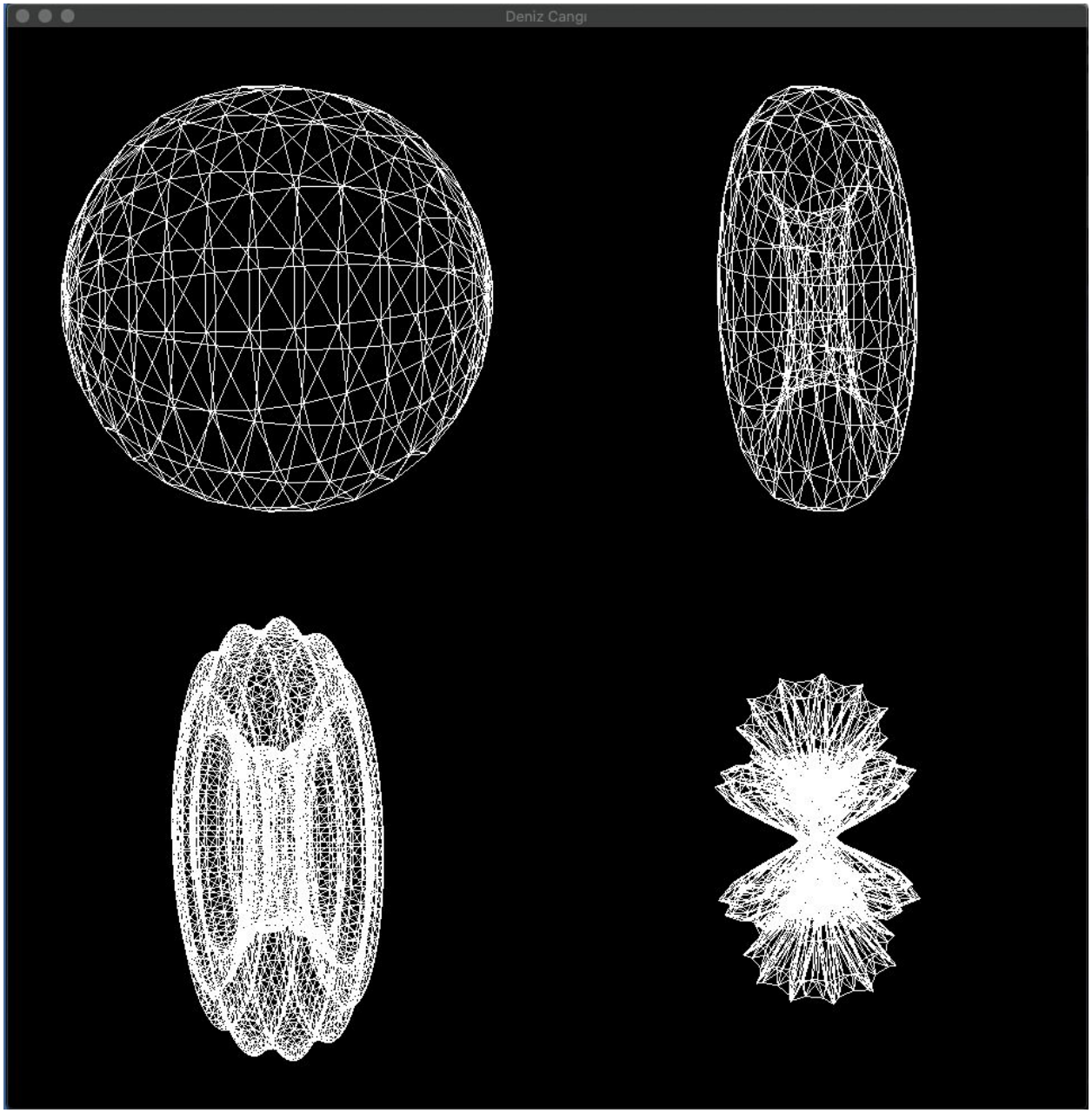
Task 1 :

In the Task1 I have created 4 objects, and used PolygonMode GL_LINE. The first object is sphere that is located at (-0.5, 0.5, 0), I have translated the sphere using transformation. The sphere is rotating around (1,1,0) and it's scale is 0.3. The second object is torus that is located at (0.5, 0.5, 0), I have translated the torus using transformation. The torus is rotating around (1,1,0) and it's scale is 0.3. The third object is spiked torus that is located at (-0.5, -0.5, 0), I have translated the spiked torus using transformation. The spiked torus is rotating around (1,1,0) and it's scale is 0.3. And last object is spiked flower that is located at (0.5, -0.5, 0), I have translated the flower using transformation. The flower is rotating around (1,1,0) and it's scale is 0.3.

For all of them I have get the surface color is white and ambient color as white either. Color is the multiplication of ambient color and surface color.

