University of Massachusetts Boston CS460

CS460 Fall 2022
Name: Josh Glazer
Github Username: talkingEagle
Due Date: 09/26/2022

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.BoxGeometry` 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.

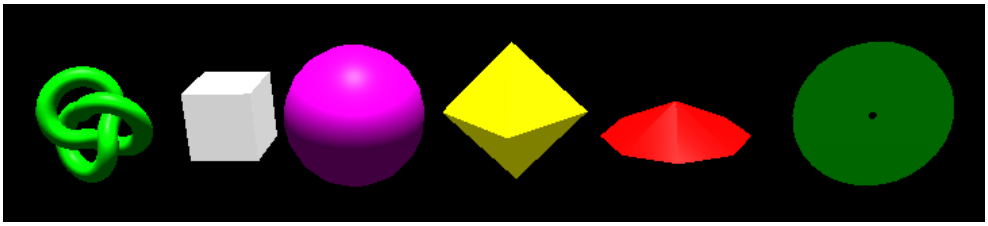
The goal of this assignment is to create multiple different geometries by clicking in the viewport. This means, rather than moving an existing mesh, we will create new ones in the `window.onclick` or `renderer.domelement.onclick` callback. On each click, our code will randomly choose a different geometry 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.BoxGeometry	(width, height, depth)
THREE.TorusKnotGeometry	(radius, tube, tubularSegments, radialSegments)
THREE.SphereGeometry	(radius, widthSegments, heightSegments)
THREE.OctahedronGeometry	(radius)
THREE.ConeGeometry	(radius, height)
THREE.RingGeometry	(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 `renderer.domElement.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.

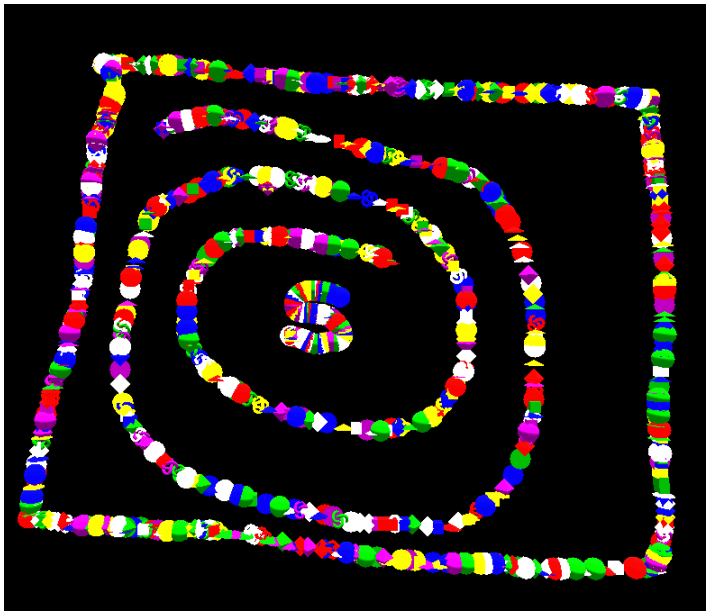
Link to your assignment: `https://talkingeagle.github.io/cs460student/03/index.html`

Bonus (33 points):

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

You wouldn't see Z fighting as much in onclick but onmousemove there is much more overlap between objects that z fighting is much more apparent. There are many objects overlapping coordinates and this causes the objects to fight for the front. Since we are using an invisible plane, all the objects are sitting on the same Z coordinate and this causes the objects to fight for the front because they all sit on the same plane.

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 `renderer.domElement.onmousemove`, after how many objects do you see a slower rendering performance?

I added code to print to console log the count and after a little over 1500 clicks my console began loading slower and giving me run time errors. I kept zooming out the screen and creating more objects with onmousemove but the rendering kept up well.

What happens if the console is not open during drawing?

If the console is not open during drawing then it will lag behind because there are so many onousemove objects that it can't keep up with all the console messages. I received a console message saying 9320 console messages are not shown so that must be all the onmousemove objects. All the renderings can lag the console messages.

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

The box has the least amount of triangles at just two per side, and the other objects having much more triangles than that, the number must be pretty high. Say there are 1500 objects and they each average a number of 25 triangles an object then 1500x25 = 37,500 triangles at slow down.