Immersive visualization of data sets in the 3-sphere

Angela George & Alexander Gilbert

Summary:

This is a report of the project conducted in the Mathematical Computing Lab at University of Illinois at Chicago for Fall 2015. For this project, we developed real-time 3D graphics programs for visualization of point-cloud data sets in the 3-dimensional sphere. They target both virtual reality platforms (such as Google Cardboard and Oculus Rift) and desktop computers. We used these programs to explore data sets related to the geometry and combinatorics of curves on compact hyperbolic surfaces.
Our faculty supervisor was David Dumas and our graduate mentors were Nathan Lopez and Jasmine Otto.

The Problem:

The 3-sphere is a higher dimension analogue of a sphere. It lives in the 4-dimensional space. Since it is difficult to visualize the 4-dimensional space, we use stereographic projection. Stereographic function is a function that projects a sphere onto a plane and it is defined on the entire sphere except at the projection point. This type of projection does not preserve the distances nor the volume but it does preserve the angles, that is a circle in a plane has the same image as the circle on a sphere.
The two type of data sets that we use are:
1. Lattices in $\mathbf{R}^4$ such as the integer lattice $\mathbf{Z}^4$
2. Simple closed curves on hyperbolic surfaces – This dataset is built on an earlier work of D. Dumas and F. Guéritaud [3]

Design & Implementation:

To visualize the arrangements of point in the 3-sphere in a useful way, we created two programs. The first is built using Unity 3D graphics framework [2] and the other is a browser-based WebGL application [1]. Each program uses a point cloud to visualize a particular dataset. We also developed a suitable JSON format to hold the information for our point clouds. To obtain point clouds from curves, the work of W. Thurston was used, that is a set of simple closed curves on a hyperbolic surface is realized as a set of dense points in a topological n-sphere, where n depends on the surface [5]. We wanted to build a software that would allow our users to view the 3-sphere in an immersive environment and have the ability to navigate the data set in a meaningful way. To do this, the users were placed at the origin in a 3D space and also have the ability to freely look around the environment created through the stereographic projection of the given 3-sphere data sets.
To implement all these features, we decided to use Unity 5, part of the Unity 3D graphics framework, to handle the graphics. Unity 5 was used as it was an engine that could be easily used and also had the ability to code in a familiar language, C#. Unity, being a 3D game engine, is also equipped with an extensive Math library that aided in many of the calculations. To place the users in the origin in Unity, a camera object was placed in the center of the environment. Unity also handles user input, so the rotation of the camera is tied to the mouse movements. To place

and display points in the environment, a Unity provided construct - “Particle System” is used. With the Particle System, we are able to define a collection of points, giving each a 3D position, size, and color. With all this in place, the next step was to take in a set of points in $\mathbf{R^4}$, radially project each of them onto the 3-sphere (normalize each vector), stereographically project these points into $\mathbf{R^3}$, feed that position to the Particle System, and then provide a way for the users to navigate this data. To navigate the data in a meaningful way, simply allowing the user to move off the origin and 'fly-around' a static point cloud would not suffice. Instead we locked the user to the origin, rotated the 3-sphere and allowed the points to move around the user. By rotating the sphere, the structure of the space and any symmetries it may have become apparent. Determining the 4 dimensional rotations of the 3-sphere was achieved with Unity 5's Quaternion library along with the fact that the space of 4D rotations could be represented with two unit quaternions.

In addition, the user is also able to select a single point out of the entire data set to view any properties the point might have. In particular, when viewing the S(0,5) and N(2,2) data sets users can select a point and view the corresponding curve's word as well as its length. A word here refers to each point in the dataset, simple closed curves on hyperbolic spaces, that can be labeled by an element of the fundamental group of the surface, presented as a word in a fixed generating set.

