CS460 Fall 2020
Github Username: BobbyXing
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.

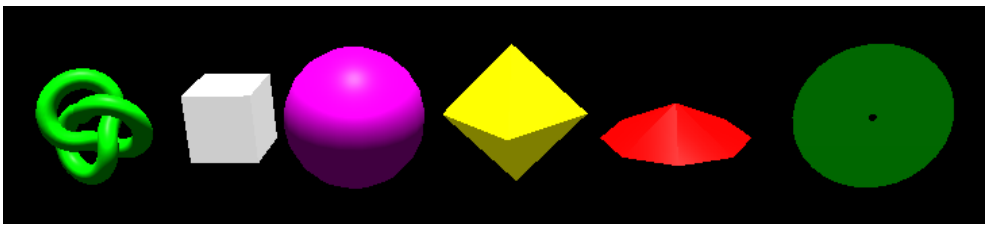
The goal of this assignment is to create multiple different geometries by clicking in the viewport. This means, rather than moving an existing mesh, we will create new ones in the `window.onclick` callback. On each click, our code will randomly choose a different geometry and a random color to place 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	(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://github.com/BobbyXing/Pengbo`

Bonus (33 points):

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

Yes, I do observe Z-Fighting when I place two or more geometric graph together, which means that their triangles overlaps and it do not know which on should be showed first.

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?

After about 5141 objects, the rendering performance become slower.

What happens if the console is not open during drawing?

If not open the console and draw the objects, the console still track the number of objects because when I open the console after finishing drawing, it still show us the number of objects. One interesting thing that I find is when we first not open the console and draw some thing, then open console, the geometric graph that we draw not on the cursor (I put my console at right of the page). I think is because if we open the console, the x-axis become smaller than it previous. So it will not show on the right place.

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

Yes, I think we could, we could use: console.log(renderer.info.render.triangles) to keep track of the triangles. The total triangles are 22529784 when slow-down occurs.