CS460 Fall 2020
Github Username: JamesEdMichaud
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.

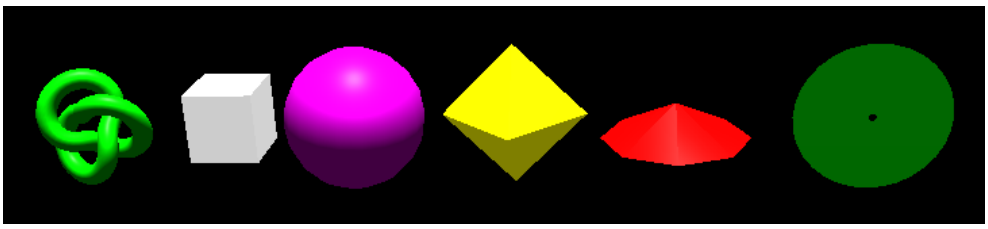
The goal of this assignment is to create multiple different geometries by clicking in the viewport. This means, rather than moving an existing mesh, we will create new ones in the `window.onclick` callback. On each click, our code will randomly choose a different geometry and a random color to place 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, tubeRadius, tubularSegments, radialSegments)
THREE.SphereBufferGeometry	(radius, widthSegments, heightSegments)
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.

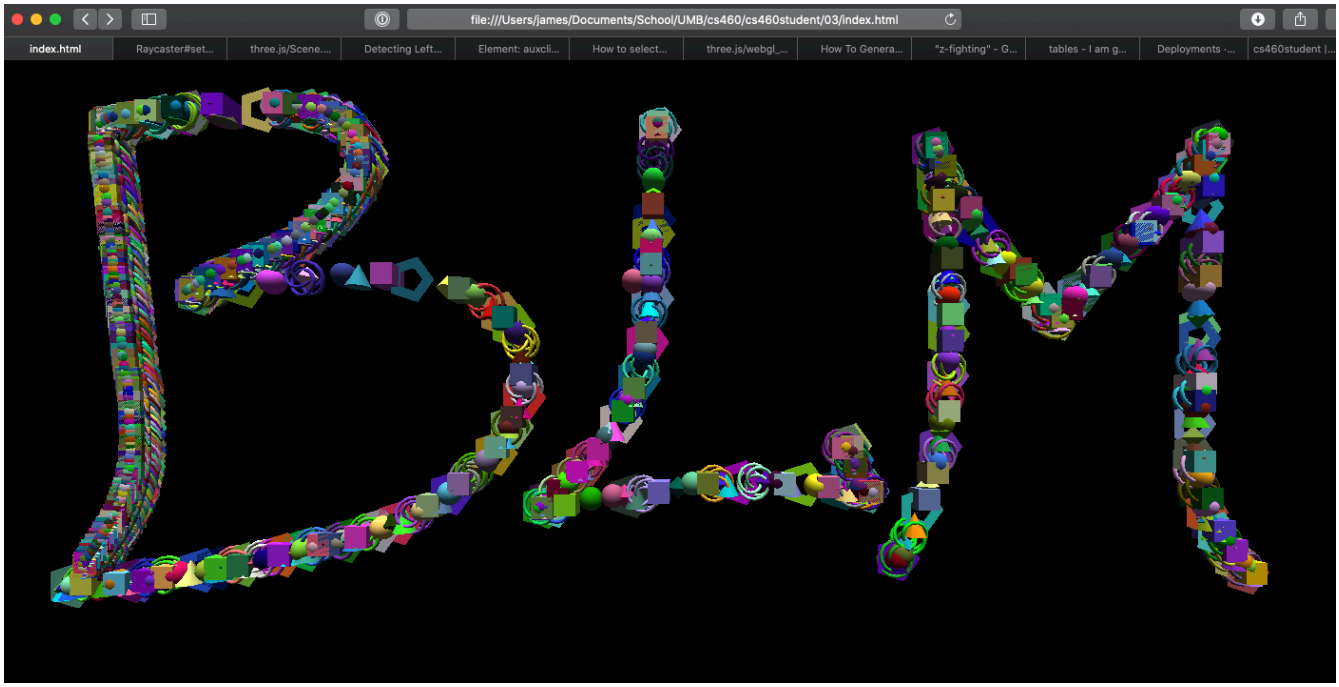
Link to your assignment: `https://jamesedmichaud.github.io`

Bonus (33 points):

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

Yes. I observed Z-Fighting when two of the same objects were placed close enough together. In particular, cubes and rings exhibited this behavior the most frequently.

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**.



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 noticed that objects began to be placed farther apart after about 1300, meaning the mouse position was being updated less frequently. At roughly 3300 objects, my browser warned me that the web page is using a significant amount of energy. At this point it's noticeably slower, but still somewhat smooth. Once 4500 objects were placed, the delay was so great that placing new objects slowed down significantly (and became time-consuming).

What happens if the console is not open during drawing?

I seem to have pre-empted this question... I reduced console output by only printing the number of objects at 50 object increments. Before doing that, printing to the console could not keep up with the rendering, causing massive delays.

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

Yes. Since we know the shapes being drawn and how many triangles each shape contains. We can adjust things a bit so that each shape is drawn once before any shape is drawn again. I've added an array of shape-generating functions. A shape function is removed from the array when drawn, and the array is re-populated with 1 of each shape and shuffled when empty. We can calculate how many triangles are drawn for each group of 6 shapes. Divide the total number of shapes by 6, then multiply by the number of triangles. For example,

	TorusKnot	Box	Sphere	Octo	Cone	Ring	**Total per 6**	1302 objs	3300 objs	4500 objs
Triangles	1024	12	480	8	16	10	1550	669,682	852,500	1,162,500