Simulating Car Parking Experience

**CS352: Computer Graphics & Visualization Lab**

Submitted By – G14

Boddupalli Karthik - 200001016

Karri Trived - 200001033

M. Sai Akshay Reddy - 200001049

Code

#include <GL/glut.h>

#include <math.h>

#include <stdlib.h>

#include <initializer\_list>

#include<vector>

#include <limits>

#include<iostream>

using namespace std;

static float angle=0.0,ratio;

static float x=0.0f,y=1.75f,z=5.0f;

static float lx=0.10f,ly=0.10f,lz=-1.0f;

static GLint carr\_display\_list;

float theta=0.01;

int xxxx=0,yyyy=0,kk=0,movecarvar=-1;

int a[6]={55,97,44,152,55,171};

int b[6]={102,194,110,152,153};

int c[6]={159,243,133,253,233,228};

float fx[7],fz[7],fxincr[7],fzincr[7],temp[7];

vector<pair<float,float>>vertices[7];

void changeSize(int w, int h)

{

        if(h == 0) // Prevent a divide by zero, when window is too

                short h = 1;

        ratio = 1.0f \* w / h; // Reset the coordinate system before modifying

        glMatrixMode(GL\_PROJECTION);

        glLoadIdentity();

        glViewport(0, 0, w, h); // Set the viewport to be the entire window

        gluPerspective(45,ratio,1,1000);

        glMatrixMode(GL\_MODELVIEW);

        glLoadIdentity();

        gluLookAt(x, y, z,x + lx,y + ly,z + lz,0.0f,1.0f,0.0f);

}

void drawString(float x, float y, float z, char \*str) {

  // Save the current matrix

  glPushMatrix();

  // Translate to the appropriate starting point

  glTranslatef(x, y, z);

  glRotatef(-90,0,1,0);

  glScalef(0.001,0.001,0.001);

  // Note: We could change the line width with glLineWidth()

  // Render the characters

  for (char\* c = str; \*c != '\0'; c++) {

    glutStrokeCharacter(GLUT\_STROKE\_ROMAN, \*c);

  }

  // Another useful function

  //    int glutStrokeWidth(void \*font, int character);

  // Retrieve the original matrix

  glPopMatrix();

}

void drawcarr()

{

        glTranslatef(.0,0.8,0.0);

        glEnable(GL\_BLEND); //TRANCPARENCY1

        glBlendFunc(GL\_ONE, GL\_ZERO);//TRANCPARENCY2

        glBegin(GL\_LINE\_LOOP);

        glVertex3f(-1.12,-.48,0.7);//a

        glVertex3f(-0.86,-.48,0.7);//b

        glVertex3f(-.74,-0.2,0.7);//c

        glVertex3f(-.42,-.2,0.7);//d

        glVertex3f(-0.3,-.48,0.7);//e

        glVertex3f(.81,-0.48,0.7);//f

        glVertex3f(.94,-0.2,0.7);//g

        glVertex3f(1.24,-.2,0.7);//h

        glVertex3f(1.38,-.48,0.7);//i

        glVertex3f(1.52,-.44,0.7);//j

        glVertex3f(1.52,.14,0.7);//k

        glVertex3f(1.14,0.22,0.7);//l

        glVertex3f(0.76,.22,0.7);//m

        glVertex3f(.52,0.56,0.7);//n

        glVertex3f(-0.1,0.6,0.7);//0

        glVertex3f(-1.02,0.6,0.7);//p

        glVertex3f(-1.2,0.22,0.7);//q

        glVertex3f(-1.2,-.28,0.7);//r

        glEnd();

        glBegin(GL\_LINE\_LOOP);

        glVertex3f(-1.12,-.48,-0.7);//a'

        glVertex3f(-0.86,-.48,-0.7);//b'

        glVertex3f(-.74,-0.2,-0.7);//c'

        glVertex3f(-.42,-.2,-0.7);//d'

        glVertex3f(-0.3,-.48,-0.7);//e'

        glVertex3f(.81,-0.48,-0.7);//f'

        glVertex3f(.94,-0.2,-0.7);//g'

        glVertex3f(1.24,-.2,-0.7);//h'

        glVertex3f(1.38,-.48,-0.7);//i'

        glVertex3f(1.52,-.44,-0.7);//j'

        glVertex3f(1.52,.14,-0.7);//k'

        glVertex3f(1.14,0.22,-0.7);//l'

        glVertex3f(0.76,.22,-0.7);//m'

        glVertex3f(.52,0.56,-0.7);//n'

        glVertex3f(-0.1,0.6,-0.7);//o'

        glVertex3f(-1.02,0.6,-0.7);//p'

        glVertex3f(-1.2,0.22,-0.7);//q'

        glVertex3f(-1.2,-.28,-0.7);//r'

        glEnd();

        glBegin(GL\_LINES);

        glVertex3f(-1.12,-.48,0.7);//a

        glVertex3f(-1.12,-.48,-0.7);//a'

        glVertex3f(-0.86,-.48,0.7);//b

        glVertex3f(-0.86,-.48,-0.7);//b'

        glVertex3f(-.74,-0.2,0.7);//c

        glVertex3f(-.74,-0.2,-0.7);//c'

        glVertex3f(-.42,-.2,0.7);//d

        glVertex3f(-.42,-.2,-0.7);//d'

        glVertex3f(-0.3,-.48,0.7);//e

        glVertex3f(-0.3,-.48,-0.7);//e'

        glVertex3f(.81,-0.48,0.7);//f

        glVertex3f(.81,-0.48,-0.7);//f'

        glVertex3f(.94,-0.2,0.7);//g

        glVertex3f(.94,-0.2,-0.7);//g'

        glVertex3f(1.24,-.2,0.7);//h

        glVertex3f(1.24,-.2,-0.7);//h'

        glVertex3f(1.38,-.48,0.7);//i

        glVertex3f(1.38,-.48,-0.7);//i'

        glVertex3f(1.52,-.44,0.7);//j

        glVertex3f(1.52,-.44,-0.7);//j'

        glVertex3f(1.52,.14,0.7);//k

        glVertex3f(1.52,.14,-0.7);//k'

        glVertex3f(1.14,0.22,0.7);//l

        glVertex3f(1.14,0.22,-0.7);//l'

        glVertex3f(0.76,.22,0.7);//m

        glVertex3f(0.76,.22,-0.7);//m'

