A Relevant Research on the Establishment of A Voxel Gaming World

Kai Gao, Jun He* and Yi Qi
Xiamen Academy of Arts & Design, Fuzhou University, Xiamen, China

***Abstract*--Many three-dimensional (3D) games possess sophisticated gaming worlds and most of such worlds are constructed via a 3D triangle mesh, including landform, buildings, vegetation and other static objects. Some games focusing on natural outdoor environments, such as many RPG (Role-playing game) and MOBA (Multiplayer Online Battle Arena) games, adopt the height filed to demonstrate a landform, while some games focusing on indoor types, such as many FPS (First Person Shooting), TPS (Third Personal Shooting Game) and ACT (Action Game), build basic indoor environments via a constructive solid geometry (CGD) technology. Since the unprecedented success of Minecraft in 2009, the voxel has become another feasible manner to construct a gaming world.**

I. INTRODUCTION

The production of a gaming world accounts for a large proportion of its total production cost, which will continuously increase with the improving platform performance of gaming hardware and the surging demand for gaming contents.As virtual worlds in games become larger and more detailed, the need for rich, interactive content to realistically populate these worlds becomes greater.3D triangle meshes conform to the demonstration manner of gaming worlds developed by contemporary hardware. In addition, 3D triangle meshes possess a relatively free modelling manner and mature digital content creation tools. However, 3D triangle meshes are confronted with problems, including a higher modelling cost, the only one surface demonstration manner, difficulties in modification and in achieving the level of detail (LOD) of consecutive or discrete series. The voxel refers to the three-dimensional version of the pixel. For 2D, an image can be demonstrated via a 2D array in colors. For 3D, a gridded 3D space can be demonstrated via a 3D array of voxels. Each voxel stores one bit, which demonstrates solid or hollow features of such space (shown as Figure 1). This kind of binary voxels is the simplest voxel form. Besides, the voxel can also store other properties[1-3].

Fig. 1. Binary Voxel

II. FEATURE ANALYSIS OF VOXEL

A. Structural Advantage

As voxel data structures are simple and uniform, it's easier to modify voxels than meshes. Therefore, games can make user-generated contents (UCG) and enable game rules to dynamically change gaming worlds, such as destructible objects and terrain morphing, which can, to some extent, largely lower production costs[4-5] (shown as Figure 2).

Fig. 2. Minecraft

B. Visual Presentation

We can improve voxel data resolutions. For example, the detailed degree is increased from one voxel/m^3 to one voxel/cm^3. However, cubic data is increasing based on the cubic series, which also requires a large storage capacity. Even if every voxel accounts for only one byte, a large storage space is also required. Certainly, if a large number of correlated voxels are solid or hallow in general application scenes, some data structures can be used to compress original voxel data.

The plans of storage colors and normal provided by each voxel may fail to satisfy developed requirements of games. The key point is that every detail (such as the 1 cm accuracy) in the scene is independently edited. However, games often require virtual scenes and players don't care about whether bumps on walls of a cathedral are the exactly same as those in the real life. From the scene production perspective, scenes can be randomly arranged. But such random degree incurs control and management difficulties.

C. Isosurface Extract

As to corner angles mentioned above, interpolation manners and normal data are added when voxels experience renderings by ray tracings, which makes effects smoother. There is another method. Scalar data are firstly stored in a voxel and the isosurface of such scalar field is extracted, thus generating a 3D triangle mesh that experiences renderings in a rasterization manner. Similar to 2D situations, contour lines of a height field are extracted and then experience renderings. We can store material densities of scenes or signed distances,which act as a scalar field.

Given a scalar filed, the classical Marching Cubes (MC) algorithm is used to extract its isosurface. The MC algorithm is quite simple. Only voxel data are scanned. If the isovalue lies between two adjacent voxel values, the linear interpolation is used to figure out vertexes of the

isosurface in the middle position and a look-up table is used to decide how to make an isosurface triangle mesh (shown as Figure 3) by connecting such vertexes. The MC algorithm generates only pliable isosurfaces and fails to demonstrate sharp vertexes and arrises (shown as Figure4), which can be solved by dual countering (DC) and an addition of normal data into a voxel.

