同濟大學

TONGJI UNIVERSITY

装

订

线

《Soft Render》

Final report

项目名称	Soft Render
小组成员	厚志安（1452183）
（G2432）	韩海洲（1452219）
	黄仕坤（1452189）
	谢天皓（1452333）
	吴岩（1552207）
学院（系）	电子与信息工程学院
专　　业	计算机科学与技术
指导教师	赵君峤
日　　期	2016 年 12 月 21 日

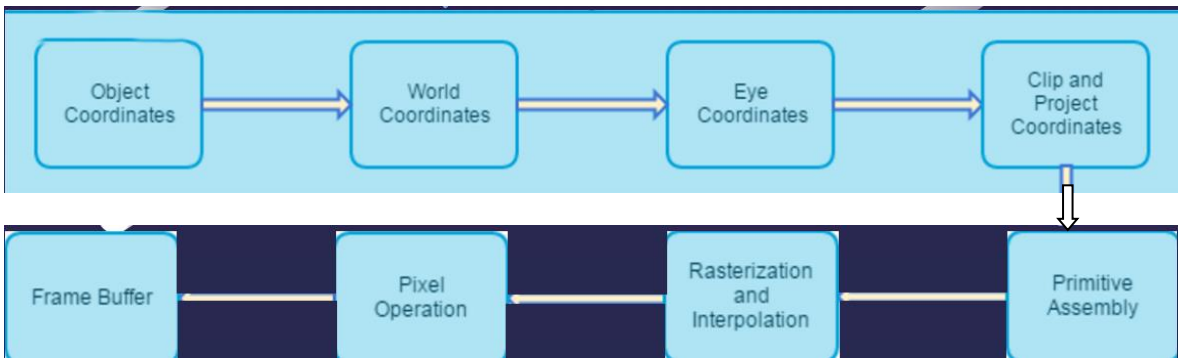
- 项目目标 **original goal of our project**

- 线框渲染
- CVV 裁剪
- 透视纹理映射
- 深度缓冲

- 项目分工 **Task distribution**

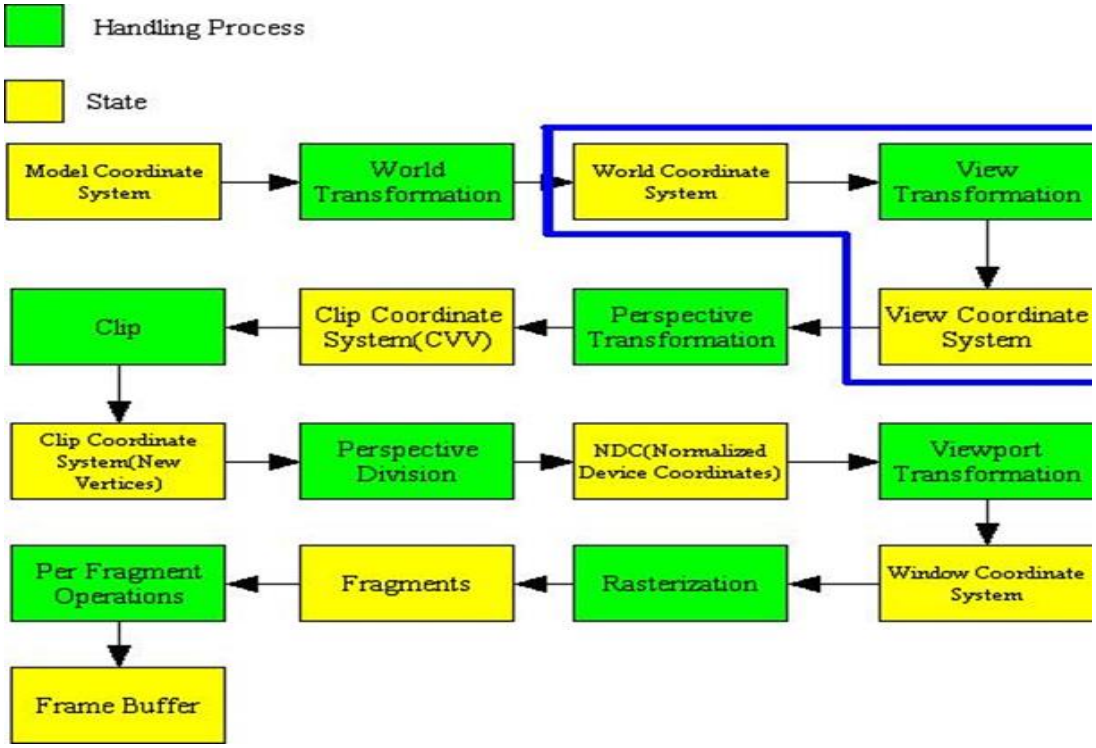
	Algorithm	**Coding**	**Other**
1452183 厚志安	Depth test and other	Main framework	Task distribution, schedule control
1452189 黄仕坤	Perspective texture mapping	Texture mapping	
1452219 韩海洲	Rasterization	Linear rasterization	
1452333 谢天皓	Camera transformation matrix	Matrix operation library	
155220 吴 岩	Perspective projection transformation	Matrix operation library	

