CS460 Fall 2020
Github Username: sg9293
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, 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://sg9293.github.io/cs460student/03/index.html`

Bonus (33 points):

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

Yes we observe Z-fighting when we place one object to another and GPU is confused which object will go on top, we saw this more on the onmousemove function where there were many objects were coincidice on each other.

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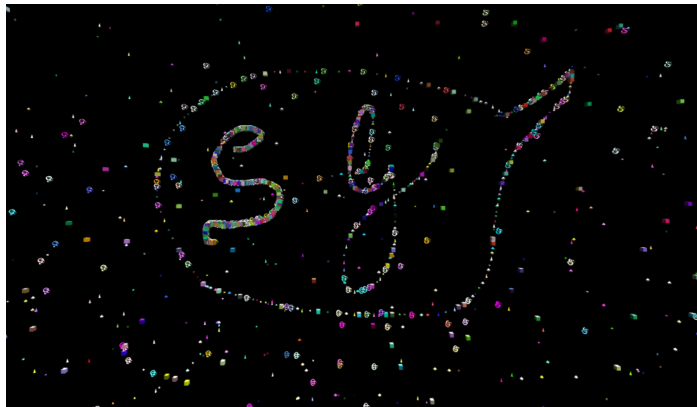
Part 3 (18 points): Please keep track of the number of placed objects and print the count in the JavaScript console. Now, with the change to `window.onmousemove`, after how many objects do you see a slower rendering performance?

The process starts slowing at 270 objects but still it's good, after that around 650. objects it becomes even slower and after 850 objects the console takes start taking noticeable processing time before showing vertices and number of count but there is no affect on speed of rendering the objects on screen even if the count reaches 5 digit number.

What happens if the console is not open during drawing?

If the console is not open there is no effect on rendering objects on the screen, we just will not able to get the positions of the object and the count of objects, rest everything will work as normal as it is supposed to work.

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

No we can not estimate the total number of triangles as only one objects has a huge number of triangles and it's hard to estimate the number of triangles at any given time considering we are taking random objects at each iteration.