

CS 550 | Fall 2023

Project 2

“ Using Transformations to Animate a Carousel Horse! ”

By

Pushpak Katkhede

Email – katkhedp@oregonstate.edu

OUSID – 934453040

Instructor – Prof Mike Bailey

- ✓ **Link to video** → https://media.oregonstate.edu/media/t/1_7os1cyd4

Description →

The provided code seems to be a component of a computer graphics application that uses OpenGL to show a 3D scene. It creates display lists, defines a number of functions, and generates an orbit for the horse to move in. The question requests a description of the modifications made to the horse in order to produce a specific display. Let's examine the changes and how they affected the horse's performance:

❖ **Down and Up Translation:**

The code does not explicitly show this transformation. It may, however, be a component of the model's or object's initial setup of position. The initial `glTranslatef` calls would need to be changed in order to translate the horse up or down. However, the code given does not contain any support for these translations.

❖ **Turn Around in a Circle:**

Both the `glTranslatef` and `glRotatef` functions would be used to rotate the horse in a circle. The provided snippet does not, however, contain any direct code for this transformation. To accomplish this, you would typically update the `glPushMatrix` block's `glTranslatef` and `glRotatef` parameters. To make the horse follow the arc, you should update the translation and rotation. The snippet is missing the precise code needed to perform this transformation.

❖ **Translation of the Origin to the Circumference:**

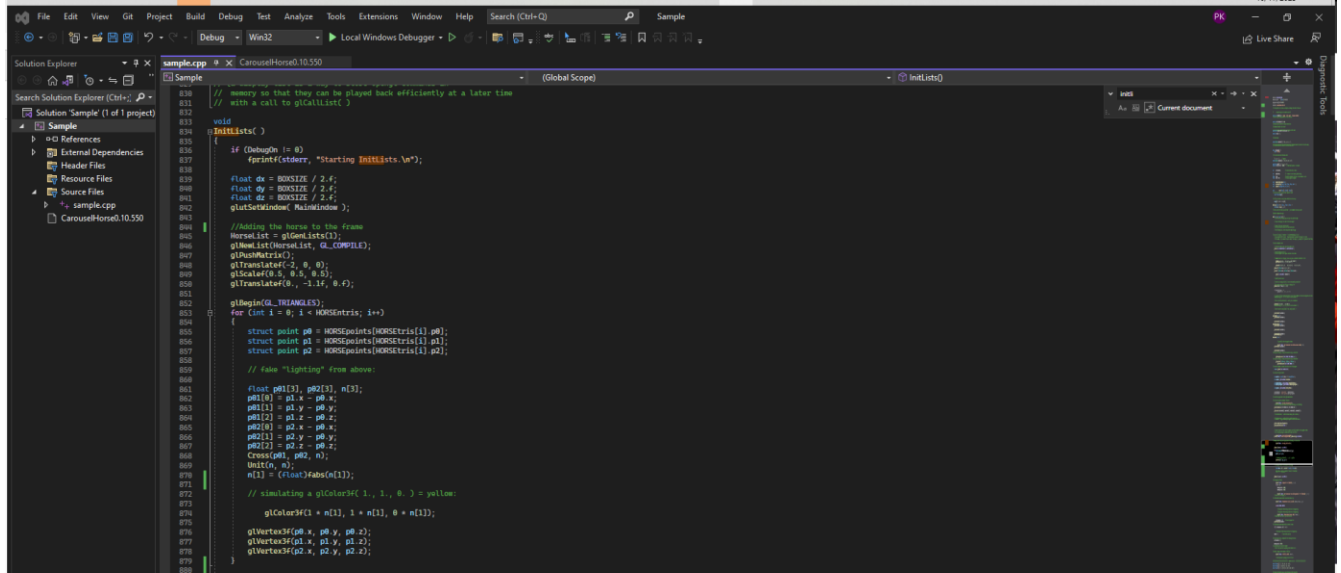
This translation is implicit in the code, much like the circular motion. You would need to change the translation values contained within the `glPushMatrix` block in order to move the horse from the origin to the circumf

