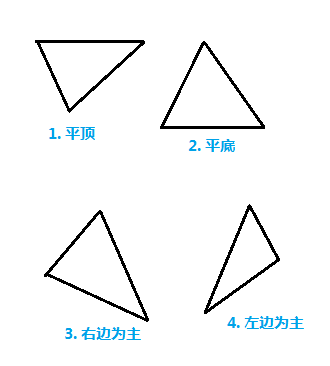
http://blog.csdn.net/cppyin/article/details/6232453 （三角形）

**1. 为什么要光栅化一个三角形**

我们不能总让我们的引擎显示线框，要支持实心颜色、光照还有纹理贴图，这些都需要光栅化一个三角形作为支持。

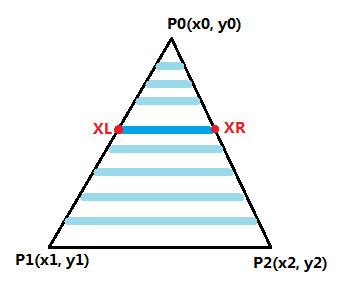
**2. 三角形的类型**

为了方便光栅化，一般将三角形分为以下4种：



**3. 平底三角形光栅化**

先上图：



光栅化平底三角形的原理很简单，就是从上往下画横线。在图里我们取任意的一条光栅化直线，这条直线左边的端点x值为XL，右边的为XR。y值就不用考虑了，因为这些线是从上往下画的，所以y就是从y0一直++，直到y1或者y2。

所以[**算法**](http://lib.csdn.net/base/31)很简单，就是每次增加一个y，就要计算一下XL和XR，然后调用我们很早以前写的那个光栅化直线的函数，来画这条线即可。

怎么求XL和XR呢？

直线有很多种形式可以表示，因为我们现在知道顶点的坐标，所以最直观的表示形式就是两点式：

对于已知直线上的两点(x1,y1)和(x2,y2)有：

(y-y1) / (y2-y1) = (x-x1) / (x2-x1)

因为y已知，我们变换一下公式，表示为x的值为：

x = (y-y1) \* (x2-x1) / (y2-y1)

根据上图的，我们只要把P0,P1代入，就可以求得XL，把P0,P2代入，就可以求得XR。不多说了，直接给出实现代码：

**[cpp]** [view plain](http://blog.csdn.net/cppyin/article/details/6232453) [copy](http://blog.csdn.net/cppyin/article/details/6232453)

1. **void** \_CPPYIN\_3DLib::DrawTriangle1(**int** x1, **int** y1, **int** x2, **int** y2, **int** x3, **int** y3, **DWORD** color) // 画实心平底三角形
2. {
3. **for** (**int** y = y1; y <= y2; ++y)
4. {
5. **int** xs, xe;
6. xs = (y - y1) \* (x2 - x1) / (y2 - y1) + x1 + 0.5;
7. xe = (y - y1) \* (x3 - x1) / (y3 - y1) + x1 + 0.5;
8. DrawLine(xs, y, xe, y, color);
9. }
10. }

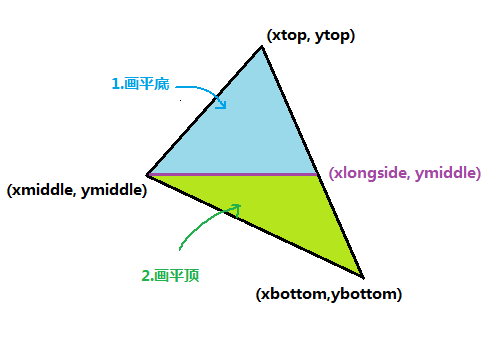
**4. 平顶三角形的光栅化**

不用多说了，原理同上，直接贴代码了。

**[cpp]** [view plain](http://blog.csdn.net/cppyin/article/details/6232453) [copy](http://blog.csdn.net/cppyin/article/details/6232453)

1. **void** \_CPPYIN\_3DLib::DrawTriangle2(**int** x1, **int** y1, **int** x2, **int** y2, **int** x3, **int** y3, **DWORD** color) // 画实心平顶三角形
2. {
3. **for** (**int** y = y1; y <= y3; ++y)
4. {
5. **int** xs, xe;
6. xs = (y - y1) \* (x3 - x1) / (y3 - y1) + x1 + 0.5;
7. xe = (y - y2) \* (x3 - x2) / (y3 - y2) + x2 + 0.5;
8. DrawLine(xs, y, xe, y, color);
9. }
10. }

**5.  任意三角形的光栅化**



其实看了这个图就应该发现特简单，我们求得一个特殊点(xlongside, ymiddle)，之后画一个平底三角形，再画一个平顶三角形就搞定了。

下面的代码实现了这个功能，有几点辅助说明一下：

1. 通过给定三个顶点的y值，可以判断出是否是平顶还是平底，如果满足其一，就直接画。

2. 传入函数的三个顶点是乱序的，所以要根据y值的情况来区分几种情况，在我的实现里，穷举了y值的几种情况，然后给上图中的三个点赋值。

3. 依次画就是了，下面是代码。

**[cpp]** [view plain](http://blog.csdn.net/cppyin/article/details/6232453) [copy](http://blog.csdn.net/cppyin/article/details/6232453)

