Voxel builder

LikhithaSri Vulasa and Vidhathri Kasagani
l.vulasa001@umb.edu , v.kasagani001@umb.edu
University of Massachusetts Boston

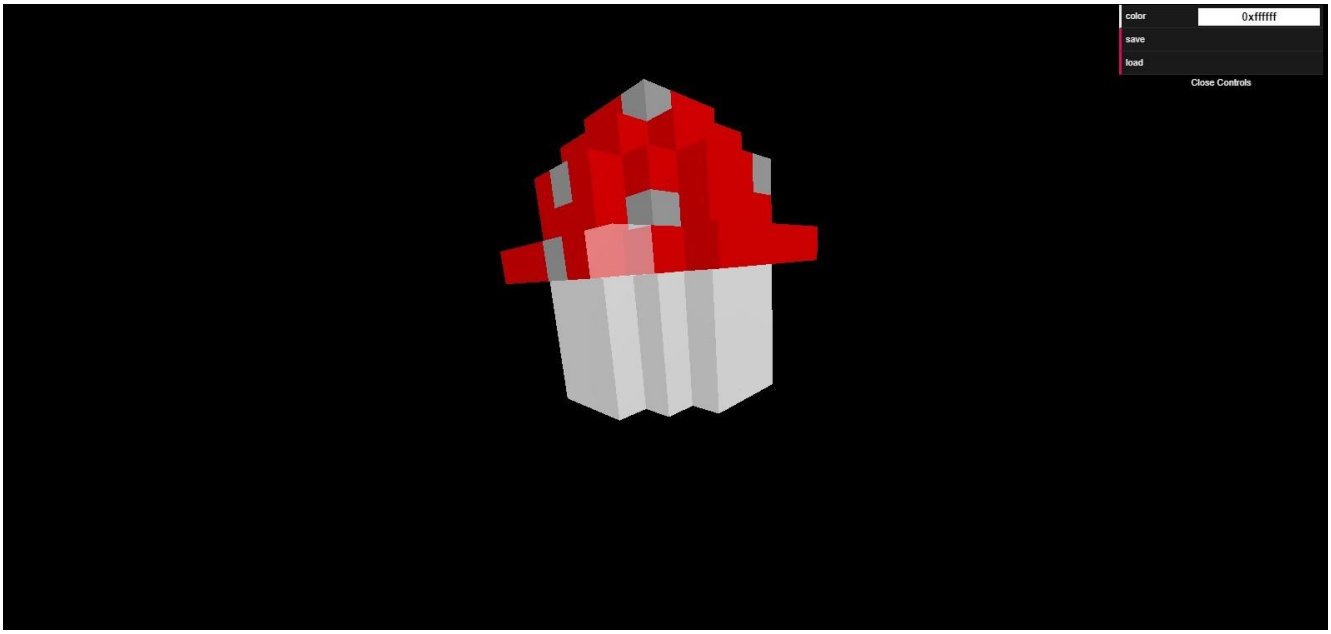

Figure 1: Voxel Mushroom created with the program

ABSTRACT

The objective of this project is to provide a simplified web-based voxel builder that enables users to generate basic voxel geometry. Further, it supports importing and saving txt files.

KEYWORDS

WebGL, Visualization, 3D Model, Voxel

ACM Reference Format:
LikhithaSri Vulasa and Vidhathri Kasagani. 2022. Voxel builder. In *CS460: Computer Graphics at UMass Boston, Fall 2022.* Boston, MA, USA, 2 pages. https://CS460.org

1 INTRODUCTION

This project is significant because it enables users to quickly generate a model without the use of any other software.

2 RELATED WORK

Three.js was utilized in the development of this project, while dat.gui was used for the menu selections. Additionally, it made use of the FileSaver.js module to save strings to text files.

3 METHOD

The rendering is handled by Three.js, and the software uses its raycasts to decide where to position the cubes. Once a raycast is fired, it chooses the second object it encounters (this is becasue the cursor or "ghost" cube is also hit by the raycast and we want to ignore that). The program then selects the face it will strike and positions the pointer there so the user can see where the voxel will be placed. To add or remove a voxel, the user can then shift+click or shift+right-click. A color choice is also available in the dat-gui menu.

3.1 Implementation

This section uses the raycast array of intersections to select the next item after determining whether the current object is the cursor and moving on to the next junction. Additionally, the face is taken.

```
for (var i = 0; i < intersects.length; i++){
if(intersects[i].object == cursorObject){
continue;
};
selectedObject = intersects[i].object;
objectFace = intersects[i].face;
break;
```

}
The "ghost" cube or cursor is positioned according to whose face normal it is striking, and since face normals are dependent on the axis and direction, they can either be -1 or 1, we can simply multiply them by a number to offset them from the position of the chosen item.

```
var newPosition = new THREE.Vector3(
selectedObject.position.x + objectFace.normal.x * 10,
selectedObject.position.y + objectFace.normal.y * 10,
selectedObject.position.z + objectFace.normal.z * 10);
```

3.2 Milestones

How did you structure the development?

3.2.1 *Milestone 1.* Brainstormed ideas on ways of implementation (mainly how to deteremine where to place the cubes)

3.2.2 *Milestone 2.* Implemented the raycast selection+ placing

3.2.3 *Milestone 3.* Implemented saving files

3.3 Challenges

Describe the challenges you faced.

- Challenge 1: It took some time to research and discover how to save files.
- Challenge 2: It took some experimenting to determine what could and couldn't be accessible via Three.js.

4 RESULTS

The project's end result is a user-friendly website that allows users to store voxel creations to a text file.

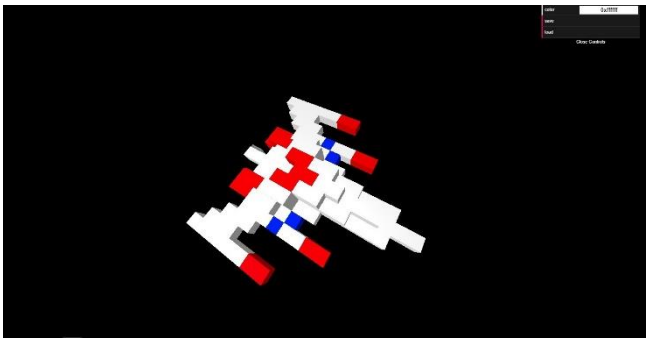

Figure 2: Galaga ship recreated

5 CONCLUSIONS

The project was successful overall, but some aspects, such as the materials, might be improved because of the way they are currently set up. Even if they are using the same color, a new material each cube would be preferable, so optimization might be improved. The scene's control method may be improved because trackball controls might make it challenging to place the object where users would like it to be.

REFERENCES

https://threejs.org/examples/#webgl_camera_cinematic

https://threejs.org/examples/#webgl_instancing_raycast

https://threejs.org/examples/#webgl_interactive_cubes

https://threejs.org/examples/#webgl_interactive_cubes_gpu

https://threejs.org/examples/#webgl_interactive_voxelpainter