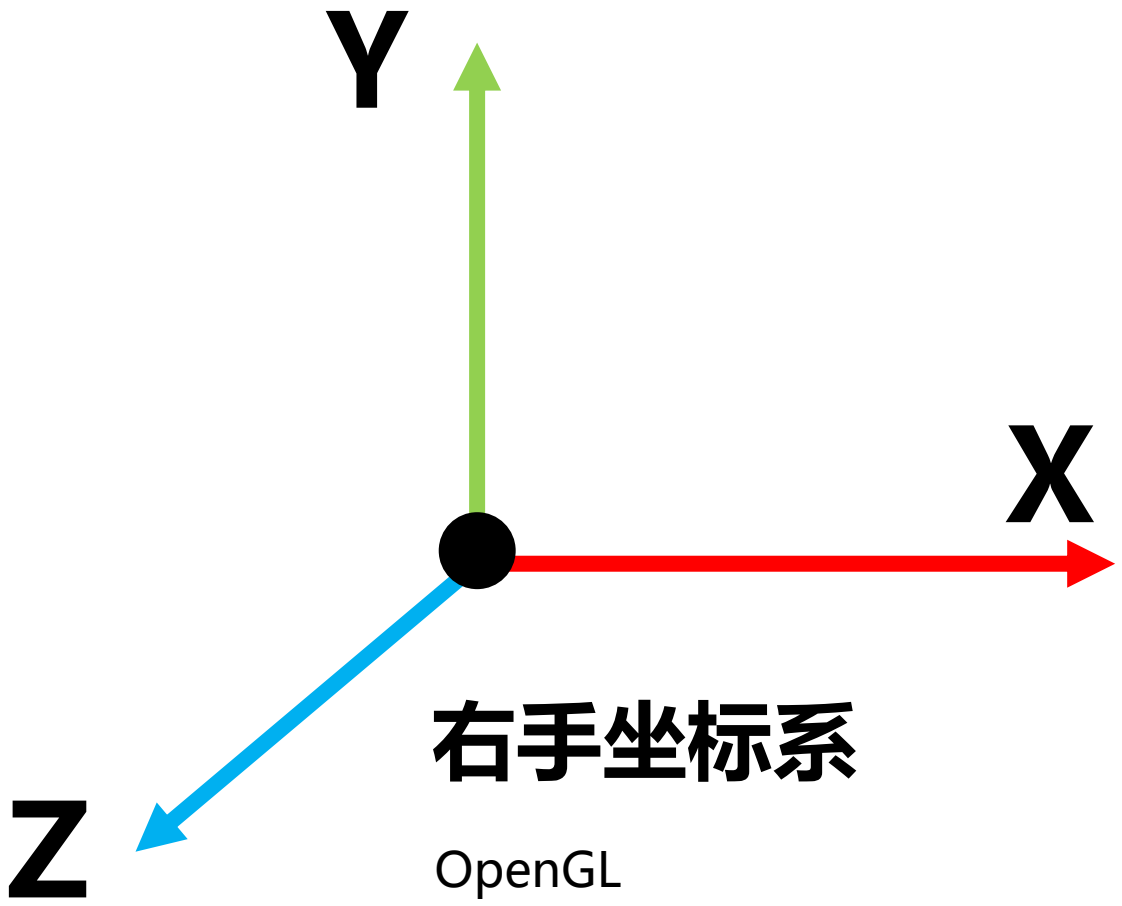
Chapter_3
坐标系，点，向量

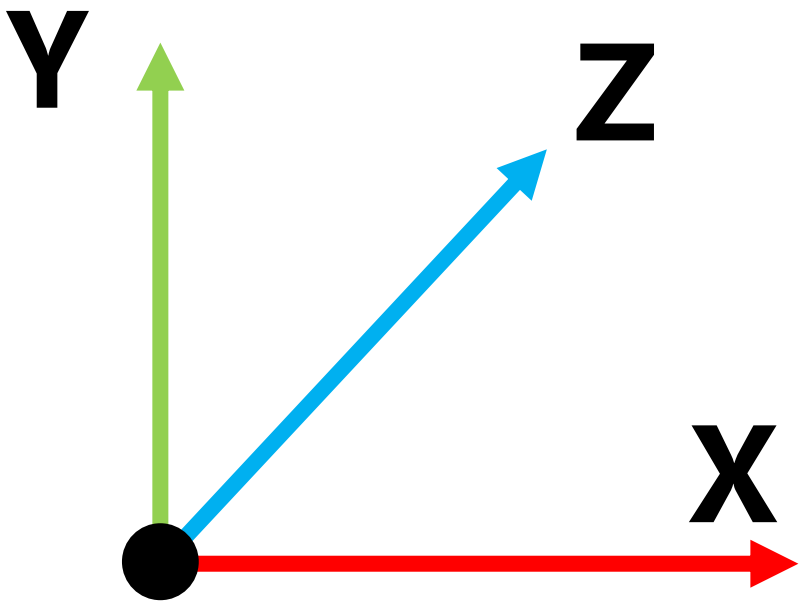
主讲人：王世元

三维笛卡尔坐标系



右手坐标系

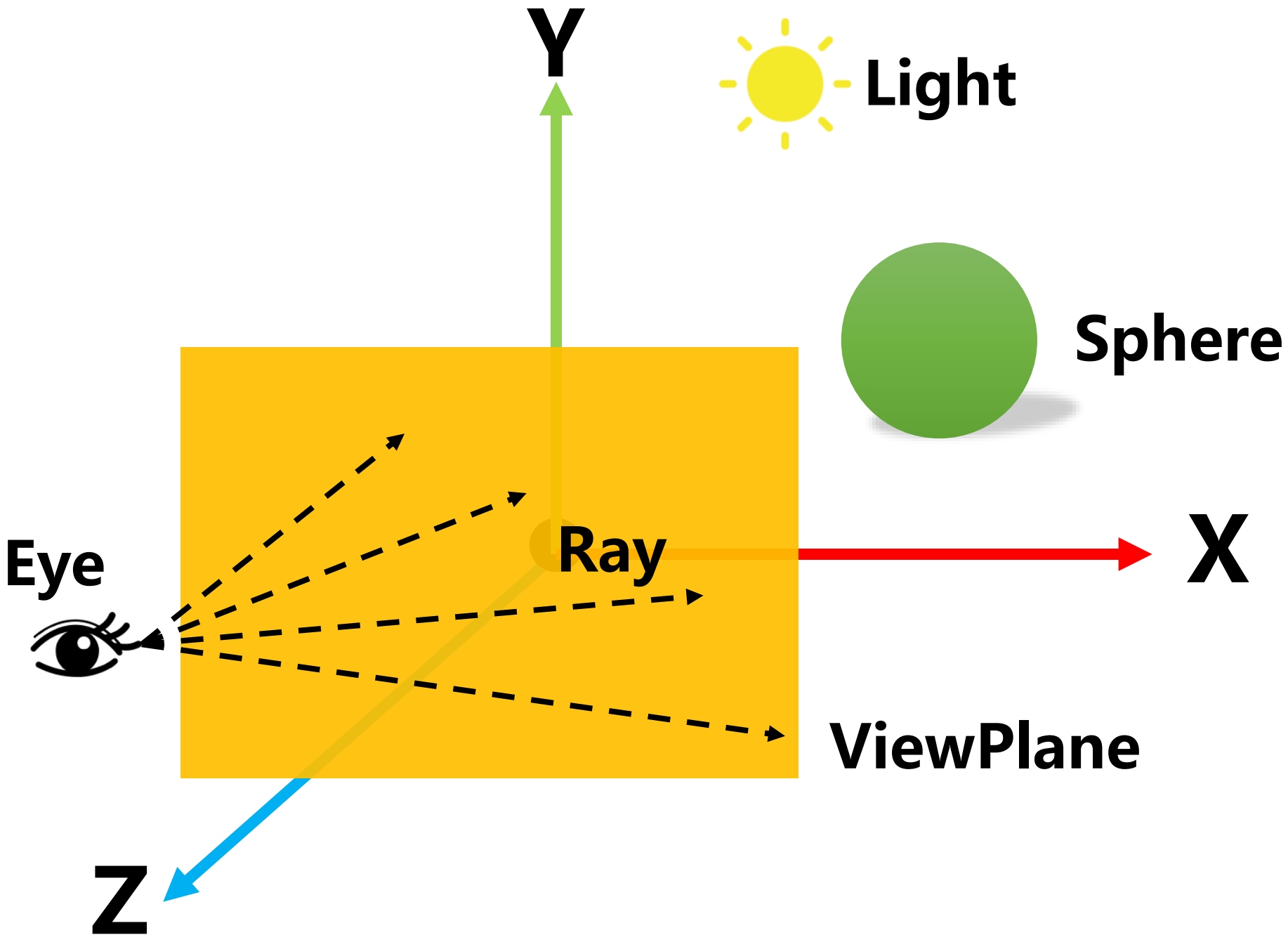
OpenGL
