Sanketh Karuturi
karutusa@oregonstate.edu

CS 450 - Computer Graphics
Project #4
Keytime Animation

1) **Description of Actions Taken:**

- **Keytime Animation**: Keytime values were initialized and set for various properties of objects and lighting using the **Keytimes** class. This involves specifying values at specific times which are then interpolated smoothly to create animation over the 10-second period.

- **Lighting and Material**: OpenGL lighting was utilized to give the animated objects a realistic appearance. Material properties of objects were animated to change their appearance over time.

- **Object Properties**: Various properties of the objects such as position, rotation, and scale were animated to make the objects move within the scene.

- **Camera Animation**: The camera's look-at position was animated to change the viewing perspective during the animation.

2) **Four "Own-Choice" Animated Quantities:**

    a. Object Transparency (**ObjectTransparency**)
    b. Light Intensity (**lightIntensity**)
    c. Light Position (**lightPosX**, **lightPosY**, **lightPosZ**)
    d. Light Color (**lightColorR**, **lightColorG**, **lightColorB**)

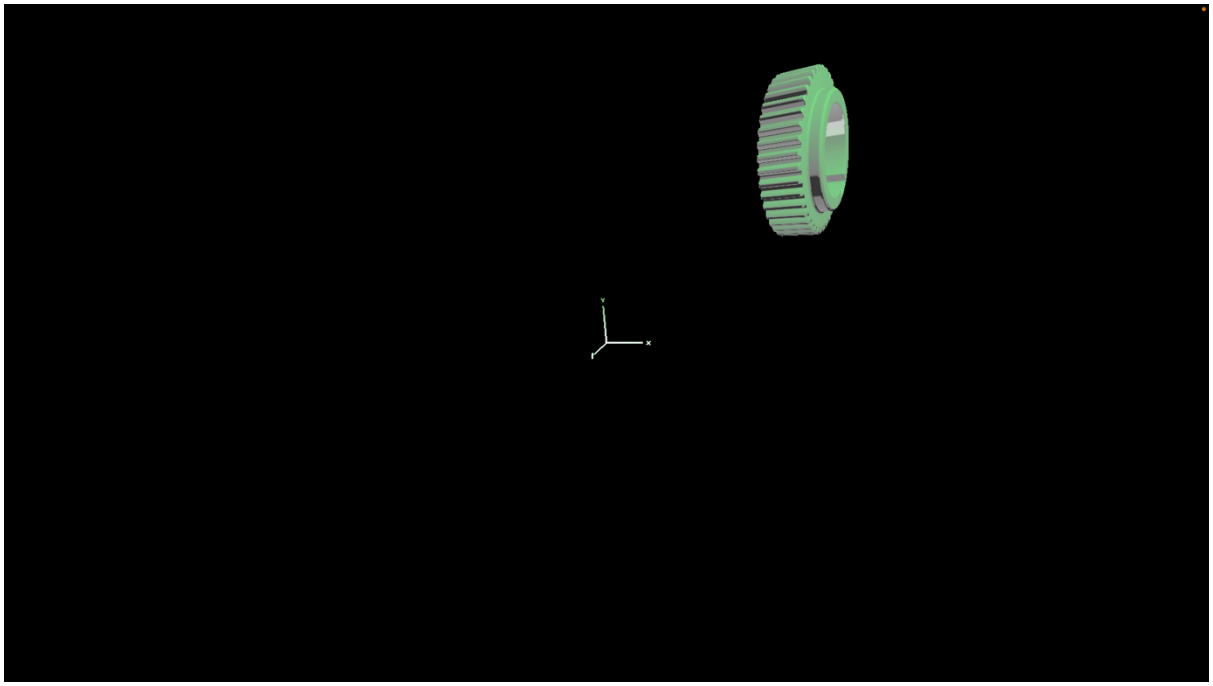
3) **Table of Keytime Values:**

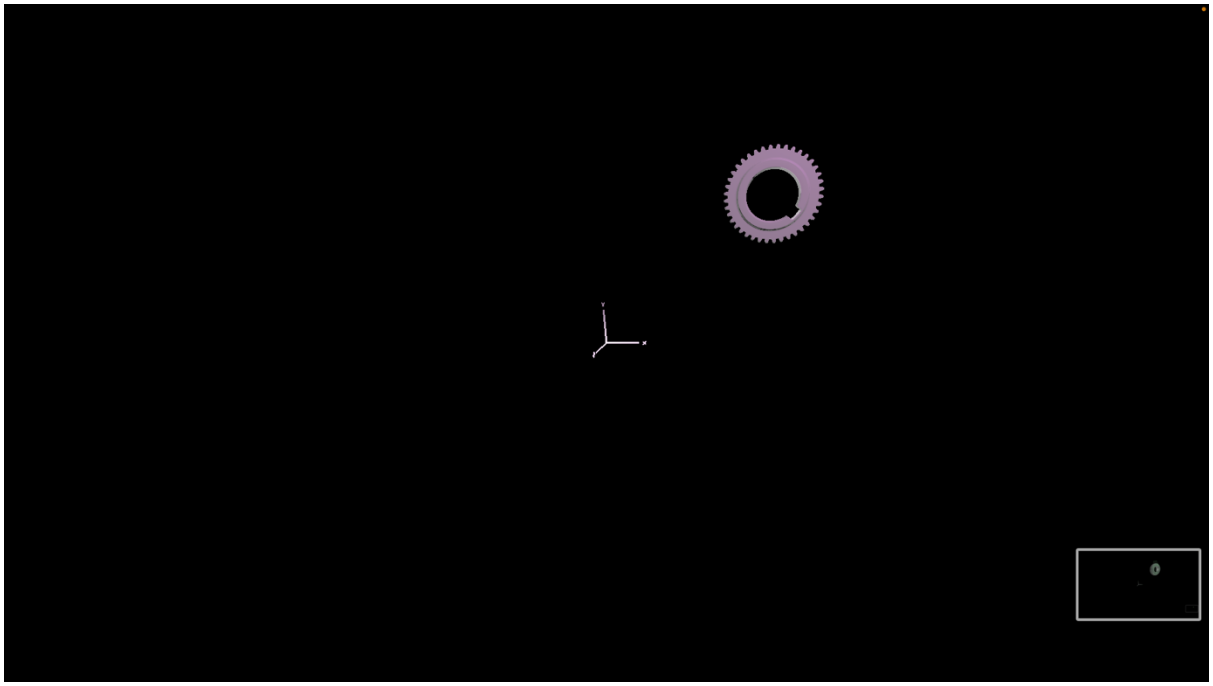
```
// Object Transparency
ObjectTransparency.AddTimeValue(0.0, 0.0);
ObjectTransparency.AddTimeValue(2.0, 0.3);
ObjectTransparency.AddTimeValue(5.0, 0.7);
ObjectTransparency.AddTimeValue(7.0, 0.9);
ObjectTransparency.AddTimeValue(10.0, 1.0);

// Light Intensity
lightIntensity.AddTimeValue(0.0, 0.2);
lightIntensity.AddTimeValue(2.0, 0.5);
lightIntensity.AddTimeValue(4.0, 0.8);
lightIntensity.AddTimeValue(6.0, 0.5);
lightIntensity.AddTimeValue(8.0, 0.2);
lightIntensity.AddTimeValue(10.0, 1.0);
```

```
// Light Position X
lightPosX.AddTimeValue(0.0, originalPosX);
lightPosX.AddTimeValue(2.0, 5.0);
lightPosX.AddTimeValue(5.0, -5.0);
lightPosX.AddTimeValue(7.0, 10.0);
lightPosX.AddTimeValue(10.0, originalPosX);

// Light Position Y
lightPosY.AddTimeValue(0.0, originalPosY);
lightPosY.AddTimeValue(2.0, 5.0);
lightPosY.AddTimeValue(5.0, -5.0);
lightPosY.AddTimeValue(7.0, 10.0);
