University of Massachusetts Boston CS460

CS460 Fall 2022
Name: Pablo Bendiksen
Github Username: pablobendiksen **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.

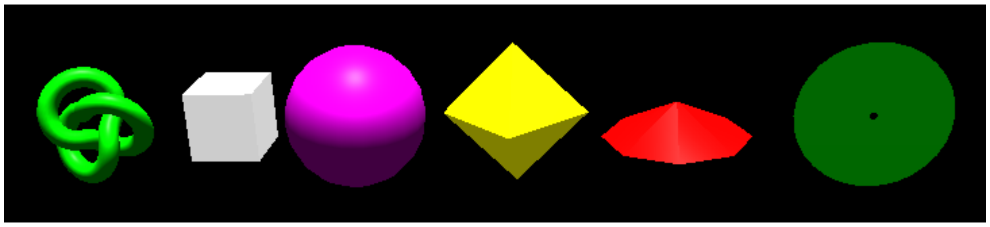
The goal of this assignment is to create multiple different geometries by clicking in the viewport. This means, rather than moving an existing mesh, we will create new ones in the `window.onclick` or `renderer.domelement.onclick` callback. On each click, our code will randomly choose a different geometry 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:



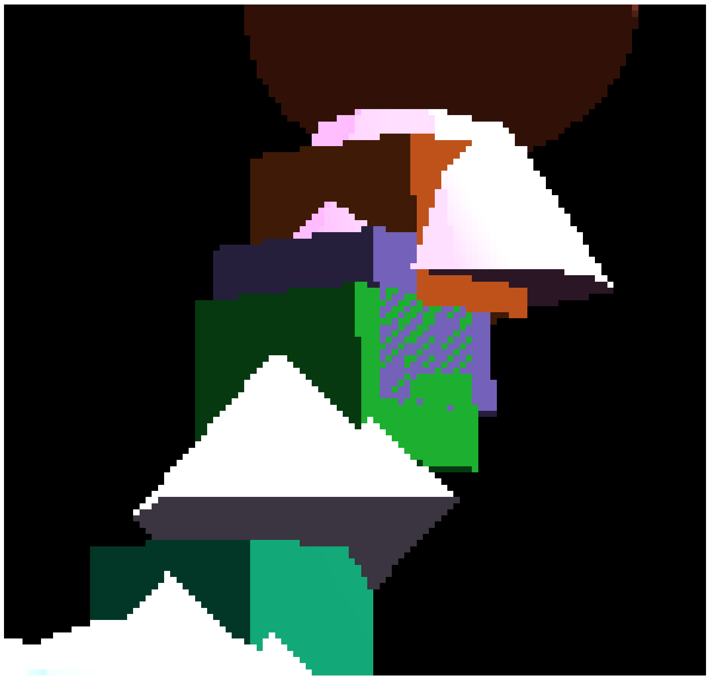
Please make sure that your code is accessible through Github Pages. Also, please commit this PDF and your final code to your Github fork, and submit a pull request.

Link to your assignment: `YOUR_GITHUB_PAGES_URL`

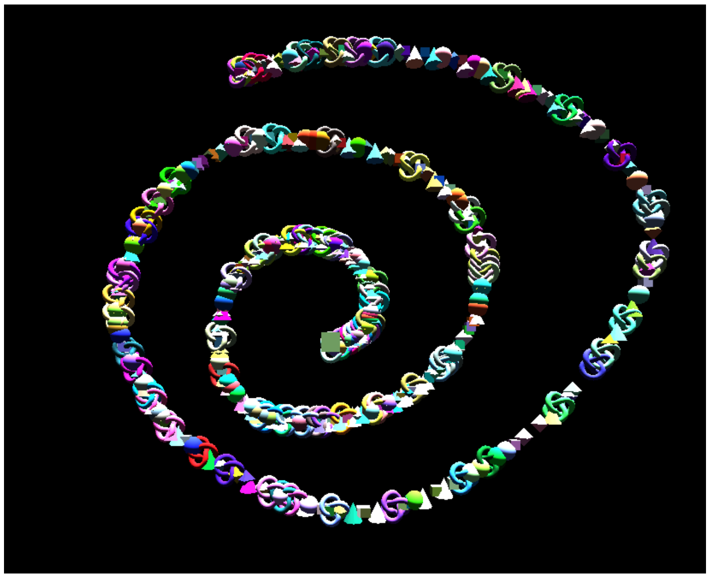
Bonus (33 points):

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

Yes. As illustrated below, we have a clear case of Z-Fighting between two cube geometries. It is during spatial overlaps between same-sized geometries that z-fighting can be observed.

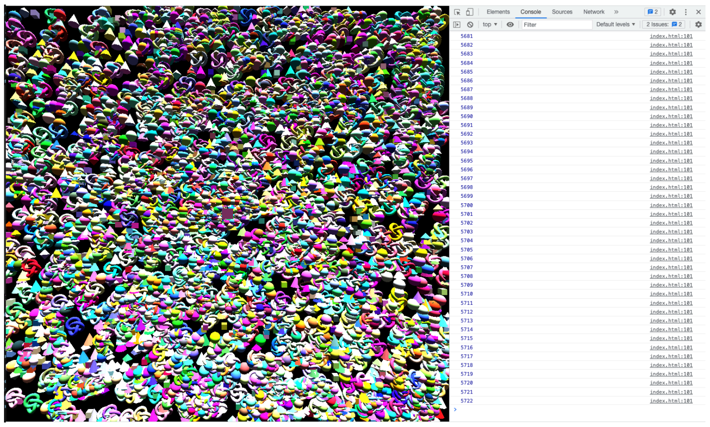


Part 2 (10 points): Please change `window.onclick` to `window.onmousemove`. Now, holding `SHIFT` and moving the mouse draws a ton of shapes. Submit your changed code as part of your `03/index.html` file and **please replace the screenshot below with your drawing**.



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?

To be perfectly honest, after multiple attempts, I still find that I cannot perceive a slower rendering performance; inspect the following image to see that I even saturated the screen with over 5,700 geometry instances ... I believe that either my ability to perceive incremental visual changes is below-average, or that the latest MacBook Pros have impressive GPUs, or both.



```
5681                                   index.html:101
5682                                   index.html:101
5683                                   index.html:101
5684                                   index.html:101
5685                                   index.html:101
5686                                   index.html:101
5687                                   index.html:101
5688                                   index.html:101
5689                                   index.html:101
5690                                   index.html:101
5691                                   index.html:101
5692                                   index.html:101
5693                                   index.html:101
5694                                   index.html:101
5695                                   index.html:101
5696                                   index.html:101
5697                                   index.html:101
5698                                   index.html:101
5699                                   index.html:101
5700                                   index.html:101
5701                                   index.html:101
5702                                   index.html:101
5703                                   index.html:101
5704                                   index.html:101
5705                                   index.html:101
5706                                   index.html:101
5707                                   index.html:101
5708                                   index.html:101
5709                                   index.html:101
5710                                   index.html:101
5711                                   index.html:101
5712                                   index.html:101
5713                                   index.html:101
5714                                   index.html:101
5715                                   index.html:101
5716                                   index.html:101
5717                                   index.html:101
5718                                   index.html:101
5719                                   index.html:101
5720                                   index.html:101
5721                                   index.html:101
5722                                   index.html:101
>
```

What happens if the console is not open during drawing?

The rendering of geometries is perceptibly (but minimally) faster.

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

This question can be answered in principle, so long as slow-down can be perceived by the user. As soon as slow down occurs one will have to rely on attributes of each geometry to answer the question. The vertex count is always found via the attributes (i.e. attributes.position.count) for each geometry. However, the number of rendered triangles depends on whether the geometry is indexed (which allows for faster rendering) or not. If so, number of rendered triangles = index.count/3, otherwise it is position.count / 3.

Assuming indexed geometries we can therefore determine the count of rendered triangles, per geometry, with index.count /3. Once we have this value per geometry, we will assume an equal number of each geometry present upon slow-down (as each geometry is sampled with equal probability) for our estimate:

geometry-count (for any given geometry) = total-geometries-count / 6.

Estimate = geometry-count * index.count /3 (for Box) + geometry-count * index.count /3 (for Torus Knot) + geometry-count * index.count /3 (for Sphere) + geometry-count * index.count /3 (for Octahedron) + geometry-count * index.count /3 (for Cone) + geometry-count * index.count /3 (for Ring)