**1.2.1:OpenGL Command Syntax:**

OpenGL commands use the prefix **gl**and initial capital letters for each word making up the command

name. Similarly, OpenGL defined constants begin with GL\_, use all capital letters and use underscores to

separate words (like GL\_COLOR\_BUFFER\_BIT).

OpenGL

application

Program

GLU

GL

GLUT

GLX

Xlib, Xtk

Frame

Buffer

3

**CHAPTER 2**

**LITERATURE SURVEY.**

This project makes extensive use of translations, rotations and scaling for creating .

**1.THE LINKS : http://www.cs.rutgers.edu/~decarlo/428/glman.html online man pages.**

**2.http://www.opengl.org/sdk/docs/man online man pages.**

**3.http://nehe.gamedev.net OpenGL tutorials.**

Provides the description of the following functions.

**void glScalef(TYPE sx, TYPE sy, TYPE sz)**

alters the current matrix by a scaling of (sx, sy, sz). TYPE here is GLfloat.

Here in the above considered example we use scaling to minimize the length of the curve at each iteration. For

this curve we use the scale factor to be 3 units because we substitute a line by 4 lines in each iteration.

**void glRotatef(TYPE angle, TYPE dx, TYPE dy, TYPE dz)**

alters the current matrix by a rotation of angle degrees about the axis(dx, dy, dz). TYPE here is GLfloat.

For a Koch curve we rotate by 60° about the z-axis.

**void glTranslatef(TYPE x, TYPE y, TYPE z)**

alters the current matrix by a displacement of (x, y, z). TYPE here is GLfloat.

We need to translate to display the new position of the line from the old position and also to go out to the

beginning of the next side while drawing.

**void glLoadIdentity()**

sets the current transformation matrix to an identity matrix.

**void glPushMatrix()**

pushes to the matrix stack corresponding to the current matrix mode.

**void glPopMatrix()**

pops from the matrix stack corresponding to the current matrix mode.

**void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top)**

defines a two-dimensional viewing rectangle in the plane z=0.

4

**void glutMouseFunc(myMouse)**

refers to the mouse callback function. The function to callback is defined as void myMouse(int button, int state, int x

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN) if (glutGetModifiers() & GLUT\_ACTIVE\_SHIFT)

decrease a level of recursion else

increase a level of recursion

**}**

Here mouse interface is given to increase a level of recursion by clicking mouse button and also to decrease a

level of recursion by doing the same holding the shift on the keyboard.

**void glutKeyboardFunc(myKey)**

refers to the keyboard callback function. The function to callback is defined as void myKey(unsigned char key, int x, int y)

{

if (c == 'q') exit

if (c == 'n')

//STATEMENTS and repeat when finished

}

Here keyboard interface is given to quit, the user can quit by pressing ‘q’ and to see next example of the

implementation, the user should press ‘n’.

**void glutSwapBuffers()**

swaps the front and back buffers.

User defined functions are used to color the curves in a standard cycle rainbow manner which becomes very

easy for the user to identify the levels of recursion for the curves.

5

**void glutInit(int \*argc, char\*\*argv)**

Initializes GLUT< the arguments from main are passed in and can by the application.

**void glutCreateWindow(char \*title)**

Creates a window on the display. The string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.

**void glutInitDisplaymode(unsigned int mode)**

Requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model(GLUT\_RGB<GLUT\_INDEX) and buffering (GLUT\_SINGLE<GLUT\_DOUBLE).

**void glutInitWindowSize(int width,int heights)**

Specifies the initial height and width of the window in pixels.

**void glutInitWindowPosition(int x,int y)**

Specifies the initial position of the top-left corner of the window in pixels.

**void glViewport(int x,int y,GLsizei width,GLsizei height)**

Specifies a width \* height viewport in pixels whose lower-left corner is at (x,y) measured from he origin of the window.

**void glutMainLoop()**

Cause the program to enter an event –processing loo p.it should be the statement in main.

**void glutPostRedisplay()**

Requests that the display callback be executed after the current callback returns.

**void gluLookAt(GLdouble eyex,GLdouble eyey, GLdouble eyez, GLdouble atx, GLdouble**

**aty, GLdouble**

**atz, GLdouble upx, GLdouble upy, GLdouble upz)**

Postmultiplies the current matrix determined by the viewer at the eye point looking at the point

with specified up direction.

**void gluPerscpective(GLdouble fov, GLdouble aspect, GLdouble near, GLdouble far)**

Defines a perspective viewing volume using the y direction field of view fov measured in degree,the aspect ratio of the front clipping plane, and the near and far distance.

