1、坐标

- canvas 坐标 左上角是原点（0,0）。X轴向右，Y轴向下。所以webgl 的Y需要反转。
- webgl是三维坐标系（笛卡尔坐标系）的原点是（0,0）在中心，右上角（1.0, 1.0, 0.0）。左下角是（-1.0, -1.0, 0.0）
- 纹理图像的坐标左下角是（0,0），右上角是（1,1）。但是webgl的坐标原点是（0,0）左下角是（-1.0,-1.0,0.0）

OpenGL将坐标限定在-1.0到1.0 有以下几个原因。

- 一种统一且高效的方式处理顶点数据，无论它们最初位于哪个坐标系统或具有多大的范围。
- 使用较少的浮点运算来处理顶点数据，从而减少了计算负担。这种标准化的坐标范围也能容易地进行各种图形变换，如平移、旋转和缩放等。
- 投影矩阵用于将3D坐标投影到2D屏幕上。使用-1.0到1.0的坐标范围可以使得投影矩阵的设计更加简单和直观。例如，正交投影矩阵可以将这个范围内的坐标直接映射到屏幕上的相应位置，而无需进行复杂的计算。
- OpenGL在将顶点数据渲染到屏幕上之前，会进行裁剪和视口变换等处理。裁剪操作会移除那些不在视锥体（View Frustum）内的顶点，而视口变换则负责将NDC坐标转换为屏幕上的像素坐标。使用-1.0到1.0的坐标范围可以使得这些处理过程更加高效和准确。

OpenGL的坐标系统限定在-1.0到1.0的范围内主要是出于标准化设备坐标、性能优化、投影矩阵的便利性、裁剪和视口变换的需要以及灵活性等方面的考虑。这种设计使得OpenGL能够以一种高效且统一的方式处理图形数据，并提供了丰富的功能和灵活性来满足不同的应用需求。

2、纹理坐标

- 顶点着色器接收纹理坐标，光栅化传给片元
- 片元根据纹理坐标，抽取纹理颜色，赋值给片元
- 设置顶点的纹理坐标（initVertextBuffers()）
- 准备待加载的纹理图像，浏览器来读取它（initTextures()）
- 监听纹理图像的加载事件，一旦加载完成，就在webgl系统中使用纹理（loadTexture()）

3、坐标变换

canvas尺寸是500x500，那如果我们的canvas尺寸不是500x500呢？让我们把canvas的尺寸改为`480x270`,结果又如何呢？变形了. 推导出一个坐标变换矩阵。

```
export function createProjectionMat(l, r, t, b, n, f) {
    return [
        2 / (r - l), 0, 0, 0,
        0, 2 / (t - b), 0, 0,
        0, 0, 2 / (f - n), 0,
        -(r + l) / (r - l), -(t + b) / (t - b), -(f + n) / (f - n), 1
    ]
}
```

4、相机原理

相机的本质其实是 一个 **观察者空间 + 投影变换。**

1. 首先，相机本身是具有一个坐标系，我们称其为“观察空间”
2. 相机合并了我们之前提高的投影变换，它可能是正交投影，也可能是透视投影。

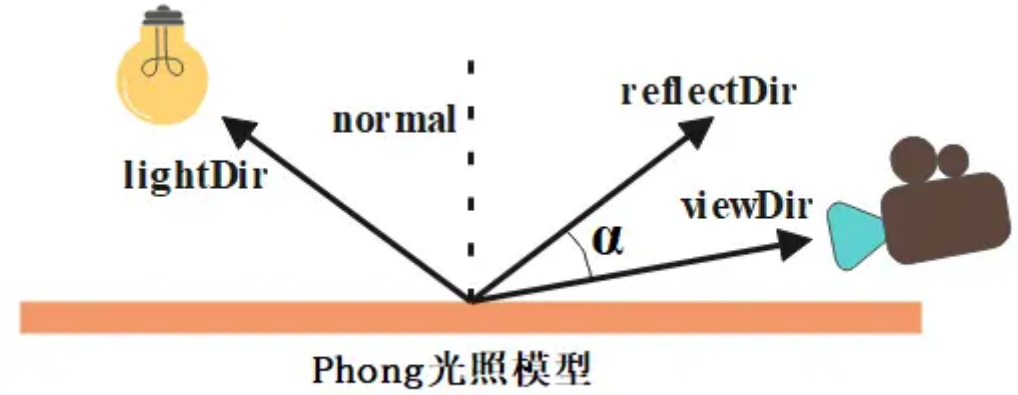
```
export function lookAt(cameraPos: vec3, targetPos: vec3): mat4 {
    const z = vec3.create();
    const y = vec3.fromValues(0, 1, 0);
    const x = vec3.create();
    vec3.sub(z, cameraPos, targetPos);
    vec3.normalize(z, z);
    vec3.cross(x, y, z);
    vec3.normalize(x, x);
    vec3.cross(y, z, x);
    vec3.normalize(y, y);

    // prettier-ignore
    return mat4.fromValues(
        x[0], x[1], x[2], 0,
        y[0], y[1], y[2], 0,
        z[0], z[1], z[2], 0,
        cameraPos[0], cameraPos[1], cameraPos[2], 1
    );
}
```