        glVertex3f(.52,0.56,0.7);//n

        glVertex3f(.52,0.56,-0.7);//n'

        glVertex3f(-0.1,0.6,0.7);//0

        glVertex3f(-0.1,0.6,-0.7);//o'

        glVertex3f(-1.02,0.6,0.7);//p

        glVertex3f(-1.02,0.6,-0.7);//p'

        glVertex3f(-1.2,0.22,0.7);//q

        glVertex3f(-1.2,0.22,-0.7);//q'

        glVertex3f(-1.2,-.28,0.7);//r

        glVertex3f(-1.2,-.28,-0.7);//r'

        glEnd();

        glBegin(GL\_POLYGON); // top filling

        glVertex3f(-0.1,0.6,0.7);//o

        glVertex3f(-0.1,0.6,-0.7);//o'

        glVertex3f(-1.02,0.6,-0.7);//p'

        glVertex3f(-1.02,0.6,0.7);//p

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-0.1,0.6,0.7);//o

        glVertex3f(-0.1,0.6,-0.7);//o'

        glVertex3f(.52,0.56,-0.7);//n'

        glVertex3f(.52,0.56,0.7);//n

        glEnd();

        glBegin(GL\_POLYGON); //back filling

        glVertex3f(-1.2,0.22,0.7);//q

        glVertex3f(-1.2,0.22,-0.7);//q'

        glVertex3f(-1.2,-.28,-0.7);//r'

        glVertex3f(-1.2,-.28,0.7);//r

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(1.52,.14,0.7);//k

        glVertex3f(1.14,0.22,0.7);//l

        glVertex3f(1.14,0.22,-0.7);//l'

        glVertex3f(1.52,.14,-0.7);//k'

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(0.76,.22,0.7);//m

        glVertex3f(0.76,.22,-0.7);//m'

        glVertex3f(1.14,0.22,-0.7);//l'

        glVertex3f(1.14,0.22,0.7);//l

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-1.12,-.48,0.7);//a

        glVertex3f(-0.86,-.48,0.7);//b

        glVertex3f(-.74,-0.2,0.7);//c

        glVertex3f(-0.64,0.22,0.7);//cc

        glVertex3f(-1.08,0.22,0.7);//dd

        glVertex3f(-1.2,0.22,0.7);//q

        glVertex3f(-1.2,-.28,0.7);//r

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-.74,-0.2,0.7);//c

        glVertex3f(-0.64,0.22,0.7);//cc

        glVertex3f(-0.5,0.22,0.7);//hh

        glVertex3f(-0.5,-0.2,0.7);//pp

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(0.0,0.22,0.7);//gg

        glVertex3f(1.14,0.22,0.7);//l

        glVertex3f(1.24,-.2,0.7);//h

        glVertex3f(0.0,-0.2,0.7);//oo

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-1.12,-.48,-0.7);//a'

        glVertex3f(-0.86,-.48,-0.7);//b'

        glVertex3f(-.74,-0.2,-0.7);//c'

        glVertex3f(-0.64,0.22,-0.7);//cc'

        glVertex3f(-1.08,0.22,-0.7);//dd'

        glVertex3f(-1.2,0.22,-0.7);//q'

        glVertex3f(-1.2,-.28,-0.7);//r'

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-.74,-0.2,-0.7);//c'

        glVertex3f(-0.64,0.22,-0.7);//cc'

        glVertex3f(-0.5,0.22,-0.7);//hh'

        glVertex3f(-0.5,-0.2,-0.7);//pp'

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(0.0,0.22,-0.7);//gg'

        glVertex3f(1.14,0.22,-0.7);//l'

        glVertex3f(1.24,-.2,-0.7);//h'

        glVertex3f(0.0,-0.2,-0.7);//oo'

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-1.2,0.22,0.7);//q

        glVertex3f(-1.08,0.22,0.7);//dd

        glVertex3f(-0.98,0.5,0.7);//aa

        glVertex3f(-1.02,0.6,0.7);//p

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-1.02,0.6,0.7);//p

        glVertex3f(-0.98,0.5,0.7);//aa

        glVertex3f(0.44,0.5,0.7);//jj

        glVertex3f(.52,0.56,0.7);//n

        glVertex3f(-0.1,0.6,0.7);//0

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-0.64,0.5,0.7);//bb

        glVertex3f(-0.64,0.22,0.7);//cc

        glVertex3f(-0.5,0.22,0.7);//hh

        glVertex3f(-0.5,0.5,0.7);//ee

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(0.0,0.5,0.7);//ff

        glVertex3f(0.0,0.22,0.7);//gg

        glVertex3f(0.12,0.22,0.7);//ll

        glVertex3f(0.12,0.5,0.7);//ii

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(.52,0.56,0.7);//n

        glVertex3f(0.44,0.5,0.7);//jj

        glVertex3f(0.62,0.22,0.7);//kk

        glVertex3f(0.76,.22,0.7);//m

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-.42,-.2,0.7);//d

        glVertex3f(.94,-0.2,0.7);//g

        glVertex3f(.81,-0.48,0.7);//f

        glVertex3f(-0.3,-.48,0.7);//e

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(1.14,0.22,0.7);//l

        glVertex3f(1.52,.14,0.7);//k

        glVertex3f(1.52,-.44,0.7);//j

        glVertex3f(1.38,-.48,0.7);//i

        glVertex3f(1.24,-.2,0.7);//h

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-1.2,0.22,-0.7);//q'

        glVertex3f(-1.08,0.22,-0.7);//dd'

        glVertex3f(-0.98,0.5,-0.7);//aa'

        glVertex3f(-1.02,0.6,-0.7);//p'

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-1.02,0.6,-0.7);//p'

        glVertex3f(-0.98,0.5,-0.7);//aa'

        glVertex3f(0.44,0.5,-0.7);//jj'

        glVertex3f(.52,0.56,-0.7);//n'

        glVertex3f(-0.1,0.6,-0.7);//0'

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-0.64,0.5,-0.7);//bb'

        glVertex3f(-0.64,0.22,-0.7);//cc'

        glVertex3f(-0.5,0.22,-0.7);//hh'

        glVertex3f(-0.5,0.5,-0.7);//ee'

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(0.0,0.5,-0.7);//ff'

        glVertex3f(0.0,0.22,-0.7);//gg'

        glVertex3f(0.12,0.22,-0.7);//ll'

        glVertex3f(0.12,0.5,-0.7);//ii'

