```
#include"stdafx.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <gl/glut.h>
#include<iostream>

#define SCENE 10

enum { FRONTPAGE,ENCRYPTION,EXIT };

int width = 650, height = 650;

void *font =GLUT_BITMAP_HELVETICA_12;
void *fonts[] =
{
GLUT_BITMAP_9_BY_15,
GLUT_BITMAP_TIMES_ROMAN_10,
GLUT_BITMAP_TIMES_ROMAN_24,          // Text Styles
GLUT_BITMAP_HELVETICA_12,
GLUT_BITMAP_HELVETICA_10,
GLUT_BITMAP_HELVETICA_18,
};

// TO DISPLAY THE TEXT INFORMATION

void output(int x, int y, char *string,int j)
{
int len, i;
```

```
glRasterPos2f(x, y);
len = (int) strlen(string);
for (i = 0; i < len; i++)
glutBitmapCharacter(fonts[j], string[i]);
}

// FUNCTIONS. FOR DELAY

void delay(void)
{
int i,j,k;
for(i=0;i<5000;i++)
{
for(j=0;j<10000;j++);
for(k=0;k<15000;k++);
}
}
void delay1(void)
{
int i;
for(i=0;i<10000;i++);
}

// FIRST SCREEN - FRONT PAGE
void front_page()
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glMatrixMode(GL_MODELVIEW);
glPushMatrix();

output(220,550,"COLLEGE NAME",2);
```

```
output(320,500,"A",2);
output(220,450,"MINI PROJECT ON",2);

output(220,400,"ENCRYPTION AND DECRYPTION",2);

output(130,200,"Guides:",3);
output(175,180,"Name 1",2);
output(175,150,"Name 2",2);

output(450,200,"By:",3);
output(475,180,"Name 1",2);
output(475,150,"Name 2",2);
output(275,50,"Press S to start",2);
  glFlush();
glCallList(SCENE);
glPopMatrix();
glutSwapBuffers();
}

/* Movement angles */
GLint
movement_angle=0,packet_angle=0,head_angle=0,packet_angle1=0,head_angle1=0,packet_angle2
=0,head_angle2=0,packet_angle3=0,head_angle3=0,packet_angle4=0;
GLfloat mov_speed = 1;
GLint
head_angle4=0,packet_angle5=0,head_angle5=0,packet_angle6=0,head_angle6=0,packet_angle7,he
ad_angle7=0,packet_angle8=40;



/* Movement angles */
```

```
GLint
movement_angle1=0,arrow_angle=0,rev_arrow_angle=0,phy_header_angle=0,rev_phy_header_ang
le=0,rev_phy_header_angle1=0,phy_header_angle1=0,analog_sig_angle=0,rev_analog_sig_angle =
0;



void animation_encryp(void)

{

if ((movement_angle += mov_speed) >= 600)

movement_angle = 600;

if ((arrow_angle += mov_speed) >= 150)

arrow_angle = 150;

if(arrow_angle==150)

if ((phy_header_angle += mov_speed) >= 100)

phy_header_angle = 100;

if(phy_header_angle==100)

if ((phy_header_angle1 += mov_speed) >= 100)

phy_header_angle1 = 100;

if(phy_header_angle1==100)

if ((analog_sig_angle += mov_speed) >= 100)

analog_sig_angle = 100;

if(analog_sig_angle==100)

if ((movement_angle1 += mov_speed) >= 420)

movement_angle1 = 420;

if(movement_angle1 ==420)

if ((rev_analog_sig_angle += mov_speed) >= 100)

rev_analog_sig_angle = 100;

if(rev_analog_sig_angle ==100)

if ((rev_phy_header_angle += mov_speed) >= 200)

rev_phy_header_angle = 200;

if(rev_phy_header_angle ==200)

