also use light probes), but will then not impact the rest of the scenes lighting. [9] This means that there are three different settings for each light source in unity; namely baked, mixed and realtime. These will all have to be used at different instances as baked for example performs the most precomputations and should therefore have the highest visual quality, but does not allow for moving lights. Mixed also pre-computes a lot (although less than baked) but still does a lot during run-time which is why it can handle moving lights. Realtime is the one that performs most of its calculations during run-time, as the name implies.

Unreal Engine: While Unreal Engine is capable of using many third party global illumination implementations, its primary technique is called lightmass. The diffuse interreflection

offered by lightmass is described by Unreal as "*by far the most visually important global illumination lighting effect*"[12]. It lets light bounce in all directions with their outgoing strength and color based on the impacted material. This is also known as color bleeding. It makes use of four diffuse bounces in order to create realistic illumination. Like Unity this is a method that requires baking. The baking is not as extensive however. Like Unity unreal also has three different settings for lights. These are: static, stationary and movable. Even movable which is calculated in real time still makes use of shadow caching, this allows it to reuse shadows if the lights are not currently moving.[13]
Comparing these to Unity static is Unreals version of Baked and movable is the equivalent of realtime. Stationary and mixed are similar but stationary does not allow for moving lights. It does however allow for changing light intensities and color and similar changes, which separates it from static.

Using these different implementations (and their different settings for GI) allows us to test a somewhat large spectrum of approaches. Going all the way from fully baked lighting in Unity Baked, methods making use of shadow caching like Unreal Movable and methods meant to be fully dynamic in the form of the VCT engine.

5 Evaluation Methods

The evaluation of the various implementations is done by using several different evaluation methods. This is done in order to properly evaluate the implementations based on their visual quality as well as their dynamics and scalability.

5.1 Visual Quality

The visual quality of the implementation is tested by using perceptual quality metrices. Two differnt metrices will be used for this task: Structural Similarity Index (SSIM) and HDR-VDP-2. SSIM is a metrice that evaluates the structural similarity of two images by comparing local patterns of pixel intensities that have been normalized for luminance and contrast.[15] HDR-VDP-2 instead looks at the luminance conditions of the test image in comparison to the reference image, without considering color.[4] This means that SSIM looks at how things change from one area to the next in the test image compared to how they change in the reference image; things like too hard shadows can therefore be something that has a negative effect on the score. HDR-VDP-2 on the other hand looks at the luminance of individual areas in the two images. Therefore it will instead react negatively to areas being either too bright, or not bright enough.

These tests gives us the ability to make comparisons between the results from the various implementations and those generated by a higher end program like Blender. This allows us to evaluate how close the images rendered by these real-time implementations are to more time demanding methods of achieving realistic lighting. An example of one of these tests can be seen in figure 1.

The reference images are generated using blender. In order for them to bee free of noise, they are rendered for several hours. The test images are simply screenshots taken from the various programs displaying the same scenes as the reference images.

