Skylar Wurster

CSE 5544

Jian Chen

Assignment 1 – Part 1

Processing

Processing is a sublanguage of Java that includes its own IDE for development. A helpful framework was added on to Java in order to support a rendering engine and a standard game loop. Examples of this framework include the setup() and draw() function calls, which occur on start and at a regular interval, respectively. Processing has also created handy drawing functions, such as line() and ellipse() to draw shapes onto the window automatically created. The color of these shapes can also be defined, as well as the edge color, using color() and stroke() with the RGBA values as inputs. Typical Java had a rather cumbersome and extensive process in order to get a GUI running. Processing takes that difficulty away but keeps the power of Java. Processing can import .jar packages that use Java libraries. It can be run on a host of machines using JVM. Its lightweight and can be built upon to support many kinds of projects. The processing website keeps updated versions of packages made by users, which currently holds hundreds spanning from Data to Animation to Simulation.

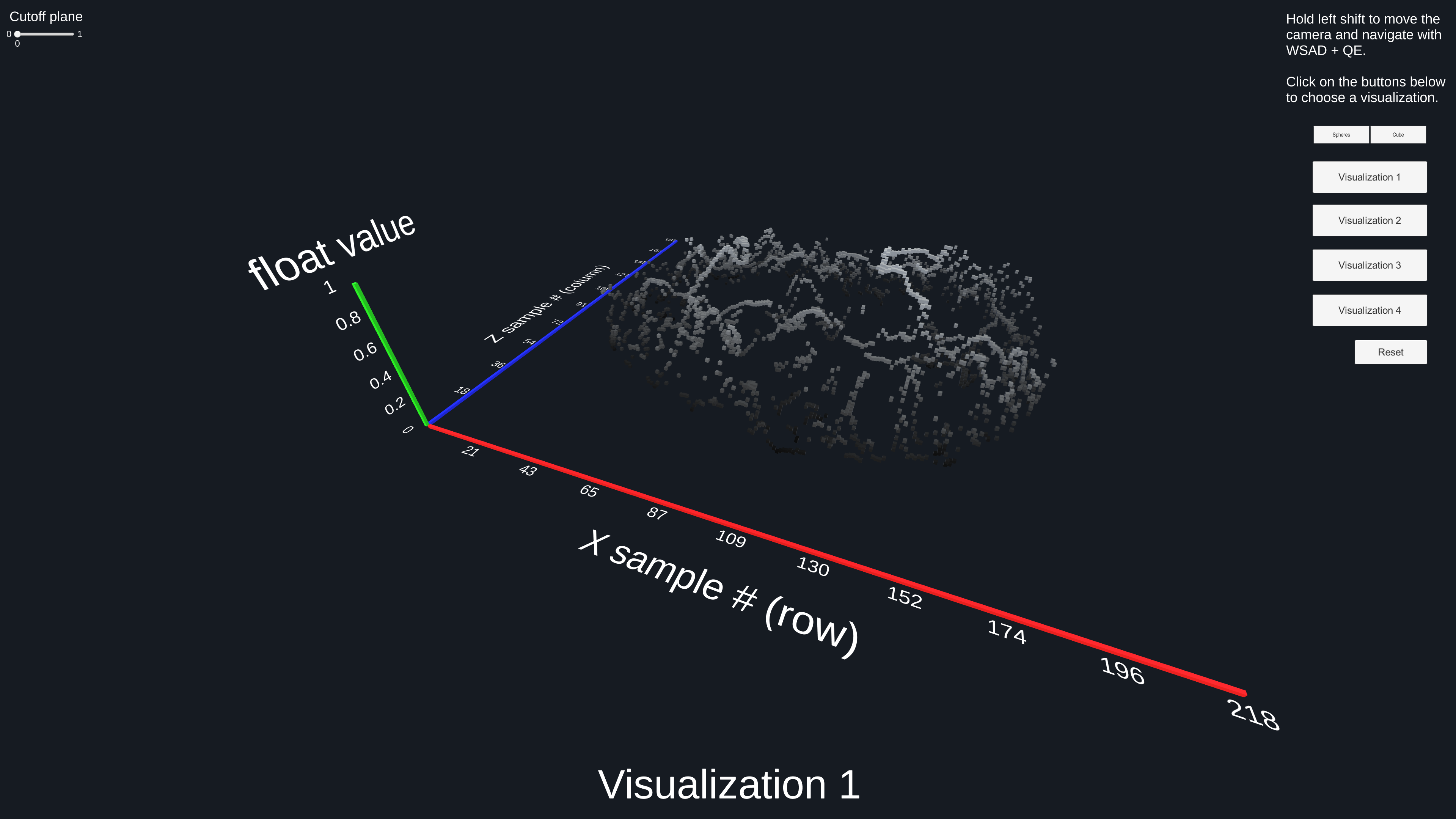
Based on its high-level interface, hundreds of packages available, and the support on the forums, Processing seems to be a good choice for a visualization project. There seem to be many ways to represent both spacial and non-spacial data. Since it lies on top of Java, there is an abundance of other libraries that can be pulled from in order to best display data. This also means it’s not too low level and isn’t going to take a long time to get started. Overall, I feel Processing works very well in the scope of a class like CSE 5544.

