```
1  // normal vector is n, n dot p is the point, let s, t be coordinates
2  Matrix4d coordFrame( const Vec3f &n, const Vec3f &p)
3  {
4    // if it is near x axis
5    Vec3f s,t;
6    if(n.x > 0.9f) {
7      s = Vec3f(0.0f, 1.0f, 0.0f );
8    } else {
9      s = Vec3f(1.0f, 0.0f, 0.0f);
10   }
11   s -= n* dot(s, n); // make s orthogonal to n
12   s *= rsqrt(dot(s, s)); // normalize s
13   t = cross(n, s);
14   return (new double[] {
15     t.x, s.x, n.x, p.x,
16     t.y, s.y, n.y, p.y,
17     t.z, s.z, n.z, p.z
18     0, 0, 0, 1
19   })
20 }
```

Assume the axis pass through (x_1, y_1, z_1) and (x_2, y_2, z_2)

(1) Create the axis passing through origin by translating space by $-P_1$ for example

$$T = \begin{pmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(2) Rotate axis into one of the original axis, Fix z, let u be a unit vector (p, q, r)

Project u onto yz-plane to get u' and rotate by α in order to get u in xz-plane

$$u' = \sqrt{q^2 + r^2}, \ cos\alpha = r/u' \ sin\alpha = q/u' \ Rx = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{r}{u'} & -\frac{q}{u'} & 0 \\ 0 & \frac{q}{u'} & \frac{r}{u'} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(3) We then rotate u' to overlap z-axis, $cos\beta = u'/||u|| = u' \ sin\beta = p/||u|| = p \ Ry =$
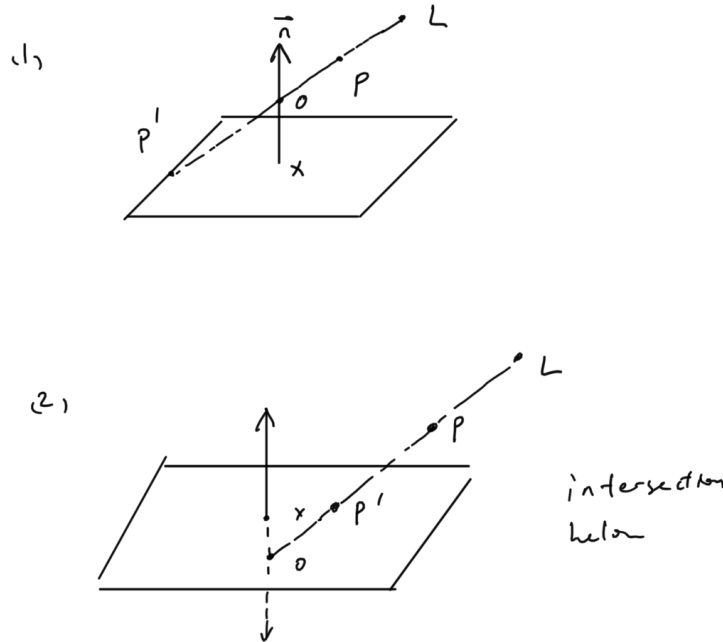$$\begin{pmatrix} u' & 0 & -p & 0 \\ 0 & 1 & 0 & 0 \\ p & 0 & u & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(4) We then rotate around z-axis by $\theta \ Rz = \begin{pmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Finally, $M = T^{-1} R_x^{-1} R_y^{-1} R_z R_y R_x T$

$$P = \begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Where this will take in point $(x, y, z, 1)^T$ to $(nx, ny, nz, -z)^T$, After dividing by the z coordinate we have $(-nx/z, -ny/z, -n, 1)$ which is the desired point of the near plane

(1)

(2)

intersection
below

As shown in (1) a point light project point $p \in \mathbb{R}^3$ onto the plane with normal n at p'. We let x be the origin of the canonical basis and n be the z-axis

$$L_{LP} = L + (p - L)t$$

$$L_n = x + nt$$

We let $L_{LP} = L_n$ and find their intersection point o.

Case 1: If o above x, then by Q1 we can construct a new frame, and its frame matrix F by point o and $\vec{n}$ then consider the plane to be the near plane with distance n.

