Three.Js Rubik's Cube

Ishan Dubey
Ishan.Dubey001@umb.edu
University of Massachusetts Boston

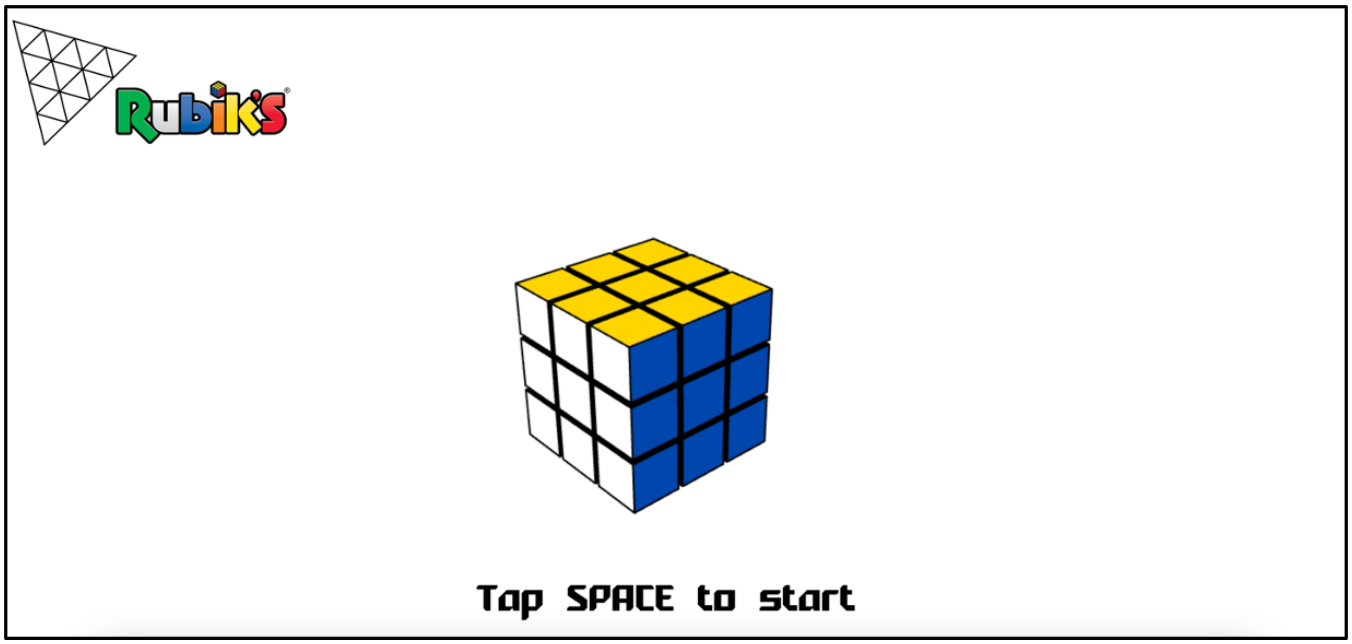

Figure 1: Rubik's Game launch screen.

ABSTRACT

Three.Js Rubik's cube is the virtual implementation of a classic 3x3x3 Rubik's puzzle utilizing a lightweight WebGL based JavaScript library Three.Js, which allows us to create and render animated 3D computer graphics in a web browser. The game has the same set of rotations to solve the puzzle as the physical Rubik's cube and also includes an auto-solver based on reverse rotation principle to solve for different colors.

KEYWORDS

WebGL, Three.Js, Rubik's, Visualization

ACM Reference Format:
Ishan Dubey. 2022. Three.Js Rubik's Cube. In *CS460: Computer Graphics at UMass Boston, Fall 2022*. Boston, MA, USA, 3 pages. https://CS460.org

CS460, Fall 2022, Boston, MA

ACM ISBN 1337.
https://CS460.org

1 INTRODUCTION

Learning Computer Graphics requires deep understanding and implementation of various graphical components. In this project I gamified the conceptual learning by implementing a puzzle in a 3D virtual world. This helped me in understanding the fundamentals as well as the structure of the underlying library which includes the scene structure, geometries, materials, transformation of coordinates and object's parent-child relationship. Also, creating an auto-solver for the puzzle developed my understanding on binding mathematical algorithms to graphical objects.

