**计算机图形学实验**

**姓 名：白文强**

**学 号：20191060064**

**专 业：计算机科学与技术**

**教 师：钱文华**

实验七 直线段裁剪实验

时间：2020年12月04日

地点：信息学院2202机房

1. 实验内容

指定矩形框，输入直线段端点并显示直线，采用Cohen-Sutherland算法对直线段进行裁剪，显示裁剪之后结果

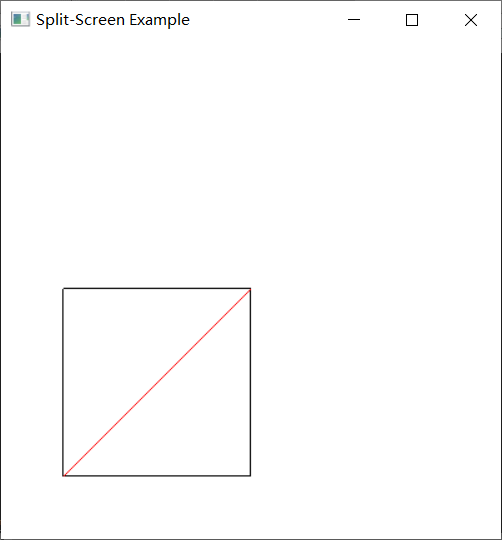
2、实验目的

验证Cohen-Sutherland直线段裁剪算法。

3、实验代码

1. #include <windows.h>
2. #include <GL/glut.h>
3. #include <stdio.h>
4. #include <stdlib.h>
6. **class** wcPt2D
7. {
8. **public** :
9. GLfloat x, y;
10. wcPt2D(GLfloat x, GLfloat y);
11. wcPt2D();
13. };
15. wcPt2D \*p1 = **new** wcPt2D(0,0);
16. wcPt2D \*p2 = **new** wcPt2D(400, 400);
18. wcPt2D \*winMin = **new** wcPt2D(50, 50);
19. wcPt2D \*winMax = **new** wcPt2D(200,200);
21. wcPt2D::wcPt2D( GLfloat tx, GLfloat ty )
22. {
23. x = tx;
24. y = ty;
25. }
26. wcPt2D::wcPt2D( )
27. {
29. }
30. **void** init(**void**)
31. {
32. glClearColor(1.0,1.0,1.0,0.0);
33. glMatrixMode(GL\_PROJECTION);
34. gluOrtho2D(0, 400, 0, 400);
35. }
36. **inline** GLint round(**const** GLfloat a)
37. {
38. **return** GLint(a+0.5);
39. }
40. **const** GLint winLeftBitCode = 0x1;
41. **const** GLint winRightBitCode = 0x2;
42. **const** GLint winBottomBitCode = 0x4;
43. **const** GLint winTopBitCode = 0x8;
45. **inline** GLint inside(GLint code)
46. {
47. **return** GLint (!code);
48. }
49. **inline** GLint reject(GLint code1, GLint code2)
50. {
51. **return** GLint(code1 & code2);
52. }
53. **inline** GLint accept(GLint code1, GLint code2)
54. {
55. **return** GLint (!(code1 | code2));
56. }
58. GLubyte encode( wcPt2D pt, wcPt2D winMin, wcPt2D winMax )
59. {
60. GLubyte code = 0x00;
61. **if**( pt.x < winMin.x )
62. code = code | winLeftBitCode;
63. **if**( pt.x > winMax.x )
64. code  = code | winRightBitCode;
65. **if**( pt.y < winMin.y )
66. code = code | winBottomBitCode;
67. **if**( pt.y > winMax.y )
68. code = code | winTopBitCode;
69. **return** code;
70. }
72. **void** swapPts( wcPt2D \* p1, wcPt2D \*p2 )
73. {
74. wcPt2D tmp;
75. tmp = \*p1;
76. \*p1 = \*p2;
77. \*p2 = tmp;
78. }
80. **void** swapCodes(GLubyte \*c1, GLubyte \*c2)
81. {
82. GLubyte tmp;
83. tmp = \*c1;
84. \*c1 = \*c2;
85. \*c2 = tmp;
86. }
88. **void** lineClipCohSuth( /\*wcPt2D winMin, wcPt2D winMax, wcPt2D p1, wcPt2D p2 \*/)
89. {
90. GLubyte code1, code2;
91. GLint done = **false**, plotLine = **false**;
92. GLfloat m;
93. **while**(!done)
94. {
95. code1 = encode(\*p1, \*winMin, \*winMax);
96. code2 = encode(\*p2, \*winMin, \*winMax);
97. **if**(accept(code1, code2))
98. {
99. done = **true**;
100. plotLine = **true**;
101. }
102. **else**
103. {
104. **if**(reject(code1,code2))
105. done = **true**;
106. **else**
107. {
108. **if**(inside(code1))
109. {
110. swapPts(p1, p2);
111. swapCodes(&code1, &code2);
112. }
113. **if**( p2->x != p1->x )
114. m = (p2->y - p1->y) / (p2->x - p1->x);
115. **if**(code1 & winLeftBitCode)
116. {
117. p1->y += (winMin->x - p1->x)\*m;
118. p1->x  = winMin->x;
119. }
120. **else**
121. {
122. **if**(code1 & winBottomBitCode)
123. {
124. **if**(p2->x != p1->x)
125. p1->x += (winMin->y - p1->y) / m;
126. p1->y = winMin->y;
127. }
128. **else**
129. {
130. **if**(code1 & winTopBitCode )
131. {
132. **if**(p2->x != p1->x)
133. p1->x += (winMax->y - p1->y) / m;
134. p1->y = winMax->y;
135. }
136. }
137. }
138. }
139. }
140. }
141. **if**(plotLine)
142. {
143. glClear(GL\_COLOR\_BUFFER\_BIT);   // 清空显示窗口
144. glColor3f(0.0, 0.0, 0.0);
145. glBegin(GL\_LINES);
146. glVertex2i(winMin->x, winMin->y);
147. glVertex2i(winMax->x, winMin->y);
149. glVertex2i(winMin->x, winMin->y);
150. glVertex2i(winMin->x, winMax->y);
152. glVertex2i(winMax->x, winMin->y);
153. glVertex2i(winMax->x, winMax->y);
155. glVertex2i(winMax->x, winMax->y);
156. glVertex2i(winMin->x, winMax->y);
157. glEnd();
158. glColor3f(1.0, 0.0, 0.0);       // 指定前景色
159. glBegin(GL\_LINES);
160. glVertex2i(round(p1->x), round(p1->y));
161. glVertex2i(round(p2->x), round(p2->y));
162. glEnd();
163. glFlush();
164. }
165. }
167. **int** main(**int** argc, **char** \*\* argv)
168. {
170. glutInit(&argc, argv);
171. glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);
172. glutInitWindowPosition(200,200);
173. glutInitWindowSize(400, 400);
174. glutCreateWindow("Split-Screen Example\n");
175. init();
177. glutDisplayFunc(lineClipCohSuth);
179. glutMainLoop();
180. **return** 0;
181. }

4、实验结果



5、实验总结

通过将点进行编码判断是否在裁剪区域内；

通过两个点编码之间的“与”操作和“或”操作判断这两点之间的线段是否在裁剪区域内