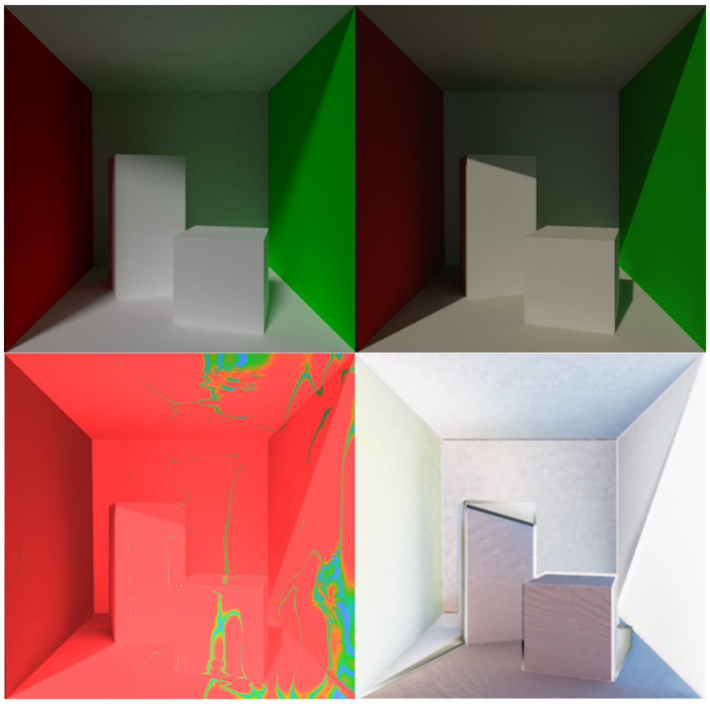

Figure 1: Top left is a reference image rendered in Blender. Top right is a image made in Unity. Bottom left is the resulting image from the HDR-VDP-2 test with a quality score of 54.08%. Bottom right is the resulting image from the SSIM test with a quality score of 73.63%

5.2 Scalability

This tests if the various methods can maintain their speed when the number of lights increases. This allows us too see how a more complex lighting setup effects the performance, which is important to keep in mind so that the focus does not simply fall towards pure visual quality without taking usability into account. The results will simply be measured in frames per second.

5.3 Dynamics

Being able to handle dynamic scenes and lighting is important for any type of interactivity to be possible. The goal of this test is similar to that of the scalability test in that it allows us to look at another aspect besides the viusals. In this study an implementations dynamics are tested using moving lights. This is another case where the results are measured in frames per second.

6 Test Scenes

The scenes chosen are all relatively simple and meant to establish an environment where specific light transport conditions are explorable. The scenes being simple makes it easier to test those specific conditions. The only scene that is more complex is Crytek Sponza.

Cornell Box Diffuse: This is a recreation of the classic cornell box. It is meant to provide a scene with simple lighting conditions and entirely diffuse materials.

Cornell Box Mirror: A scene in almost all ways identical to the previous Cornell Box except the larger square has been replaced with kone that works as a mirror. Allowing us to test specular reflections.

Crytek Sponza: A version of the CRYENGINE Sponza scene. Provides a scene with more complicated lighting conditions and materials. Allows for testing with a scene that is more demanding for the engines.

7 Test Setup

Setting up the testing environment on all the different systems requires choices to be made that can potentially have a significant effect on the results of the subsequent tests. Since programs like Unity and Unreal are highly customizable with long lists of settings that can be changed; countless hours could be spent trying to optimize the different programs for the testing conditions. The choice was therefore made to keep most settings the way they are when first installed. Thereby keeping changes limited to remedy immediate problems, such as Unity experiencing light bleeding at the edges of the cornell box. These different setting does mean that the different programs will not give entirely matching results, which can be seen in figure 2. Everything is rendered on a computer running Windows 10 64 bits, Intel(R) Core(TM) i5-6500 processor running at 3.20GHz with 4 cores and 4 threads, 16 GB of RAM memory and an AMD Radeon(TM) R9 390.

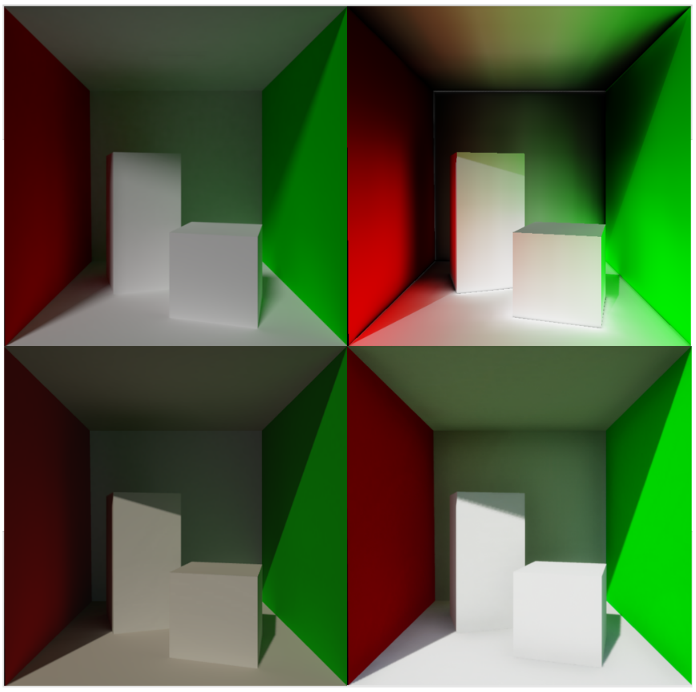

Figure 2: Renders of the Cornell Box scene. Left to right and top to bottom: Blender, VCT engine, Unity Baked, and Unreal Static.

