CG programs

2D

```
#include<iostream>
#include<math.h>
#include<GL/glut.h>

using namespace std;

void myinit(void)
{
    glClearColor(1,1,1,0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,100.0,0.0,100.0);
}

void line(int x1,int y1,int x2,int y2)
{
    //int x1=0,y1=0,x2=8,y2=6;
    int dx=abs(x2-x1),dy=abs(y2-y1);
    float e=2*dy;
    e=e-2*dx;
    int i=0,x=x1,y=y1,incx=1,incy=1;
    if(x2<x1) incx=-incx;
    if(y2<y1) incy=-incy;
    while(i<dx)
    {
        glVertex2f(x,y);
        while(e>0)
        {
            y=y+incy;
            e=e-2*dx;
        }
```

```
        x=x+incx;
        e=e+2*dy;
        i++;
    }

}
void display_DDA()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0,0.0,0.0);
    glPointSize(6.0);

    glBegin(GL_POINTS);

    line(0,27,18,0);

    glEnd();
    glFlush();
}

int main(int argc,char** argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowPosition(50,50);
    glutInitWindowSize(500,500);
    glutCreateWindow("DDA");
    myinit();
    glutDisplayFunc(display_DDA);

    glutMainLoop();
    return 0;


}
```

3D

```
#include<iostream>
#include<math.h>
#include<GL/glut.h>

#define PI 3.141592653589793

#define mode GL_LINE_LOOP
//#define mode GL_POLYGON

using namespace std;

float Xangle=0,Yangle=0,Zangle=0;
float xx=0,yy=0,zz=0;
float angle=0;

void home();
void road();
void windmill();
void bg();
void myinit()
{
    glClearColor(1,1,1,0);
    glEnable(GL_DEPTH_TEST);
    //glMatrixMode(GL_PROJECTION);
    //glOrth2D(0,100,0,100);
}

void resize(int w,int h)
{
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
```

```
    glFrustum(-5.0,5.0,-5.0,5.0,5.0,200.0);

    glMatrixMode(GL_MODELVIEW);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    glTranslatef(-20.0+xx,-20.0+yy,-50.0+zz);
    glRotatef(Xangle,1.0,0.0,0.0);
    glRotatef(Yangle,0.0,1.0,0.0);
    glRotatef(Zangle,0.0,0.0,1.0);



    glColor3f(1,0,0);

    windmill();
    glutSwapBuffers();
}

void windmill()
{
    glColor3f(0,0,1);
    glBegin(mode);
    glVertex3f(-20,0,0);
    glVertex3f(-15,0,0);
    glVertex3f(-17.5,40,-5);
    glEnd();
    glBegin(mode);
    glVertex3f(-20,0,0);
    glVertex3f(-20,0,-10);
    glVertex3f(-17.5,40,-5);
    glEnd();
    glBegin(mode);
    glVertex3f(-15,0,0);
```

```
    glVertex3f(-15,0,-10);
    glVertex3f(-17.5,40,-5);
    glEnd();
    glBegin(mode);
    glVertex3f(-20,0,-10);
    glVertex3f(-15,0,-10);
    glVertex3f(-17.5,40,-5);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3f(-17.5,40,-5);
    glVertex3f(-17.5,40,-4);
    glEnd();
    ////FAN
    float radius=10;
    glColor3f(1,0,0);
    for(float i=1;i<=3;i+=1)
    {
        float ang=angle+120*i;
        glBegin(mode);
        glVertex3f(-17.5,40,-5);

glVertex3f(-17.5+cos(ang*PI/180.0f)*radius,40+sin(ang*PI/180.0f)*radius,-5
);

glVertex3f(-17.5+cos((ang+30)*PI/180.0f)*radius,40+sin((ang+30)*PI/180.0f)
*radius,-5);
    glEnd();
    }
}
void bg()
{
    glBegin(GL_POLYGON);
    glVertex3f(0,0,0);
    glEnd();
}
```

```
void keyInput(unsigned char key,int x,int y)
{
    switch(key)
    {
        case 27:
            exit(0);
            break;
        case 'x':
            Xangle+=5.0;
            if(Xangle>360.0) Xangle-=360.0;
            glutPostRedisplay();
            break;
        case 'X':
            Xangle-=5.0;
            if(Xangle<0.0) Xangle+=360.0;
            glutPostRedisplay();
            break;
        case 'y':
            Yangle+=5.0;
            if(Yangle>360.0) Yangle-=360.0;
            glutPostRedisplay();
            break;
        case 'Y':
            Yangle-=5.0;
            if(Yangle<0.0) Yangle+=360.0;
            glutPostRedisplay();
            break;
        case 'z':
            Zangle+=5.0;
            if(Zangle>360.0) Zangle-=360.0;
            glutPostRedisplay();
            break;
        case 'Z':
            Zangle-=5.0;
            if(Zangle<0.0) Zangle+=360.0;
            glutPostRedisplay();
            break;
```

```
        case 'p':
            zz=zz-1;
            if(zz<=-45) zz=-44;
            glutPostRedisplay();
            break;
        case 'P':
            zz=zz+1;
            if(zz>=60) zz=59;
            glutPostRedisplay();
            break;

        default:
            break;
    }

}

void specialKeyInput(int key,int x,int y)
{
    switch(key)
    {
        case GLUT_KEY_UP:
                yy=yy+1;
                glutPostRedisplay();
                break;
        case GLUT_KEY_DOWN:
                yy=yy-1;
                glutPostRedisplay();
                break;
        case GLUT_KEY_LEFT:
                xx=xx-1;
                glutPostRedisplay();
                break;
        case GLUT_KEY_RIGHT:
                xx=xx+1;
                glutPostRedisplay();
                break;
```

```
    }

}

int main(int argc, char** argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowPosition(50,50);
    glutInitWindowSize(500,500);
    glutCreateWindow("CardBoard House");

    myinit();
    glutDisplayFunc(display);
    glutReshapeFunc(resize);
    glutKeyboardFunc(keyInput);
    glutSpecialFunc(specialKeyInput);

    glutMainLoop();


    return 0;
}
```