1. **void** \_CPPYIN\_3DLib::DrawTriangle(**int** x1, **int** y1, **int** x2, **int** y2, **int** x3, **int** y3, **DWORD** color) // 画任意实心三角形
2. {
3. **if** (y1 == y2)
4. {
5. **if** (y3 <= y1) // 平底
6. {
7. DrawTriangle1(x3, y3, x1, y1, x2, y2, color);
8. }
9. **else** // 平顶
10. {
11. DrawTriangle2(x1, y1, x2, y2, x3, y3, color);
12. }
13. }
14. **else** **if** (y1 == y3)
15. {
16. **if** (y2 <= y1) // 平底
17. {
18. DrawTriangle1(x2, y2, x1, y1, x3, y3, color);
19. }
20. **else** // 平顶
21. {
22. DrawTriangle2(x1, y1, x3, y3, x2, y2, color);
23. }
24. }
25. **else** **if** (y2 == y3)
26. {
27. **if** (y1 <= y2) // 平底
28. {
29. DrawTriangle1(x1, y1, x2, y2, x3, y3, color);
30. }
31. **else** // 平顶
32. {
33. DrawTriangle2(x2, y2, x3, y3, x1, y1, color);
34. }
35. }
36. **else**
37. {
38. **double** xtop, ytop, xmiddle, ymiddle, xbottom, ybottom;
39. **if** (y1 < y2 && y2 < y3) // y1 y2 y3
40. {
41. xtop = x1;
42. ytop = y1;
43. xmiddle = x2;
44. ymiddle = y2;
45. xbottom = x3;
46. ybottom = y3;
47. }
48. **else** **if** (y1 < y3 && y3 < y2) // y1 y3 y2
49. {
50. xtop = x1;
51. ytop = y1;
52. xmiddle = x3;
53. ymiddle = y3;
54. xbottom = x2;
55. ybottom = y2;
56. }
57. **else** **if** (y2 < y1 && y1 < y3) // y2 y1 y3
58. {
59. xtop = x2;
60. ytop = y2;
61. xmiddle = x1;
62. ymiddle = y1;
63. xbottom = x3;
64. ybottom = y3;
65. }
66. **else** **if** (y2 < y3 && y3 < y1) // y2 y3 y1
67. {
68. xtop = x2;
69. ytop = y2;
70. xmiddle = x3;
71. ymiddle = y3;
72. xbottom = x1;
73. ybottom = y1;
74. }
75. **else** **if** (y3 < y1 && y1 < y2) // y3 y1 y2
76. {
77. xtop = x3;
78. ytop = y3;
79. xmiddle = x1;
80. ymiddle = y1;
81. xbottom = x2;
82. ybottom = y2;
83. }
84. **else** **if** (y3 < y2 && y2 < y1) // y3 y2 y1
85. {
86. xtop = x3;
87. ytop = y3;
88. xmiddle = x2;
89. ymiddle = y2;
90. xbottom = x1;
91. ybottom = y1;
92. }
93. **int** xl; // 长边在ymiddle时的x，来决定长边是在左边还是右边
94. xl = (ymiddle - ytop) \* (xbottom - xtop) / (ybottom - ytop) + xtop + 0.5;
96. **if** (xl <= xmiddle) // 左三角形
97. {
98. // 画平底
99. DrawTriangle1(xtop, ytop, xl, ymiddle, xmiddle, ymiddle, color);
101. // 画平顶
102. DrawTriangle2(xl, ymiddle, xmiddle, ymiddle, xbottom, ybottom, color);
103. }
104. **else** // 右三角形
105. {
106. // 画平底
107. DrawTriangle1(xtop, ytop, xmiddle, ymiddle, xl, ymiddle, color);
109. // 画平顶
110. DrawTriangle2(xmiddle, ymiddle, xl, ymiddle, xbottom, ybottom, color);
111. }
112. }
113. }

**6. 实现物体的实心渲染**

之前我们只有一种渲染模式就是线框，有了上面的三角形光栅化函数，我们就可以做实心渲染了，下面是实现渲染的代码，非常简单。

就是遍历物体的每个三角面，然后调用上面的函数而已，嘿嘿。

**[cpp]** [view plain](http://blog.csdn.net/cppyin/article/details/6232453) [copy](http://blog.csdn.net/cppyin/article/details/6232453)

1. **void** \_CPPYIN\_3DLib::ObjectDrawSolid(OBJECT\_PTR obj) // 绘制物体实心多边形
2. {
3. **for** (**int** i = 0; i < obj->PolyCount; ++i)
4. {
5. POINT4D\_PTR p0 = &(obj->VertexListTrans[obj->PolyList[i].VertexIndexs[0]]);
6. POINT4D\_PTR p1 = &(obj->VertexListTrans[obj->PolyList[i].VertexIndexs[1]]);
7. POINT4D\_PTR p2 = &(obj->VertexListTrans[obj->PolyList[i].VertexIndexs[2]]);
9. // 只画正面
10. **if** (obj->PolyList[i].State == POLY\_STATE\_ACTIVE)
11. {
12. DrawTriangle(p0->x, p0->y, p1->x, p1->y, p2->x, p2->y, obj->PolyList[i].Color);
13. }
14. }
15. }

**7. 截图**

这个就是实心渲染的Demo截图了。

