VirtualVoxelCrowd: Rendering One Billion Characters at Real-Time

Jinyuan Yang[1] and Abraham G. Campbell[1]

[1]University College Dublin, Ireland

Abstract

In this paper, we introduce VirtualVoxelCrowd, which aims to address the challenges of data scale and overdraw in massive crowd rendering applications. The approach leverages multiple levels of detail and multi-pass culling to reduce rendering workload and overdraw. VirtualVoxelCrowd supports rendering of up to one billion characters, achieving unprecedented scale on standard graphics hardware while rendering subpixel-level voxels to prevent the level of detail transition artifacts. This method offers significant improvements in handling massive animated crowd visualization, establishing a new possibility for dynamic, large-scale scene rendering.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Rasterization;***

1. Introduction

Rendering dense crowds efficiently presents significant challenges due to the massive scale of data and overdraw issues. To address these, researchers have explored various strategies, such as mesh-based methods [DP*19], imposter-based methods [DHOO05, MR06], and point-based methods [WS02, TDGR14]. However, these methods suffer from problems such as the difficulty of retaining characters' original shapes after multiple levels of simplification, rendering novel views, and filling holes.

In our paper, we introduce VirtualVoxelCrowd, which extends the original VirtualVoxel approach [YC23] with significant enhancements. Unlike the original VirtualVoxel approach which can also render static scenes, this technique can render crowds on an unprecedented scale, supporting up to one billion characters, equivalent to trillions of triangles, on conventional graphics hardware.

VirtualVoxelCrowd innovates by generating multiple LoDs to decrease the rendering workload. It employs hierarchical depth culling [SBOT08] to eliminate invisible character parts and reduce overdraw. This approach renders subpixel-level voxels, preventing level-of-detail transition artifacts. Since this approach uses 3D voxels as geometry representation, the method can smoothly render novel views without generating holes that need to be filled.

2. Method Overview

VirtualVoxelCrowd aims to address the challenges of overdraw and the heavy rendering workload associated with crowd rendering. This section demonstrates the details of our solution, which leverages a carefully designed data structure and effective culling techniques to mitigate these issues.

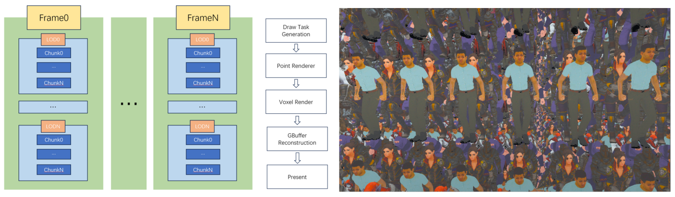

Figure 1: *Left: data layout of VirtualVoxelCrowd. Middle: VirtualVoxelCrowd rendering Workflow. Right: The test scene overview.*

2.1. Data Structure

To mitigate overdraw and lessen the rendering workload, the animated VirtualVoxel model was developed to feature multiple LoD and render the subpixel-level voxels. This approach not only preserves the model's shape but also streamlines rendering.

As demonstrated in Fig. 1, each VirtualVoxel model incorporates various LoDs, and voxels at a given LoD organized into chunks, each chunk contains 4096 voxels. Drawing inspiration from the imposter-based crowds rendering method, which captures N animation frames, we represent a mesh character's animation as N VirtualVoxel models. Each of these models comprises M LoDs, with each LoD containing K chunks. The same data layout is used based

DOI: 10.2312/egp.20241040

on the prior work of VirtualVoxel [YC23] to represent all the voxels, enabling efficient representation and animation of VirtualVoxel models. In our design, we set the parameter N to 30 and M to 7. The value for K is determined offline, varying according to the input meshes.

2.2. Offline Asset Processing

This method, diverging from the original VirtualVoxel approach, creates VirtualVoxel objects offline. The textured mesh is inputted into the voxelizer, which employs software rasterization for mesh voxelization. For animated skeleton meshes, we sample the animation sequence to calculate the animation matrices for 30 frames. Then these matrices are applied to the body vertices to achieve transformed vertices suitable for voxelization. In creating the VirtualVoxel model. Voxels are sorted in Morton order by position, grouping 4096 voxels into a chunk and computing an axis-aligned bounding box (AABB) for visibility culling at runtime. Then each voxel's neighborhood is tested if it is empty to eliminate invisible contact faces. The model's LoDs are constructed by sequentially voxelizing each level. The system supports voxel resolutions from 4 to 256. This resolution range prevents underfilling a 4096-voxel chunk, avoiding space wastage. Since we use 24-bit to store the voxel's 3D position, the resolution of a VirtualVoxel model can not exceed 256. This process outlines our approach to generating a single animation frame for a VirtualVoxel model.

