Utilising ray marching and signed distance functions to render a scene of primitives

By Anthony Sturdy (18015368)

BSc (Hons) Computer Games Programming

Supervised by Dr. David White

May 2022

Table of Contents

[Glossary 2](#_Toc86834818)

[Key Words 2](#_Toc86834819)

[Abstract 3](#_Toc86834820)

[Introduction 3](#_Toc86834821)

[Aim 3](#_Toc86834822)

[Objectives 3](#_Toc86834823)

[Literature Review 4](#_Toc86834824)

[Research Methodology 5](#_Toc86834825)

[Project Management 5](#_Toc86834826)

[Waterfall 5](#_Toc86834827)

[Kanban 6](#_Toc86834828)

[Scrum 6](#_Toc86834829)

[Conclusion 6](#_Toc86834830)

[Platform 6](#_Toc86834831)

[Unity 6](#_Toc86834832)

[DirectX11 6](#_Toc86834833)

[Vulkan 6](#_Toc86834834)

[Conclusion 6](#_Toc86834835)

[Project Plan 6](#_Toc86834836)

[Gantt Chart 6](#_Toc86834837)

[Scope 7](#_Toc86834838)

[Analysis 8](#_Toc86834839)

[Design 9](#_Toc86834840)

[Implementation 10](#_Toc86834841)

[Testing 11](#_Toc86834842)

[Results 11](#_Toc86834843)

[Critical Evaluation 12](#_Toc86834844)

[Bibliography 13](#_Toc86834845)

[Appendices 14](#_Toc86834846)

[Appendix 1: Point-cloud rendering 14](#_Toc86834847)

# Glossary

* **3D** – Shorthand for Three Dimensional
* **2D** – Shorthand for Two Dimensional
* **SDF** – Signed Distance Function
* **CSG** – Constructive Solid Geometry
* **FOV** – Field of View
* **API** – Application Programming Interface

# Key Words

Real-time rendering, Visual effects, Graphics processing, Fractals, 3D, Lighting.

# Abstract

This research explores using ray marching as a method of rendering. Ray marching steps along a ray toward a specified direction, until an intersection is found. The distance of each step is determined by the shortest distance to any object in the scene. The signed distance function which each object uses, defines the shape of the object.

For each pixel on the screen, a direction can be determined by its position on the screen and the cameras FOV. Each object in the scene is sampled for its distance from the starting point and the shortest distance is used to determine how far the ray can be stepped without intersecting with any object. This step is repeated until the distance becomes less than a specified threshold, which is then considered to be an intersection.

This method of using a signed distance function to determine the shape of objects in the scene allows for many effects which are difficult to achieve using traditional rasterised rendering. For example, object smoothing/morphing, real-time CSG boolean operations, rendering fractals, and more. Ray marching is commonly used for rendering volumetric objects, such as clouds.

# Introduction

Recent advances in graphics processing hardware have brought ray tracing into the forefront of real-time rendering (Akenine-Möller et al., 2018). Ray tracing can be used to simulate how light travels which results in a higher fidelity render than traditional methods of rendering.

Whereas ray tracing tests for an intersection between a ray segment and objects in the scene to find the nearest intersection (Whitted & Foley, 1980), ray marching differs by stepping along a ray incrementally until an intersection is found.

A common ray marching technique is known as sphere tracing. Sphere tracing utilises signed distance functions (SDFs) to calculate the distance of each step along the ray. At each ray step, all objects are sampled for their distance to the current point, the ray can then be stepped forward by the shortest distance which ensures the ray will not travel inside, nor skip over, any object in the scene (Hart, 1996). Due to the type of SDF defining the shape of that object, it is possible to render fractals or modify the output distance to manipulate the shape or appearance of the object.

## Aim

This research aims to explore the areas needed to produce a game engine which utilises ray marching as a method of rendering a 3D scene, and to take advantage of the characteristics of ray marching to achieve various visual effects.

## Objectives

To achieve this aim, the research will enable the development of:

* A performant, real-time ray marching renderer.
* An engine framework surrounding the renderer, providing a platform to produce games/real-time applications.
* GUI controls to interact with the engine and scene at runtime.
* A 3D modelling tool which allows for primitives to be combined and visual effects to be applied to output a detailed model, which can then be imported to the engine.

