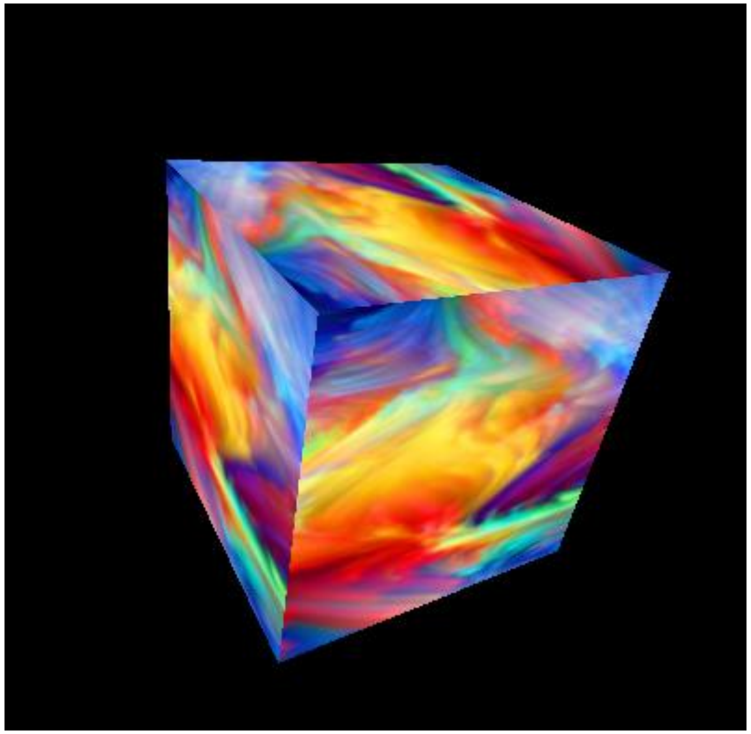
Open GL Assignment 2
Computer Graphics

Eduardo Bier s3065979
Nadia Hartsuiker s2355809

Texture Mapping

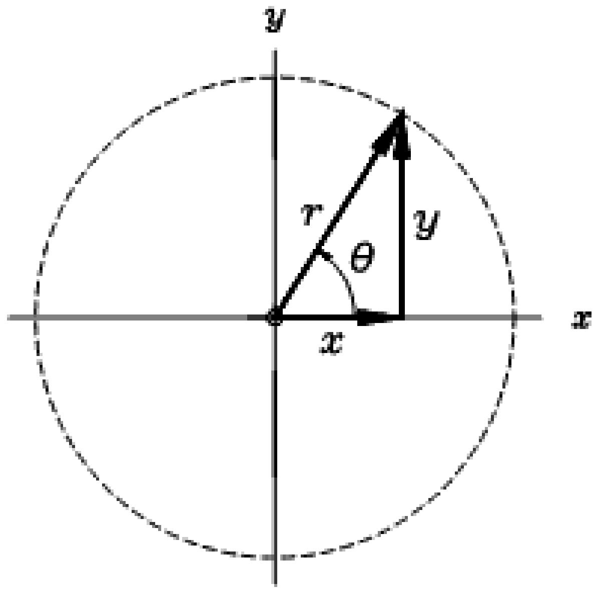
Used texture (.png image)

Texture Mapping (2)



Texture mapped on the cube in the interactive environment of the first Open GL lab assignment

