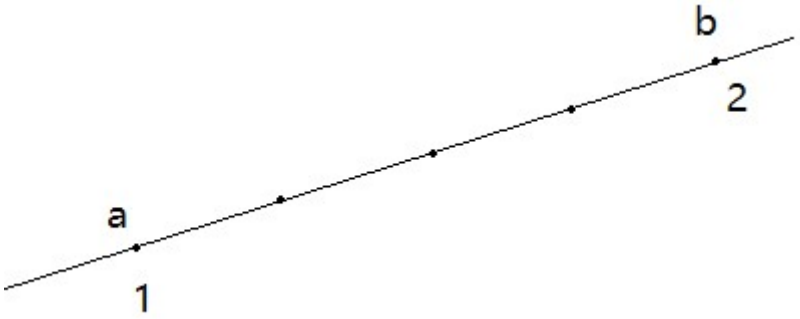
四、插值：

1：线性插值

当一条直线上点 P 的某个属性是**与该点的位置成线性关系**的时候，如果知道直线上任意不重合的两个点的属性，就可以用线性插值的方法计算直线上任意点的属性值。

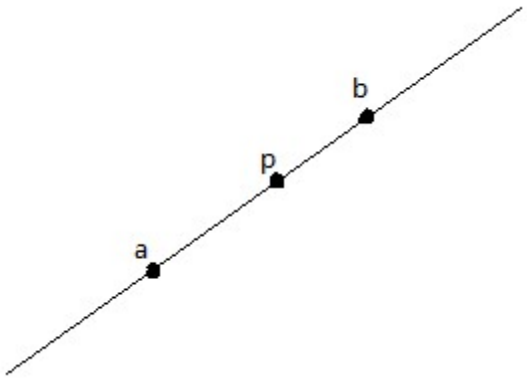
直接这样说可能会有点绕，举个例子:



如上图所示，已知 ab 两点的某个属性值分别为 1 和 2，这个值在整个直线上面是均匀变化的，在 ab 中间有三个点，平均的把 ab 距离分成四等分。则从 a 到 b 方向的三个点该属性值依次为 1.25，1.50，1.75，实际上我们可以得到如下的一个公式计算 V:

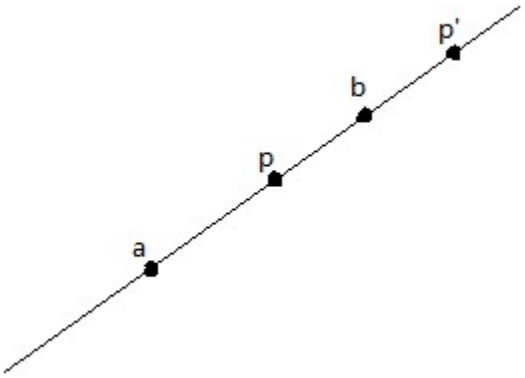
$$\mathrm{V} = \mathrm{t} * \mathrm{V}_a + (1-t)V_b$$

其中 t 表示点 a 对属性 V 的贡献，也就是他的权值W_a，(1-y)为 b 点的权值W_b。并且有:



$$\begin{cases} W_a = \dfrac{|bp|}{|ba|} \\ W_b = \dfrac{|pa|}{|ba|} \end{cases}$$

上当点 p 不在区间[a,b]之内时，上述公式就错误了，如下图:

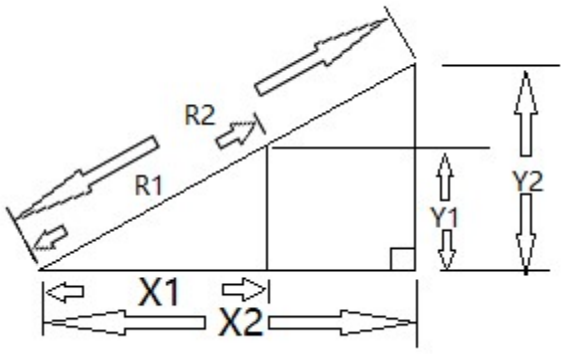


现在有一点 p'落在 b 外面，该点的W_b实际上是个负值($V_{p'} = \frac{|ap|}{|ab|} \times (V_b - V_a) + V_a$)，按照实际值计算的的话$W_b < 0$, 并且当 p 和 p'落在点 b 的两侧时, 当$|bp| = |bp'|$ 有$pW_b = -p'W_b$。而上述公式永远都是大于等于 0 的，所以上述公式中线段的长度应该是有方向的(叫做**有向长度**，和下一节的**有向面积**相对应),对于 b 点来说，当$\overrightarrow{bp}$和$\overrightarrow{ba}$同向时，长度取正值，异向则取负值。改动之后的公式如下:

$$W_a = \frac{x_b - x_p}{x_b - x_a} \text{或者} W_a = \frac{y_b - y_p}{y_b - y_a}$$

$$W_b = \frac{x_p - x_a}{x_b - x_a} \text{或者} W_a = \frac{y_p - y_a}{y_b - y_a}$$