6

**CHAPTER 3**

**REQUIREMENTS AND SPECIFICATION.**

**Hardware Constraints**

∙Processor : Pentium PC

∙RAM : 512MB

∙Hard Disk : 20GB(approx)

∙Display : VGA Color Monitor

**Software Constraints**

∙Operating System : Windows 98SE/2000/XP/Vista/UBUNTU

∙Language : Open Gl

∙Compiler : Eclipse/Microsoft Visual studio 2005

7

**CHAPTER 4**

**SOFTWARE DESIG .**

START **N.**

MAIN( )

INIT( )

DISPLAY( )

MOUSE( ) KEYBOARD( )

LEFT

BUTTON

RIGHT

BUTTON

ON

MOVE

CAR( )

PROCESS

NORMAL

KEYS( )

HOUSE

E

CONTROLS

STOP

8

**CHAPTER 5**

**IMPLEMENTATION.**

5.1:Keyboard function.

5.2:Display function.

5.3:Reshape function.

5.1:Keyboard function:

KEYBOARD

FUNC

if (key == 'q') exit(0); if(key=='t') gluLookAt(1,190,50,0,0 ,- 10,0.0,1.0,.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;

switch (key) {

case

GLUT\_KEY\_LEFT : angle -=

0.05f;orientMe(angle);break;

case

GLUT\_KEY\_RIGHT : angle

+=0.05f;orientMe(angle);break;

9

case

GLUT\_KEY\_UP :

moveMeFlat(2);xxxx=0,yyyy=0;br

eak;

case

GLUT\_KEY\_DOWN :

moveMeFlat(-

2);xxxx=0,yyyy=0;break;

}

5.2:Display function:

DISPLAY FUNC glColor3f(0.25f, 0.25f, 0.25f); 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();

for( i = -3; i < 3; i++) for( j=-3; j < 3; j++)

{

glPushMatrix(); glTranslatef((i)\*10.0,0,(j) \* 10.0); glColor3ub(a[i],b[j],c[i]); glCallList(carr\_display\_list); glPopMatrix();

}

glTranslatef(-20.0,0.0,0.0);glCallList(house\_display\_list);

10

5.3:Reshape function:

RESHAPE

FUNCTION if(h == 0)

h = 1;

ratio = 1.0f \* w / h; glMatrixMode(GL\_PROJECTION); glLoadIdentity();

glViewport(0, 0, w, h); 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);

11

**CHAPTER 6**

**6.1: INITIAL SCENE:**

This is the first scene which appears when the program is executed.

**6.2:FRONT VIEW :**

On pressing the “S” key the camera moves backwards and upwards simultaneously. The user can press “W” key to move the camera in the front direc tion in the same way but the path traversed is exactly opposite.

**6.3:TOP VIEW :**

On pressing the “t” key the camera changes to a pos ition so that the user can see the top view of the whole parking area.

12

**CHAPTER 7**

**CONCLUSION and FUTURE SCOPE.**

This project allows the user to rove in the parking lot and can even enter the houses that are present

along the parking area. So, it’s like a virtualization of that area.

**Future scope:**

**1.3-D MAP:**

This project can be modified and a lot of other objects can be added for example:-

Trees , boundary walls, multiplexes ,roads etc.

THUS A WHOLE CITY CAN BE CONSTRUCTED.

**2.GAME:**

This program can be developed in to a fully-fledged game like Counter Strike, IGI etc

13



// Prevent a divide by zero, when window is too short

// (you cant make a window of zero width).

// Reset the coordinate system before modifying

**APPENDIX.**

#include <GL/glut.h> #include <math.h> #include <stdlib.h>

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,house\_display\_list;

float theta=0.01,fxincr=0.1,fzincr=0,temp,theta1,fx=-10,fz=80; int xxxx=0,yyyy=0,kk=0,housevisible=0,movecarvar=0;

int a[36]={55,97,44,152,55,171,108,86,168,99,147,207,238,55,233,167,105,80,134,29 ,253,130,32, 240,110,199,224,121,93,199,180,61,110,251,77,237};

int b[36]={102,194,110,152,153,184,137,113,55,138,104,43,240,255,203,8,100,53,88, 64,127,64,87 ,5,2,144,211,128,10,89,27,11,175,185,157,241};

int c[36]={159,243,133,253,233,228,141,18,46,195,75,52,253,204,169,30,78,94,68,11 7,4,2,33,12, 2,25,195,76,26,54,98,103,205,173,65,242};

void changeSize(int w, int h)

{

if(h == 0)

h = 1;

ratio = 1.0f \* w / h;

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);

14

glLoadIdentity();