- 图形流水线 **Graphics pipeline**

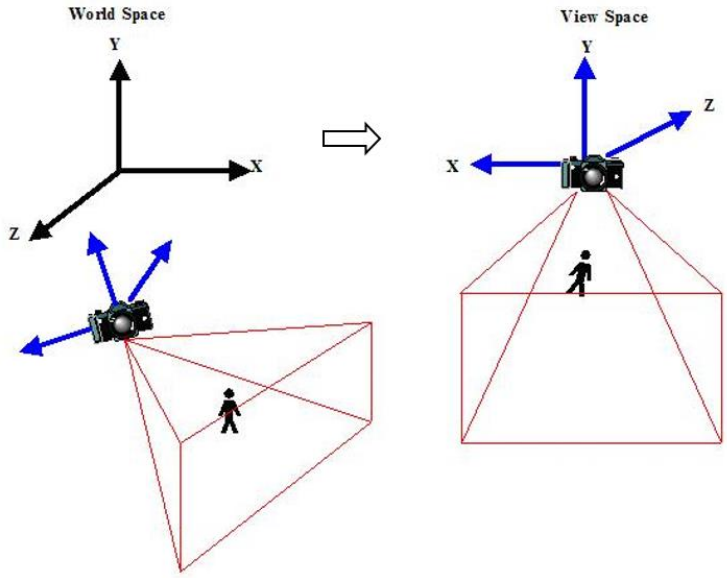


- **推导相机矩阵 Camera transformation matrix**

在流水线中，当物体从模型坐标通过世界矩阵变换到世界空间之后，它将通过相机变换从世界空间变换到相机空间。
下图的固定流水线中，蓝色框中的部分就是这个过程。



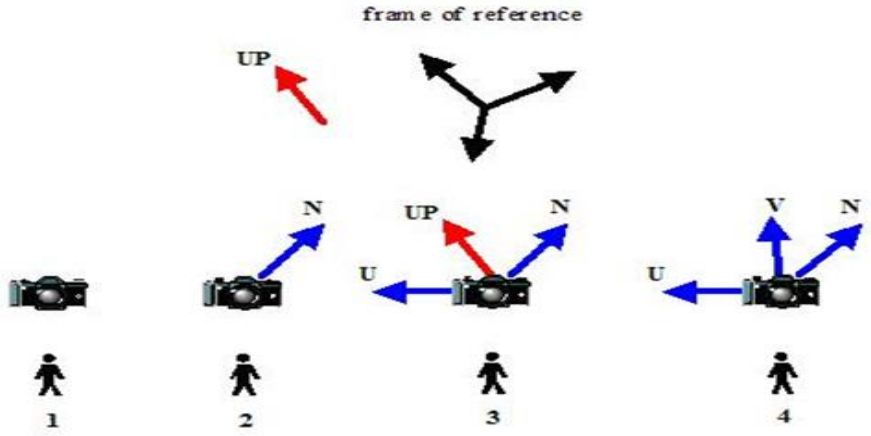
示例：



左半部分是小人在世界空间中的位置，右半部分是小人变换到相机空间后的位置。

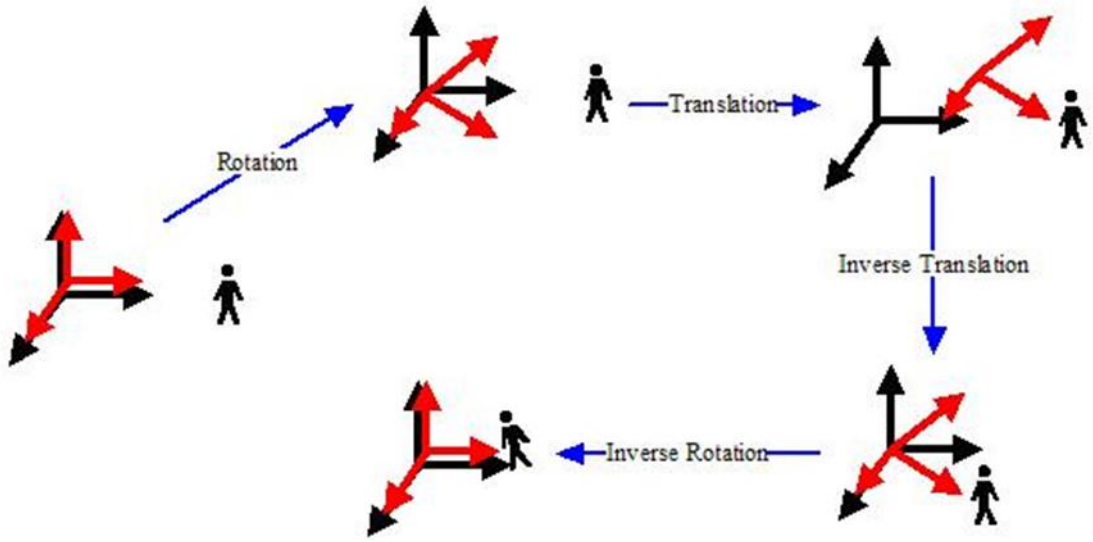
推导：

建立 UVN 系统：
如下图。
小人位置：lookat；相机位置：eye；辅助向量——参考系中表示“上方”的向量 UP



其中，N = lookat - eye ； U = UP × N ； V = N × U
用这个方式，我们就能构建相机的坐标系，为后面的矩阵变换提供了必要条件。

矩阵推导：



推导相机矩阵，其实就是在已知世界坐标下小人的矩阵，以及相机坐标系之后，如何推导小人在相机坐标系里的矩阵的过程。

首先。T 是平移变换矩阵，R 是旋转变化矩阵。相机本身矩阵 C = TR ，可以推出 。其中 C 是相机本身的矩阵，而由于相机矩阵变换就是相机本身矩阵的逆矩阵，所以 C-1 即为所求。其中，容易求得：

$$T^{-1}=\begin{pmatrix}1 & 0 & 0 & -Tx\\ 0 & 1 & 0 & -Ty\\ 0 & 0 & 1 & -Tz\\ 0 & 0 & 0 & 1\end{pmatrix}$$

下面的式子表示，参考系中向量 v 在基 Q 中的坐标是 v’，在基 R 中的坐标是 v’’

$$v = Qv' = Rv''$$

$$v=\begin{pmatrix}Q_i & Q_j & Q_k\end{pmatrix}\begin{pmatrix}v_x'\\ v_y'\\ v_z'\end{pmatrix}=\begin{pmatrix}R_i & R_j & R_k\end{pmatrix}\begin{pmatrix}v_x''\\ v_y''\\ v_z''\end{pmatrix}$$

