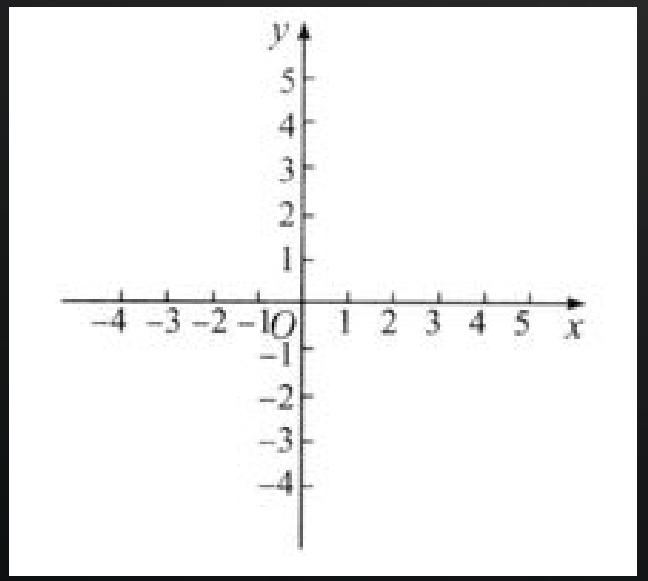
3D Technical Introduction

（一）3D数学基础

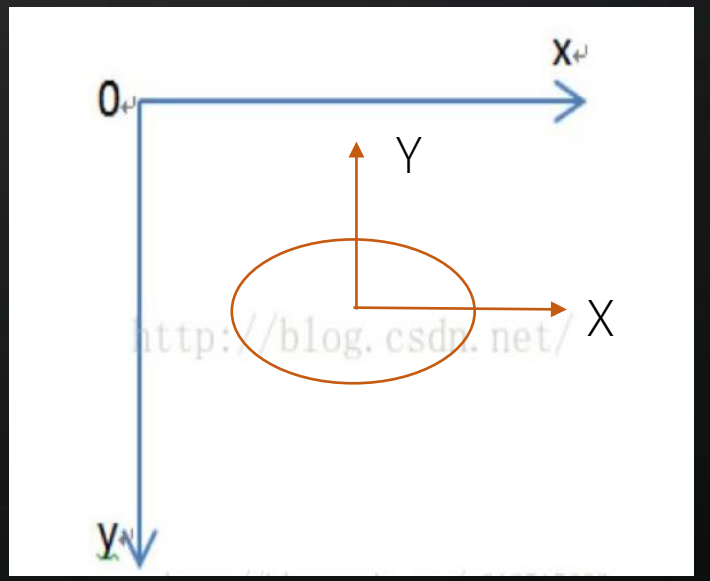
王丰 （Feng WANG）

坐标系

1. 2D数学坐标系，2D屏幕坐标系



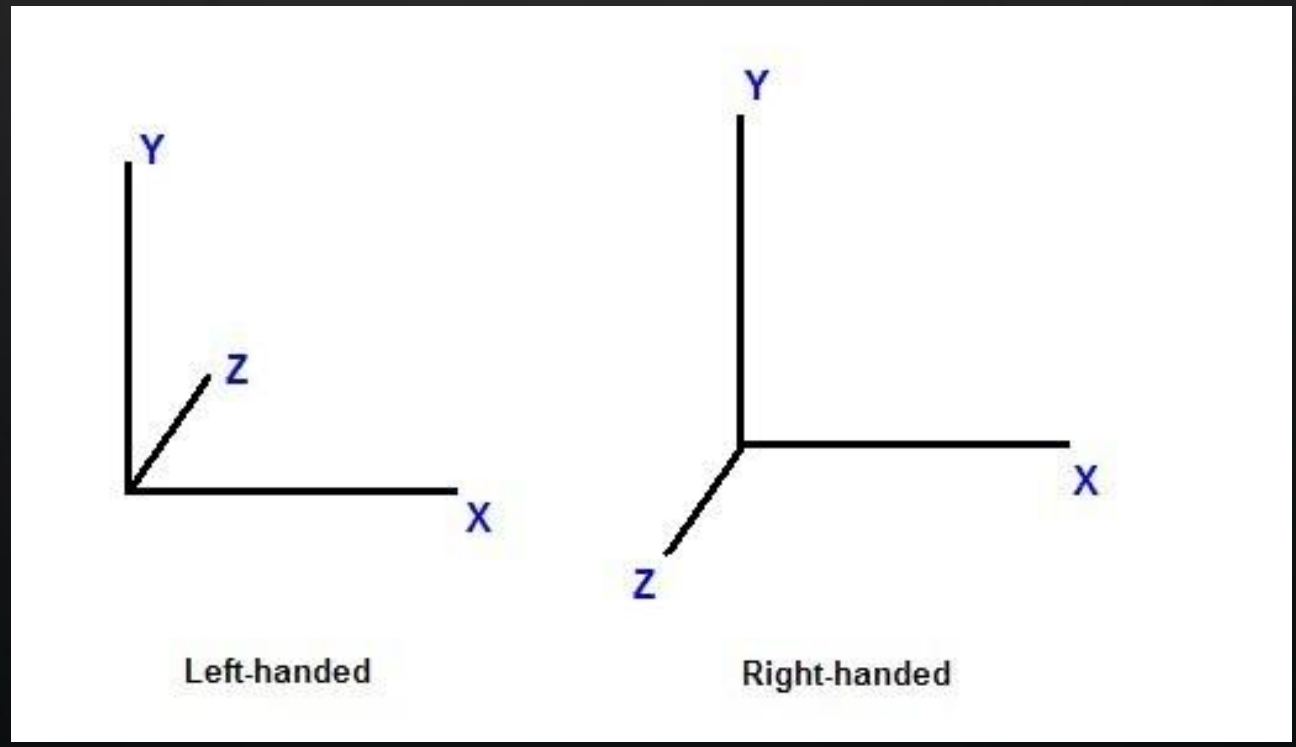