D3.js

D3.js is a JavaScript spec that allows for simple manipulations to web pages for visualization purposes. HTML content can be accessed and changed, event listeners can be used, and attributes or styles are easily changed. Functions such as enter() and exit() allow for management of data nodes as needed. D3 uses only web standards, such as HTML, SVG, and CSS, so the spec is well defined. This ensures that it plays nicely on all devices from a browser. In order to begin using it, a simple script import at the top of your HTML document will do it. Since JavaScript doesn’t use explicit typing, many data types are supported. JSON files/variables will be useful when importing data and storing them in memory for display.

Due to its ease of use and dissemination, D3.js could be a very useful data visualization tool for this class. Project updates can be easily presented from the GitHub page housing the D3 document. There is no software or library dependence that could restrict anyone from viewing the visualization. However, I don’t know JavaScript very well, and think that this might make it difficult to do well in a single semester.

Visualization 1:

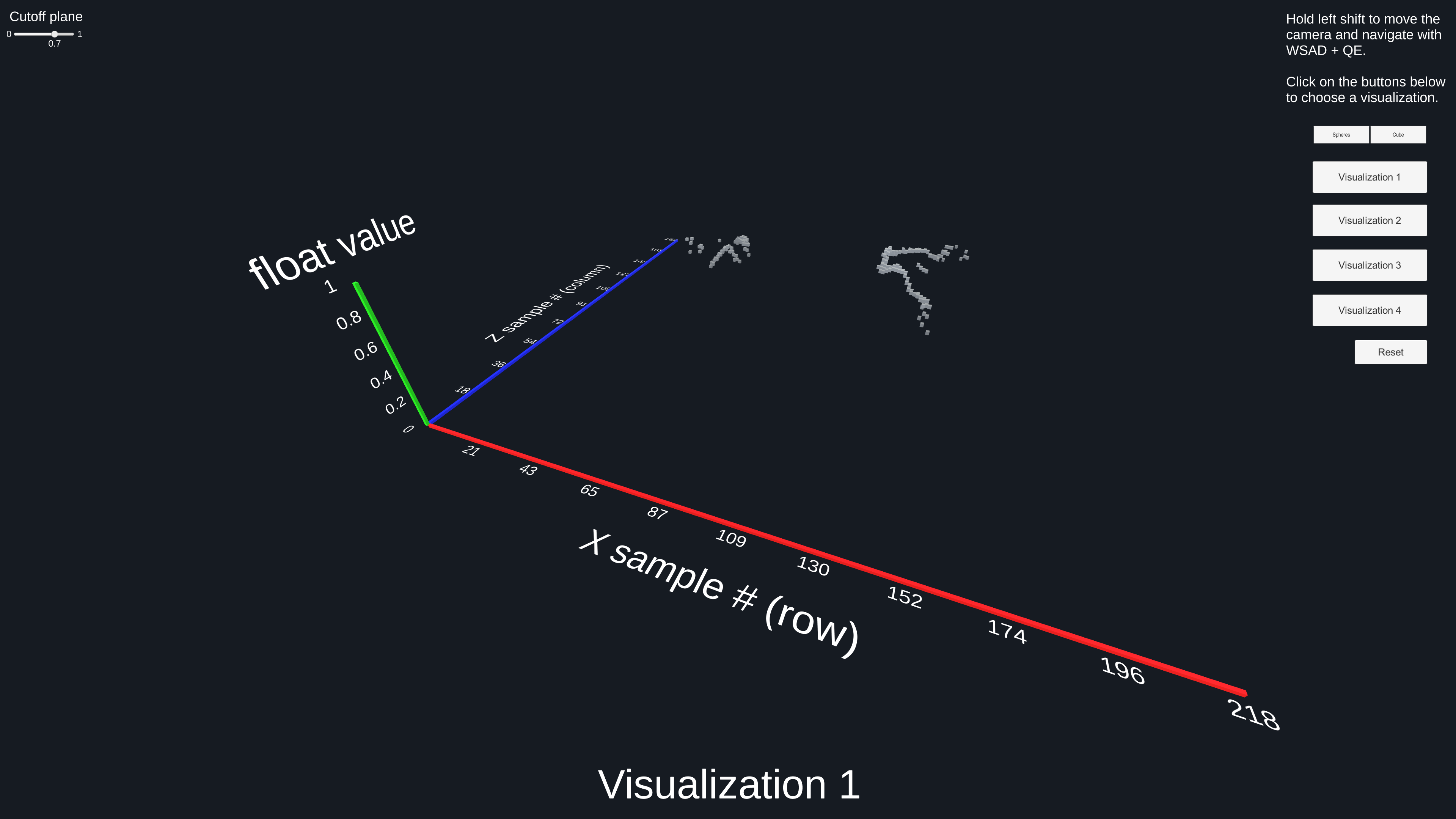