根据矩阵变换的知识，已知一个基 Q 和向量 v 在它之中的坐标 v’，以及另外一个基 R，我们可以通过 v=Qv’=Rv’’公式来计算 v’’。

$$v = Qv' = Rv'' \Rightarrow$$

$$v'' = R^{-1}Qv'$$

同样地，我们将经过逆平移的基作为 Q，对于世界坐标系中的向量 v’，它在坐标系 R 中的坐标是 v’’带入以下式子，可以根据相同的计算方法求得：

$$R=\begin{pmatrix} Ux & Vx & Nx & 0 \\ Uy & Vy & Ny & 0 \\ Uz & Vz & Nz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

其中， $v''=R^T T^{-1} v$

求出 R 和 T 之后，就可以带入公式，即可求得

$$C^{-1}=\begin{pmatrix} Ux & Uy & Uz & 0 \\ Vx & Vy & Vz & 0 \\ Nx & Ny & Nz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & -Tx \\ 0 & 1 & 0 & -Ty \\ 0 & 0 & 1 & -Tz \\ 0 & 0 & 0 & 1 \end{pmatrix}=\begin{pmatrix} Ux & Uy & Uz & -U\bullet T \\ Vx & Vy & Vz & -V\bullet T \\ Nx & Ny & Nz & -N\bullet T \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

这就是相机变换矩阵。

装 订 线

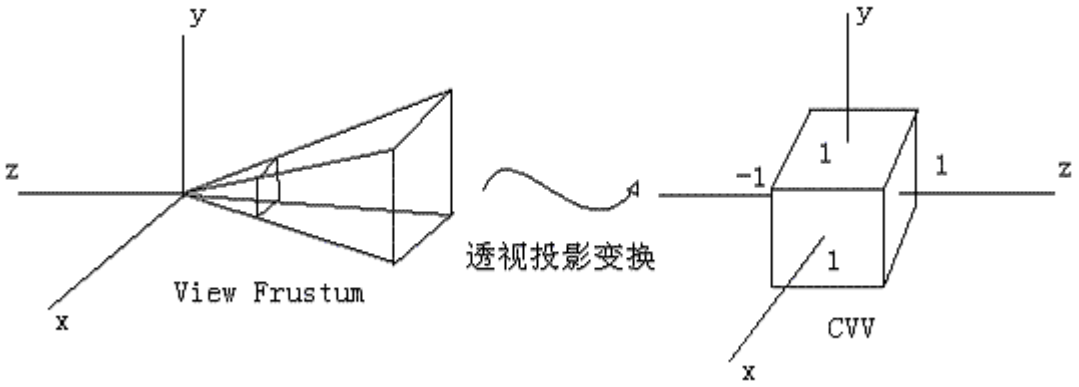
• 透视投影变换 Perspective Projection

经过相机矩阵的变换，顶点被变换到了相机空间。这时，多边形也许会被视锥体裁剪，但在这个不规则的体中进行裁剪并非容易，所以裁剪被安排到规则观察体(Canonical View Volume, CVV)中进行，CVV 是一个正方体，x, y, z 的范围都是[-1，1]，多边形裁剪就是用这个规则体完成的。

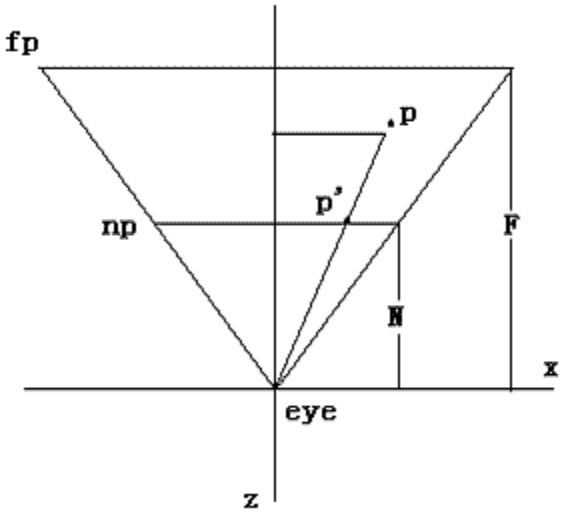
所以，事实上是透视投影变换由两步组成：

1) 用透视变换矩阵把顶点从视锥体中变换到裁剪空间的 CVV 中；

2) CVV 裁剪完成后进行透视除法。



先按照一个方向来考察投影关系。



上图是右手坐标系中 顶点在相机空间中的情形。

设 P(x,z)是经过相机变换之后的点，视锥体由 eye——眼睛位置，np——近裁剪平面，fp——远裁剪平面组成。N 是眼睛到 近裁剪平面的距离，F 是眼睛到远裁剪平面的距离。投影面可以选择任何平行于近裁剪平面的平面，这里我们选择近裁剪平面作为投影平面。设 P′(x′,z′)是投影之后的点，则有 z′ = -N。

