Interactive Graphics Final Project Report

Teacher: Prof. Marco Schaerf

Author: Xu Wei, 1797103

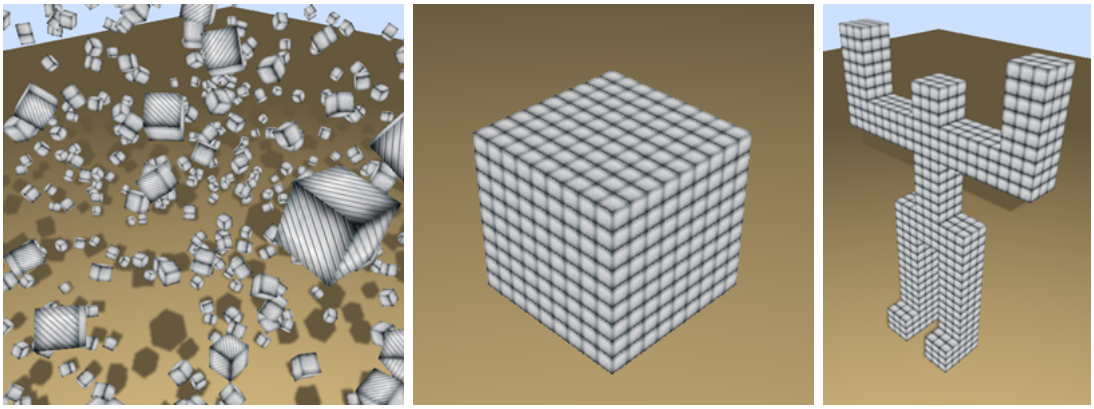
February 21, 2019

1 Introduction

Inspired by an animated video on a maker faire, I came up with the idea of this project. In the original video, there is an animated face composed by flying objects with different shapes as in the below screenshot from the video.

Instead of making a face, I make a robot. And since the process is a bit like how a transformer transforming from a car to a robot superhero. I name it "Transformer Lite".

In the initial state, hundreds of cubes are floating in the air with random positions and rotations and they can transform to two other states by clicking corresponding buttons, the one big cube state and the robot state. The one big cube state is just a state that all the cubes gathering together and composing a big cube. I set the number of cubes to 729 so as to make a perfect big cube without missing or having extra cubes. Then, the robot state is a simple cubic robot with two arms and two legs. The robot seems to be quite simple but to set the positions and rotations of these hundreds of cubes is quite trivial. And there is another "dance" button which can trigger a dance animation: the robot moving his legs, arms up and down and sitting down on the floor then standing up again.



2 Libraries and tools

The main library I used in this project is Three.js. It is a cross-browser Javascript library and API used to create and display animated 3D computer graphics in a web browser. Three.js allows the GPU-accelerated 3D animations using the JavaScript language as part of a website without relying on proprietary browser plugins. This is possible due to the advent of WebGL. Three.js is a high-level library of WebGL which makes all the work in WebGL much easier and faster.

The project also includes three tools or small libraries:

- OrbitControls.js
- stats.js
- tween.js

Orbitcontrols.js is actually a tool in the three.js. It allows the camera to orbit around a target as well as zoom in and zoom out. Stats.js is a Javascript performance monitor. It provides a simple info box that will help us monitor code performance with frames rendered in the last second, milliseconds needed to render a frame and Mbytes fo allocated memory. Tween.js is a Javascript tweening engine that allows us to change the values of the properties of an object like position and rotation in a smooth way. All the animations in my project including transforming and dancing are achieved by it. Several helpers in the three.js are also used during the work, including AxesHelper, LightHelper and ShadowCameraHelper.

3 Technical aspects

3.1 Hierarchical model

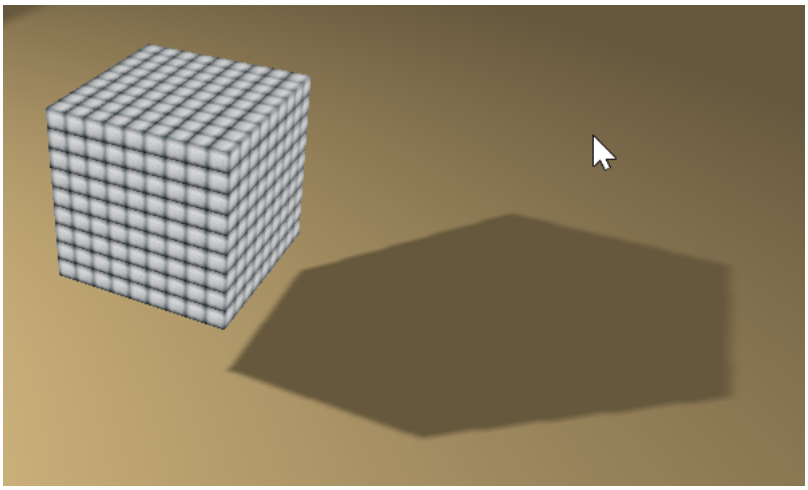
The number of small cubes is set to 729 and the size of each cube is 20x20x20. The size of the big cube is 180x180x180. To compose the "transformer" robot, cubes are grouped

into 6 parts: head with 27 cubes, body with 216 cubes, two arms with 81 cubes each and two legs with 162 cubes each.

```
if (i < 27) {
    head.add(object1);
} else if (i < 243) {
    body.add(object1);
} else if (i < 324) {
    leftarm.add(object1);
} else if (i < 405) {
    rightarm.add(object1);
} else if (i < 567) {
    leftleg.add(object1);
} else {
    rightleg.add(object1);
}
```

3.2 Light and texture

A spotlight is used with an ambient light to cast shadows of cubes and the robot on the solid ground. Spotlight gets emitted from a single point in one direction, along a cone that increases in size the further from the light it gets. The texture of the cubes is loaded by TextureLoader from a steel texture picture which I found online.



3.3 User Interaction

The main interaction is achieved by the four buttons on the bottom of the scene. The first three represent the three transforming state: small cubes, one big cube and robot correspondingly. By clicking the buttons, it's very easy to switch between these states. There is another "dance" button which triggers the dancing animation of the robot by moving his legs, arms up and down and sitting down on the floor then standing up again. Making use of OrbitControls, position of the camera can be set by dragging the

left mouse button around and scrolling up and down the mouse wheel to zoom in and out.

3.4 Animation

All the cubes are created with random positions and orientations around the zero point but also above the solid ground. The values of these cubes' information are saved in an array named "objects" for later use as well as in a dictionary named "targets" for implementing transforming. The positions and orientations of the cubes in the big cube state and the robot state are also computed and saved in the same dictionary. During transforming, a "transform" function is called and each cube then moves slowly from the original position and rotation to the target position and rotation.

```
function transform( targets, duration ) {
    TWEEN.removeAll();
    for ( var i = 0; i < objects.length; i ++ ) {
        var object = objects[ i ];
        var target = targets[ i ];
        new TWEEN.Tween( object.position )
            .to( { x: target.position.x, y: target.position.z }, Math.random() * duration + duration )
            .easing( TWEEN.Easing.Exponential.InOut )
            .start();
        new TWEEN.Tween( object.rotation )
            .to( { x: target.rotation.x, y: target.rotation.z }, Math.random() * duration + duration )
            .easing( TWEEN.Easing.Exponential.InOut )
            .start();
    }
}
```

All the cubes are grouped into 6 groups corresponding to the 6 parts of the robot: head, body, two arms and two legs. The dancing animation basically is just moving these parts separately. One problem I encountered is that the parts can only rotate around the zero point instead of the joint of itself. To solve it, I create another new pivot object for each part after moving the part to the zero point. So the new pivot object can rotate around the joint then I move it to the right place it belongs. Another method to solve this problem is to change the position of the geometry of the object, but since I use the same geometry for all the cubes. This method doesn't work in my case.

```
leftarm.position.y = -210;
leftarm_pivot.add( leftarm );
leftarm_pivot.position.y = 210;
```

To make the animation run forever, I tried to use the yoyo function of tween.js. It allows the object to return back to the original place after animation. But an issue also comes with it. If I stop the animation in the middle, the new initial state can be any of

the intermediate states which will be a problem if I restart it again later. So I explore another way, trying to chain two tweens with each other and it works well in my case. Even if I stop in the middle when I restart the robot can still resume the animation it was doing before.

```
tween1.chain(tween7);
tween7.chain(tween1);
```

4 Reference

- Three.js - Wikipedia
- three.js / docs
- three.js / examples / layers
- three.js / examples / materials / cubemap / balls / refraction
- three.js / examples / lights / spotlight
- three.js / examples / periodictable