Sami Emre Erdogan
2019280513

Rubik's Cube

For this project it took me a really long time to understand how the code was structured but the hardest part was to configure the dependencies. I always used my MacOS and Cmake to build my projects and the first time I have managed to do it but I had an unfortunate event and my whole data was wiped out. The second time when I tried to build the project it wasn't showing the cube when I built and ran the framework because I had a limited time I decided to use my old windows laptop instead to build my project and it worked.

In the given code it was stating or a hint was given that we could use trackball and I also noticed in the RubikCube.h file " #include "trackball.h" was commented out. I haven't created my own trackball.h but I have created my own vec3 trackball as shown below: (The rough explanation has been commented). This allowed us to get the vectors on the trackball

```
// *******Created my own trackball ***********
// get a point on the unit sphere from screen coordinates
glm::vec3 RubikCube::get_trackball_vec(GLfloat x, GLfloat y, GLfloat WIDTH, GLfloat HEIGHT){
    // get x and y in a range between -1 and 1
    glm::vec3 p(1.f * x / WIDTH * 2 - 1.f, 1.f * y / HEIGHT * 2 - 1.f, 0);
    p.y *= -1;
    // compute z variable on sphere if (x,y) is in unit circle, otherwise put (x,y) on circle border
    float p_squared = p.x * p.x + p.y * p.y;
    if(p_squared <= 1.f){
        p.z = sqrt(1 - p_squared);
    }else{
        p = glm::normalize(p);
    }
    return p;
}
```

The next function I have implemented was to rotate the whole cube using the trackball that I have created. This will allow us transform the rotation axis to object coordinates and rotate the model matrix.

```
// rotate the whole cube based on the current and last cursor position using a trackball model
void RubikCube::Mouse_LeftMove(GLfloat current_x, GLfloat last_x, GLfloat current_y, GLfloat last_y,
    // get the vectors on the trackball anc compute the rotation axis and angle
    GLfloat WIDTH, GLfloat HEIGHT, glm::mat4 view){
    glm::vec3 a = get_trackball_vec(last_x, last_y, WIDTH, HEIGHT);
    glm::vec3 b = get_trackball_vec(current_x, current_y, WIDTH, HEIGHT);
    float angle = acos(fmin(1.f, glm::dot(a, b)));
    glm::vec3 axis_in_camera_coords = glm::cross(a, b);
    // transform the rotation axis to object coordinates
    glm::mat3 cam_to_object_trafo = glm::inverse(view * model);
    glm::vec3 axis_in_object_coords = cam_to_object_trafo * axis_in_camera_coords;
    // rotate the model matrix. The factor of 2 makes the rotating experience more satisfying
    model = glm::rotate(model, angle * 2.f, axis_in_object_coords);
}
```

Next, I had to rotate and update all sub cube model matrices and applying it to the layer. This was a part of the rotation step function. I had to update all sub cube model matrices.

```
    /*
    TODO: rotate the subcubes
    You should compute a rotation matrix and apply it to the layer
    */
    // update all subcube model matrices
    for(int i = 0; i < 9; i++){
        Cube* sub_cube = layers[rotation.ax_ind][rotation.layerY].cubes[i];
        glm::mat4 rot = glm::rotate(glm::mat4(1), glm::radians(rotation.offset), axis_true);
        sub_cube->m_cube = rot * sub_cube->m_cube;
    }
}
```

Sami Emre Erdogan
2019280513

As stated on the comments, the positions of cubes have been updated but the connection between the cubes and the layers have not been updated and this causes problems.
For the sub-cubes of rotated layers, we had to update which cubes belonged to a layer. This function included two cases: clockwise and counterclockwise. It updates the affiliation of sub cubes to the layers and the layer reference in each sub cube by making a copy of the sub cube data, moving all cubes on the outer ring positions counterclockwise and updates the layers based on our updated sub cube data that we have gathered.

```
//TO DO
// updates the affiliation of subcubes to the layers and the layer reference in each subcube
void RubikCube::RotateFinish(){
    // make copy of subcube data (use vectors, as they automatically handle memory allocation)
    vector<vector<vector<int>>> sub_cube_data(8, vector<vector<int>>(2, vector<int>(3)));
    vector<Cube*> sub_cubes(8);
    for(int i = 0; i < 8; i++){
        sub_cubes[i] = layers[rotation.ax_ind][rotation.layerY].cubes[i];
        for(int j = 0; j < 3; j++){
            sub_cube_data[i][0][j] = sub_cubes[i]->layer[j];
            sub_cube_data[i][1][j] = sub_cubes[i]->lay_ind[j];
        }
    }

    for(int i = 0; i < 8; i++){
        // move all cubes on the outer ring x positions counterclockwise
        for(int j = 0; j < 3; j++){
            sub_cubes[i]->layer[j] = sub_cube_data[(i + (int)rotation.ang_roted / 45) % 8][0][j];
            sub_cubes[i]->lay_ind[j] = sub_cube_data[(i + (int)rotation.ang_roted / 45) % 8][1][j];
        }
        // update layers based on updated subcube data
        layers[0][sub_cubes[i]->layer[0]].cubes[sub_cubes[i]->lay_ind[0]] = sub_cubes[i];
        layers[1][sub_cubes[i]->layer[1]].cubes[sub_cubes[i]->lay_ind[1]] = sub_cubes[i];
        layers[2][sub_cubes[i]->layer[2]].cubes[sub_cubes[i]->lay_ind[2]] = sub_cubes[i];
    }
}
```

Working Screenshots: