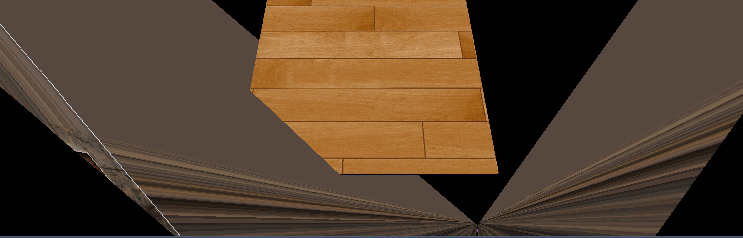
1. UV错位：  
   scanline的时候一开始写的right.x – left.x求出line的最右端减最左端的一个单位平均步长值step，后面从左到右插值的时候 left + i\*step   
   因为right.x和left.x都是float，后面由可能计算的是1.3到10.1之间的单位步长，  
   但是填入屏幕像素的时候是填整数2到11的位置，所以这个小数点误差造成了错位。  
   先把left和right插值到2.0和11.0再计算单位步长step或者每次i++的时候重新计算新的left插值都可以。
2. 纹理边缘会有彩色点  
     
   计算uv位置的时候直接v\*\_uv\_size + u，因为比如此时纹理是256的贴图，其实再内存里取值应该是0-255.直接用256就多出来一位，取到超出内存块最后的值，所以应该是用\_uv\_max\_size = \_uv\_size -1来给上面v乘。
3. 相机和观察对象移到很近的时候会把对象拉花  
   

因为相机和观察很近的时候mvp\*v得到的w分量是小于零的数，此时w已经失真了。  
∵iew坐标系是先移动到世界坐标原点，再旋转朝向到指定角度；  
又∵ject\*P(x,y, z,1) =>p’(x’,y’,z’,w)， x’ = d\*x/z; y’=d\*y/z; z’=(a\*z+b)/z, w=z  
∴这个w是原先透视投影前的z. 当w<0.0f的时候其实就是在相机后面了。

(这个最坑，一直以为是裁剪写得不对)。

1. 正反面剔除写在归一化之前，导致法线不对。正反面剔除应该在归一化之后。

Pipeline\_Process:

1. model = WVP\*V;

2. normalization(){postion/w, uv/w, color/w}

3. backface\_culling();

4. clip(){split\_triangle()}

5. toscreen();

6. scanline(){postion, uv*\*w, color\**w}