笛卡尔坐标系



2D UI 坐标系

坐标系

2. 3D数学坐标系，3D屏幕坐标系



Vector3

1. 基本概念

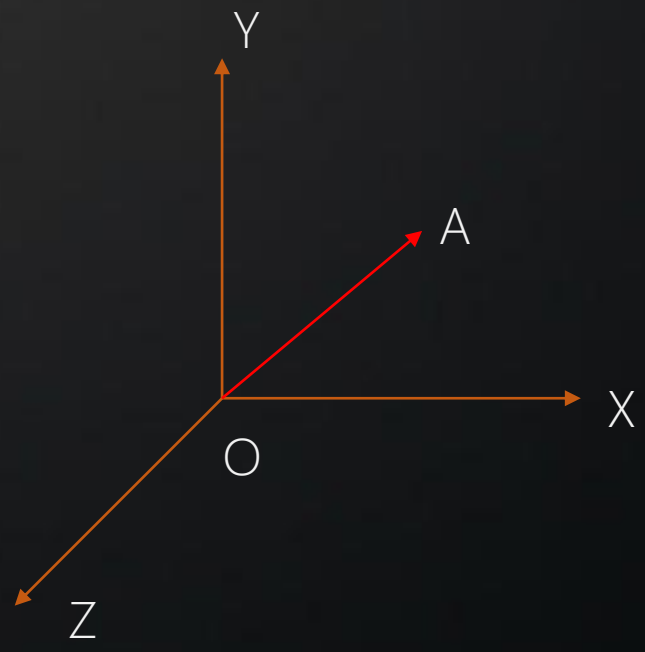
   a. 3D空间里一个点: A
   b. 3D空间里一个向量: $\overrightarrow{OA}$

2. 表示方法

   Vector3(x, y, z)

