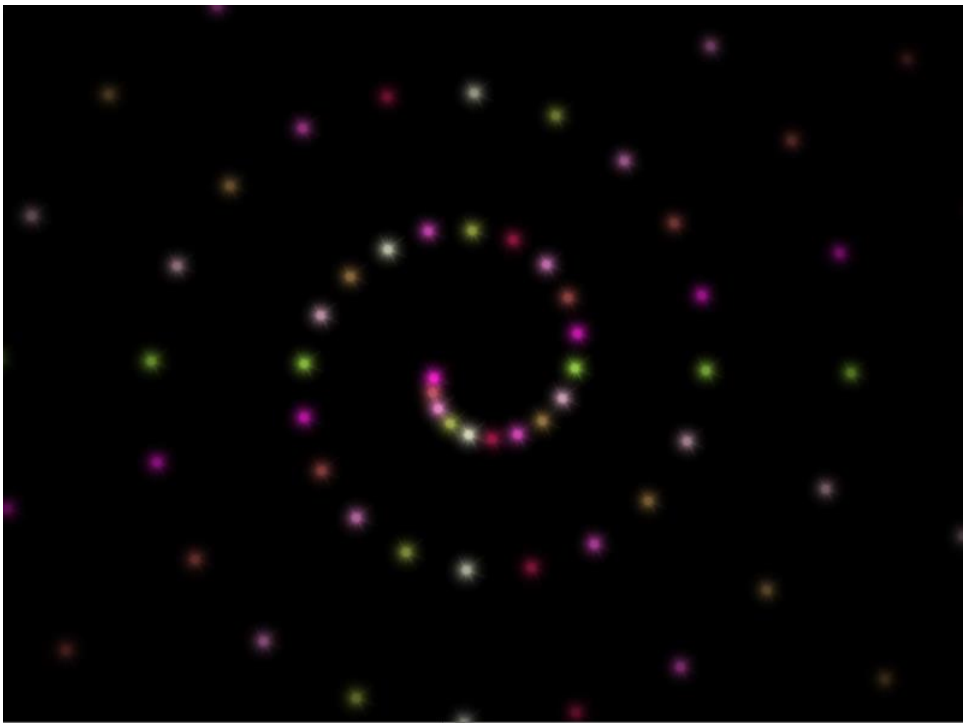
Assignment 2

Fedor Ivachev 2019280373

Star rotation

The idea is to render stars of different colors rotating around the center spreading out in a spiral path.

In the attached video it can be seen that I managed to make the final result look similar to the video of Archimedes spiral provided.



Screenshot of the program.

The code is based on tutorial 6 code.

```
7   uniform sampler2D texture1;
8   uniform vec4 my_color;
9
10  void main()
11  {
12      color = my_color * texture(texture1, TexCoords);
13  }
14
```

Each star has a different color, which is achieved by multiplying color based on the angle on the texture clipped to the object color.

I created an array of star objects. The star class has six fields:

Angle, which is responsible of the position on the spiral

color, which is based on the angle, so stars with same angle have same color,

A, b, which are responsible for the size of the spiral

And x, y, the coordinates of the star

```
for (int loop = 0; loop < num; loop++) {
    stars[loop].angle = M_PI / 10 * loop;
    stars[loop].color = glm::vec4(0.65 + loop % 2 / 4.0,  0.1 + (loop % 5) / 5.0, 0.29 + (loop % 2) / 2.0, 1.0f);
    stars[loop].a = init_a * (1 + M_PI / 10) / (1 + M_PI / 10 * (loop));
    stars[loop].b = init_b * (1 + M_PI / 10) / (1 + M_PI / 10 * (loop));
    stars[loop].x = (init_a + init_b * stars[loop].angle) * cos(stars[loop].angle);
    stars[loop].y = (init_a + init_b * stars[loop].angle) * sin(stars[loop].angle);
}
```

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

    stars[loop].x *= 1 + M_PI / 1000 / stars[loop].distance;
    stars[loop].y *= 1 + M_PI / 1000 / stars[loop].distance;
```

Each of the stars' coordinates are multiplied by 1 + 0.00314 / distance from the center.

The formula is like this to make the stars move away from center with the same speed.

```
model = glm::translate(model, glm::vec3(stars[loop].x, stars[loop].y, zoom + loop * 0.001));\
```

Here the model is translated to the star's coordinates . The star's number loop is multiplied by a small number to make stars lie on different planes. Then the blending function glBlendFunc(GL_SRC_ALPHA, GL_ONE); can be used correctly.

```
if (stars[loop].distance >= max_dist - 0.0001) {
    stars[loop].angle = M_PI / 10 * (iter + 0.05);
    stars[loop].a = init_a * (1 + M_PI / 10) / (1 + M_PI / 10 * (iter));
    stars[loop].b = init_b * (1 + M_PI / 10) / (1 + M_PI / 10 * (iter));
    stars[loop].x = (stars[loop].a + stars[loop].b * stars[loop].angle) * sin(stars[loop].angle);
    stars[loop].y = (stars[loop].a + stars[loop].b * stars[loop].angle) * cos(stars[loop].angle);
    iter = (iter + 1) % num;
}
```

If the distance from center is too big, the star is teleported to the center. Iter is the number of the star teleported. For example, when the first star is teleported, iter is equal to one, etc.

Then the angle is counted. Here I add a small number to iter to make the spiral look smoother.

A,b,x,y values are counted as in the initialization.

A and b values are calculated by building similar triangles before and after the teleportation and taking in account the changed angles.

The alpha value is the smaller, the further the star is from the center.

```
stars[loop].color.w = first_loop ? 0 :  1 - stars[loop].distance / max_dist;
if (first_loop) {
    if (iter > num * 0.9) {
        first_loop = false;
    }
}
GLint color_location = glGetUniformLocation(shader.Program, "my_color");
glUniform4f(color_location, stars[loop].color.x, stars[loop].color.y, stars[loop].color.z, stars[loop].color.w);
glUniformMatrix4fv(glGetUniformLocation(shader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
glDrawArrays(GL_TRIANGLES, 0, 12);
```

During the first loop the stars are not shown because the spirals from the first loop and from the next loops have different shape. Probably I made a mistake when counting the formulas somewhere.