CS460 Fall 2021
Name: TUNG DAO
Github Username: tdao71149
Due Date: 09/27/2021

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.

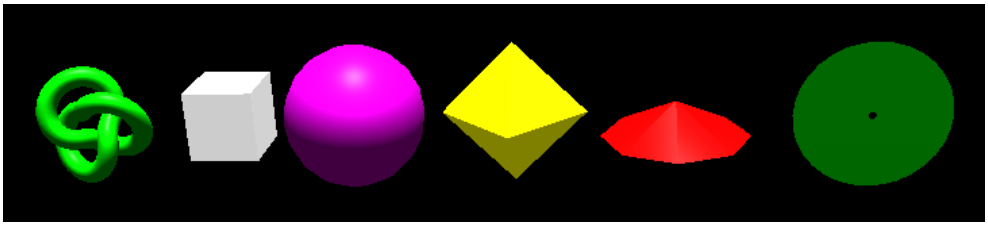
The goal of this assignment is to create multiple different geometries by clicking in the viewport. This means, rather than moving an existing mesh, we will create new ones in the `window.onclick` callback. On each click, our code will randomly choose a different geometry and a random color to place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

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://tdao71149.github.io/cs460student/03/`

Bonus (33 points):

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

I can only observe Z-Fighting when 2 Octahedron stacks on top of each other at the same coordinate. No other geometries did this. However, if a Sphere of size (8, 8, 8) stacks on top of a Ring of size (5, 8, 64, 1, 0, Math.PI*2), it created a one-way outline where you can only see the ring from 1 direction, if you rotates the camera it disappear. I don't think it's Z-Fighting though.

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?

The browser started to get slower at around 500-600 objects.

What happens if the console is not open during drawing?

Not sure what this question means because you don't need the console opened while drawing.

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

Amount of triangles each geometry has:
Ring: 64*2 = 128
Cone: 8*2 = 16
Box: 2*6 = 12
Sphere: 14*8 = 112
Octahedron: 4*2 = 8
TorusKnot: (Tubular segments * Radial segments * 2): 100*16*2 = 3200
Average: (128 + 16 + 12 + 112 + 8 + 3200) / 6 = 580 triangles/object
=> 580 triangles/object * 600 objects = 347600 triangles is when it starts slowing down.