

Computer Graphics (UCS505)

Project on
The Lighthouse

Submitted by

Sanyam Sood 102003163

COE 7

B.E. Third Year COE

Submitted to:

Dr. Anupam Garg



Computer Science and Engineering Department

Thapar Institute of Engineering & Technology

(Deemed to be University), Patiala, Punjab - 147004

India

Table of Contents

S.No.	Description	Page No.
1	Introduction to the Project	2
2	Computer Graphics concepts used	3
3	User defined functions	4
4	Code	5
5	Output/Screenshots	11

Introduction to the Project

Movies. Who doesn't like movies? Even in our adulthood, we get amazed at scenes like waves hitting against a lighthouse on a stormy night. Inspired by such fascinating visual experiences, our group presents a unique Computer Graphics project – The Lighthouse.

In this project, we have built a lighthouse along the sea shore which works in a simulation of patrolling and guiding in the night and being the centre of attraction for people visiting the beach.

We have used polygons to make the structure of the lighthouse, land, sea and sky. To give the sea a flowing effect at the shore, we made the corner point of the sea polygon move in a sin wave.

We have made a bright yellow sun for the day using Bresenham Circle Drawing Algorithm and filled it with subsequent circles of decreasing radii. For the night we have beautified our sky with twinkling stars which we have simply achieved by changing the colour of pixels at the time of refreshing our buffer.

As for our pride, the guiding light of our lighthouse is another polygon originating from the hole in the lighthouse and translating back and forth continuously with increasing gap between the end points of light at its farthest and decreasing the gap at its nearest.

With this basic idea, let us dive into the actual working and concepts used to make our imagination come true.

Computer Graphics concepts used

OpenGL and GLUT: The project is implemented using OpenGL, a widely-used cross-platform graphics API, and the Graphics Library Utility Toolkit (GLUT), a user interface library, to create an efficient and portable application compatible with multiple platforms.

Output Primitives: The project uses various output primitives, such as triangles, circles, polygons, and rectangles, to create the various shapes and objects found within the lighthouse environment.

Animation: Our day and night switching feature gives a very impressive animation experience which can be controlled via keyboard keys.

Transformation: Transformation techniques, such as translation is used to position and orient objects like our light.

Logical Programming: The project uses logics to create the various shapes and objects found within the lighthouse environment.

User Input: The project incorporates user input to make it interactive, allowing users to control the movement of the boat and switching between daytime/night time.

Circle Drawing Algorithm: The project uses Bresenham Circle Drawing Algorithm for designing the circular components of the project such as the Sun/Moon, and the black circle on top of the lighthouse.

User-defined functions

Function	Description
Land ()	To design the land polygon using glBegin(GL_POLYGON); it includes if-else conditions based on which the color of the land changes according to day and night.
Lighthouse ()	To design the lighthouse; it consists of various glBegin(GL_POLYGON) functions for drawing different polygonal parts of the lighthouse, while also having if-else conditions to change their color according to day and night time.
SunorMoon ()	To design the Sun/Moon using Bresenham's Circle Drawing Algorithm. It contains an if-else condition to draw the Sun or Moon according to the day and night time.
light ()	To design the guiding light of the lighthouse using glBegin(GL_POLYGON); it includes a SetPixel() function that has the coordinates of the polygon which are variables that change with time and hence the dimensions of the yellow light changes
star ()	To draw the stars for the night sky, The function is used various times to draw multiple stars and has their colour as a variable, which on changing, causes a twinkling effect.
Water ()	To draw the water polygon; its colour for the day and night is again changed with the help of if-else condition, while one corner of the polygon is moved in a sine wave to make it feel like flowing water.
sky ()	To design the sky polygon; if-else conditions are used to vary colours for day and night time.
boat ()	To draw the boat, Using glBegin(GL_POLYGON).
boatleft ()	To move the boat towards the left, which is called when the users presses certain button
boatright ()	To move the boat towards the right, which is called when the users presses certain button
keyboard ()	Includes switch cases to toggle between day and night modes, and to change the direction of the boat (left/right).