3. 在IDE中的意义

   a. 模型在空间里的位置、三个轴向上的缩放系数
   b. 光源的位置、照射方向
   c. Camera的位置、观察点、向上的方向



Vector3

4. 使用场景

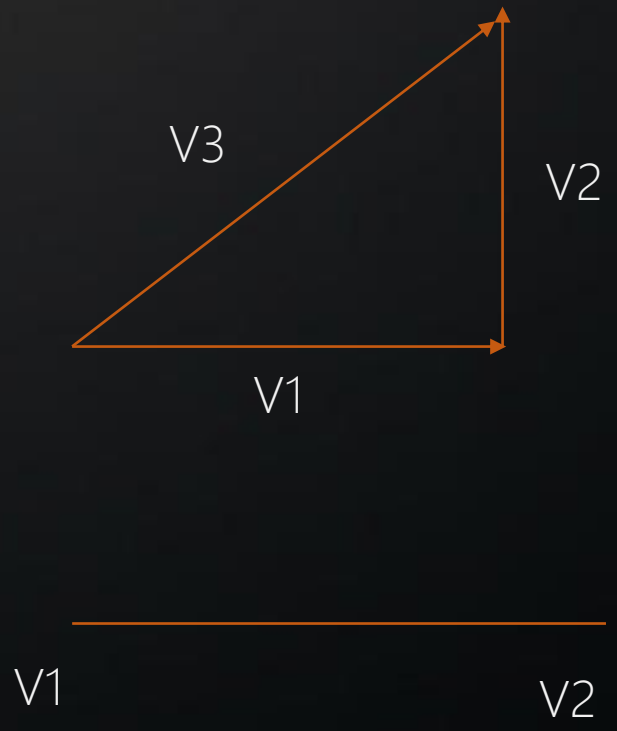
1）移动一个点：V1 + V2 = V3

2）V1到V3的方向：V3 – V1 = V2

3）一个线段的中点：（V1 + V2）/ 2

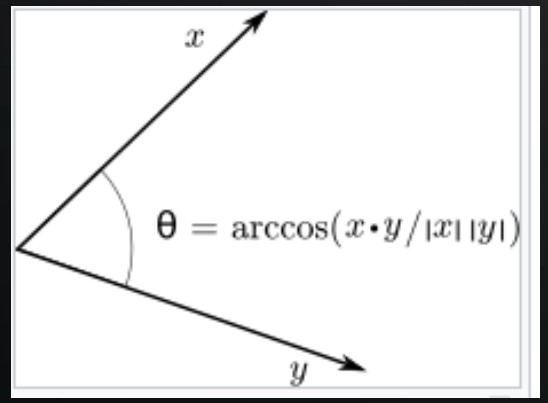
4）单位向量：Vector3.normalize()，得到长度是1的向量

5）指定方向和长度的向量：Vector3.normalize() ×length



4. 使用场景

6）Vector3 dot product



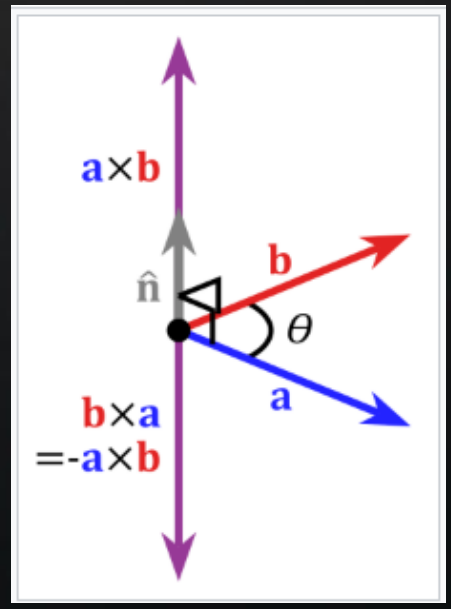
点乘得到的是一个值

$$\vec{a} \cdot \vec{b} = |\vec{a}|\,|\vec{b}|\cos\theta$$

求两个向量的夹角

$$\cos\theta = \frac{\mathbf{a}\cdot\mathbf{b}}{|\vec{a}|\,|\vec{b}|}$$

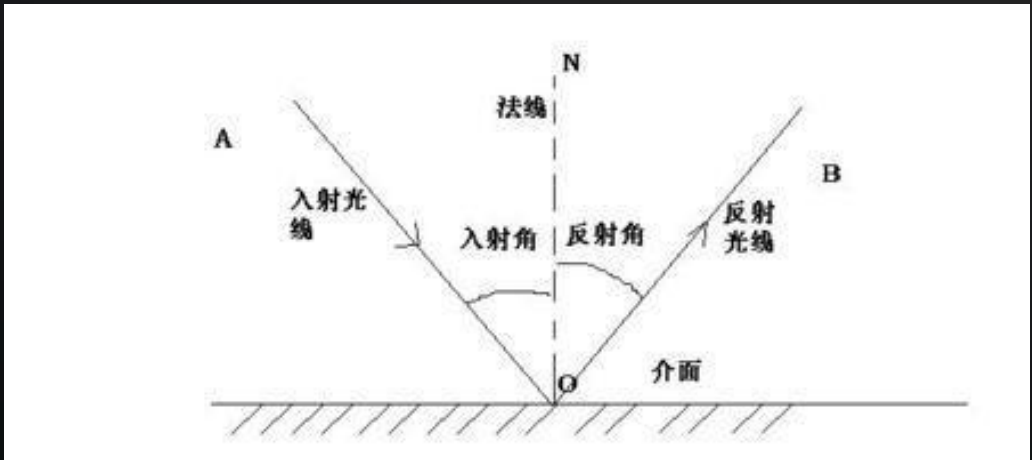