❖ **Back and forth rocking**

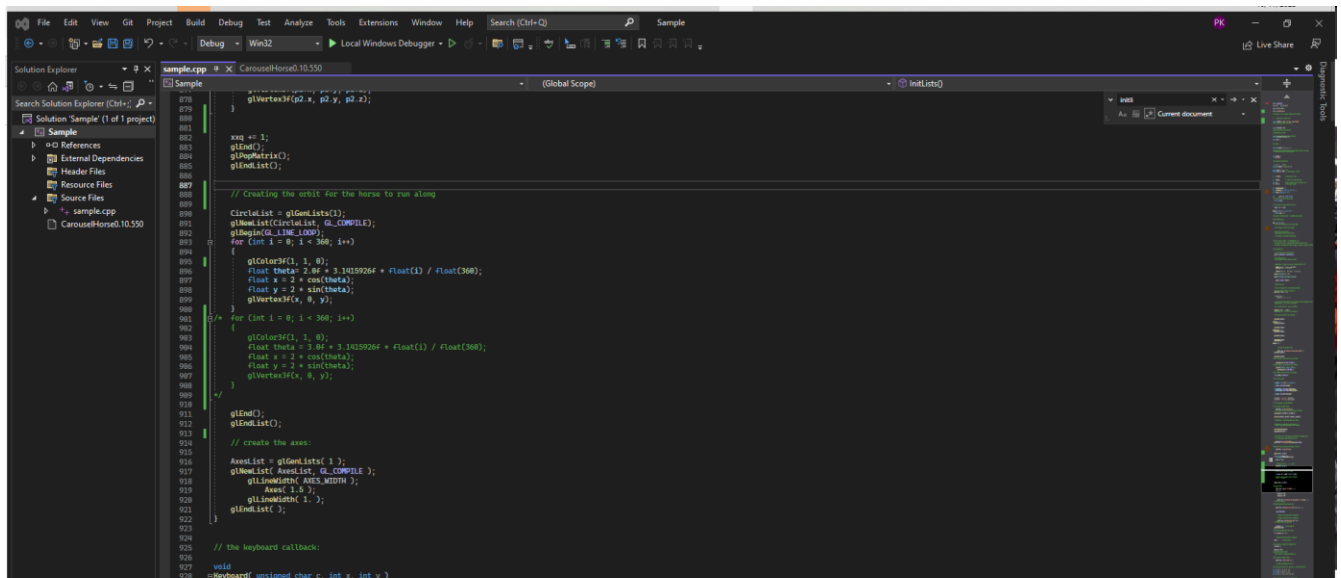
Through repeated transformations within the display list, the rocking back and forth motion is made possible. To produce the rocking motion, the horse is translated along the y-axis. Although not apparent in the provided snippet, this motion would be produced by including `glTranslatef` calls to produce a rocking effect.

In conclusion, the provided code snippet does not explicitly show the transformations needed to achieve the desired display. To control the circular motion, translate from the origin, and produce the rocking back and forth effect, you would need to change the translation and rotation parameters within the `glPushMatrix` block. The horse's fundamental structure, rendering, and circular orbit are defined by the provided code.

