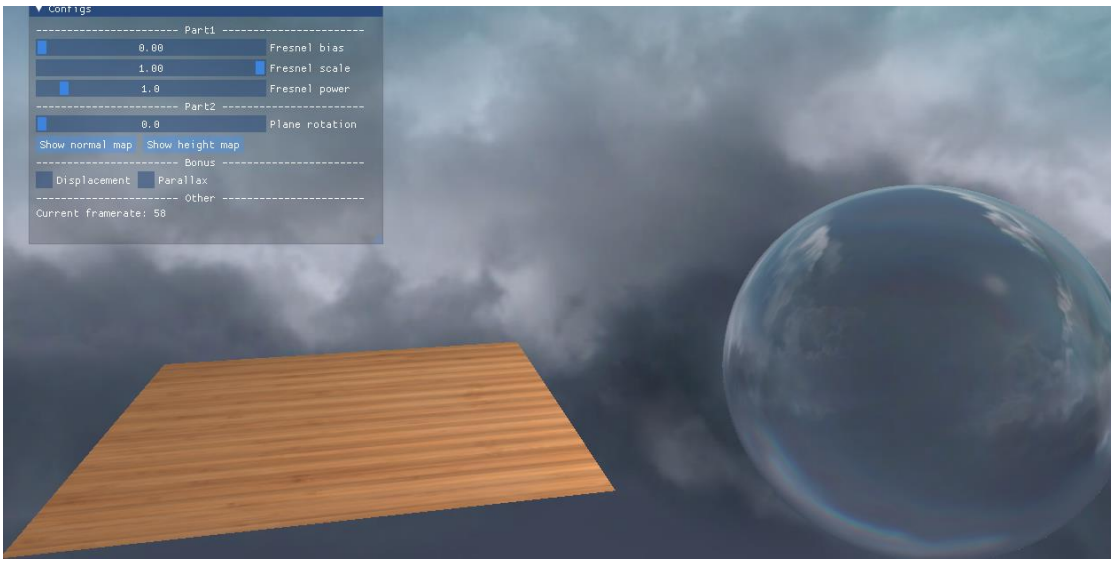
310605009 吳公耀 HW3

1. Skybox

Here, I made my skybox ,but it is infinite far from us so at last I change Z to 1.0 by gl_Position.xyww ,Also I add mat4(mat3(view)) in matrix to let me stay in the cube map. At first I did not add this so I can only see one square of cube map

```
void main() {
  textureCoordinate = position_in;

  gl_Position = projection  * mat4(mat3(view)) *vec4(position_in, 1.0);
  gl_Position = gl_Position.xyww;
}
```



2. fresnel

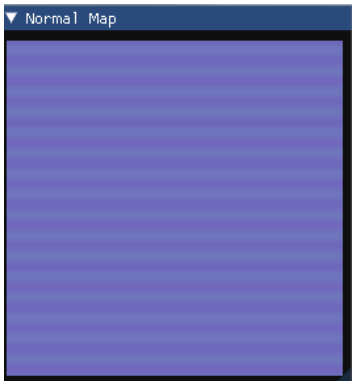
I make the I and N in the beginning to calculate the Reflect and the Refract , here I have only find lot of information that tell us how to calculate color red in refract but the Eta that give is RGB so I refract each color by their Eta and mix the Reflect , Refract and the fresnel radio.

```
vec3 i = normalize(fs_in.position - fs_in.viewPosition);
vec3 n =  normalize(fs_in.normal);
float F = clamp(fresnelBias + fresnelScale * pow(1 + dot(i, n), fresnelPower), 0.0, 1.0);

vec3 Reflect = reflect(i ,n);
vec4 reflectedColor = texture(skybox, Reflect);
vec3 RefractR = refract(i, n, Eta.r);
vec3 RefractG = refract(i, n, Eta.g);
vec3 RefractB = refract(i, n, Eta.b);
vec4 refractColor;
refractColor.r = vec3(texture(skybox, RefractR)).r;
refractColor.g = vec3(texture(skybox, RefractG)).g;
refractColor.b = vec3(texture(skybox, RefractB)).b;

FragColor =  mix(refractColor, reflectedColor,  F);
```

3. calculatenormal

Here I calculate the normal follow by spec by the problem I have mat is that I use the position_in instead of gl_FragCoord so I get all position wrong. Also I did not calculate all position z with different Z value so the normal will be all the same.

```
vec4 tmpposition = gl_FragCoord;
vec3 pos1 ;
pos1.x = tmpposition.x-delta;
pos1.y = tmpposition.y;
pos1.z =  sin(offset - 0.1 * pos1.y);

vec3 pos2;
pos2.x = tmpposition.x;
pos2.y = tmpposition.y-delta;
pos2.z =  sin(offset - 0.1 * pos2.y);


vec3 pos3;
pos3.x = tmpposition.x+delta;
pos3.y = tmpposition.y;
pos3.z =  sin(offset - 0.1 * pos3.y);

vec3 pos4;
pos4.x = tmpposition.x;
pos4.y = tmpposition.y+delta;
pos4.z =  sin(offset - 0.1 * pos4.y);

vec3 edge1 = normalize(pos2 - pos1);
vec3 edge2 = normalize(pos3 - pos1);
vec3 edge3 = normalize(pos3 - pos1);
vec3 edge4 = normalize(pos4 - pos1);


vec3 normal1 =  normalize(cross(edge1, edge2));
vec3 normal2 =  normalize(cross(edge3, edge4));
vec4 tmpnormal = normalize( (vec4((normal1 + normal2)/2,1.0 )));

normal =  tmpnormal* 0.5 + 0.5;
```



4. normalmap

Here is how I get the TBN matrix and also set the value for out in vertex

```
vec4 worldposition = modelMatrix*vec4(position_in,0.0);
vec3 T = normalize(vec3(modelMatrix * vec4(tangent_in,   0.0)));
vec3 B = normalize(vec3(modelMatrix * vec4(bitangent_in, 0.0)));
vec3 N = normalize(vec3(modelMatrix * vec4(normal_in,    0.0)));
mat3 TBN = transpose(mat3(T, B, N));
vs_out.TBN = transpose(mat3(T, B, N));
vs_out.lightDirection = TBN* lightDirection;
vs_out.position =TBN * (worldposition.xyz);
vs_out.viewPosition = TBN* (viewPosition.xyz);
```

Here is how I use Blinn-Phong shading , because of HW2 here I did not mat any big problem.

```
vec3 normal = texture(normalTexture,textureCoordinate).rgb;
normal = normalize(normal * 2.0 - 1.0 );
vec3 lightDir =  -normalize(fs_in.lightDirection);
float diff = 0.75*max(dot(normal,lightDir ), 0.0);
vec3 halfwayDir = normalize(lightDir+ viewDirectionection);
float spec = 0.75*pow(max(dot(normal, halfwayDir), 0.0), 8.0);


float cosTheta = clamp( dot( normal,lightDir ), 0,1 );
float lighting = ambient + diff + spec;
FragColor = vec4(lighting * diffuseColor, 1.0);
```