# Literature Review

The definition of ray marching is to step along a ray in a specified direction until an intersection has been found. The naïve approach to implement this would be to step along the ray linearly using a fixed step size (Biagioli, 2016), this algorithm is simple but introduces potential problems. The accuracy of this algorithm is tied to the step size – a shorter step is more accurate, but computationally more expensive – which means a perfectly accurate step size is unachievable.

One of the potential problems with using a linear step size is that an intersection can be found inside the geometry, as seen in this visual representation:

Another problem is that geometry can be entirely stepped over meaning it would not be rendered. This would occur when trying to render, for example, a thin wall or an object with a high amount of detail (such as a fractal).

An improved algorithm is to ray march across distance fields, where the step size at each increment is determined by the shortest distance from the current position to any object in the scene. As the ray approaches a surface, the distance travelled becomes increasingly shorter (Hart et al., 1989), when the distance becomes lower than a specified threshold, the ray is considered to have intersected with the geometry. This method provides increased accuracy as it is impossible for the ray to pass inside, or through, any geometry. This method is referred to as sphere tracing.

This diagram is a 2D visual representation of sphere tracing, each circle’s radius represents the shortest distance to a surface at each step, which is how far the ray is stepped at each iteration along the direction of the ray.

Ray marching is commonly used for the real-time rendering of volumetric objects, such as clouds. Volumes can be simulated using traditional rasterised rendering techniques by layering multiple flat surfaces, but this lacks visual fidelity especially when up-close or inside the volume. Using ray marching for volume rendering allows for a much more realistic representation of the volume, as the ray can sample density multiple times inside the volume and calculate the colour for the current pixel (Häggström, 2018).

Media Molecule utilises SDFs for rendering in their game “Dreams”. The game allows the user to sculpt objects and apply effects to them. A point-cloud rendering technique is used to achieve a unique painterly effect by rendering the points as paint splats (Evans, 2015).

Source of the following images can be found at appendix 1.

|  |  |
| --- | --- |
| A picture containing indoor, decorated  Description automatically generated  Scene rendered with a high-density point-cloud | A map of the world  Description automatically generated with medium confidence  Scene rendered with a low-density point-cloud |

# Research Methodology

## Project Management

For this research, three project management frameworks have been considered. The reasonings for choosing between Waterfall, Kanban, and Scrum, are that each framework offers advantages and disadvantages, and they are each unique. Kanban and Scrum are both Agile frameworks, yet they are both very different in how they operate, which is why both are considered.

### Waterfall

The Waterfall methodology is a linear approach to project management. Each stage of development must be complete before the next one begins as each stage depends on the previous one. Once a stage of development is complete, it should not be returned to.

The advantages of using the Waterfall methodology are that the structure is clear and easy to understand. The following phase of development cannot be started until the current one is finished. Another advantage is that the end goal is defined early in the project’s life cycle, which helps to keep the team aligned to the goal and prevents the project from growing out of proportion.

The disadvantages of using the Waterfall methodology are, due to the linearity, it is difficult to make changes as it deviates from the planned goal. Another disadvantage is that testing does not start until after completion. (Lucid Content Team, n.d.)

### Kanban

Kanban is an agile methodology which focuses on flexibility and adaptability. Agile frameworks allow for multiple stages of development to be in progress simultaneously, as the project is broken down into small segments which allow for iteration. Kanban uses cards to represent tasks, which can be moved from stage-to-stage of development, the ‘work-in-progress’ stage of development must have limits to how many tasks can assigned simultaneously.

The advantages of using Kanban come from its flexibility. Due to each task having its own development cycle, there is constant progress being made even when other features may still be in an early phase.

Flexibility is also what gives Kanban disadvantages too, as an improperly maintained board can result in issues with development and the board can become overcomplicated. Kanban also does not have specified time-frames alike Scrum, which can more easily lead to delays or roadblocks.

### Scrum

Scrum is also an agile methodology, which allows for tasks to be completed incrementally and simultaneously. Scrum differs by using sprints of a specified time (usually two weeks) to complete tasks. Tasks are then reviewed as a team following the sprint, where they can be evaluated, and team members can sync to avoid any interference with development progress. A scrum master is assigned to ensure the team is following the correct processes and to remove obstacles.

Scrum offers more transparency and visibility across the entire workflow than other frameworks, which is an advantage as changes are easy, and the team is always in sync. This leads to better productivity for a team.

The disadvantages of using Scrum for this research stems from Scrum having many different roles. Scrum can only be taken full advantage of when working in a team, and due to this research being a solo project, Scrum would be ineffective (PLANETTOGETHER, 2019).

### Conclusion

Kanban will be the methodology used for research and development throughout this project.

Waterfall will be too restrictive for this project due to its linearity. There would be a risk of planning and researching more than is viable to implement within the time constraint of the project, which is not a problem that is present for other, agile, methodologies.

The Scrum methodology is well-suited for use in a team, as it allows for iteration and simultaneous stages of development whilst keeping all members in sync. Due to the nature of this research project being a solo effort, Scrum would not be suitable.

Kanban is a flexible methodology which doesn’t require a team to be fully taken advantage of. Tasks can be researched and developed simultaneously, which mitigates the risk of planning more than is able to be implemented within the time constraint.

## Platform

Three platforms have been considered for the development of this project. Each platform is considerably unique and offer advantages and disadvantages. DirectX11 and Vulkan are both graphics APIs which would allow for the creation of a rendering engine or other applications which need to utilise graphics hardware. Unity differs, as it is already a complete real-time development platform (Unity Technologies, n.d.) and offers many tools which assist with the creation of games or other real-time rendered media.

### Unity

Unity is a real-time development platform which includes many tools to assist the creation of games or other real-time rendered media. Ray marching in Unity would be possible, as it allows for user-made shaders and control over the rendering pipeline via the scriptable render pipeline.

The advantage of using Unity for the development of this project would be the many tools already built into the engine, which could be utilised with the ray marching renderer.

These tools and extra and unused features could lead to overhead on performance, which would be one of the disadvantages of using Unity for this project. This would not likely be a problem if using DirectX or Vulkan as the engine would be built from the ground-up to only include what is necessary.

### DirectX11

DirectX11 is an API which can be used by an application to utilise graphics hardware, it supports multiple shader types, such as pixel and compute shaders (Microsoft, 2020), which could be utilised for ray marching. An alternative to DirectX11 would be OpenGL, which is comparable in terms of functionality and design.

The advantages of using DirectX11 are that there would not be unused feature overhead which could be present when using Unity. DirectX is relatively (to Vulkan) high level, which means development progress would be faster than using an API such as Vulkan.

The disadvantages of using DirectX11 are that there would be more work than if using Unity, as some of the existing tools would have to be made from scratch. The performance also may not be as fast as if using a lower-level API such as Vulkan, as Vulkan offers more control over the entire rendering process/pipeline.

### Vulkan

Vulkan is also an API to utilise graphics hardware, it differs from DirectX11 as it is much lower-level and verbose, giving the developer more control over the entire rendering process. An alternative to Vulkan would be DirectX12, which is similarly low-level and also offers fine control to the developer.

The advantage of using Vulkan over another API such as DirectX11, is that the lower-level control which the API provides can lead to performance increases if utilised properly.

This control is also one of Vulkan’s disadvantages. Having more control over the API could cause the program, if improperly used, to operate slower than a higher-level API. The verbosity of the code also makes using Vulkan much more complex than a higher-level API, which is likely to slow the development progress.

### Conclusion

The platform which this project will be built upon is DirectX11, as it strikes a balance between being lightweight to avoid unnecessary overhead, but high-level enough that the project will be able to progress at a reasonable rate.

Using DirectX11 would also enable research into implementing other standard features of a game engine, for example: Gizmos, Scene Management, Camera Controls, etc.

# Project Plan

The project planning process (actual plan should be placed in an appendix), scope with a statement of the evaluation criteria and how the project will proceed. Reflect upon your Aim and Objectives.

## Gantt Chart

See appendix 1: Gantt Chart *(for example)*

## Scope

Is the project achievable?

# Analysis

A description of the choice of an analysis or problem-solving method. It is important to describe the process by which the method is chosen to show that it is appropriate for the problem situation.

A narrative description of the application of the analysis method, indicating the problems which arose during this process and how they were identified and overcome. Obviously, most projects will include models, charts, or diagrams at this stage. These may be included in the chapter or in an appendix.

In short, investigate all possible options available to you in order to successfully achieve the scope of the project in the time given.

# Design

Here you will justify your chosen tools and processes from your previous research into the background of the project and the options available to you in the analysis.

Identify and justify:

* The choice of an appropriate method.
* Your experience of its application.

# Implementation

The Implementation chapter should show clearly how the solution to the problem is realised. As with the other parts of the project, the selection of the implementation method should be described and justified. Also, the nature of the solution will depend on the nature of the project and the course.