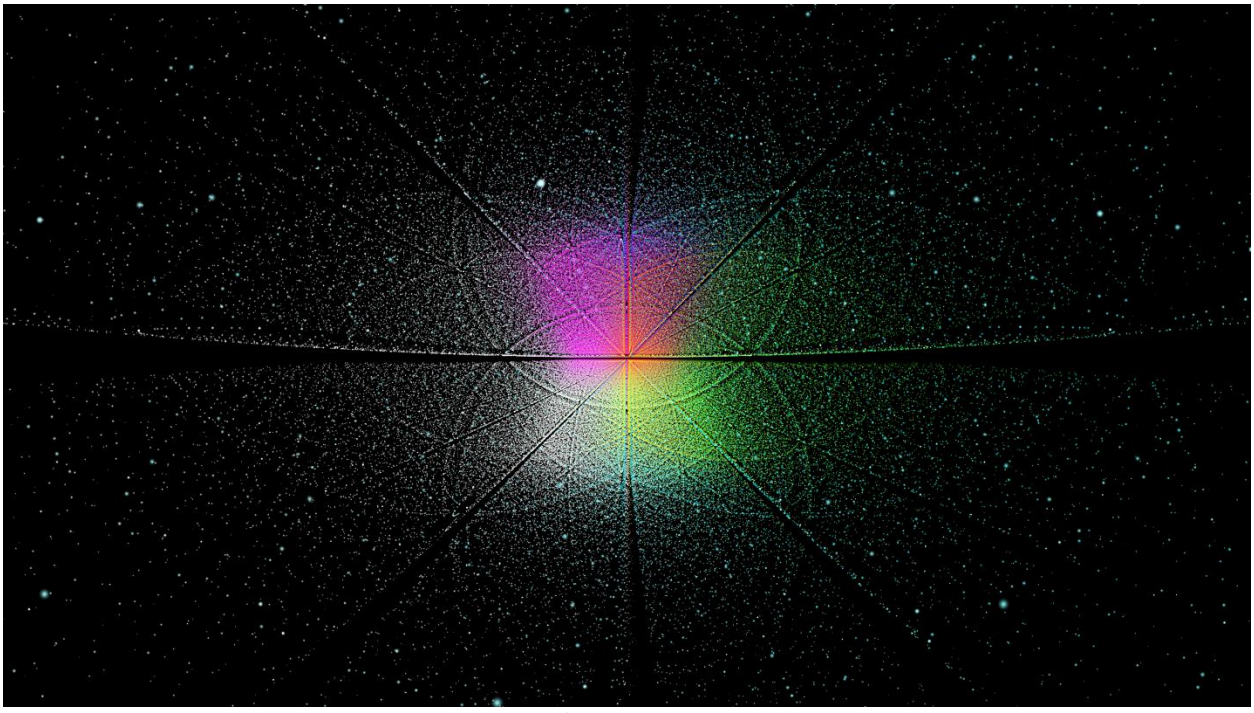
For WebGL, a third party library, known as Potree, was used to load the 3D datasets [4]. Potree is built on the WebGL 3D rendering library, Three.js and plas.io, a point cloud viewer. Potree takes files in xyz, laz, and las format and uses a converter to convert into octree format. This is then generated as a web page for users to view through any modern browser. Since multiple browsers enable strict security policies, a web server, such as xampp, is used to view this web page. The web page that Potree generates to view has multiple camera controls such as Earth controls, that lets users pivot around a point, users can use the right mouse to pan the camera and the pivot, the Fly Navigation that lets users use the WASD keys or the arrow keys to move through the environment and the left mouse to rotate the camera, the Orbit Navigation that lets users rotate the point cloud as a whole. In addition to the camera controls, Potree also has controls to adjust the size, shape, colour and opacity of the points visualized. Because Potree takes data sets as x, y, z coordinates, 4D datasets cannot be visualized by this library. A custom loader is required to visualize 4D datasets.

Future Directions:

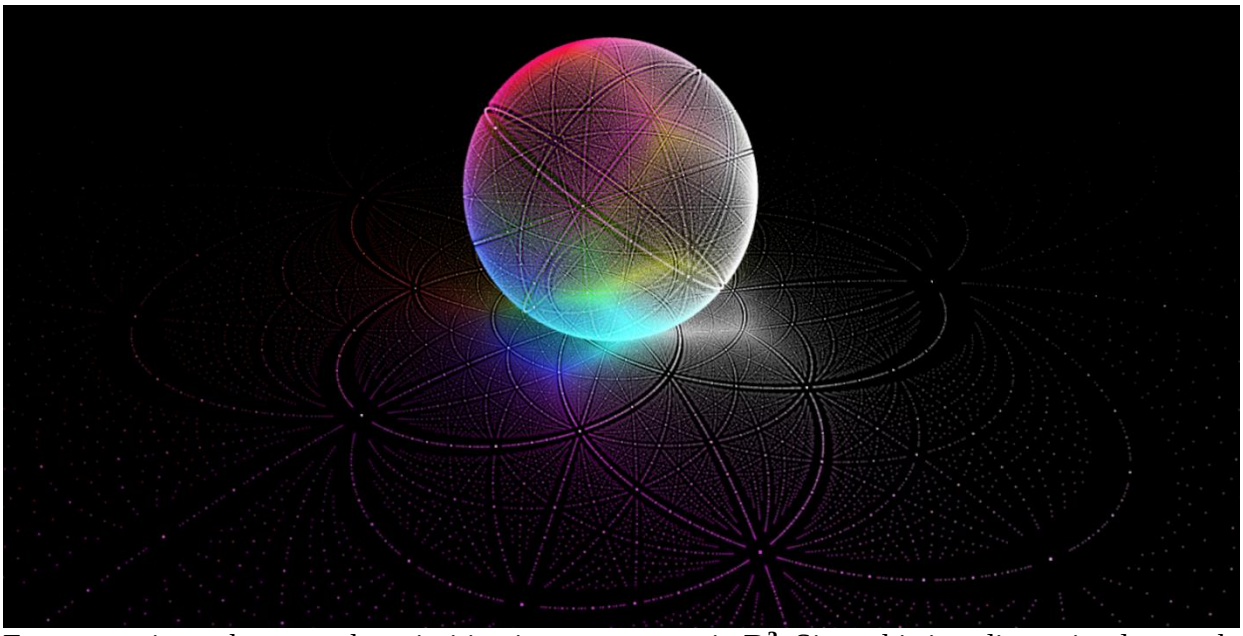
The next step, in the Unity program, would be to allow the user to select a point in the simple curve data set and have a representative of the curve show up in a heads-up display. We would also like to port the program to other platforms and to use it for other 4-dimensional lattices. For the WebGL program, the next step would be to build a custom loader to load the 4D datasets and to use the camera controls of Potree (if possible) to navigate through the dataset.

