Ang Jin Yang

Frustum Culling

Frustum culling is a technique used in computer graphics to improve rendering performance by excluding objects or portions of a scene that lie outside the view frustum from the rendering process. The view frustum is the pyramid-shaped viewing volume defined by the camera's position and orientation.

Implementation for discussion

This discussion is conducted within a C++ environment utilizing OpenGL and its accompanying math library, GLM. The measurement unit for performance is frames per second (FPS), which is being rendered with Vsync enabled and capped at 70 FPS.

Computing view frustum dynamically

To maintain consistency in a dynamic scene, the view frustum needs to be dynamically reconstructed. This ensures that the frustum accurately reflects changes in the camera's position, orientation, and field of view as objects move within the scene.

Bounding Volume

The core concept of frustum culling revolves around identifying intersections with the mesh's bounding volume. While this explanation focuses solely on the PCA sphere method for brevity, it's worth noting that various other bounding volume techniques are also applicable and have been tested.

These include:

- Axis-Aligned Bounding Box (AABB)
- Object-Oriented Bounding Box (OBB)
- Larsson Sphere
- Ritter Sphere

Each of these methods comes with its own computational complexity, making them suitable for different scenarios and mesh geometries.

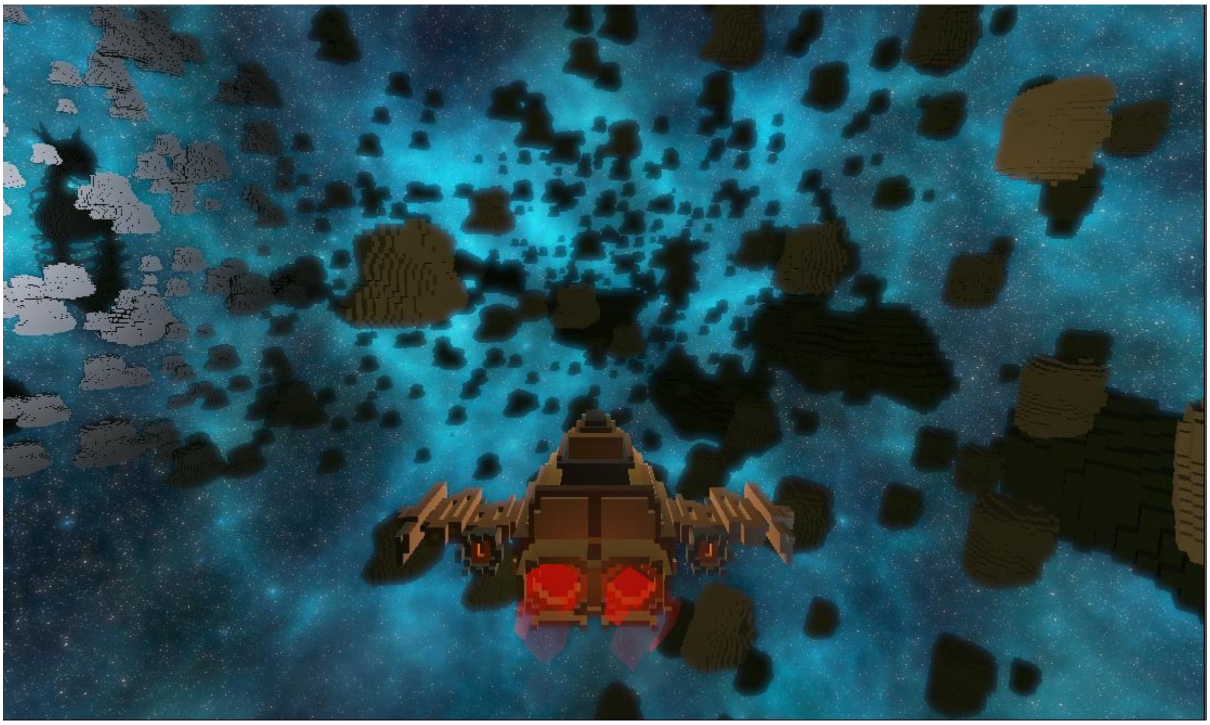

Figure 1: When the screen is filled with numerous entities, the average frames per second (FPS) stands at 53.

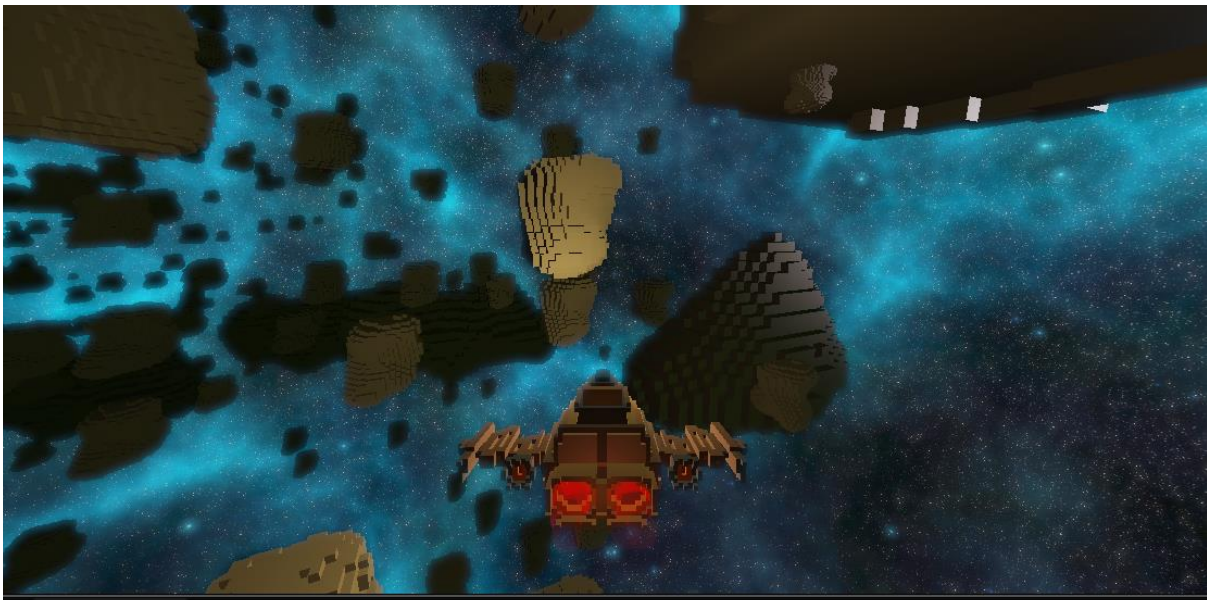

Figure 2: When the screen is partially filled with numerous entities, the average frames per second (FPS) stands at 65.

Conclusion

The absence of frustum culling resulted in an average frames-per-second (FPS) calculation of 20, highlighting its significant impact on performance improvement by rendering only what's in view. Selecting the appropriate bounding volume is equally critical to achieving a balance between effective culling and computational efficiency.