3D Max
Maya

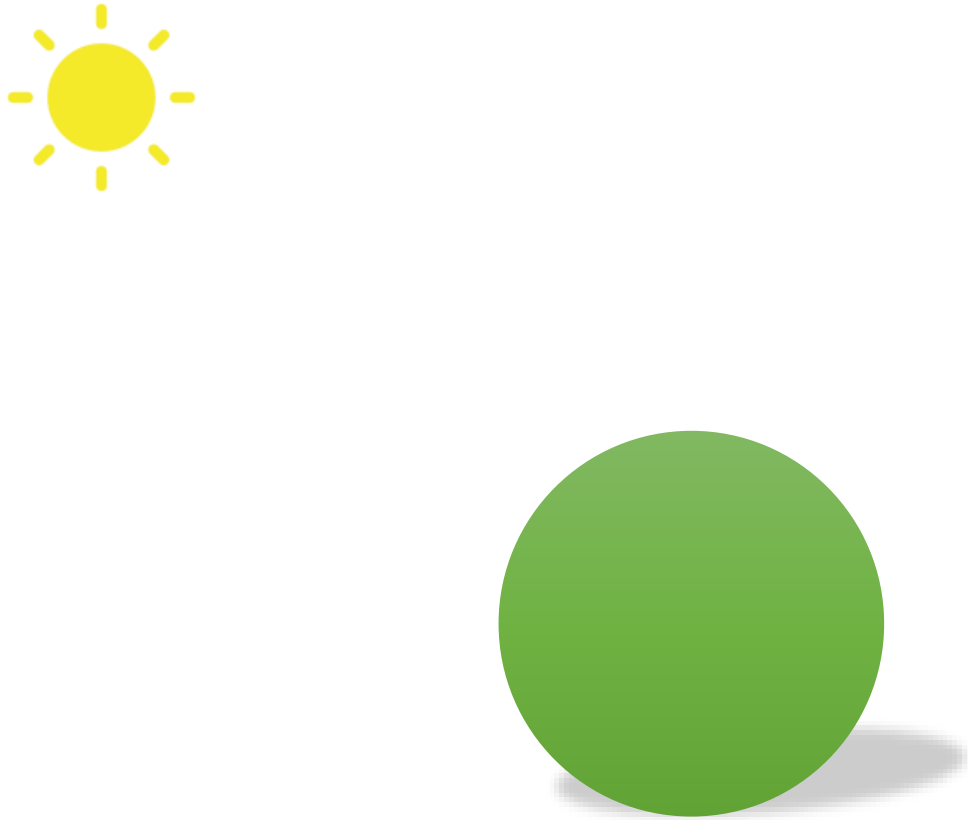


左手坐标系

DirectX
U3D
PBRT

三维世界中放置一切

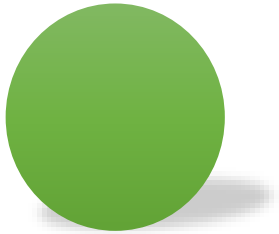




RT中的基本物体

Eye → 位置

Ray → 起点，方向



Sphere → 球心，半径

ViewPlane → 位置，大小，分辨率

Light → 位置，颜色/亮度

RT中的基本类

位置(Point)

起点(Point)， 方向(Vector)

球心(Point)， 半径(double)

位置(Point)，大小(double)，分辨率(int)

位置(Point)， 颜色/亮度(Color)