可能有的观众会问：为什么距离之比直接使用了 x 或者 y 的距离之比就计算出来了？距离不是等于$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$吗？请看下图:



由相似三角形的性质可知: $\frac{X_1}{X_2} = \frac{Y_1}{Y_2} = \frac{R_1}{R_2}$，这里的$X_1 = |x_t - x_a|$，$X_2 = |x_b - x_a|$。

对于线段上的线性插值来说，有$W_a + W_b = 1$。

因为$W_a + W_b = 1$, 所以

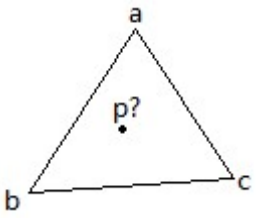
$$\mathrm{V} = \mathrm{t} * \mathrm{V}_a + (1 - t)V_b$$

还可以写成

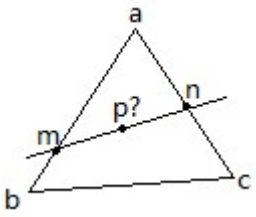
$$\mathrm{V} = \mathrm{V}_a + (\mathrm{V}_b - \mathrm{V}_a)\boldsymbol{W_b}$$

2:重心坐标插值:

假设我们知道平面中三个点的属性值，并且该属性是在平面上均匀变化的，如何根据这三个点计算出任意一点的属性值？如下图：

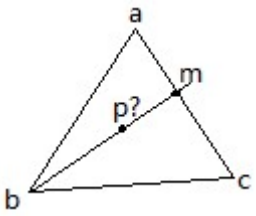


怎么计算出 p 的属性值呢？我们可以用下图的方式计算：

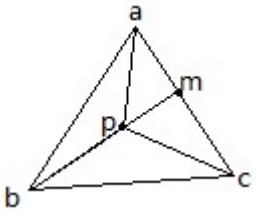


作一条任意的辅助线经过点 p，并且与三角形 abc 相交与两点 mn，先使用直线 ab 经过线性插值计算出 m 点的属性 AttributeM(以后简称 Am)，再用同样的方法计算出 An，然后再在<mn>上面插值计算出 Ap。

为了计算更加简单，我们这次作的辅助线经过三角形的某一顶点：



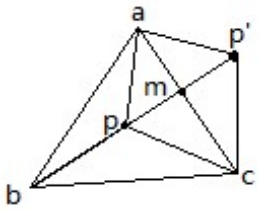
我们先计算出 m 的属性 Am，这时候会发现，点 p 的属性和点 b 相关，并且 b 的权值为$W_b = \frac{x_m - x_p}{x_m - x_b}$。我们先只考虑点 p 在三角形内部的情况，将三个顶点和点 p 都连接起来，如下图：



可以发现$W_b = \frac{|mp|}{|mb|} = \frac{S_{apc}}{S_{abc}}$, 其中$|mp|$表示线段 mp 的长度, S_{apc}表示三角形 apc 的面积。用同样的方法可以分别计算出W_a和W_c，并且可以发现$\boldsymbol{W_m = W_a + W_c}$，则有$\boldsymbol{W_a + W_b + W_c = 1}$。

对于三角形上的线性插值来说使用三角形 ABC 进行线性插值的时候，**每个顶点的权重等于该顶点对边和被插值点组成的三角形和三角形 ABC 的面积之比**

现在来考虑一下顶点在三角形之外的情况：

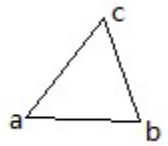


当|pm|=|P'm|时，根据第一节所说的知识，可以知道$pW_b = -p'W_b$，而且这时候可以发现三角形 apm 和三角形 acp'的面积一样(过点 p 和 p'分别作垂直于 ac 的线段并且各自相交于 ac，可以证明这两条线段长度一样，并且 ac 作为底边，则两个三角形的面积一样了)。如果采用上面的面积公式，类似的问题又出现了，所以我们可以定义一个**有向面积**来计算权值。现在对该问题中的面积做出如下定义：