2.3. Draw Task Generation

We implement hierarchical Z culling using the depth buffer from the previous frame. For each character model, we first determine its animation frame index based on elapsed time, then retrieve the corresponding VirtualVoxel model. We select the appropriate LoD for the VirtualVoxel model by analyzing the screen space axis-aligned bounding box (AABB). This involves calculating the bounding sphere's radius from the AABB; for instance, a radius of 128 prompts the selection of the LoD with a 256 resolution. While characters with a screen space radius exceeding 128 are rendered as regular mesh characters. Upon selecting the most appropriate LoD for each character, visibility tests are performed for each chunk within the chosen LoD. Chunks that are not occluded are atomically added to the draw list.

2.4. Two-Pass Voxel Rendering

The two-pass voxel renderer addresses the significant overdraw challenge in voxel scenes. While Hierarchical Z culling effectively removes unseen voxel chunks, overlapping visible chunks can still burden the GPU. This renderer takes the draw task list as input and initially processes voxels as points, then reconstructs them from these points to mitigate overdraw effects, following the original VirtualVoxel approach. In rendering each geometry cube, we omit the contact faces between adjacent voxels that are always hidden from the camera's view. Additionally, the early-Z test is performed to cull invisible pixels. This strategy significantly reduces overdraw in the scene, enhancing rendering efficiency.

3. Result and Conclusions

We evaluated VirtualVoxelCrowd's anti-overdraw capability with a 1000^3 array (one billion characters) test, positioning 5 unique human characters repeatedly within this array and selecting voxel sizes based on pixel screen sizes for rendering LoD.

Our demonstration renderer, built using C++20 and Vulkan, operates on a Windows 10 desktop equipped with an AMD Ryzen 5950X and an RTX 4090. It achieves an average frame time of 16.897 ms at 4K resolution, assessed via a camera fly-through of the test scene. Our test models and animations are downloaded from Mixamo, the vram consumption for each VirtualVoxel model instance is: Paladin 22.6245 MB, Vampire 38.625 MB, Ninja 30.1245 MB, Eve 28.2495 MB, Lewis 17.927MB.

This work illustrates the effectiveness of using Hierarchical Z culling and a two-pass voxel renderer to address overdraw, and VirtualVoxel for geometry simplification and rendering workload reduction.

Nonetheless, our approach has limitations. It supports only a limited number of frames for animation sequences, affecting animation smoothness. This trade-off is necessary for the real-time rendering of one billion human characters at 4K resolution. Despite these challenges, this method lays a solid foundation and offers a novel approach to facilitate future advancements in real-time voxel rendering technologies.

References

[DHOO05] Dobbyn S., Hamill J., O'Conor K., O'Sullivan C.: Geopostors: a real-time geometry/impostor crowd rendering system. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games* (2005), pp. 95–102. 1

[DP*19] Dong Y., Peng C., et al.: Real-time large crowd rendering with efficient character and instance management on gpu. *International Journal of Computer Games Technology 2019* (2019). 1

[MR06] Millan E., Rudomin I.: Impostors and pseudo-instancing for gpu crowd rendering. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia* (2006), pp. 49–55. 1

[SBOT08] Shopf J., Barczak J., Oat C., Tatarchuk N.: March of the froblins: simulation and rendering massive crowds of intelligent and detailed creatures on gpu. In *ACM SIGGRAPH 2008 Games*. 2008, pp. 52–101. 1

[TDGR14] Toledo L., De Gyves O., Rudomín I.: Hierarchical level of detail for varied animated crowds. *The Visual Computer 30* (2014), 949–961. 1

[WS02] Wand M., Strasser W.: Multi-resolution rendering of complex animated scenes. In *Computer Graphics Forum* (2002), vol. 21, Wiley Online Library, pp. 483–491. 1

[YC23] Yang J., Campbell A.: Virtualvoxel: Real-time large scale scene visualization and modification. In *ACM SIGGRAPH 2023 Posters*. 2023, pp. 1–2. 1, 2