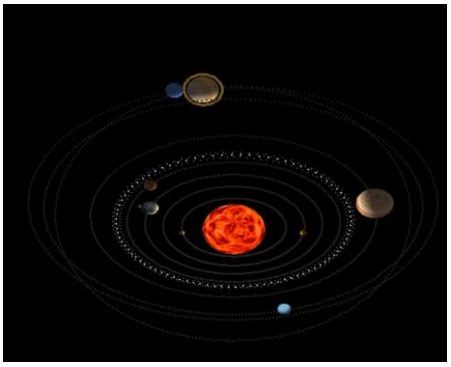
Group members:
Abdikani Mohamed Muse
Soroush Sajadi

DH2323
Project report

In this project we are trying to simulate the solar system which includes the sun, eight planets, asteroid belt and two moons.



Methods

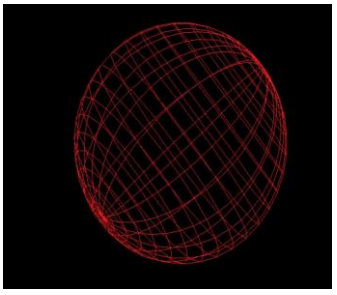
To complete the project we examine and discuss several steps:

- Simulation of the spheres and setting them in right coordinate and size
- Rotating the planets around the center (The sun)
- Rotating the planets around their own axle
- Simulation of orbits
- Simulation the light source
- Texturing
- Setting up the cameras

In this project we use freeglut and glew-1.11.0 library

Simulation of the spheres and setting them in right coordinate and size

At the first step we tried to create spheres with different size which are located in different coordinate. They represent our planets.



Group members:
Abdikani Mohamed Muse
Soroush Sajadi

We can obtain our goal with three functions that Glew library offers

$$void\ glutSolidSphere(GLdouble\ radius,\ GLint\ slices,\ GLint\ stacks);$$

Which creates the spheres

Radius: The radius of the sphere.

Slices: The number of subdivisions around the Z axis (similar to lines of longitude). Stacks :The number of subdivisions along the Z axis (similar to lines of latitude).

$$void\ glTranslated\ (GLdouble\ x, GLdouble\ y, GLdouble\ z);$$

Which determine the position of the sphere. Specify the *x*, *y*, and *z* coordinates of a translation vector.

$$void\ glScalef(GLfloat\ x, GLfloat\ y, GLfloat\ z);$$

Determine the relative size of the sphere. Specify scale factors along the x, y and z axes, respectively.

Rotating the planets around the center (The sun)

In the next step we try to rotate the planets around the center. We proceed our goal with function

$$Void\ glRotatef\ (\ GLfloat\ Angle, GLfloat\ x, GLfloat\ y, GLfloat\ z)$$

Specifies the angle of rotation, in degrees and the x, y, and z-coordinates of a vector. We do need to update our angle. For reaching this goal we make a function to update the angle between +360 , -360 degrees.

Rotating the planets around their own axle

This part of project is similar to the previous part. The only different part is GLfloat Angle must get different variables than GLfloat Angle for rotating around the center. In this case our GLfloat Angle gets the value 0.1.

Simulation of orbits

In this step we tried to implement the orbits which are the path that our planets circles around the center which is the sun. With use of three function

$$glRotatef, glScalef\ and\ glVertex2d$$

We are able to write our own orbit function. With help of these three functions we can determine the size, coordinate and angle of the orbits.

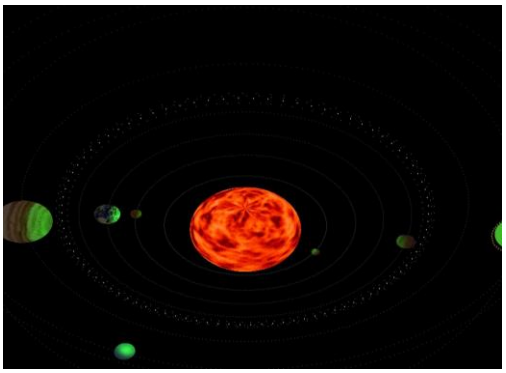
Simulation the light source

With help of the function glLightf we can set up a source light. In the function initLighting we set up light color, number and angle. In the next step we set up the position of the source light which in this project is the sun. By adding

$$void\ glLightf(GLenum\ light, GLenum\ pname, GLfloat\ param);$$

In “SUN” function we can implement the source light. In the function we Specifies the light. The number of lights depends on the implementation, but at least eight lights are supported. They are identified by symbolic names of the form GL_LIGHT i, where i ranges from 0 to the value of GL_MAX_LIGHTS - 1. The function also specifies a single-valued light source parameter for *light* and the value that parameter *pname* of light source *light* will be set to. The output that we get after implementing these function is pictured below

Group members:
Abdikani Mohamed Muse
Soroush Sajadi



A picture of light source with green light.