对于一个三角形，如果其顶点顺序为顺时针则称其有向面积为负，为逆时针时为正，且有向面积的绝对值为三角形的面积。

3：有向面积：

对于任意两条三维向量，我们可以将两条向量叉乘之后会得到一条新的向量，这条向量和原来两条组成的平面垂直，方向可以用右手法则判断出来。同时这个向量的模长还等于以这两条向量为两条边的三角形的面积的两倍(数学推导请自行翻阅数学课本)。如下图所示：



$$\frac{\left|\overrightarrow{ac} \times \overrightarrow{ab}\right|}{2} = S_{abc}$$

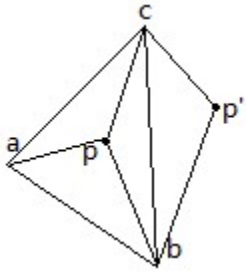
向量 a 叉乘向量 b 的公式为

$$a \times b = \begin{vmatrix} i & j & k \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix} = \left(a_y b_z - a_z b_y\right)i + (a_z b_x - a_x b_z)j + \left(a_x b_y - a_y b_x\right)k$$

我们在构造 ab 向量的时候可以认为这些向量是在 XOY 平面上的，这样他们的 z 值就都为 0，则$a \times b = \left(a_x b_y - a_y b_x\right)k$，面积为：

$$\mathrm{S} = \frac{\sqrt{\left(a_y b_z - a_z b_y\right)^2 + (a_z b_x - a_x b_z)^2 + \left(a_x b_y - a_y b_x\right)^2}}{2}$$

因为$a_z = 0, b_z = 0$，所以$\mathrm{s} = \frac{|a_x b_y - a_y b_x|}{2}$，如果我们把绝对值符号去掉，就可以使用叉乘作为有向面积的计算工具，并且叉乘结果的符号表示了有向面积的方向(如果面积不为 0，则叉乘得到的向量必然指向纸张里面或者外面，指向纸张外面时，面积为正，指向纸张里面时面积为负)：



对于上图如果我们构造出这些向量$\overrightarrow{ba}$，$\overrightarrow{bp}$，$\overrightarrow{bc}$，则当点 p 和点 a 在 bc 边同一侧的时候，$\overrightarrow{bc}\times\overrightarrow{ba}$和$\overrightarrow{bc}\times\overrightarrow{bp}$同时大于 0 或者小于 0。把 p 点换成 p'时，两者异号。所以 a 点的权值可以这样计算：

$$W_a = \frac{\frac{\overrightarrow{bc}\times\overrightarrow{bp}}{2}}{\frac{\overrightarrow{bc}\times\overrightarrow{ba}}{2}} = \frac{\overrightarrow{bc}\times\overrightarrow{bp}}{\overrightarrow{bc}\times\overrightarrow{ba}}$$

同样的 b、c 点权值计算如下：

$$W_b = \frac{\overrightarrow{ca}\times\overrightarrow{cp}}{\overrightarrow{ca}\times\overrightarrow{cb}}$$

$$W_c = \frac{\overrightarrow{ab}\times\overrightarrow{ap}}{\overrightarrow{ab}\times\overrightarrow{ac}}$$

因为$\overrightarrow{ab}\times\overrightarrow{ac}=\overrightarrow{ca}\times\overrightarrow{cb}=\overrightarrow{bc}\times\overrightarrow{ba}$，这里我只证明$\overrightarrow{ab}\times\overrightarrow{ac}=\overrightarrow{ca}\times\overrightarrow{cb}$

为了手写简单，令$h=\overrightarrow{ab}, i=\overrightarrow{ac}, j=\overrightarrow{cb}, -\mathrm{i}=\overrightarrow{ca}$,并且在三角形中可以知道$j=h-i$，

把$\overrightarrow{ab}\times\overrightarrow{ac}=\overrightarrow{ca}\times\overrightarrow{cb}$写成$h\times i=-i\times j$，并且把$j=h-i$带入$-i\times j$,则原等式为

$$h\times i=-i\times(h-i)$$

上式右边：

$$\begin{aligned}
&=-i\times(h-i)\\
&=-i\times\mathrm{h}+(-\mathrm{i}\times-\mathrm{i}) \quad \text{加法分配律}\\
&=-i\times h\\
&=h\times i \qquad\qquad \text{反交换律}
\end{aligned}$$

