Sami Emre Erdogan
Student ID: 2019280513

Computer Graphics Star Rotation

In my assignment I have chosen to do the Archimedean Spiral.

```
class star
{
public:
    GLfloat angle, a, b, x, y;
    glm::vec4 color;
};
```

```
// put stars in the array
const int num = 100;
const float pi = 3.1415;
star stars[num];

// distance to the stars
GLfloat zoom = -25.0f;
// initial angle
GLfloat def_a, def_b = 0.5;
GLint iter = 0;
GLfloat max_distance = 5.0;
```

First I have created a class called star to declare variable to be used in our logarithmic spiral formula.

```
// Set the object data (buffers, vertex attributes)
GLfloat starVertices[] = {
    // Positions          // Texture Coords
    0.0f,  0.5f,  0.0f,  0.0f,  1.0f,
    1.0f, -0.5f,  0.0f,  1.0f,  0.0f,
    1.0f,  0.5f,  0.0f,  1.0f,  1.0f,

    0.0f,  0.5f,  0.0f,  0.0f,  1.0f,
    0.0f, -0.5f,  0.0f,  0.0f,  0.0f,
    1.0f, -0.5f,  0.0f,  1.0f,  0.0f,
};
```

Set the star vertices for our star.bmp file

```
glm::vec4 random_colors[] = {
    glm::vec4( _x: 160.0 / 255,  _y: 0,  _z: 200.0 / 255,  _w: 1.0f),
    glm::vec4( _x: 110.0 / 255,  _y: 0,  _z: 220.0 / 255,  _w: 1.0f),
    glm::vec4( _x: 30.0 / 255,  _y: 60.0 / 255,  _z: 1.0f,  _w: 1.0f),
    glm::vec4( _x: 0,  _y: 160.0 / 255,  _z: 1.0f,  _w: 1.0f),
    glm::vec4( _x: 0,  _y: 200.0 / 255,  _z: 200.0 / 255,  _w: 1.0f),
    glm::vec4( _x: 0,  _y: 210.0 / 255,  _z: 140.0 / 255,  _w: 1.0f),
    glm::vec4( _x: 0,  _y: 220.0 / 255,  _z: 0,  _w: 1.0f),
    glm::vec4( _x: 160.0 / 255,  _y: 230.0 / 255,  _z: 50.0 / 255,  _w: 1.0f),
};
```

Get random colours from our stars. I couldn't figure how to use a randomizer function that would do it for me. I have seen examples codes which randomized r g b but I was unable to implement it somehow so I manually added some colours which isn't that efficient.

Sami Emre Erdogan
Student ID: 2019280513

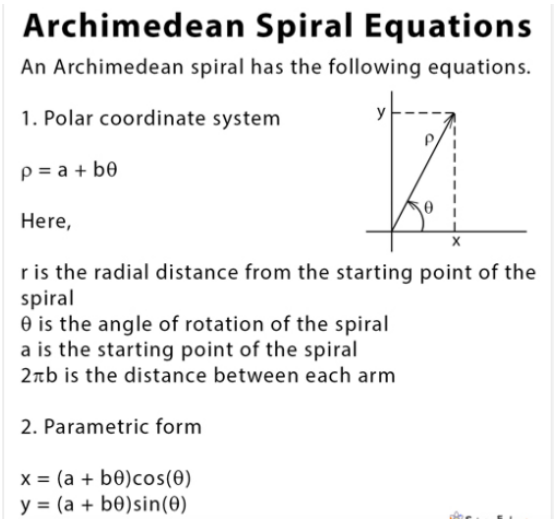
```
for (int j = 0; j < num; j++) {
    stars[j].color = random_colors[j % 10];
    stars[j].angle = pi / 10 * j;
    stars[j].a = def_a * (1 + pi / 10) / (1 + pi / 10 * (j));
    stars[j].b = def_b * (1 + pi / 10) / (1 + pi / 10 * (j));
    stars[j].x = (def_a + def_b * stars[j].angle) * cos(stars[j].angle);
    stars[j].y = (def_a + def_b * stars[j].angle) * sin(stars[j].angle);
```

```
for (int i = 0; i < num; i++) {
    glBindVertexArray(starVAO);
    glBindTexture(GL_TEXTURE_2D, starTexture);
    model = glm::mat4( s: 1);
    GLfloat distance = glm::sqrt( lcpp_x: stars[i].x * stars[i].x + stars[i].y * stars[i].y);

    stars[i].x *= 1 + pi / 1000 / distance;
    stars[i].y *= 1 + pi / 1000 / distance;

    model = glm::translate(model, v: glm::vec3(stars[i].x, stars[i].y, _z: zoom + i * 0.001));
    if (distance >= max_distance) {
        iter = (iter + 1) % num;
        stars[i].a = def_a * (1 + pi / 10) / (1 + pi / 10 * (iter));
        stars[i].b = def_b * (1 + pi / 10) / (1 + pi / 10 * (iter));
        stars[i].angle = pi / 10 * iter;
        stars[i].x = (stars[i].a + stars[i].b * stars[i].angle) * cos(stars[i].angle);
        stars[i].y = (stars[i].a + stars[i].b * stars[i].angle) * sin(stars[i].angle);
    }
    stars[i].color.w = 1 - distance / max_distance;
```

I have set the maximum number of stars to a 100 so the program will run until it computes 100 stars.

When I ran the code, it was working fine but since I am using a MacBook Air. The GPU isn't good enough to perform the task faster. Also, OpenGL doesn't really run well on MAC OS.

I have refenced form this link to create the Archimedean Spiral Equation it was easy to understand and simple enough to reference it from
https://www.sciencefacts.net/archimedean-spiral.html



Archimedean Spiral Equations

An Archimedean spiral has the following equations.

1. Polar coordinate system

$\rho = a + b\theta$

Here,

r is the radial distance from the starting point of the spiral
θ is the angle of rotation of the spiral
a is the starting point of the spiral
$2\pi b$ is the distance between each arm

2. Parametric form

$x = (a + b\theta)\cos(\theta)$
$y = (a + b\theta)\sin(\theta)$