Viewer Movement in OpenGL

Tom Kelliher, CS 320

Apr. 3, 2013

1 Administrivia

Announcements

New project handout.

Assignment

Read 4.6–4.7.

From Last Time

The rotating color cube.

Outline

1. Understanding clipping volumes and their specifications.
2. Projections.
3. Movements in 3-D.
4. Toward a better movement model.

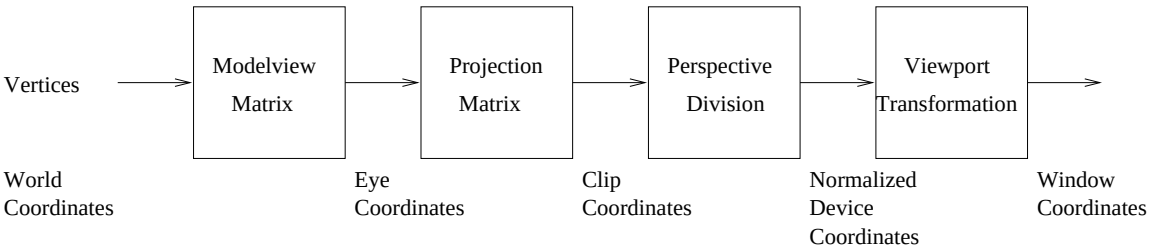
Coming Up

Movement through a room.

2 Preliminary: Viewing Volumes

Are our viewing volume coordinates relative or absolute?

Consider:



1. By default, the eye is at $(0, 0, 0)$ looking down the $-z$ axis.

2. What does

   ```
   Ortho(-10.0, 10.0, -5.0, 5.0, -2.0, 2.0);
   ```

   mean?

3. Other viewing modes:

   (a) `Frustum`: same parameters as `Ortho`. What's a frustum? Truncated pyramid.

   (b) `Perspective`: fovy, aspect ratio, zNear, and zFar.

   znear and zfar need to be **positive**.

3 Moving and Positioning the Eye

View specification:

1. One way of specifying eye position and viewing angle:

   (a) Specify position of eye.

   (b) Specify center of field of view.

   (c) Specify "up."

2. Use of `LookAt()` in `display()` of `cubeview.cpp`:

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

    point4  camera( cameraPos[X],
                    cameraPos[Y],
                    cameraPos[Z],
                    1.0 );
    point4  at( 0.0, 0.0, 0.0, 1.0 );
    vec4    up( 0.0, 1.0, 0.0, 0.0 );

    mat4  mv = LookAt( camera, at, up ) * RotateZ(theta[Z])
               * RotateY(theta[Y]) * RotateX(theta[X]);

    glUniformMatrix4fv( model_view, 1, GL_TRUE, mv );

    glDrawArrays( GL_TRIANGLES, 0, NumVertices );

    glutSwapBuffers();
}
```

   Note order of matrix multiplications: view, then model transformations.

3. Is it really necessary to have view and model transformations?

4. `reshape()` handles the projection transformation:

```
void reshape( int w, int h )
{
   mat4 p;
   GLfloat ratio;

   glViewport( 0, 0, w, h );

   if(w <= h)
   {
      ratio = (GLfloat) h / (GLfloat) w;
      p = Frustum( LEFT, RIGHT, ratio * BOTTOM, ratio * TOP,
                   Z_NEAR, Z_FAR);
   }
   else
   {
      ratio = (GLfloat) w / (GLfloat) h;
      p = Frustum( ratio * LEFT, ratio * RIGHT, BOTTOM, TOP,
                   Z_NEAR, Z_FAR);
   }

   /* Or, just use perspective(). */

   //p = Perspective(45.0, w/h, 2.0, 20.0);

   glUniformMatrix4fv( projection, 1, GL_TRUE, p );
}
```

5. Interactivity handled in `keyboard()` and `mouse()` handlers:

```
void keyboard( unsigned char key, int x, int y )
{
    switch( key )
    {

    case ESC:
    case 'q':
    case 'Q':
       exit( EXIT_SUCCESS );
       break;

    case 'g':
    case 'G':
       glutIdleFunc(idle);
       timeStamp = GetTickCount();
```

```
        break;

    case 's':
    case 'S':
        glutIdleFunc(NULL);
        break;

    case 'x': cameraPos[X] -= 1.0; break;
    case 'X': cameraPos[X] += 1.0; break;

    case 'y': cameraPos[Y] -= 1.0; break;
    case 'Y': cameraPos[Y] += 1.0; break;

    case 'z': cameraPos[Z] -= 1.0; break;
    case 'Z': cameraPos[Z] += 1.0; break;

    case ' ':  // reset values to their defaults
        glutIdleFunc(NULL);
        cameraPos = vec3(0.0, 0.0, 5.0);
        theta = vec3(0.0, 0.0, 0.0);
        axis = Z;
        break;
    }

    std::cout << cameraPos[X] << " "
              << cameraPos[Y] << " "
              << cameraPos[Z] << std::endl;

    glutPostRedisplay();
}


void mouse(int btn, int state, int x, int y)
{
   if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN) axis = X;
   if(btn==GLUT_MIDDLE_BUTTON && state == GLUT_DOWN) axis = Y;
   if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN) axis = Z;

   theta[axis] += 2.0;

   if( theta[axis] > 360.0 ) theta[axis] -= 360.0;

   glutPostRedisplay();
}
```

6. `idle()` does its thing:

```
void idle( void )
{
   int currentTime = GetTickCount();

   theta[axis] += 0.001 * (currentTime - timeStamp) * 360.0 * revsPerSec;
   timeStamp = currentTime;

   if ( theta[axis] > 360.0 ) {
      theta[axis] -= 360.0;
   }

    glutPostRedisplay();
}
```

3.1 Example Runs

1. P1: Stock cubeview using frustum. Demonstrate clipping, invisibility when up vector is parallel to line of sight, walking through the cube.

2. P2: Perspective view with fovy 45, near 2, and far 20.

3. P3: Perspective view with fovy 135, near 0.1, far 100.

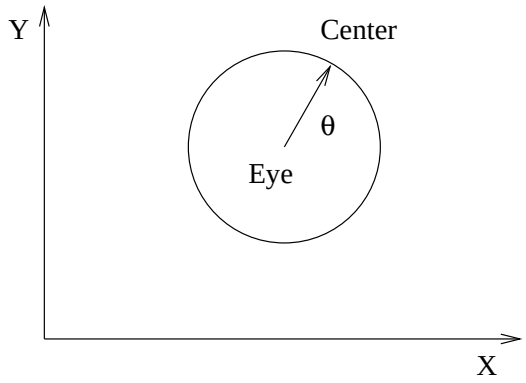
4 A Movement Model

Problems with viewer movement in cubeview:

1. Must specify movement in global coordinate values.

2. Can't speak of left, right, forward, backward, etc.

Consider this model:



Eye: (x, y)
Center: (x + Δx, y + Δ y)

1. What should the radius of the circle be?

2. Given x, y, and θ, what's Δx and Δy?

3. How do we handle left, right, forward and backward?

   Hint:

$$\begin{aligned} \sin(\theta \pm \phi) &= \sin\theta\cos\phi \pm \cos\theta\sin\phi \\ \cos(\theta \pm \phi) &= \cos\theta\cos\phi \mp \sin\theta\sin\phi \end{aligned}$$

4. Suppose, to see the "big picture," I wanted to elevate on the Z-axis. What should I do with center? Is that easy to do?