Visualization 1 interprets the data as three-dimensional. Since the data points are on a grid, we place the data point at a height value equal to the float value at that grid point. This means a larger value will be shown as a point higher on the Y axis. Data with a value of 0 is not shown, since it seems unhelpful for a viewer to see the data at 0. This is done to not overwhelm the viewer with more information than necessary.

Color is also added; a grayscale gradient is used to color data points based on the float value. Values closer to 0 are black, values closer to 1 are while. This can help in the 3D scene, since the depth may be hard to understand. A gradient of different colors isn’t used since that won’t add any information and may be misleading.

Cubes are used as each data point, all with a size of 1. Spheres may be selected as well, but cubes are less misleading. Because of their sharp edges, it’s also easier to locate them in 3D space.

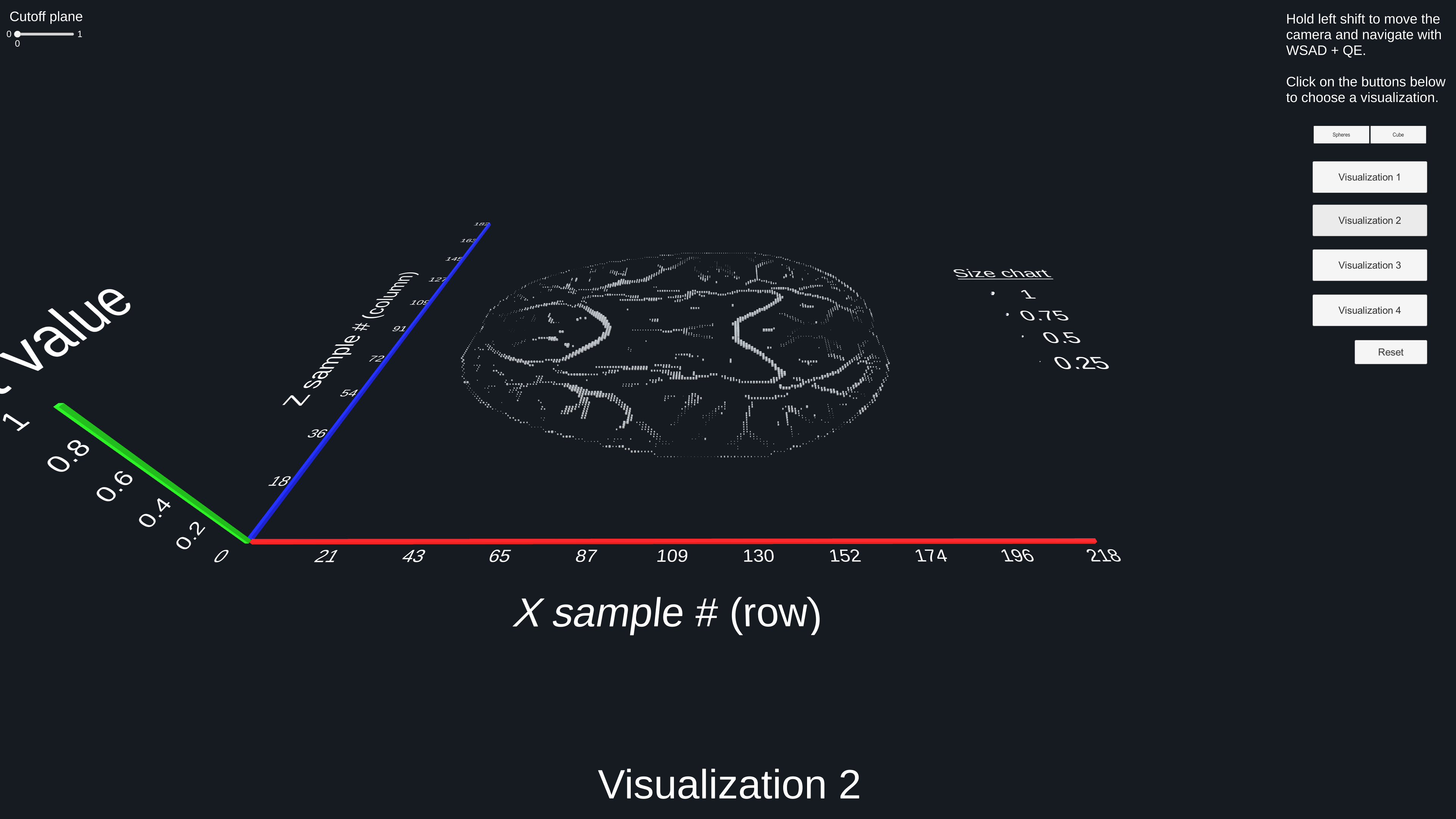
Lastly, the “cutoff plane” is set to 0. This means all data points with a value greater than 0 are shown. Below, observe the visualization when the cutoff plane is set to 0.7.