8 Results and Analysis

The results here are split up into the three different categories. The results from each category are also analysed both in their own vacuum and taking the previous tests into account. This analysis allows us to give theories regarding some odd results before moving on to the next set of results. The images tested are all rendered at a resolution of 1442 times 811 and then cut down in order to be able to test the scenes at their full scopes without irrelevant things such as the background color impacting the results of the visual-quality metrices.

8.1 Visual Quality

During the visual quality tests, all the settings discussed in Section 3 are used. This means that Unity will be tested using baked, mixed and realtime lighting settings; Unreal tested with static, stationary and movable mode, and VCT will only be tested once since it does not have multiple modes in the same way that the others have. This has to be done to allow the results from these tests to be applicable to the later analysis of the scalability and dynamics tests, where not every lighting mode will be usable.

The reference images were rendered for several hours on blender. Cornell Box and Cornell Box Mirror are both lit by a single directional light while Crytek Sponza is lit by six point lights of different colors. The reference images can be seen in figure 3.

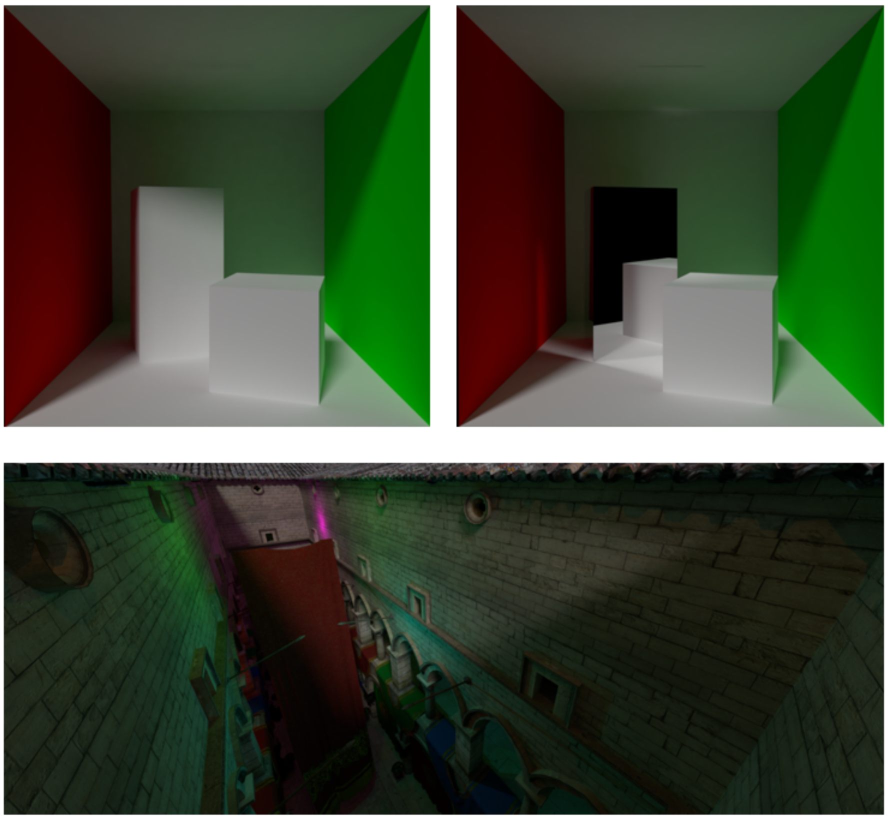

Figure 3: Reference images for the different benchmark scenes. From left to right and top to bottom: Cornell Box, Cornell Box Mirror and Crytek Sponza.