Screenshots (Unity):

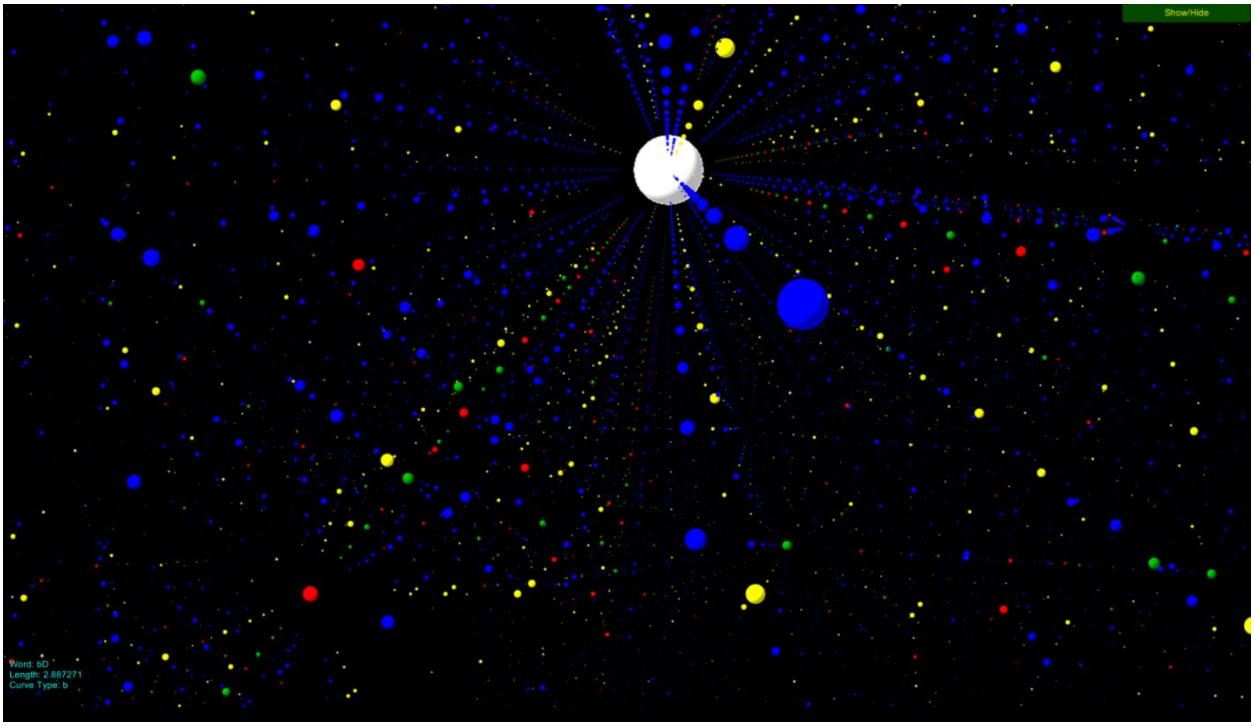
The following are the screenshots of the datasets run through the Unity program.