2 RELATED WORK

The Cube [2], Rubik-Js [3].

3 METHOD

The project is implemented in Three.Js, a javascript library which allows us to use WebGL in an HTML5 canvas element. WebGL is a JavaScript API which creates 2D and 3D graphic rendering. I utilized this library to create a Rubik's cube and give rotation to the different faces as per the official Rubik's puzzle. These faces are UP, DOWN, LEFT, RIGHT, FRONT and BACK. Corresponding rotations of these faces are U, U', D, D', L, L', R, R', F, F', B, B'.

These rotations are bound to the cube by HTML5 buttons. Also, buttons are added to shuffle and solve the puzzle. For styling, transition and animation of the components, JavaScript and CSS is used.

3.1 Implementation

Implementation of the project includes creating top, middle and bottom layers with each layer consisting of 9 small cubes with same face color resulting in 6 different colors, one for each face of the Rubik's. The cubes are then added to a list and every time a rotation is to be made, the list is traversed to get corresponding cubes to be transformed. These are then attached to an Object3D group to get rotated. Once the rotation is done, the world matrix of cubes get updated and they are attached back to the scene.

To shuffle the Rubik's, a set of 30 predefined rotations are performed which guarantees maximum possible solutions, making the puzzle a little challenging to solve. For auto-solve, a track of all the rotations done by the user are kept in a list. The rotations are then reversed by their counter rotations resulting in the solution of each face. Once the puzzle is solved, it cannot be auto-solved again and has to be shuffled to start-over.

Following are the code snippets for the implementation described:

```
/* Creating individual cubes layer by layer */
var [C1, lMat1, L1] = getCube(['#000000','#ffffff',
        '#ffd500','#000000','#0046ad','#000000']);
lineMats.push(lMat1);
C1.position.set(-11,0,0);
L1.position.set(-11,0,0);
allCubes.push(C1);
allCubes.push(L1);
scene.attach( C1 );
scene.attach( L1 );
```

```
/* Function to shuffle the Rubik's. */
function shuffle(shuffleList){
  for (let i = 0; i < shuffleList.length; i++){
    setTimeout(function() {
        rotations[shuffleList[i]](scene,allCubes);
    }, 100*(i+1));
  }
};
```

```
/* Function to solve the Rubik's. */
function autoSolve(){
  if (rotHist.length > 0) {
    for (let i = rotHist.length - 1; i >= 0; i--){
        setTimeout(function() {
        rotations[rotMap[rotHist[i]]](scene,allCubes);
        }, 100 * (Math.abs(i - rotHist.length) + 1));
    }
    setTimeout(function() {
        rotHist = [];
    }, 100 * (rotHist.length + 1));
  }
};
```

3.2 Milestones

Following are the milestones that structured the development:

3.2.1 Milestone 1. Used pen-paper to plan the position of every cube in the coordinate system and also gave a distinct no. to each small cube in order to identify the layer it belongs to.

3.2.2 Milestone 2. Applied linear algebra to get the translated position of the cubes once they are rotated. Once it worked, created a function to do the same every time when a rotation is made.

3.2.3 Milestone 3. Created a key,value pair of rotations and their counter so that they can be passed to the auto-solve function. Once the solution is done, the list maintaining rotations is cleared to optimize the code and avoid repetitions.

3.2.4 Milestone 4. Added buttons for each rotation, shuffle, solve and help. This is done to make the puzzle user-friendly to solve.