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(.52,0.56,-0.7);//n'

        glVertex3f(0.44,0.5,-0.7);//jj'

        glVertex3f(0.62,0.22,-0.7);//kk'

        glVertex3f(0.76,.22,-0.7);//m'

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-.42,-.2,-0.7);//d'

        glVertex3f(.94,-0.2,-0.7);//g'

        glVertex3f(.81,-0.48,-0.7);//f'

        glVertex3f(-0.3,-.48,-0.7);//e'

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(1.14,0.22,-0.7);//l'

        glVertex3f(1.52,.14,-0.7);//k'

        glVertex3f(1.52,-.44,-0.7);//j'

        glVertex3f(1.38,-.48,-0.7);//i'

        glVertex3f(1.24,-.2,-0.7);//h'

        glEnd();

        glBegin(GL\_POLYGON); // door1 body- rear, near

        glVertex3f(-0.5,0.22,0.7);//hh

        glVertex3f(0.0,0.22,0.7);//gg

        glVertex3f(0.0,-0.2,0.7);//oo

        glVertex3f(-0.5,-0.2,0.7);//pp

        glEnd();

        glBegin(GL\_POLYGON); // door body- rear, far

        glVertex3f(-0.5,0.22,-0.7);//hh'

        glVertex3f(0.0,0.22,-0.7);//gg'

        glVertex3f(0.0,-0.2,-0.7);//oo'

        glVertex3f(-0.5,-0.2,-0.7);//pp'

        glEnd();

        glBegin(GL\_POLYGON); // door2 body- near, driver

        glVertex3f(0.12,0.22,0.7);//ll

        glVertex3f(0.62,0.22,0.7);//kk

        glVertex3f(0.62,-0.2,0.7);//mm

        glVertex3f(0.12,-0.2,0.7);//nn

        glEnd();

        glBegin(GL\_POLYGON); // door2 body- far, driver

        glVertex3f(0.12,0.22,-0.7);//ll'

        glVertex3f(0.62,0.22,-0.7);//kk'

        glVertex3f(0.62,-0.2,-0.7);//mm'

        glVertex3f(0.12,-0.2,-0.7);//nn'

        glEnd();

        glBegin(GL\_POLYGON);//front\*\*

        glVertex3f(1.52,.14,0.7);//k

        glVertex3f(1.52,.14,-0.7);//k'

        glVertex3f(1.52,-.44,-0.7);//j'

        glVertex3f(1.52,-.44,0.7);//j

        glEnd();

        glTranslatef(-.58,-.52,0.7); //translate to 1st tyre

        glColor3f(0.09,0.09,0.09); // tyre color\*\*\*\*\*\*\*\*

        glutSolidTorus(0.12f, .14f, 10, 25);

        glTranslatef(1.68,0.0,0.0); //translate to 2nd tyre

        glutSolidTorus(0.12f, .14f, 10, 25);

        glTranslatef(0.0,0.0,-1.4); //translate to 3rd tyre

        glutSolidTorus(0.12f, .14f, 10, 25);

        glTranslatef(-1.68,0.0,0.0); //translate to 4th tyre which is behind 1st tyrerearback

        glutSolidTorus(0.12f, .14f, 10, 25);

        glTranslatef(.58,.52,0.7); //translate to origin

        glRotatef(90.0,0.0,1.0,0.0);

        glTranslatef(0.0,0.0,-1.40);

        glutSolidTorus(0.2f, .2f, 10, 25);

        glTranslatef(0.0,0.0,1.40);

        glRotatef(270.0,0.0,1.0,0.0);

        glBegin(GL\_POLYGON); //bottom filling

        glColor3f(0.25,0.25,0.25);

        glVertex3f(-0.3,-.48,0.7);//e

        glVertex3f(-0.3,-.48,-0.7);//e'

        glVertex3f(.81,-0.48,-0.7);//f'

        glVertex3f(.81,-0.48,0.7);//f

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-.42,-.2,0.7);//d

        glVertex3f(-.42,-.2,-0.7);//d'

        glVertex3f(-0.3,-.48,-0.7);//e'

        glVertex3f(-0.3,-.48,0.7);//e

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-1.2,-.28,0.7);//r

        glVertex3f(-1.2,-.28,-0.7);//r'

        glVertex3f(-1.12,-.48,-0.7);//a'

        glVertex3f(-1.12,-.48,0.7);//a

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-1.12,-.48,0.7);//a

        glVertex3f(-1.12,-.48,-0.7);//a'

        glVertex3f(-0.86,-.48,-0.7);//b'

        glVertex3f(-0.86,-.48,0.7);//b

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-0.86,-.48,0.7);//b

        glVertex3f(-0.86,-.48,-0.7);//b'

        glVertex3f(-.74,-0.2,-0.7);//c'

        glVertex3f(-.74,-0.2,0.7);//c

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(-.74,-0.2,0.7);//c

        glVertex3f(-.74,-0.2,-0.7);//c'

        glVertex3f(-.42,-.2,-0.7);//d'

        glVertex3f(-.42,-.2,0.7);//d

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(.81,-0.48,0.7);//f

        glVertex3f(.81,-0.48,-0.7);//f'

        glVertex3f(.94,-0.2,-0.7);//g'

        glVertex3f(.94,-0.2,0.7);//g

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(.94,-0.2,0.7);//g

        glVertex3f(.94,-0.2,-0.7);//g'

        glVertex3f(1.24,-.2,-0.7);//h'

        glVertex3f(1.24,-.2,0.7);//h

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(1.24,-.2,0.7);//h

        glVertex3f(1.24,-.2,-0.7);//h'

        glVertex3f(1.38,-.48,-0.7);//i'

        glVertex3f(1.38,-.48,0.7);//i

        glEnd();

        glBegin(GL\_POLYGON);

        glVertex3f(1.38,-.48,0.7);//i

        glVertex3f(1.38,-.48,-0.7);//i'

        glVertex3f(1.52,-.44,-0.7);//j'

        glVertex3f(1.52,-.44,0.7);//j

        glEnd();

        glBegin(GL\_LINE\_LOOP); // door outline- rear, front

        glColor3f(1.0,1.0,1.0);

        glVertex3f(-0.5,0.22,0.7);//hh

        glVertex3f(0.0,0.22,0.7);//gg

        glVertex3f(0.0,-0.2,0.7);//oo

        glVertex3f(-0.5,-0.2,0.7);//pp

        glEnd();

