MANCHESTER
1824
The University of Manchester

Building a Ray-Tracing Engine on Sparse Voxel Grids

Dissertation submitted to The University of Manchester for the degree of
BSc. (Hons) in Computer Science
in the Faculty of Science and Engineering**Year of submission**
2024

Student ID
10834225

Department of Computer Science

Contents

Word count: many

Abstract

This is abstract text.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Declaration of originality

I hereby confirm that this dissertation is my own original work unless referenced clearly to the contrary, and that no portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Intellectual property statement

i The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made *only* in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see `http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420`), in any relevant Dissertation restriction declarations deposited in the University Library, and The University Library's regulations (see `http://www.library.manchester.ac.uk/about/regulations/_files/Library-regulations.pdf`).

Acknowledgements

What should I aknowledge?! I am citing [1], and [2] hell

1 Introduction

1.1 Motivation

1.2 Aims

1.3 Objectives

1.4 Report structure

2 Background and Literature Review

2.1 Rendering engines

Graphics engines serve as the core software components responsible for rendering visual content in applications ranging from video games to scientific simulations and visual effects in movies. Engines abstract the complexities of rendering by providing developers with high-level tools and interfaces to represent digital enviorments.

The evolution of rendering engines over time reflects the advancments in computational techniques and hardware capabilities enabling more realisitic and immersive experiences

2.1.1 Primitves

At the heart of any graphical engine is the concept of primitives, the simplest forms of graphical objects that the engine can process and render. Primitives are builiding blocks from which more complex shapes and scenes can be constructed.

Polygons, particullary triangles, are the most commonly used primitives in 3D graphics. This is owed to their simplicity and flexibility, allowing the construction of virtually any 3D shape through *tesselation*. Polygonal meshes define the surfaces of objects in a scene, with each vertex of a polygon typically associated with additional information such as color, texture coordinates, and normal vectors for lighting calculations.

Voxels represent a different approach to defining 3D shapes, they are essentially three-dimensional pixels. Where polygons define surfaces, voxels establish volume, with each voxel being able to contain color and density information. This charactheristic makes voxels particuallry well-suited for rendering scenes with materials that have intricate internal structures, such as fog, smoke, fire and fluids.

2.1.2 Ray-tracing vs. Rasterization

Rendering engines can utilise two main rendering techniques for rendering scenes: ray tracing and rasterization, both having their advantages and trade-offs.

Rasterization is the most widespread technique used in real-time applications. It converts the 3D scene into a 2D image by projecting vertices onto the screen, filling in pixels that make up polygons, and applying textures and lighting. Over the development of the industry of graphics programming, graphics hardware has become extremely effcient at performing rasterization, making it the standard for video games and interactive applications.

Ray-Tracing, in contrast, simulatesthe path of light as rays travelling through a scene to produce images with realistic lighting, shadows, reflections, and refractions. Ray tracing is computationally intensive but yields higher-quallity images, makeing it favored for applications where visual fidelity is critical. However, recent advancements in hardware have begun to bring real-time ray tracing to interactive applications.

2.2 Representing voxels

To efficiently represent and manipualte voxels in program memory, various data structures can be employed. Each method entails trade-offs between memory usage, access speed and complexity of implementation. Access speed referes to the time complexity of querying the datastructure at an arbitray point in space to retrieve a potential voxel.

2.2.1 Voxel grids

A voxel grid is the most straightforwad and intuitive approach to representing volumetric data. The 3D space is divided into a regular grid of voxels, each holding information such as color, material properties, or density. This method provides direct $O(1)$ access to voxel data.

However, this simplicity comes at a signficant disatvantage: memory consumption. As the bounding volume or the level of detail of the of the scene increases, the memory required to store the voxel grows by $O(N^3)$. Additionally, empty space can occupy a majority of the memory space. For example, consider a scene with two voxels that are a million units appart in all axes. A voxel grid would have to store all the empty voxels inbetween; 10^{18} memory units reserved, 2 of which carry usefull data. This limitation makes the naive voxel grids impractical for large or highly detailed scenes.

2.2.2 Hierarchical voxel grids (N-trees)

To mitigate this issues, hierarchical grids, such as octrees, are employed. An octree is a tree data structure where each node represents a cubic portion of 3D space and has up to eight children. This division continues recursively, allowing for varying levels of detail within the scene: larger volumes are represented by higher-level nodes, while finer details are captured in lowere levels.

The primary advantage of using an octree is spatial efficiency. Regions of the space that are empty or contain uniform data can be represented by a single node, significantly reducing the memory footprint. Furthermore, octrees facilitate efficient querying operations, such as collision detection and ray tracing, by allowing the algorithm to quickly discard large empty or irrelevant regions of space.

Hierarchical grids introduce complexity in terms of implementation and management. Operations such as updating the structure or balancing the tree to ensure efficient access can be more challenging compared to uniform grids. Another sacrifice is access-time, as querying an arbitrary region of space can entail walking down the tree for several levels. Nonetheless, for applications requiring large, detailed scenes with a mix of dense and sparse regions, the benefits of hierarchical representations often outweigh these drawbacks. This is why N-trees are frequently used in voxel engines.

2.2.3 VDB

References

[1] A. J. Casson and E. Rodriguez-Villegas, "Toward online data reduction for portable electroencephalography systems in epilepsy," *IEEE T. Biomed. Eng.*, vol. 56, no. 12, pp. 2816–2825, 2009 (cit. on p. 6).

[2] A. J. Casson and E. Rodriguez-Villegas, "Toward online data reduction for portable electroencephalography systems in epilepsy," *IEEE T. Biomed. Eng.*, vol. 56, no. 12, pp. 2816–2825, 2009 (cit. on p. 6).

Appendices

A Project outline

Project outline as submitted at the start of the project is a required appendix.

B Risk assessment

Risk assessment is a required appendix. Put here. And there as well