Source code

#include<stdlib.h>

#include<GL/glut.h>

GLfloat vertices[][3]={{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},{1.0,1.0,-1.0},{-1.0,1.0,-1.0},{-1.0,-1.0,1.0},{1.0,-

1.0,1.0},{1.0,1.0,1.0},{-1.0,1.0,1.0}};

GLfloat normals[][3]={{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},{1.0,1.0,-1.0},{-1.0,1.0,-1.0},{-1.0,-1.0,1.0},{1.0,-

1.0,1.0},{1.0,1.0,1.0},{-1.0,1.0,1.0}};

GLfloat colors[8][3]={

{0.0,0.0,0.0}, //front bottom-left

{1.0,0.0,0.0}, //front bottom-right

{1.0,1.0,0.0}, //front top-right

{0.0,1.0,0.0}, //front top-left

{0.0,0.0,1.0}, //back bottom-left

{1.0,0.0,1.0}, //back bottom-right

{1.0,1.0,1.0}, //back top-right

{0.0,1.0,1.0} //back top-left

};

GLfloat divi=1.0,scale=1.0,v=0.5,nd=2, ik=0.0,m=0.0,h,c=0.0;

GLint il=10,projection=0,flag[5],i=0,elseflag=0,lighting=0;

//projection is the no. of times m is pressed

//flag[] is used to select iteration nos.

//nd is used to select the no. of divisions

/\*elseflag is used to restrict the original cube to be

displayed only once followed by its divided constituents when using 'm'\*/

GLfloat amb[]={0.7,0.7,0.7,1.0};

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

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

GLfloat shininess[]={50.0};

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

GLfloat lp[]={-15.0,15.0,3.0,0.0};

GLfloat lp1[]={45.0,15.0,3.0,0.0};

void polygon(int a,int b,int c,int d)

{

glBegin(GL\_POLYGON);

glColor3fv(colors[a]);

glNormal3fv(normals[a]);

glVertex3fv(vertices[a]);

glColor3fv(colors[b]);

glNormal3fv(normals[b]);

glVertex3fv(vertices[b]);

glColor3fv(colors[c]);

glNormal3fv(normals[c]);

glVertex3fv(vertices[c]);

Pixelator

Department of CSE, BIT, 2016-17 11

glColor3fv(colors[d]);

glNormal3fv(normals[d]);

glVertex3fv(vertices[d]);

glEnd();

}

void colorcube(void)

{

polygon(0,3,2,1);

polygon(2,3,7,6);

polygon(4,5,6,7);

polygon(0,4,7,3);

polygon(0,1,5,4);

polygon(1,2,6,5);

}

void di(GLfloat x,GLfloat y,GLfloat z,GLfloat scale,GLfloat divi,int il,GLfloat v,int col,GLfloat ik,GLint

i)

{

/\* This function is a replica of divide() but is used to call

original cube in each iteration only once\*/

GLfloat m;

if(projection==1)

m=scale/2.0;

else

m=scale/2.0+v+ik;

glPushMatrix();// topleft

if(il==10)

glColor3fv(colors[3]);

else

glColor3fv(colors[col]);

glTranslatef(x-m,y+m,-z);

glutSolidCube(divi);

glPopMatrix();

glPushMatrix();// topright

if(il==10)

glColor3fv(colors[2]);

else

glColor3fv(colors[col]);

glTranslatef(x+m,y+m,-z);

glutSolidCube(divi);

glPopMatrix();

glPushMatrix();// bottomleft

if(il==10)

Pixelator

Department of CSE, BIT, 2016-17 12

glColor3fv(colors[0]);

else

glColor3fv(colors[col]);

glTranslatef(x-m,y-m,-z);

glutSolidCube(divi);

glPopMatrix();

glPushMatrix();// bottomright

if(il==10)

glColor3fv(colors[1]);

else

glColor3fv(colors[col]);

glTranslatef(x+m,y-m,-z);

glutSolidCube(divi);

glPopMatrix();

}

void divide(GLfloat x,GLfloat y,GLfloat z,GLfloat scale,GLfloat divi,int il,GLfloat v,int col,GLfloat

ik,GLint i)

{

GLfloat m;//should declare m again!

GLfloat c;

//x,y are centre of the cube to which we translate it to

//scale is used to give the lenght of cube

//divi is the size of the cube

//col is the index needed to be given to colors[]

//i is the iteration no.

