School of Computing
FACULTY OF ENGINEERING
AND PHYSICAL SCIENCES

Procedural Generation of Infinite Voxel Cave Based on Weighted Multi-Noise

Simiao Wang

Submitted in accordance with the requirements for the degree of MSc High Performance Graphics and Games Engineering

2023/2024

The candidate confirms that the following have been submitted.
<As an example>

Items	Format	Recipient(s) and Date
Beliverable 1, 2, 3	Report	SSO (DD/MM/YY)
Participant consent forms	Signed forms in envelop	SSO (DD/MM/YY)
Deliverable 4	Software codes or URL	Supervisor, Assessor (DD/MM/YY)
Deliverable 5	User manuals	Client, Supervisor (DD/MM/YY)

Type of project: _____

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of Student) _____

© 2023/2024 The University of Leeds and Simiao Wang

Summary

With the advancement of game development technology, procedural generation techniques are playing an increasingly crucial role in creating virtual environments, particularly in the fields of terrain generation and cave simulation. Utilizing mathematical algorithms and noise functions, procedural techniques can achieve highly variable environments, offering players experiences filled with exploration and novelty in each game session. Voxel representation is pivotal in the storage and depiction of terrain, serving not only to describe the geometric structure but also to facilitate real-time rendering. Proper voxelization processes effectively optimize scene performance.
In the context of cave generation, noise-based techniques simulate the intricate natural structures of underground spaces through the weighted combination of multiple noise functions. This approach enables the automatic generation of cave systems that exhibit diversity and realism.
This paper delves into the methodology and implementation of procedurally generating infinite voxel caves based on weighted multi-noise. By optimizing combinations of various noise functions, it achieves heightened naturalism and visual fidelity in terrain generation. Additionally, performance optimization is achieved through surface extraction techniques, facilitating dynamic loading and unloading of voxel worlds using block-based data structures.

Acknowledgements

I would like to sincerely thank Professor Marcus for providing the inspiration for my research direction and for guiding me in developing my evaluation methods. His support and insights have been invaluable to the completion of this thesis.

Contents

Chapter 1

Introduction

1.1 Context

In recent years, procedural generation has become a fundamental technique in the development of virtual environments, particularly within the gaming industry. The application of procedural generation techniques enables the creation of expansive, complex, and meticulously detailed virtual environments with minimal manual input, thereby offering players distinctive and immersive experiences with each engagement with the content.

Notable examples include the video games *Minecraft* and *Terraria*. *Minecraft*, developed by Mojang Studios, employs procedural generation to create an infinite world comprising a variety of terrains, including mountains, caves, forests, and oceans. Similarly, *Terraria*, developed by Re-Logic, employs procedural generation to construct a two-dimensional world filled with caves, dungeons, and diverse biomes. The randomness and variety inherent in the generated worlds contribute significantly to the game's replayability, ensuring that each player's adventure is unique.

1.2 Problem statement

The primary objective of this project is to develop a dynamic voxel cave terrain generation system using C++ and OpenGL. This system faces several challenges:

Dynamic loading and unloading of terrain chunks is a major challenge, requiring real-time generation and removal as the player moves through the environment. Efficient management of these chunks is crucial, involving their creation, updating, and removal without impacting performance. Multi-threading is used to handle terrain generation in parallel, reducing the load on the main thread.

Another key aspect is using noise functions for terrain generation. Effectively blending Perlin and Simplex noise to create realistic cave structures, and dynamically adjusting their influence, is essential for achieving desired results.

Performance optimization is critical, with techniques such as removing invisible voxel faces and excluding out-of-view chunks necessary to maintain high frame rates. Efficient implementation of these techniques is vital.

1.3 Aim and objectives

The aim of this project is to develop a dynamic voxel cave terrain generation system using C++ and OpenGL. The system will create and manage cave structures using Perlin and Simplex noise. Users can control the camera using mouse and keyboard to navigate the scene, view the generated cave terrain from different angles, and modify noise weights and control terrain generation status through the user interface.
The main objectives for completing this project are:

1. Develop a dynamic voxel terrain generation system that uses Perlin and Simplex noise to create cave structures. This includes implementing algorithms for generating terrain chunks based on noise values.

2. Implement chunk management for real-time loading and unloading of terrain. The system should handle the creation, updating, and removal of chunks while minimizing impact on overall performance. Multi-threading will be employed to manage terrain generation concurrently with rendering.

3. Optimize performance by applying techniques such as voxel face culling and frustum culling. Ensure high frame rates and smooth performance through effective removal of redundant and invisible voxel data.

4. Create a user interface with ImGui to allow real-time adjustments of terrain generation parameters, including noise function weights, and to display performance metrics such as frame rate.

5. Ensure seamless integration of system components by enabling real-time changes to terrain generation settings, supporting functionality to start/stop chunk loading, and providing the option to reset the camera position.

1.4 Deliverables

The expected deliverables are as follows:

1. A dynamic voxel cave generation system developed with C++ and OpenGL. It uses Perlin and Simplex noise to create voxel-based cave systems, supports real-time chunk loading and unloading, and optimizes performance with techniques like frustum culling and voxel face removal. The system includes a user interface for real-time adjustments and performance monitoring.

2. A GitLab repository that contains the source code of the system.

3. Developer documentation.

4. The MSc project report.

Chapter 2

Background Research

2.1 Literature Survey

2.1.1 Procedural Terrain Generation

2.1.2 Noise

2.1.3 Voxel

2.1.4 OpenGL

2.2 Methods and Techniques

2.2.1 Noise Function

2.2.2 Data Structure for Storing World Data

2.2.3 Performance Optimization Techniques

2.3 Choice of methods

2.3.1 Perlin Noise and Simplex Noise

2.3.2 Chunk Data Structure

2.3.3 Back-Face Culling, Neighbor Face Culling and Frustum Culling

Chapter 3

Software Requirements and System Design

3.1 Software Requirements

3.1.1 abcdefg

3.2 System Design

Chapter 4

Software Implementation

4.1 abcdefg

4.1.1 abcdefg

Chapter 5

Software Testing and Evaluation

5.1 abcdefg

5.1.1 abcdefg

5.1.2 abcdefg

Chapter 6

Conclusions and Future Work

6.1 Conclusions

6.2 Future Work

6.2.1 Terrain Generation Improvements

6.2.2 Performance Optimization

6.2.3 Chunk file storage

6.2.4 Multi-threading Optimization

References

[1] D. Parikh, N. Ahmed, and S. Stearns. An adaptive lattice algorithm for recursive filters. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(1):110–111, 1980.

Appendices

Appendix A

External Material

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Appendix B

Ethical Issues Addressed