Computer Visualizations on the Hyperbolic Plane

Yiheng Su

December 2022

1 Introduction

I played a game called HyperRogue a few years ago. The characters in the game lived in a strange space other than the Euclidean world. I explored many different maps in this game, each with its own unique treasures, enemies, and terrain obstacles, and they are all in strange spaces! After taking the geometry class this semester, I realized that HyperRogue was taking place on the hyperbolic plane, in the Poincaré disk model. The strange curves and polygons are the geodesics and polygons in the disk model of the hyperbolic plane. When I searched HyperRogue, I found that the designer of the game, Zeno Rogue, actually created a website[1] for HyperRogue. We not only can play a free version of the game on the website but also can learn how HyperRogue was implemented and the mathematical knowledge behind the game on the website.

Inspired by HyperRogue, I am wondering if there are other games designed in non-Euclidean space since most games I have played take place in either 2D or 3D Euclidean spaces. Then, I found Hyperbolica which is a hyperbolic "walking simulator" with polished graphics and some mini-games in hyperbolic and spherical geometry, and Hypermine which makes a Minecraft-like game that is playable in hyperbolic geometry.

[1] Here is the link to the website: https://www.roguetemple.com/z/hyper/dev.php

Can I design my own computer games or visualizations in hyperbolic geometry? That is the question I asked myself after finding so many interesting games in non-Euclidean space. Then, I found an amazing JavaScript library, Hyperbolic Canvas[2], made by Nick Barry and released under an MIT license. It uses the Poincaré disk model of the hyperbolic plane to map the hyperbolic space onto an HTML canvas. Since I learned a little bit of JavaScript during the summer, I decided to use this library to visualize some surfaces in the Hyperbolic plane. In this article, I will show how I designed my computer visualization to show some surfaces and geodesics in the disk model of the hyperbolic plane.

2 Background Scene: Glued Octagon!

Hyperbolic Canvas is a JavaScript library which uses the Poincaré disk model of the hyperbolic plane to map hyperbolic space onto an HTML canvas. I used functions in this library to build a Poincaré disk model, draw Hyperbolic geodesics, circles and polygons, and computer Hyperbolic angles in the disk model. Since I did not find the documentation for this library, I looked at some example demos built on this library and learned what each function does from those demo codes.

In my computer visualization, I defined a Poincaré disk first and drew a black unit circle on the canvas to represent the boundary of my disk model. Then, I drew a hyperbolic octagon at the center of the canvas using the function:

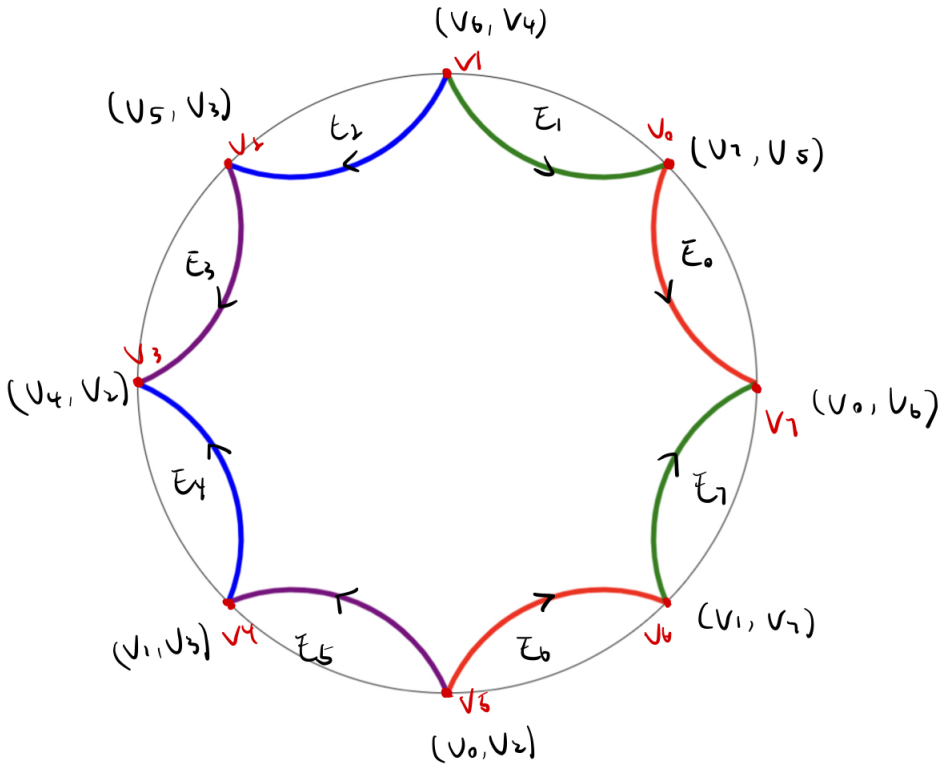
```
HyperbolicCanvas.Polygon.
givenHyperbolicNCenterRadius(8, HyperbolicCanvas.Point.CENTER, RADIUS)
```

Each side of the octagon is a geodesic in the hyperbolic plane with the same length.