/\*v is the amount of spacing/projection onto the next

divided cube-set\*/

//il is used to recongnise the no. of iterations to be given

//ik=-v initially and incrementally goes towards 0

h=v;

if(projection==0)

{

m=scale/2.0+v+ik;

glPushMatrix();// topleft

if(il==10)

glColor3fv(colors[3]);

else

glColor3fv(colors[col]);

glTranslatef(x-m,y+m,-z);

glutSolidCube(divi);

glPopMatrix();

glPushMatrix();// topright

Pixelator

Department of CSE, BIT, 2016-17 13

if(il==10)

glColor3fv(colors[2]);

else

glColor3fv(colors[col]);

glTranslatef(x+m,y+m,-z);

glutSolidCube(divi);

glPopMatrix();

glPushMatrix();// bottomleft

if(il==10)

glColor3fv(colors[0]);

else

glColor3fv(colors[col]);

glTranslatef(x-m,y-m,-z);

glutSolidCube(divi);

glPopMatrix();

glPushMatrix();// bottomright

if(il==10)

glColor3fv(colors[1]);

else

glColor3fv(colors[col]);

glTranslatef(x+m,y-m,-z);

glutSolidCube(divi);

glPopMatrix();

c=m;

if(il>=nd && il<=20)

{

if(il==10)

{

divide(x-m,y+m,z+1.0,scale/2,divi/2,il/2,v/2,3,0.0,i+1); //tl

divide(x+m,y+m,z+1.0,scale/2,divi/2,il/2,v/2,2,0.0,i+1); //tr

divide(x-m,y-m,z+1.0,scale/2,divi/2,il/2,v/2,0,0.0,i+1); //bl

divide(x+m,y-m,z+1.0,scale/2,divi/2,il/2,v/2,1,0.0,i+1); //br

}

else

{

divide(x-m,y+m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,0.0,i+1); //tl

divide(x+m,y+m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,0.0,i+1); //tr

divide(x-m,y-m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,0.0,i+1); //bl

divide(x+m,y-m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,0.0,i+1); //br

}

}

}

else if(projection>=0)

{

if(projection==1)

Pixelator

Department of CSE, BIT, 2016

-17 14

m=scale/2.0;

else m=scale/2.0+v+ik;

if(flag[0]==1)

{

glPushMatrix();// topleft

if(il==10)

glColor3fv(colors[3]);

else

glColor3fv(colors[col]);

glTranslatef(x

-m,y+m,

-z);

glutSolidCube(divi);

glPopMatrix();

glPushMatrix();// topright

if(il==10)

glColor3fv(colors[2]);

else

glColor3fv(colors[col]);

glTranslatef(x+m,y+m,

-z);

glutSolidCube(divi);

glPopMatrix();

glPushMatrix();// bottomleft

if(il==10)

glColor3fv(colors[0]);

else

glColor3fv(colors[col]);

glTranslatef(x

-m,y

-m,

-z);

glutSolidCube(divi);

glPopMatrix();

glPushMatrix();// bottomright

if(il==10)

glColor3fv(colors[1]);

else

glColor3fv(colors[col]);

glTranslatef(x+m,y

-m,

-z);

glutSolidCube(divi);

glPopMatrix();

}

if(flag[1]==1) {

if(il==10)

{

di(x

-m,y+m,z+1.0,scale/2,divi/2,il/2,v/2,3,ik,i+1); //tl

di(x+m,y+m,z+1.0,scale/2,divi/2,il/2,v/2,2,ik,i+1); //tr

di(x

-m,y

-m,z+1.0,scale/2,divi/2,il/2,v/2,0,ik,i+1); //bl

di(x+m,y

-m,z+1.0,scale/2,divi/2,il/2,v/2,1,ik,i+1); //br

Pixelator

Department of CSE, BIT, 2016

-17 15

}

else {

di(x

-m,y+m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //tl

di(x+m,y+m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //tr

di(x

-m,y

-m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //bl

di(x+m,y

-m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //br

}}

if(flag[2]==1) {

if(i==0

)

{

if(il==10)

