University of Massachusetts Boston CS460

CS460 Fall 2019
Name: Yahiya Hussain
Student ID: 01715881
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.

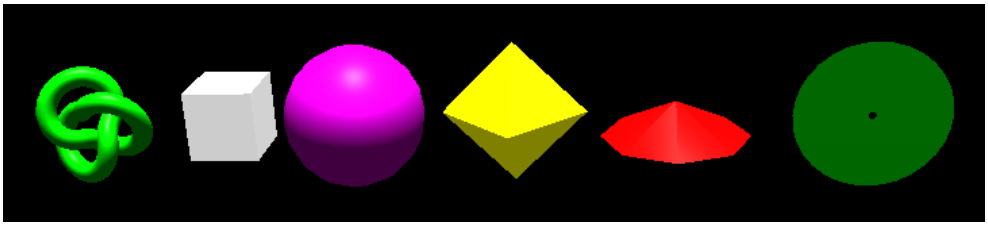
The goal of this assignment is to create multiple different geometries by clicking in the viewport. This means, rather than moving an existing mesh, we will create new ones in the `window.onclick` callback. On each click, our code will randomly choose a different geometry and a random color to place 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, tubeRadius, 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://yahtzeerage.github.io/CS460Assignment3Website/`

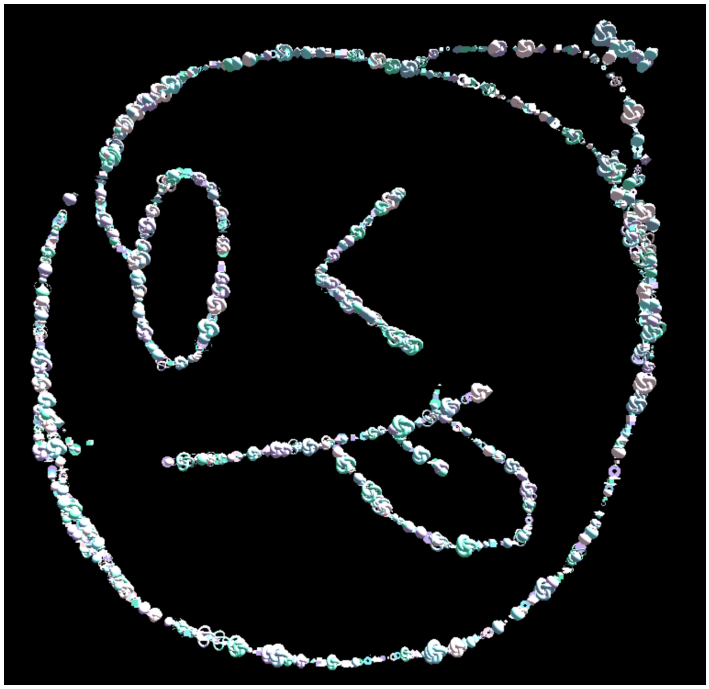
Bonus (33 points):

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

YOUR ANSWER

Yes I observed a lot of z-fighting when shapes were positioned on top of each other and it was ambiguous whose color value was on top. This effect went away a bit when I started varying the sizes of the shapes because more often now, shapes have clear differences in height to solve z-fighting. However at borders of textures, it still often occurs.

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?

YOUR ANSWER I went to 3500 objects before seeing stuttering performance. It seems zooming in and out of the scene starts slowing down and stuttering when too many objects are on the scene.

What happens if the console is not open during drawing?

YOUR ANSWER It appears that the stuttering is a quite reduced when the console is closed, it seems I can handle another 500 objects perhaps before achieving the same levels of slowdown.

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

YOUR ANSWER Using quick and dirty calculations based on my understanding of the mesh shapes, About 403420 triangles appear to need to be rendered on the scene before slow-down occurs.