        glBegin(GL\_LINE\_LOOP); // door2 outline- near, driver

        glVertex3f(0.12,0.22,0.7);//ll

        glVertex3f(0.62,0.22,0.7);//kk

        glVertex3f(0.62,-0.2,0.7);//mm

        glVertex3f(0.12,-0.2,0.7);//nn

        glEnd();

        glColor3f(0.0,0.0,0.0);

        glBegin(GL\_LINE\_LOOP); // door2 outline- far, driver

        glVertex3f(0.12,0.22,-0.7);//ll'

        glVertex3f(0.62,0.22,-0.7);//kk'

        glVertex3f(0.62,-0.2,-0.7);//mm'

        glVertex3f(0.12,-0.2,-0.7);//nn'

        glEnd();

        glBegin(GL\_LINE\_LOOP); // door outline- rear, far

        glVertex3f(-0.5,0.22,-0.7);//hh'

        glVertex3f(0.0,0.22,-0.7);//gg'

        glVertex3f(0.0,-0.2,-0.7);//oo'

        glVertex3f(-0.5,-0.2,-0.7);//pp'

        glEnd();

        glBegin(GL\_POLYGON); //front\*\*

        glVertex3f(1.52,.14,0.7);//k

        glVertex3f(1.52,.14,-0.7);//k'

        glVertex3f(1.52,-.44,-0.7);//j'

        glVertex3f(1.52,-.44,0.7);//j

        glEnd();

        glColor3f(0.0,0.0,1.0);

        // transparent objects are placed next ..

        glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA); //TRANCPARENCY3

        //windscreen

        glBegin(GL\_POLYGON);

        glColor4f(0.0,0.0,0.0,0.7); //COLOR =WHITE TRANSPARENT

        glVertex3f(0.562,.5,.6);//AAA

        glVertex3f(.562,.5,-.6);//AAA'

        glVertex3f(.76,.22,-.6);//MMM'

        glVertex3f(.76,.22,.6);//MMM

        glEnd();

        glBegin(GL\_POLYGON); //rear window

        //COLOR =WHITE TRANSPARENT

        glVertex3f(-1.068,0.5,0.6);//pp

        glVertex3f(-1.068,0.5,-0.6);//pp'

        glVertex3f(-1.2,0.22,-0.6);//qq'

        glVertex3f(-1.2,0.22,0.6);//qq

        glEnd();

        glBegin(GL\_POLYGON); //leftmost window front

        glVertex3f(-0.98,0.5,0.7);//aa

        glVertex3f(-0.64,0.5,0.7);//bb

        glVertex3f(-0.64,0.22,0.7);//cc

        glVertex3f(-1.08,0.22,0.7);//dd

        glEnd();

        glBegin(GL\_POLYGON); //leftmost window back

        glVertex3f(-0.98,0.5,-0.7);//aa

        glVertex3f(-0.64,0.5,-0.7);//bb

        glVertex3f(-0.64,0.22,-0.7);//cc

        glVertex3f(-1.08,0.22,-0.7);//dd

        glEnd();

        glBegin(GL\_POLYGON); //middle window front

        glVertex3f(-0.5,0.5,0.7);

        glVertex3f(0.0,0.5,0.7);

        glVertex3f(0.0,0.22,0.7);

        glVertex3f(-0.5,0.22,0.7);

        glEnd();

        glBegin(GL\_POLYGON); //middle window back

        glVertex3f(-0.5,0.5,-0.7);

        glVertex3f(0.0,0.5,-0.7);

        glVertex3f(0.0,0.22,-0.7);

        glVertex3f(-0.5,0.22,-0.7);

        glEnd();

        glBegin(GL\_POLYGON); //rightmost window front

        glVertex3f(0.12,0.5,0.7);//ii

        glVertex3f(0.44,0.5,0.7);//jj

        glVertex3f(0.62,0.22,0.7);//kk

        glVertex3f(0.12,0.22,0.7);//ll

        glEnd();

        glBegin(GL\_POLYGON); //rightmost window back

        glVertex3f(0.12,0.5,-0.7);//ii'

        glVertex3f(0.44,0.5,-0.7);//jj'

        glVertex3f(0.62,0.22,-0.7);//kk'

        glVertex3f(0.12,0.22,-0.7);//ll'

        glEnd();

        glColor3f(0.0,0.0,1.0);

}

float dot\_product(pair<float,float>&a,pair<float,float>&b){

    return a.first\*b.first + a.second\*b.second;

}

float dxi = -1.22\*3,dxa = 1.54\*3,dzi = -0.72\*3,dza = 0.72\*3;

bool cars\_intersecting(vector<pair<float,float>>& a, vector<pair<float,float>>& b)

{

    for(auto i = 0; i < 2; i++) {

        auto current = a[i];

        auto next = a[(i + 1) % a.size()];

        pair<float,float>axis = {(current.second - next.second), (next.first - current.first)};

        float inf = numeric\_limits<float>::infinity();

        auto aMaxProj = -inf;

        auto aMinProj = inf;

        auto bMaxProj = -inf;

        auto bMinProj = inf;

        for(auto& v : a) {

            auto proj = dot\_product(axis, v);

            if(proj < aMinProj) aMinProj = proj;

            if(proj > aMaxProj) aMaxProj = proj;

        }

        for(auto& v : b) {

            auto proj = dot\_product(axis, v);

            if(proj < bMinProj) bMinProj = proj;

            if(proj > bMaxProj) bMaxProj = proj;

        }

        if(aMaxProj < bMinProj or aMinProj > bMaxProj) {

            return false;

        }

    }

    return true;

}

void movecar(int key, int x, int y)

{

float fxincrnew = fxincr[movecarvar-1],fzincrnew = fzincr[movecarvar-1],fxnew = fx[movecarvar-1],fznew = fz[movecarvar-1];

switch (key)