{

divide(x

-m,y+m,z+1.0,scale/2,divi/2,il/2,v/2,3,ik,i+1); //tl

divide(x+m,y+m,z+1.0,scale/2,divi/2,il/2,v/2,2,ik,i+1); //tr

divide(x

-m,y

-m,z+1.0,scale/2,divi/2,il/2,v/2,0,ik,i+1); //bl

divide(x+m,y

-m,z+1.0,scale/2,divi/2,il/2,v/2,1,ik,i+1); //br

}

else {

divide(x

-m,y+m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //tl

divide(x+m,y+m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //tr

divide(x

-m,y

-m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //bl

divide(x+m,y

-m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //br

}}

else if(i==1) {

if(il==10) {

di(x

-m,y+m,z+1.0,scale/2,divi/2,il/2,v/2,3,ik,i+1); //tl

di(x+m,y+m,z+1.0,scale/2,divi/2,il/2,v/2,2,ik,i+1); //tr

di(x

-m,y

-m,z+1.0,scale/2,divi/2,il/2,v/2,0,ik,i+1); //bl

di(x+m,y

-m,z+1.0,scale/2,divi/2,il/2,v/2,1,ik,i+1); //br

}

else {

di(x

-m,y+m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //tl

di(x+m,y+m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //tr

di(x

-m,y

-m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //bl

di(x+m,y

-m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //br

}}}

Pixelator

Department of CSE, BIT, 2016

-17 16

if(flag[3]==1) {

if(i==0)

{

if(il==10)

{

divide(x

-m,y+m,z+1.0,scale/2,divi/2,il/2,v/2,3,ik,i+1); //tl

divide(x+m,y+m,z+1.0,scale/2,divi/2,il/2,v/2,2,ik,i+1); //tr

divide(x

-m,y

-m,z+1.0,scale/2,divi/2,il/2,v/2,0,ik,i+1); //bl

divide(x+m,y

-m,z+1.0,scale/2,divi/2,il/2,v/2,1,ik,i+1); //br

}

else {

divide(x

-m,y+m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //tl

divide(x+m,y+m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //tr

divide(x

-m,y

-m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //bl

divide(x+m,y

-m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //br

}}

else if(i==1) {

if(il==10) {

divide(x

-m,y+m,z+1.0,scale/2,divi/2,il/2,v/2,3,ik,i+1); //tl

divide(x+m,y+m,z+1.0,scale/2,divi/2,il/2,v/2,2,ik,i+1); //tr

divide(x

-m,y

-m,z+1.0,scale/2,divi/2,il/2,v/2,0,ik,i+1); //bl

divide(x+m,y

-m,z+1.0,scale/2,divi/2,il/2,v/2,1,ik,i+1); //br

}

else {

divide(x

-m,y+m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //tl

divide(x+m,y+m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //tr

divide(x

-m,y

-m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //bl

divide(x+m,y

-m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //br

}}}

if(i==2) {

if(il==10)

{

di(x

-m,y+m,z+1.0,scale/2,divi/2,il/2,v/2,3,ik,i+1); //tl

di(x+m,y+m,z+1.0,scale/2,divi/2,il/2,v/2,2,ik,i+1); //tr

di(x

-m,y

-m,z+1.0,scale/2,divi/2,il/2,v/2,0,ik,i+1); //bl

di(x+m,y

-m,z+1.0,scale/2,divi/2,il/2,v/2,1,ik,i+1); //br

}

Pixelator

Department of CSE, BIT, 2016-17 17

else

{

di(x-m,y+m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //tl

di(x+m,y+m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //tr

di(x-m,y-m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //bl

di(x+m,y-m,z+1.0,scale/2,divi/2,il/2,v/1.8,col,ik,i+1); //br

}

}

}

}

static GLfloat theta[]={0.0,0.0,0.0,0.0,0.0,0.0};

static GLint axis=2;

void display()

{

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

glClearColor(1.0,1.0,1.0,1.0);

glMaterialfv(GL\_FRONT,GL\_AMBIENT,amb);

glMaterialfv(GL\_FRONT,GL\_DIFFUSE,diff);

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

glMaterialfv(GL\_FRONT,GL\_SHININESS,shininess);

glColorMaterial(GL\_FRONT,GL\_DIFFUSE);

glLightfv(GL\_LIGHT0,GL\_POSITION,lp);

glLightfv(GL\_LIGHT0,GL\_DIFFUSE,li);

glLoadIdentity();

glRotatef(theta[0],1.0,0.0,0.0);

glRotatef(theta[1],0.0,1.0,0.0);

glRotatef(theta[2],0.0,0.0,1.0);

glRotatef(theta[3],-1.0,0.0,0.0);

glRotatef(theta[4],0.0,-1.0,0.0);

glRotatef(theta[5],0.0,0.0,-1.0);

colorcube();

if(nd>0 && projection==0)

divide(0.0,0.0,1.75,scale,1.0,il,v,0,0.0,0);

else if(projection>0)

{

if(flag[0]==1)

{

nd=25;

divide(0.0,0.0,1.75,scale,1.0,il,v,0,ik,0);

}

if(flag[1]==1)

{

Pixelator

Department of CSE, BIT, 2016

-17 18

if(elseflag)

{

di(0.0,0.0,1.75,scale,1.0,il,v,0,ik,0);

elseflag=0;