The screenshot of the scene is:

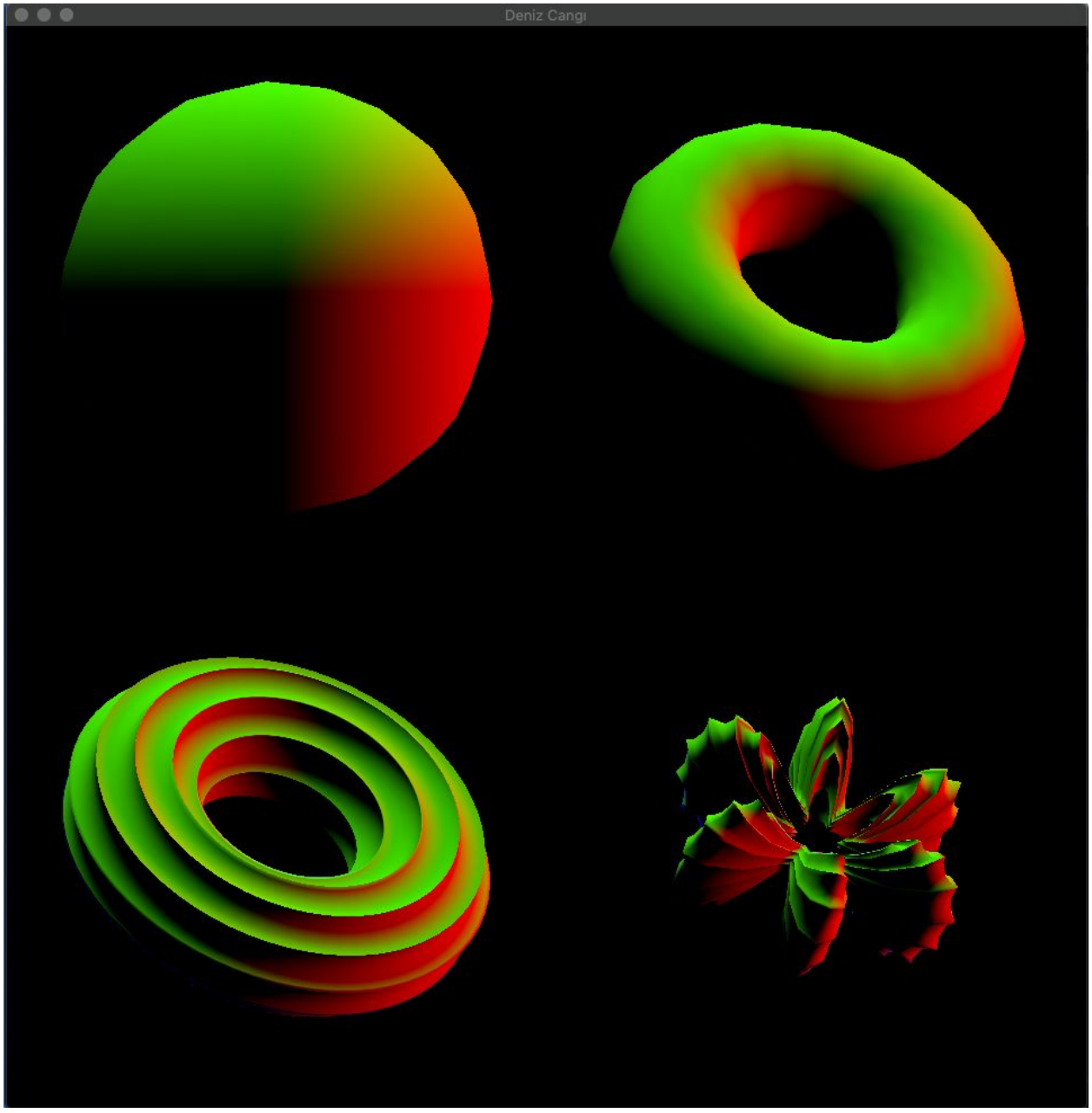


Task 2:

The same objects were created but this time I have used PolygonMode as GL_FILL. The objects, their rotation and position are the same as the Task1.

This time I have used ambient as 2, ambient color is white and color is ambient * ambient color * vertex_normal which is the normal of the vectors. So the color is normalized normal vectors.

The screenshot of the scene is:



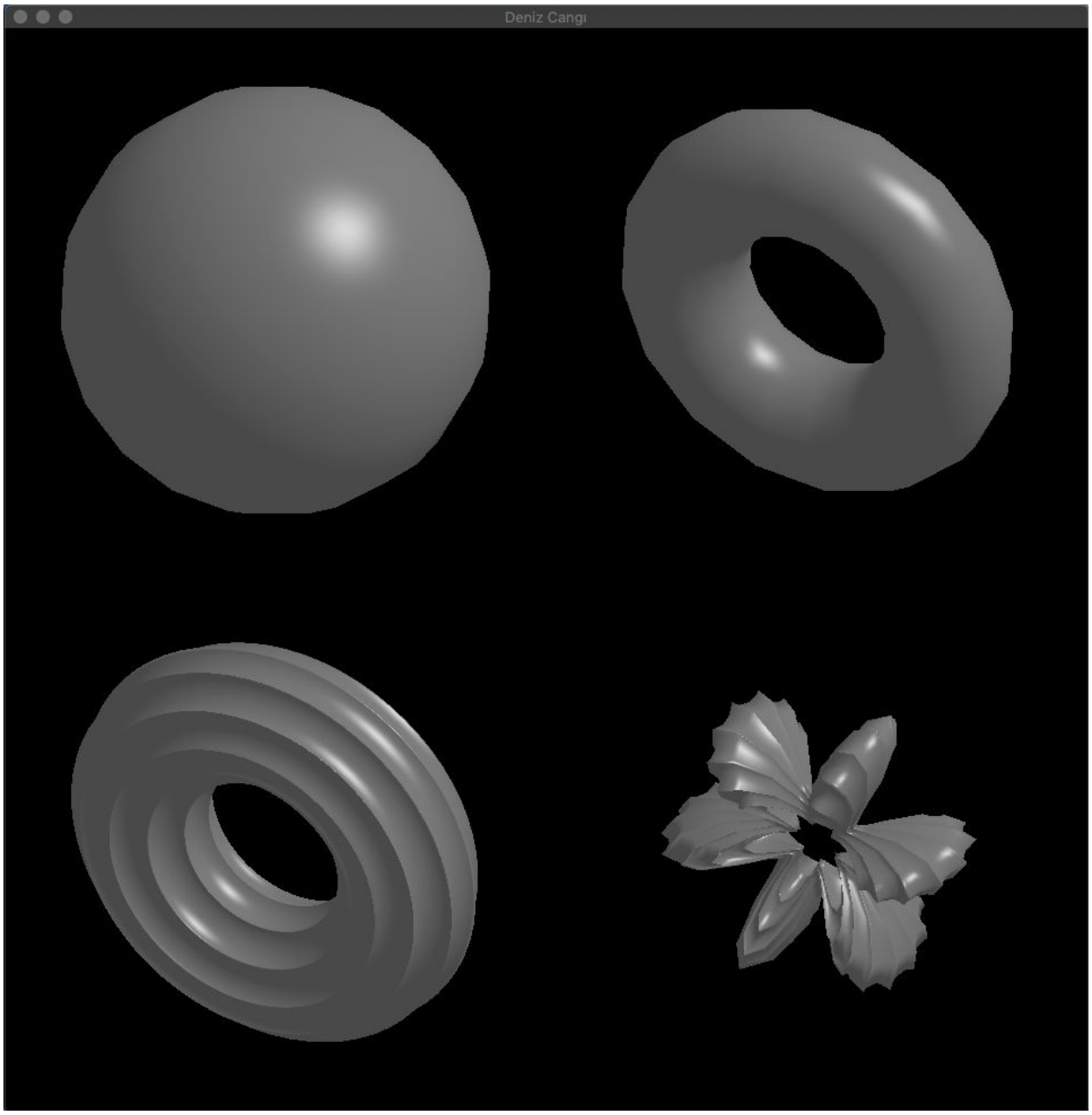
Task 3:

The same objects were created but this time I have used PolygonMode as GL_FILL. The objects, their rotation and position are the same as the Task1 and Task2.

This time I have used surface color as 0.5, ambient color as 0.5 either. Light direction is from -1,-1,1 and light color is 0.4.since we are using Blinn-Phong reflection model, diffuse intensity is

maximum of dot product of light direction and surface normal and 0. View direction is normalize(vec3(vertex_position.xy,-1)-surface_position) which is (0,0,-1). Halfway direction is the halfway vector between view direction and light direction, also since it's direction we should normalize it. Specular intensity is maximum of the dot product of halfway direction and surface normal and 0. For shininess I have used 64. Color is ambient color * surface color + diffuse intensity * light color * surface color + ((specular intensity)^shininess) * light color.

The screenshot of the scene is:

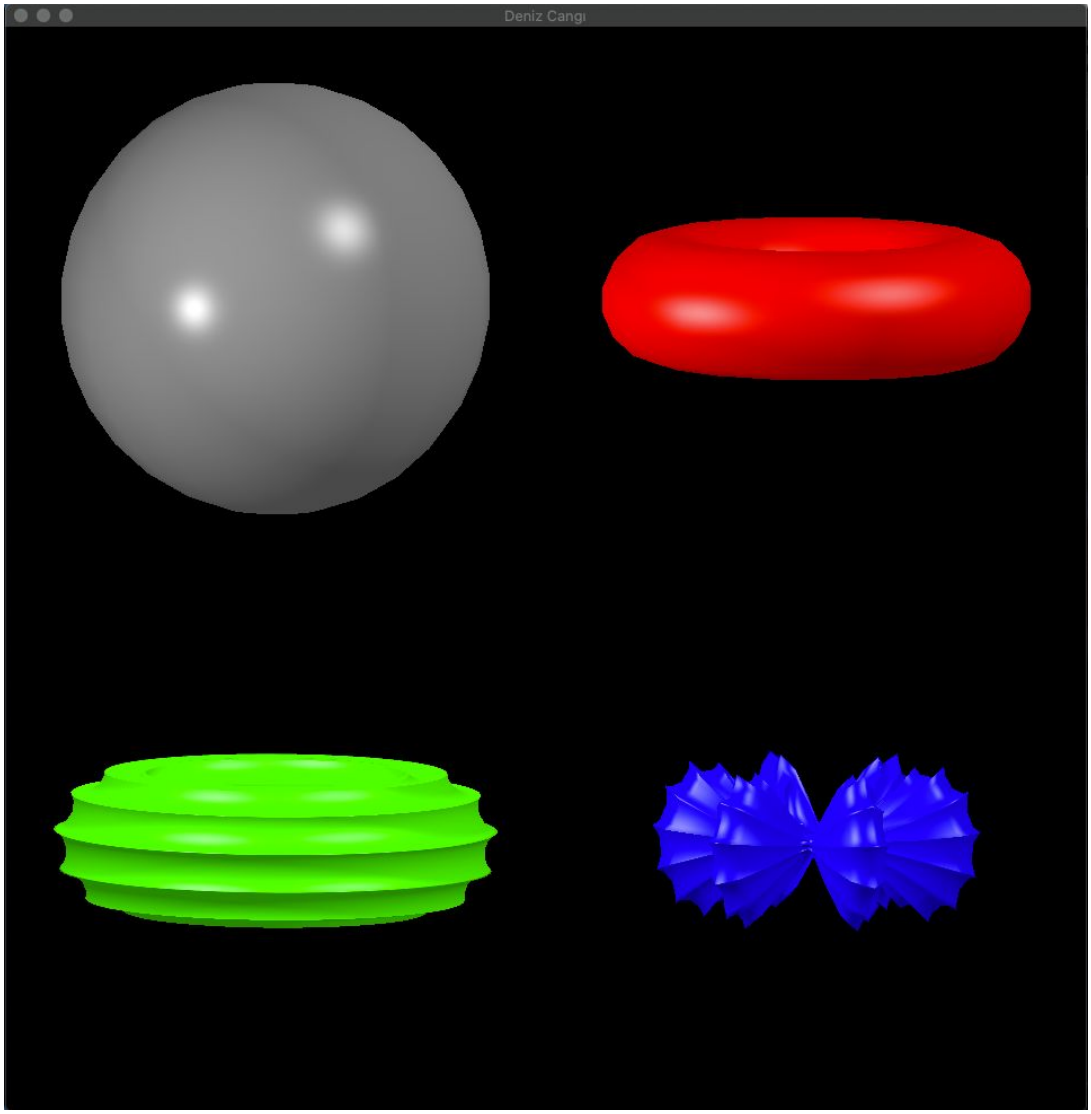


Task 4:

The same objects were created but this time I have used PolygonMode as GL_FILL. The objects, their rotation and position are the same as the Task1, Task2 and Task3.

In the Task 4 I've added a point light that we control its position by the mouse, to create the point light I have changed directional light to point light as I have added to_point_light variable where we find the vector that is from vertex position to point light. To find diffuse intensity this time I have got the dot product of to_point_light vector and surface normal and get the maximum of 0 and result of dot product. In that task the shininess depends on the object so shininess is a uniform that we take when the object created. The rest of the color calculation is the same as described in the Task 3. In that task each mesh has its own surface color that we assign with uniform vector. For the first one I have assigned 0.5,0.5,0.5 for the second I have assigned 1,0,0 for the third I have assigned 0,1,0 for the last one I have assigned 0,0,1. For the shininess for the first one I have assigned 128 and for rest I have assigned 32.

The screenshot of the scene is:



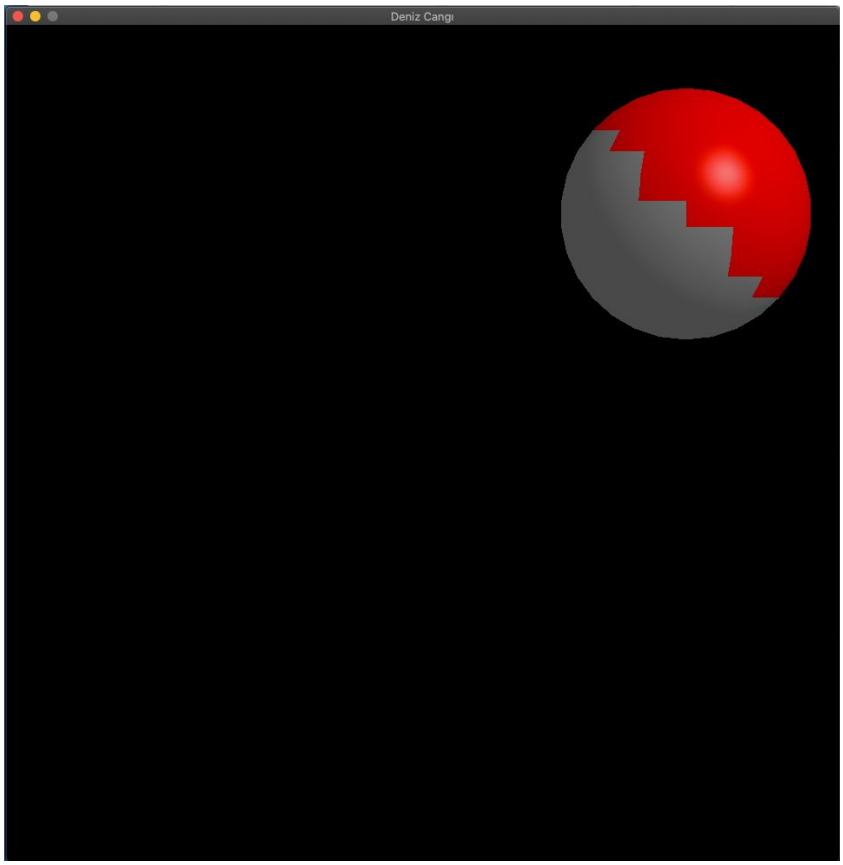
Task 5:

For the game task I have created 2 spheres, one of them's position is controlled by the mouse position and the other's position is controlled by the first one. I have used translate to assign their position. The first one has position glm::vec3(mouse_position, 0) and for the second I have calculate the chasing_pos as given in the documentation:
glm::dvec2 chasing_pos;
chasing_pos = glm::mix(mouse_position, chasing_pos, 0.99);

Then I have calculated the distance between the two spheres by the distance function where I have send the mouse_position which is the position of the first sphere and chasing_pos. If the distance is greater than 0.6 then I have assigned the color green to the first sphere, else ı have assigned the color red. Then I have created the first sphere. The color of the second sphere is gray so I have assigned 0.5,0.5,0.5 and created the second sphere too. I have used the Blinn-Phong reflection model to shade the meshes and used the shininess as 64.

In the demonstration the objects do not go outside of the screen but I'm using a MacBook and I think because of the difference between the OS, my spheres go outside of the screen when I position the mouse too far.

The screenshot of the scene is: Unfortunately I couldn' take screenshot of the scene properly.



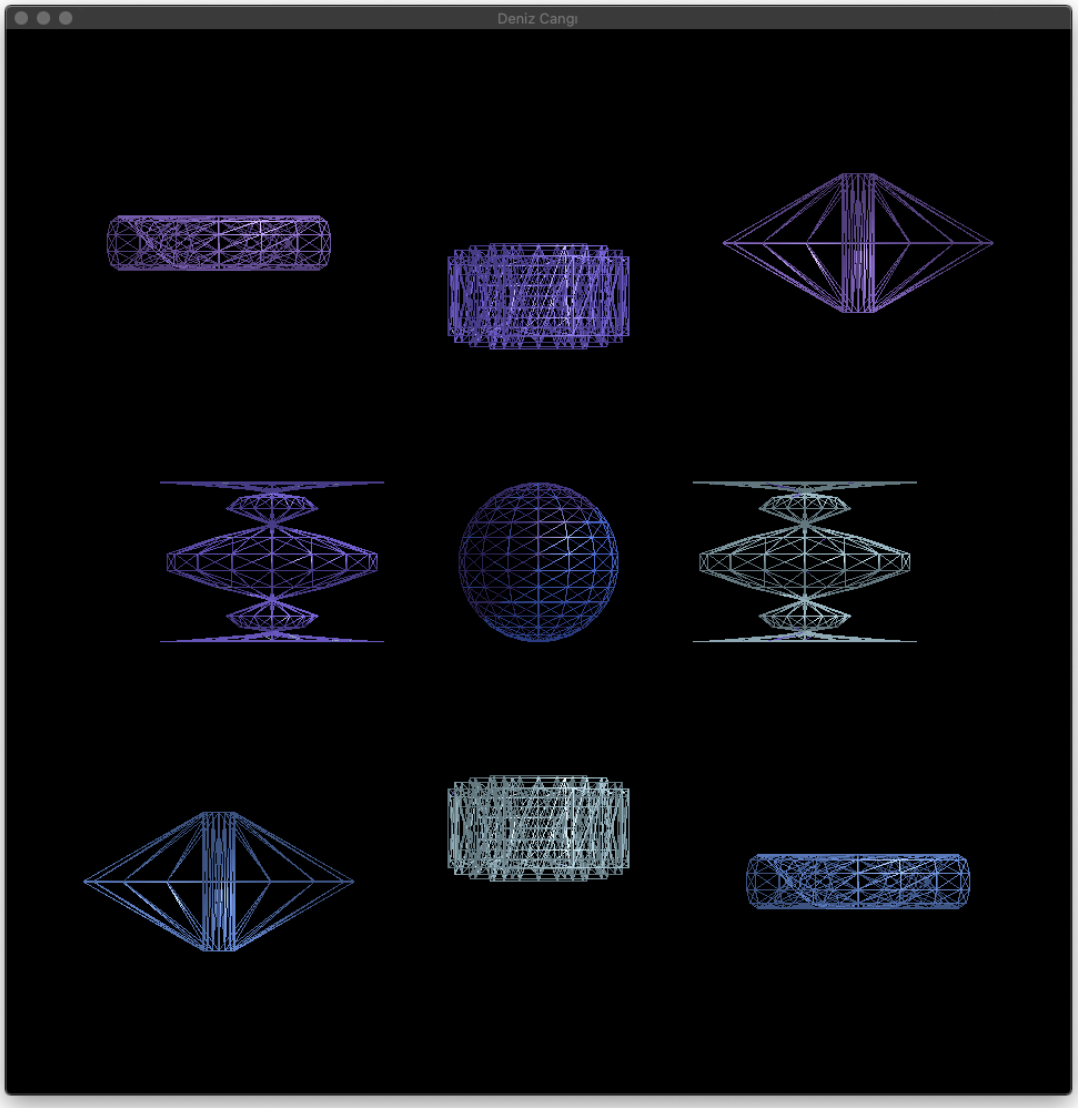
Task 6:

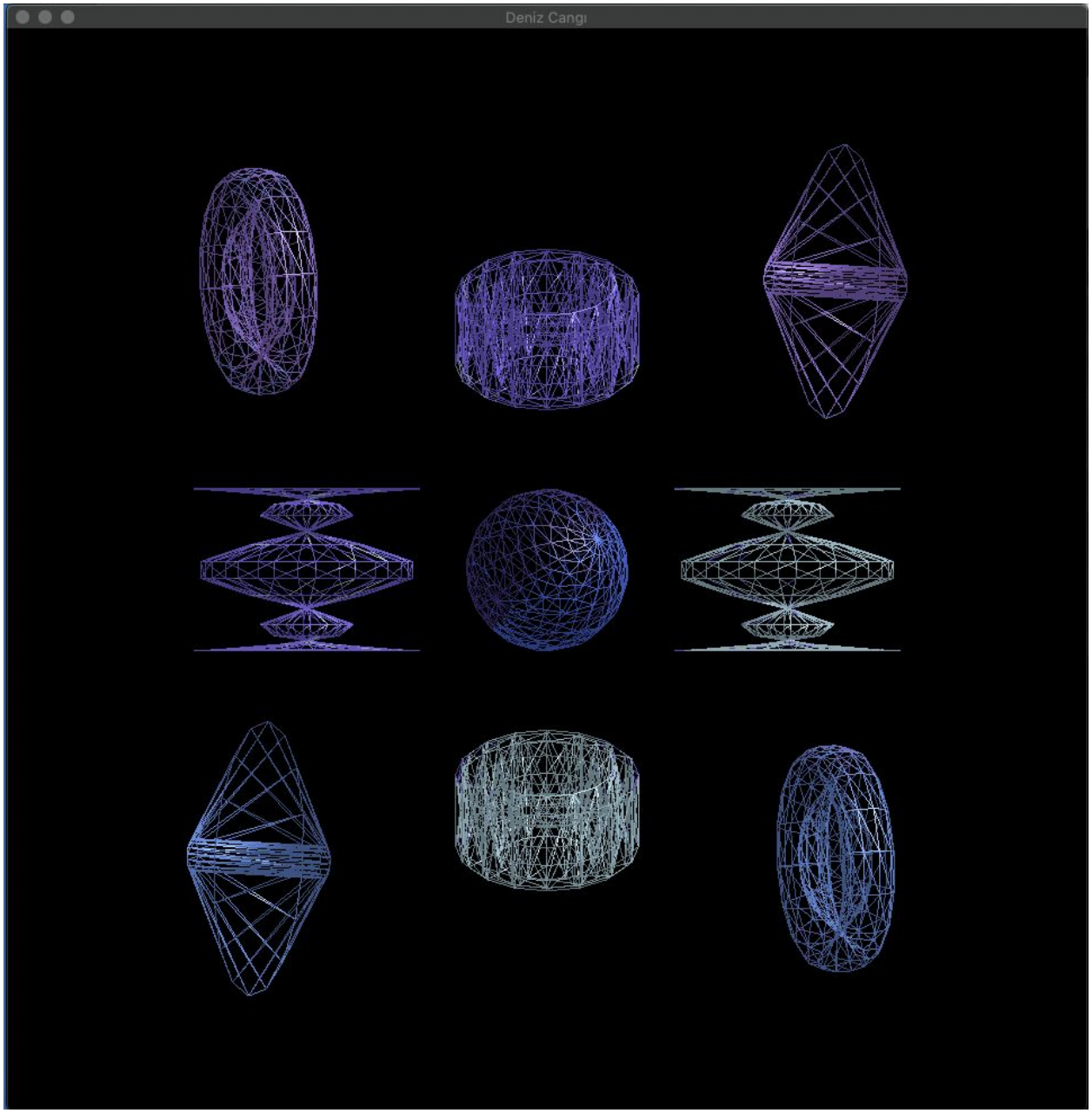
For this task I have created different meshes using the given functions in the meshgeneration.h and meshgeneration.cpp but altered some of them to get different shapes. In this program I have used PolygonMode GL_LINE. First I have created a sphere in the middle which is half blue and half purple. I have done this by creating two different spheres on top of each other with different colors. This sphere is rotating around itself and it acts as a sun because all the other meshes are rotating around this sphere.