{

case GLUT\_KEY\_LEFT :temp[movecarvar-1]=fxincr[movecarvar-1];

 fxincrnew=fxincr[movecarvar-1]\*cos(theta)+fzincr[movecarvar-1]\*sin(theta);

 fzincrnew=-temp[movecarvar-1]\*sin(theta)+fzincr[movecarvar-1]\*cos(theta);

 fxnew+=fxincr[movecarvar-1];

 fznew+=fzincr[movecarvar-1];

 break;

case GLUT\_KEY\_RIGHT :temp[movecarvar-1]=fxincr[movecarvar-1];

 fxincrnew=fxincr[movecarvar-1]\*cos(-theta)+fzincr[movecarvar-1]\*sin(-

theta);

 fzincrnew=-temp[movecarvar-1]\*sin(-theta)+fzincr[movecarvar-1]\*cos(-theta);

 fxnew+=fxincr[movecarvar-1];

 fznew+=fzincr[movecarvar-1];

 break;

case GLUT\_KEY\_UP :fxnew+=fxincr[movecarvar-1];

 fznew+=fzincr[movecarvar-1];break;

case GLUT\_KEY\_DOWN :fxnew-=fxincr[movecarvar-1];

 fznew-=fzincr[movecarvar-1]; break;

}

float theta1 = 0;

if(fxincrnew!=0)

theta1=(atan(fzincrnew/fxincrnew)\*180)/3.141;

else if(fzincrnew>0)

theta1=-90.0;

else theta1=90.0;

if(fxincrnew>0&&fzincrnew<0)

{

theta1=-theta1;

}

else if(fxincrnew<0&&fzincrnew<0)

{

theta1=180-theta1;

}

else if(fxincrnew<0&&fzincrnew>0)

{

theta1=-180-theta1;

}else if(fxincrnew>0&&fzincrnew>0)

{

theta1=-theta1;

}

vector<pair<float,float>>vertice;

for(auto x : {dxi,dxa}){

    for(auto z : {dzi,dza}){

        vertice.push\_back({x,z});

    }

}

for(auto &x : vertice){

    float tx = x.first,tz = x.second;

    x.first = tx\*cos(theta1) + tz\*sin(theta1);

    x.second = tx\*sin(-theta1) + tz\*cos(theta1);

    x.first += fxnew;

    x.second += fznew;

}

for(auto vert : vertice){

    float sx = 20,sy = 5;

    if(vert.first <= -100 or vert.first >= 100 or vert.second <= -100 or (vert.second >= 100 and vert.first <= -91) or (vert.second >= 100 and vert.first >= -69))

        return void(cout << "Collided with fence" << endl);

    for(auto x : {-75.0f,0.0f,75.0f}){

        for(auto y : {-48.0f,0.0f,48.0f}){

            if((vert.first >= x - sx and vert.first <= x + sx) and (vert.second >= y - sy and vert.second <= y + sy))

                return void(cout << "Collided" << endl);

        }

    }

}

for(int i = 0; i < 6; i++){

    if(i + 1 == movecarvar)continue;

    if(cars\_intersecting(vertice,vertices[i])){

        return void(cout << "Collided" << " " << i << endl);

    }

}

vertices[movecarvar - 1] = vertice;

switch (key)

{

case GLUT\_KEY\_LEFT :temp[movecarvar-1]=fxincr[movecarvar-1];

 fxincr[movecarvar-1]=fxincr[movecarvar - 1]\*cos(theta)+fzincr[movecarvar - 1]\*sin(theta);

 fzincr[movecarvar - 1] =-temp[movecarvar - 1]\*sin(theta)+fzincr[movecarvar - 1]\*cos(theta);

 fx[movecarvar - 1] +=fxincr[movecarvar - 1] ;

 fz[movecarvar - 1] +=fzincr[movecarvar - 1] ;

 break;

case GLUT\_KEY\_RIGHT :temp[movecarvar - 1] =fxincr[movecarvar-1];

 fxincr[movecarvar - 1] =fxincr[movecarvar - 1] \*cos(-theta)+fzincr[movecarvar - 1] \*sin(-

theta);

 fzincr[movecarvar - 1] =-temp[movecarvar - 1] \*sin(-theta)+fzincr[movecarvar - 1] \*cos(-theta);

 fx[movecarvar - 1] +=fxincr[movecarvar - 1] ;

 fz[movecarvar - 1] +=fzincr[movecarvar - 1] ;

 break;

case GLUT\_KEY\_UP :fx[movecarvar - 1] +=fxincr[movecarvar - 1] ;

 fz[movecarvar - 1] +=fzincr[movecarvar - 1] ;break;

case GLUT\_KEY\_DOWN :fx[movecarvar - 1] -=fxincr[movecarvar - 1] ;

 fz[movecarvar - 1] -=fzincr[movecarvar - 1] ; break;

}

glutPostRedisplay();

}

int light\_mode = 1;

float col[10][3];

class car{

public:

int id;

float theta1;

 car(int iD){

     id = iD;

     fx[id]=-5,fz[id]=35,fxincr[id]=0.1,fzincr[id]=0,theta1=0,temp[id]=0;

 }

//float fx=-10,fz=80,fxincr=0.1,fzincr=0,theta1=0,temp=0;

void showcar(){

        if(fxincr[id]!=0)theta1=(atan(fzincr[id]/fxincr[id])\*180)/3.141;

        else if(fzincr[id]>0)theta1=-90.0;

        else theta1=90.0;

        if(fxincr[id]>0&&fzincr[id]<0)

        {

            theta1=-theta1;

        }

        else if(fxincr[id]<0&&fzincr[id]<0)

        {

            theta1=180-theta1;

        }

        else if(fxincr[id]<0&&fzincr[id]>0)

        {

            theta1=-180-theta1;

        }else if(fxincr[id]>0&&fzincr[id]>0)

        {

            theta1=-theta1;

        }

        glPushMatrix();

        glTranslatef(fx[id],0,fz[id]);

        glRotatef(theta1,0,1,0);

        glScalef(3,3,3);

        glColor3f(col[id][0],col[id][1],col[id][2]);

        glCallList(carr\_display\_list);

        glPopMatrix();

}

};

vector<car>cars;

GLuint createDL()

{

        GLuint carrDL;

        carrDL = glGenLists(1); // Create the id for the list

        glNewList(carrDL,GL\_COMPILE); // start list

        drawcarr(); // call the function that contains the rendering commands

        glColor3f(0,0,0);

        drawString(-1.21,0.09,0.4,"Audi");

        glEndList(); // endList

        return(carrDL);

}

void initialize(){

    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

    GLfloat globalAmbient[] = {0.9, 0.9, 0.9, 1.0};

    glEnable(GL\_DEPTH\_TEST);

    glEnable(GL\_LIGHTING);

    glShadeModel(GL\_SMOOTH);

    glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, globalAmbient);

    glColorMaterial(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE);

    GLfloat L\_Ambient[] = {1, 1, 1, 1.0};