Animation

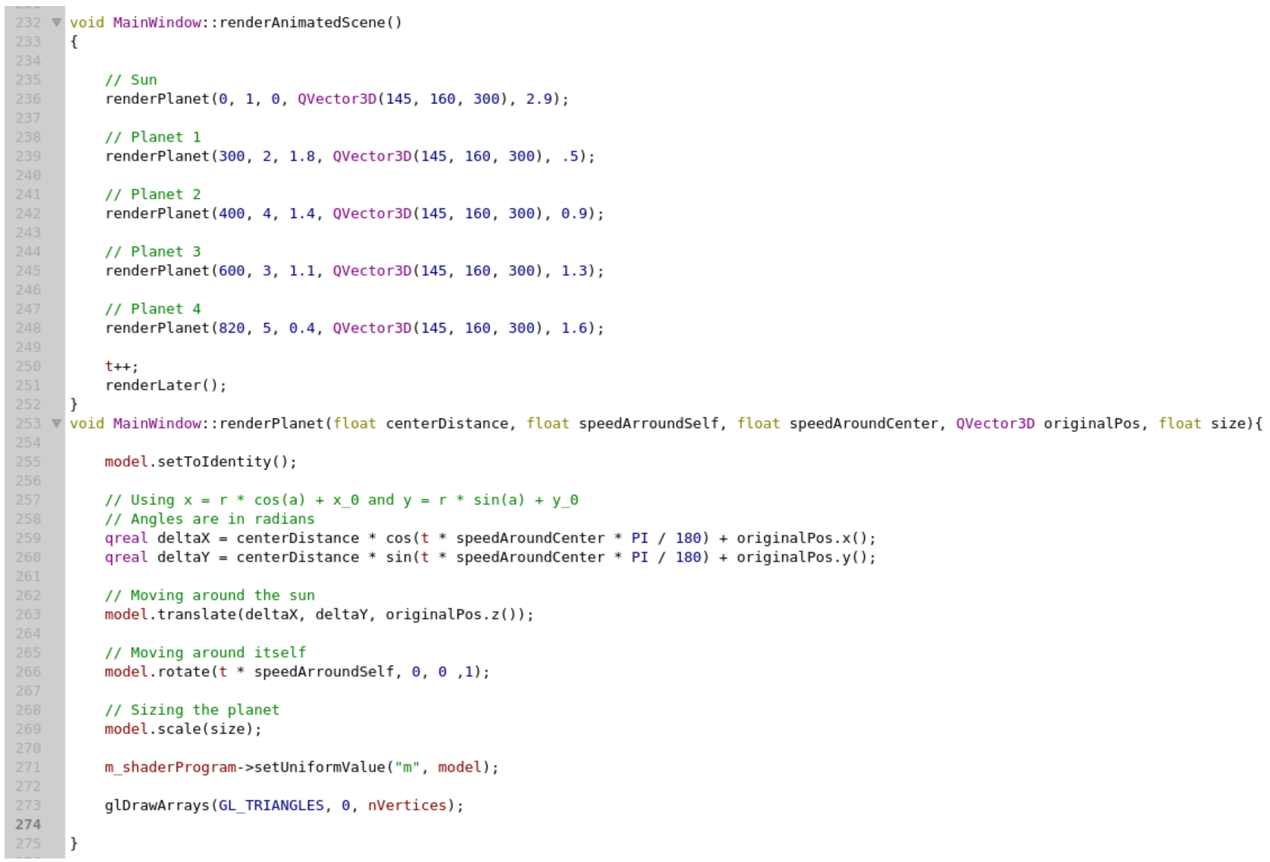


x and y coordinates when rotating around a center
Source: http://mathonweb.com/help_ebook/html/trigonometry.htm

$$x = r * \cos(speed * t) + x.pos\ center$$

$$y = r * sin(speed * t) + y.pos\ center$$

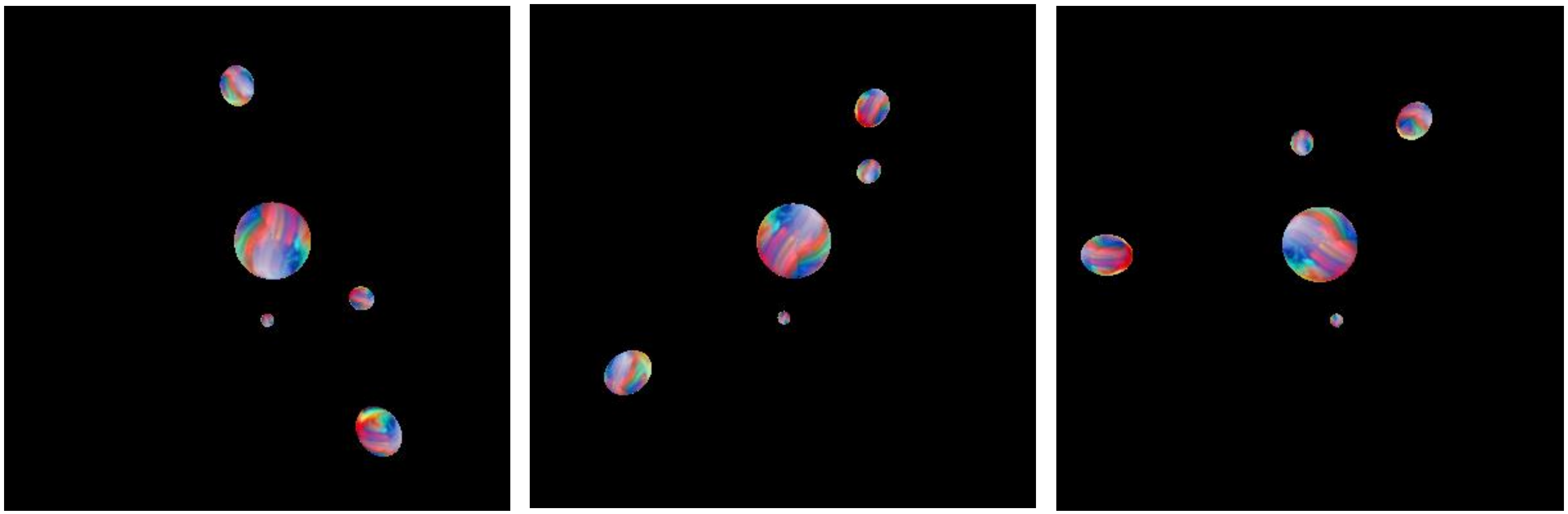
Animation (2)



```
void MainWindow::renderAnimatedScene()
{

    // Sun
    renderPlanet(0, 1, 0, QVector3D(145, 160, 300), 2.9);

    // Planet 1
    renderPlanet(300, 2, 1.8, QVector3D(145, 160, 300), .5);

    // Planet 2
    renderPlanet(400, 4, 1.4, QVector3D(145, 160, 300), 0.9);

    // Planet 3
    renderPlanet(600, 3, 1.1, QVector3D(145, 160, 300), 1.3);

    // Planet 4
    renderPlanet(820, 5, 0.4, QVector3D(145, 160, 300), 1.6);

    t++;
    renderLater();
}
void MainWindow::renderPlanet(float centerDistance, float speedArroundSelf, float speedAroundCenter, QVector3D originalPos, float size){

    model.setToIdentity();

    // Using x = r * cos(a) + x_0 and y = r * sin(a) + y_0
    // Angles are in radians
    qreal deltaX = centerDistance * cos(t * speedAroundCenter * PI / 180) + originalPos.x();
    qreal deltaY = centerDistance * sin(t * speedAroundCenter * PI / 180) + originalPos.y();

    // Moving around the sun
    model.translate(deltaX, deltaY, originalPos.z());

    // Moving around itself
    model.rotate(t * speedArroundSelf, 0, 0 ,1);

    // Sizing the planet
    model.scale(size);

    m_shaderProgram->setUniformValue("m", model);

    glDrawArrays(GL_TRIANGLES, 0, nVertices);

}
```

Code for rendering the animated scene

Animation (3)



Screenshots of animation of a ‘solar system’