```cpp
#include <math.h>
#include "GLUT.h"
#include <time.h>
#include <stdio.h>

const int WIDTH = 600;
const int HEIGHT = 600;
const double PI = 4*atan(1.0);
const int GSIZE = 200;


double ground[GSIZE][GSIZE] = {0};

double offset=0;
double eyex=2,eyey=30,eyez=35;
double dz = 0, dx=0,dy=0;
double speed = 0;
double dirx=0,diry=0,dirz=-1;
double sight_angle = PI;

unsigned char tx0[256][256][4];
unsigned char tx1[128][1024][4];

unsigned char* bmp;


void LoadBitmap(char *fname)
{
    FILE* pf;
    BITMAPFILEHEADER bf;
    BITMAPINFOHEADER bi;
    int sz;

    pf = fopen(fname,"rb");
    fread(&bf,sizeof(BITMAPFILEHEADER),1,pf);
    fread(&bi,sizeof(BITMAPINFOHEADER),1,pf);

    sz = bi.biHeight*bi.biWidth*3;
    bmp = (unsigned char*) malloc(sz);

    fread(bmp,1,sz,pf);

    fclose(pf);
}


void SetTexture(int tnum)
{
    int i,j;
    int r;
    switch(tnum)
    {
    case 0: // wood
            for(i=0;i<256;i++)
                for(j=0;j<256;j++)
                {
                    tx0[255-i][j][0] = bmp[3*(256*i+j)+2]; // red
                    tx0[255-i][j][1] = bmp[3*(256*i+j)+1]; // green
                    tx0[255-i][j][2] = bmp[3*(256*i+j)]; // blue
                    tx0[255-i][j][3] = 0; // alpha
                }
        break;
    case 1: // Afeka
            for(i=0;i<128;i++)
                for(j=0;j<1024;j++)
                {
                    tx1[i][j][0] = bmp[3*(1024*i+j)+2]; // red
                    tx1[i][j][1] = bmp[3*(1024*i+j)+1]; // green
                    tx1[i][j][2] = bmp[3*(1024*i+j)]; // blue
                    tx1[i][j][3] = 0; // alpha
                }
        break;
    }
}
void init()
{
    glClearColor(0.7,0.7,1,1);
    glEnable(GL_DEPTH_TEST); // set objects by their depth
    srand(time(0));

    LoadBitmap("wood.bmp");
    SetTexture(0); // wood
    glBindTexture(GL_TEXTURE_2D,0);
    glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
    glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
    glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT);
    glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT);
    glTexImage2D(GL_TEXTURE_2D,0,GL_RGBA,256,256,0,GL_RGBA,
        GL_UNSIGNED_BYTE,tx0);

    LoadBitmap("afeka.bmp");
    SetTexture(1); // Afeka
    glBindTexture(GL_TEXTURE_2D,1);
    glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
    glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
    glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT);
    glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT);
    glTexImage2D(GL_TEXTURE_2D,0,GL_RGBA,1024,128,0,GL_RGBA,
        GL_UNSIGNED_BYTE,tx1);

}


void DrawCilynder(int n)
{
    double alpha,teta;

    teta = 2*PI/n;

    for(alpha = 0; alpha<2*PI;alpha+=teta)
    {
        glBegin(GL_POLYGON);
```

```
                    glColor3d(fabs(sin(alpha)),fabs(cos(alpha)),(1+sin(alpha*3))/2);
            glVertex3d(sin(alpha),1,cos(alpha));
                    glColor3d(fabs(cos(alpha-teta)),fabs(sin(alpha-teta)),(1+sin((alpha-teta)*3))/2);
            glVertex3d(sin(alpha+teta),1,cos(alpha+teta));
                    glColor3d(fabs(cos(alpha-teta)),fabs(sin(alpha-teta)),(1+sin((alpha-teta)*3))/2);
        glVertex3d(sin(alpha+teta),0,cos(alpha+teta));