Unity

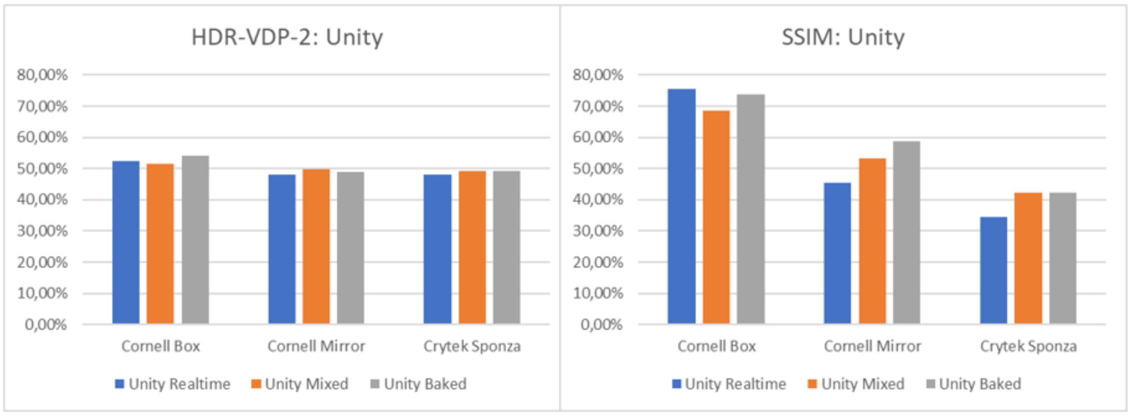

Figure 4: Results from the visual quality tests on the three Unity methods.

From the results given in figure 4, we can see that Unity performs best in the cornell box scene judging by both HDR-VDP-2 and SSIM. There is then a clear drop as we move on to the more complex scenes. This is not surprising as they introduce the difficulties of accurate reflection in Mirror and the increased lighting- and scene-complexity of Sponza. The drop in HDR-VDP-2 is however not nearly as dramatic as it is for SSIM. Looking specifically at the results for HDR-VDP-2 there is not a clear pattern. None of the lighting modes consistently achieve results that are substantially better then the other modes and they all stay around 50%.

Moving on to SSIM we see larger differences between the different modes across the three scenes. In the cornell box they are all at around 70%. Whereas they are all below 50% in the Sponza test. As the difference between the different modes is larger here we can also make clearer judgements regarding the results. Namely that Unity Realtime starts to fall clearly behind the others once we move on from the Cornell Box. The results points to the solutions making use of some amount of pre-computed data getting better results, expecially when judging by structural similarity.

Unreal

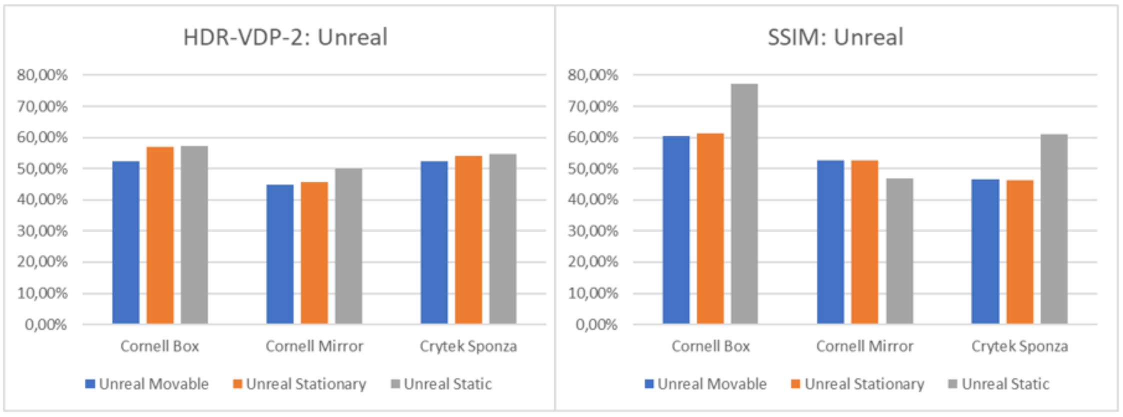

Figure 5: Results from the visual quality tests on the three Unreal methods.

Unreal seems to follow the same pattern as Unity when it comes to the HDR-VDP-2 results, altough the variance is slight larger it still stays around 50%. Here we can not however see a clear decline going from Cornell Box to Crytek Sponza. It does still seem however that Unreal just like Unity has a hard time getting good results when judging by the luminance output. There does still appear slight benefit to using the modes with more pre-computed data.

