**Moving Car:**

#include <iostream>

#include<GL/gl.h>

#include <GL/glut.h>

#include <math.h>

using namespace std;

float \_move = 0.0f;

float \_angle1=0.0f;

void wheel()

{

glLoadIdentity(); //Reset the drawing perspective

glMatrixMode(GL\_MODELVIEW);

glPushMatrix();

glTranslatef(\_move, 0.0f, 0.0f);

glTranslatef(0.4,0,0);

glRotatef(\_angle1, 0.0f, 0.0f,1.0f);

glBegin(GL\_LINES);// Draw a Red 1x1 Square centered at origin

for(int i=0;i<200;i++)

{

glColor3f(1.0,1.0,0.0);

float pi=3.1416;

float A=(i\*2\*pi)/200;

float r=0.065;

float x = r \* cos(A);

float y = r \* sin(A);

glVertex2f(x,y );

}

glEnd();

glPopMatrix();

glLoadIdentity(); //Reset the drawing perspective

glMatrixMode(GL\_MODELVIEW);

glPushMatrix();

glTranslatef(\_move, 0.0f, 0.0f);

glTranslatef(0.2,0,0);

glRotatef(\_angle1, 0.0f, 0.0f,1.0f);

glBegin(GL\_LINES);// Draw a Red 1x1 Square centered at origin

for(int i=0;i<200;i++)

{

glColor3f(1.0,1.0,0.0);

float pi=3.1416;

float A=(i\*2\*pi)/200;

float r=0.065;

float x = r \* cos(A);

float y = r \* sin(A);

glVertex2f(x,y );

}

glEnd();

glPopMatrix();

}

void drawScene() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3d(1,0,0);

glLoadIdentity(); //Reset the drawing perspective

glMatrixMode(GL\_MODELVIEW);

glPushMatrix();

glTranslatef(\_move, 0.0f, 0.0f);

glBegin(GL\_QUADS);

glVertex2f(0.1f, 0.0f);

glVertex2f(0.5f, 0.0f);

glVertex2f(0.5f, 0.2f);

glVertex2f(0.1f, 0.2);

glEnd();

wheel();

glPopMatrix();

glutSwapBuffers();

}

void update(int value) {

\_move += .02;

if(\_move > 1.3)

{

\_move = -1.0;

}

glutPostRedisplay();

glutTimerFunc(20, update, 0);

}

void update1(int value) {

\_angle1+=2.0f;

if(\_angle1 > 360.0)

{

\_angle1-=360;

}

glutPostRedisplay(); //Notify GLUT that the display has changed

glutTimerFunc(20, update1, 0); //Notify GLUT to call update again in 25 milliseconds

}

int main(int argc, char\*\* argv) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(800, 800);

glutCreateWindow("Transformation");

glutDisplayFunc(drawScene);

gluOrtho2D(-2,2,-2,2);

glutTimerFunc(20, update, 0); //Add a timer

glutTimerFunc(20, update1, 0); //Add a timer

glutMainLoop();

return 0;

}