

```
1  #include <math.h>
2  #include "GLUT.h"
3  #include <time.h>
4  #include <stdio.h>
5
6  const int WIDTH = 600;
7  const int HEIGHT = 600;
8  const double PI = 4*atan(1.0);
9  const int GSIZE = 200;
10
11
12 double ground[GSIZE][GSIZE] = {0};
13
14 double offset=0;
15 double eyex=2,eyey=30,eyez=35;
16 double dz = 0, dx=0,dy=0;
17 double speed = 0;
18 double dirx=0,diry=0,dirz=-1;
19 double sight_angle = PI;
20
21 unsigned char tx0[256][256][4];
22 unsigned char tx1[128][1024][4];
23
24 unsigned char* bmp;
25
26
27 void LoadBitmap(char *fname)
28 {
29     FILE* pf;
30     BITMAPFILEHEADER bf;
31     BITMAPINFOHEADER bi;
32     int sz;
33
34     pf = fopen(fname,"rb");
35     fread(&bf,sizeof(BITMAPFILEHEADER),1,pf);
36     fread(&bi,sizeof(BITMAPINFOHEADER),1,pf);
37
38     sz = bi.biHeight*bi.biWidth*3;
39     bmp = (unsigned char*) malloc(sz);
40
41     fread(bmp,1,sz,pf);
42
43     fclose(pf);
44 }
45
46
47 void SetTexture(int tnum)
48 {
49     int i,j;
50     int r;
51     switch(tnum)
52     {
53     case 0: // wood
54         for(i=0;i<256;i++)
55             for(j=0;j<256;j++)
56                 {
```

```

57         tx0[255-i][j][0] = bmp[3*(256*i+j)+2]; // red
58         tx0[255-i][j][1] = bmp[3*(256*i+j)+1]; // green
59         tx0[255-i][j][2] = bmp[3*(256*i+j)]; // blue
60         tx0[255-i][j][3] = 0; // alpha
61     }
62     break;
63     case 1: // Afeka
64         for(i=0;i<128;i++)
65             for(j=0;j<1024;j++)
66             {
67                 tx1[i][j][0] = bmp[3*(1024*i+j)+2]; // red
68                 tx1[i][j][1] = bmp[3*(1024*i+j)+1]; // green
69                 tx1[i][j][2] = bmp[3*(1024*i+j)]; // blue
70                 tx1[i][j][3] = 0; // alpha
71             }
72     break;
73 }
74 }
75 void init()
76 {
77     glClearColor(0.7,0.7,1,1);
78     glEnable(GL_DEPTH_TEST); // set objects by their depth
79     srand(time(0));
80
81     LoadBitmap("wood.bmp");
82     SetTexture(0); // wood
83     glBindTexture(GL_TEXTURE_2D,0);
84     glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
85     glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
86     glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT);
87     glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT);
88     glTexImage2D(GL_TEXTURE_2D,0,GL_RGBA,256,256,0,GL_RGBA,
89                 GL_UNSIGNED_BYTE,tx0);
90
91     LoadBitmap("afeka.bmp");
92     SetTexture(1); // Afeka
93     glBindTexture(GL_TEXTURE_2D,1);
94     glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
95     glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
96     glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT);
97     glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT);
98     glTexImage2D(GL_TEXTURE_2D,0,GL_RGBA,1024,128,0,GL_RGBA,
99                 GL_UNSIGNED_BYTE,tx1);
100
101 }
102
103
104 void DrawCylinder(int n)
105 {
106     double alpha,teta;
107
108     teta = 2*PI/n;
109
110     for(alpha = 0; alpha<2*PI;alpha+=teta)
111     {
112         glBegin(GL_POLYGON);

```

```

...p\old_exams\graphics_Winter2016B_solution\exam\exam.cpp 3
113         glColor3d(fabs(sin(alpha)),fabs(cos(alpha)),(1+sin  ↗
            (alpha*3))/2);
114         glVertex3d(sin(alpha),1,cos(alpha));
115         glColor3d(fabs(cos(alpha-teta)),fabs(sin(alpha-teta)),(1  ↗
            +sin((alpha-teta)*3))/2);
116         glVertex3d(sin(alpha+teta),1,cos(alpha+teta));
117         glColor3d(fabs(cos(alpha-teta)),fabs(sin(alpha-teta)),(1  ↗
            +sin((alpha-teta)*3))/2);