    GLfloat L\_Diffuse[]= {0.0, 0.70, 0.0, 0.0};

    GLfloat L1\_postion[] = {0,290,0, 1.0};

    GLfloat L2\_postion[] = {0,290,0, 1.0};

    GLfloat L3\_postion[] = {0,290,0, 1.0};

    GLfloat L4\_postion[] = {0,290,0, 1.0};

    GLfloat L5\_postion[] = {0,290,0, 1.0};

    GLfloat L6\_postion[] = {0,290,0, 1.0};

    GLfloat L7\_postion[] = {0,290,0, 1.0};

    glLightfv(GL\_LIGHT1, GL\_AMBIENT, L\_Ambient);

    glLightfv(GL\_LIGHT1, GL\_DIFFUSE, L\_Diffuse);

    glLightfv(GL\_LIGHT1, GL\_POSITION, L1\_postion);

    glLightf(GL\_LIGHT1, GL\_CONSTANT\_ATTENUATION, 0.750);

    glLightf(GL\_LIGHT1, GL\_LINEAR\_ATTENUATION, 0.750);

    glLightf(GL\_LIGHT1, GL\_QUADRATIC\_ATTENUATION, 0.750);

    glLightfv(GL\_LIGHT2, GL\_AMBIENT, L\_Ambient);

    glLightfv(GL\_LIGHT2, GL\_DIFFUSE, L\_Diffuse);

    glLightfv(GL\_LIGHT2, GL\_POSITION, L2\_postion);

    glLightf(GL\_LIGHT2, GL\_CONSTANT\_ATTENUATION, 0.750);

    glLightf(GL\_LIGHT2, GL\_LINEAR\_ATTENUATION, 0.750);

    glLightf(GL\_LIGHT2, GL\_QUADRATIC\_ATTENUATION, 0.750);

    glLightfv(GL\_LIGHT3, GL\_AMBIENT, L\_Ambient);

    glLightfv(GL\_LIGHT3, GL\_DIFFUSE, L\_Diffuse);

    glLightfv(GL\_LIGHT3, GL\_POSITION, L3\_postion);

    glLightf(GL\_LIGHT3, GL\_CONSTANT\_ATTENUATION, 0.750);

    glLightf(GL\_LIGHT3, GL\_LINEAR\_ATTENUATION, 0.750);

    glLightf(GL\_LIGHT3, GL\_QUADRATIC\_ATTENUATION, 0.750);

    glLightfv(GL\_LIGHT4, GL\_AMBIENT, L\_Ambient);

    glLightfv(GL\_LIGHT4, GL\_DIFFUSE, L\_Diffuse);

    glLightfv(GL\_LIGHT4, GL\_POSITION, L4\_postion);

    glLightf(GL\_LIGHT4, GL\_CONSTANT\_ATTENUATION, 0.750);

    glLightf(GL\_LIGHT4, GL\_LINEAR\_ATTENUATION, 0.750);

    glLightf(GL\_LIGHT4, GL\_QUADRATIC\_ATTENUATION, 0.750);

    glLightfv(GL\_LIGHT5, GL\_AMBIENT, L\_Ambient);

    glLightfv(GL\_LIGHT5, GL\_DIFFUSE, L\_Diffuse);

    glLightfv(GL\_LIGHT5, GL\_POSITION, L5\_postion);

    glLightf(GL\_LIGHT5, GL\_CONSTANT\_ATTENUATION, 0.750);

    glLightf(GL\_LIGHT5, GL\_LINEAR\_ATTENUATION, 0.750);

    glLightf(GL\_LIGHT5, GL\_QUADRATIC\_ATTENUATION, 0.750);

    glLightfv(GL\_LIGHT6, GL\_AMBIENT, L\_Ambient);

    glLightfv(GL\_LIGHT6, GL\_DIFFUSE, L\_Diffuse);

    glLightfv(GL\_LIGHT6, GL\_POSITION, L4\_postion);

    glLightf(GL\_LIGHT6, GL\_CONSTANT\_ATTENUATION, 0.750);

    glLightf(GL\_LIGHT6, GL\_LINEAR\_ATTENUATION, 0.750);

    glLightf(GL\_LIGHT6, GL\_QUADRATIC\_ATTENUATION, 0.750);

    glLightfv(GL\_LIGHT7, GL\_AMBIENT, L\_Ambient);

    glLightfv(GL\_LIGHT7, GL\_DIFFUSE, L\_Diffuse);

    glLightfv(GL\_LIGHT7, GL\_POSITION, L5\_postion);

    glLightf(GL\_LIGHT7, GL\_CONSTANT\_ATTENUATION, 0.750);

    glLightf(GL\_LIGHT7, GL\_LINEAR\_ATTENUATION, 0.750);

    glLightf(GL\_LIGHT7, GL\_QUADRATIC\_ATTENUATION, 0.750);

    GLfloat specularReflectance[] = {1.0, 1.0, 1.0, 1.0};

    glMaterialfv(GL\_FRONT, GL\_SPECULAR, specularReflectance);

    glMateriali(GL\_FRONT, GL\_SHININESS, 75);

    glEnable(GL\_COLOR\_MATERIAL);

    glEnable(GL\_LIGHT0);

    glEnable(GL\_LIGHT1);

    glEnable(GL\_LIGHT2);

    glEnable(GL\_LIGHT3);

    glEnable(GL\_LIGHT4);

    glEnable(GL\_LIGHT5);

    glEnable(GL\_LIGHT6);

    glEnable(GL\_LIGHT7);

}

void initScene()

{

        glEnable(GL\_DEPTH\_TEST);

        carr\_display\_list = createDL();

        col[0][0] = 0.8,col[0][1] = 0.8,col[0][2] = 0;

        col[1][0] = 0,col[1][1] = 0.8,col[1][2] = 0.8;

        col[2][0] = 0.8,col[2][1] = 0,col[2][2] = 0.8;

        col[3][0] = 0.2,col[3][1] = 0.4,col[3][2] = 0.6;

        col[4][0] = 1.0,col[4][1] = 0,col[4][2] = 0;

        col[5][0] = 228.0/255,col[5][1] = 120.0/255,col[5][2] = 69.0/255;

        col[6][0] = 0,col[6][1] = 0.8, col[6][2] = 0;

        cars.push\_back(car(0));

        cars.push\_back(car(1));

        cars.push\_back(car(2));

        cars.push\_back(car(3));

        cars.push\_back(car(4));

        cars.push\_back(car(5));

        cars.push\_back(car(6));

        fx[1] += 65;

        fx[2] += 20,fz[2] -= 25;

        fx[3] += 0,fz[3] += 40;

        fx[4] -= 54;

        fxincr[4] = 0, fzincr[4] = 0.001;

        fx[5] -= 64, fz[5] -= 95;