7）Vector3 cross product



叉乘得到是一个向量，垂直于两个向量组成的平面

4. 使用场景

8）Reflection



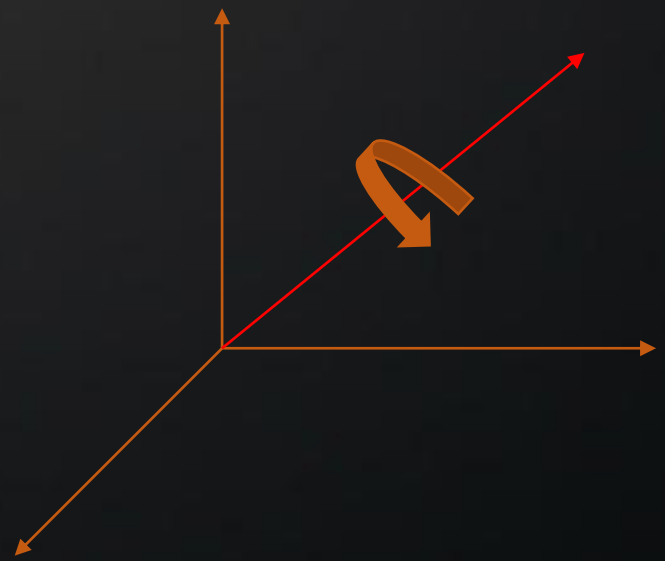
B = A - 2 * A dotProduct N * N

(推导): https://www.cnblogs.com/graphics/archive/2013/02/21/2920627.html

Quaternion

1. 基本概念

指定空间中的一个轴，绕这个轴做一个角度的旋转。



2. 表示方法

Quaternion(x, y, z，w)

绕向量n(x, y, z)，旋转θ

3. 在IDE中的意义

a. 模型在空间里的旋转信息，不过一般IDE上输入时，分别指定绕x，y，z轴的旋转角度。
b. 模型的朝向
c. Camera的朝向

4. 使用场景

1）构建方式：

旋转轴，旋转角度：Quaternion(const Vector3& rAxis, float rAngle)

轴旋转分量：Quaternion(const Vector3& xAxis, const Vector3& yAxis, const Vector3& zAxis)
Quaternion(float Yaw(Y), float Pitch(X), float Roll(Z))

