A picture containing tool

Description automatically generatedDemaecker Daan

How do different AO methods affect real-time rendering?

Supervisor: Deweppe Alexander

Coach: Verspecht Marijn

Graduation Work 2024-2025

Digital Arts and Entertainment

Howest.be

A close up of a card

Description automatically generated

Contents

[Abstract & Key words 3](#_Toc206170138)

[Preface 4](#_Toc206170139)

[List of Figures 5](#_Toc206170140)

[Introduction 6](#_Toc206170141)

[Literature Study / Theoretical Framework 7](#_Toc206170142)

[1. Deferred rendering 7](#_Toc206170143)

[2. Global illumination 7](#_Toc206170144)

[3. Ambient occlusion 8](#_Toc206170145)

[4. Screen space ambient occlusion 8](#_Toc206170146)

[4.1 Screen-based ambient occlusion 9](#_Toc206170147)

[4.2 Horizon-based ambient occlusion 9](#_Toc206170148)

[4.3 Ground-truth ambient occlusion 9](#_Toc206170149)

[Research 10](#_Toc206170150)

[1. Implementing the different techniques 10](#_Toc206170151)

[1.1 Screen space ambient occlusion 10](#_Toc206170152)

[1.2 Horizon-based ambient occlusion 10](#_Toc206170153)

[1.3 Ground truth ambient occlusion 10](#_Toc206170154)

[2. Taking the necessary measurements 10](#_Toc206170155)

[2.1 Time needed for AO generation 10](#_Toc206170156)

[2.2 CPU Usage 10](#_Toc206170157)

[2.3 VRam usage 10](#_Toc206170158)

[2.4 Visuals 10](#_Toc206170159)

[3. Comparing results 11](#_Toc206170160)

[3.1 Time needed for AO generation 11](#_Toc206170161)

[3.2 CPU Usage 11](#_Toc206170162)

[3.3 VRam usage 11](#_Toc206170163)

[3.4 Visuals 11](#_Toc206170164)

[Discussion 12](#_Toc206170165)

[Conclusion 13](#_Toc206170166)

[Future work 14](#_Toc206170167)

[Critical Reflection 15](#_Toc206170168)

[References 16](#_Toc206170169)

[Acknowledgements 17](#_Toc206170170)

[Appendices 18](#_Toc206170171)

# Abstract & Key words

Ambient occlusion is a global illumination model and an important step in rendering to make the world seem more realistic by making the stark CGI look a lot less harsh. In non-real-time rendering or pre-rendering, this can be done quite accurately because time is less of an issue, but in real-time rendering, it’s a different story. In real-time rendering, the ambient occlusion has to be calculated every frame, so we will have to cut a few corners to make this happen. In this paper, I will be talking about and comparing 3 different types of real-time ambient occlusion. I will discuss how to implement these methods in C++, Vulkan, and which one you should pick, given the choice.

Ambient occlusion is een globale verlichtingsmethode en een belangrijke stap in renderen om de wereld meer realistisch te laten lijken door de harde CGI look wat zachter te maken. In pre-renderen is kan dit redelijk accuraat gedaan worden omdat tijd een minder groot probleem is, maar in real-time renderen is dit een ander verhaal. In real-time renderen moet de ambient occlusion elk frame opnieuw berekend worden, dus zullen we een paar hoeken moeten afsnijden. In deze paper zal ik het hebben over 3 verschillende types real-time ambient occlusion. Ik zal bespreken hoe je ze kan implementeren in C++, Vulkan en welke je best kiest, gegeven de keuze.

# Preface

During my studies at Howest – Digital Arts and Entertainment (DAE), I was introduced to graphics programming through our course that is aptly named Graphics Programming 1. In this course, we started with writing a CPU Ray tracer application from a minimal framework. Then we did the same with a CPU Rasterizer, and to conclude the course, we made a dual rasterizer using our CPU Rasterizer and a GPU rasterizer created with DirectX 11.

This sparked my interest in graphics programming a lot, and it made me want to learn more, so, as a challenge, I started creating a small engine by myself using the Rendering API Vulkan. This taught me a lot about graphics programming, and my interest grew more and more

In one of the following semesters, we had a follow-up course to Graphics Programming 1, namely Graphics Programming 2. In this course, we delved deeper into the workings of GPU rendering using Vulkan as well. This gave me many new perspectives on the API and gave me a lot of motivation to keep researching on my own.