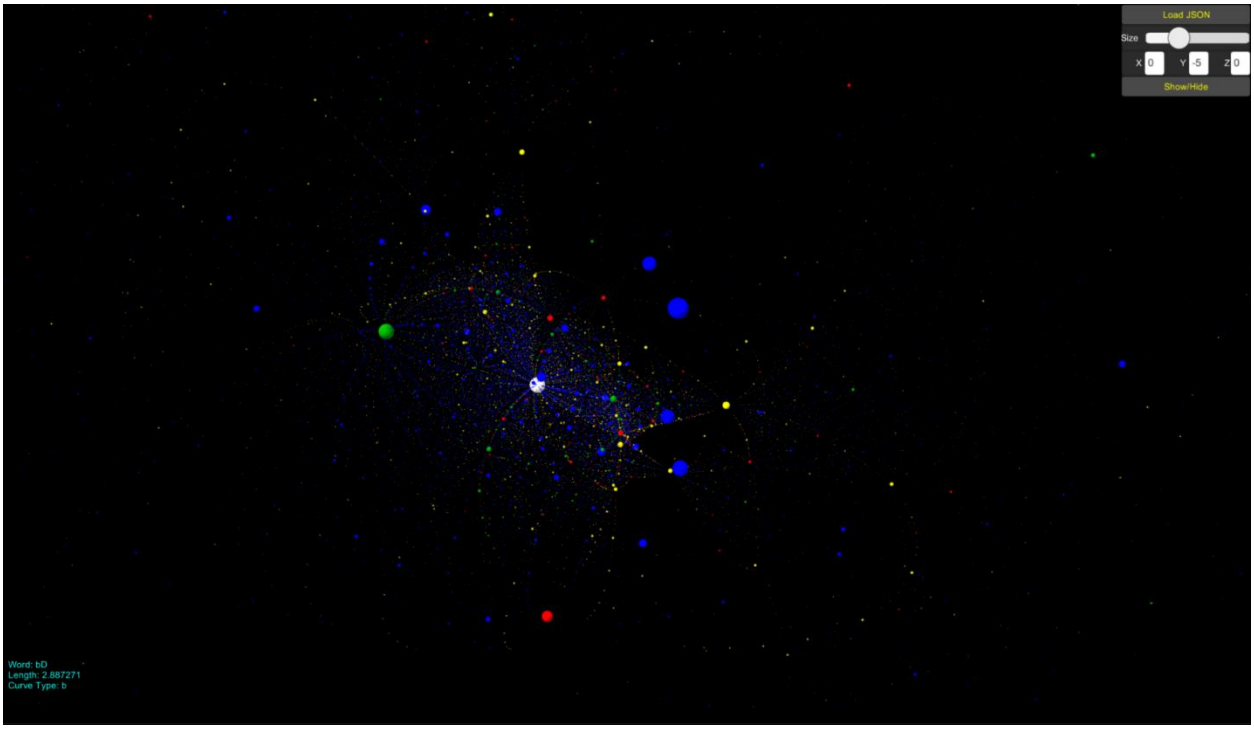
These are the primitive integer vectors in $\mathbf{R}^4$, normalized and seen in stereographic projection using our program.



For comparison, these are the primitive integer vectors in $\mathbf{R}^3$. Since this is a dimension lower, the projection process and the actual sphere can be visualized.



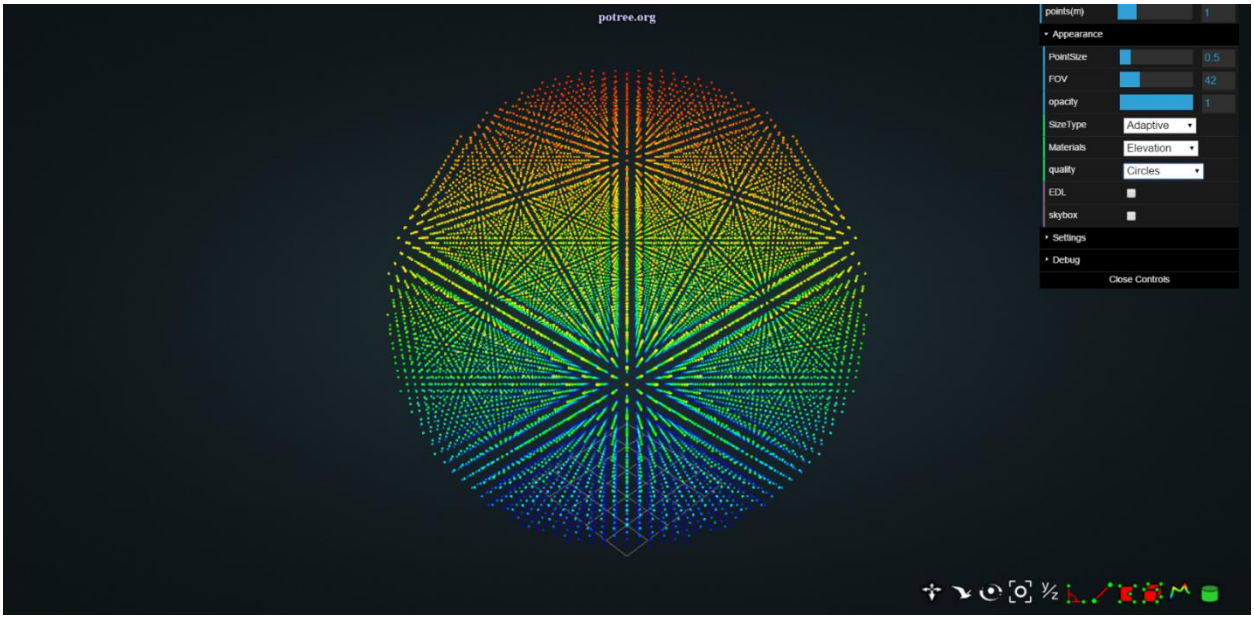
Immersive view of the N(2,2) dataset in first perspective. White point is the point selected.