最后左边等于右边，所以原等式成立。所以我们可以把分母全部变成同一个值$\overrightarrow{ab}\times\overrightarrow{ac}$。

Chapter4/BarycentricInterpolation 代码可以运行得到如下的结果：

三个顶点的颜色 RGB 值分别为(255,0,0),(0,255,0),(0,0,255)，对于每一个顶点使用重心坐标插值计算出每个像素的 RGB 各个分量分别是多少，然后输出到屏幕。

本代码把第二章的 FillPolygon 改成了 FillTriangle 函数，只是简单的把 count 直接写成了 3。

在 FillTriangle 函数的开始处先计算了三角形的面积，如果为 0 则放弃绘制本三角形，并且会提前准备好三角形三边的向量：

```
Vector3 ab(ps[1].X - ps[0].X, ps[1].Y - ps[0].Y, 0.0);//ps[0]->ps[1]
```

```
Vector3 bc(ps[2].X - ps[1].X, ps[2].Y - ps[1].Y, 0.0);//ps[1]->ps[2]
Vector3 ca(ps[0].X - ps[2].X, ps[0].Y - ps[2].Y, 0.0);//ps[2]->ps[0]
Vector3 ac(ps[2].X - ps[0].X, ps[2].Y - ps[0].Y, 0.0);//ps[2]->ps[0]
double square = ab.X * ac.Y - ab.Y * ac.X;//得到三角形有向面积的2倍
if (square == 0)//三角形面积为0则不进行后面的绘制处理
{
    return;
}
```

然后在绘制每个像素的时候计算三个顶点对本像素的权值:

```
double WeightA, WeightB, WeightC;
Vector3 bp(x - ps[1].X, y - ps[1].Y, 0.0);
Vector3 ap(x - ps[0].X, y - ps[0].Y, 0.0);
Vector3 cp(x - ps[2].X, y - ps[2].Y, 0.0);
WeightA = (bc.X * bp.Y - bc.Y * bp.X) / square;
WeightB = (ca.X * cp.Y - ca.Y * cp.X) / square;
WeightC = (ab.X * ap.Y - ab.Y * ap.X) / square;//得到三个顶点的权值
```

得到权值之后再使用该权值对每个属性进行插值，本示例代码插的是rgb的值，纹理坐标的值在三角形中也是均匀变化的，也可以使用线性插值的方法进行插值,线性插值在本系列文章里面最重要的应用就是纹理坐标的计算。

4:纹理的计算

纹理的计算和上一小节中的渐变色插值采用一样的算法，现在先介绍一下纹理的基本知识，纹理通常会以一张图片的形式出现，可以用来给予模型材质。本节介绍一下2D纹理的处理，现在假设有一张纹理图片，通常我们会以图片的左下角作为纹理的原点，横向的叫做u轴，纵向的叫做v轴。uv轴分别是和图片的xy轴平行的，那为什么要弄一个uv坐标呢？假设图片的宽高是h*w，则图片左下角的xy坐标是(0,0)，右上角的xy坐标为(w,h)，而uv坐标则是将[0,w]和[0,h]映射成[0,1]，所以在uv坐标的表示下，图片的左下角uv坐标为(0,0)，右上角坐标为(1,1)，而将uv坐标换算成xy坐标的公式其实就是将uv从[0,1]插值成xy的[0,w]和[0,h]，则:

$$x = u * h$$

$$y = v * v$$

图片中有像素这个概念，也就是取得图片(x,y)处某一像素的颜色值，而纹理也有类似的纹素的概念，即取得纹理(u,v)处的颜色值，通常取纹素值就是直接取得对应xy坐标处的像素值，当然有的时候也会略有不同，比如不仅仅取得xy处的像素，同时还在xy处的像素周围取得额外的像素，然后做一个高斯模糊之类的操作。当然这些都不是本节的重点了。

本节将介绍如何给三角形添加纹理：/chapter4/Texture.main.cpp

先创建一个用于将uv坐标转换成x,y坐标的函数

```
//把uv坐标转换成xy坐标
void UV2XY(double u,double v,double &x,double &y,double w,double h)
{
    x = u * w;
    y = v * h;
    x = min(max(x, 0), w);//把x限制在[0,w]区间
    y = min(max(y, 0), h);//把y限制在[0,h]区间
}
```