To make the surface more interesting, I define some gluing rules here. Firstly, I label my octagon's edges as $E_0, E_2, ..., E_7$ and group these edges into pairs s.t. $\{E_0, E_6\}$, $\{E_1, E_7\}$, $\{E_2, E_4\}$, and $\{E_3, E_5\}$. For each pair, we design an isometry $\varphi_i : E_i \to E_j$.[3] This means that φ_i sends each line segment in the edge E_i to a line segment of the same length in E_j. Also, φ_i matching orientations on E_i and E_j. In this way, every E_i is glued to E_j by an isometry φ_i and E_i, E_j have the same orientation. Here is a diagram showing my glued octagon. I glued the red, green, blue, and purple sides separately. And, the arrow shows the orientation of each side. We can see that the glued sides have the same orientation. Next, let's focus on the vertices of the glued octagon, O.

[2]Here is a website for this library: https://itsnickbarry.github.io/hyperbolic-canvas/about.html.

[3]Here E_i and E_j are the edges in the same group we defined above.



Claim that all eight vertices are glued together. I label the vertices from V_0 to V_7. The bracket near each vertex contains the vertices each vertex is glued to. For example, V_0 is glued to V_7 and V_5, since $\varphi_0(V_0) = V_5$ and $\varphi_1(V_0) = V_7$. V_0 is glued with V_5 and we write it as $V_0 \to V_5$. Notice that

$$V_0 \to V_5 \to V_2 \to V_3 \to V_4 \to V_1 \to V_6 \to V_7 \to V_0.$$

Thus, all vertices are glued together. Thus, there is only one vertex in the quotient space O.

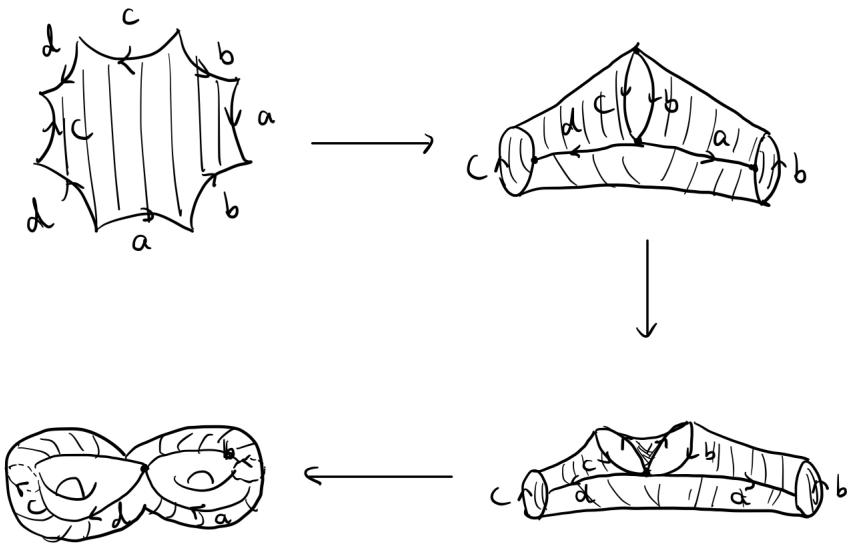
It follows that the quotient space, glued octagon, is an orientable surface. Since we paired up edges, so there are four edges in O. Then, the Euler characteristic is

$$\chi = V - E + F = 1 - 4 + 1 = -2,$$

where V, E, and F are respectively the numbers of vertices, edges, and faces in O. The genus of O is

$$g = \frac{2 - \chi}{2} = \frac{2 - (-2)}{2} = 2.$$

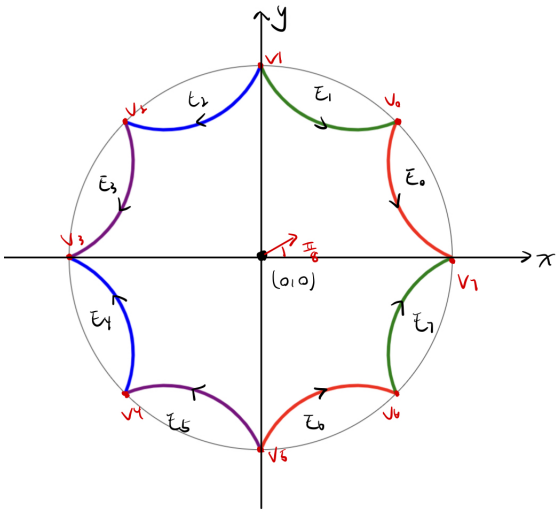
Therefore, O is a surface of genus 2! Here is a visualization from O to a genus two surface.