Code

```
1  #include <gl/GLUT.h>
2  #include <iostream>
3  #include <math.h>
4  #include <cmath>
5  using namespace std;
6
7  // Water parameters
8  float waterHeight = 290.0;           // Height of water surface
9  float waveSpeed = 0.1;               // Speed of waves
10 float waveAmplitude = 10.0;          // Amplitude of waves
11 float waveFrequency = 0.025;         // Frequency of waves
12 float M_PI = 3.14159265358979323846; // pi
13 float xpos = 500;                    // Starting position of Light
14 float d = 50;                        // Thickness of Light
15 bool moving_right = true;             // For movement of Light
16 int delay = 0;                       // Delay for Light on ends
17 int n = 10;                          // Number of Stars
18 float col[10];                       // Colour of Stars
19 bool isDayMode = true;
20 float boatx = 0.0;
21
22 void SetPixel(int x, int y) {
23     glVertex2f(x, y);
24 }
25
26 void Land() {
27     if (isDayMode) {
28         glColor3f(251.0/256, 185.0/256, 149.0/256); //Light Brown
29     }
30     else {
31         glColor3f(0.4101, 0.164, 0.164); //Dark Brown
32     }
33
34     glBegin(GL_POLYGON);
35     SetPixel(500, 0);
36     SetPixel(1024, 0);
37     SetPixel(1024, 300);
38     SetPixel(500, 300);
39     SetPixel(200, 0);
40     glEnd();
41 }
42
43 void Lighthouse() {
44     if (isDayMode) {
45         glColor3f(1.0, 1.0, 1.0); //White Base
46     }
47     else {
48         glColor3f(0.7f, 0.7f, 0.7f);
49     }
50
51     glBegin(GL_POLYGON);
52     SetPixel(740, 300);
53     SetPixel(750, 350);
54     SetPixel(850, 350);
55     SetPixel(860, 300);
56     glEnd();
57
58     if (isDayMode) { ... }
59     else { ... }
60
61     glBegin(GL_POLYGON);
62     SetPixel(750, 350);
63     SetPixel(750, 500);
64     SetPixel(850, 500);
65     SetPixel(850, 350);
66     glEnd();
67
68     if (isDayMode) {
69         glColor3f(1.0, 1.0, 1.0); //White mid
70     }
71 }
```

```

CGProject (Global Scope) Lighthouse()
74     else {
75         glColor3f(0.7f, 0.7f, 0.7f);
76     }
77
78     glBegin(GL_POLYGON);
79     SetPixel(750, 400);
80     SetPixel(750, 450);
81     SetPixel(850, 450);
82     SetPixel(850, 400);
83     glEnd();
84
85
86     if (isDayMode) {
87         glColor3f(1.0, 0.0, 0.0); //Red Top
88     }
89     else {
90         glColor3f(0.5f, 0.0f, 0.0f);
91     }
92
93     glBegin(GL_TRIANGLES);
94     SetPixel(765, 540);
95     SetPixel(800, 560);
96     SetPixel(835, 540);
97
98     glEnd();
99
100    if (isDayMode) {
101        glColor3f(1.0, 1.0, 1.0); //White Portion under Triangle Top
102    }
103    else {
104        glColor3f(0.7f, 0.7f, 0.7f);
105    }
106
107    glBegin(GL_POLYGON);
108    SetPixel(775, 500);
109    SetPixel(775, 540);
110    SetPixel(825, 540);
111    SetPixel(825, 500);
112    glEnd();
113
114    if (isDayMode) {
115        glColor3f(0.5, 0.5, 0.5); //Grey Railing
116    }
117    else {
118        glColor3f(0.3, 0.3, 0.3);
119    }
120    glBegin(GL_POLYGON);
121    SetPixel(750, 500);
122    SetPixel(750, 510);
123    SetPixel(755, 510);
124    SetPixel(755, 500);
125    glEnd();
126    glBegin(GL_POLYGON);
127    SetPixel(845, 500);
128    SetPixel(845, 510);
129    SetPixel(850, 510);
130    SetPixel(850, 500);
131    glEnd();
132    glBegin(GL_POLYGON);
133    SetPixel(795, 500);
134    SetPixel(795, 510);
135    SetPixel(805, 510);
136    SetPixel(805, 500);
137    glEnd();
138    glBegin(GL_POLYGON);
139    SetPixel(750, 510);
140    SetPixel(750, 515);
141    SetPixel(850, 515);
142    SetPixel(850, 510);
143    glEnd();

