CS460 Fall 2020
Github Username: Msoohoo1
Due Date: 09/30/2020

Assignment 3: Three.js Cubes ... and other geometries

We will use Three.js to create multiple different geometries in an interactive fashion.

In class, we learned how to create a `THREE.Mesh` by combining the `THREE.BoxBufferGeometry` and the `THREE.MeshStandardMaterial`. We also learned how to *unproject* a mouse click from 2D (viewport / screen space) to a 3D position. This way, we were able use the `window.onclick` callback to move a cube to a new position in the 3D scene. Now, we will extend our code.

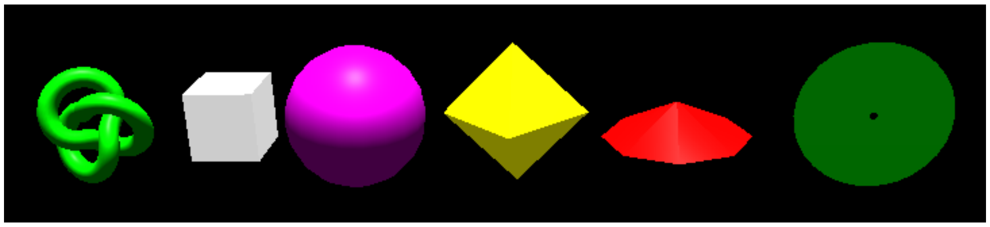
The goal of this assignment is to create multiple different geometries by clicking in the viewport. This means, rather than moving an existing mesh, we will create new ones in the `window.onclick` callback. On each click, our code will randomly choose a different geometry and a random color to place the object at the current mouse position.

We will be using six different geometries. Before we start coding, we want to understand their parameters. Please complete the table below. You can find this information in the Three.js documentation at `https://threejs.org/docs/` (scroll down to Geometries). In most cases, we only care about the first few parameters (**please replace the Xs**).

Constructor	Parameters
THREE.BoxBufferGeometry	(width, height, depth)
THREE.TorusKnotBufferGeometry	(radius, tube, radialSegments, tubularSegments)
THREE.SphereBufferGeometry	(radius, widthSegment, heightSegment)
THREE.OctahedronBufferGeometry	(radius)
THREE.ConeBufferGeometry	(radius, height)
THREE.RingBufferGeometry	(innerRadius, outerRadius, thetaSegments)

Please write code to create one of these six geometries with a random color on each click at the current mouse position. We will use the `SHIFT`-key to distinguish between geometry placement and regular camera movement. Copy the starter code from `https://cs460.org/shortcuts/08/` and save it as **03/index.html** in your github fork. This code includes the `window.onclick` callback, the `SHIFT`-key condition, and the `unproject` functionality.

After six clicks, if you are lucky and you don't have duplicate shapes, this could be your result:



Please make sure that your code is accessible through Github Pages. Also, please commit this PDF and your final code to your Github fork, and submit a pull request.

Link to your assignment: `https://Msoohoo1.github.io/cs460student/03/`
Credit to: Professor Haehn

Bonus (33 points):

Part 1 (5 points): Do you observe Z-Fighting? If yes, when?

I do see Z-Fighting. It occurs when two shapes are overlapping each other. Because of that, it is confused on which color to show.

Part 2 (10 points): Please change `window.onclick` to `window.onmousemove`. Now, holding `SHIFT` and moving the mouse draws a ton of shapes. Submit your changed code as part of your `03/index.html` file and **please replace the screenshot below with your drawing**.

My screenshot

For Fun

Part 3 (18 points): Please keep track of the number of placed objects and print the count in the JavaScript console. Now, with the change to `window.onmousemove`, after how many objects do you see a slower rendering performance?

I saw a slower rendering performance after 4,252. It started to lag a little which allowed me to recognize that this is the point of slow rendering.

What happens if the console is not open during drawing?

To be honest, I don't see a difference. Maybe it is due to the fact that there is no longer a measurement, a counter to truly see the moment that it slowly lags. And because of that, I can't see if there is even slight change without the console.

Can you estimate the total number of triangles drawn as soon as slow-down occurs?

Yes and here is a way we could do it
Use multiple counters that measures the amount for each shape like number of cubes

Now we need to know how many triangles are in each shape.
Once we know how many triangles, now we could estimate it by doing this formula

Number of triangles total = (number of cubes * number of triangles in a cube) + (number of torus * number of triangles in a torus) + (number of sphere * number of triangles in a sphere) + (number of rings * number of triangles in a ring) + number of cones * number of triangles in a cone) + (number of octahedron * number of triangles in a octahedron)

Then after calculating that, you will have your estimated total of triangles.