if ((rev_phy_header_angle1 += mov_speed) >= 100)
```

```
rev_phy_header_angle1 = 100;
if(rev_phy_header_angle ==200)
if ((rev_arrow_angle += mov_speed) >= 100)
rev_arrow_angle = 100;
glutPostRedisplay();
}


void computer()
{
glColor3f(0.75,0.85,0.65);//keyboard
glBegin(GL_QUADS);
glVertex2f(55.0,340.0);
glVertex2f(145.0,340.0);
glVertex2f(150.0,350.0);
glVertex2f(60.0,350.0);
glEnd();

glColor3f(0.75,0.85,0.65);//cabinet
glBegin(GL_LINE_LOOP);
glVertex2f(60.0,355.0);
glVertex2f(150.0,355.0);
glVertex2f(150.0,370.0);
glVertex2f(60.0,370.0);
glEnd();

glColor3f(0.75,0.85,0.65);
glBegin(GL_LINE_LOOP);
glVertex2f(75.0,380.0);
glVertex2f(135,380.0);
glVertex2f(135.0,430.0);
```

```
glVertex2f(75.0,430.0);
glEnd();

glColor3f(0.7,0.8,0.6);
glBegin(GL_QUADS);
glVertex2f(80.0,385.0);
glVertex2f(130.0,385.0);
glVertex2f(130.0,425.0);
glVertex2f(80.0,425.0);
glEnd();

  glColor3f(0.75,0.85,0.65);
glBegin(GL_LINES);
glVertex2f(90.0,370.0);
glVertex2f(90.0,380.0);
glVertex2f(120.0,370.0);
glVertex2f(120.0,380.0);
glEnd();
}

void computer_dest()
{
glColor3f(0.75,0.85,0.65);//keyboard
glBegin(GL_QUADS);
glVertex2f(545.0,525.0);
glVertex2f(635.0,525.0);
glVertex2f(640.0,535.0);
glVertex2f(550.0,535.0);
glEnd();

glColor3f(0.75,0.85,0.65);//cabinet
```

```
glBegin(GL_LINE_LOOP);
glVertex2f(550.0,540.0);
glVertex2f(640.0,540.0);
glVertex2f(640.0,555.0);
glVertex2f(550.0,555.0);
glEnd();

glColor3f(0.75,0.85,0.65);//cpu
glBegin(GL_LINE_LOOP);
glVertex2f(565.0,565.0);
glVertex2f(625,565.0);
glVertex2f(625.0,615.0);
glVertex2f(565.0,615.0);
glEnd();

glColor3f(0.7,0.8,0.6);//monitor
glBegin(GL_QUADS);
glVertex2f(570.0,570.0);
glVertex2f(620.0,570.0);
glVertex2f(620.0,610.0);
glVertex2f(570.0,610.0);
glEnd();

glColor3f(0.75,0.85,0.65);
glBegin(GL_LINES);
glVertex2f(580.0,555.0);
glVertex2f(580.0,565.0);
glVertex2f(610.0,555.0);
glVertex2f(610.0,565.0);
glEnd();
}
```

```
void cipher()
{

glColor3f(0.0f,0.0f,1.0f);
glPushMatrix();
glScalef(40,20,.5);
glTranslatef(3,14,0);
glutWireCube(2);
output(-1,0," CIPHER",2);
glPopMatrix();

}


void message_data()
{

glColor3f(1.0f,1.0f,0.0f);
glPushMatrix();
glScalef(42,20,0.5);
glTranslatef(3,14,0);
glutWireCube(2);
output(-1,0," MESSAGE",2);
glPopMatrix();
}


void message_key(void)
```

```
{

glColor3f(0.0f,1.0f,0.0f);
glPushMatrix();
glScalef(20,20,.5);
glTranslatef(3,14,0);
glutWireCube(2);

output(-1,0,"KEY",2);
glPopMatrix();
}


void encryption()
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glMatrixMode(GL_MODELVIEW);
//glPushMatrix();
// glColor3f(0.25,0.15,0.36);
glColor3f(1.0f,1.0f,1.0f);
output(150,640,"ENCRYPTION AND DECRYPTION PROCESS",2);
//glPopMatrix();

glColor3f(1.0f,1.0f,1.0f);
glPushMatrix();
glTranslatef(-30,200,0);
