Real-time Rendering of 3D “Fractal-like” Geometry

Deliverable 1: Final Year Dissertation

Solomon Baarda

Solomon Baarda  [Company address]

Abstract

Ray tracing getting popular, ray marching

I, Solomon Baarda confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed:

Date:

Table of Contents

[1 Introduction 5](#_Toc86311882)

[1.1 Aims & Objectives 5](#_Toc86311883)

[1.2 Project Description 5](#_Toc86311884)

[2 Literature Review 5](#_Toc86311885)

[2.1 Rendering Methods 6](#_Toc86311886)

[2.2 Ray Tracing 6](#_Toc86311887)

[2.2.1 Benefits and Limitations 6](#_Toc86311888)

[2.3 Ray Marching 7](#_Toc86311889)

[2.3.1 Benefits and Limitations 7](#_Toc86311890)

[2.3.2 Signed Distance Function 7](#_Toc86311891)

[2.3.3 Primitives 8](#_Toc86311892)

[2.3.4 Alterations & Combinations 8](#_Toc86311893)

[2.3.5 Surface Normal 10](#_Toc86311894)

[2.3.6 Fractals 11](#_Toc86311895)

[2.3.7 Collision detection 11](#_Toc86311896)

[2.4 Existing Projects 12](#_Toc86311897)

[2.4.1 Fragmentarium 12](#_Toc86311898)

[2.4.2 Smallpt 12](#_Toc86311899)

[2.4.3 Ray Tracing in One Weekend 12](#_Toc86311900)

[3 Requirements Analysis 12](#_Toc86311901)

[3.1 Use Cases 12](#_Toc86311902)

[3.2 Requirements Specification 12](#_Toc86311903)

[3.2.1 Functional Requirements 12](#_Toc86311904)

[3.2.2 Non-functional Requirements 13](#_Toc86311905)

[4 Software Design 13](#_Toc86311906)

[4.1 Structure 13](#_Toc86311907)

[4.2 Technologies 13](#_Toc86311908)

[5 Evaluation Strategy 13](#_Toc86311909)

[6 Project Plan 13](#_Toc86311910)

[6.1 Design Methodology 14](#_Toc86311911)

[6.2 Legal, Ethical & Social Issues 14](#_Toc86311912)

[6.3 Risk Analysis 14](#_Toc86311913)

[6.4 Timetable 14](#_Toc86311914)

[7 References 14](#_Toc86311915)

[8 Appendices 14](#_Toc86311916)

# Introduction

<https://github.com/SolomonBaarda/dissertation>

## Aims & Objectives

The aim of this project is to develop a prototype real-time rendering engine, capable of displaying complex 3D “fractal-like” geometry. The performance of the engine will be benchmarked across various systems to determine whether the “real-time” aspect of the project has been achieved.

Create a realtime 3d fractal rendering engine

Must have good performance to be realtime

Find a good balance between looking good and performance

Need approximations

## Project Description

Is the topic meaningful, complex and challenging?

# Literature Review

How relevant is the literature that is covered?

• Is there missing material?

• Is it well structured?

• Are good quality sources used and properly cited?

• How strong are the comparative and critical aspects?

• Is the literature review of an appropriate length?

## Rendering Methods

## Ray Tracing

Ray tracing is a technique used to render 3D environments, where one or more “rays” are sent out

Ray tracing is a method of rendering 3D environments, often with photorealistic detail. In ray racing, a ray (simply a line in 3D space) is extended until it collides with the surface of an object. From there, the ray can be absorbed or reflected by the surface, taking into consideration light absorption, reflection, refraction, and fluorescence. Ray tracing is a computationally expensive

Ray surface intersection functions

of each pixel of the camera.

Background of ray tracing

Well suited for polygon meshes, or simple primitives

Ideal for very realistic graphics

not real-time

hybrid approaches can be used for real-time

Unsuitable for self-repeating shapes, infinite precision, non-Euclidean geometry where an object can be seen multiple times from the same view

### Benefits and Limitations

Benefits

Realistic, easy to parallelise,

Performance

Hard to get real time

## Ray Marching

### Benefits and Limitations

Ray marching is a variation of ray tracing, which differs in the method of detecting collisions between the ray and objects. Ray marching uses an iterative approach, where the current point is moved/marched along the ray in small increments until it lands on the surface of an object.