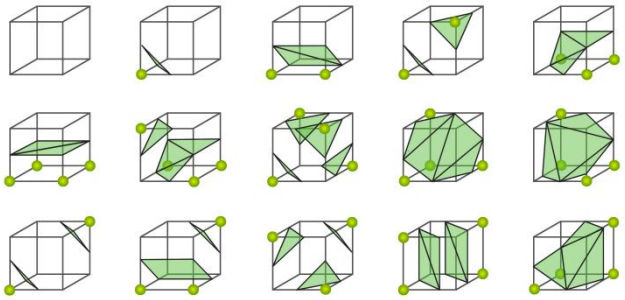


Fig. 3. 15 Cubic Configurations of the Marching Cubes Algorithm

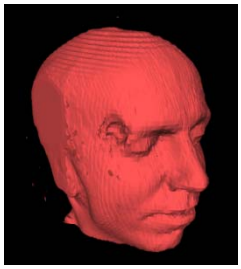


Fig. 4. Sharp Vertexes and Arries Kept by the Dual Contouring Algorithm

D. Data Amplification

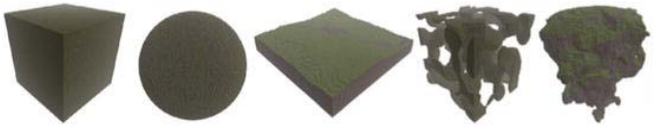
The problem of arrises is solved by the isosurface extract, rather than a pliable surface caused by a voxel with a high resolution, which also enhances the detailed degree with a small increase in data amount. The general data amplification refers to generating a large amount of data via a small amount of data. Terrains of general heights simply adopt numerous layers of repeatedly and densely tiling textures, which can also be called as a simple data amplification. This method can be applied to the isosurface extract that requires more complex texture mapping manners.

E. Voxel-Based Rasterisation

There exist several online tutorials regarding the development of voxel engines, and the main challenges that arise when attempting efficient rendering. A major consideration is the platform technology upon which these are built, which is conducive to GPU fluid simulation.

Barrett's (2014) Obbg engine was developed to illustrate the use of a highly efficient voxel render library, and it illustrates large Minecraft-styled voxel landscapes that are implemented with OpenGL. With an OpenGL voxel engine, CUDA, OpenCL, and OpenGL fluid simulations are valid candidates for integration.

The voxels rendering framework (Miller et al. 2014), implemented in Direct3D, focuses on rendering sparse 3D voxel environments with data amplification using Geometry Shaders on GPU. This approach reduces the CPU-to-GPU bandwidth costs of updating the volumetric grid data and makes it more suitable for dynamic voxel-based environment rendering[6] (Fig. 5).



Figu.5. Examples of the voxels (Miller et al. 2014) sparse voxel-based rendering engine

F. Gameplay Influences

Focusing on gameplay features, games are real-time renders. Many binary voxel games have demonstrated unique gameplay elements, but still confronted with problems concerning a dynamic voxel world at the technological level, mainly including physical and artificial intelligent (AI) problems. In the physical aspect, basic contents are taken into consideration, such as collision detection and supporting analysis of isosurfaces and rigid bodies.

III. CONCLUSION

As traditional gaming worlds have come across limitations in gameplay features and productions, a new production manner based on voxels can provide numerous innovations in order to improve game qualities and control production costs. It's collaborations at all levels, mature tools and production procedures that overturn such traditional gaming worlds. For engine technologies, demands of all aspects should be satisfied, including multi-resolution modelling, real-time renderings, physical imitations and artificial intelligence in a large-scale gaming world. The voxel demonstration manner can open a new game technology path to producing more innovative games.

REFERENCES

[1] M. Bennett, "Semantic Content Generation Framework for Game Worlds," 2014 6th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES), Valletta, pp. 1-8, 2014 .

[2] S. N. Grigor' ev, A. V. Tolok and N. B. Tolok, "Local search gradient algorithm based on functional voxel modeling," Programming and Computer Software, vol. 43, pp.300-306, Sep. 2017.

[3] L. Stemkoski, "Introduction to 3D Graphics and Games," Java Game Development with LibGDX, pp.359-389, 2018.

[4] M. Miller, A. Cumming, K. Chalmers, B. Kenwright, and K. Mitchell, (2014). Poxels: Polygonal voxel environment rendering. Proceedings of the 20th ACM symposium on virtual reality software and technology. Edinburgh, Scotland, 2014.

[5] T. Wang, Z. Jiang, Q. Kemao, F. Lin and S. H. Soon, "GPU Accelerated Digital Volume Correlation," Experimental Mechanics, vol. 56, pp.297-309, Feb. 2016.

[6] J. Zadick, B. Kenwright, and K. Mitchell, "Integrating Real-Time Fluid Simulation with a Voxel Engine," The Computer Games Journal, vol. 5, pp.55-64, Sep. 2016.