When the assignment for this paper came around, I immediately wanted to write about a topic in the field of graphics programming. During my research, I came across the implementation of ambient occlusion at runtime and decided that this was the topic I wanted to write about.

I hope this paper will spark interest in more young programmers looking for a field to specialize in.

# List of Figures

**The list of figures lists the figures in the order in which they appear throughout the thesis. They may be numbered sequentially, or be subdivided following the chapters in which they appear.**

Figure 1: A picture showing something

Figure 2: A graph showing another thing

Figure 3.1: A tabel showing yet another thing, that appears in chapter 3.

# Introduction

Something that all graphics programmers want is to make the world they are working on look better. Some try to accomplish this through stylization, but most try to do this through realism. Lighting and shadows play a very important role in making a game seem as realistic as possible; attempts to approximate these are often done with techniques that fall under the umbrella term “global lighting”.

You can imagine “global light” as the way light works in the real world; it bounces around and affects everything it touches. A red object next to a white wall will make the wall look a bit redder. The creases between your fingers have little soft shadows on them if they are close together; even when no direct light source is nearby, those shadows are the result of light bouncing and being partially blocked.

Ambient occlusion is a method of global lighting that simulates this last example. It creates small, soft shadows when geometries are close to each other.

I will be discussing and comparing 3 different techniques to generate ambient occlusion at runtime, namely: Screen Space Ambient Occlusion(SSAO), Horizon-Based Ambient Occlusion(HBAO), and Ground Truth Ambient Occlusion(GTAO). I will be implementing these 3 methods and comparing them in terms of looks, framerate, CPU usage, and VRAM usage to see what impact they will have on your rendering.

My research question for this paper is:

“**How does my choice of ambient occlusion generation affect real-time rendering?”**

I have constructed several hypotheses surrounding this question:

* In terms of looks
  + “GTAO will look better than HBAO and SSAO”
  + “HBAO will look better than SSAO but worse than HBAO”
  + “SSAO will look worse than both GTAO and HBAO”
* - In terms of performance
  + “GTAO will perform better than HBAO and SSAO”
  + “HBAO will perform better than SSAO but worse than HBAO”
  + “SSAO will perform worse than both GTAO and HBAO”
* - In terms of memory usage
  + “GTAO and HBAO will have the same RAM usage and will both be better than SSAO”
  + “All 3 methods will have the same VRAM usage.

# Literature Study / Theoretical Framework

## 1. Rendering pipeline

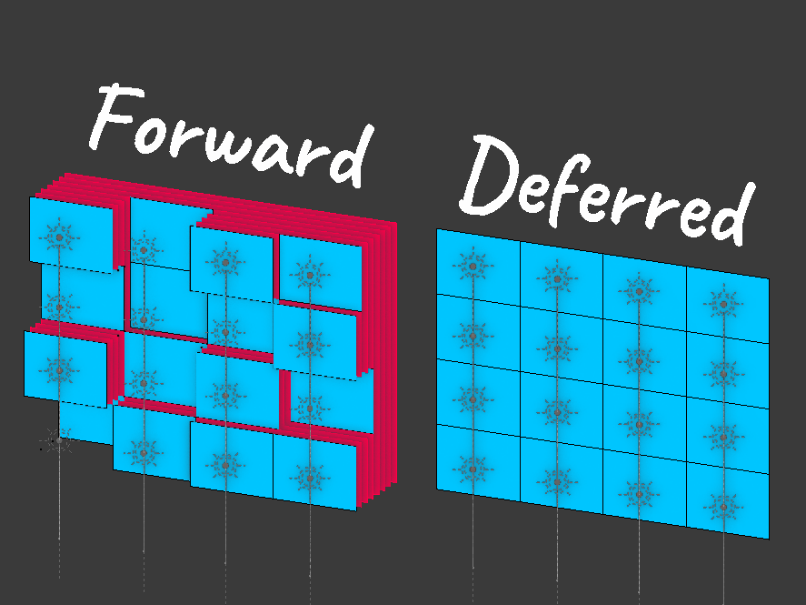
Before I get into what exactly ambient occlusion is, I will have to explain more about deferred rendering[1]. In rasterization, there are 2 types of rendering, forward and deferred. In forward rendering, each model is rendered one by one, and the lighting is calculated for each pixel.  
This is very straightforward, but it brings up one big issue. Calculating lighting is not a cheap operation, and sometimes, a pixel is rendered, the lighting for it is calculated, only for another pixel to be rendered on top of it, and the lighting calculations have to be done all over again. This is a big waste of resources.  
The solution to this problem is deferred rendering or deferred shading. In deferred rendering, we first render to a G-buffer. This buffer will hold information like normal, albedo color, position, specular value, metalness value, roughness value, and anything else you may desire.  
After rendering everything, we process the G-buffer and calculate the lighting for every pixel. This prevents multiple lighting calculations per pixel and helps increase the performance of the render, making it perfect for real-time rendering.