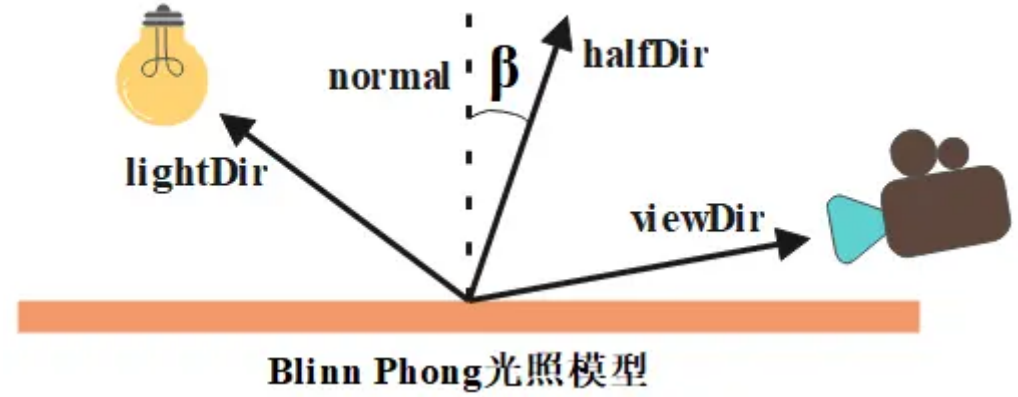
光照

光照元素主要有**环境光**（ambient）、**漫反射光**（diffuse）、**镜面反射光**（specular），基础的光照模型主要有兰伯特（Lambert）光照模型、冯氏（Phong）光照模型和改进的冯氏（Blinn Phong）光照模型。

Lambert 光照模型只包含漫反射光的计算，Phong 光照模型和 Blinn Phong 光照模型都包含环境光、漫反射光、镜面反射光的计算，两者的区别在与镜面反射光的计算，Phong 光照模型根据反向量和观察向量计算镜面反射光，Blinn Phong 光照模型根据半向量和法向量计算镜面反射光。



Phong光照模型



Blinn Phong光照模型

```
vec3 ambientColor() { // 环境光
    vec3 ambient = materialParams.ambientColor * materialParams.albedo;
    return ambient;
}

vec3 diffuseColor(vec3 normal, vec3 lightDir) { // 漫反射光
    float factor = max(dot(normal, lightDir), 0.0);
    vec3 diffuse = factor * materialParams.lightColor * materialParams.albedo;
    return diffuse;
}

vec3 specularColor(vec3 normal, vec3 lightDir, vec3 viewDire) { // 镜面反射光
    vec3 halfVec = normalize(lightDir + viewDire); // 半向量
    float factor = pow(max(dot(normal, halfVec), 0.0), materialParams.gloss);
    vec3 specular = factor * materialParams.lightColor *
materialParams.specularStrength;
    return specular;
}

material.baseColor = vec4(ambient + diffuse + specular, 1);
```

L 光线方向，N 法线方向，V 视角方向，T切线方法，H 半角向量

```
// lambert Lighting
FinalColor=saturate(dot(N,L))

// 半兰伯特光照 解决Lambert公式在灰面太暗的问题 经验值
FinalColor = pow(dot(N,L)*0.5+0.5,2);
```

5、PBR材质

材质模拟Mesh反射光照的代码算法不同，算法不同，自然模拟光照的真实程度也不同。以物理参数为依据来编写表面材质，而不必依靠粗劣的修改与调整。

相比传统的Lambert着色和Phong着色，PBR着色在效果上有着质的提升，可以表示更多更复杂的材质特征：

- 表面细节
- 物体粗糙度
- 区别明显的金属和绝缘体
- 物体的浑浊程度
- 菲涅尔现象：不同角度有不同强度的反射光
- 半透明物体
- 多层混合材质
- 清漆效果
- 其它更复杂的表面特征