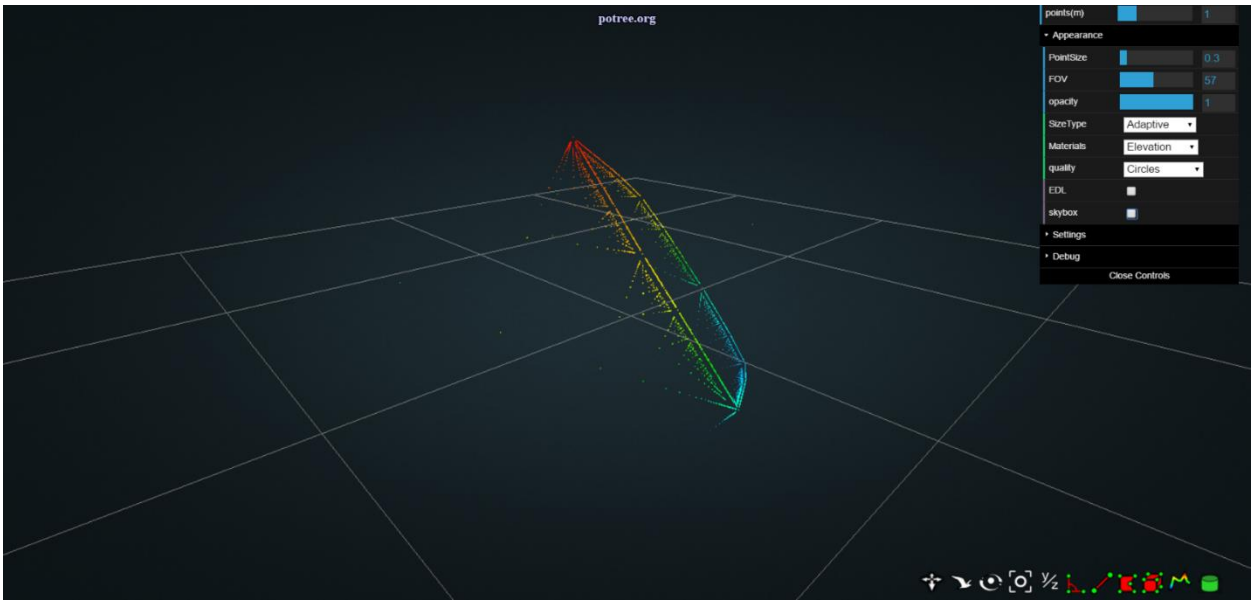
Third person perspective of the N(2,2) dataset. This is the same set as above but from a different perspective, white point is the point selected.

Screenshots (WebGL):

The following are the screenshots of the 3D datasets run through the WebGL program.



This is a visualization of the dataset Z^3, the primitive integer vectors in $\mathbf{R^3}$.



This is a visualization of the dataset $PML(N_3)$ which is the projective measured lamination of the sphere with three cross caps. This dataset is built on an earlier work of D. Dumas and F. Guéritaud [3].

References

[1] A. George https://github.com/angela107/IVS3

[2] A. Gilbert https://github.com/Alex-Gilbert/IVS3

[3] D. Dumas and F. Guéritaud, The PML Visualization Project. Accessed: 2015-09-25. http://dumas.io/PML/

[4] Schutz, M. Potree. http://potree.org. Accessed: 2015-11-28

[5] W. Thurston, Minimal stretch maps between hyperbolic surfaces. Preprint, 1986. arXiv:math/9801039