```

```

143     float r = 7;
144     for (int i = r; i > 0; i--)
145     {
146         float x = 0, y = i, xc = 800, yc = 525;
147         //float p = 1 - r;
148         float p = 3 - 2 * i;
149         glColor3f(0.0, 0.0, 0.0);
150         glPointSize(2.0);
151         glBegin(GL_POINTS);
152         while (x <= y)
153         {
154             x++;
155             if (p < 0)
156             {
157                 //p = p + 2 * (x + 1) + 1;
158                 p = p + 4 * (x + 1) + 2;
159             }
160             else
161             {
162                 y--;
163                 //p = p + 2 * (x + 1) + 1 - 2 * (y - 1);
164                 p = p + 4 * (x + 1) + 2 - 4 * (y - 1);
165             }
166             glVertex2i(xc + x, yc + y);
167             glVertex2i(xc - x, yc + y);
168             glVertex2i(xc + x, yc - y);
169             glVertex2i(xc - x, yc - y);
170             glVertex2i(xc + y, yc + x);
171             glVertex2i(xc - y, yc + x);
172             glVertex2i(xc + y, yc - x);
173             glVertex2i(xc - y, yc - x);
174         }
175         glEnd();
176     }
177 }
178
179 void SunorMoon() {
180     int xc = 100, yc = 700;
181     float r = 50;
182     for (int i = r; i > 0; i--)
183     {
184         float x = 0, y = i;
185         //float p = 1 - r;
186         float p = 3 - 2 * i;
187         if (isDayMode) {
188             glColor3f(1.0, 1.0, 0.0);
189         }
190         else {
191             glColor3f(0.9, 0.9, 0.9);
192         }
193     }
194
195     glPointSize(2.0);
196     glBegin(GL_POINTS);
197     while (x <= y)
198     {
199         x++;
200         if (p < 0)
201         {
202             //p = p + 2 * (x + 1) + 1;
203             p = p + 4 * (x + 1) + 2;
204         }
205         else
206         {
207             y--;
208             //p = p + 2 * (x + 1) + 1 - 2 * (y - 1);
209             p = p + 4 * (x + 1) + 2 - 4 * (y - 1);
210         }
211         glVertex2i(xc + x, yc + y);

```



```

212         glVertex2i(xc - x, yc + y);
213         glVertex2i(xc + x, yc - y);
214         glVertex2i(xc - x, yc - y);
215         glVertex2i(xc + y, yc + x);
216         glVertex2i(xc - y, yc + x);
217         glVertex2i(xc + y, yc - x);
218         glVertex2i(xc - y, yc - x);
219     }
220     glEnd();
221 }
222 }
223 }
224
225 void light() {
226     glColor3f(1.0, 1.0, 0.0); //Yellow
227     glBegin(GL_POLYGON);
228     SetPixel(xpos, 200);
229     SetPixel(xpos + d, 200);
230     SetPixel(800, 525);
231     glEnd();
232 }
233
234 void star(int x, int y, float col) {
235     glColor3f(col, col, col);
236     glBegin(GL_POINTS);
237     SetPixel(x, y);
238     SetPixel(x, y + 1);
239     SetPixel(x, y + 2);
240     SetPixel(x - 1, y + 1);
241     SetPixel(x + 1, y + 1);
242     SetPixel(x, y + 3);
243     SetPixel(x, y - 1);
244     SetPixel(x - 2, y + 1);
245     SetPixel(x + 2, y + 1);
246     glEnd();
247 }
248
249 void Water()
250 {
251     if (isDayMode) {
252         glColor3f(0.0f, 0.0f, 1.0f);
253     }
254     else {
255         glColor3f(0.0f, 0.0f, 0.3f);
256     }
257
258     glBegin(GL_POLYGON);
259     float x = 500 + sin(waveFrequency * glutGet(GLUT_ELAPSED_TIME) * waveSpeed) * waveAmplitude;
260     float y = waterHeight + sin(waveFrequency * glutGet(GLUT_ELAPSED_TIME) * waveSpeed + M_PI / 2) * waveAmplitude;
261     float x2 = sin(waveFrequency * glutGet(GLUT_ELAPSED_TIME) * waveSpeed + M_PI / 2) * waveAmplitude;
262     float y2 = waterHeight;
263     SetPixel(-30, 0);
264     SetPixel(500, 0);
265     SetPixel(x, y);
266     SetPixel(-30 + x2, y2);
267     glEnd();
268 }
269
270 void sky() {
271     if (isDayMode) {
272         glColor3f(0.529f, 0.808f, 0.922f);
273     }
274     else {
275         glColor3f(0.0f, 0.0f, 0.2f);
276     }
277     glClear(GL_COLOR_BUFFER_BIT);
278     glBegin(GL_POLYGON);
279     SetPixel(0, 0);
280     SetPixel(1024, 0);
281     SetPixel(1024, 768);