}

nd=10;

divide(0.0,0.0,1.75,scale,1.0,il,v,0,ik,0);

}

if(flag[2]==1)

{

nd=5;

if(elseflag)

{

nd=10;

divide(0.0,0.0,1.75,scale,1.0,il,v,0,ik,0);

elseflag=0;}

divide(0.0,0.0,1.75,scale,1.0,il,v,0,ik,0);}

if(flag[3]==1)

{

nd=2;

if(elseflag==1)

{

nd=5;

divide(0.0,0.0,1.75,scale,1.0,il,v,0,ik,0);

elseflag=0;

}

divide(0.0,0.0,1.75,scale,1.0,il,v,0,ik,0);}

if(flag[4]==1)

{

nd=2;

projection=0;

divide(0.0,0.0,1.75,scale,1.0,il,v,0,ik,0);

}

}

glFlush();

glutSwapBuffers(); }

void spincube() {

theta[axis]+=1.0;

if(theta[axis]>360.0) theta[axis]

-=360.0;

display();

glutPostRedisplay(); }

void myr(int w,int h) {

Pixelator

Department of CSE, BIT, 2016-17 19

glViewport(0,0,w,h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if(w<=h)

//glOrtho(-2.0,2.0,-2.0\*(GLfloat)h/(GLfloat)w,2.0\*(GLfloat)h/(GLfloat)w,-10.0,10.0);

//glOrtho(-2.0\*(GLfloat)w/(GLfloat)h,2.0\*(GLfloat)w/(GLfloat)h,-2.0,2.0,-10.0,10.0);

glOrtho(-5.0,5.0,-5.5\*(GLfloat)h/(GLfloat)w,5.5\*(GLfloat)h/(GLfloat)w,-10.0,10.0);

else

glOrtho(-5.0\*(GLfloat)w/(GLfloat)h,5.0\*(GLfloat)w/(GLfloat)h,-5.5,5.5,-10.0,10.0);

glMatrixMode(GL\_MODELVIEW);

}

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

{

if(key=='x') axis=0;

if(key=='y') axis=1;

if(key=='z') axis=2;

if(key=='X') axis=3;

if(key=='Y') axis=4;

if(key=='Z') axis=5;

display();

if(key=='m') {

projection+=1;

if(projection>0 && projection<=4)

flag[0]=1;

else flag[0]=0;

if(projection>=5 && projection<=7)

flag[1]=1;

else flag[1]=0;

if(projection>=8 && projection<=10)

flag[2]=1;

else flag[2]=0;

if(projection>=11)

flag[3]=1;

else flag[3]=0;

if(projection==13)

flag[4]=1;

else flag[4]=0;

if(projection==5||projection==8||projection==11)

{elseflag=1;

ik=-c-0.2;

//ik=-c;

if(projection==11||projection==8)

ik=-c-0.05;

}

else elseflag=0;

Pixelator

Department of CSE, BIT, 2016

-17 20

display();

ik+=h/5.0;

}

if(key=='n')

{ projection

-=1;

display();

ik

-=h/5.0;

}

}

void lighti() {

if(lighting==1)

{

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glShadeModel(GL\_SMOOTH);

glEnable(GL\_NORMALIZE);

glEnable(GL\_COLOR\_MATERIAL);

}

else

glDisable(GL\_LIGHTING); }

void mymenu(int id) {

switch(id)

{

case 0: nd=

-1;

glutIdleFunc(spincube);

break;

case 1: nd=25;

glutIdleFunc(spincube);

break;

case 2: nd=10;

glutIdleFunc(spincube);

break;

case 3: nd=5;

glutIdleFunc(spincube);

break;

case 4: nd=2;

glutIdleFunc(spincube);

break;

case 5: lighting=1;

lighti();

break;

Pixelator

Department of CSE, BIT, 2016-17 21

case 6: lighting=0;

lighti();

break;

}

}

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

{

ik=ik-v;

glutInit(&argc,argv);

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

glutInitWindowSize(500,500);

glutCreateWindow("Pixelator");

glutReshapeFunc(myr);

glutDisplayFunc(display);

glutCreateMenu(mymenu);

glutAddMenuEntry("Cube",0);

glutAddMenuEntry("Div 1",1);

glutAddMenuEntry("Div 2",2);

glutAddMenuEntry("Div 3",3);

glutAddMenuEntry("Div 4",4);

glutAddMenuEntry("Lighting",5);

glutAddMenuEntry("Light Off",6);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutIdleFunc(spincube);

glutKeyboardFunc(keys);

glEnable(GL\_DEPTH\_TEST);

glutMainLoop();

}