通过相似三角形性质，我们有关系：

$$\frac{x}{x'} = \frac{z}{z'} = \frac{z}{-N}$$

$$x' = -N\frac{x}{z}$$

同理，有

$$y' = -N\frac{y}{z}$$

这样，便得到了 P 投影后的点 P′：

$$p' = \left(-N\frac{x}{z} \quad -N\frac{y}{z} \quad -N \right)$$

装

从上面可以看出，投影的结果 z′始终等于-N，在投影面上。实际上，z′对于投影后的 P′已经没有意义了，这个信息点已经没用了。但对于 3D 图形管线来说，为了便于进行后面的片元操作，例如 z 缓冲消隐算法，有必要把投影之前的 z 保存下来，方便后面使用。因此，我们利用这个没用的信息点存储 z，处理成：

$$p' = \left(-N\frac{x}{z} \quad -N\frac{y}{z} \quad z \right)$$

订

这个形式最大化地使用了 3 个信息点，达到了最原始的投影变换的目的，结合 CVV 进行思考，整理为以下的形式：

$$p' = \left(-N\frac{x}{z} \quad -N\frac{y}{z} \quad -\frac{az+b}{z} \right)$$

线

那么我们就可以使用矩阵以及齐次坐标理论来表达投影变换：

$$\begin{pmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} Nx \\ Ny \\ az+b \\ -z \end{pmatrix} \quad \text{齐次坐标变普通坐标} \quad \begin{pmatrix} -Nx/z \\ -Ny/z \\ -(az+b)/z \\ 1 \end{pmatrix}$$

其中

$$P=\begin{pmatrix}x\\y\\z\\1\end{pmatrix}\qquad P'=\begin{pmatrix}-Nx/z\\-Ny/z\\-(az+b)/z\\1\end{pmatrix}$$

注意在把 $\begin{pmatrix}Nx\\Ny\\az+b\\-z\end{pmatrix}$ 变成 $\begin{pmatrix}-Nx/z\\-Ny/z\\-(az+b)/z\\1\end{pmatrix}$ 的一步是使用齐次坐标变普通坐标的规则完成的。这一步在透视投影过程中称为透视除法（Perspective Division），这是透视投影变换的第 2 步，经过这一步，就丢弃了原始的 z 值（得到了 CVV 中对应的 z 值），顶点即完成了投影。而在这两步之间的就是 CVV 裁剪过程，所以裁剪空间使用的是齐次坐标 $\begin{pmatrix}Nx\\Ny\\az+b\\-z\end{pmatrix}$。

装

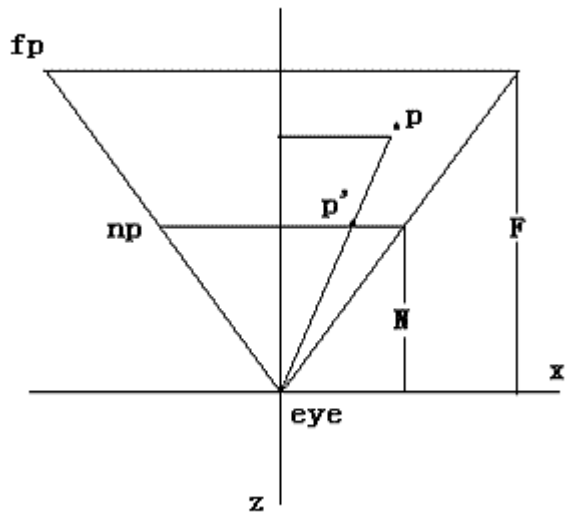
订

线

- **透视纹理映射 Perspective Projection**

数学推导

推导透视投影变换的时候用到的关系图：



上图是在相机空间的俯视图，eye 是眼睛的位置，也就是原点。np 和 fp 分别是近、远裁剪平面，N 和 F 分别是 z=0 到两个裁剪平面的距离。pq 是一个三角形 pqr 在 xy 平面上的两个点，p 的坐标为（x, y, z），p’ 是 p 投影之后的点，坐标为(x’, y’, z’)，则有