This allows the user to scroll the slider up and see which data points are smaller than others, and ultimately which points have the largest value.

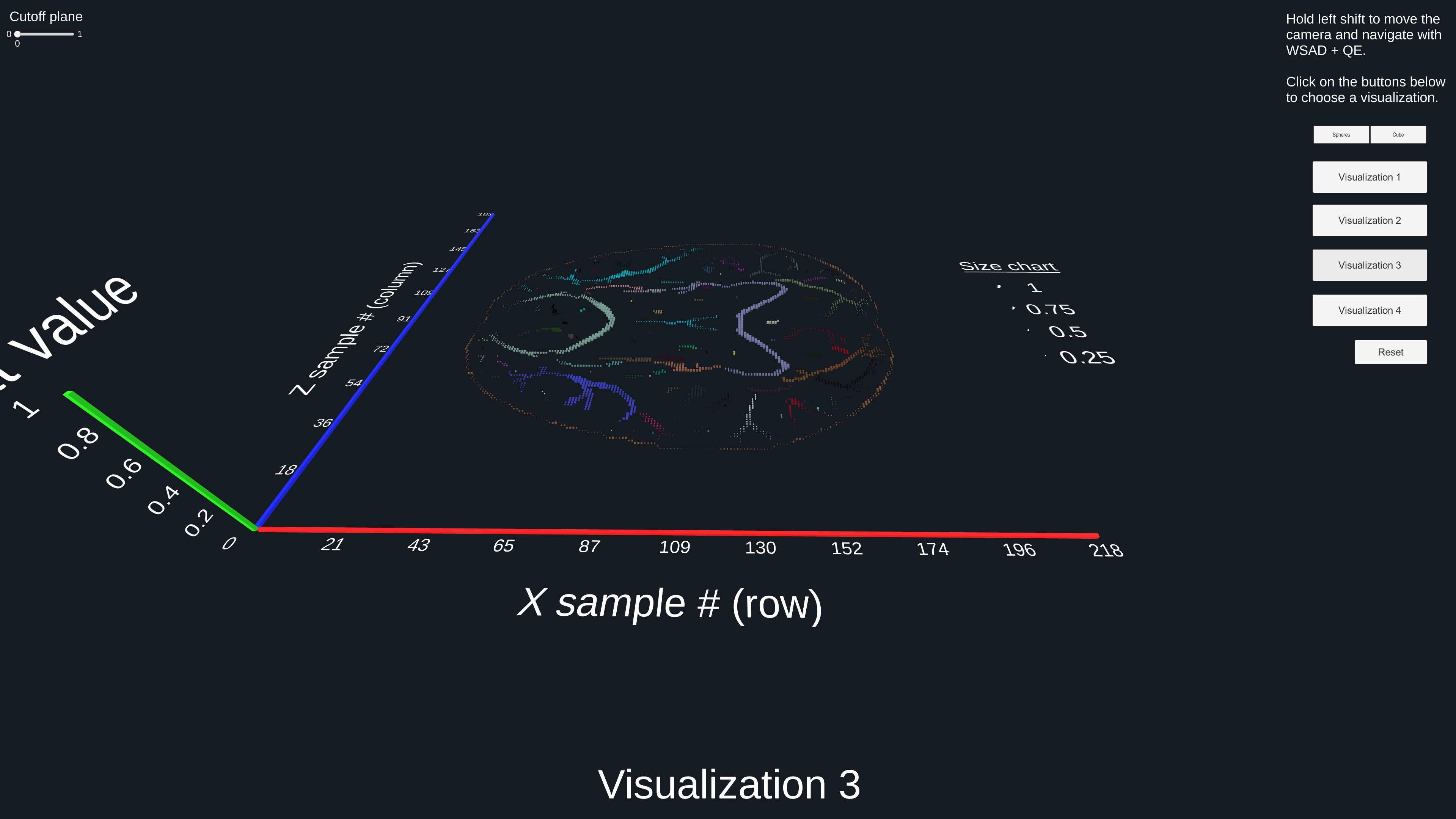
Lastly, there is an animation when transitioning into this visualization which may serve to make clearer which points are higher.