The coordinate of p is $F^{-1}p$, and projection $PF^{-1}p$, where P is the projection matrix from Q3. Using similarity transform, $p' = FPF^{-1}p$ in the canonical basis

Case 2: If o is below x then construct F with o and $-\vec{n}$, we can get $p' = FPF^{-1}p$

Case 3: If o is at x, then $p' = x$

Let $s = \begin{pmatrix} 0 \\ 0 \\ -1 \\ 1 \end{pmatrix}$ be a point on the quadrilateral, and $t = \begin{pmatrix} 0 \\ 0 \\ -2 \\ 1 \end{pmatrix}$ be a point on the wall.

$$P = \begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+10 & 10n \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

$$P_s = \begin{pmatrix} 0 \\ 0 \\ -n - 10 + 10n \\ 1 \end{pmatrix}$$

$$P_t = \begin{pmatrix} 0 \\ 0 \\ -2n - 20 + 10n \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -n - 10 + 5n \\ 1 \end{pmatrix}$$

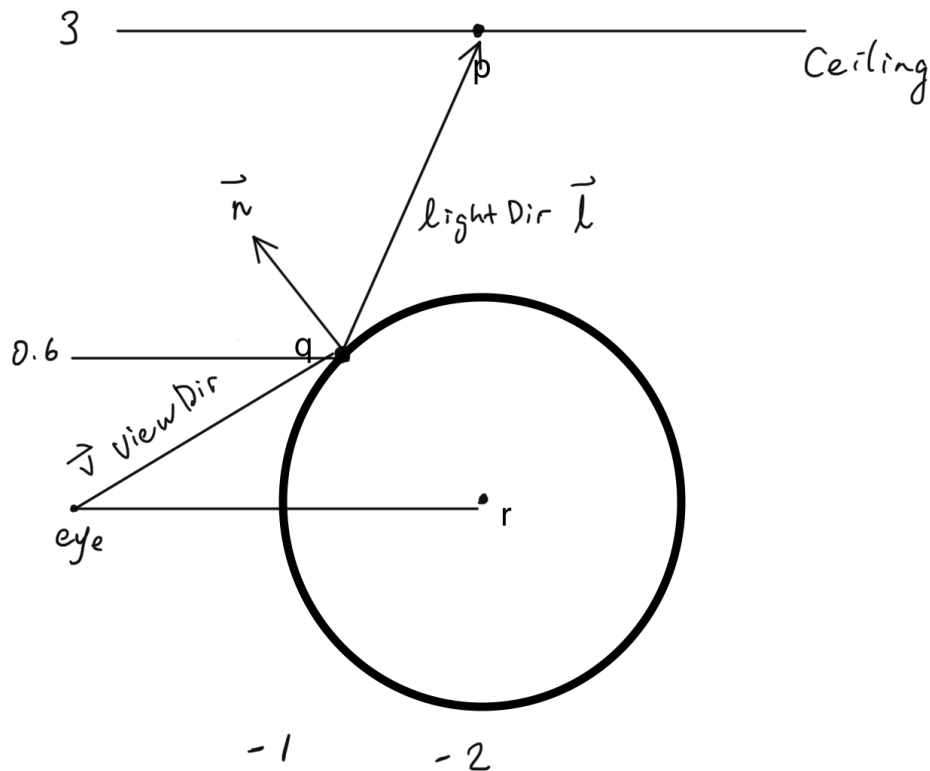We are only concerned about the z coordinates of those points

$z_{P_s} = -n - 10 + 10n$

$z_{P_t} = -n - 10 + 5n$

We want to minimize, i.e $|z_{P_s} - z_{P_t}| = \epsilon$ so:

$| - n - 10 + 10n - (-n - 10 + 5n)| = \epsilon$

$|5n| = \epsilon$

$n = \frac{\epsilon}{5}$

3 ⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯•⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯ Ceiling

p

$\vec{n}$

light Dir $\vec{l}$

0.6 ⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯ q

$\vec{v}$ View Dir

eye

r

- 1   - 2

Where p is the light position (unknown), r is the center of the circle, q is the brightest spot of a Blinn-Phong specular highlight.

