BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2025
http://excel.fit.vutbr.cz

A New Interactive Algorithm For Converting a Voxel Model To a Mesh In Unreal Engine

Pavel Balusek*

Abstract
Efficient voxel-to-mesh conversion remains a key challenge for real-time rendering in voxel engines. This paper introduces Run Directional Voxel Meshing, an interactive algorithm that generates polygonal meshes directly from RLE-compressed voxel data with minimal traversal overhead. The method enables seamless dynamic updates, supporting real-time voxel model manipulation with significant memory savings. Integrated into Unreal Engine 5.4 as a plugin using the RealtimeMeshComponent, this solution achieves reduced memory consumption and interactive frame rates even for voxel models. Comparative benchmarks against traditional meshing approaches validate the algorithm's usefulness and performance advantages, paving the way for future optimizations including dynamic LOD generation and enhanced compression strategies.

*xbalus03@stud.fit.vut.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Voxel models provide a powerful way to represent volumetric data, particularly in games, simulations, and procedural generation. However, game engines like Unreal Engine primarily operate on polygonal meshes, necessitating a conversion process that is often a major performance bottleneck.

This thesis found a split in the development of voxel engines, after popularization of voxel engines after release of Minecraft. Academic branch about voxel rasterization led by GigaVoxels [1] and second branch community-driven branch led by voxel enthusiats community. This community blogs were started by Mikola Lysenko blog [2] where he discribed basic principles of voxel engine in Minecraft. The fundamental principle is to convert voxels into polygonal meshes. This process was originally called meshing in voxel engines, however after years of community development and more general meaning of meshing it got misinterpreted. The term for the process of converting voxel models to a mesh is, in this thesis, called **Voxel Meshing**.

This thesis focuses on developing a new interactive algorithm that can efficiently convert voxel models into polygonal meshes while supporting real-time interaction in Unreal Engine. The goal is to minimize memory consumption, reduce polygon count, and maintain conversion speeds suitable for interactive applications.

Existing solution for voxel meshing is Voxel Plugin [3]. Voxel Plugin is used in performance profiling to measure stats of RLE Run Directional Voxel Meshing. Rendering using different solutions can be seen in Figure 3.

2. Implementation

My solution introduces Run Directional Voxel Meshing, a new algorithm that processes RLE-compressed voxel models in a single traversal, creating quads during iteration. This method supports seamless real-time updates and interaction while maintaining efficiency and memory compactness. This process of traversing RLE sequences and creating quads from it can be seen in Figure 1 and Figure 2. Also a very important observation was made and that is an RLE sequence can be read during mesh and written to at the same time. This advanced possible interactivity during meshing as can be seen in Diagram 2. The voxel models are represented either as flat 1D arrays or RLE-compressed streams.

The main contributions are:

- A new voxel meshing algorithm optimized for compressed voxel grids.

- Full support for interactive voxel editing and real-time updates.
- Integration with Unreal Engine 5 as a plugin using the RealtimeMeshComponent for efficient rendering and collision.
- Unreal Engine framework for performance profiling of voxel meshing techniques.

3. Integration to Unreal Engine

The implementation leverages Unreal Engine 5.4 [4], C++ programming, and the RealtimeMeshComponent plugin for dynamic mesh rendering. Compression of voxel models is achieved using Run-Length Encoding (RLE) to minimize memory footprint and speed up traversal. FastNoiseGenerator is employed to generate procedural voxel datasets for profiling. Usage of Unreal Engine can be seen in Figure 6 where voxels are rendered into a combined scene. Diagram of the integration into Unreal Engine can be seen in Diagram 1.

Interaction with voxel models is facilitated via Unreal Engine's PlayerController and Blueprint scripting, enabling dynamic voxel editing and immediate remeshing.

The real-time performance is measured using Unreal Insights, tracing CPU timing and memory usage, with results visualized through exported CSVs processed in Python.

4. Architecture

The system is structured around a modular Unreal Engine plugin. Core components include:

- Voxel Model Storage: Flat arrays and RLE streams.
- Meshing Module: Implements Run Directional Meshing, generating quads with optional merging for optimized mesh output.
- Interaction System: Captures player inputs (e.g., voxel editing) and updates voxel data with immediate remeshing.
- Profiling Framework: Generates structured datasets and scenarios to test meshing algorithms under various sparsity conditions.
- Each voxel model is divided into chunks to facilitate memory management and localized meshing.
-

4.1 Achievements

Multiple test scenarios were created to measure performance of voxel meshing. Wireframe of such scenario can be seen in Figure 5 from which Chart 1 and Chart 2 were made.

The project successfully demonstrates:

- Possibility of real-time conversion of RLE-compressed voxel models into meshes.
- Interactive editing of voxel models with immediate visual updates.
- Visible memory savings.
- Framework for performance profiling.

The most important achivement is reducing in memory as can be seen in Chart 1.

5. Future

Goal of this was to prove that voxel meshing from RLE is possible with advantages and this was proven. Howerver the algorithms need improvement in the future and this invention may motivate other developers to improve on this initial idea. Right now the algorithm needs optimization when it comes to reducing number of vertices as can be seen in Chart 2. Such future improvements may include:

- Full voxel face culling to further reduce triangle counts without compromising speed.
- Extension of the algorithm to support dynamic LOD generation compatible with Unreal Engine's Nanite system.
- Further reduction of necessary iterations.
- Voxelization and fast terrain generation.
- Advancements in cullar automata visualization and interaction.
- Cross-chunk culling.
- Introduction of custom compression.
- Adapting this algorithm into existing voxel engines.

Acknowledgements

I would like to thank my supervisor Ing. Tomáš Milet, Ph.D., for his support, guidance, and valuable feedback during the development of this thesis.)

References

[1] Cyril Crassin. *GigaVoxels: A Voxel-Based Rendering Pipeline For Efficient Exploration Of Large And Detailed Scenes.* PhD thesis, Grenoble University, 7 2011.

[2] Mikola Lysenko. Meshing in a minecraft game. online, 6 2012.

[3] Voxel Plugin. Voxel Plugin Documentation.

[4] Epic Games. *Unreal Engine 5.4 Documentation.*