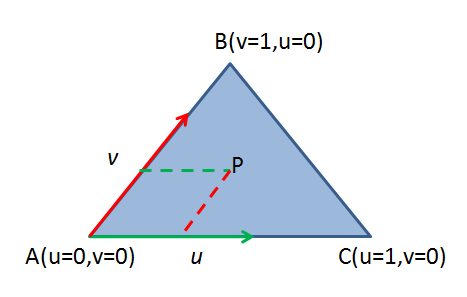
参考：<http://www.cnblogs.com/graphics/archive/2010/08/05/1793393.html>

http://www.cnblogs.com/wangchengfeng/p/3453194.html

三角形的三个点在同一个平面上，如果选中其中一个点，其他两个点不过是相对该点的位移而已，比如选择点A作为起点，那么点B相当于在AB方向移动一段距离得到，而点C相当于在AC方向移动一段距离得到。



所以对于平面内任意一点，都可以由如下方程来表示

P = A +  u \* (C – A) + v \* (B - A) // 方程1

如果系数u或v为负值，那么相当于朝相反的方向移动，即BA或CA方向。那么如果想让P位于三角形ABC内部，u和v必须满足什么条件呢？有如下三个条件

u >= 0

v >= 0

u + v <= 1

在三角形的三个顶点具有3个不同颜色，如何通过插值计算出三角形中每个点的颜色？

　　应用实例：高洛德着色使用3个顶点的颜色进行线性插值，结果如下图：



**解决方案：**

　　显然，无论是线性插值还是双线性插值的都无法解决这个问题。而使用重心坐标则可以很好的解决这个问题。简单的来说，重心坐标就是子三角形与大三角形的面积比，具体的解释参看维基百科，计算过程如下：

　　已知三角形的三个顶点坐标P1, P2, P3, 在三角形内的任意点P, 都存在u和v（由于三角形是一个2D图形，只有两个自由度，所以只要u和v即可）,使得

　　　　P = (1 - u - v) \* P1 + u \* P2 + v \* P3

　　P点在三角形内，所以(u, v)必须满足条件u ≥ 0, v ≥ 0, u + v ≤ 1。u、v体现了每个顶点对特定区域的权重贡献，(1 - u - v)则是第三个权重，只要计算出u和v，就可以计算出每个顶点对P点的贡献。现在已知P1, P2, P3和P的坐标值，求解u和v，只需要解二元一次方程即可：

　　　　P.x = (1 - u - v) \* P1.x + u \* P2.x + v \* P3.x

　　　　P.y = (1 - u - v) \* P1.y + u \* P2.y + v \* P3.y

　　有了u、v值，对P1, P2, P3的颜色值进行加权平均，即可得到P点颜色值

代码：

/// <summary>

/// 该函数用于判断，1：输入的参数点是否在三角形内，2：输入的参数点在三角形内的重心坐标（http://www.cnblogs.com/wangchengfeng/p/3453194.html）

/// </summary>

/// <param name="inputx">输入点x</param>

/// <param name="inputy">输入点y</param>

/// <param name="u">输出的重心坐标1</param>

/// <param name="v">输出的重心坐标2</param>

/// <returns>是否在三角形内</returns>

public bool CalInterpoPara(double inputx,double inputy, out double u, out double v)

{

//重心坐标参考以及根据重心坐标判断点是否在三角形内，并在三角形内插值的文章

//http://www.cnblogs.com/graphics/archive/2010/08/05/1793393.html

//http://www.cnblogs.com/wangchengfeng/p/3453194.html

//解网站里给出的方程组，得到u,v

u = -(Vertex1.X \* Vertex3.Y - Vertex3.X \* Vertex1.Y + Vertex1.Y \* inputx - Vertex3.Y \* inputx - Vertex1.X \* inputy + Vertex3.X \* inputy) / (Vertex1.X \* Vertex2.Y - Vertex2.X \* Vertex1.Y - Vertex1.X \* Vertex3.Y + Vertex3.X \* Vertex1.Y + Vertex2.X \* Vertex3.Y - Vertex3.X \* Vertex2.Y);

v = (Vertex1.X \* Vertex2.Y - Vertex2.X \* Vertex1.Y + Vertex1.Y \* inputx - Vertex2.Y \* inputx - Vertex1.X \* inputy + Vertex2.X \* inputy) / (Vertex1.X \* Vertex2.Y - Vertex2.X \* Vertex1.Y - Vertex1.X \* Vertex3.Y + Vertex3.X \* Vertex1.Y + Vertex2.X \* Vertex3.Y - Vertex3.X \* Vertex2.Y);

if (u < 0 || u > 1 || v < 0 || v > 1) // 超出范围，则表示点不在三角形内

return false;

else if (u + v > 1) return false;

else return true;

}