Animation

```
#include<iostream>
#include<math.h>
#include<GL/glut.h>

#define PI 3.141592653589793

#define mode GL_LINE_LOOP
//#define mode GL_POLYGON
```

```
using namespace std;

float Xangle=0,Yangle=0,Zangle=0;
float xx=0,yy=0,zz=0;
float angle=0;

void home();
void road();
void windmill();
void bg();
static bool isAnimate=0;
static int animatePeriod=100;

void animate(int value)
{
    if(isAnimate)
    {
        angle+=10;
        if(angle>=360)  angle-=360;

        glutPostRedisplay();
        glutTimerFunc(animatePeriod,animate,1);
    }
}

void myinit()
{
    glClearColor(1,1,1,0);
    glEnable(GL_DEPTH_TEST);
    //glMatrixMode(GL_PROJECTION);
    //glOrth2D(0,100,0,100);
}

void resize(int w,int h)
{
    glViewport(0,0,w,h);
```

```
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-5.0,5.0,-5.0,5.0,5.0,200.0);

    glMatrixMode(GL_MODELVIEW);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    glTranslatef(-20.0+xx,-20.0+yy,-50.0+zz);
    glRotatef(Xangle,1.0,0.0,0.0);
    glRotatef(Yangle,0.0,1.0,0.0);
    glRotatef(Zangle,0.0,0.0,1.0);

    home();
    road();
    windmill();
    bg();
    glutSwapBuffers();
}
void windmill()
{
    glColor3f(0,0,1);
    glBegin(mode);
    glVertex3f(-20,0,0);
    glVertex3f(-15,0,0);
    glVertex3f(-17.5,40,-5);
    glEnd();
    glBegin(mode);
    glVertex3f(-20,0,0);
    glVertex3f(-20,0,-10);
    glVertex3f(-17.5,40,-5);
    glEnd();
    glBegin(mode);
```

```
    glVertex3f(-15,0,0);
    glVertex3f(-15,0,-10);
    glVertex3f(-17.5,40,-5);
    glEnd();
    glBegin(mode);
    glVertex3f(-20,0,-10);
    glVertex3f(-15,0,-10);
    glVertex3f(-17.5,40,-5);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3f(-17.5,40,-5);
    glVertex3f(-17.5,40,-4);
    glEnd();
    ////FAN
    float radius=10;
    glColor3f(1,0,0);
    for(float i=1;i<=3;i+=1)
    {
        float ang=angle+120*i;
        glBegin(mode);
        glVertex3f(-17.5,40,-5);

glVertex3f(-17.5+cos(ang*PI/180.0f)*radius,40+sin(ang*PI/180.0f)*radius,-5
);

glVertex3f(-17.5+cos((ang+30)*PI/180.0f)*radius,40+sin((ang+30)*PI/180.0f)
*radius,-5);
    glEnd();
    }
}
void bg()
{
    glBegin(GL_POLYGON);
    glVertex3f(0,0,0);
    glEnd();
}
```

```
void keyInput(unsigned char key,int x,int y)
{
    switch(key)
    {
        case 27:
            exit(0);
            break;
        case ' ':
            if(isAnimate) isAnimate=0;
            else{
                isAnimate=1;
                animate(1);
            }
            break;

        case 'x':
            Xangle+=5.0;
            if(Xangle>360.0) Xangle-=360.0;
            glutPostRedisplay();
            break;
        case 'X':
            Xangle-=5.0;
            if(Xangle<0.0) Xangle+=360.0;
            glutPostRedisplay();
            break;
        case 'y':
            Yangle+=5.0;
            if(Yangle>360.0) Yangle-=360.0;
            glutPostRedisplay();
            break;
        case 'Y':
            Yangle-=5.0;
            if(Yangle<0.0) Yangle+=360.0;
            glutPostRedisplay();
            break;
```

```
        case 'z':
            Zangle+=5.0;
            if(Zangle>360.0) Zangle-=360.0;
            glutPostRedisplay();
            break;
        case 'Z':
            Zangle-=5.0;
            if(Zangle<0.0) Zangle+=360.0;
            glutPostRedisplay();
            break;
        case 'p':
            zz=zz-1;
            if(zz<=-45) zz=-44;
            glutPostRedisplay();
            break;
        case 'P':
            zz=zz+1;
            if(zz>=60) zz=59;
            glutPostRedisplay();
            break;

        default:
            break;
    }

}

void specialKeyInput(int key,int x,int y)
{
    switch(key)
    {
        case GLUT_KEY_UP:
                yy=yy+1;
                glutPostRedisplay();
                break;
        case GLUT_KEY_DOWN:
                yy=yy-1;
```

```
                glutPostRedisplay();
                break;
        case GLUT_KEY_LEFT:
                xx=xx-1;
                glutPostRedisplay();
                break;
        case GLUT_KEY_RIGHT:
                xx=xx+1;
                glutPostRedisplay();
                break;
    }

}

int main(int argc, char** argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowPosition(50,50);
    glutInitWindowSize(500,500);
    glutCreateWindow("CardBoard House");

    myinit();
    glutDisplayFunc(display);
    glutReshapeFunc(resize);
    glutKeyboardFunc(keyInput);
    glutSpecialFunc(specialKeyInput);

    glutMainLoop();


    return 0;
}
```