                glColor3d(fabs(sin(alpha)),fabs(cos(alpha)),(1+sin(alpha*3))/2);
        glVertex3d(sin(alpha),0,cos(alpha));
        glEnd();
    }
}
void DrawCilynder1(int n, double topr, double bottomr)
{
    double alpha,teta;

    teta = 2*PI/n;

    for(alpha = 0; alpha<2*PI;alpha+=teta)
    {
        glBegin(GL_POLYGON);
                    glColor3d(fabs(sin(alpha)),fabs(cos(alpha)),(1+sin(alpha*3))/2);
            glVertex3d(topr*sin(alpha),1,topr*cos(alpha));
            glVertex3d(topr*sin(alpha+teta),1,topr*cos(alpha+teta));
                    glColor3d(0,fabs(cos(alpha-teta)),fabs(sin(alpha-teta)));
            glVertex3d(bottomr*sin(alpha+teta),0,bottomr*cos(alpha+teta));
            glVertex3d(bottomr*sin(alpha),0,bottomr*cos(alpha));
        glEnd();
    }
}

void DrawTexCilynder1(int n,double topr, double bottomr,int tnum, int horiz_rep,
    int vert_rep,bool color)
{
    double teta = 2*PI/n;
    double alpha;
    int c;
    double part = horiz_rep/(double)n;

    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D,tnum);
    if(color)
    glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_MODULATE);
    else
    glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_REPLACE);


    for(alpha = 0,c=0; alpha<2*PI;alpha+=teta,c++)
    {
        if(alpha<PI/2 || alpha>3*PI/2 )
            glColor3d(0.4+0.5*cos(alpha),0.4+0.5*cos(alpha),0.4+0.5*cos
                (alpha));
            else
                glColor3d(0.4,0.4,0.4);

//          glNormal3d(sin(alpha),tan(bottomr-topr),cos(alpha));
            glBegin(GL_POLYGON);
if(topr!=0)
{
            glTexCoord2d(c*part,0);
                glVertex3d(topr*sin(alpha),1,topr*cos(alpha)); // 1
//          glNormal3d(sin(alpha+teta),tan(bottomr-topr),cos(alpha+teta));
            glTexCoord2d((c+1)*part,0);
                glVertex3d(topr*sin(alpha+teta),1,topr*cos(alpha+teta)); // 2
}
else
{
                glTexCoord2d(c*part+0.5,0);
                glVertex3d(0,1,0); // 1
}
                glTexCoord2d((c+1)*part,vert_rep);
                glVertex3d(bottomr*sin(alpha+teta),0,bottomr*cos(alpha+teta)); //
                  3
//          glNormal3d(sin(alpha),tan(bottomr-topr),cos(alpha));

            glTexCoord2d(c*part,vert_rep);
                glVertex3d(bottomr*sin(alpha),0,bottomr*cos(alpha)); // 4
            glEnd();
        }
        glDisable(GL_TEXTURE_2D);

}
void DrawTexCilynder2(int n,double topr, double bottomr,int tnum, int
  horiz_rep,
    int vert_rep,bool color,double start_angle, double stop_angle)
{
    double teta = 2*PI/n;
    double alpha;
    int c;
    double part = horiz_rep/(double)n;

    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D,tnum);
    if(color)
    glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_MODULATE);
    else
    glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_REPLACE);


    for(alpha = start_angle,c=0; alpha<stop_angle;alpha+=teta,c++)
    {
        if(alpha<PI/2 || alpha>3*PI/2 )
            glColor3d(0.4+0.5*cos(alpha),0.4+0.5*cos(alpha),0.4+0.5*cos
              (alpha));
        else
            glColor3d(0.4,0.4,0.4);

```

```
//          glNormal3d(sin(alpha),tan(bottomr-topr),cos(alpha));
        glBegin(GL_POLYGON);
    if(tnum==1)     glTexCoord2d(0,c*part);
    else        glTexCoord2d(c*part,0);