Visualization 2:



In visualization 2, the data is projected onto the XZ plane in a grid. Now, the data points are sized based on their float value. Smaller float values will have a smaller point in the grid, with larger float values being larger. A legend is visible on the right to show what sized cubes correspond to what float values. Though this may be difficult to tell exact values, it is very easy for our eyes to discern the size/volume difference between cubes. For that reason, this visualization is great for comparing the values within the grid, especially for values close to each other on the grid.

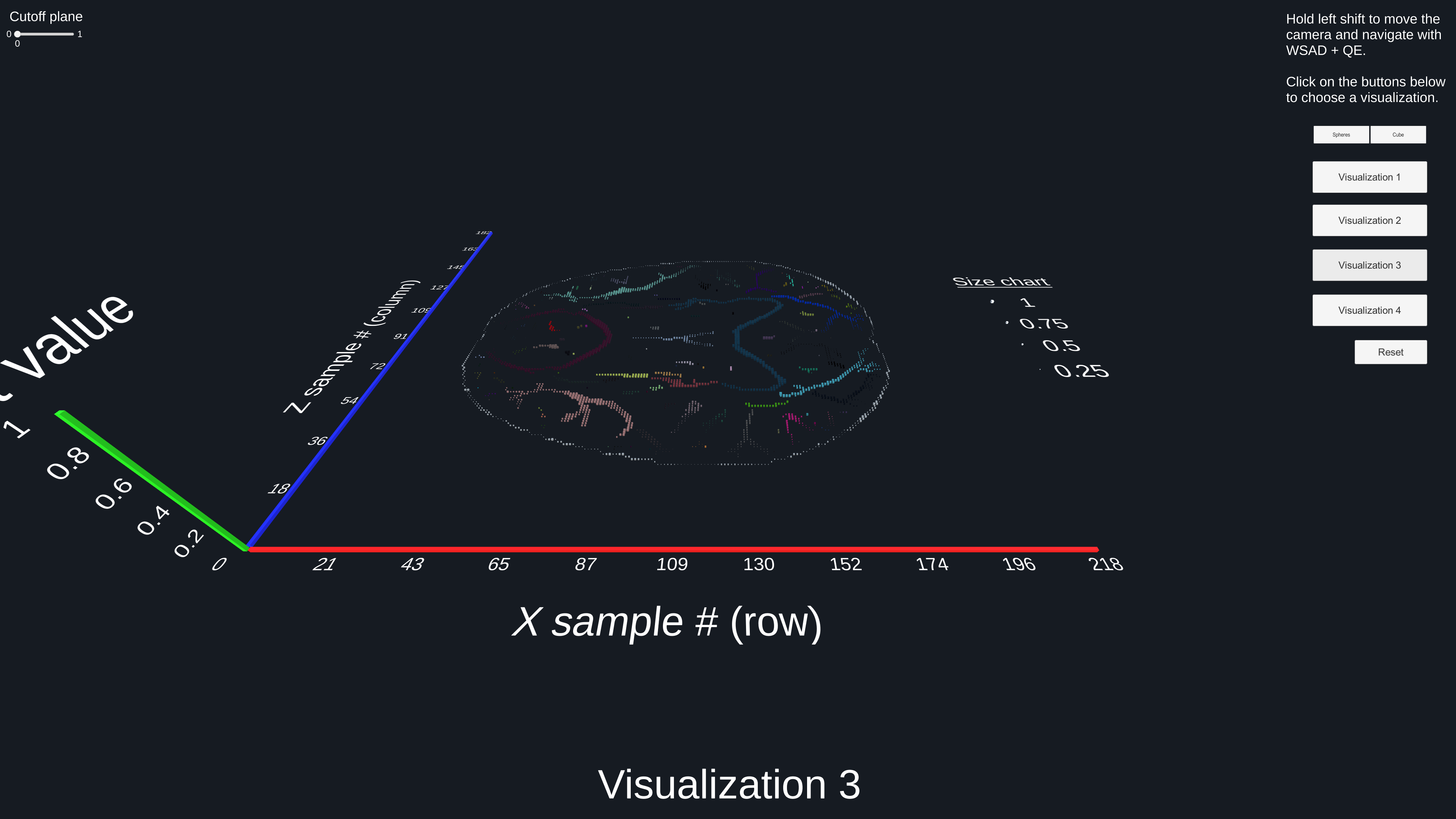
Visualization 3:

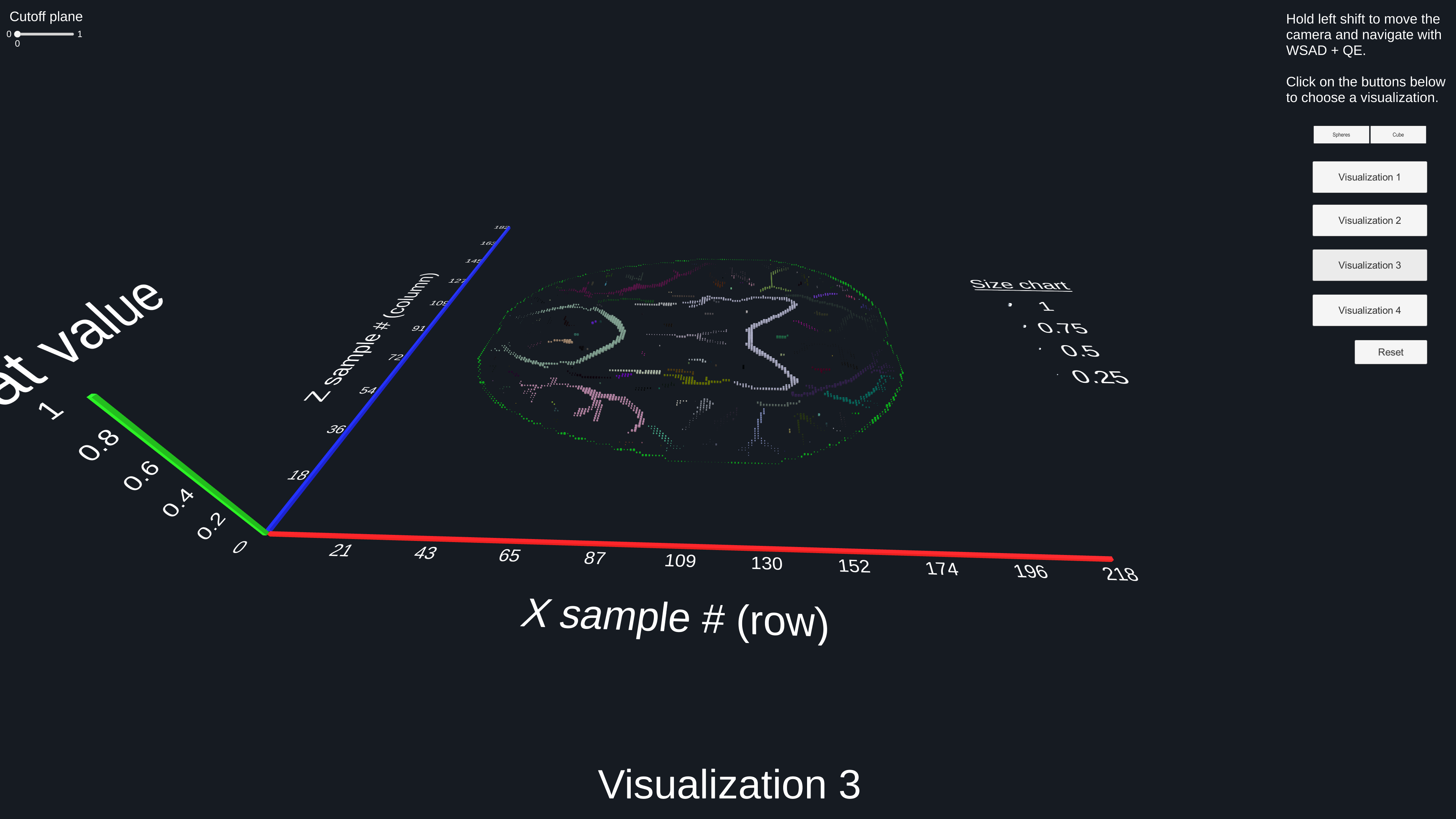


Visualization 3 is the same as visualization 2, with the addition of colored components. The entire dataset has been colored based on connectivity. For instance, all the lavender cubes are nonzero entries in the dataset that are next to each other on the grid. This can help show the connectivity, and perhaps more important the disconnection, between these components in the data.