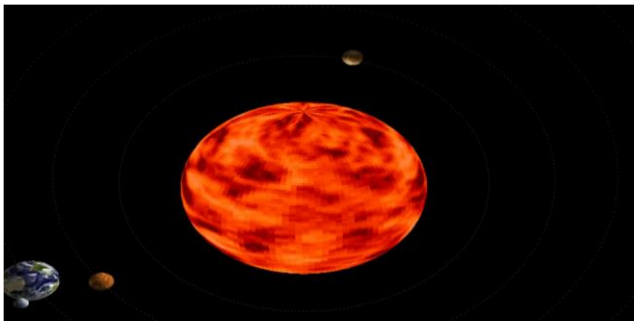
Texturing

To make our spheres look like real planets we need to use texturing method. By loading images with the format bmp. We want to consider that the size of image must be 64*64, 128*128 or 256*256. In this method we need to define a new class which calls the Image **.** Image will be used as class of our parameters in the function

$$GLuint\ loadTexture\ (Image\ \ image)\mathbf{.}$$

Image class considers width height and the pixels of the image. We use also use the function glTexImage2D in the loadTexture function which points at width, height and pixel of loaded image. Next we need to recall loadTexture function in the function initRendering to load the image and by gluNewQuadric creates and returns a pointer to a new quadrics object.

The next step is to complete the texture process by changing glutSolidSphere to gluSphere**.**
By changing the function we add the ability for the sphere to take the texture which gives us the output pictures below



Output of Texturing for Sun, earth, and…

Group members:
Abdikani Mohamed Muse
Soroush Sajadi

Setting up the cameras

In this section we use the function calls gluLookAt to define the camera.

$$Void\ gluLookAt\ (GLdouble\ eyeX, GLdouble\ eyeY, GLdouble\ eyeZ, GLdouble\ centerX, GLdouble\ centerY, GLdouble\ centerZ, GLdouble\ upX, GLdouble\ upY, GLdouble\ upZ)$$

The code for this sample application is now presented with comments where appropriate. First we need some global variables to store the camera parameters. The variables will store both the camera position and the vector that gives us the aiming direction. We will also store the angle by giving new inputs to the new variables when we can change the camera's perspective.

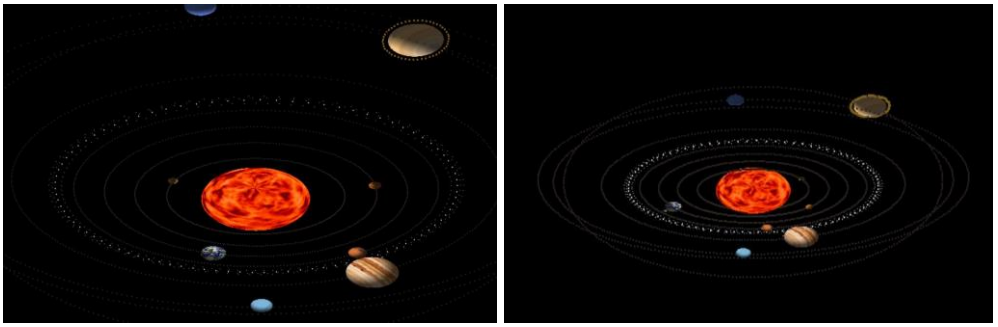
- Angle: the angle of rotation in the y axis. this variable will allow us to rotate the camera
- x,z: The camera position in the XZ plane
- lx,lz: A vector defining our line of sight

We also made a new function to control the cameras view.

$$void\ processSpecialKeys\ (int\ key,\quad int\ xx,\quad int\ yy)$$

In this function by defining the keys of keyboard we change the value of our variables .

- **case GLUT_KEY_LEFT:** moving to left.
- **case GLUT_KEY_RIGHT:** moving to right. ● **case GLUT_KEY_UP:** zoom in.
- **case GLUT_KEY_DOWN:** zoom out.
- **case GLUT_PAGE_UP:** looking up.
- **case GLUT_KEY_DOWN** looking down.



Output in different perspective

Group members and contact information

Abdikani Mohamed Muse
E-mail: uvv0831@gmail.com
Soroush Sajadi
E-mail: soorooshsadjadee@gmail.com

In the steps stated above (Method) we divided the steps. Abdikani worked on step 1, 2, 3 and 4 while Soroush worked on 4, 5, 6 and 7. The fourth step we worked together to implement it and for the most part in the project we had weekly meetups to discuss, give feedback and help each other so that the lab was completed. Even though the steps were divided between us we worked together most of the time so that all group members understood how the steps worked in the project.