3 Adventurer in the Hyperbolic Space!

We have set up an Poincaré disk model and a glued hyperbolic octagon O inside it. It is time for us to create an adventurer to explore this hyperbolic world. I created a black ball as our adventurer to move around inside the disk.

Firstly, we need to set up a coordinate for our O. The hyperbolic canvas provides me with a Cartesian coordinate. It is defined below:



The ball object has two fields: position and direction. The position field records the current

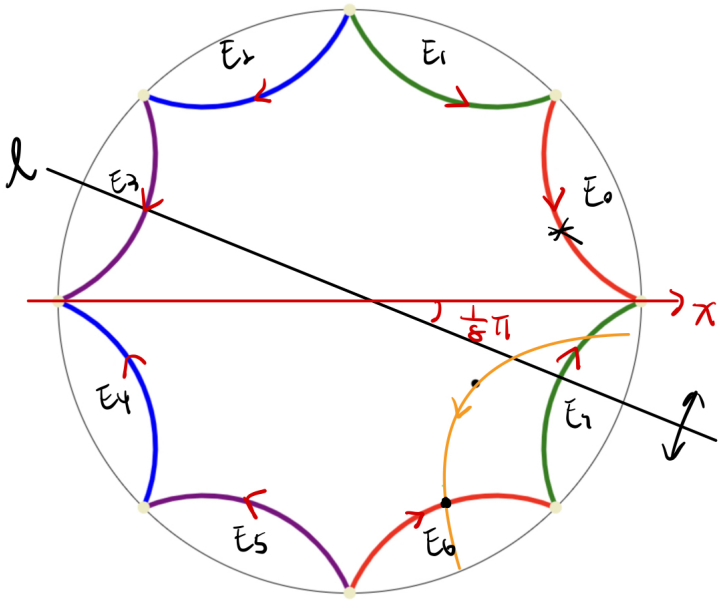
position of the ball on the plane (in the Cartesian coordinate) and the direction field records what direction the ball moves toward. We initialize the ball at $(0, 0)$ and set its angle to $\frac{\pi}{8}$. Thus, the ball will move toward the midpoint of E_0 at first. The ball will move along geodesics in the Poincaré disk model. If the ball hits E_i of O, it will reappear from E_i's corresponding glued edge E_j. I also designed left and right turning functions for the ball. Clicking left and right on the keyboard will change the direction of the ball $\frac{pi}{2}$ in the hyperbolic plane. Now, let's talk about more details on programming the reappearing movement.

3.1 Compute Position After Reappearing

We need to figure out the reappearing position after the ball hits one edge E_i of O. Look at one example below. The ball is moving along the orange geodesic g and it will hit E_6 if we do not change its direction. We drew a black dot p on E_6 to indicate the intersection point between g and E_6. Let's find the line l on O. Notice that the reflection of p across l is the reappearing position of the ball. We need to figure that the equation for l. Since l is passing through the original point, we only need to know the slope of l. Notice that l also passes through the midpoints of E_3 and E_7, so we know the angle between l and x axis is $\frac{1}{8}\pi$. Therefore, $y = sin(-\frac{1}{8})x$. In general the reflection of the point (x, y) across the line $ay + bx = 0$ is

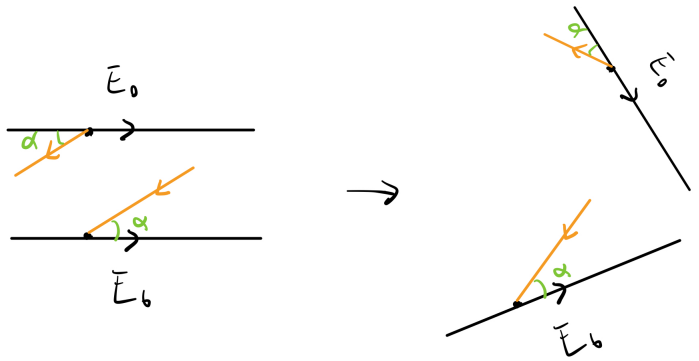
$$(\frac{x(a^2 - b^2) - 2aby}{a^2 + b^2}, \frac{y(b^2 - a^2) - 2abx}{a^2 + b^2}).$$

Thus, we can easily find the reflection point of p across l on E_0 by this formula.