Looking at the SSIM results we can see that movable and stationary both interestingly enough get basically identical results declining at the same speed across the three scenes. Static does appear to have outperform them significantly, except for the mirror scene where it drops slightly below the others. The introduction of such a reflective material possibly causing problems for the accuracy of the pre-computed lighting.

VCT

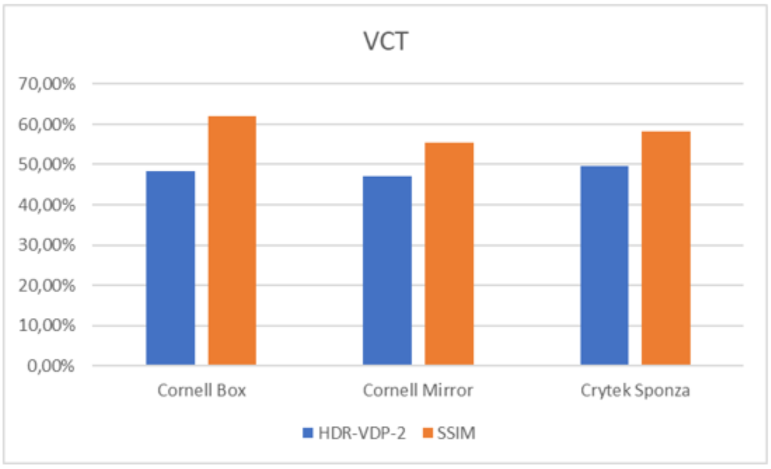

Figure 6: Results from the visual quality tests on the VCT engine.

Since there is no version of this engine making use of pre-computation, we can only look at the VCT engines performance from scene to scene and how it holds up as the environment becomes more complex. First we can see that it continues the pattern of the other two programs in having a hard time getting out of the 50% area for HDR-VDP-2. In this case being just under in for every scene. It does manage to get pretty good results when judging by structural similarity however; staying remarkably consistent for every scene, someting neither Unity nor Unreal can say for any of their modes. Despite the scenes becoming more complex and the lack of any pre-computing. It stays around 60%.

8.2 Scalability

For the scalability tests the settings used for the various programs are the same as those for visual-quality. Testing is done with the number of point lights going from zero to six, the seventh light added is a directional light.