Figure Representation of forward and deferred rendering.

## 2. Global illumination

Global illumination[1, 2], also known as indirect illumination, is an umbrella term in the rendering world for techniques and algorithms that enhance the realism of a 3D environment's lighting, through darkening certain areas under objects and near edges, giving the illusion of color bleeding, etc. A couple of examples of these techniques are:

* Ambient Occlusion: This is the technique that I will be breaking down into detail throughout this paper.
* A group of spheres with reflection

  Description automatically generated with medium confidenceImage-based illumination[2]: Used in raytracing, when a ray bounces off a surface and then proceeds to miss, we can look up a blurred cube map representing the environment and shade the surface with a color corresponding to the map.

Figure Example of image-based global illumination

* A red ball and a green cube

  Description automatically generatedSingle-bounce global illumination for color bleeding[2]: Also used in raytracing, when a ray bounces off a surface and proceeds to hit the other surfaces, the color of surface 1 can be mixed with the color of surface 2, depending on the distance, to give the illusion of color bleeding.

Figure 3 Example of color bleeding [2]

## 3. Ambient occlusion

The idea behind ambient occlusion is that in a 3D scene, some geometry will stop the light from reaching other geometry, making corners, creases, and points surrounded by a lot of other geometry darker. This might sound like I am talking about regular shadow mapping, but there are some stark differences.

Shadow mapping uses direct light sources and checks if any objects are occluding a receiving point from the light, no matter the distance, it will be darkened.

In ambient occlusion, the area around a point in the world is checked for occluding geometry nearby, the filter region is often quite small and in the shape of a dome around the point, in the direction of the normal. No direct light is in the works here, this effect is to simulate how ambient light would bounce around the scene and get blocked by certain geometry.

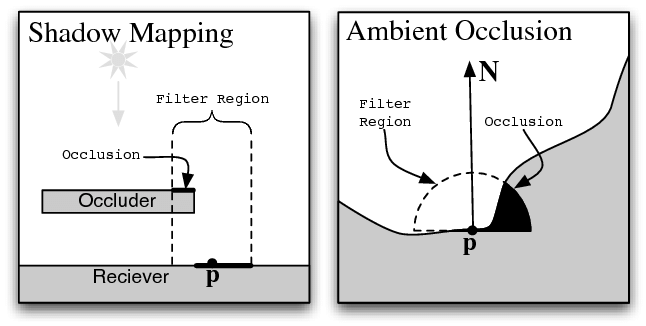
In non-real-time rendering or pre-rendering, we can calculate very accurate ambient occlusion because time is less of an issue, but in real-time rendering, this is more of a challenge. One of the ways we can get around this is by baking our ambient occlusion into textures. This means that the actual texture of the object will be darkened in certain places based on premade calculations done by programs like Autodesk Maya, Blender, or Adobe Substance Painter. But this only works in static scenes, dynamic scenes will have to calculate the ambient occlusion from scratch every single frame. This brings a whole lot of challenges to the table, and programmers all over the world have been trying to solve this problem for decades.