output(75,440,"Sender",2);
computer();
glPopMatrix();

glColor3f(1.0f,1.0f,1.0f);
```

```
glPushMatrix();
output(565,630,"Receiver",2);
computer_dest();
glPopMatrix();

glColor3f(1.0f,1.0f,1.0f);
glPushMatrix();
glScalef(70,40,.5);
glTranslatef(1.5,7,0);
glutWireCube(2);
glPopMatrix();

if(phy_header_angle1!=100)
{
glPushMatrix();
glTranslatef(0,-phy_header_angle1,0);
glPushMatrix();
glTranslatef(0,-arrow_angle,0);
glTranslatef(0,150,0);
message_data();
glPopMatrix();
glPushMatrix();
glTranslatef(phy_header_angle,0,0);
glTranslatef(-100,0,0);
message_key();
glPopMatrix();
glPopMatrix();
}
glPushMatrix();
glTranslatef(movement_angle1,0,0);
if(phy_header_angle1==100)
```

```
{
glPushMatrix();
if(movement_angle1>=420)
{
// glRotatef(180,0,1,0);
glTranslatef(0,rev_analog_sig_angle,0);
}
glTranslatef(0,-analog_sig_angle,0);
if(rev_analog_sig_angle!=100)
cipher();
glPopMatrix();
}
glPopMatrix();
glPushMatrix();
glTranslatef(movement_angle1,0,0);
if(phy_header_angle1==100)
{
glPushMatrix();
glScalef(50,30,.5);
glTranslatef(2.5,2.5,0);
glutWireCube(2);
glPopMatrix();
}
glPopMatrix();
if(rev_analog_sig_angle==100)
{
glPushMatrix();
glTranslatef(450,rev_phy_header_angle1,0);
glPushMatrix();
//glTranslatef(0,rev_arrow_angle,0);
//glTranslatef(0,0,0);
```

```
message_data();
glPopMatrix();
glPushMatrix();
glTranslatef(rev_phy_header_angle,0,0);
glTranslatef(-10,0,0);
if(rev_phy_header_angle<200)
message_key();
glPopMatrix();
glPopMatrix();
}
glColor3f(1.0f,1.0f,1.0f);
glPushMatrix();
glTranslatef(450,0,0);
glScalef(70,40,.5);
glTranslatef(1.5,7,0);
glutWireCube(2);//right layer
glPopMatrix();
animation_encryp();
glFlush();
glutSwapBuffers();
}




void myinit()
{
glColor3f(1.0,0.0,0.0);
glPointSize(1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
```

```
gluOrtho2D(0.0,700.0,0.0,700.0);
}

void display()
{
glClearColor(0.0,0.0,0.0,0.0);
glClear(GL_COLOR_BUFFER_BIT);

front_page();
glFlush();
glutSwapBuffers();
}



void key (unsigned char key, int x, int y)
{
switch(key) {

case 'S' :
case 's' :glutDisplayFunc(encryption);
            break;
  case 'q':
  case 'Q':
        exit(0);
 }
 glutPostRedisplay();
}

static void menu(int mode)
{
switch (mode)
```

```
{
case FRONTPAGE:glutDisplayFunc(front_page);
break;
case ENCRYPTION:glutDisplayFunc(encryption);
break;
case EXIT: exit(0);
}
glutPostRedisplay();
}

void main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(1000,700);
glutInitWindowPosition(0,0);
glutCreateWindow("ENCRYPTION");

glutKeyboardFunc(key);

myinit();
glutDisplayFunc(display);
glutCreateMenu(menu);
glutAddMenuEntry("Front Page", FRONTPAGE);
glutAddMenuEntry("Encryption",ENCRYPTION);
glutAddMenuEntry("Exit", EXIT);
glutAttachMenu(GLUT_RIGHT_BUTTON);

glutMainLoop();
}
```