CS460 Fall 2019
Name: Loraine Franke
Student ID: 01881004
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. We will be using six different geometries. Before we start coding, we want to understand their parameters. Please complete the table below.

Constructor	Parameters
THREE.BoxBufferGeometry	(width, height, depth)
THREE.TorusKnotBufferGeometry	(radius, tube, 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. **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.**

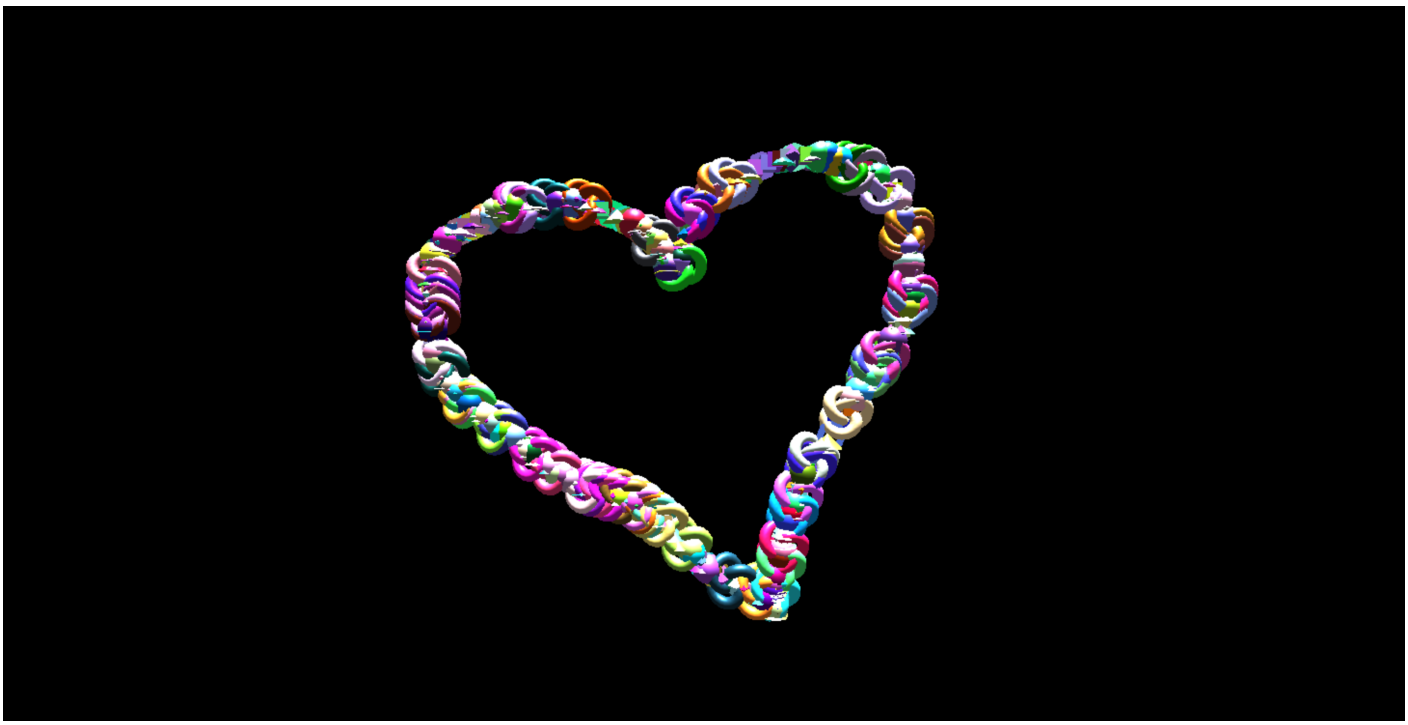
Link: `https://lorifranke.github.io/cs460student/03/`

Bonus (33 points):

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

Yes, when the rendered geometries have parts that are randomly lying in the same plane (above each other) and are occupying the same space.

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?

The rendering performance could for example be measured with the frames per second (fps). `https://github.com/mrdoob/stats.js/` shows different info boxes on the javascript performace that can be implemented as script in the html document. At the beginning the performance is at 60fps, drops at 1000 objects to 50 fps, at 2500 objects the frames per second are below 25.

What happens if the console is not open during drawing?

Slight increase in performance because log statements are not visible.

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

Bottom up estimation (checked number of triangles per geometry via logging):
The different geometries have the following number of triangles:

- BoxBuffer: 12
- TorusKnot: 1024
- Sphere: 80
- Octahedron: 8
- Cone: 24
- Ring: 16

Average: (12 + 1024 + 80 + 8 + 24 + 16) / 6 = 194

Since every geometry is equally likely, at 2500 geometries we should have around 2500 * 194 triangles. That estimates around 485000 triangles. In fact at 2500 objects renderer.info.render shows around 470,000 triangles.