We need to find the position of q first. Since we know $y_q = 0.6$, we get:

$$x_q^2 + 0.6^2 = 1$$

$$x_q = 0.8$$

Thus, $z_q = -2 + 0.8 = 1.2$

Now we can find the view direction v and the normal n.

$$n = q - r = \begin{pmatrix} 0 \\ 0.6 \\ 0.8 \\ 0 \end{pmatrix}$$

$$v = eye - q = \begin{pmatrix} 0 \\ -0.6 \\ 1.2 \\ 0 \end{pmatrix}$$

Now let's find l. Suppose u is a vector such that $v - 2u = l$

$$u = v - proj_n v = \begin{pmatrix} 0 \\ -0.6 \\ 1.2 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 0.36 \\ 0.48 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ -0.96 \\ 0.72 \\ 0 \end{pmatrix}$$

$$l = v - 2u = \begin{pmatrix} 0 \\ -0.6 \\ 1.2 \\ 0 \end{pmatrix} - 2 \begin{pmatrix} 0 \\ -0.96 \\ 0.72 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1.32 \\ -0.24 \\ 0 \end{pmatrix}$$

Now let's find out the location of the Light

Parametric equations of the direction: $x = 0 \quad y = 0.6 + 1.32t \quad z = -1.2 - 0.24t$

Equation of the ceiling:$y = 3$

$$3 = 0.6 + 1.32t$$

$$t = \frac{20}{11}$$

$$z = -1.2 - 0.24t$$

$$z = -\frac{18}{11}$$

So the light should be at $\begin{pmatrix} 0 \\ 3 \\ -\frac{18}{11} \\ 1 \end{pmatrix}$

Vertex Shader

```glsl
#version 400 core
uniform mat4 M;
// modeling matrix
uniform mat4 V;
// view matrix
uniform mat4 P;
// projection matrix
uniform mat3 MinvT; // inverse transpose of linear part of M
uniform mat3 VinvT; // inverse transpose of linear part of V

in vec3 VertexNormal;
in vec4 VertexPosition;

out vec4 PositionForFP; // camera coordinates
out vec3 NormalForFP;
// interpolate the normalized surface normal

void main() {

NormalForFP = MinvT * VinvT * VertexNormal; // Should change to NormalForFP=
    normalize(V*MinvT* vec4(VertexNormal,0))

PositionForFP = V * M * VertexPosition;

gl_Position = P * V * M * VertexPosition;
}
```

Fragment Shader

```glsl
1  #version 400 core
2  uniform vec3 LightColor;
3  // rgb intensity of light
4  uniform vec3 LightPosition; // light position in camera coordiantes
5  uniform float Shininess;
6  // exponent for Blinn-Phong model
7  uniform vec3 kd;
8  // diffuse material property
9
10 in vec4 PositionForFP; // fragment position in camera coordinates
11 in vec3 NormalForFP;
12 // normal at each fragment in camera coordinates
13
14 out vec4 FragColor;
15
16 void main() {
17
18 vec3 LightDirection = PositionForFP - LightPosition; // Should change to
       LightDirection = normalize(LightPosition - PositionForFP)
19
20 float diffuse = dot( NormalForFP, LightDirection ); // Should change to max(
       dot(NormalForFP, LightDirection), 0)
21
22 vec3 ViewDirection = vec3(0,0,0) - PositionForFP; // Should change to
       normalize (vec3(0,0,0) - PositionForFP)
23
24 HalfVector = (LightDirection + ViewDirection) / 2; // Should change to
       normalize (LightDirection + ViewDirection)
25
26 float specular = max(0.0, dot(NormalForFP, HalfVector));
27
28 if (diffuse == 0.0) {
29
30 specular = 0.0;
31
32 } else {
33
34 specular = pow( specular, Shininess );
35
36 }
37
38 vec3 scatteredLight = kd * LightColor * diffuse;
39
40 vec3 reflectedLight = LightColor * specular;
41
42 vec3 rgb = min( scatteredLight + reflectedLight, vec3(1,1,1) );
43
44 FragColor = vec4( rgb, 1 );
45 }
```

Yes, I actually looked into his work of OpenVDB, apparently its very widely used as a library of manipulating sparse volumetric data. In addition to Ken Museth's Keynote, I also attended Papers 2 Waves. I was only familiar with J. Tessendorf's Work (Clemson) prior to coming to SCA 2020, its very refreshing to see new development on procedural generation of ocean waves.