$$(1)\begin{cases} x' = -N\dfrac{x}{z} \Rightarrow x = -\dfrac{x'z}{N} \\ y' = -N\dfrac{y}{z} \Rightarrow y = -\dfrac{y'z}{N} \end{cases}$$

另外，在相机空间中，三角形 pqr 是一个平面，因此它内部的每一条边上的 x 和 z，以及 y 和 z 都是线性关系，即

$$(2)\begin{cases} x = Az + B \\ y = Az + B \end{cases}$$

这样，把上面投影之后的结果（1）带入这个线性式（2），有

$$-\frac{x'z}{N}=Az+B\Rightarrow$$

$$-\frac{x'}{N}=A+\frac{B}{z}\Rightarrow$$

$$x'=-N\frac{B}{z}-AN\Rightarrow$$

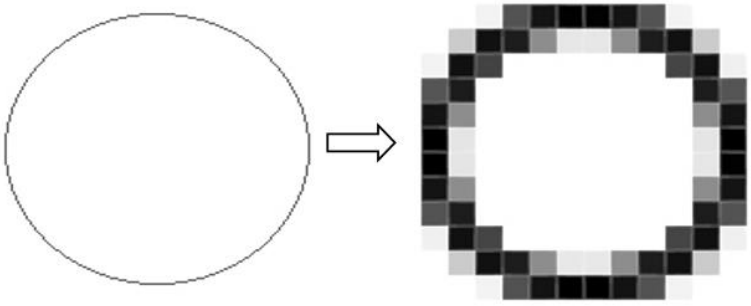
$$(3)\begin{cases}x'=C\dfrac{1}{z}+D\Rightarrow\dfrac{1}{z}=Ax'+B\\ y'=E\dfrac{1}{z}+F\Rightarrow\dfrac{1}{z}=Ay'+B\end{cases}$$

通过这个式子推出了投影之后的 x’和原始 z 之间的关系——x’和 1/z 是线性关系，y’和 1/z 也是线形关系。我们的到了透视纹理映射的思路：在投影平面上通过 x’和 y’对 1/z 线性插值，计算出 1/z 后，通过上面的（1）式计算出原始的 x 和 y，然后在 3D 空间中通过 x 和 y 计算出 s 和 t（x、y 和 s、t 都是在 3D 空间中的三角形上定义的，是线性关系）。这样就找到了投影面上一个点所对应的纹理坐标的正确值了。

• 光栅化 Rasterization

光栅化：光栅化是一种将几何图元变为二维图像的过程。该过程包含了两部分的工作。第一部分工作：决定窗口坐标中的哪些整型栅格区域被基本图元占用；第二部分工作：分配一个颜色值和一个深度值到各个区域。光栅化过程产生的是片元。

示例：



应用示例

1.直线

Bresenham 算法:

假设：k=dy/dx。因为直线的起始点在象素中心，所以误差项 d 的初值 d0＝0。

X 下标每增加 1，d 的值相应递增直线的斜率值 k，即 d＝d＋k。一旦 d≥1，就把它减去 1，这样保证 d 在 0、1 之间。

当 d≥0.5 时，最接近于当前象素的右上方象素（x+1,y+1）

而当 d<0.5 时，更接近于右方象素(x+1,y）

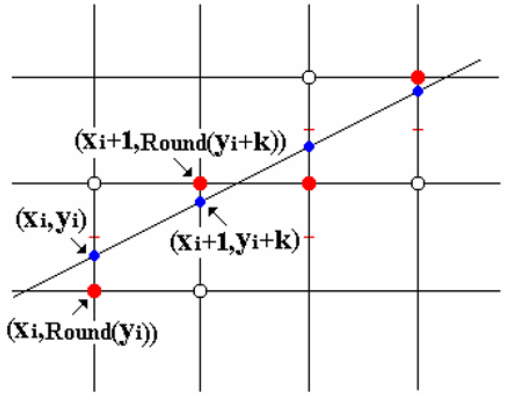
为方便计算，令 e＝d-0.5，e 的初值为-0.5，增量为 k。

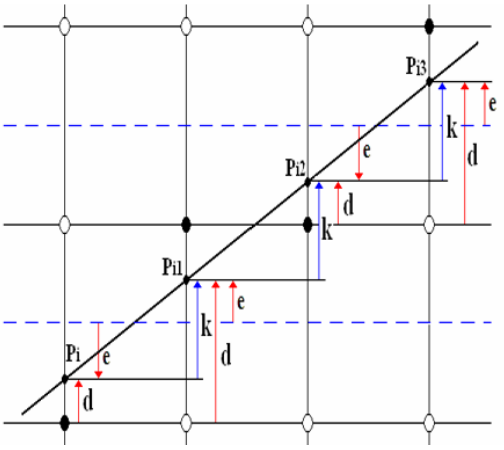
当 e≥0 时，取当前象素（xi，yi）的右上方象素（x+1,y+1）

而当 e<0 时，更接近于右方象素(x+1,y）

可以改用整数以避免除法。由于算法中只用到误差项的符号，因此可作如下替换：

e1 = 2*e*dx





2.三角形

平底三角形：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。

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。

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：

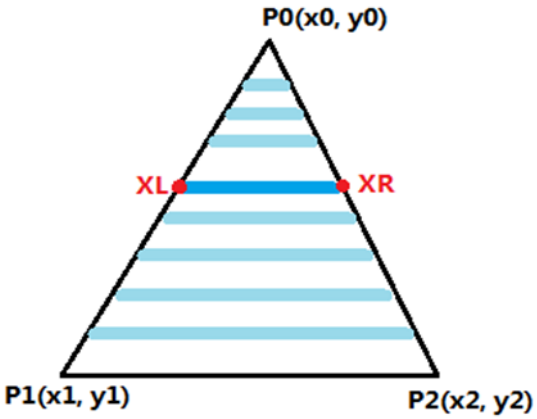
对于已知直线上的两点(x1,y1)和(x2,y2)有：

(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)

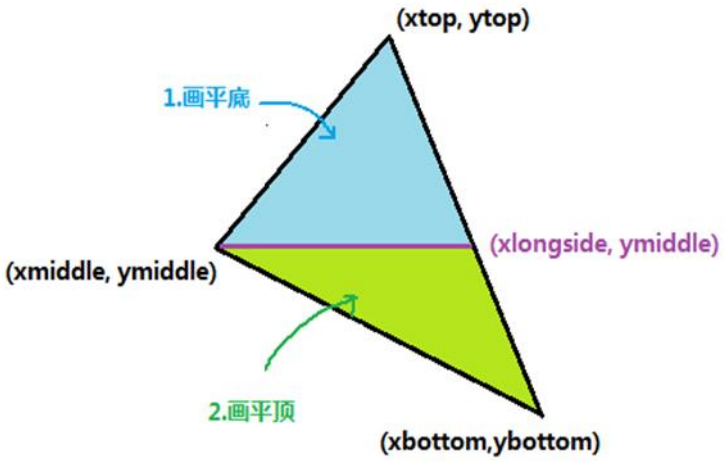
我们只要把 P0,P1 代入，就可以求得 XL，把 P0,P2 代入，就可以求得 XR。



任意三角形：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。

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，先对顶点进行排序，然后再分成平底和平底三角形。


(xtop, ytop)
1.画平底
(xlongside, ymiddle)
(xmiddle, ymiddle)
2.画平顶
(xbottom,ybottom)

装

订

线