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

}

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);//aglVertex3f(-0.86,-.48,0.7);//bglVertex3f(-.74,-0.2,0.7);//cglVertex3f(-.42,-.2,0.7);//dglVertex3f(-0.3,-.48,0.7);//eglVertex3f(.81,-0.48,0.7);//fglVertex3f(.94,-0.2,0.7);//gglVertex3f(1.24,-.2,0.7);//hglVertex3f(1.38,-.48,0.7);//iglVertex3f(1.52,-.44,0.7);//jglVertex3f(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);//0glVertex3f(-1.02,0.6,0.7);//pglVertex3f(-1.2,0.22,0.7);//qglVertex3f(-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'

15

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);//aglVertex3f(-1.12,-.48,-0.7);//a'glVertex3f(-0.86,-.48,0.7);//bglVertex3f(-0.86,-.48,-0.7);//b'glVertex3f(-.74,-0.2,0.7);//cglVertex3f(-.74,-0.2,-0.7);//c'glVertex3f(-.42,-.2,0.7);//dglVertex3f(-.42,-.2,-0.7);//d'glVertex3f(-0.3,-.48,0.7);//eglVertex3f(-0.3,-.48,-0.7);//e'glVertex3f(.81,-0.48,0.7);//fglVertex3f(.81,-0.48,-0.7);//f'glVertex3f(.94,-0.2,0.7);//gglVertex3f(.94,-0.2,-0.7);//g'glVertex3f(1.24,-.2,0.7);//hglVertex3f(1.24,-.2,-0.7);//h'glVertex3f(1.38,-.48,0.7);//iglVertex3f(1.38,-.48,-0.7);//i'glVertex3f(1.52,-.44,0.7);//jglVertex3f(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

16

glVertex3f(.52,0.56,-0.7);//n'glVertex3f(-0.1,0.6,0.7);//0glVertex3f(-0.1,0.6,-0.7);//o'glVertex3f(-1.02,0.6,0.7);//pglVertex3f(-1.02,0.6,-0.7);//p'glVertex3f(-1.2,0.22,0.7);//qglVertex3f(-1.2,0.22,-0.7);//q'glVertex3f(-1.2,-.28,0.7);//rglVertex3f(-1.2,-.28,-0.7);//r'glEnd();

glBegin(GL\_POLYGON); // top filling glVertex3f(-0.1,0.6,0.7);//oglVertex3f(-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);//oglVertex3f(-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);//qglVertex3f(-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'

17

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

glEnd();

glBegin(GL\_POLYGON); glVertex3f(-1.12,-.48,0.7);//aglVertex3f(-0.86,-.48,0.7);//bglVertex3f(-.74,-0.2,0.7);//cglVertex3f(-0.64,0.22,0.7);//ccglVertex3f(-1.08,0.22,0.7);//ddglVertex3f(-1.2,0.22,0.7);//qglVertex3f(-1.2,-.28,0.7);//rglEnd(); glBegin(GL\_POLYGON); glVertex3f(-.74,-0.2,0.7);//cglVertex3f(-0.64,0.22,0.7);//ccglVertex3f(-0.5,0.22,0.7);//hhglVertex3f(-0.5,-0.2,0.7);//ppglEnd(); 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);//hglVertex3f(0.0,-0.2,0.7);//ooglEnd(); 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();

18

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);//qglVertex3f(-1.08,0.22,0.7);//ddglVertex3f(-0.98,0.5,0.7);//aaglVertex3f(-1.02,0.6,0.7);//pglEnd(); glBegin(GL\_POLYGON); glVertex3f(-1.02,0.6,0.7);//pglVertex3f(-0.98,0.5,0.7);//aaglVertex3f(0.44,0.5,0.7);//jj glVertex3f(.52,0.56,0.7);//n glVertex3f(-0.1,0.6,0.7);//0glEnd(); glBegin(GL\_POLYGON); glVertex3f(-0.64,0.5,0.7);//bbglVertex3f(-0.64,0.22,0.7);//ccglVertex3f(-0.5,0.22,0.7);//hhglVertex3f(-0.5,0.5,0.7);//eeglEnd(); 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);

19

glVertex3f(-.42,-.2,0.7);//dglVertex3f(.94,-0.2,0.7);//gglVertex3f(.81,-0.48,0.7);//fglVertex3f(-0.3,-.48,0.7);//eglEnd(); glBegin(GL\_POLYGON); glVertex3f(1.14,0.22,0.7);//l glVertex3f(1.52,.14,0.7);//k glVertex3f(1.52,-.44,0.7);//jglVertex3f(1.38,-.48,0.7);//iglVertex3f(1.24,-.2,0.7);//hglEnd(); 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);