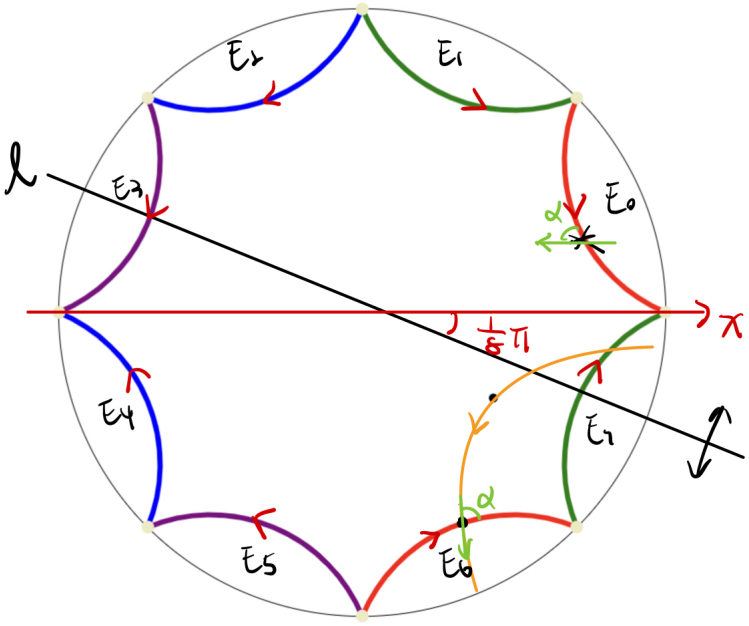


3.2 Compute the Geodesic that the Ball Moves Along After Reappearing

Now, we know the reappearing position after the ball hits one edge E_i. Next, we need to figure out the new geodesics[4] the ball moves along. Notice that we showed that one geodesic in the quotient space X looks below. If we rotate two sides, then we will have something similar to E_0 and E_6.



Our algorithm is when the ball hits E_6 on point p, we compute the angle α between g and E_6 using the euclideanAngleAt function in Hyperbolic Canvas. Based on the angle and reappearing position p', we can compute a new geodesic g' which intersects with E_0 on p' and the angle between g' and E_0 is α. Then, we update the direction of the ball based on the new geodesic g'.

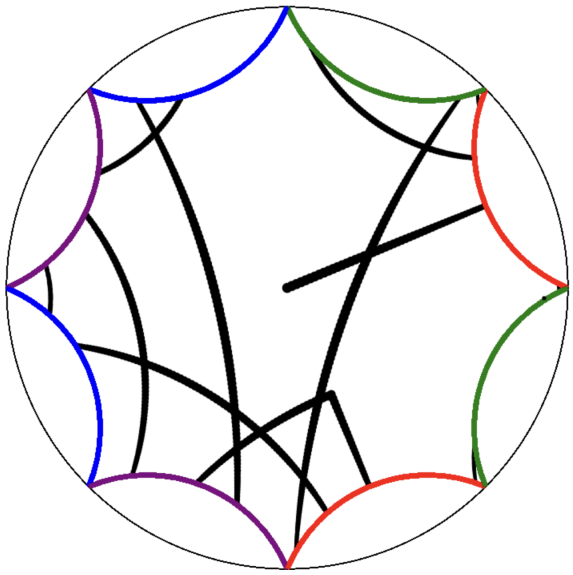


[4]In the quotient space O, the ball is always moving on the same geodesic in O if we do not change the direction of the ball. Here, we split it into different geodesics because it is easier for computation.

4 Possible Extensions on the Existing Programs

4.1 Leave the Tail of the Ball on the Canvas

One easy extension is that we can leave the tail of the ball on the canvas. Let's talk about how canvas shows the movement of the ball. In the main loop, we call myCanvas.clear(), drawSurface(), and move(ball) these three functions. myCanvas.clear() can clean the canvas. drawSurface() function can repaint the background scene. move(ball) can compute the new position of the ball and repaint it on the canvas. Since the ball updated its position during the move function, the ball will move to the new position in the canvas after repainting. When the ball is moving on the canvas, we do not erase the previous path it took. Then, we can see that the ball is moving along geodesics and how the reappearing algorithm works. Thus, in our main loop, we comment out the myCanvas.clear(). Then, the canvas will leave the path of the ball on the screen. Here is a screenshot showing the path of the ball.