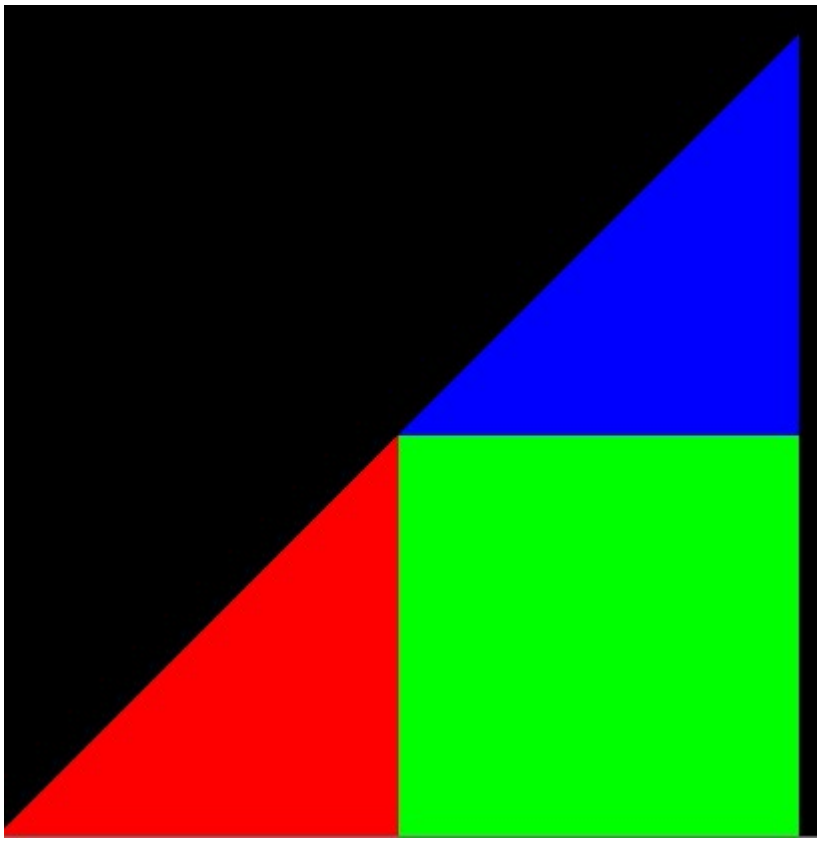
文中对x和y进行了限制，这样即使给出了超出[0,1]范围的uv也不会造成程序错误。
然后创建一个获取像素颜色的函数。

```
COLORREF picture[2][2] = {//创建一个2*2的二值图片
        {RGB(255,0,0),RGB(255,255,255)},
        {RGB(0,255,0),RGB(0,0,255)}
    };//获取图片颜色
COLORREF getPixel(int x,int y)
{
    return picture[x][y];
}
```

在插值上一章插颜色值的地方改成插uv值:

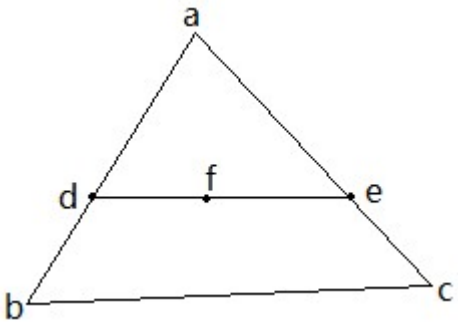
```
double u, v;//纹理的uv
u = WeightA * coordinate[0] + WeightB * coordinate[2] + WeightC * coordinate[4];//使用线性插值计算当前绘制像素的u值
v = WeightA * coordinate[1] + WeightB * coordinate[3] + WeightC * coordinate[5];//使用线性插值计算当前绘制像素的v值
double px,py;//图片的x,y
UV2XY(u, v, px, py, 2, 2);//将uv转换成xy坐标
COLORREF c= getPixel((int)px, (int)py);
gp.setPixel(x, y, c);//绘制像素
```

运行效果:



5: 双线性插值

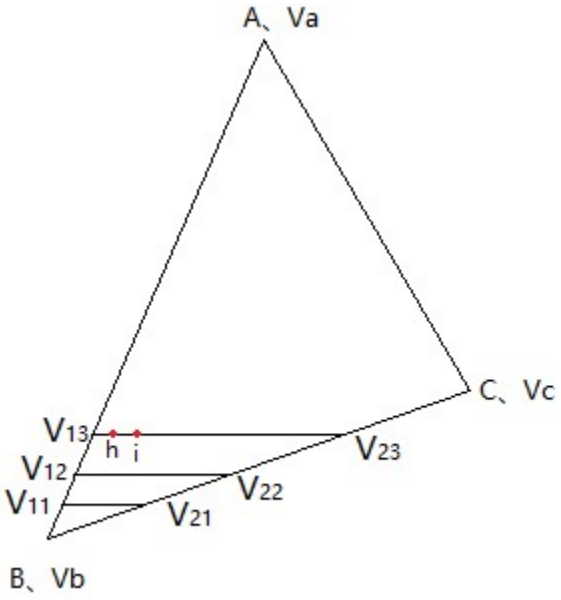
如下图所示：



已知Δabc和待插值点f，我们除了使用面积比计算插值点之外还可以使用双线性插值来计算。可以先作一条辅助线经过f，并且使得该直线和三角形相交至少两点(当f在三角形的某条边上时，辅助线如果和该边重合，可以认为有无数个交点)，则我们可以在辅助线上面选两个交点并使用线性插值计算交点的值，然后再在辅助线上面使用线性插值计算待插值点f的值。如上图中，我们可以使用线段ab和线段ac分别插值计算出d和e的值(我们可以发现对于d点来说，顶点c对d的权重为0，因为顶点c权重对应的Δabd的面积为0，所以可以直接在线段ab上使用线性插值计算d点的值)。这种在两个方向上面进行插值方法叫做双线性插值，双线性插值和重心坐标插值的结果一致。

6：双线性插值在扫描线上面的优化

对对应于上一节中的图，如果我们使得线段de和扫描线重合，并且我们知道对于线性变化的属性来说，在同一直线上面，属性的变化量和在直线上面的位移量也是线性相关的。所以我们可以先根据顶点属性值求出某属性d、e两点在y方向上面的增量，然后在绘制横向扫描线时，根据d、e属性值求出属性在该扫描线上面(即x方向)的增量，这样即可节省运行时间(重心坐标插值每个像素的一个属性值需要计算至少两个小三角形面积，最后一个三角形的权值可以用1减去前两个的权值，大的可以提前计算，需要4次乘法和两次加法，然后计算插值结果的时候需要3次乘法和加法。而双线性插值对于同一行扫描线的各个像素只需要一个累加操作即可)。如下图所示：



上图中的横线表示扫描线，我们先在边AB上面使用Va和Vb计算属性在扫描线上面的增量DV_1。因为扫描线都是平行的，所以可以知道在边AB上$V_{11} - V_b = V_{12} - V_{11} = DV_1$(因为这些扫描线平行，且距离一样，可知这些扫描线和边的各个交点也是等距离的，则这些点的差

值是相等的，依次增加DV_1。并且在BC边和CA边也是一样的)。这样可以得到$V_{11} = V_b + DV_1$，$V_{12} = V_{11} + DV_1$……。同样的我们使用BC边计算DV_2得到$V_{21} = V_b + DV_2$，$V_{22} = V_{21} + DV_2$……。然后假设我们现在在绘制$V_{13}V_{23}$对应的这条扫描线，我们通过$V_{13}V_{23}$计算出属性在该扫描线x方向的增量DxV(因为在一条横向的扫描线上面，各个像素间距也是一样的，则各个像素的属性也是依次增加DxV)，则我们可以知道$V_h = V_{13} + DxV$，这样依次简单的递增即可求出所有像素对应的属性值。

相关代码见chapter4/BilinearInterpolation/main.cpp

我们在Edge结构体中新增了四个变量:

```
double U;//属性的起始值
double V;
double DU;//记录当前边上属性在不同扫描线上面的增量
double DV;
```

用以记录本条边的UV属性初值和在不同扫描线的增量，并且DU和DV的计算方法和DX的计算方法一样，

```
dx = (ps[(i + 1) % 3].X - ps[i].X) / (ps[(i + 1) % 3].Y - ps[i].Y);
du = (coordinate[((i + 1) % 3) * 2] - coordinate[(i % 3) * 2]) / (ps[(i + 1) % 3].Y - ps[i].Y);
dv = (coordinate[((i + 1) % 3) * 2 + 1] - coordinate[(i % 3) * 2 + 1]) / (ps[(i + 1) % 3].Y - ps[i].Y);
```

