**计算机图形学**

**实习报告**

班级： 191174

姓名： 牟鑫一

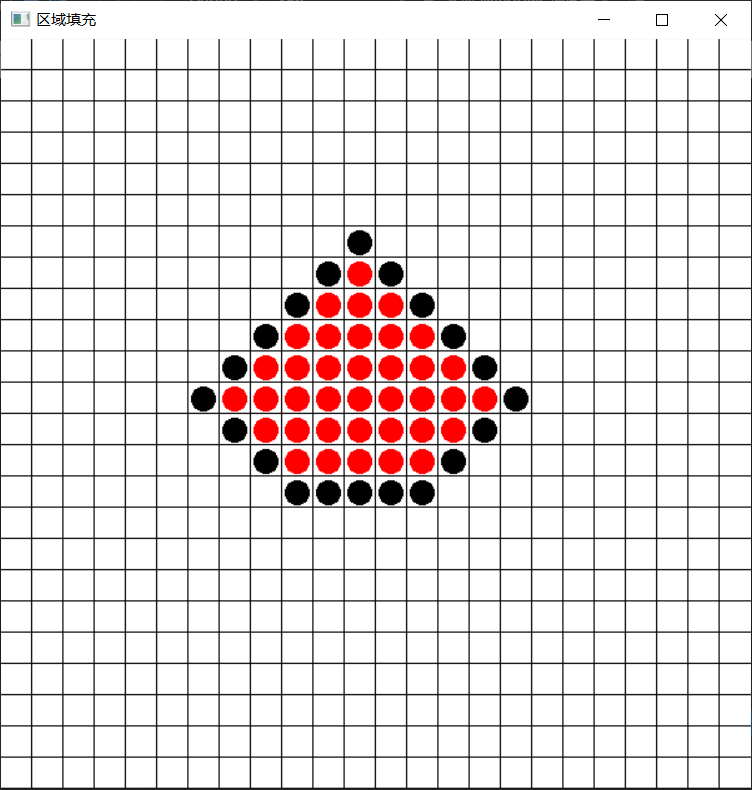
学号： 20161001764

日期： 2019/06/21

1. **区域填充**
2. **题目**

区域填充算法，由区域内一点开始，由内向外将填充色扩展到整个区域。

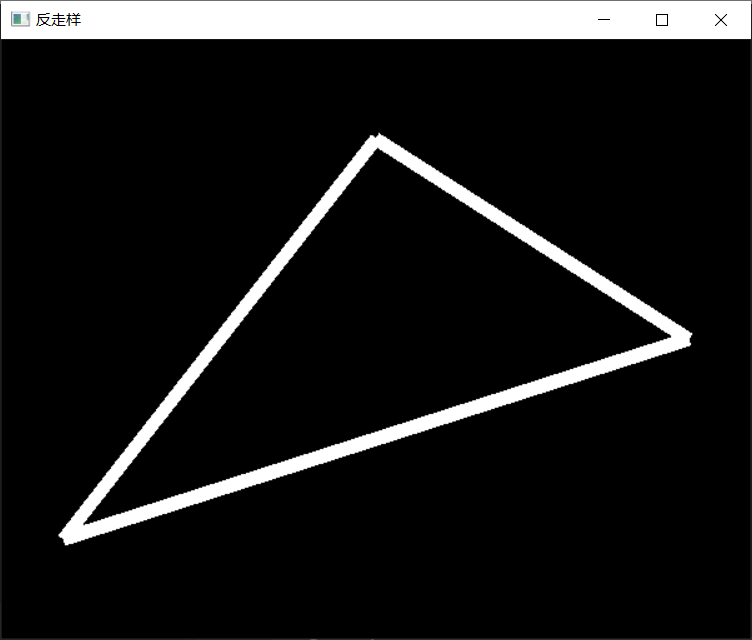
1. **源代码**
2. #include <iostream>
3. #include <cmath>
4. #include <stack>
5. #include "gl/glut.h"
6. **using** **namespace** std;
7. #define PI 3.14
9. **void** Display();
10. **void** DrawGrid();
11. **void** Drawtri(**int** x, **int** y, **int** color);
12. **void** FourConnected(**int** x, **int** y);
13. **void** ScanFill(**int** x, **int** y);
14. **void** mouse(GLint button, GLint action, GLint x, GLint y);
16. **struct** Position
17. {
18. **int** x;
19. **int** y;
20. Position()
21. {
22. x = 0;
23. y = 0;
24. };
25. Position(**int** px, **int** py)
26. {
27. x = px;
28. y = py;
29. };
30. };
32. stack<Position> sta;
33. **int** a[24][24] = { 0 };
35. **void** Display()
36. {
37. glClear(GL\_COLOR\_BUFFER\_BIT);
38. DrawGrid();
39. glFlush();
40. }
42. **void** DrawGrid()
43. {
44. glColor3f(0.0, 0.0, 0.0);       //格子颜色，黑色
45. glBegin(GL\_LINES);
46. **for** (**int** i = 0; i < 600; i += 25)
47. {
48. glVertex2f(i, 0);
49. glVertex2f(i, 600);
50. }
51. **for** (**int** j = 0; j < 600; j += 25)
52. {
53. glVertex2f(0, j);
54. glVertex2f(600, j);
55. }
56. glEnd();
57. **for** (**int** k = 0; k < 24; k++)
58. {
59. **for** (**int** l = 0; l < 24; l++)
60. {
61. **if** (a[k][l] == 1)
62. {
63. Drawtri(k \* 25 + 12, l \* 25 + 12, 1);
64. }
65. **else** **if** (a[k][l] == 2)
66. {
67. Drawtri(k \* 25 + 12, l \* 25 + 12, 2);
68. }
69. }
70. }
71. }
73. **void** Drawtri(**int** x, **int** y, **int** color)
74. {
75. **double** n = 200;
76. **float** R = 10;
77. **int** i;
78. **if** (color == 1)
79. {
80. glColor3f(0.0, 0.0, 0.0);   //点颜色
81. }
82. **else** **if** (color == 2)
83. {
84. glColor3f(1.0, 0.0, 0.0);   //填充颜色
85. }
86. glBegin(GL\_POLYGON);
87. glVertex2f(x, y);
88. **for** (i = 0; i <= n; i++)
89. glVertex2f(R \* cos(2 \* PI / n \* i) + x, R \* sin(2 \* PI / n \* i) + y);
90. glEnd();
91. glPopMatrix();
92. }
94. **void** FourConnected(**int** x, **int** y)
95. {
96. **if** (x < 0 || y < 0 || x>23 || y>23)
97. **return**;
98. **if** (a[x][y] == 0)
99. {
100. a[x][y] = 2;
101. FourConnected(x - 1, y);
102. FourConnected(x + 1, y);
103. FourConnected(x, y - 1);
104. FourConnected(x, y + 1);
105. }
106. }
108. **void** ScanFill(**int** x, **int** y)
109. {
110. **if** (a[x][y] != 0)