Figure Difference between shadow mapping and AO [4]

## 4. Screen space ambient occlusion

Because 3D environments are often quite big and complicated, it is impossible to check every single point in the world for surrounding and occluding geometry, this is where Screen Space Ambient Occlusion(SSAO) comes in the works. Instead of checking all the points in the world, it checks all the pixels visible on the screen, this is where the name Screen-Space comes from. The pixels on screen are compared to surrounding pixels to see how much the original pixel should be shaded. This makes SSAO a post-processing effect.

SSAO needs 2 main things to make the correct calculations: the view space position and the view space normals. View space means that it is relevant to the camera; the view space position of the camera would be (0,0,0), the view space position of an object 1 unit in front of the camera would be (0,0,1), etc. The same is true of view space normals. A normal facing towards the camera would hold a value of (0,0,-1), while a normal facing away from the camera would hold a value of (0,0,1).

A green and red object in a box

Description automatically generated A group of rocks in a box

Description automatically generated with medium confidence  
There are many versions of screen space ambient occlusion, but in this paper, I will focus on 3 of them. Namely: Screen-based ambient occlusion (SSAO), Horizon-based ambient occlusion (HBAO), and Ground-truth ambient occlusion(GTAO).

The result of this calculation is often quite noisy, so we apply a simple blur to it to make it more smooth.

A drawing of a building

Description automatically generatedA drawing of a person in a room

Description automatically generated

### 4.1 Screen-based ambient occlusion

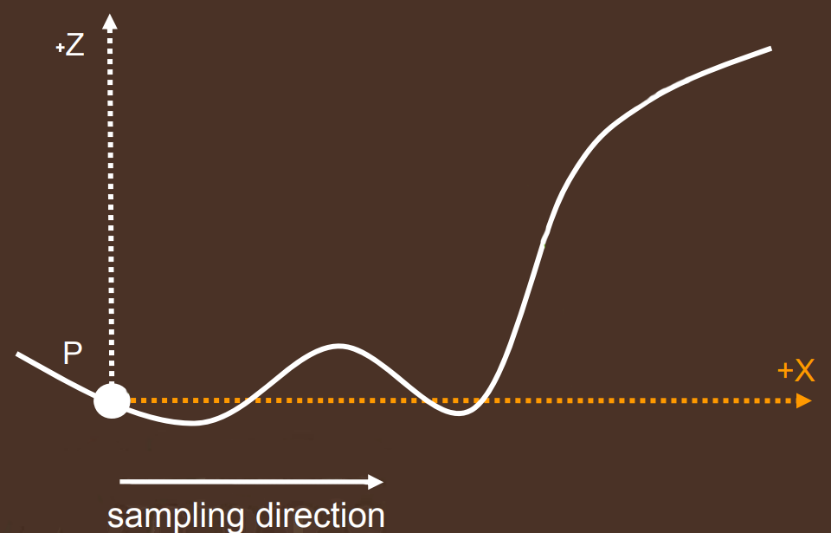
The original SSAO is simply called SSAO[5]. It was developed by Vladimir Kajalin and was used for the first time in 2007 by the game Crysis, developed by Crytek.  
The general idea is that we project a small dome around a single point in the direction of the view space normal of that point. We then project several points within that dome and check which ones are visible to the camera. We do this by calculating the view space position of said points and checking if they are visible to the camera with the help of the previously mentioned view space position map.

A diagram of a vehicle

Description automatically generated

### 4.2 Horizon-based ambient occlusion

Horizon-based ambient occlusion (HBAO) is a similar but newer version of SSAO. The algorithm starts with the information available in the view space position and normal buffers.

First, we pick a random direction, then we project the height in relation to the camera as a graph.

A graph of a curve

Description automatically generated with medium confidenceWe then pick a point further in the X-direction and check the angle between the original point and the point further on the graph. We will call this angle the Horizon angle.

We do this a couple of times, we call this “marching on the heightfield. Every time we do this, we check if the new angle is bigger than the last. The goal is to find the highest horizon angle possible.