3.3 Challenges

The complexity of using Three.Js is less than WebGL but what makes it challenging is the limited documentation. It has less freedom to implement few functionalities and the only workaround is to go through the examples with complex code base. Few challenges which I faced are:

- Challenge 1: Giving each face of cube a different color was a little tricky. I got the position of each vertex, looped through to group them into a pair of 6, and then assigned a different color to each group [6].
- Challenge 2: Rotations work differently when objects are added to an Object3D group. It took me a lot of time to wrap around local and world coordinates in order to perform correct transformation [5]. Despite trying very hard I did not get any success in animating the rotations.
- Challenge 3: Giving rounded edges to the cube required a custom implementation which was time consuming and also affected the rotations. Eventually, I had to drop the idea.
- Challenge 4: When using original LineMaterial from Three.Js, line width property does not work for all the platforms. Thus, to resolve this a custom library i.e., three.js-webgl-lines-fat [4] has to be implemented. This added a few more Js scripts to the project.

4 RESULTS

The results of the project are quite satisfactory and includes most of the primary functionalities as required and proposed. Figure 2 shows the puzzle in a shuffled state and Figure 3 shows the puzzle after using auto-solve. The corresponding buttons work well and have micro animations to add interactivity.

Link to source code: https://github.com/idubey-code/RubiksCube
Link to game hosted on GitHub pages: https://idubey-code.github.io/RubiksCube/

5 CONCLUSIONS

The project was a good way to learn the fundamentals of Computer Graphics and also, to apply the learning done in the class to develop something creative and fun. Working on this project not only helped

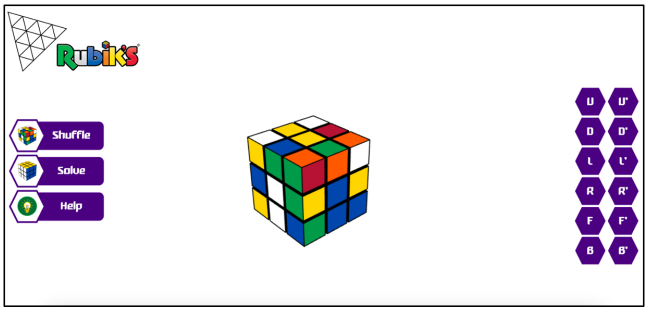

Figure 2: Shuffled cube.

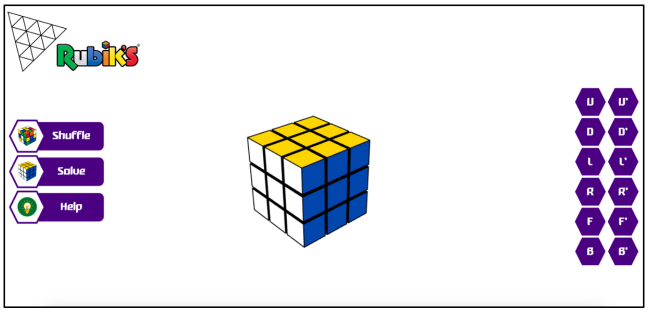

Figure 3: Solved cube using autosolve.

me in taking up a new skill but, It also taught me how to study existing work, understand it, extract important functionalities and customize them for my own problem statement.

The project also has a few points of improvement. One is to develop a generalize algorithm which solves the cube in the most optimized way with minimum number of moves from any unsolved state. Second is to add animated interactivity to the Rubik's so that the layers can be rotated by dragging in the respective direction making the game overall more realistic. These improvements require more understanding of matrices and a little time for trial and error.

REFERENCES

[1] Ricardo Cabello et al. 2010. Three.js. https://github.com/mrdoob/three.js(2010).
[2] Boris Šehovac. The Cube. https://codepen.io/bsehovac/pen/EMyWVv
[3] Whitfield-Seed J. rubik-js. GitHub. Published November 8, 2022. Accessed December 22, 2022. https://github.com/joews/rubik-js
[4] three.js webgl - lines - fat. threejs.org. https://threejs.org/examples/webgl_lines_fat.html
[5] 3d - Three.js: Adding and Removing Children of Rotated Objects. Stack Overflow. Accessed December 22, 2022. https://stackoverflow.com/questions/20089098/three-js-adding-and-removing-children-of-rotated-objects
[6] colors - Coloring faces of a Three.js BoxGeometry. Stack Overflow. Accessed December 22, 2022. https://stackoverflow.com/questions/67989801/coloring-faces-of-a-three-js-boxgeometry