TUTORIAL ON ISOLATING TRANSFORMATIONS OpenGL

Marcelo Walter - UFRGS

September, 2010

1 Composing Several Transformations

In this tutorial we will see how to combine several transformations to achieve a particular result. The example we will discuss is a solar system, in which objects need to rotate on their axes as well as in orbit around each other.

2 Building a Solar System

The program here described draws a simple solar system with one sun, two planets and a moon around one planet. You can get the executable and source code from the course page available via moodle.

All planets are drawn using the `glutWireSphere` call with appropriate scaling. This example was inspired in a similar one presented in the OpenGL Programming Guide. To use the program you can press the `LEFT ARROW` and `RIGHT ARROW` keys to revolve the planets around the sun (a year) and the `UP ARROW` and `DOWN ARROW` keys to rotate the planets around their own axes (a day). To rotate the moon around its axis use the `F1` and `F2` keys.

You can rotate the whole scene but only around the Y-axis pressing down the left mouse button and moving the mouse right or left. To exit the program press `Q`.

How would you structure your OpenGL code to achieve the functionality provided by the above mentioned program? Let's try first a simpler example: only the sun and one planet. Pseudo-code for displaying just the sun and one planet without including the axes would look something like:

```
pushMatrix();
   draw_sun();
   /* rotates "year" degrees around Y-axis */
   revolve_around_sun(year, 0.0, 1.0, 0.0);
   /* translates to draw the first planet */
   Translate( 2.0, 0.0, 0.0);
```

```
    /* rotates "day" degrees around Y-axis */
    Rotate(day, 0.0, 1.0, 0.0);
    /* draws first planet */
    draw_planet();
popMatrix();
```

Drawing the sun is straightforward, since it should be located at the origin of the world, fixed coordinate system, which is where the `draw_sun` routine places it. Thus, drawing the sun doesn't require translation. Drawing a planet rotating around the sun requires several modeling transformations. The planet needs to rotate about its own axis once a day. And once a year, the planet completes one revolution around the sun.

To determine the order of modeling transformations, we have to visualize what happens to the local coordinate system. An initial revolution (`revolve_around_sun(year, 0.0, 1.0, 0.0)`) rotates the local coordinate system around the Y-axis (`year` is a variable which holds a value between 1 and 365). This initial revolution actually determines where along the orbit the planet is (i.e., one day from the 365 possible ones). Next, a translation moves the local coordinate system to a position on the planet's orbit; the distance moved should equal the radius of the orbit (in this case 2.0). A second rotation rotates the local coordinate system around the local axes, thus determining the time of day for the planet. Once we've issued all these transformation commands, the planet can be drawn.

Let's say now that we want to improve our program to draw a second planet with a different orbit around the sun. The pseudo code below does that:

```
pushMatrix();
    draw_sun();
    /* rotates "year" degrees around Y-axis */
    revolve_around_sun(year, 0.0, 1.0, 0.0);
    pushMatrix();
        /* translates to draw the first planet */
        Translate( 2.0, 0.0, 0.0);
        /* rotates "day" degrees around Y-axis */
        Rotate(day, 0.0, 1.0, 0.0);
        /* draws first planet */
        draw_planet();
    popMatrix();

    pushMatrix();
        /* translates to draw the second planet */
        Translate( -2.5, 0.0, 0.0);
        /* rotates "day" degrees around Y-axis */
        Rotate(day, 0.0, 1.0, 0.0);
        /* draws another planet */
        draw_planet();
    popMatrix();
popMatrix();
```

We want to draw two planets positioned relative to the Sun's position. In order to do that we have to isolate the two drawings by saving and restoring the current transformation matrix between drawings. Once we do that, drawing the two planets demands exactly the same commands, a translation and a rotation, like we did in the first example. In the code above the two planets will have slightly different orbits (2.0 and -2.5) and will revolve the same amount around their own axes (expressed by the same variable `day`).

Finally, let's say we want to include a small moon revolving around the second planet. The final pseudo code including everything looks something like:

```
pushMatrix();
   draw_sun();
   /* rotates "year" degrees around Y-axis */
   revolve_around_sun(year, 0.0, 1.0, 0.0);
   pushMatrix();
      /* translates  to draw the first planet */
      Translate( 2.0, 0.0, 0.0);
      /* rotates "day" degrees around Y-axis */
      Rotate(day, 0.0, 1.0, 0.0);
      /* draws first planet */
      draw_planet();
   popMatrix();
   pushMatrix();
      /* translates to draw the second planet */
      Translate( -2.5, 0.0, 0.0);
      /* rotates "day" degrees around Y-axis */
      Rotate(day, 0.0, 1.0, 0.0);
/* draws second planet */
      draw_planet();
/* translates to draw the moon */
      Translate(1.0,0.0,0.0);
/* rotates "moonday" degrees around Y-axis */
      Rotate(moonday, 0.0, 1.0, 0.0);
      /* draws a moon */
      draw_moon();
   popMatrix();
popMatrix();
```

In order to draw the moon around the second planet we first translate to a position on the second planet's orbit and the rotation determines the time of the day for the moon (where the variable `moonday` expresses the time of the day for the moon).