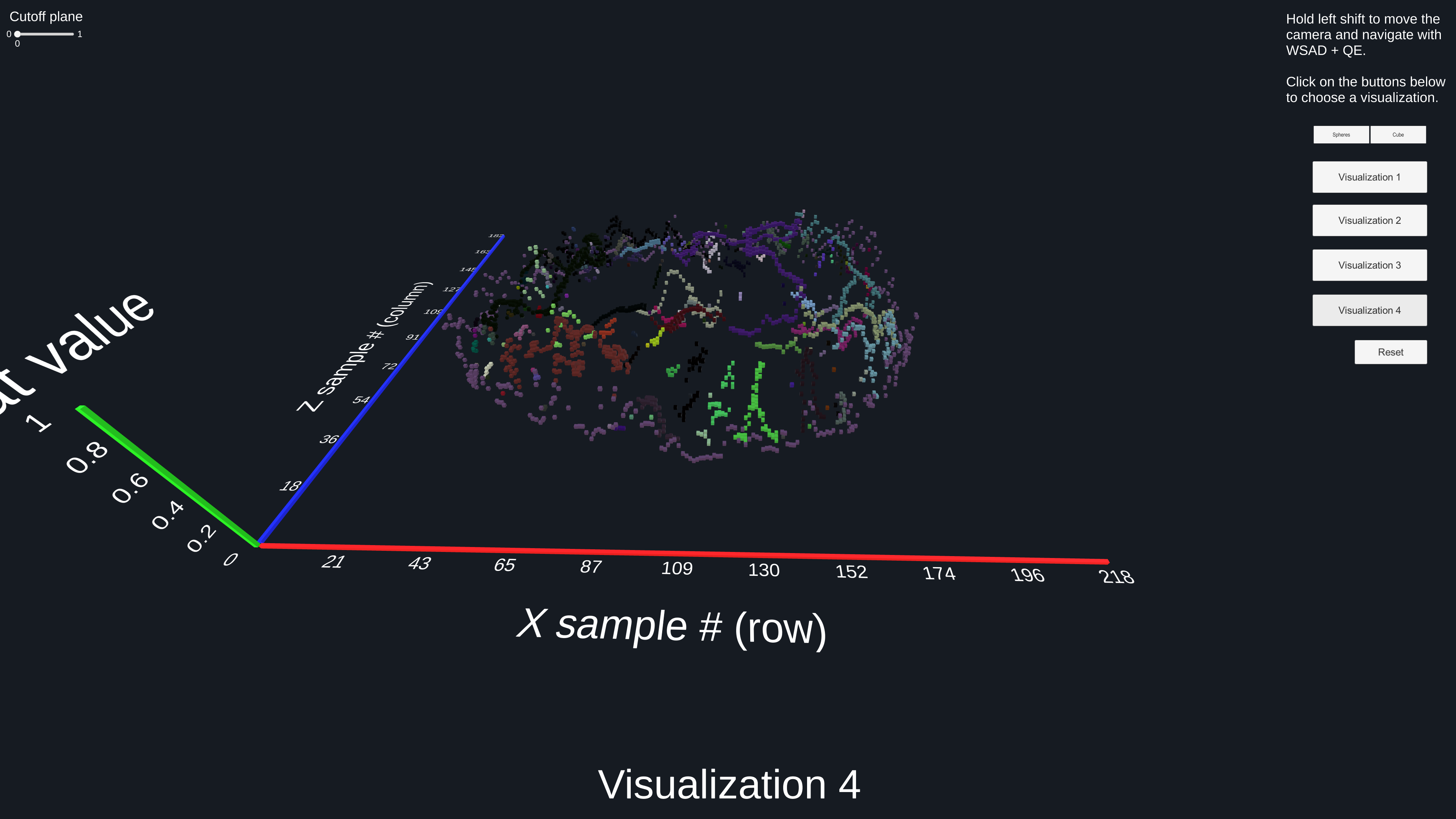
The size component has been carried over from visualization 2, but it doesn’t help here as much with the randomized colors. It might be best removed from this visualization, since it might only confuse the viewer.

As brought up, the colors are randomized. This has good and bad attributes. The good part is that because the visualizations are animated, it is simple and pleasing to watch the colors change when randomized. When doing this, its easy to see new connected components come into focus. Another good thing here is that there probably isn’t a “perfect” coloring for each connected component. A down side is that there isn’t much clarity often. Dark colors distort and murk up the view. Also, multiple people viewing this in live 3D from different computers will have different colors randomized. This means that one collaborator cannot point out the “red connected piece” to the other, since that will be something different. An upgrade for this may involve having a large list of colors for the algorithm to walk through, so that it will always be the same coloring. Below are 2 more randomized color combinations of this visualization.