118         glVertex3d(sin(alpha+teta),0,cos(alpha+teta));
119         glColor3d(fabs(sin(alpha)),fabs(cos(alpha)),(1+sin  ↗
            (alpha*3))/2);
120         glVertex3d(sin(alpha),0,cos(alpha));
121         glEnd();
122     }
123 }
124 void DrawCylinder1(int n, double topr, double bottomr)
125 {
126     double alpha,teta;
127
128     teta = 2*PI/n;
129
130     for(alpha = 0; alpha<2*PI;alpha+=teta)
131     {
132         glBegin(GL_POLYGON);
133         glColor3d(fabs(sin(alpha)),fabs(cos(alpha)),(1+sin  ↗
            (alpha*3))/2);
134         glVertex3d(topr*sin(alpha),1,topr*cos(alpha));
135         glVertex3d(topr*sin(alpha+teta),1,topr*cos(alpha+teta));
136         glColor3d(0,fabs(cos(alpha-teta)),fabs(sin(alpha-teta)));
137         glVertex3d(bottomr*sin(alpha+teta),0,bottomr*cos(alpha+teta));
138         glVertex3d(bottomr*sin(alpha),0,bottomr*cos(alpha));
139         glEnd();
140     }
141 }
142
143 void DrawTexCylinder1(int n,double topr, double bottomr,int tnum, int  ↗
    horiz_rep,
144     int vert_rep,bool color)
145 {
146     double teta = 2*PI/n;
147     double alpha;
148     int c;
149     double part = horiz_rep/(double)n;
150
151     glEnable(GL_TEXTURE_2D);
152     glBindTexture(GL_TEXTURE_2D,tnum);
153     if(color)
154         glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_MODULATE);
155     else
156         glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_REPLACE);
157
158
159     for(alpha = 0,c=0; alpha<2*PI;alpha+=teta,c++)
160     {
161         if(alpha<PI/2 || alpha>3*PI/2 )
162             glColor3d(0.4+0.5*cos(alpha),0.4+0.5*cos(alpha),0.4+0.5*cos  ↗

```

```

        (alpha));
163     else
164         glColor3d(0.4,0.4,0.4);
165
166     //         glNormal3d(sin(alpha),tan(bottomr-topr),cos(alpha));
167     glBegin(GL_POLYGON);
168     if(topr!=0)
169     {
170         glTexCoord2d(c*part,0);
171         glVertex3d(topr*sin(alpha),1,topr*cos(alpha)); // 1
172     //         glNormal3d(sin(alpha+teta),tan(bottomr-topr),cos(alpha+teta));
173         glTexCoord2d((c+1)*part,0);
174         glVertex3d(topr*sin(alpha+teta),1,topr*cos(alpha+teta)); // 2
175     }
176     else
177     {
178         glTexCoord2d(c*part+0.5,0);
179         glVertex3d(0,1,0); // 1
180     }
181     glTexCoord2d((c+1)*part,vert_rep);
182     glVertex3d(bottomr*sin(alpha+teta),0,bottomr*cos(alpha+teta)); // 3
183     //         glNormal3d(sin(alpha),tan(bottomr-topr),cos(alpha));
184
185     glTexCoord2d(c*part,vert_rep);
186     glVertex3d(bottomr*sin(alpha),0,bottomr*cos(alpha)); // 4
187     glEnd();
188 }
189 glDisable(GL_TEXTURE_2D);
190
191 }
192 void DrawTexCylinder2(int n,double topr, double bottomr,int tnum, int
    horiz_rep,
193     int vert_rep,bool color,double start_angle, double stop_angle)
194 {
195     double teta = 2*PI/n;
196     double alpha;
197     int c;
198     double part = horiz_rep/(double)n;
199
200     glEnable(GL_TEXTURE_2D);
201     glBindTexture(GL_TEXTURE_2D,tnum);
202     if(color)
203         glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_MODULATE);
204     else
205         glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_REPLACE);
206
207
208     for(alpha = start_angle,c=0; alpha<stop_angle;alpha+=teta,c++)
209     {
210         if(alpha<PI/2 || alpha>3*PI/2 )
211             glColor3d(0.4+0.5*cos(alpha),0.4+0.5*cos(alpha),0.4+0.5*cos
                (alpha));
212         else
213             glColor3d(0.4,0.4,0.4);
214

```

```

215 //      glNormal3d(sin(alpha),tan(bottomr-topr),cos(alpha));
216      glBegin(GL_POLYGON);