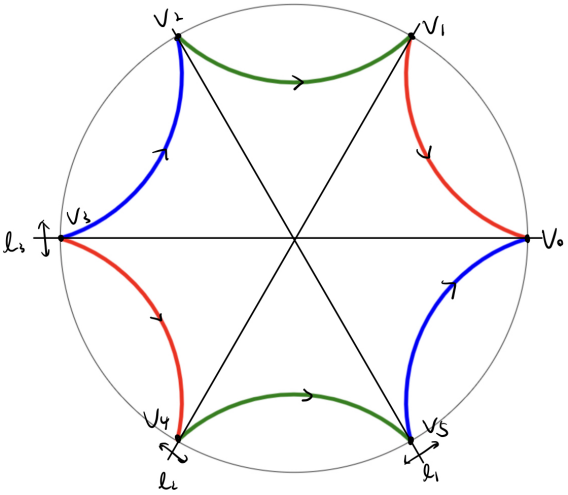
4.2 Hyperbolic Polygons with even number of sides.

In my program, I chose to background scene to be an octagon. I can also make a hyperbolic hexagon or any polygons with an even number of sides. Then, we can glue the opposite sides of the polygon together. Suppose we want to change the background scene to an octagon. The following are the things I need to modify:

1. In my program, I hard-coded the gluing rule. I labeled the edges of the octagon from index 0 to 7. I defined a gluing_rules() function whose input is the index of an edge and output returns the index of the corresponding gluing side. For example, we glue E_0 with E_6, so gluing_rules(0) returns 6 and gluing_rules(6) returns 0. Thus, if we have a hyperbolic hexagon

or any polygons with an even number of sides, we need to redefine the gluing_rules() function.

2. In my program, I also hard-coded the reflection lines at the beginning of the program. The reflection line can help me figure out the position of the ball after reappearing. Now, if we have a hexagon, we need to rewrite a function to find the new reflection lines. For a hexagon, we can use the vertices to find the equations for the reflection lines. The reflection lines are shown below:



4.3 Glue the Octagon in a different way!

In my program, each pair of glued sides have the same orientation. We can also create a surface that is not orientable by gluing the octagon in the way below. In this case, E_0 is glued in the opposite direction as E_7. Thus, we will have a locally Mobius strip in our octagon.

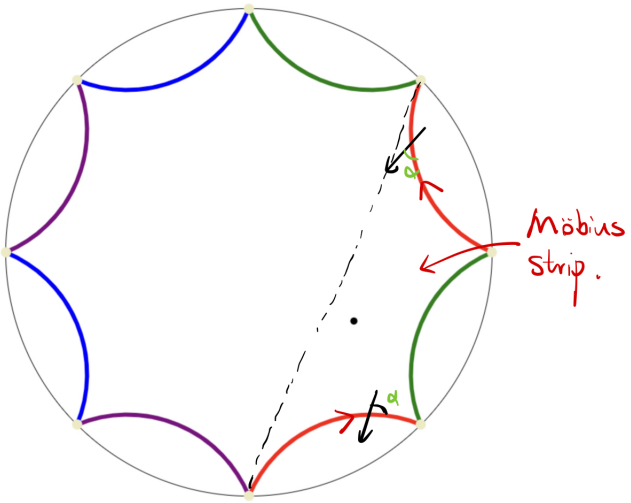
In this case, we cannot use the reappearing algorithm for the octagon. We cannot use the reflection lines to compute the reappearing position of the ball, since they are not symmetric across the reflection lines anymore. I have not come up with a good algorithm to find the position of the ball after reappearing. This could be great future work!

5 Usage of Disk Model Program

The disk_model folder includes five files. To run the program, please click the "disk_model.html" or "disk_model_trace.html".

1. "hyperbolic_canvas.js" file is the hyperbolic canvas library with the Copyright (c) 2015 Nick Barry.

2. "ball.js" file contains my programs for this project.



3. "disk_model.html" file creates an HTML canvas to visualize the ball and hyperbolic space in a disk model.

4. "ball_trace.js" file is an extension on "ball.js". The ball leaves a trace on the canvas when moving around.

5. "disk_model_trace.html" file creates an HTML canvas to visualize the ball and hyperbolic space in a disk model. The ball leaves a trace on the canvas when moving around.

6 Acknowledgement

1. https://zenorogue.medium.com/non-euclidean-geometry-and-games-fb46989320d4

2. https://itsnickbarry.github.io/hyperbolic-canvas/about.html

3. https://h2snake.netlify.app/

4. https://www.roguetemple.com/z/hyper/dev.php

5. https://math.stackexchange.com/questions/479371/how-to-construct-a-genus-2-surface-from-8-gon

At last, great thanks to Professor Scott Taylor who helped me build an outline for this project.