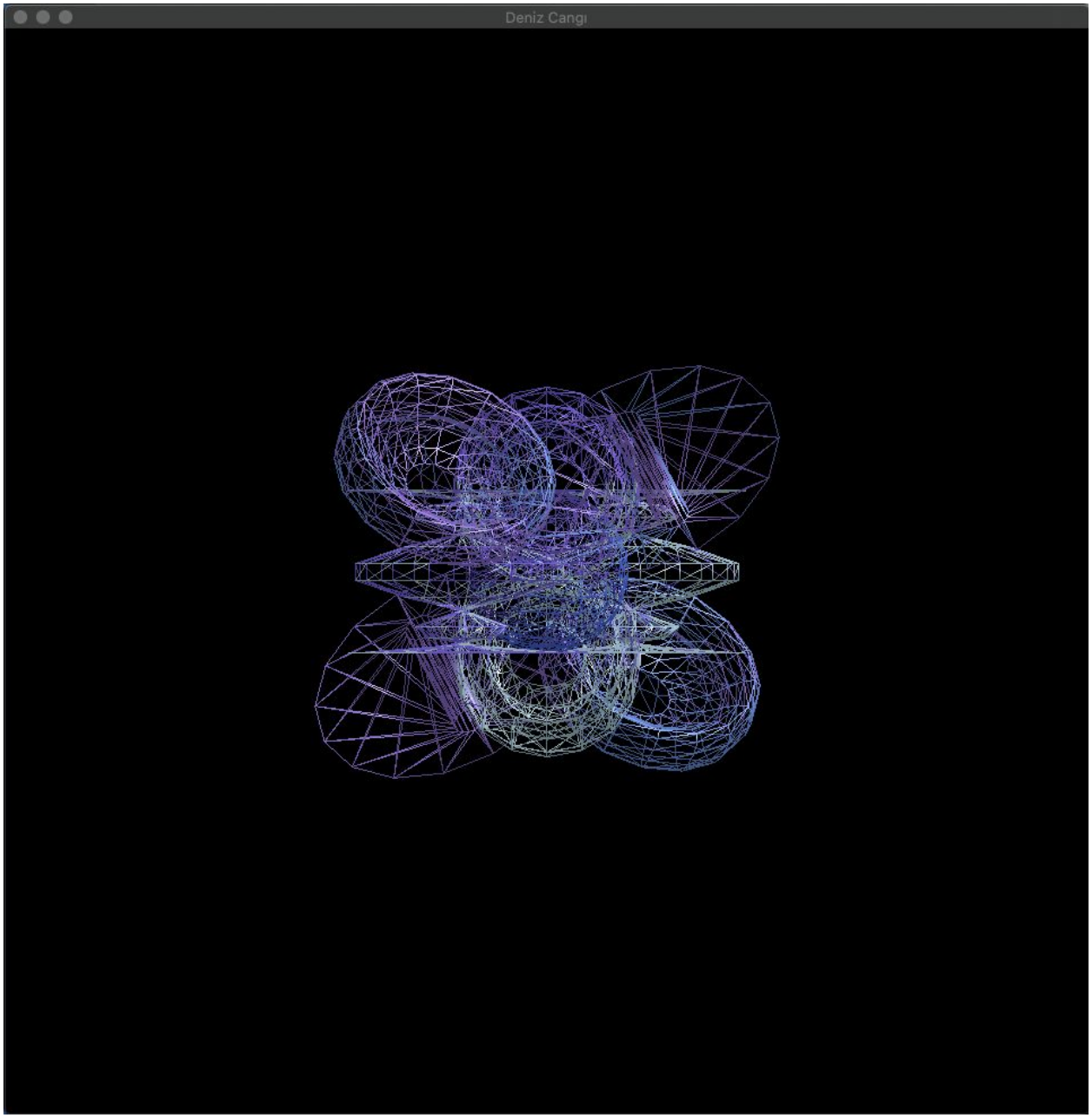
Other than that I have created 4 different meshes where all of them are created with some changes on ParametricHalfCircle. For the first one I have created two variables a= 4 and c = glm::dvec2(0.4,0) and returned glm::dvec2(cos(t*a), sin(t))+c.
For the second one I have created two variables a= 3 and c = glm::dvec2(0.4,0) and returned glm::dvec2(cos(t), sin(t*a)/a)+c.
For the third one I have created two variables a= 20 and c = glm::dvec2(0.7,0) and returned glm::dvec2(cos(t*a), sin(t*a))+c.
For the last one I have created two variables a= 8 and c = glm::dvec2(0.7,0) and returned glm::dvec2(cos(t), sin(t*a))+c.