```

```

281     SetPixel(0, 768);
282     glEnd();
283 }
284
285 void boat() {
286     glColor3f(0.6f, 0.0f, 0.2f);
287     glBegin(GL_POLYGON);
288     SetPixel(0 + boatx, 330 + boatx*0.002*sin(waveFrequency * glutGet(GLUT_ELAPSED_TIME) * waveSpeed) * waveAmpl
289     SetPixel(35 + boatx, 280 + boatx * 0.002 * sin(waveFrequency * glutGet(GLUT_ELAPSED_TIME) * waveSpeed) * wav
290     SetPixel(115 + boatx, 280 + boatx * 0.002 * sin(waveFrequency * glutGet(GLUT_ELAPSED_TIME) * waveSpeed) * wa
291     SetPixel(150 + boatx, 330 + boatx * 0.002 * sin(waveFrequency * glutGet(GLUT_ELAPSED_TIME) * waveSpeed) * wa
292     glEnd();
293     glColor3f(0.8f, 0.8f, 0.8f);
294     glBegin(GL_POLYGON);
295     SetPixel(35 + boatx, 330 + boatx * 0.002 * sin(waveFrequency * glutGet(GLUT_ELAPSED_TIME) * waveSpeed) * wav
296     SetPixel(85 + boatx, 395 + boatx * 0.002 * sin(waveFrequency * glutGet(GLUT_ELAPSED_TIME) * waveSpeed) * wav
297     SetPixel(85 + boatx, 330 + boatx * 0.002 * sin(waveFrequency * glutGet(GLUT_ELAPSED_TIME) * waveSpeed) * wav
298     glEnd();
299 }
300
301 void boatleft() {
302     if(boatx>0)
303         boatx -= 0.001 * 1024;
304 }
305
306 void boatright() {
307     if (boatx < 350)
308         boatx += 0.001 * 1024;
309 }
310
311 void display()
312 {
313     glClear(GL_COLOR_BUFFER_BIT);
314     glMatrixMode(GL_MODELVIEW);
315     sky();
316     Lighthouse();
317     if (isDayMode == false) {
318         light();
319         star(010, 750, col[0]);
320         star(190, 675, col[1]);
321         star(185, 590, col[2]);
322         star(305, 700, col[3]);
323         star(356, 685, col[4]);
324         star(502, 650, col[5]);
325         star(550, 725, col[6]);
326         star(750, 700, col[7]);
327         star(852, 750, col[8]);
328         star(900, 640, col[9]);
329     }
330     boat();
331     Water();
332     Land();
333     SunorMoon();
334     glFlush();
335 }
336
337 void myInit()
338 {
339     glClearColor(1.0, 1.0, 1.0, 0.0);
340
341     glMatrixMode(GL_PROJECTION);
342     gluOrtho2D(0.0, 1024.0, 0.0, 768.0);
343 }
344
345 void update(int value)
346 {
347     for (int i = 0; i < n; i++) {
348         col[i] = rand() / 32767.00;
349     }

```

```

349     }
350     if (moving_right) {
351         xpos += 10;
352         d += -3;
353         if (xpos >= 500) {
354             delay += 1;
355             xpos -= 10;
356             d += 3;
357
358             if (delay == 10)
359                 moving_right = false;
360         }
361     }
362     else {
363         xpos -= 10;
364         d += 3;
365         if (xpos <= -300) {
366             delay -= 1;
367             xpos += 10;
368             d += -3;
369             if (delay == 0)
370                 moving_right = true;
371         }
372     }
373     glutPostRedisplay(); // Request a redraw of the window
374     glutTimerFunc(100, update, 0); // Call update function after 100 milliseconds
375 }
376 void keyboard(unsigned char key, int x, int y) {
377     switch (key) {
378         case 'm': // Switch to day mode
379             isDayMode = true;
380             break;
381         case 'n': // Switch to night mode
382             isDayMode = false;
383             break;
384         case 'a': // Switch to night mode
385             boatleft();
386             break;
387         case 'd': // Switch to night mode
388             boatright();
389             break;
390     }
391 }
392
393 int main(int argc, char** argv)
394 {
395     glutInit(&argc, argv);
396     glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
397     glutInitWindowSize(1024, 768);
398     glutInitWindowPosition(0, 0);
399     glutCreateWindow("CG Project");
400     myInit();
401     glutDisplayFunc(display);
402     glutTimerFunc(0, update, 0); // Call update function immediately
403     glutKeyboardFunc(keyboard);
404     glutMainLoop();
405
406
407
408 }

```

Output/Screenshots