111. **return**;
112. Position first(x, y);
113. sta.push(first);
114. **while** (!sta.empty())
115. {
116. **int** rightX = 0;
117. **int** leftX = 0;
118. Position cur = sta.top();
119. sta.pop();
120. a[cur.x][cur.y] = 2;
121. **for** (**int** i = 1; i < 24; i++)
122. {
123. **if** (cur.x + i < 24)
124. {
125. **if** (a[cur.x + i][cur.y] == 0)
126. a[cur.x + i][cur.y] = 2;
127. **else**
128. {
129. rightX = cur.x + i - 1;
130. **break**;
131. }
132. }
133. **if** (i == 23)
134. {
135. rightX = 23;
136. }
137. }
138. **for** (**int** i = 1; i < 24; i++)
139. {
140. **if** (cur.x - i > -1)
141. {
142. **if** (a[cur.x - i][cur.y] == 0)
143. a[cur.x - i][cur.y] = 2;
144. **else**
145. {
146. leftX = cur.x - i + 1; **break**;
147. }
148. }
149. **if** (i == 0)
150. {
151. leftX = 0;
152. }
153. }
154. cout << leftX << "," << rightX << endl;
155. **int** upRightX = -1;
156. **for** (**int** i = leftX; i <= rightX; i++)
157. {
158. **if** (a[i][cur.y + 1] == 0 && cur.y + 1 < 24)
159. {
160. upRightX = i;
161. }
162. }
163. **if** (upRightX != -1)
164. {
165. Position temPos(upRightX, cur.y + 1);
166. sta.push(temPos);
167. }
169. **int** downRightX = -1;
170. **for** (**int** i = leftX; i <= rightX; i++)
171. {
172. **if** (a[i][cur.y - 1] == 0 && cur.y - 1 >= 0)
173. {
174. downRightX = i;
175. }
176. }
177. **if** (downRightX != -1)
178. {
179. Position temPos(downRightX, cur.y - 1);
180. sta.push(temPos);
181. }
182. }
183. }
185. **void** mouse(GLint button, GLint action, GLint x, GLint y)
186. {
187. **int** curX, curY;
188. **if** (button == GLUT\_LEFT\_BUTTON && action == GLUT\_DOWN)
189. {
190. curX = x / 25;
191. curY = (600 - y) / 25;
192. a[curX][curY] = 1;
193. glutPostRedisplay();
194. }
195. **if** (button == GLUT\_RIGHT\_BUTTON && action == GLUT\_DOWN)
196. {
197. curX = x / 25;
198. curY = (600 - y) / 25;
199. ScanFill(curX, curY);
200. glutPostRedisplay();
201. }
202. }
204. **int** main(**int** argc, **char**\*\* argv)
205. {
206. glutInit(&argc, argv);
207. glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);
208. glutInitWindowPosition(100, 100);
209. glutInitWindowSize(600, 600);
210. glutCreateWindow("区域填充");
211. glClearColor(1.0, 1.0, 1.0, 1.0);
212. glMatrixMode(GL\_PROJECTION);
213. gluOrtho2D(0.0, 600.0, 0.0, 600.0);
214. glPointSize(12.0f);
215. glutDisplayFunc(Display);
216. glutMouseFunc(mouse);
217. glutMainLoop();
218. **return** 0;
219. }
220. **测试**