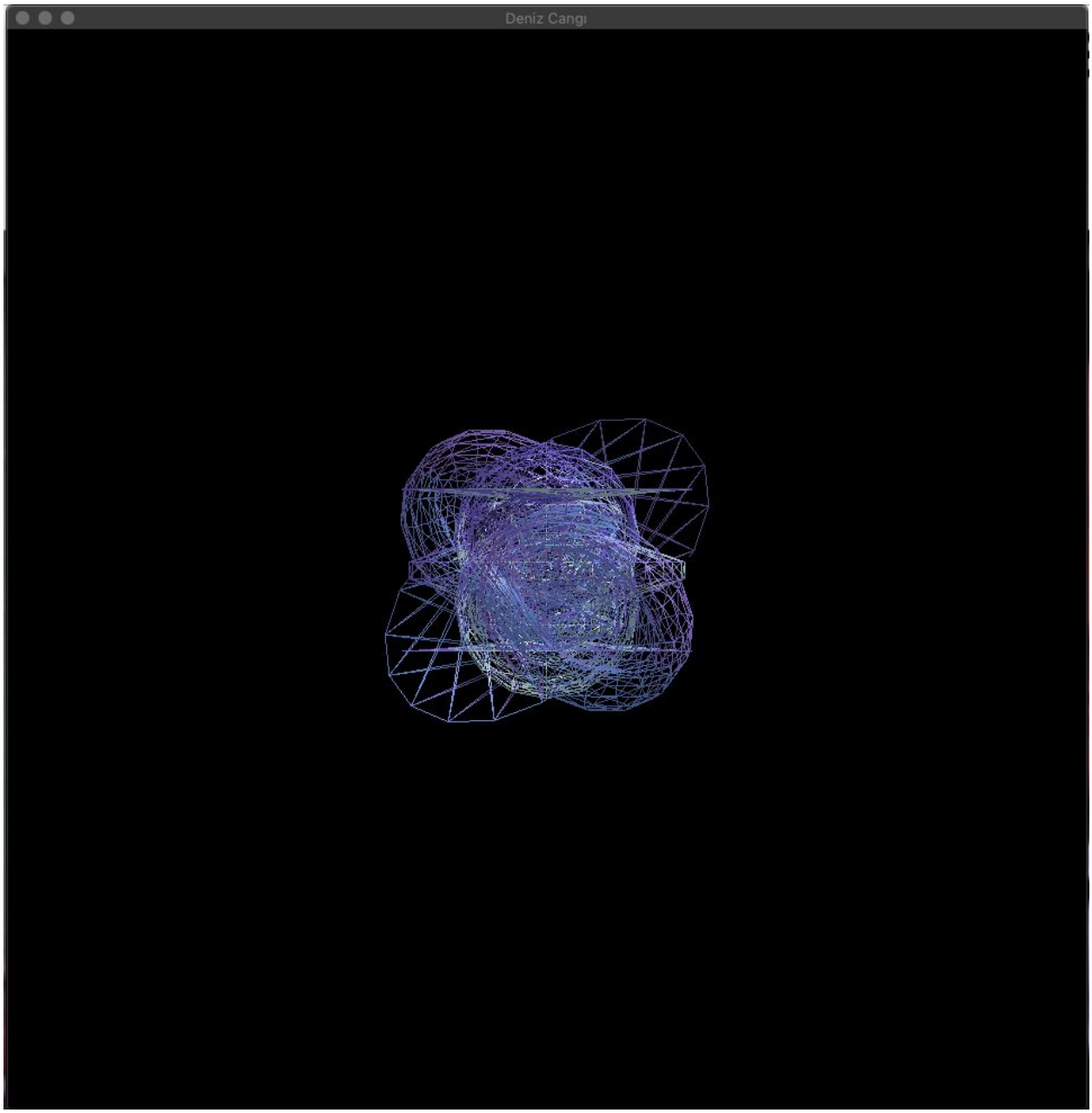
With these 4 meshes I have created 8 objects, 2 of each one and they are on opposite sides of the sun sphere. Each object is rotating around the sun with different diameters. 4 of them, the ones at horizontal and vertical axes are at positions (0.5,0,0), (-0.5,0,0), (0,0.5,0), (0,-0.5,0). The other 4 of them are at (0.6,0.6,0), (-0.6,0.6,0), (-0.6,-0.6,0), (0.6,-0.6,0). And they are changing color when they reach the opposite side. I have created this effect by creating 2 identical objects with different colors for each of 8 objects. The color scheme of this program is between blue and purple. In the end all of them rotate around themselves and rotate around the sun object and change colors while rotating.

The screenshots of the scene is:


Deniz Cangı


Deniz Cangı


Deniz Cangı


Deniz Cangı