        //fxincr[5] = 0, fzincr[5] = 0.001;

        fx[6] += 64, fz[6] -= 95;

        fxincr[6] = 0, fzincr[6] = 0.001;

        for(int i = 0; i < 7; i++){

            for(auto x : {dxi,dxa}){

                for(auto z : {dzi,dza}){

                    if(i == 4 or i == 6) vertices[i].push\_back({z + fx[i],-x + fz[i]});

                    else vertices[i].push\_back({x + fx[i],z + fz[i]});

                }

            }

        }

}

void renderScene(void)

{

    glEnable (GL\_POLYGON\_SMOOTH);

    glEnable (GL\_BLEND);

    glBlendFunc (GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

    glHint (GL\_POLYGON\_SMOOTH\_HINT, GL\_DONT\_CARE);

    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

    if(light\_mode)

             glClearColor(.7,0.85,1.0,1.0);

        else

             glClearColor(.25,.25,.25,1);

    glColor3f(0.15f, 0.15f, 0.15f); // Draw ground

    glBegin(GL\_QUADS);

        glVertex3f(-100.0f, 0.0f, -100.0f);

        glVertex3f(-100.0f, 0.0f, 100.0f);

        glVertex3f( 100.0f, 0.0f, 100.0f);

        glVertex3f( 100.0f, 0.0f, -100.0f);

    glEnd();

    glColor3f(0.5f, 0.15f, 0.15f);

    glBegin(GL\_QUADS);

        glVertex3f(-100.0f, 0.0f, 100.0f);

        glVertex3f(-100.0f, 2.2f, 100.0f);

        glVertex3f( -90.0f, 2.2f, 100.0f);

        glVertex3f( -90.0f, 0.0f, 100.0f);

        glColor3f(0.568f, 0.294f, 0.0f);

        glVertex3f(-90.5f,0.0f,100.0f);

        glVertex3f(-90.5,10.0f,100.0f);

        glVertex3f(-89.5f,10.0f,100.0f);

        glVertex3f(-89.5f,0.0f,100.0f);

        glVertex3f(-70.5f,0.0f,100.0f);

        glVertex3f(-70.5,10.0f,100.0f);

        glVertex3f(-69.5f,10.0f,100.0f);

        glVertex3f(-69.5f,0.0f,100.0f);

        glVertex3f(-90.0f,8.0f,100.0f);

        glVertex3f(-90.0,10.0f,100.0f);

        glVertex3f(-70.0f,10.0f,100.0f);

        glVertex3f(-70.0,8.0f,100.0f);

        glColor3f(0.5f, 0.15f, 0.15f);

        glVertex3f(-70.0f, 0.0f, 100.0f);

        glVertex3f(-70.0f, 2.2f, 100.0f);

        glVertex3f( 100.0f, 2.2f, 100.0f);

        glVertex3f( 100.0f, 0.0f, 100.0f);

        glVertex3f(-100.0f, 0.0f, -100.0f);

        glVertex3f(-100.0f, 2.2f, -100.0f);

        glVertex3f(-100.0f, 2.2f, 100.0f);

        glVertex3f(-100.0f, 0.0f, 100.0f);

        glVertex3f(-100.0f, 0.0f, -100.0f);

        glVertex3f(-100.0f, 2.2f, -100.0f);

        glVertex3f( 100.0f, 2.2f, -100.0f);

        glVertex3f( 100.0f, 0.0f, -100.0f);

        glVertex3f( 100.0f, 0.0f, -100.0f);

        glVertex3f( 100.0f, 2.2f, -100.0f);

        glVertex3f( 100.0f, 2.2f, 100.0f);

        glVertex3f( 100.0f, 0.0f, 100.0f);

    glEnd();

    glPushMatrix();

      // Translate to the appropriate starting point

      glTranslatef(-85.0f, 8.2, 100.2f);

      glScalef(0.01,0.01,0.01);

      // Note: We could change the line width with glLineWidth()

      // Render the characters

      char\* str = "IITI Parking Lot";

      for (char\* c = str; \*c != '\0'; c++) {

        glutStrokeCharacter(GLUT\_STROKE\_ROMAN, \*c);

      }

      // Another useful function

      //    int glutStrokeWidth(void \*font, int character);

      // Retrieve the original matrix

      glPopMatrix();

    glColor3f(0.9f, 0.9f, 0.9f);

    float sx = 20,sy = 5,sz = 2,szin = 1.75;

    for(auto x : {-75.0f,0.0f,75.0f}){

        for(auto y : {-48.0f,0.0f,48.0f}){

            glBegin(GL\_QUADS);

                glVertex3f(x + sx, 0.0f, sy + y);

                glVertex3f(x - sx, 0.0f, sy + y);

                glVertex3f(x - sx, 0.0f, -sy + y);

                glVertex3f(x + sx, 0.0f, -sy + y);

                /\*\*

                glVertex3f(x + sx, 3.0f, sy + y);

                glVertex3f(x - sx, 3.0f, sy + y);

                glVertex3f(x - sx, 3.0f, -sy + y);

                glVertex3f(x + sx, 3.0f, -sy + y);