Crytek Sponza

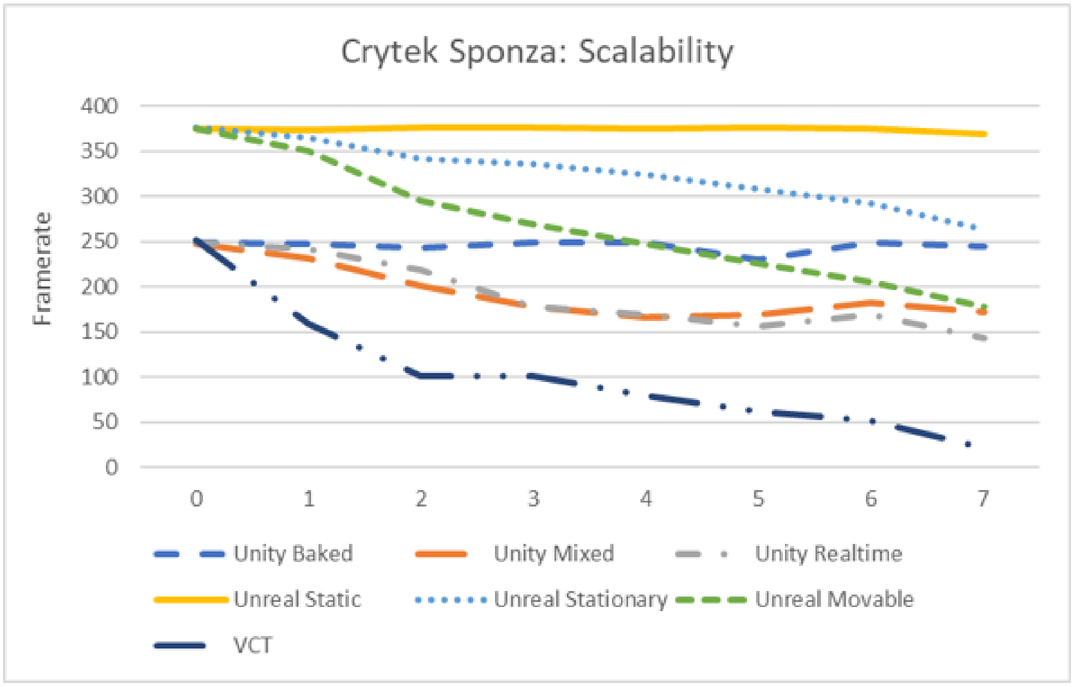

Figure 7: Framerate on the Crytek Sponza scene with various number of lights.

As we can see in Figure 7, the VCT engine manages to stay over 60 frames per second up until the sixth light is added. Then once the directional light is added it drops to around 20. Both Unity and Unreal stay quite stable from a framerate perspective. Neither of which drop over one hundred frames per second. Both staying over two hundred frames when there are 6 lights. The dramatic difference between how much effect increasing the number of lights had on VCT engine in comparison with the other two makes sense as VCT updates each frame and does the calculations for the dynamic voxelization, voxel direct illumination, voxel global illumination and necessary anisotropic filtering steps each update.[14]

Unreal and unity both cache some of the calculations for reuse in later frames. Unreal for example caches the shadow maps even when lighting is set to movable, as long as the light is not actually changing position.[13] Unity also does this through pre-computed lightmaps that the real-time lights can reference thanks to its "pre-computed global illumination".[10] Looking at these results we can see how much of an effect baking can have. As both Unity Baked and Unreal Static manages to keep a consistant framerate as the number of lights increases. The same is not true when it comes to their movable alternatives however. As Mixed and Realtime Unity along with Unreal Movable, all face a large drop as the number of lights increases.

8.3 Dynamics

All tests done for this section requires movable lights. This means that only certain settings are viable for some of the implementations. Unreal will have to make use of its "movable" setting for all lights, and Unity will likewise use the "realtime" and "mixed" lighting setting, with realtime global-illumination activated. Testing is done with the number of point lights going from zero to six, the seventh light added is a directional light.

Crytek Sponza

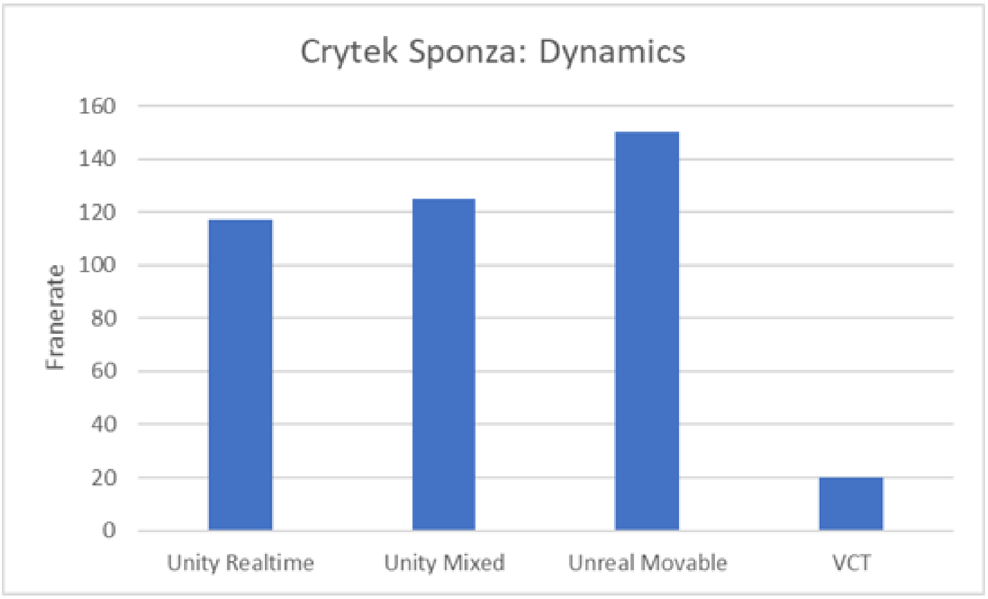

Figure 8: Framerate on the Crytek Sponza scene with six moving point lights and one rotating directional light.