A graph of a function

Description automatically generated with medium confidenceA graph of a function

Description automatically generated with medium confidence

### 4.3 Ground-truth ambient occlusion

# Research

In my research, I will be implementing the previously discussed methods of SSAO and comparing them in terms of looks, duration of AO generation, CPU usage, and VRAM usage. To accomplish this, I will follow these steps:

* Implement each of these Ambient occlusion techniques:
  + Screen space ambient occlusion
  + Horizon-based ambient occlusion
  + Ground Truth ambient occlusion
* Take the measurements necessary:
  + Time needed for AO generation in milliseconds
  + CPU usage in megabytes
  + VRAM usage in megabytes
  + Visuals based on questionnaire
* Compare results

## Implementing the different techniques

### Screen space ambient occlusion

### Horizon-based ambient occlusion

### 1.3 Ground truth ambient occlusion

## Taking the necessary measurements

### 2.1 Time needed for AO generation

### 2.2 CPU Usage

### 2.3 VRam usage

### 2.4 Visuals

## Comparing results

### 3.1 Time needed for AO generation

### 3.2 CPU Usage

### 3.3 VRam usage

### 3.4 Visuals

# Discussion

**In this section, you offer an interpretation of the results you obtained and try to relate them to the theoretical framework you presented. This is typically not a very long section, but obviously one of the most important ones.**

# Conclusion

**In this section, you ascertain the demonstrable outcomes of your study and outline the merits of the project for the academic field and the discourse community. This is typically not a very long section, but obviously also one of the more important ones.**

# Future work

**This section is sometimes standalone, sometimes incorporated in the conclusion. It looks at the shortcomings of the study, alternative strategies, and what could be the next course of action in the research field. This is typically not a very long section.**

# Critical Reflection

**This section is typically associated with a bachelor paper, not other forms of serious writing. It allows the student to reflect on the learning outcomes, both academically and in terms of personal growth.**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum ac quam nec arcu semper dignissim. Nulla quam magna, varius sit amet pharetra et, dictum quis elit. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Proin ullamcorper, ipsum sit amet scelerisque rhoncus, leo quam rhoncus elit, sit amet ullamcorper tellus nisi eget sapien. Suspendisse potenti. Ut non justo viverra, tempus felis vitae, elementum mi. Morbi at dui sed lacus fringilla condimentum. Duis non odio ac arcu volutpat vehicula eu et turpis.

Praesent vitae magna ante. Nulla in orci lacus. Donec quis vestibulum mi. Sed ipsum sapien, pretium maximus purus sed, bibendum consequat lectus. Aliquam porttitor dolor eu gravida vulputate. Vestibulum ut urna eget massa tincidunt ultricies. Morbi hendrerit sapien at diam tincidunt semper. Aliquam ut quam dictum quam maximus tempor sed at felis.

# References

**In this section, you list all the references you made in alphabetical order; consequently adhere to the referencing style you have chosen.**

[1]

‘LearnOpenGL - Deferred Shading’. Accessed: Aug. 15, 2025. [Online]. Available: <https://learnopengl.com/Advanced-Lighting/Deferred-Shading>

[1]

‘Global illumination’, *Wikipedia*. July 04, 2024. Accessed: Aug. 15, 2025. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Global_illumination&oldid=1232619679>

[2]

[Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=393a84a1eb9c1c5605a5944f909ff86eb3eb0eac#page=41

[3]

‘Ambient occlusion’, *Wikipedia*. May 23, 2025. Accessed: Aug. 05, 2025. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Ambient_occlusion&oldid=1291767983#cite_note-6>

[4]

‘Figure 4: Comparison of the soft shadow mapping problem to the...’, ResearchGate. Accessed: Aug. 15, 2025. [Online]. Available: <https://www.researchgate.net/figure/Comparison-of-the-soft-shadow-mapping-problem-to-the-approximate-ambient-occlusion_fig3_221257807>

[5]

‘Screen space ambient occlusion’, *Wikipedia*. Apr. 14, 2025. Accessed: Aug. 15, 2025. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Screen_space_ambient_occlusion&oldid=1285557129>

[2]

S. Graham and K. John, *Vulkan Programming Guide: The Official Guide to Learning Vulkan*. Boston Munich: Financial Times Prentice Hall, 2017.

# Acknowledgements

**In this section, you can thank people who contributed to your work in a meaningful way.**

I would like to thank some people

# Appendices

The code behind all the implementations and benchmarking for this project can be found on my GitHub trough the following link:  
<https://github.com/DaanDemaecker/DDM3-Lite-Engine>