                \*\*/

                glColor3f(0.25f, 0.36f, 0.12f);

                glVertex3f(x + sx - 0.1, szin, sy + y - 0.1);

                glVertex3f(x - sx + 0.1,szin, sy + y - 0.1);

                glVertex3f(x - sx + 0.1, szin, -sy + y + 0.1);

                glVertex3f(x + sx - 0.1, szin, -sy + y + 0.1);

                glColor3f(0.9f, 0.9f, 0.9f);

                glVertex3f(x + sx, sz, sy + y);

                glVertex3f(x + sx, 0.0f, sy + y);

                glVertex3f(x + sx, 0.0f, -sy + y);

                glVertex3f(x + sx, sz, -sy + y);

                glVertex3f(x - sx, sz, sy + y);

                glVertex3f(x - sx, 0.0f, sy + y);

                glVertex3f(x - sx, 0.0f, -sy + y);

                glVertex3f(x - sx, sz, -sy + y);

                glVertex3f(x - sx, 0.0f, sy + y);

                glVertex3f(x + sx, 0.0f, sy + y);

                glVertex3f(x + sx, sz, sy + y);

                glVertex3f(x - sx, sz, sy + y);

                glVertex3f(x - sx, 0.0f, -sy + y);

                glVertex3f(x + sx, 0.0f, -sy + y);

                glVertex3f(x + sx, sz, -sy + y);

                glVertex3f(x - sx, sz, -sy + y);

                float incx = 2\*sx/4,incy = 12,cx = x - sx;

                for(int i = 0; i < 5; i++){

                    glVertex3f(cx - 1,0.005,-sy + y);

                    glVertex3f(cx - 1,0.005,-sy + y - incy);

                    glVertex3f(cx + 1,0.005,-sy + y - incy);

                    glVertex3f(cx + 1,0.005,-sy + y);

                    glVertex3f(cx - 1,0.005, sy + y);

                    glVertex3f(cx - 1,0.005, sy + y + incy);

                    glVertex3f(cx + 1,0.005, sy + y + incy);

                    glVertex3f(cx + 1,0.005, sy + y);

                    cx += incx;

                }

            glEnd();

        }

    }

    for(auto c : cars){

          c.showcar();

    }

glutSwapBuffers();

}

void orientMe(float ang)

{

 lx = sin(ang);

lz = -cos(ang);

glLoadIdentity();

gluLookAt(x, y, z, x + lx,y + ly,z + lz,0.0f,1.0f,0.0f);

}

void moveMeFlat(int i)

{

if(xxxx==1)

y=y+i\*(lz)\*0.1; //\*\*\*\*\*\*\*\*\*

if(yyyy==1)

{

x=x+i\*(lz)\*.1;

}

else

{

z = z + i\*(lz)\*0.5;

 x = x + i\*(lx)\*0.5;}

glLoadIdentity();

gluLookAt(x, y, z,x + lx,y + ly,z + lz,0.0f,1.0f,0.0f);

}

void processNormalKeys(unsigned char key, int x, int y)

{

glLoadIdentity();

if (key == 'q')

exit(0);

if(key=='t')

 gluLookAt(0, 190, 0, 0, 0, -10, 0.0, 1.0,0.0);

if(key=='a')

 moveMeFlat(4);xxxx=1,yyyy=0;

if(key=='s')

 moveMeFlat(-4);xxxx=1,yyyy=0;

if(key=='w')

 moveMeFlat(4);yyyy=1;xxxx=0;

if(key=='d')

 moveMeFlat(-4);yyyy=1;xxxx=0;

if(key=='c'){

  gluLookAt(100, 150, 100, 0, 0 , 0, 1.0, 1.0, 1.0);

}

if(key=='v'){

  gluLookAt(0, 150, 0, 0, 0 , 0, 1.0, 1.0, 1.0);

}

else if(key=='l')

    {

        glEnable(GL\_LIGHT0);

        glEnable(GL\_LIGHT1);

        glEnable(GL\_LIGHT2);

        glEnable(GL\_LIGHT3);

        glEnable(GL\_LIGHT4);

        glEnable(GL\_LIGHT5);

        glEnable(GL\_LIGHT6);

        glEnable(GL\_LIGHT7);

        light\_mode = 1;

    }

    else if(key=='L')

    {

        glDisable(GL\_LIGHT0);

        glDisable(GL\_LIGHT1);

        glDisable(GL\_LIGHT2);

        glDisable(GL\_LIGHT3);

        glDisable(GL\_LIGHT4);

        glDisable(GL\_LIGHT5);

        glDisable(GL\_LIGHT6);

        glDisable(GL\_LIGHT7);

        light\_mode = 0;

    }

}

void inputKey(int key, int x, int y)

{

switch (key)

{

case GLUT\_KEY\_LEFT : angle -= 0.05f;orientMe(angle);break;

case GLUT\_KEY\_RIGHT : angle +=0.05f;orientMe(angle);break;

case GLUT\_KEY\_UP : moveMeFlat(2);xxxx=0,yyyy=0;break;

case GLUT\_KEY\_DOWN : moveMeFlat(-2);xxxx=0,yyyy=0;break;

}

}

void ProcessMenu(int value) // Reset flags as appropriate in response to menuselections

{

 glutPostRedisplay();

}

void ProcessMenu1(int value)

{

    if(movecarvar != value){

            glutSpecialFunc(movecar);

            movecarvar = value;

    }

    else{

            glutSpecialFunc(inputKey);

            movecarvar=0;

    }

}

void menu()

{

int control;

 int control1;

control= glutCreateMenu(ProcessMenu);

glutAddMenuEntry("\*\*CONTROLS\*\*",1);

glutAddMenuEntry("1) UP KEY:to move in Forward Direction.",1);

glutAddMenuEntry("2) DOWN KEY:to move in Backward Direction.",1);

glutAddMenuEntry("3) LEFT KEY:to Turn Left .",1);

glutAddMenuEntry("4) RIGHT KEY:to Turn Right .",1);

glutAddMenuEntry("5) d:moves Towards Right. ",1);

glutAddMenuEntry("6) a:moves Towards Left.",1);

glutAddMenuEntry("7) s:moves Away.",1);

glutAddMenuEntry("8) w:moves Near.",1);

glutAddMenuEntry("9) t:Top view.",1);

glutAddMenuEntry("10) q:Quit.",1);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

 control1=glutCreateMenu(ProcessMenu1);

glutAddMenuEntry("Move Yellow Car",1);

glutAddMenuEntry("Move Cyan Car",2);

glutAddMenuEntry("Move Purple Car",3);

glutAddMenuEntry("Move Blue Car",4);

//glutAddMenuEntry("Move Red Car",5);

glutAddMenuEntry("Move Orange Car",6);

glutAttachMenu(GLUT\_LEFT\_BUTTON);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DEPTH | GLUT\_DOUBLE | GLUT\_RGBA);

glutInitWindowPosition(0,0);

glutInitWindowSize(1010,710);

glutCreateWindow("car lot");

initialize();

initScene();

glutKeyboardFunc(processNormalKeys);

glutSpecialFunc(inputKey);

 menu();

glutDisplayFunc(renderScene);

glutIdleFunc(renderScene);

glutReshapeFunc(changeSize);

glutMainLoop();

return(0);

}