CS460 Fall 2019
Name: Shijie An
Student ID: 01809165
Due Date: 09/23/2019

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.

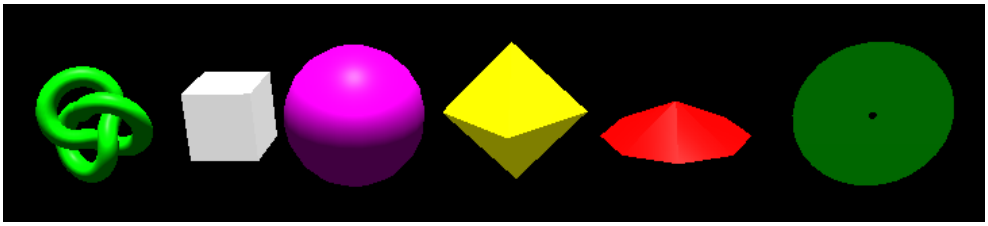
The goal of this assignment is to create multiple different geometries by clicking in the viewport. This means, rather than moving an existing mesh, we will create new ones in the `window.onclick` callback. On each click, our code will randomly choose a different geometry and a random color to place 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, rubularSegament, radiusSegament)
THREE.SphereBufferGeometry	(radius, widthSegament > 3, heightSegament > 2)
THREE.OctahedronBufferGeometry	(radius)
THREE.ConeBufferGeometry	(radius, height, radiusSegament)
THREE.RingBufferGeometry	(innerRadius, outerRadius, thetaSegament > 3)

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.

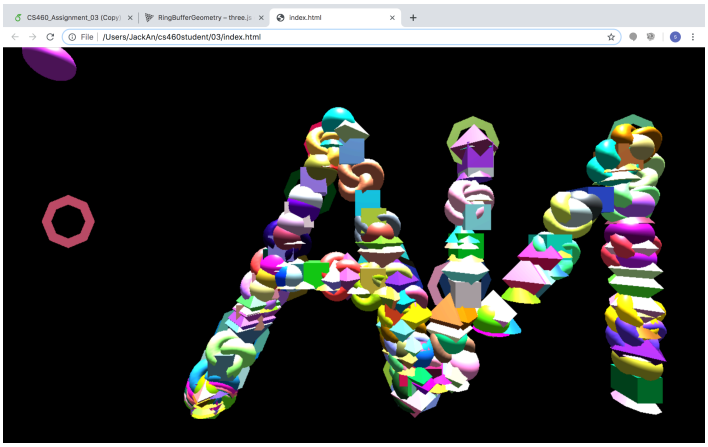
the homework page will be in `https://github.com/Mushaddict/cs460student/blob/master/03/index.html`.

Bonus (33 points):

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

Yes, when two of my objects are very close to each other and they are big enough, there will have z-fighting.

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?

When there are around 150 objects are being created, the renderer will go slower than in the first place.

What happens if the console is not open during drawing?

If the console is not opening, the drawing will not slow down that fast because for the most of time, the console page slows the webpage down. So if we doesn't turn the console on, we would draw more objects faster.

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

Let's say I'm creating 150 objects when the renderer slows down. and for equal probability we have 25 objects for each mesh. Based on my parameter for creating the objects, the cube will have 12 tris, torusKnot will have 64x8 = 512 tris, sphere will have 32*22*2 1408 tris, the octahedron will have 8 tris, the cone will have 10+10 = 20 tris, and the ring will have 11*8*2 = 176. So the totol number we have is (176+20+8+1408+512+12)*25 = 53400 triangles.