PBR的主要参数

- 基础色 为材质提供基础纹理色，是Vector3（RGB），它们的值都限定在0~1之间。

材质(Material)	基础色强度(BaseColor Intensity)
木炭(Charcoal)	0.02
新沥青(Fresh asphalt)	0.02
旧沥青(Worn asphalt)	0.08
土壤(Bare soil)	0.13
绿草(Green Grass)	0.21
沙漠沙(desert sand)	0.36
新混泥土(Fresh concrete)	0.51
海洋冰(Ocean Ice)	0.56
鲜雪(Fresh snow)	0.81

- **粗糙度（Roughness）**：表示材质表面的粗糙程度，值限定在0~1之间。越粗糙材质高光反射越不明显，金属和非金属的粗糙度有所区别。
- **金属度（Metallic）**：表示材质像金属的程度，0是电介质（绝缘体），1是金属。金属没有漫反射，只有镜面反射。
- **镜面度（Specular）**：表示材质的镜面反射强度，从0（完全无镜面反射）~1（完全镜面反射。UE4的默认值是0.5。万物皆有光泽（镜面反射），对于强漫反射的材质，可通过调节粗糙度，而不应该将镜面度调成0。下面是镜面度的参数值

材质(Material)	镜面度(Specular)
草(Glass)	0.500
塑料(Plastic)	0.500
石英(Quartz)	0.570
冰(Ice)	0.224
水(Water)	0.255
牛奶(Milk)	0.277
皮肤(Skin)	0.350

判断一种PBR光照模型是否是基于物理的，必须满足以下三个条件

1. 基于微平面(Microfacet)的表面模型。

   微观上所有材质表面都是由很多朝向不一的微小平面组成，有的材质表面光滑一些，有的粗糙一些。基于一个平面的粗糙度来计算出某个向量的方向与微平面平均取向方向一致的概率。这个向量便是位于光线向量ll和视线向量vv之间的中间向量，被称为**半角向量(Halfway Vector)**。 *粗糙度从0.1~1.0的变化图。粗糙度越小，镜面反射越亮范围越小；粗糙度越大，镜面反射越弱。*

   ```
   // lightPos是光源位置，viewPos是摄像机位置，FragPos是像素位置
   vec3 lightDir   = normalize(lightPos - FragPos);
   vec3 viewDir    = normalize(viewPos - FragPos);
   vec3 halfwayDir = normalize(lightDir + viewDir);
   ```

2. 能量守恒。

   PBR会简化折射光，将平面上所有折射光都视为被完全吸收而不会散开。而有一些被称为次表面散射(Subsurface Scattering)技术的着色器技术会计算折射光散开后的模拟，它们可以显著提升一些材质（如皮肤、大理石或蜡质）的视觉效果，不过性能也会随着下降。

   金属(Metallic)材质会立即吸收所有折射光，故而金属只有镜面反射，而没有折射光引起的漫反射。

3. 应用基于物理的BRDF。

- **反射率**（Albedo）：反射率纹理指定了材质表面每个像素的颜色，如果材质是金属那纹理包含的就是基础反射率。这个跟我们之前用过的漫反射纹理非常的类似，但是不包含任何光照信息。漫反射纹理通常会有轻微的阴影和较暗的裂缝，这些在Albedo贴图里面都不应该出现，仅仅只包含材质的颜色（金属材质是基础反射率）。
- **法线**（Normal）：法线纹理跟我们之前使用的是完全一样的。法线贴图可以逐像素指定表面法线，让平坦的表面也能渲染出凹凸不平的视觉效果。
- **金属度**（Metallic）：金属度贴图逐像素的指定表面是金属还是电介质。根据PBR引擎各自的设定，金属程度即可以是[0.0，1.0]区间的浮点值也可以是非0即1的布尔值。
- **粗糙度**（Roughness）：粗糙度贴图逐像素的指定了表面有多粗糙，粗糙度的值影响了材质表面的微平面的平均朝向，粗糙的表面上反射效果更大更模糊，光滑的表面更亮更清晰。有些PBR引擎用光滑度贴图替代粗糙度贴图，因为他们觉得光滑度贴图更直观，将采样出来的光滑度使用（1-光滑度）= 粗糙度 就能转换成粗糙度了。

- **环境光遮挡**（Ambient Occlusion，AO）：AO贴图为材质表面和几何体周边可能的位置，提供了额外的阴影效果。比如有一面砖墙，在两块砖之间的缝隙里Albedo贴图包含的应该是没有阴影的颜色信息，而让AO贴图来指定这一块需要更暗一些，这个地方光线更难照射到。AO贴图在光照计算的最后一步使用可以显著的提高渲染效果，模型或者材质的AO贴图一般是在建模阶段手动生成的。