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);
}
```