Looking at the framerate results in Figure 8 we can see a large difference between the different programs and where their framerate stands. VCT does not face any significant decline in framerate when compared to the static lights in the scalability test. This makes sense since (as previously explained in the scalability part of the results) the VCT engine updates its calculations each frame.[14] Both Unity and Unreal manage to make the move from stationary-to moving lights without a too severe drop in framerate. Looking at pure changing numbers from the scalability test it is still much more noticable than for VCT. Each of these methods having been around or above 150 previously and now both Unity methods drop well below that point. Unreal faces the smallest drop with an effect appearing to be similar to adding on another light to the scalability test.

9 Discussion

In total we have compared seven different methods for global illumination using three test scenes and four different tests. Two of them were meant to evaluate the visual quality of the rendered scenes by looking at structural similarity (SSIM) and luminance conditions (HDR-DRP-2). The remaining two on the other hand provide contrast to the desire for visual quality, by allowing us to make sure that the speed required for use in a real-time setting is still achieved. This was tested through framerate tests when scaling up the number of lights, and then adding motion on top of that.

The results gained from this allows us to draw some conclusions regarding a couple of things. The first of which is how fully run-time solutions compare to their baked counterparts from a visual quality perspective, and how much it has to sacrifice from a render time perspective. Lets start with part one of that question. Looking at the results achieved we can conclude that the solutions that makes use of more pre-computations does seem to achieve a generally higher level of visual quality; at least when judging by SSIM. This shows that there still are some problems with making the lighting in the pure run-time solutions achieve equal structural similarity. The VCT engine did however manage to keep a remarkably consistent result compared to all the other methods, being the only one that did not face any real drop moving on to the more complex scenes. The results from HDR-DRP-2 were unfortunately not very impressive for any solution, with almost every test giving results around 50. There is however still a trend towards improved results when using more pre-computation.

Moving on to the second part of the question we can clearly see that all the non baked solutions face more problems. With Baked and Static keeping a relatively consistent framerate no matter the number of lights. Whereas all other solutions face quite large drops. For Unity and Unreal the results in this test still shows a manageable situation. Both maintaining a framerate above 100, which will be more than acceptable for most instances. This is true even in the Crytek Sponza case, which uses 6 lights in the visual qulity tests; even in that scene the mostly run-time calculated methods keep up relatively well on the visual quality while maintaining an acceptable framerate. That they manage this is a good sign for the field. VCT does however run into problems here, as it falls below 50 frames per second when using too many lights. So while it did manage to keep a high level of visual quality in a scene like Crytek Sponza, it ends up sacrificing so many frames per second that it becomes difficult to use in a scene with this level of lighting complexity.

Making use of the dynamics test we can also draw conclusions regarding the run-time methods ability to keep their framerates up, when put in a position where they are not able to use as many shortcuts. Since constantly moving lights makes it harder to re-use things like shadow maps for several frames. In this case VCT faced the smallest drop when compared to its stationary counterpart. It is however hard to draw many conclusions from this since the framerate was already really low. Unity and Unreal did however show a noticeable drop, indicating that previously mentioned shortcuts were important to keeping the framerate as high as they were. It does however still stay at a very usable level.

Finally there is the question of how useful the use of perceptual quality metrices Compared to previous attempts of using SSIM and HDR-VDP-2 in order to compare the visual quality of different methods made by Giovani Balen Meneghel and Marcio Lobo Netto[5], the results are still very low. In their study the implementations frequently got close to 100%. A large part of this difference is probably down to their solutions not being real-time and therefore being able to render for a longer time. The dramatic difference between our results and theirs when looking at HDR-VDP-2 may be due to it being much easier to create similar luminance for different algorithms implemented into the same program (as is done in their report) then it is when using different programs (as is done in this report).