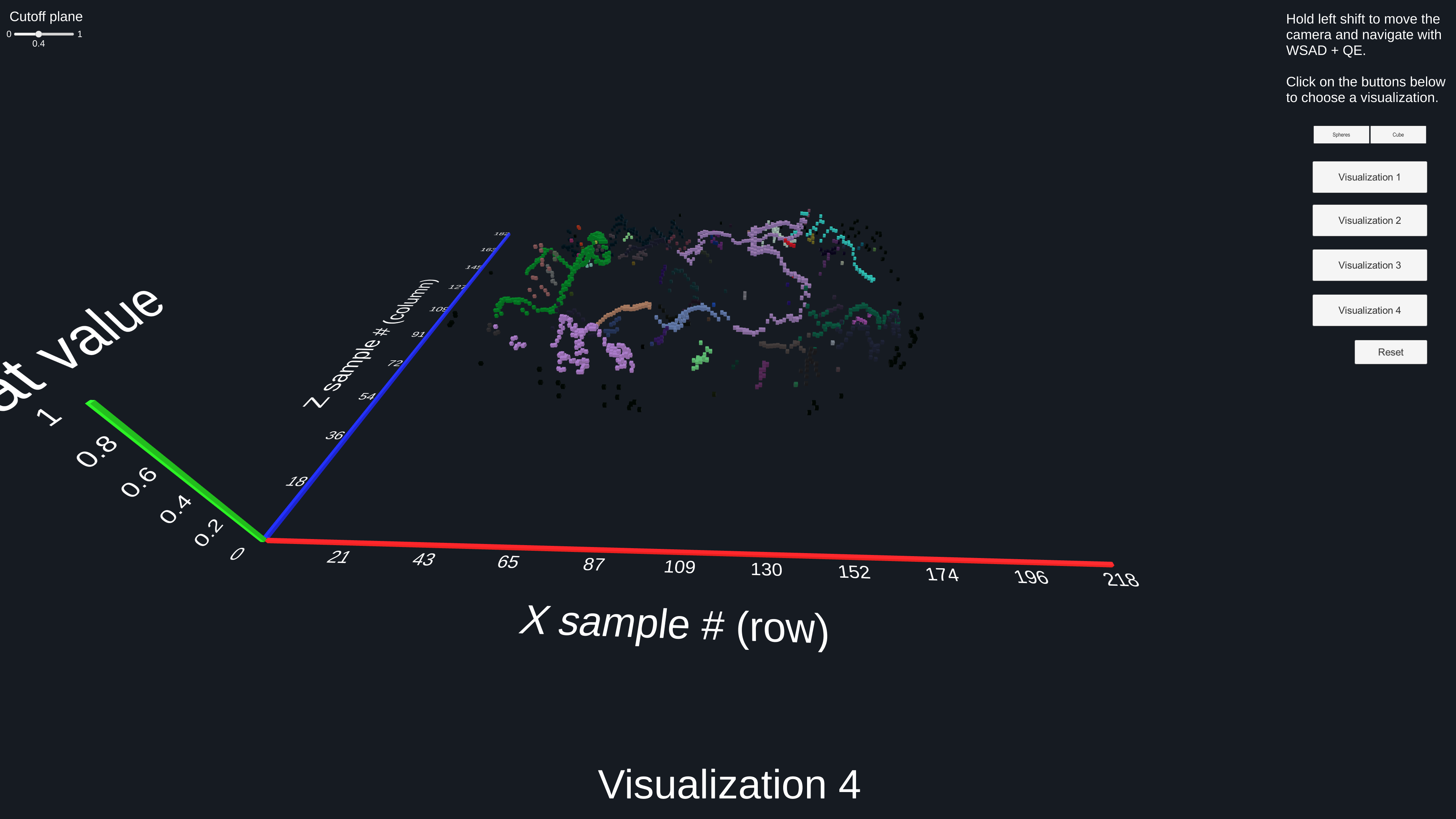
Visualization 4:



Visualization 4 combines visualization 1 with the coloring technique introduced in visualization 3. The grayscale gradient of visualization 1 has been scrapped and replaced with solid colors for connected components.

The intent with this visualization is to allow the viewer to follow connected components in 3D easier than in visualization 1. This also allows the viewer to see the large changes within a single connected component, or the similarity of it. For instance, in the picture above, we see that the purple ring around the outside is all about at the same height. This may mean more to someone more familiar with this data.

The same option for the cutoff plane exists here as it did in visualization 1, and may allow a viewer to see how many connected components have values above some threshold. Here it is for 0.4.



The same pros and cons of using a randomized palate exist here as they did in visualization 3. See two more randomizations below.