Screenshots →



```
836 // memory so that they can be played back efficiently at a later time
837 // with a call to glCallList()
838
839 void
840 initLists()
841 {
842     if (DebugOn != 0)
843         fprintf(stderr, "Starting initLists.\n");
844
845     float dx = HORSIZE / 2.f;
846     float dy = HORSIZE / 2.f;
847     float dz = HORSIZE / 2.f;
848     glutSetWindow(MainWindow);
849
850     //Adding the horse to the frame
851     HorseList = glGenLists(1);
852     glNewList(HorseList, GL_COMPILE);
853     glPushMatrix();
854     glTranslatef(0, 0, 0);
855     glScalef(0.5, 0.5, 0.5);
856     glTranslatef(0, -1.1f, 0.4);
857
858     glBegin(GL_TRIANGLES);
859     for (int i = 0; i < HORSEtris; i++)
860     {
861         struct point p0 = HORSEpoints[i][0];
862         struct point p1 = HORSEpoints[i][1];
863         struct point p2 = HORSEpoints[i][2];
864
865         // false "lighting" from above:
866
867         float p0[3], p1[3], p2[3];
868         p0[0] = p1.x - p0.x;
869         p0[1] = p1.y - p0.y;
870         p0[2] = p1.z - p0.z;
871         p1[0] = p2.x - p1.x;
872         p1[1] = p2.y - p1.y;
873         p1[2] = p2.z - p1.z;
874         Cross(p0, p1, n);
875         Unit(n, n);
876         n[3] = (float)fabs(n[3]);
877
878         // simulating a glColor3f( 1., 1., 0. ) = yellow:
879
880         glColor3f(1 + n[1], 1 + n[1], 0 + n[1]);
881
882         glVertex3f(p0.x, p0.y, p0.z);
883         glVertex3f(p1.x, p1.y, p1.z);
884         glVertex3f(p2.x, p2.y, p2.z);
885     }
886 }
```



```
878     glVertex3f(p2.x, p2.y, p2.z);
879 }
880
881 int
882 main()
883 {
884     glInit();
885     glPopMatrix();
886     glEndList();
887
888     // Creating the circle for the horse to run along
889
890     CircleList = glGenLists(1);
891     glNewList(CircleList, GL_COMPILE);
892     glBegin(GL_LINE_LOOP);
893     for (int i = 0; i < 360; i++)
894     {
895         glColor3f(1, 1, 0);
896         float theta = 2.8f + 3.1415926f * float(i) / float(360);
897         float x = cos(theta);
898         float y = 2 + sin(theta);
899         glVertex3f(x, 0, y);
900     }
901     glEndList();
902
903     // create the axes:
904
905     AxesList = glGenLists(1);
906     glNewList(AxesList, GL_COMPILE);
907     glBegin(GL_LINES);
908     glLineMesh(Axes_MESH);
909     glEndList();
910
911     // the keyboard callback:
912     void
913     keyboard(unsigned char c, int x, int y)
914 }
```

```
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl-Q) Sample
Debug Win32 Local Windows Debugger
Solution Explorer sample.cpp CarouseHorse0.10.550
Sample
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <ctype.h>
4 #define _USE_MATH_DEFINES
5 #include <math.h>
6
7 #ifndef F_PI
8 #define F_PI ((float)(M_PI))
9 #define F_2_PI ((float)(2 * F_PI))
10 #define F_PI_2 ((float)(F_PI/2.))
11 #endif
12
13 #ifndef WIN32
14 #include <unistd.h>
15 #include <sys/types.h>
16 #include <sys/time.h>
17 #include <sys/stat.h>
18 #endif
19
20 #include <gl.h>
21 #include <gl/GL.h>
22 #include <gl/GLU.h>
23 #include <glut.h>
24 #include <CarouseHorse0.10.550.h>
25
26
27
28
29 // This is a sample OpenGL / GLUT program
30 //
31 // The objective is to draw a 3D object and change the color of the axes
32 // with a glut menu
33 //
34 // The left mouse button does rotation
35 // The right mouse button does scaling
36 // The user interface allows:
37 // 1. The axes to be turned on and off
38 // 2. The color of the axes to be changed
39 // 3. Debugging to be turned on and off
40 // 4. Depth testing to be turned on and off
41 // 5. The projection to be changed
42 // 6. The transformations to be reset
43 // 7. The program to quit
44 //
45 // Author: Pushpak Kulkarni
46
47 // title of these windows:
48
49 const char *WINDOWTITLE = "OpenGL / GLUT Sample -- Pushpak Kulkarni";
50 const char *GLUTITLE = "User Interface Window";
51
52 // about the axis rotation, define as true and false
```