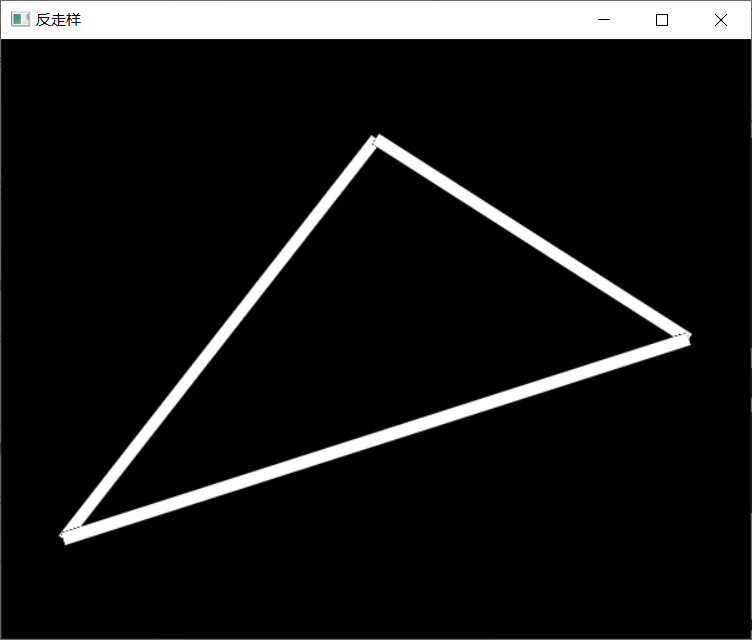
1. **反走样**
2. **题目**

利用反走样技术减少或消除锯齿现象。

1. **源代码**
2. #include<iostream>
3. #include "gl/glut.h"
4. **using** **namespace** std;
5. #define YES 1
6. #define NO 0
7. **int** Drawing;
9. //菜单
10. **void** Menu(**int** value)
11. {
12. Drawing = value;
13. glutPostRedisplay();
14. }
16. //初始化
17. **void** Initialization()
18. {
19. glClearColor(0.0f, 0.0f, 0.0f, 1.0f);   //背景色，白色
20. glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_DST\_ALPHA);
21. glEnable(GL\_POINT\_SMOOTH);
22. glHint(GL\_POINT\_SMOOTH\_HINT, GL\_NICEST);
23. glEnable(GL\_LINE\_SMOOTH);
24. glHint(GL\_LINE\_SMOOTH\_HINT, GL\_NICEST);
25. Drawing = NO;
26. }
28. //未处理
29. **void** Original(**void**)
30. {
31. glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);
32. glMatrixMode(GL\_MODELVIEW);
33. glLoadIdentity();
34. **if** (Drawing == YES)
35. glEnable(GL\_BLEND);
36. **else**
37. glDisable(GL\_BLEND);
38. glColor4f(1.0f, 1.0f, 1.0f, 1.0f);  //画笔颜色，黑色
39. glLineWidth(10);
40. glBegin(GL\_LINE\_LOOP);
41. glVertex2f(-2.0f, -2.0f);
42. glVertex2f(0.0f, 2.0f);
43. glVertex2f(2.0f, 0.0f);
44. glEnd();
45. glutSwapBuffers();
46. }
48. //反走样处理
49. **void** AntiAliasing(**int** w, **int** h)
50. {
51. glViewport(0, 0, w, h);
52. glMatrixMode(GL\_PROJECTION);
53. glLoadIdentity();
54. **if** (h != 0)
55. {
56. GLfloat aspect = (**float**)w / (**float**)h;
57. **if** (w < h)
58. gluOrtho2D(-3, 3, -3 \* aspect, 3 \* aspect);
59. **else**
60. gluOrtho2D(-3 / aspect, 3 / aspect, -3, 3);
61. }
62. glMatrixMode(GL\_MODELVIEW);
63. glLoadIdentity();
64. }
66. **int** main(**int** argc, **char**\* argv[])
67. {
68. glutInit(&argc, argv);
69. glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);
70. glutInitWindowSize(600, 480);
71. glutCreateWindow("反走样");
72. glutReshapeFunc(AntiAliasing);
73. glutDisplayFunc(Original);
74. glutCreateMenu(Menu);
75. Initialization();
76. glutAddMenuEntry("Original", NO);
77. glutAddMenuEntry("Antialiasing", YES);
78. glutAttachMenu(GLUT\_RIGHT\_BUTTON);
79. glutMainLoop();
80. **return** 0;
81. }
82. **测试**



**处理前**



**处理后**