This approach is more computationally complex in the worst case when compared to ray tracing, however, it does provide several benefits. Most notably, ray marching does not require a surface collision function like ray tracing does, so it can be used to render geometry for which these functions do not exist. This allows more complex shapes to be rendered.

Ray marching also has infinite precision when zooming in, as no polygons are used.

Use ray marching optimisation techniques

Ray marching – lots of effects (ambient occlusion, hard or soft shadows, glow, fog) are free

To determine how feasible a real-time rendering of 3D environments using ray marching.

### Signed Distance Function

A distance function is a function which given any point in 3D space, will return the distance to the surface of the closest object. A signed distance function is simply a distance function which contains a positive sign if the point is outside of the object, and a negative sign if the point is inside of the object. If a distance function returns 0 for any point, then the point must be exactly on the surface of an object.

DIAGRAM TODO

The sign returned by the distance function is very useful as it allows the ray marcher to determine if a camera ray is inside of an object or not, and from there it can use that information to render the objects differently.

In the scene below

![Shape

Description automatically generated with medium confidence](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAmgAAAFaCAMAAACda1FAAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAGUExURbq6ugAAADaON5cAAAACdFJOU/8A5bcwSgAAAAlwSFlzAAAh1QAAIdUBBJy0nQAAAOZJREFUeF7twTEBAAAAwqD1T20LLyAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAOCuBkIXAAFtyuL2AAAAAElFTkSuQmCC)

Figure 1 - DF and SDF

### Primitives

Signed distance functions exist for most primitive 3D shapes, such as a sphere, box, plane etc.

For a sphere with radius R, positioned on the origin 0,0,0, the

Deriving an SDF

### Alterations & Combinations

Signed distance functions can be translated, rotated, and scaled.

Signed distance functions can be combined using union, subtraction, and intersection operations.

A picture containing dishware

Description automatically generated

Figure 2 - Union of Sphere and Box

Icon

Description automatically generated with low confidence

Figure 3 – Intersection of Sphere and Box

Signed distance functions can also be combined using a version that uses smoothing.



Figure 4 - Smooth Union of Sphere and Box

There are several alterations that can be applied to primitives once we have their signed distance function. A primitive can be elongated along any axis, its edges can be rounded, it can be extruded, and it can be “onioned” – a process of adding concentric layers to a shape. All these operations are very cheap.

Signed distance functions can also be repeated, twisted, bent, and surfaces displaced using an equation e.g., a noise function or sin wave.

### Surface Normal

The surface normal of any point on the surface of an SDF can be determined by probing the SDF on each axis, using an arbitrary epsilon value.