打地基

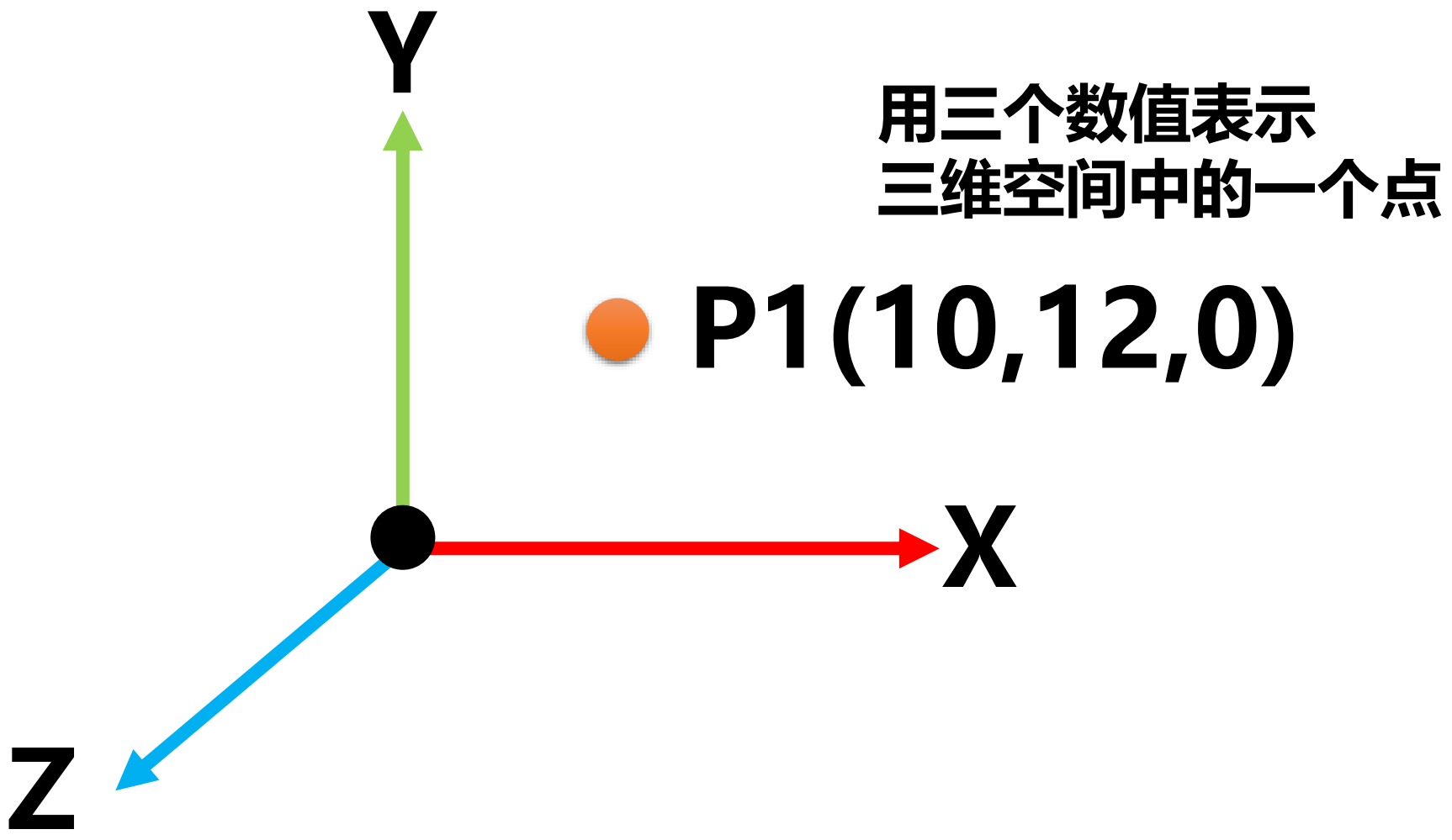
避免重复

点 Point → Point3D

向量 Vector → Vector3D

颜色 Color → SColor

点



用三个数值表示
三维空间中的一个点

Point3D类

```
class Point3D
{
    //xyz坐标
    private double _x;
    private double _y;
    private double _z;
```

```
//属性
2 个引用
public double X { get => _x; set => _x = value; }
2 个引用
public double Y { get => _y; set => _y = value; }
2 个引用
public double Z { get => _z; set => _z = value; }
```

```
//默认构造函数
0 个引用
public Point3D()
{
    this.X = 0;
    this.Y = 0;
    this.Z = 0;
}
```

```
//构造函数
0 个引用
public Point3D(double x, double y, double z)
{
    this.X = x;
    this.Y = y;
    this.Z = z;
}
```

Point3D类： float 还是 double

```
class Point3D
{
    //xyz坐标
    private float x;
    private float y;
    private float z;
```

```
class Point3D
{
    //xyz坐标
    private double x;
    private double y;
    private double z;
```

double（√）

float	double
精度已足够	C#处理方便
4 byte 节约空间	8 byte 空间耗损

float length = 4.3 F ；（烦躁）
double length = 4.3 ;

