Oregon State University

CS 450/550 -- Fall Quarter 2023

Project #4

100 Points

Due: November 6

Keytime Animation

This page was last updated: September 6, 2023

Introduction

This project is about performing a keytime animation. You will be the Senior Animator and will choose key values to use in your animation. The computer will be the Grunt Animator and will fill in interpolated values in between your key values.

Learning Objective:

When you are done with this assignment, you will understand how to get your scene's objects to move in ways that you get to control, without having to write any equations or do any physics.

Instructions

1. Do a keytime animation on a 3D object that is capable of being lighted (your choice). There need to be at least **8** quantities being animated, each with at least **6** keytimes that define each animation.

2. The 8 quantities to be animated must be:
   1 quantity that has something to do with a light source (position, color, etc.)
   1 quantity that has something to do with viewing (eye position, look position, etc.)
   1 quantity that has something to do with positioning, orienting, or scaling the object
   1 quantity that has something to do with the color of the object (r, g, or b)
   4 more quantities -- your choice

   Make the animation last 10 seconds. At time=10 seconds, bring each quantity back to its original time=0. value

3. Use lighting in the scene. Your choice of lighting parameters is up to you, but choose them so that it is easy for us to understand your scene.

4. The object to be displayed is your choice of any 3D geometry. As lighting is required, be sure that object has surface normal vectors.

5. Animate your quantities by smoothly interpolating them between the keytimes. Use the smooth interpolation C++ class that we discussed. For your convenience, the class methods are re-covered below.

Smooth Interpolation:

The smooth interpolation technique that you will be using employs the Coons cubic curve (end points and end slopes). But, rather than you having to specify all of the slopes, the technique makes a reasonable approximation of them based on the surrounding keytime information.

The methods for the Keytimes class are:

```
void    AddTimeValue( float time, float value );
float   GetFirstTime( );
float   GetLastTime( );
int     GetNumKeytimes( );
float   GetValue( float time );
void    Init( );
void    PrintTimeValues( );
```

Set the values like this:

```
// a defined value:
const int MSEC = 10000;         // 10000 milliseconds = 10 seconds


// a global:
Keytimes Xpos1;


// in InitGraphics( ):
        Xpos1.Init( );
        Xpos1.AddTimeValue(  0.0,  0.000 );
        Xpos1.AddTimeValue(  0.5,  2.718 );
        Xpos1.AddTimeValue(  2.0,  0.333 );
        Xpos1.AddTimeValue(  5.0,  3.142 );
        Xpos1.AddTimeValue(  8.0,  2.718 );
        Xpos1.AddTimeValue( 10.0,  0.000 );


// in Animate( ):
        glutSetWindow( MainWindow );
        glutPostRedisplay( );


// in Display( ):
        // turn # msec into the cycle ( 0 - MSEC-1 ):
        int msec = glutGet( GLUT_ELAPSED_TIME )  %  MSEC;

        // turn that into a time in seconds:
        float nowTime = (float)msec  / 1000.;

        glPushMatrix( );
                glTranslatef( Xpos1.GetValue( nowTime ), 0., 0. );
                glRotatef( 0.,  1., 0., 0. );
                glRotatef( 0.,  0., 1., 0. );
                glRotatef( 0.,  0., 0., 1. );
                << draw object #1 >>
        glPopMatrix( );
```

A Debugging Suggestion:

- One thing that has always helped Joe Graphics debug animation programs is to have a "freeze" option, toggled with the '**f**' key. This freezes the animation so you can really look at your Cessna and propellers and see if they are being drawn correctly. Remember what the current freeze status is with a boolean global variable:

  ```
  bool    Frozen;
  ```

  Set **Frozen** to *false* in **Reset( )**. Then, freezing the animation is just a matter is setting the Idle Function to **NULL**. To un-freeze it, set the Idle Function back to **Animate( )**. So, in the **Keyboard( )** callback, you could say something like:

  ```
  case 'f':
  case 'F':
          Frozen = ! Frozen;
          if( Frozen )
                  glutIdleFunc( NULL );
          else
                  glutIdleFunc( Animate );
          break;
  ```

+10 points Extra Credit:

Do the same keytime process (8 quantities, 6 keytimes each) with a second object. Make it look like it is somehow "involved" with the first object.

Turn-in:

Use the Teach system to turn in:

1. Your .cpp file
2. A PDF report containing:
   - Project number and title
   - Your name
   - Your email address
   - A description of what you did to get the display you got.
   - Tell us what your 4 "own-choice" animated quantities were.
   - A table showing your keytime values for each quantity. It is OK just to include the lines of code that set them.
   - A couple of cool-looking screen shots from your program.
   - Tell us what convinces you that your animation is indeed doing what you set it up to do.
   - The link to the your video demonstrating that your project does what the requirements ask for. If you can, we'd appreciate it if you'd narrate your video so that you can tell us what it is doing.
   - If you did the Extra Credit, for the second object also tell us:
     - A description of what you did to get the display you got.
     - Tell us what your 4 "own-choice" animated quantities were.
     - A table showing your keytime values for each quantity. It is OK just to include the lines of code that set them.
     - A couple of cool-looking screen shots from your program.
     - Tell us what convinces you that your animation is indeed doing what you set it up to do.

3. **Be sure that your video's permissions are set to *unlisted*.**. The best place to set this is on the OSU Media Server.

4. A good way to test your video's permissions is to ask a friend to try to open the same video link that you are giving us.

5. The video doesn't have to be made with Kaltura. Any similar tool will do.

Grading:

Item	Points
Animation lasted 10 seconds	10
Scene used OpenGL lighting	10
Light source-related animation quantity	10
View-related animation quantity	10
Position/orientation/scale animation quantity	20
Color-related animation quantity	20
The four own-choice animation quantities	20
Extra Credit	10
Potential Total	**110**

If you do the Extra Credit, be sure you sufficiently document what you did in your PDF report and make it obvious what you did in your video!