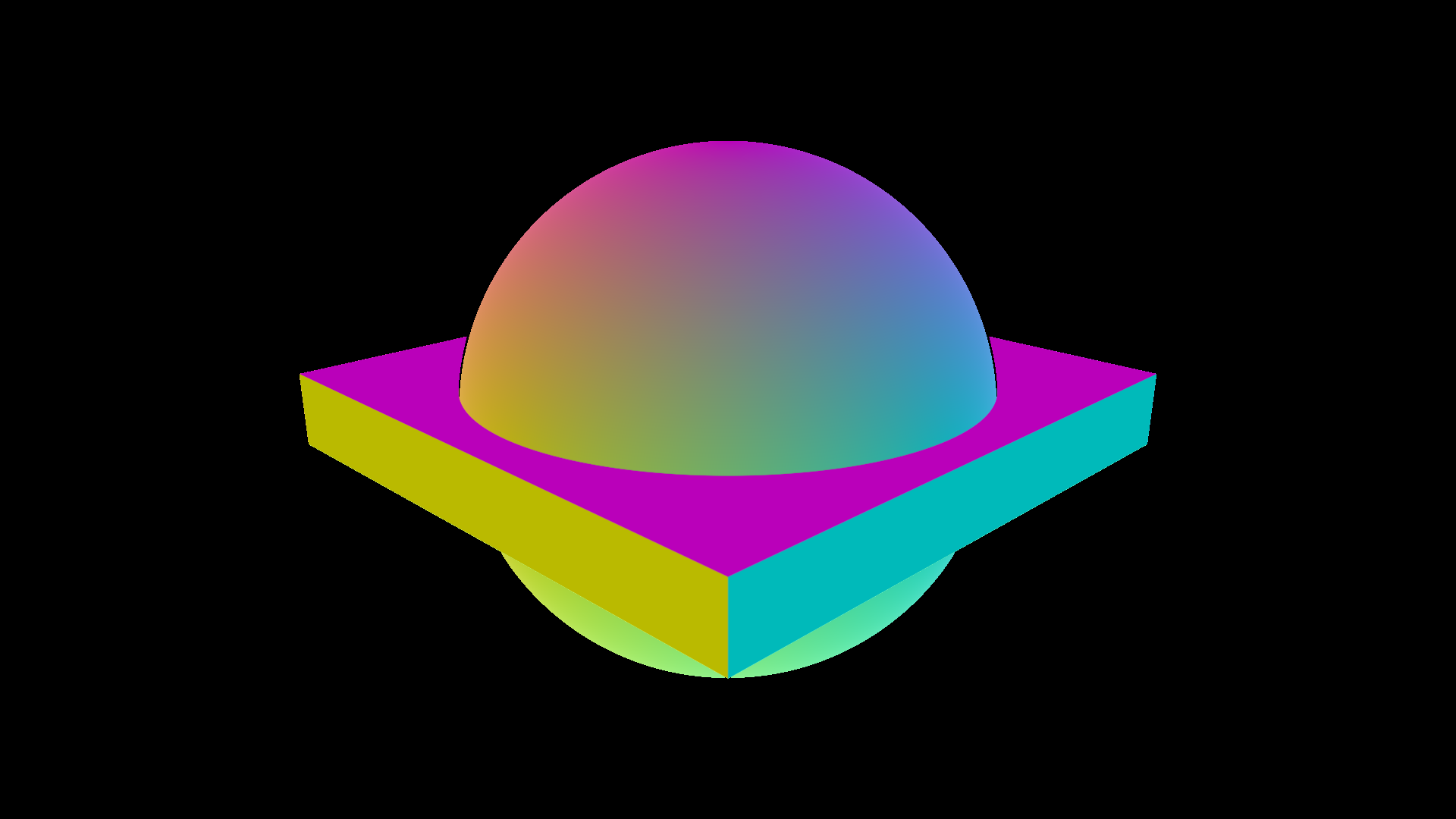


Figure 5 - Surface Normal of Sphere and Box Scene

### Fractals

In mathematics, a fractal is a complicated pattern built from simple repeated shapes, which are reduced in size every time they are repeated. <https://dictionary.cambridge.org/dictionary/english/fractal> These shapes are self-similar, though not often symmetrical.

The idea of fractal geometry appeared in the late 1970s, inspired through the work of Benua Mandelbrot and his book “Fractals: form, chance and dimension”, released 1977. This book introduced the concept of a “fractal dimension”, a measure of the complexity of how the detail in a pattern will change in respect to the scale at which it is measured.

How they are calculated – running sum etc

3D fractals

### Collision detection

Collision detection is possible

Marble marcher

## Existing Projects

### Fragmentarium

<https://github.com/Syntopia/Fragmentarium>

<https://github.com/3Dickulus/FragM>

### Smallpt

<https://www.kevinbeason.com/smallpt/>

### Ray Tracing in One Weekend

<https://github.com/RayTracing/raytracing.github.io>

# Requirements Analysis

## Use Cases

## Requirements Specification

### Functional Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Priority | Justification |
|  |  | The application must be capable of rendering scenes in real time | MUST |  |
|  |  | The application must |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

### Non-functional Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Priority | Justification |
|  | Executable | The application must run from a compiled executable | MUST |  |
|  | Display resolutions | The application must support the following common display resolutions: 1366x768, 1920x1080, 2560x1440 and 3840x2160 |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | Maximum framerate | The application should limit the maximum framerate |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Software Design

## Structure

## Technologies

# Evaluation Strategy

# Project Plan

## Design Methodology

## Legal, Ethical & Social Issues

A well-researched consideration of any Professional, Legal, Ethical, and Social Issues pertinent to the project. (e.g. codes of conduct (BCS), codes of practice, standards, computer law, ethical decision making, intellectual property, social aspects, copyright, data protection, and so on)

## Risk Analysis

## Timetable

# References

# Appendices