RT中可能需要存储几百万个点，内存吃紧

向量

在数学中，向量（也称为欧几里得向量、几何向量、矢量），指具有大小（magnitude）和方向的量。它可以形象化地表示为带箭头的线段。箭头所指：代表向量的方向；线段长度：代表向量的大小。与向量对应的量叫做数量（物理学中称标量），数量（或标量）只有大小，没有方向。

在空间直角坐标系中，分别取与x轴、y轴，z轴方向相同的3个单位向量*i*，*j*，*k*作为一组基底。若为该坐标系内的任意向量，以坐标原点O为起点作向量a。由空间基本定理知，有且只有一组实数(x,y,z)，使得a=ix+jy+kz，因此把实数对(x,y,z)叫做向量a的坐标，记作a=(x,y,z)。这就是向量a的坐标表示。其中(x,y,z)，就是点P的坐标。向量a称为点P的位置向量。

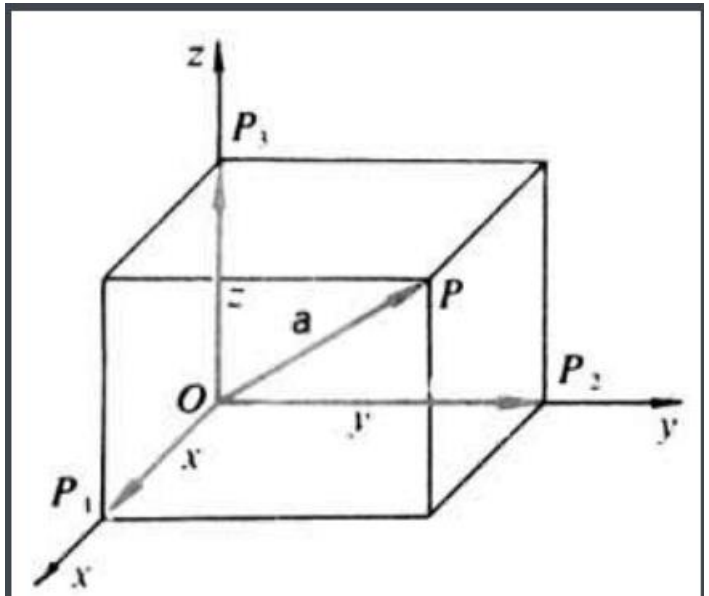
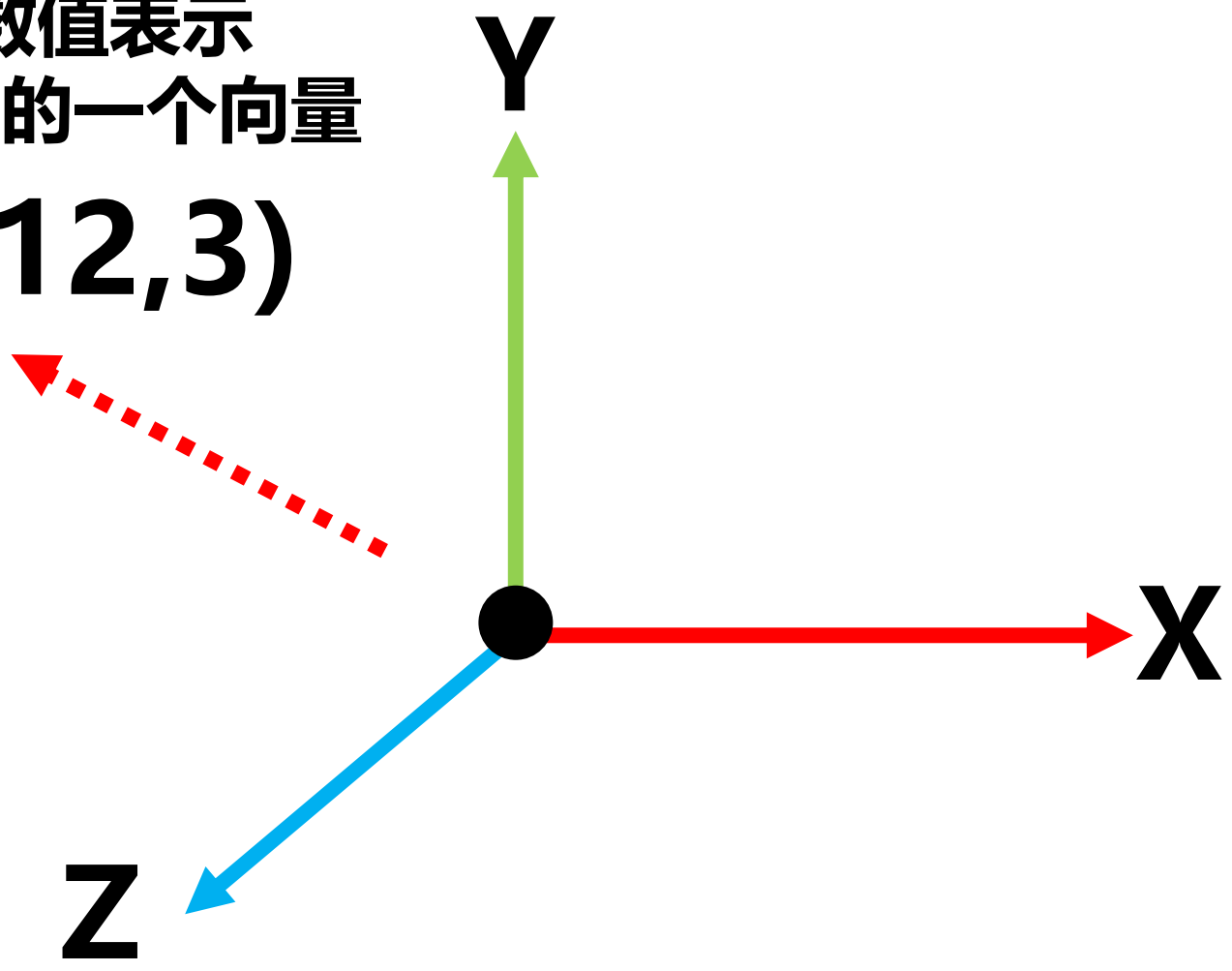