lightPosY.AddTimeValue(10.0, originalPosY);

// Light Position Z
lightPosZ.AddTimeValue(0.0, originalPosZ);
lightPosZ.AddTimeValue(2.0, 5.0);
lightPosZ.AddTimeValue(5.0, -5.0);
lightPosZ.AddTimeValue(7.0, 10.0);
lightPosZ.AddTimeValue(10.0, originalPosZ);

// Light Color Red
lightColorR.AddTimeValue(0.0, 1.0);
lightColorR.AddTimeValue(1.0, 0.0);
lightColorR.AddTimeValue(2.0, 1.0);
lightColorR.AddTimeValue(3.0, 1.0);
lightColorR.AddTimeValue(4.0, 0.0);
lightColorR.AddTimeValue(5.0, 1.0);
lightColorR.AddTimeValue(6.0, 1.0);
lightColorR.AddTimeValue(7.0, 0.0);
lightColorR.AddTimeValue(8.0, 1.0);
lightColorR.AddTimeValue(9.0, 0.0);

// Light Color Green
lightColorG.AddTimeValue(0.0, 0.0);
lightColorG.AddTimeValue(1.0, 1.0);
lightColorG.AddTimeValue(2.0, 0.0);
lightColorG.AddTimeValue(3.0, 0.0);
lightColorG.AddTimeValue(4.0, 1.0);
lightColorG.AddTimeValue(5.0, 0.0);
lightColorG.AddTimeValue(6.0, 0.0);
lightColorG.AddTimeValue(7.0, 1.0);
lightColorG.AddTimeValue(8.0, 0.0);
lightColorG.AddTimeValue(9.0, 1.0);

// Light Color Blue
lightColorB.AddTimeValue(0.0, 0.0);
lightColorB.AddTimeValue(1.0, 0.0);
lightColorB.AddTimeValue(2.0, 1.0);
lightColorB.AddTimeValue(3.0, 0.0);
lightColorB.AddTimeValue(4.0, 0.0);
```

```
lightColorB.AddTimeValue(5.0, 1.0);
lightColorB.AddTimeValue(6.0, 0.0);
lightColorB.AddTimeValue(7.0, 0.0);
lightColorB.AddTimeValue(8.0, 1.0);
lightColorB.AddTimeValue(9.0, 0.0);
```

4) **Cool-Looking Screenshots from the Program:**





5) **What Convinces You That Your Animation Is Doing What You Set It Up to Do:**

The use of **Keytimes** class with specified values at different times and the in-built interpolation functionality ensures that the animation transitions smoothly between key frames. The consistent increase and decrease of values according to the keytimes, returning to their original state at the end of the 10-second cycle, is indicative of the animation performing as expected. The animation is driven by real-time calculations based on system time (**glutGet(GLUT_ELAPSED_TIME)**), which provides a reliable basis for smooth animation over the designated period.

6) link to the video demonstrating project #4:
https://media.oregonstate.edu/media/t/1_4gbpz8bf