Stephen Round
Programming Final

This pretty storm of petals/tops/primitive shapes are whirling around in 3D space. The camera orbits the storm slowly, and everything orbits around a small flashing ball. Moving the mouse around will shine a multicolored flashing light on the middle sphere. Clicking the screen will reorient a random amount of the objects. It's a mesmerizing display that could be a screen saver or a time waster.

Why tops? I tried to construct swords the same way I made the 2D ones, but there's not really a way to move the 3D shapes relative to each other. Instead, they became a smooshed pile of cubes and a cone. I played around with variables and removed shapes until this design popped up, and I decided with the normalMaterial() (which is similar to fill() in 2D shapes) they looked the best in this new rotating world.

The build is pretty simple. There isn't a ton of change from the original swords sketch that I wrote in week six, but I had to study a lot of p5's 3D capabilities. The biggest discovery was frameCount and WEBGL, which helped make the movement constant and enabled the 3D respectively. I experimented with lights as well, which didn't prove to be too difficult. I used built-in examples to explore this function.

```
13 function draw() {
14   background(0);
15   var mouseLocY = (mouseY / height - 0.5) * (-2);
16   var mouseLocX = (mouseX / width - 0.5) * 2;
17
18   pointLight(random(40), random(40), random(40), mouseLocX, mouseLocY, 0);
19   //this makes a mouse light that points at the center ball,and makes it appear to roll
20
```

In the comments are a lot of early experiments that I decided to leave in, just in case I wanted to explore them again. The most interesting one is a built-in function called orbitControl(), which allows you to click and drag the mouse to move the camera, snapping the camera back in place when you let

go. However, it's a very limited movement and I left it out because clicking the mouse already serves a function.

An interesting development (as said in the code as well) was figuring out how to make the ball centered. The original width/2 and height/2 put the ball in the bottom corner of the screen, so the conclusion I came to was to divide width and height by themselves, which surprisingly worked.

```
26  translate(width / width, height / height);
```

My greatest achievement in this project was figuring out how to find the simple solution instead of an overly complicated one. The sketch spins not because the camera actually orbits it, but because the sketch is spinning in its entirety on the Y axis while the camera stays still. I had the same end result with only one line of code: rotateY(frameCount * 0.01); above the rest of the draw loop. The frameCount keeps the rotation going without a for loop.

A similar line of code makes the cycles of tops orbiting the center like an atom design. Before the for loop to draw the circle of objects is called, a rotateX(frameCount * 0.01); is called, making each circle rotate along the X axis.

```
58  Swords.prototype.spin = function() {
59    rotateX(frameCount * 0.01);
60    for (var r = 0; r < 18; r++) {
61      rotateX(PI / 10);
62      rotateY(PI / 3);
63      rotateZ(PI / 5); //each rotation axis here contributes to the positions and boundaries of the top cycles.
64      this.draw(); //for example, making rotateZ (PI/8) lowers the ceiling and has the tops bounce downwards wh
65    }
66  }
```

This function is where most of the magic can happen. It took me a few tries before I decided this was the one that looked best. Another interesting display can be made by multiplying the PI/10 in rotateX by the variable r from the for loop, and this makes the tops spawn in an almost completely random fashion. Messing around with this function will create some interesting results, whether you change the PI division numbers, add r to any of them, or change the rotation axis.

Stephen Round
Programming Final

For future directions, perhaps if I had more time and was a little more code savvy, I was thinking of making all these different sketch possibilities available, and making it either a keyPressed (with any key) event or just change the mouseClicked event to not just shuffle the objects, but alter the spin() function to a new random viable sketch. Another option would be to make the camera move with the WASD keys or the arrow keys, so you can explore the space more freely.