图 2 向量的坐标表示

向量

用三个数值表示
三维空间中的一个向量

V1(-3,12,3)



Vector3D类

```
class Vector3D
{
    //xyz坐标
    private double _x;
    private double _y;
    private double _z;
```

```
    //属性--Expression-Bodied Property Accessors
    2 个引用
    public double X { get => _x; set => _x = value; }
    2 个引用
    public double Y { get => _y; set => _y = value; }
    2 个引用
    public double Z { get => _z; set => _z = value; }
```

```
    //默认构造函数
    0 个引用
    public Vector3D()
    {
        this.X = 0;
        this.Y = 0;
        this.Z = 0;
    }
```

```
    //构造函数
    0 个引用
    public Vector3D(double x, double y, double z)
    {
        this.X = x;
        this.Y = y;
        this.Z = z;
    }
```

向量就这样了吗?

1) 求向量的大小（模）

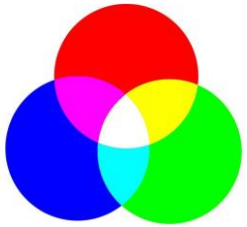
2) 向量的单位化

3) 向量与标量的加减乘除

4) 向量与向量的加减，点乘，叉乘

5) ……

三原色

	名称	R	G	B
C#:	Color	[0,255]	[0,255]	[0,255]
Me:	SColor	[0,1.0]	[0,1.0]	[0,1.0]

[0,255]-----用于显示

[0,1.0]-----用于计算

SColor类

```
class SColor
{
    private double _r;
    private double _g;
    private double _b;

    0 个引用
    public SColor()
    {
        this.R = 0.0;
        this.G = 0.0;
        this.B = 0.0;
    }

0 个引用
public SColor(double r, double g, double b)
{
    this.R = r;
    this.G = g;
    this.B = b;
}
```

```
2 个引用
public double R
{
    get
    {
        return _r;
    }

    set
    {
        _r = value;
        if (_r < 0.0)
        {
            _r = 0.0;
        }

        if (_r > 1.0)
        {
            _r = 1.0;
        }
    }
}
```

```
public double G
{
    get
    {
        return _g;
    }

    set
    {
        _g = value;

        if (_g < 0.0)
        {
            _g = 0.0;
        }

        if (_g > 1.0)
        {
            _g = 1.0;
        }
    }
}
```

SColor类

```
set
{
    _b = value;

    if (_b < 0.0)
    {
        _b = 0.0;
    }

    if (_b > 1.0)
    {
        _b = 1.0;
    }
}
```

让r g b的取值 限制在[0,1]之间。超过无法对应颜色，将会导致程序bug。