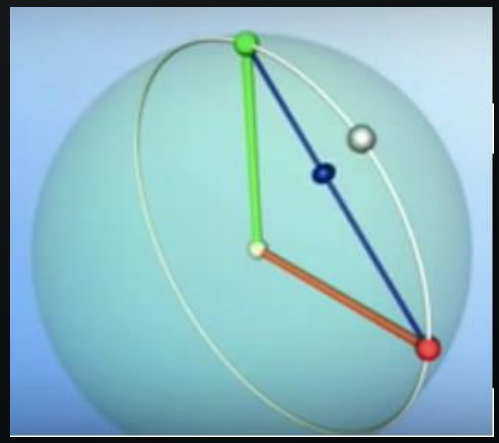
旋转开始向量，旋转结束向量：Quaternion(const Vector3 &rotateFrom, const Vector3& rotateTo)

2）把向量V1旋转到向量V2：
V2 = V1 * Quat1

3）连续旋转：
Quat3 = Quat1 * Quat2

4）连续插值：

Verctor3：
lerp：线性插值
slerp：球面插值



Quat = slerp(float fT, const Quaternion& rkP, const Quaternion& rkQ, bool shortestPath = false);

Matrix

1. 基本概念

矩阵表示模型的移动、缩放、旋转；camera的投影（透视、正交）、view等各种变换

2. 表示方法

```
float m00, float m01, float m02, float m03,
float m10, float m11, float m12, float m13,
float m20, float m21, float m22, float m23,
float m30, float m31, float m32, float m33
```

3. 在IDE中的意义

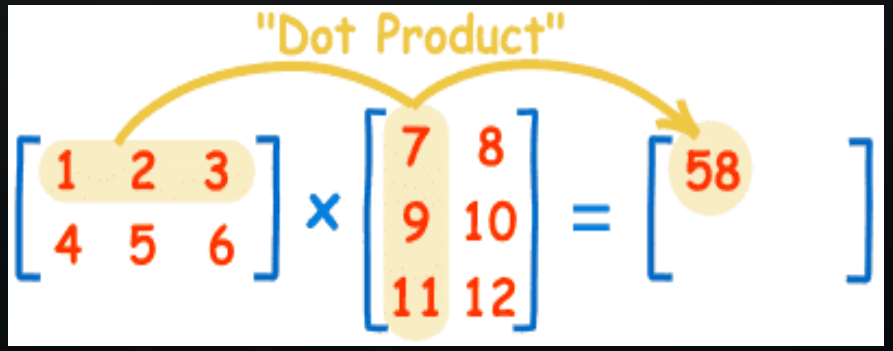
一般不会在IDE中直接指定Matrix4

4. 使用场景

1）矩阵乘法

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,j} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,j} \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{n} a_{1,k} b_{k,1} & \sum_{k=1}^{n} a_{1,k} b_{k,2} & \cdots & \sum_{k=1}^{n} a_{1,k} b_{k,j} \\ \sum_{k=1}^{n} a_{2,k} b_{k,1} & \sum_{k=1}^{n} a_{2,k} b_{k,2} & \cdots & \sum_{k=1}^{n} a_{2,k} b_{k,j} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} a_{m,k} b_{k,1} & \sum_{k=1}^{n} a_{m,k} b_{k,2} & \cdots & \sum_{k=1}^{n} a_{m,k} b_{k,j} \end{bmatrix}$$

AB != BA， 矩阵不符合交换定律



4. 使用场景

2）单位矩阵

IDENTITY （I），代表不做变换

```
const Matrix4 Matrix4::IDENTITY(
    1, 0, 0, 0,
    0, 1, 0, 0,
    0, 0, 1, 0,
    0, 0, 0, 1 );
```

3）矩阵的逆

Inverse（I），代表相反的变换　AB = I，　B = A^{-1}

4）矩阵的转制

处理行矩阵和列矩阵

$$\begin{vmatrix} a & b \\ c & d \\ e & f \end{vmatrix}^{T} = \begin{vmatrix} a & c & e \\ b & d & f \end{vmatrix}$$