20

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);//hhglVertex3f(0.0,0.22,0.7);//gg glVertex3f(0.0,-0.2,0.7);//ooglVertex3f(-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);//mmglVertex3f(0.12,-0.2,0.7);//nn

glEnd();

glBegin(GL\_POLYGON); // door2 body- far, driver glVertex3f(0.12,0.22,-0.7);//ll'

21

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);//jglEnd();

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

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);//eglVertex3f(-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);//dglVertex3f(-.42,-.2,-0.7);//d'glVertex3f(-0.3,-.48,-0.7);//e'

22

glVertex3f(-0.3,-.48,0.7);//eglEnd(); glBegin(GL\_POLYGON); glVertex3f(-1.2,-.28,0.7);//rglVertex3f(-1.2,-.28,-0.7);//r'glVertex3f(-1.12,-.48,-0.7);//a'glVertex3f(-1.12,-.48,0.7);//aglEnd(); glBegin(GL\_POLYGON); glVertex3f(-1.12,-.48,0.7);//aglVertex3f(-1.12,-.48,-0.7);//a'glVertex3f(-0.86,-.48,-0.7);//b'glVertex3f(-0.86,-.48,0.7);//bglEnd(); glBegin(GL\_POLYGON); glVertex3f(-0.86,-.48,0.7);//bglVertex3f(-0.86,-.48,-0.7);//b'glVertex3f(-.74,-0.2,-0.7);//c'glVertex3f(-.74,-0.2,0.7);//cglEnd(); glBegin(GL\_POLYGON); glVertex3f(-.74,-0.2,0.7);//cglVertex3f(-.74,-0.2,-0.7);//c'glVertex3f(-.42,-.2,-0.7);//d'glVertex3f(-.42,-.2,0.7);//dglEnd(); glBegin(GL\_POLYGON); glVertex3f(.81,-0.48,0.7);//fglVertex3f(.81,-0.48,-0.7);//f'glVertex3f(.94,-0.2,-0.7);//g'glVertex3f(.94,-0.2,0.7);//gglEnd(); glBegin(GL\_POLYGON); glVertex3f(.94,-0.2,0.7);//gglVertex3f(.94,-0.2,-0.7);//g'glVertex3f(1.24,-.2,-0.7);//h'glVertex3f(1.24,-.2,0.7);//hglEnd();

23

glBegin(GL\_POLYGON); glVertex3f(1.24,-.2,0.7);//hglVertex3f(1.24,-.2,-0.7);//h'glVertex3f(1.38,-.48,-0.7);//i'glVertex3f(1.38,-.48,0.7);//iglEnd(); glBegin(GL\_POLYGON); glVertex3f(1.38,-.48,0.7);//iglVertex3f(1.38,-.48,-0.7);//i'glVertex3f(1.52,-.44,-0.7);//j'glVertex3f(1.52,-.44,0.7);//jglEnd();

glBegin(GL\_LINE\_LOOP); // door outline- rear, front glColor3f(1.0,1.0,1.0); glVertex3f(-0.5,0.22,0.7);//hhglVertex3f(0.0,0.22,0.7);//gg glVertex3f(0.0,-0.2,0.7);//ooglVertex3f(-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);//mmglVertex3f(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();

24

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);//jglEnd(); 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);//ppglVertex3f(-1.068,0.5,-0.6);//pp'glVertex3f(-1.2,0.22,-0.6);//qq'glVertex3f(-1.2,0.22,0.6);//qqglEnd();

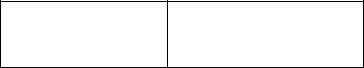
glBegin(GL\_POLYGON); //leftmost window front glVertex3f(-0.98,0.5,0.7);//aaglVertex3f(-0.64,0.5,0.7);//bbglVertex3f(-0.64,0.22,0.7);//ccglVertex3f(-1.08,0.22,0.7);//dd

glEnd();

glBegin(GL\_POLYGON); //leftmost window back glVertex3f(-0.98,0.5,-0.7);//aaglVertex3f(-0.64,0.5,-0.7);//bbglVertex3f(-0.64,0.22,-0.7);//ccglVertex3f(-1.08,0.22,-0.7);//dd glEnd();

glBegin(GL\_POLYGON); //middle window front glVertex3f(-0.5,0.5,0.7);

25



//rightmost window front

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)

;

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);

}

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"); initScene(); glutKeyboardFunc(processNormalKeys); glutSpecialFunc(inputKey); menu(); glutDisplayFunc(renderScene); glutIdleFunc(renderScene);

26

glutReshapeFunc(changeSize);

glutMainLoop();

return(0);

}