217      if(tnum==1)      glTexCoord2d(0,c*part);
218      else      glTexCoord2d(c*part,0);
219      glVertex3d(topr*sin(alpha),1,topr*cos(alpha)); // 1
220 //      glNormal3d(sin(alpha+teta),tan(bottomr-topr),cos(alpha+teta));
221      if(tnum==1) glTexCoord2d(0,(c+1)*part);
222      else glTexCoord2d((c+1)*part,0);
223      glVertex3d(topr*sin(alpha+teta),1,topr*cos(alpha+teta)); // 2
224      if(tnum==1) glTexCoord2d(vert_rep,(c+1)*part);
225      else glTexCoord2d((c+1)*part,vert_rep);
226      glVertex3d(bottomr*sin(alpha+teta),0,bottomr*cos(alpha+teta)); // 3
227 //      glNormal3d(sin(alpha),tan(bottomr-topr),cos(alpha));
228      if(tnum==1) glTexCoord2d(vert_rep,c*part);
229      else glTexCoord2d(c*part,vert_rep);
230      glVertex3d(bottomr*sin(alpha),0,bottomr*cos(alpha)); // 4
231      glEnd();
232  }
233  glDisable(GL_TEXTURE_2D);
234
235 }
236
237 void DrawSphere(int sectors, int slices)
238 {
239     double phi = PI/slices,beta;
240     double tr,br; // topr, bottomr
241     double height;
242
243     for(beta=-PI/2;beta<PI/2;beta+=phi)
244     {
245         br = cos(beta);
246         tr = cos(beta+phi);
247         height = fabs(sin(beta)-sin(beta+phi));
248         glPushMatrix();
249             glTranslated(0,sin(beta),0);
250             glScaled(1,height,1);
251             DrawCylinder1(sectors,tr,br);
252         glPopMatrix();
253     }
254 }
255
256 void DrawSphere1(int sectors, int slices, double start, double stop,bool
257 color)
258 {
259     double phi = PI/slices,beta;
260     double tr,br; // topr, bottomr
261     double height;
262
263     for(beta=start;beta<stop-phi;beta+=phi)
264     {
265         br = cos(beta);
266         tr = cos(beta+phi);
267         height = fabs(sin(beta)-sin(beta+phi));
268         glPushMatrix();
269             glTranslated(0,sin(beta),0);

```

```

269         glScaled(1,height,1);
270         DrawTexCylinder1(sectors,tr,br,0,sectors,1,color);
271         glPopMatrix();
272     }
273 }
274
275 void DrawWater()
276 {
277     int i,j;
278     for(i=2;i<GSIZE;i++)
279         for(j=2;j<GSIZE;j++)
280         {
281
282             glBegin(GL_POLYGON);
283             glColor3d((1+ground[i][j])/4,(1+ground[i][j])/3,(1+ground[i]  ↗
284                 [j])/2);
285             glVertex3d(j-GSIZE/2,ground[i][j],i-GSIZE/2);
286             glColor3d((1+ground[i][j-1])/4,(1+ground[i][j-1])/3,(1+ground[i]  ↗
287                 [j-1])/2);
288             glVertex3d(j-GSIZE/2-1,ground[i][j-1],i-GSIZE/2);
289             glColor3d((1+ground[i-1][j-1])/4,(1+ground[i-1][j-1])/3,(1+ground  ↗
290                 [i-1][j-1])/2);
291             glVertex3d(j-GSIZE/2-1,ground[i-1][j-1],i-GSIZE/2-1);
292             glColor3d((1+ground[i-1][j])/4,(1+ground[i-1][j])/3,(1+ground[i-1]  ↗
293                 [j])/2);
294             glVertex3d(j-GSIZE/2,ground[i-1][j],i-GSIZE/2-1);
295             glEnd();
296         }
297     }
298 }
299
300 void DrawShip()
301 {
302     glPushMatrix();
303     glTranslated(0,15,0);
304     glScaled(10,20,30);
305     DrawSphere1(7,10,-PI/2,0,true);
306     glPopMatrix();
307
308     glPushMatrix();
309     glTranslated(0,7,0);
310     glScaled(10,20,30);
311     glScaled(0.95,0,0.95);
312     DrawSphere1(7,10,-PI/2,0,false);
313     glPopMatrix();
314
315     glPushMatrix();
316     glTranslated(0,0,-11);
317     glPushMatrix(); //sail 1
318     glTranslated(10,20,0);
319     glRotated(-90,0,1,0);
320     glRotated(90,1,0,0);
321     glScaled(4,20,6);
322     DrawTexCylinder2(60,1,1,1,2,1,true,0,2*PI);