All in all we can see that real-time global illumination shows some impressive results, even though it still has a long way to go before it can truly match the methods that are not chained to a strict time limit. We can also see how important it is to find a balance between pushing for higher visual quality and maintaining a framerate high enough to actually be able to use the term ”real-time” in a meaningful way. We also see that there are problems with using HDR-VDP-2 in a test like these that were not as prominent with SSIM.

9.1 Limitations

One of the main limitations of this study is its limited scope when considering what elements of global illumination are looked at. Since in the three test scenes we really only look at diffuse reflection and specular reflection. Meaning the effect that elements like refraction can have, are completely ignored.

Another limitation comes from the decision to keep the programs settings close to the way they are by default. This means that all these programs may potentially be able to achieve far greater results if you were to configure them in specific ways. Either making them look closer to the Blender renders or letting them deliver a higher framerate. While this decision allowed more time to be spent on the actual testing process, it means that the results may not be representative for the programs maximum level of quality.

9.2 Future Work

There are many interesting ways that the work here could be explored further in future works. First we have those mentioned in section 9.1: looking at different aspects of global illumination is necessary if we want to truly get a good overview of where the problems lie with current solutions to global illumination, and testing the programs at their maximum quality gives us a better idea of how far current technology could take us.

Another interesting direction that a study could take is attempt to test different algorithms all implemented in the same program. As previously mentioned this was done for none real-time solutions by Giovani Balen Meneghel and Marcio Lobo Netto[5] and applying that approach to real-time solutions could allow us to compare different algorithms without program specifics being in the way.

References

[1] John Amanatides. Ray tracing with cones. *SIGGRAPH Comput. Graph.*, 18(3):129–135, January 1984.

[2] J. D. Foley and Turner Whitted. An improved illumination model for shaded display.

[3] Henrik Wann Jensen. Global illumination using photon maps. In Xavier Pueyo and Peter Schröder, editors, *Rendering Techniques '96*, pages 21–30, Vienna, 1996. Springer Vienna.

[4] Rafat Mantiuk, Kil Joong Kim, Allan G. Rempel, and Wolfgang Heidrich. Hdr-vdp-2: a calibrated visual metric for visibility and quality predictions in all luminance conditions. *ACM SIGGRAPH 2011 papers on - SIGGRAPH 11*, Jul 2011.

[5] Giovani Balen Meneghel and Marcio Lobo Netto. A comparison of global illumination methods using perceptual quality metrics. *2015 28th SIBGRAPI Conference on Graphics, Patterns and Images*, Aug 2015.

[6] Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. The state of the art in interactive global illumination. *Computer Graphics Forum*, 31(1):160–188, 2012.

[7] Che Sun and Emmanuel Agu. Many-lights real time global illumination using sparse voxel octree. *Advances in Visual Computing Lecture Notes in Computer Science*, page 150–159, Dec 2015.

[8] Dean Takahashi. How pixar made monsters university, its latest technological marvel. https://venturebeat.com/2013/04/24/the-making-of-pixars-latest-technological-marvel-monsters-university/, Dec 2018.

[9] Unity Technologies. Manual: Global illumination. https://docs.unity3d.com/Manual /GIIntro.html. (visited 2019-04-18).

[10] Unity. Choosing a lighting technique. https://unity3d.com/learn/tutorials/topics/graphi cs/choosing-lighting-technique.

[11] Unity. Introducing the nvidia rtx ray tracing platform. https://developer.nvidia.com/ rtx/raytracing, May 2019.

[12] Unreal. Lightmass global illumination. https://docs.unrealengine.com/en-us/Engine/ Rendering/LightingAndShadows/Lightmass.

[13] Unreal. Movable lights. https://docs.unrealengine.com/en-US/Engine/Rendering/ LightingAndShadows/LightMobility/DynamicLights.

[14] Jose Villegas and Esmitt Ramirez. Deferred voxel shading for real-time global illumination. *2016 XLII Latin American Computing Conference (CLEI)*, Oct 2016.

[15] Z. Wang, A.c. Bovik, H.r. Sheikh, and E.p. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, Apr 2004.

[16] Chunhui Yao, Bin Wang, Bin Chan, Junhai Yong, and Jean-Claude Paul. Multi-image based photon tracing for interactive global illumination of dynamic scenes. *Computer Graphics Forum*, 29(4):1315–1324, Aug 2010.