

CS460 Fall 2020

Github Username: YueSun0609

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.

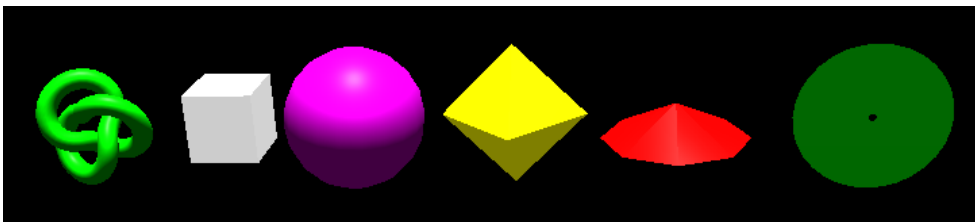
The goal of this assignment is to create multiple different geometries by clicking in the viewport. This means, rather than moving an existing mesh, we will create new ones in the `window.onclick` callback. On each click, our code will randomly choose a different geometry and a random color to place 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
<code>THREE.BoxBufferGeometry</code>	(width, height, depth)
<code>THREE.TorusKnotBufferGeometry</code>	(radius, tube, tubularSegments, radialSegments)
<code>THREE.SphereBufferGeometry</code>	(radius, widthSegments, heightSegments)
<code>THREE.OctahedronBufferGeometry</code>	(radius)
<code>THREE.ConeBufferGeometry</code>	(radius, height)
<code>THREE.RingBufferGeometry</code>	(innerRadius, outerRadius, thetaSegments)

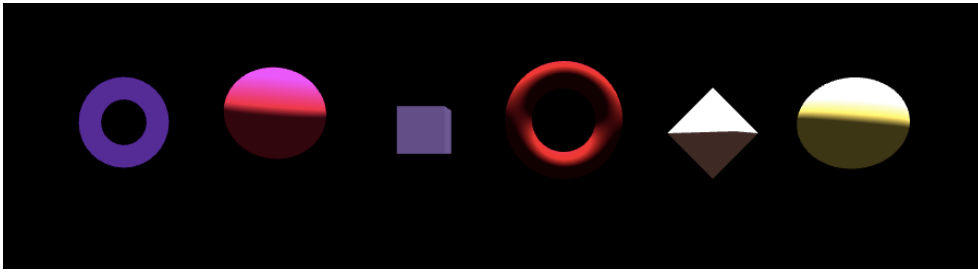
Please write code to create one of these six geometries with a random color on each click at the current mouse position. We will use the `SHIFT`-key to distinguish between geometry placement and regular camera movement. Copy the starter code from <https://cs460.org/shortcuts/08/> and save it as **03/index.html** 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.

My result after clicking mouse on the screen for 6 times shows like this:



Link to my assignment: <https://github.com/YueSun0609/cs460student/tree/master/03>

Note that you need to comment/uncomment line75 and line76 to switch between "onclick" and "onmousemove".

Bonus (33 points):

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

MY ANSWER:

Yes, when I click two positions so close to each other that two symbols will overlap.

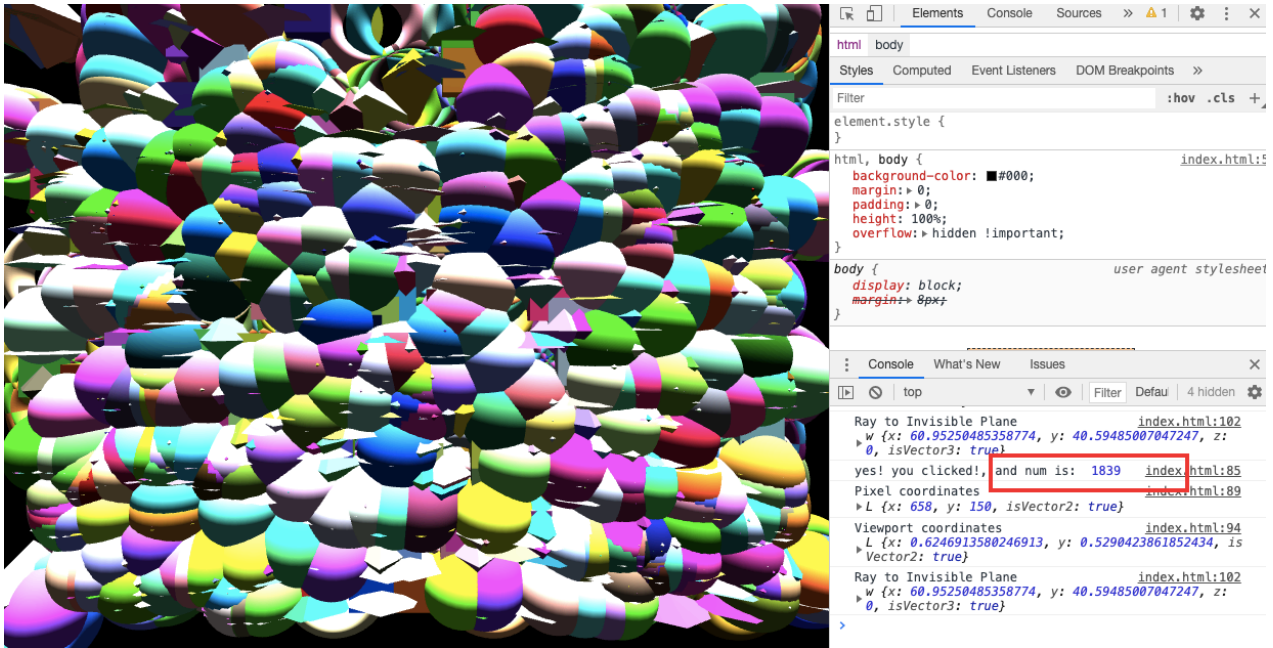
Part 2 (10 points): Please change `window.onclick` to `window.onmousemove`. Now, holding **SHIFT** and moving the mouse draws a ton of shapes. Submit your changed code as part of your `03/index.html` file and **please replace the screenshot below with your drawing**.



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?

MY ANSWER:

I track a variable called "num" to count the number of objects that I placed, and it can be tracked in `console.log` shows below. After num is 1839, I felt rendering becomes slower.



What happens if the console is not open during drawing?

MY ANSWER:

I found I can draw more objects before it becomes slower if I don't open console window.

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

MY ANSWER:

There are many methods to do so.

Let's define the total number of triangles is N .

Method 1: I can roughly calculate the average number of triangles per object among these 6 objects, say n_{aver} . Let's define the number of objects when slow is n_0 . So the total number of triangle is: $N = n_0 \times n_{average}$.

Method 2: Write some lines in our index.html to calculate it accurately. Define the number of triangles of each type of object as n_i , where $i \in [0, 5]$. When it appears, add that number to the total number variable N , and we can also track N in console.log.