我们在填充扫描线的时候也使用和处理X坐标类似的方法，只不过在同一条扫描线上面,X的增量是1，所以x使用了X++来处理，而UV的增量则需要通过计算UV在扫描线上随X的增量，填充扫描线的代码如下:

```
double su = edgeStar->U;//取得当前扫描线上面X,U,V初始值和结束值
double eu = edgeEnd->U;
double sv = edgeStar->V;
double ev = edgeEnd->V;
double sx = edgeStar->X;
double ex = edgeEnd->X;

double dxu, dxv;
dxu = (eu - su) / (ex - sx);//计算属性在扫描线上面的增量
dxv = (ev - sv) / (ex - sx);

int x = sx;//初始化当前像素的x,u,v属性值
double u = su;
double v = sv;
for (; x < ex; x++)//绘制这对交点组成的线段
{
    double px, py;//图片的x,y
    UV2XY(u, v, px, py, 2, 2);//将uv转换成xy坐标
    COLORREF c = getPixel((int)px, (int)py);
    gp.setPixel(x, y, c);//绘制像素
    u += dxu;
    v += dxv;
```

}

通过使用计时函数我们可以看到使用重心坐标插值的方法花费时间比大约为双线性插值的两倍左右(在不同机器或者不同时刻测试结果可能会不一样，但是偏差不会太大)，使用双线性插值可以在一定程度上优化计算量。下面是Texture和BilinearInterpolation两个项目的绘制耗时对比，其中耗时长的是Texture(使用重心坐标插值)、耗时短的是BilinearInterpolation(将Texture项目改造成双线性插值)

Debug版本，重心坐标插值绘制耗时:25.281000 毫秒

Debug版本，双线性插值绘制耗时:10.006000 毫秒

Release版本，重心坐标插值绘制耗时:1.404000 毫秒

Release版本，双线性插值绘制耗时:0.727000 毫秒

7:裁剪与插值

假如现在有一个线段AB，同时他们包含了顶点属性值Va和Vb，现在需要对线段进行裁剪,设裁剪点为C，则$V_c = W_a * \mathrm{V}_a + (1 - W_a)\mathrm{V}_b$。因为我们使得直线上的所有属性都是线性相关的，所以假设现在裁剪掉点A，那么我们可以用CB进行后续的线性插值。

8:任意属性的插值

在直线上，我们可以针对直线的任意属性进行插值，之前我们都是已知待插值点P的坐标，然后计算出t(W_a)，使用$V_p = W_a * \mathrm{V}_a + (1 - W_a)\mathrm{V}_b$来计算出P的属性。实际上我们不仅仅能计算一些额外属性值，如果我们能够计算出t，则可以使用t插值出p的坐标。例如已知点A(1,1),B(2,2)，$\mathrm{V}_a = 5, \mathrm{V}_b = 6, \mathrm{V}_p = 5.5$,那么我们使用公式$V_p = W_a * \mathrm{V}_a + (1 - W_a)\mathrm{V}_b$很容易计算出$W_a$=0.5,这时候可以将$X_p$和$Y_p$也当作属性，计算出P的坐标为(1.5,1.5)。当然，假如在$\mathrm{V}_a = \mathrm{V}_b = \mathrm{V}_p$的情况下，我们计算不出t的结果来，这时候插值就有点麻烦了，比如A(0,0), B(0,6),现在已知点P的x为0，求点P的y值，如果把X值当作属性，Y值当作坐标，t值无法计算出来，即：$0W_a + (1 - W_a)0 = 0$,本式无法确定t值，自然也算不出Y值。但是另外一种情况A(0,0), B(6,6),已知P的x为3,则t=0.5,x=3。我们可以发现，二维直线的线性插值其实是一维数轴的一种推广，坐标也可以当作一种额外属性进行插值。

9:三维空间体积插值的推广

如果在三维空间中插值，则是由四个不共面的顶点组成一个四面体，将待插值点和各个顶点连接成直线，各个顶点的权值之比等于该顶点对面和待插值点围成的四面体体积之比，并且四个顶点的权值之和为1。同样我们可以使用重心坐标插值或者三线性插值的方法来计算