    glVertex3d(topr*sin(alpha),1,topr*cos(alpha)); // 1
//          glNormal3d(sin(alpha+teta),tan(bottomr-topr),cos(alpha+teta));
    if(tnum==1) glTexCoord2d(0,(c+1)*part);
        else glTexCoord2d((c+1)*part,0);
            glVertex3d(topr*sin(alpha+teta),1,topr*cos(alpha+teta)); // 2
        if(tnum==1) glTexCoord2d(vert_rep,(c+1)*part);
        else glTexCoord2d((c+1)*part,vert_rep);
            glVertex3d(bottomr*sin(alpha+teta),0,bottomr*cos(alpha+teta)); // 3
//          glNormal3d(sin(alpha),tan(bottomr-topr),cos(alpha));
    if(tnum==1) glTexCoord2d(vert_rep,c*part);
        else glTexCoord2d(c*part,vert_rep);
            glVertex3d(bottomr*sin(alpha),0,bottomr*cos(alpha)); // 4
        glEnd();
    }
    glDisable(GL_TEXTURE_2D);

}

void DrawSphere(int sectors, int slices)
{
    double phi = PI/slices,beta;
    double tr,br; // topr, bottomr
    double height;

    for(beta=-PI/2;beta<PI/2;beta+=phi)
    {
        br = cos(beta);
        tr = cos(beta+phi);
        height = fabs(sin(beta)-sin(beta+phi));
        glPushMatrix();
            glTranslated(0,sin(beta),0);
            glScaled(1,height,1);
            DrawCilynder1(sectors,tr,br);
        glPopMatrix();
    }
}

void DrawSphere1(int sectors, int slices, double start, double stop,bool color)
{
    double phi = PI/slices,beta;
    double tr,br; // topr, bottomr
    double height;

    for(beta=start;beta<stop-phi;beta+=phi)
    {
        br = cos(beta);
        tr = cos(beta+phi);
        height = fabs(sin(beta)-sin(beta+phi));
        glPushMatrix();
            glTranslated(0,sin(beta),0);
```

```cpp
            glScaled(1,height,1);
            DrawTexCilynder1(sectors,tr,br,0,sectors,1,color);
        glPopMatrix();
    }
}

void DrawWater()
{
    int i,j;
    for(i=2;i<GSIZE;i++)
        for(j=2;j<GSIZE;j++)
        {

        glBegin(GL_POLYGON);
            glColor3d((1+ground[i][j])/4,(1+ground[i][j])/3,(1+ground[i][j])/2);
                glVertex3d(j-GSIZE/2,ground[i][j],i-GSIZE/2);
            glColor3d((1+ground[i][j-1])/4,(1+ground[i][j-1])/3,(1+ground[i][j-1])/2);
                glVertex3d(j-GSIZE/2-1,ground[i][j-1],i-GSIZE/2);
            glColor3d((1+ground[i-1][j-1])/4,(1+ground[i-1][j-1])/3,(1+ground[i-1][j-1])/2);
                glVertex3d(j-GSIZE/2-1,ground[i-1][j-1],i-GSIZE/2-1);
            glColor3d((1+ground[i-1][j])/4,(1+ground[i-1][j])/3,(1+ground[i-1][j])/2);
                glVertex3d(j-GSIZE/2,ground[i-1][j],i-GSIZE/2-1);
        glEnd();
        }
}


void DrawShip()
{
    glPushMatrix();
    glTranslated(0,15,0);
    glScaled(10,20,30);
    DrawSphere1(7,10,-PI/2,0,true);
    glPopMatrix();

    glPushMatrix();
        glTranslated(0,7,0);
        glScaled(10,20,30);
        glScaled(0.95,0,0.95);
        DrawSphere1(7,10,-PI/2,0,false);
    glPopMatrix();

    glPushMatrix();
        glTranslated(0,0,-11);
        glPushMatrix(); //sail 1
            glTranslated(10,20,0);
            glRotated(-90,0,1,0);
            glRotated(90,1,0,0);
            glScaled(4,20,6);
                DrawTexCilynder2(60,1,1,1,2,1,true,0,2*PI);
        glPopMatrix();
        glPushMatrix(); //sail2
```

```
                glTranslated(8,31,0);
                glRotated(-90,0,1,0);
                glRotated(90,1,0,0);
                glScaled(3,16,4);