4. 使用场景

5）平移物体

$$\begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

平移矩阵（Translation matrices），是列矩阵的表示。
如果是行矩阵，则X、Y、Z在最下一行。

若想把向量(10, 10, 10, 1)沿X轴方向平移10个单位

$$\begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 10 \\ 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 1*10+0*10+0*10+10*1 \\ 0*10+1*10+0*10+0*1 \\ 0*10+0*10+1*10+0*1 \\ 0*10+0*10+0*10+1*1 \end{bmatrix} = \begin{bmatrix} 10+0+0+10 \\ 0+10+0+0 \\ 0+0+10+0 \\ 0+0+0+1 \end{bmatrix} = \begin{bmatrix} 20 \\ 10 \\ 10 \\ 1 \end{bmatrix}$$

列矩阵：右乘

行矩阵：左乘 [10, 10, 10, 1] * Matrix = [20, 10, 10, 1]

4. 使用场景

6）缩放物体

$$\begin{bmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

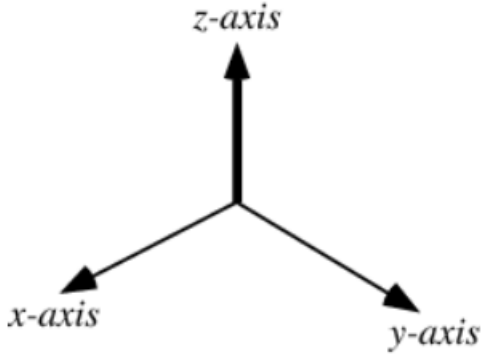
例如把一个向量（点或方向皆可）沿各方向放大2倍

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} 2*x+0*y+0*z+0*w \\ 0*x+2*y+0*z+0*w \\ 0*x+0*y+2*z+0*w \\ 0*x+0*y+0*z+1*w \end{bmatrix} = \begin{bmatrix} 2*x+0+0+0 \\ 0+2*y+0+0 \\ 0+0+2*z+0 \\ 0+0+0+1*w \end{bmatrix} = \begin{bmatrix} 2*x \\ 2*y \\ 2*z \\ w \end{bmatrix}$$

4. 使用场景

7）旋转物体

绕z轴进行旋转（在xy平面顺时针旋转）



$$\begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

不要用欧拉角，使用四元数

4. 使用场景

8）行列优先矩阵

列优先的位移矩阵

```
m[0][0] = 1.0f; m[0][1] = 0.0f; m[0][2] = 0.0f; m[0][3] = x;
m[1][0] = 0.0f; m[1][1] = 1.0f; m[1][2] = 0.0f; m[1][3] = y;
m[2][0] = 0.0f; m[2][1] = 0.0f; m[2][2] = 1.0f; m[2][3] = z;
m[3][0] = 0.0f; m[3][1] = 0.0f; m[3][2] = 0.0f; m[3][3] = 1.0f;
```

行优先的位移矩阵

```
m[0][0] = 1.0f; m[0][1] = 0.0f; m[0][2] = 0.0f; m[0][3] = 0.0f;
m[1][0] = 0.0f; m[1][1] = 1.0f; m[1][2] = 0.0f; m[1][3] = 0.0f;
m[2][0] = 0.0f; m[2][1] = 0.0f; m[2][2] = 1.0f; m[2][3] = 0.0f;
m[3][0] = x;    m[3][1] = y;    m[3][2] = z;    m[3][3] = 1.0f;
```

glsl shader中是列优先矩阵

列优先矩阵是右乘，
行优先矩阵是左乘

mvp = proj * view * model，列优先，右乘

mvp = model * view * proj，行优先，左乘

glUniformMatrix，传递的行优先矩阵

Color

1. 基本概念

光从物体反射到人的眼睛所引起的一种视觉心理感受。

2. 表示方法

Color(r, g, b, a)　　其中r，g，b，a，要么取值 [0, 255] 要么取值 [0.0f, 1.0f],
在GLES API和shader中一般用0到1之间的浮点数表示分量值。

3. 在IDE中的意义

a. 光源的颜色
b. 材质的自发光颜色
c. 材质对光的反射系数是另外的概念，但是一般也用颜色形式表示。
d. 有的IDE会直接指定物体表面的颜色，在没有光源的环境下，使物体仍然可以呈现出来。

THANK YOU