323     glPopMatrix();
324     glPushMatrix(); //sail2

```

```
321         glTranslated(8,31,0);
322         glRotated(-90,0,1,0);
323         glRotated(90,1,0,0);
324         glScaled(3,16,4);
325         DrawTexCylinder2(60,1,1,0,2,1,true,0,PI);
326     glPopMatrix();
327     glPushMatrix(); //sail3
328         glTranslated(4,38,0);
329         glRotated(-90,0,1,0);
330         glRotated(90,1,0,0);
331         glScaled(2,8,2);
332         DrawTexCylinder2(60,1,1,1,2,1,true,0,PI);
333     glPopMatrix();
334     glScaled(1,43,1);
335     DrawTexCylinder1(20,0.1,1,0,1,1,true);
336 glPopMatrix();
337
338 glPushMatrix();
339     glTranslated(0,0,13);
340     glPushMatrix(); //sail 1
341         glTranslated(10,20,0);
342         glRotated(-90,0,1,0);
343         glRotated(90,1,0,0);
344         glScaled(4,20,6);
345         DrawTexCylinder2(60,1,1,1,2,1,true,0,PI);
346     glPopMatrix();
347     glPushMatrix(); //sail2
348         glTranslated(8,31,0);
349         glRotated(-90,0,1,0);
350         glRotated(90,1,0,0);
351         glScaled(3,16,4);
352         DrawTexCylinder2(60,1,1,1,2,1,true,0,PI);
353     glPopMatrix();
354     glPushMatrix(); //sail3
355         glTranslated(4,38,0);
356         glRotated(-90,0,1,0);
357         glRotated(90,1,0,0);
358         glScaled(2,8,2);
359         DrawTexCylinder2(60,1,1,1,2,1,true,0,PI);
360     glPopMatrix();
361     glScaled(1,43,1);
362     DrawTexCylinder1(20,0.1,1,0,1,1,true);
363 glPopMatrix();
364
365
366 }
367
368 void display()
369 {
370     glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
371     glMatrixMode(GL_PROJECTION);
372     glLoadIdentity();
373     glFrustum(-1,1,-1,1,0.7,300);
374     gluLookAt(eyex,eyey,eyez,
375             eyex+dirx,eyey-0.5,eyez+dirz,0,1,0);
376 }
```

```
377     glMatrixMode(GL_MODELVIEW);
378     glLoadIdentity(); // starts transformations from 0
379
380     glRotated(2*offset,0,1,0);
381
382     DrawWater();
383     glPushMatrix();
384         glTranslated(0,sin(0.07*GSIZE/2+0.3*offset),0);
385         glRotated(-0.07*cos(0.07*GSIZE/2+0.3*offset)*180/PI,1,0,0);
386         DrawShip();
387     glPopMatrix();
388
389     glutSwapBuffers();
390 }
391
392 void idle()
393 {
394     int i,j;
395     offset+=0.1;
396
397     for(i=0;i<GSIZE;i++)
398         for(j=0;j<GSIZE;j++)
399             ground[i][j] = sin(0.07*i+0.3*offset);
400
401     dirx = sin(sight_angle);
402     dirz= cos(sight_angle);
403     eyez+=speed*dirz;
404     eyex+=speed*dirx;
405     eyey+=dy;
406
407     glutPostRedisplay();
408 }
409
410
411
412 void special(int key, int x, int y)
413 {
414     switch(key)
415     {
416     case GLUT_KEY_UP:
417         speed+=0.001;
418         break;
419     case GLUT_KEY_DOWN:
420         speed-=0.001;
421         break;
422     case GLUT_KEY_LEFT:
423         sight_angle+=0.01;
424         break;
425     case GLUT_KEY_RIGHT:
426         sight_angle-=0.01;
427         break;
428     case GLUT_KEY_PAGE_UP:
429         dy+=0.001;
430         break;
431     case GLUT_KEY_PAGE_DOWN:
432         dy-=0.001;
```



```
433         break;
434
435     }
436 }
437
438
439 void main(int argc, char* argv[])
440 {
441     glutInit(&argc,argv);
442     glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE|GLUT_DEPTH);
443     glutInitWindowSize(600,600);
444     glutInitWindowPosition(100,100);
445     glutCreateWindow("exam");
446
447     glutDisplayFunc(display);
448     glutIdleFunc(idle);
449     glutSpecialFunc(special);
450
451     init();
452     glutMainLoop();
453 }
454
```