                    DrawTexCilynder2(60,1,1,0,2,1,true,0,PI);
            glPopMatrix();
            glPushMatrix(); //sail3
                glTranslated(4,38,0);
                glRotated(-90,0,1,0);
                glRotated(90,1,0,0);
                glScaled(2,8,2);
                        DrawTexCilynder2(60,1,1,1,2,1,true,0,PI);
            glPopMatrix();
            glScaled(1,43,1);
            DrawTexCilynder1(20,0.1,1,0,1,1,true);
        glPopMatrix();

        glPushMatrix();
            glTranslated(0,0,13);
            glPushMatrix(); //sail 1
                glTranslated(10,20,0);
                glRotated(-90,0,1,0);
                glRotated(90,1,0,0);
                glScaled(4,20,6);
                    DrawTexCilynder2(60,1,1,1,2,1,true,0,PI);
            glPopMatrix();
            glPushMatrix(); //sail2
                glTranslated(8,31,0);
                glRotated(-90,0,1,0);
                glRotated(90,1,0,0);
                glScaled(3,16,4);
                    DrawTexCilynder2(60,1,1,1,2,1,true,0,PI);
            glPopMatrix();
            glPushMatrix(); //sail3
                glTranslated(4,38,0);
                glRotated(-90,0,1,0);
                glRotated(90,1,0,0);
                glScaled(2,8,2);
                    DrawTexCilynder2(60,1,1,1,2,1,true,0,PI);
            glPopMatrix();
            glScaled(1,43,1);
            DrawTexCilynder1(20,0.1,1,0,1,1,true);
        glPopMatrix();


    }

    void display()
    {
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glFrustum(-1,1,-1,1,0.7,300);
        gluLookAt(eyex,eyey,eyez,
            eyex+dirx,eyey-0.5,eyez+dirz,0,1,0);

        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity(); // starts transformations from 0

        glRotated(2*offset,0,1,0);

        DrawWater();
        glPushMatrix();
            glTranslated(0,sin(0.07*GSIZE/2+0.3*offset),0);
            glRotated(-0.07*cos(0.07*GSIZE/2+0.3*offset)*180/PI,1,0,0);
            DrawShip();
        glPopMatrix();

        glutSwapBuffers();
    }

    void idle()
    {
        int i,j;
        offset+=0.1;

        for(i=0;i<GSIZE;i++)
            for(j=0;j<GSIZE;j++)
                ground[i][j] = sin(0.07*i+0.3*offset);

        dirx = sin(sight_angle);
        dirz= cos(sight_angle);
        eyez+=speed*dirz;
        eyex+=speed*dirx;
        eyey+=dy;

        glutPostRedisplay();
    }



    void special(int key, int x, int y)
    {
        switch(key)
        {
        case GLUT_KEY_UP:
            speed+=0.001;
            break;
        case GLUT_KEY_DOWN:
            speed-=0.001;
            break;
        case GLUT_KEY_LEFT:
            sight_angle+=0.01;
            break;
        case GLUT_KEY_RIGHT:
            sight_angle-=0.01;
            break;
        case GLUT_KEY_PAGE_UP:
            dy+=0.001;
            break;
        case GLUT_KEY_PAGE_DOWN:
            dy-=0.001;
```

```
            break;

        }
}


void main(int argc, char* argv[])
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE|GLUT_DEPTH);
    glutInitWindowSize(600,600);
    glutInitWindowPosition(100,100);
    glutCreateWindow("exam");

    glutDisplayFunc(display);
    glutIdleFunc(idle);
    glutSpecialFunc(special);

    init();
    glutMainLoop();
}

```