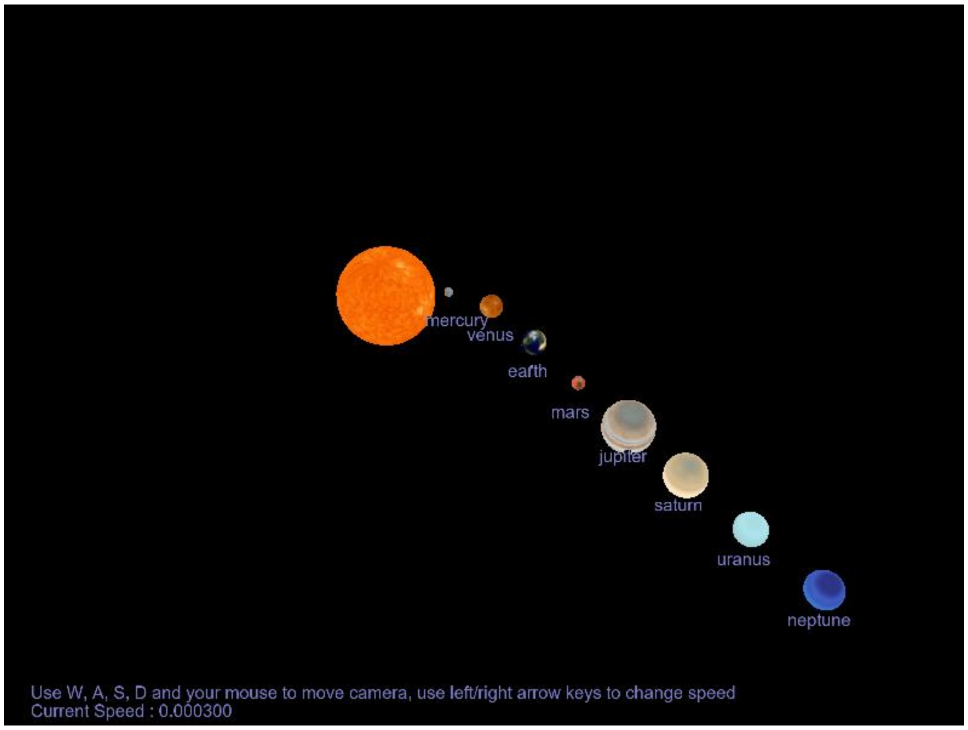
Assignment 3

Fedor Ivachev 费杰 2019280373

Planet rotates around sun



Use W, A, S, D and your mouse to move camera, use left/right arrow keys to change speed
Current Speed : 0.000300

In this assignment solar system needed to be rendered.

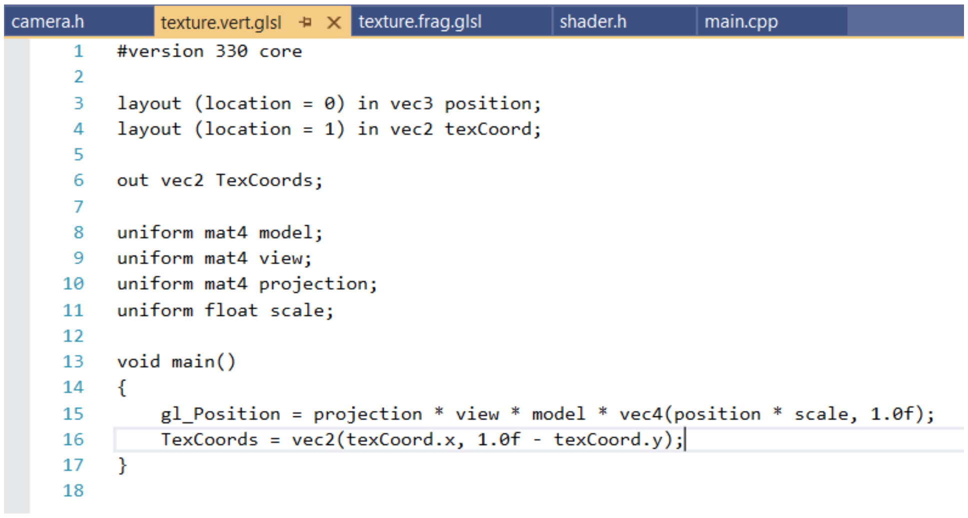
The project consists of files: camera.h, shader.h - camera and shader class objects, main.cpp, vertex and fragment shaders for rendering text and vertex and fragment shaders for rendering the solar system.

```
class planet
{
public:
    GLfloat angle;
    GLfloat distance;
    GLuint texture;
    GLfloat scale;
    double x;
    double y;
    std::string name;
};
```

1) Setting the orbits, sizes and textures of planets (no need to be same as real scale).

Each planet is defined by its angle, distance from the sun, texture id, scale as size, x and y coordinates on the screen space (for clipping their names), and the name. The orbits are different from real ones, same as the sizes of the planets. Although, the values were calculated that way that the user won't feel angry and cringe for seeing an ugly brother of solar system 😊.

