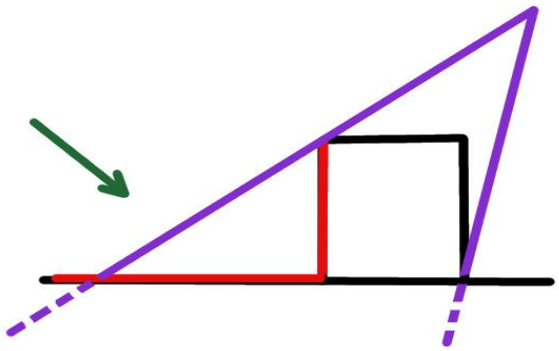
Ombres par Stencil Buffer.

Laurent Cancé Francis
12/03/2018

I. Concept des ombres par le Stencil Buffer

En considérant les arrêtes d'un maillage, il est possible de représenter le volume d'ombre par rapport à une direction ou un point de lumière grâce au Stencil Buffer en même temp en utilisant correctement le Zbuffer.

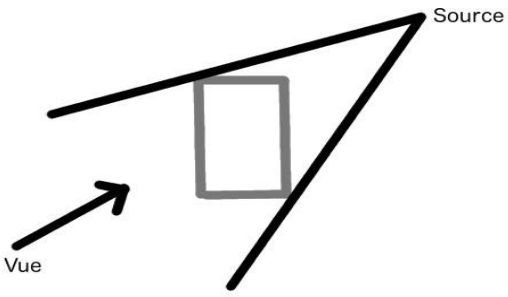


```
FrontFace.StencilFailOp = STENCIL_OP_KEEP;
FrontFace.StencilDepthFailOp = STENCIL_OP_INCR;
FrontFace.StencilPassOp = STENCIL_OP_KEEP;
FrontFace.StencilFunc = COMPARISON_ALWAYS;

BackFace.StencilFailOp = STENCIL_OP_KEEP;
BackFace.StencilDepthFailOp = STENCIL_OP_DECR;
BackFace.StencilPassOp = STENCIL_OP_KEEP;
BackFace.StencilFunc = COMPARISON_ALWAYS;
```

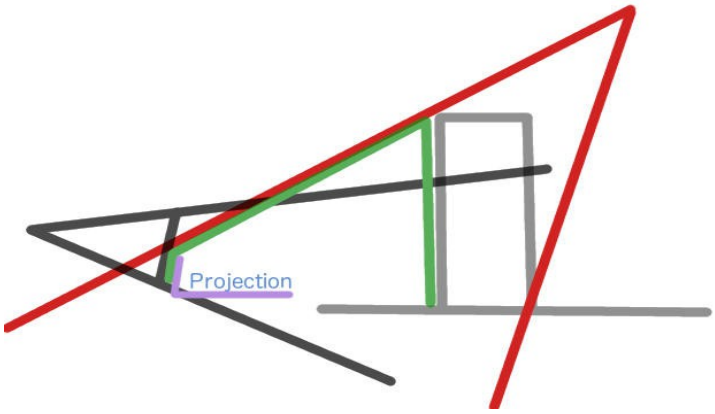
II. Le problème

Dans une mesure applicative, il convient de considérer le cas où le point de vue se situe à l'intérieur du volume d'ombre, ce que la modélisation par des arrêtes dégénérées des objets provoquant une ombre ne suffit pas à en dessiner la forme :



III. La solution optimale

La solution optimale est de considérer une fermeture du volume d'ombre par la topologie des objets provoquant une ombre en plus du volume engendré par les arrêtes.



Le principe est par la suite de projeter le volume d'ombre aussi sur l'écran, autrement dit le plan orthogonal au vecteur de vue de la scène 3D.

Une simple multiplication de matrice suffit et le fill rate est optimal, le tout juste dans l'application de la procédure pour les sommets.

```
MATRIX VIEWPROJ;                // Matrice VIEW * PROJ
MATRIX VIEW;                    // Matrice VIEW
MATRIX WORLD;                   // Matrice de transformation de l'objet
MATRIX PROJNEARPLANE;           // Matrice de Projection sur l'écran

[POSITION]

        p = iPos *4 WORLD;
        N = iNorm *3 WORLD;

        u = tmp - SourceDeLumiere;
        u = normalise( u );
        s = (u | N);

        s = slt( s , FrontZ );

        s = s * ContinuiteMaillageArrete * LongueurDeVolumeDOmbre;
        u = tmp + s * u;
        s = u *4 VIEW;
        p = u;

        if ( s.z > FrontZ )
        {
         p = u;
        }
        else
        {
         p = u *4 PROJNEARPLANE;
        }

        oPos = p *4 VIEWPROJ;
```

La matrice de projection sur un plan est donnée par :

Matrice ShadowProjection(Vecteur Light,Plan P)

```
float dot=(Light||P);

a[0][0]=dot-Light.x*P.a;
a[1][0]= 0 -Light.x*P.b;
a[2][0]= 0 -Light.x*P.c;
a[3][0]= 0 -Light.x*P.d;

a[0][1]= 0 -Light.y*P.a;
a[1][1]=dot-Light.y*P.b;
a[2][1]= 0 -Light.y*P.c;
a[3][1]= 0 -Light.y*P.d;

a[0][2]= 0 -Light.z*P.a;
a[1][2]= 0 -Light.z*P.b;
a[2][2]=dot-Light.z*P.c;
a[3][2]= 0 -Light.z*P.d;

a[0][3]= 0 -Light.w*P.a;
a[1][3]= 0 -Light.w*P.b;
a[2][3]= 0 -Light.w*P.c;
a[3][3]=dot-Light.w*P.d;
```

à appliquer au plan (0,0,1,-ZNear) et Light(x,y,z)