# Testing

This chapter should address the evaluation of the solution against its objectives and success criteria.

Consider:

* A description of the testing strategy and the choice of testing method.
* The planning and application of the tests. How have you concluded this is the correct type of test to run?

# Results

The conclusions that may be drawn from the results of the tests and any modifications to the design and implementation that could be recommended.

What are you results?

What analysis can you identify from them?

Did you find something you did not expect?

Or was it exactly what you expected?

# Critical Evaluation

This chapter is of **crucial importance** to the whole work. It deals with the success of the project in academic terms, rather than the success criteria for the solution. Even the best analysis, design and implementation will be let down by an inadequate critical evaluation. The examiners will look at this chapter most carefully when determining the success (or otherwise) of the project. Although the exact nature of the evaluation will vary between projects, it is possible to identify certain issues which should be addressed:

* Your evaluation of the degree of success in carrying out the project
* What you have learned by doing the project
* What you would do differently if the project were to be repeated
* Any extra features you would recommend if the project could be extended
* The value to you of the learning process and the extent to which the project has added to your professional and academic expertise
* What future projects open the next chapter, should this projects research be continued further into Master Degree for example?

# Bibliography

Akenine-Möller, T., Haines, E., Hoffman, N., Pesce, A., Hillaire, S. & Iwanicki, M. (2018). *Real-Time Rendering*. 4th Ed. [Online]. A K Peters/CRC Press. Available from: https://www.realtimerendering.com/Real-Time\_Rendering\_4th-Real-Time\_Ray\_Tracing.pdf. [Accessed: 25 October 2021].

Biagioli, A. (2016). *Raymarching Distance Fields: Concepts and Implementation in Unity*. [Online]. 1 October 2016. Available from: https://adrianb.io/2016/10/01/raymarching.html. [Accessed: 30 October 2021].

Evans, A. (2015). *Learning from failure: A Survey of Promising, Unconventional and Mostly Abandoned Renderers for ‘Dreams PS4’, a Geometrically Dense, Painterly UGC Game*. [Online]. Available from: http://advances.realtimerendering.com/s2015/AlexEvans\_SIGGRAPH-2015-sml.pdf. [Accessed: 2 November 2021].

Häggström, F. (2018). *Real-time rendering of volumetric clouds*. [Online]. Umeå. Available from: http://www.diva-portal.org/smash/get/diva2:1223894/FULLTEXT01.pdf. [Accessed: 2 November 2021].

Hart, J.C. (1996). Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*. [Online]. 12 (10). Available from: https://www.researchgate.net/publication/2792108\_Sphere\_Tracing\_A\_Geometric\_Method\_for\_the\_Antialiased\_Ray\_Tracing\_of\_Implicit\_Surfaces. [Accessed: 26 October 2021].

Hart, J.C., Sandin, D.J. & Kauffman, L.H. (1989). Ray tracing deterministic 3-D fractals. *ACM SIGGRAPH Computer Graphics*. [Online]. 23 (3). Available from: https://www.cs.drexel.edu/~david/Classes/Papers/rtqjs.pdf. [Accessed: 30 October 2021].

Lucid Content Team (n.d.). *The Pros and Cons of Waterfall Methodology*. [Online]. Available from: https://www.lucidchart.com/blog/pros-and-cons-of-waterfall-methodology. [Accessed: 7 November 2021].

Microsoft (2020). *Graphics Pipeline*. [Online]. 4 November 2020. Available from: https://docs.microsoft.com/en-us/windows/win32/direct3d11/overviews-direct3d-11-graphics-pipeline. [Accessed: 8 November 2021].

PLANETTOGETHER (2019). *Agile Methodologies: Kanban Vs Scrum - Advantages and Disadvantages*. [Online]. 31 December 2019. Available from: https://www.planettogether.com/blog/agile-methodologies-kanban-vs-scrum-advantages-and-disadvantages. [Accessed: 7 November 2021].

Unity Technologies (n.d.). *Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine*. [Online]. Available from: https://unity.com/. [Accessed: 8 November 2021].

Whitted, T. & Foley, J.D. (1980). An Improved Illumination Model for Shaded Display. *Graphics and Image Processing*. [Online]. 23. p.pp. 343–349. Available from: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.156.1534. [Accessed: 26 October 2021].

# Appendices

## Appendix 1: Point-cloud rendering

Images by Media Molecule:

<http://advances.realtimerendering.com/s2015/AlexEvans_SIGGRAPH-2015-sml.pdf>