Using the sphere.obj file with precalculated uvs, textures of each planet can be clipped to the object very smoothly.



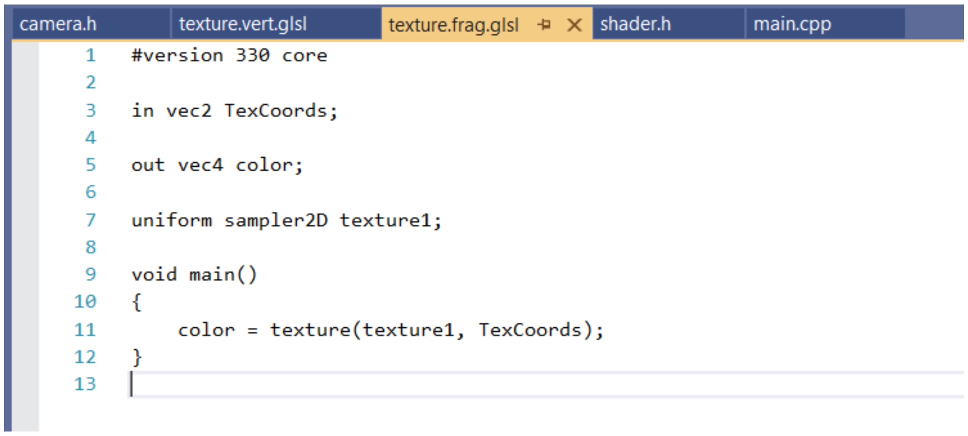
```glsl
#version 330 core

layout (location = 0) in vec3 position;
layout (location = 1) in vec2 texCoord;

out vec2 TexCoords;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;
uniform float scale;

void main()
{
    gl_Position = projection * view * model * vec4(position * scale, 1.0f);
    TexCoords = vec2(texCoord.x, 1.0f - texCoord.y);
}
```



```glsl
#version 330 core

in vec2 TexCoords;

out vec4 color;

uniform sampler2D texture1;

void main()
{
    color = texture(texture1, TexCoords);
}
```

Vertex shader and fragment shader.

2) Add keyboard control. E.g., press → key to increase the rotation speed along the x-axis

Each frame new coordinates of each planet, Sun and Moon are being calculated.

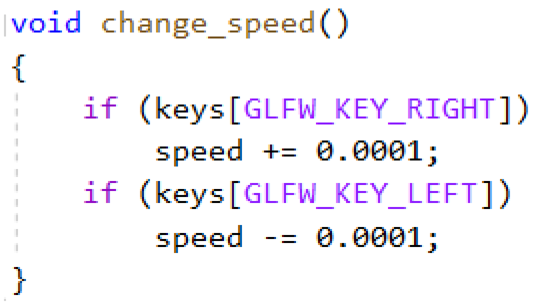
For planets: planets[i].distance * sin(planets[i].angle), 0.0f, planets[i].distance * cos(planets[i].angle)

For the Moon: planets[2].distance * sin(planets[2].angle) + planets[2].distance / 5 * cos(planets[2].angle * 28), 0.0f, planets[2].distance * cos(planets[2].angle) + planets[2].distance / 5 * sin(planets[2].angle * 28)

The angle for each planet is also updated. Every frame it linearly increases by the speed value.

planets[i].angle = planets[i].angle > 2 * M_PI ? planets[i].angle + M_PI * speed / log(i * i / 2.5 + 2) - 2 * M_PI : planets[i].angle + M_PI * speed / log(i * i / 2.5 + 2);

Here speed variable is responsible for the rotation speed. If speed is less than zero, planets start to move in a reverse direction.



```
void change_speed()
{
    if (keys[GLFW_KEY_RIGHT])
        speed += 0.0001;
    if (keys[GLFW_KEY_LEFT])
        speed -= 0.0001;
}
```

By pressing left and right keys, user can change the speed.

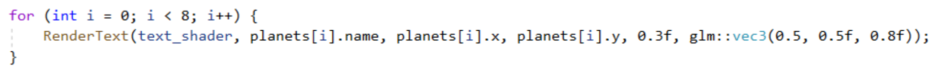
3) Add text besides planet. Planets move with names.

To successfully add FREETYPE.dll I had to 1) add freetype.dll to the project folder, 2) Change the font path to relative.

Text is rendered by RenderText function.

To calculate the position of planets on screen, glhProjectf function was used, which is based on gluProject function of getting screen coordinates from object coordinates.

After multiplying view by model_vertex, we can easily calculate the window coordinates of each planet and store them in x and y fields of corresponding planet object.



```
for (int i = 0; i < 8; i++) {
    RenderText(text_shader, planets[i].name, planets[i].x, planets[i].y, 0.3f, glm::vec3(0.5, 0.5f, 0.8f));
}
```

After that, text is rendered near each of the planets.

The video and code can also be downloaded from Github: https://github.com/FedorIvachev/GraphicsCourse