**labyrinth generator**

**function of the app**

This game allows the user to generate a verity of labyrinths and explore the shortest distance it would take to reach the end point. The users can compare the distances and see their changes to the parent shape can effect the labyrinth. If they wish to further explore they can even download the generated geometry

**Features of the app**  
Geometry of the labyrinth can be controlled through parameters which are as follows

**Radius**   
 the radius of the parent shape(sphere) that affects the surface on which the maze is mapped. Radius ranged between 20 and 70

**U-count** u-count represents the number of divisions of the mesh on its x axis. U count range is between 1 and 30

**V-count** V-count represents the number of divisions of the mesh on its y axis. V count range is between 1 and 30

**Time**   
 Perlin noise as described by Ken Perlin is used to distort the sphere. Time ranged between 0 and 10.

**Scale**   
 Perlin noise as described by Ken Perlin is used to distort the sphere. scale ranged between 0 and 10 which translate to 0 and 0.06.

**Seed**   
controls the randomness within the algorithm effects how the maze is generated. Seed ranges between 0 and 100.

**Start**   
 this is the starting point that is used to connect with the end point on the maze through the shortest path as expressed in the app. Points ranges from 0 and 3000 but also depends on u and v counts for total number of points present in the parent shape.

**Tube**thickness of the shortest path as selected via the start point and a fixed end point. Tube thickness ranges from 0 to 10.   
  
**Length** This variable provides two outputs generated from your choices which are the total length of all the paths within the labyrinth and the distance of the path you have picked via the start point variable.   
  
**download**   
This output allows the user to download the chosen geometry in a 3dm format

**contents of repo**

* here is the repo that contains files needed to run the labyrinth generator as a web server
* files that have been added and are useful for the generator are in src\examples\perlin-noise and \src\files

**Location of the generator within the repo**

Perlin-noise folder contains

1. the html code
2. css code
3. js code
4. perlin\_maze.gh ( found in src\files)

* these are important to maintain and functioning of web app.
* html handles the web interface css handles the aesthetics and js makes sure grasshopper files can connect with aws rhino compute server and Heroku and .gh hands the algorithm that generates the geometry