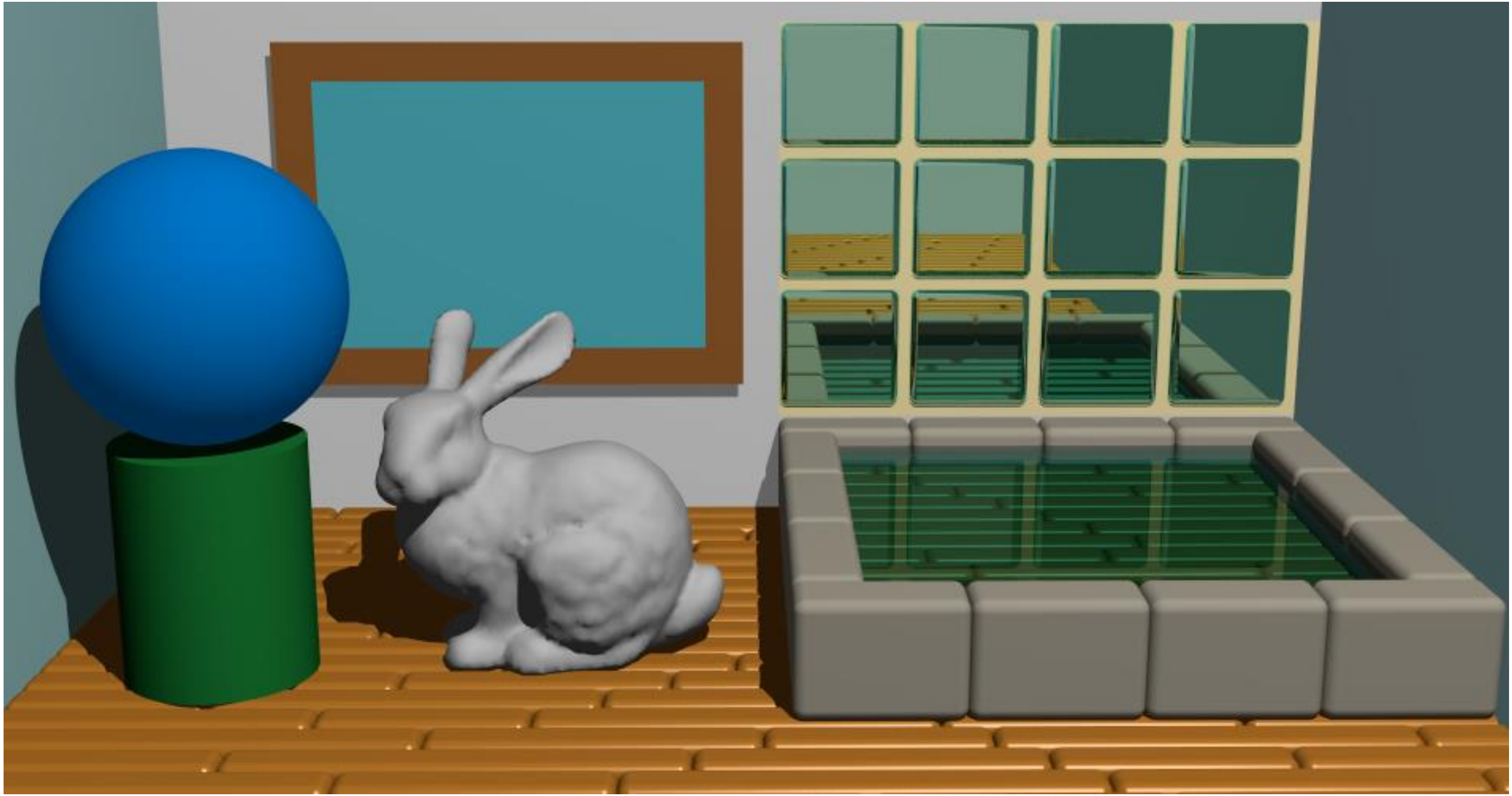
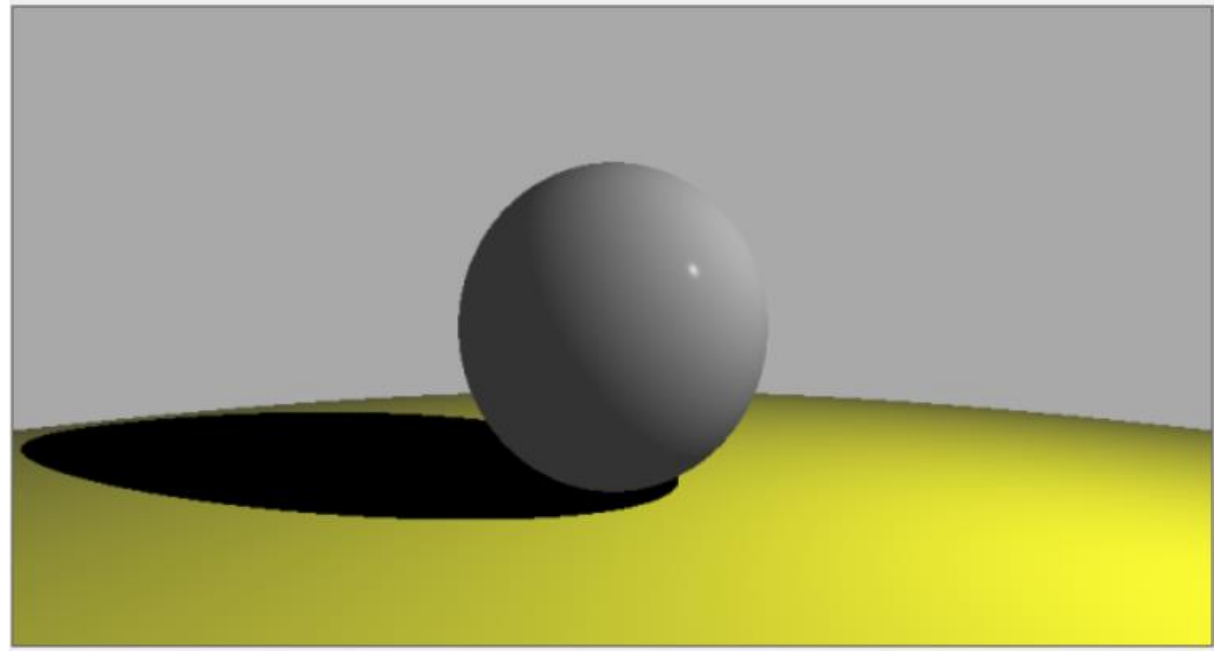
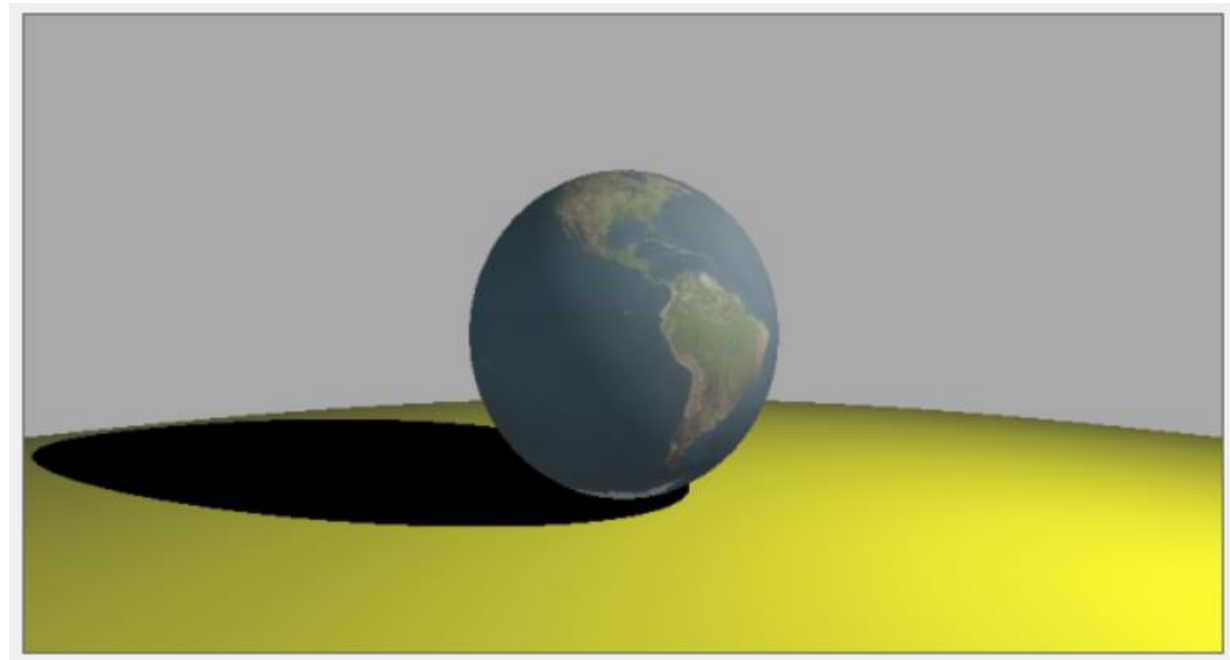
Chapter_12
纹理 / Texture

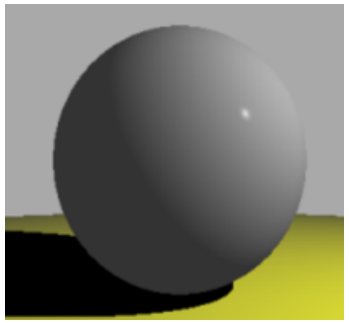
主讲人：王世元

西南石油大学 2020





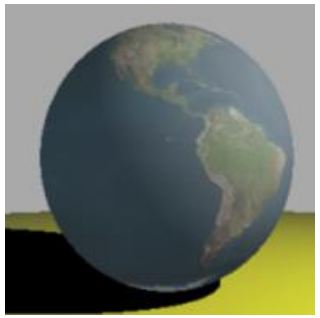


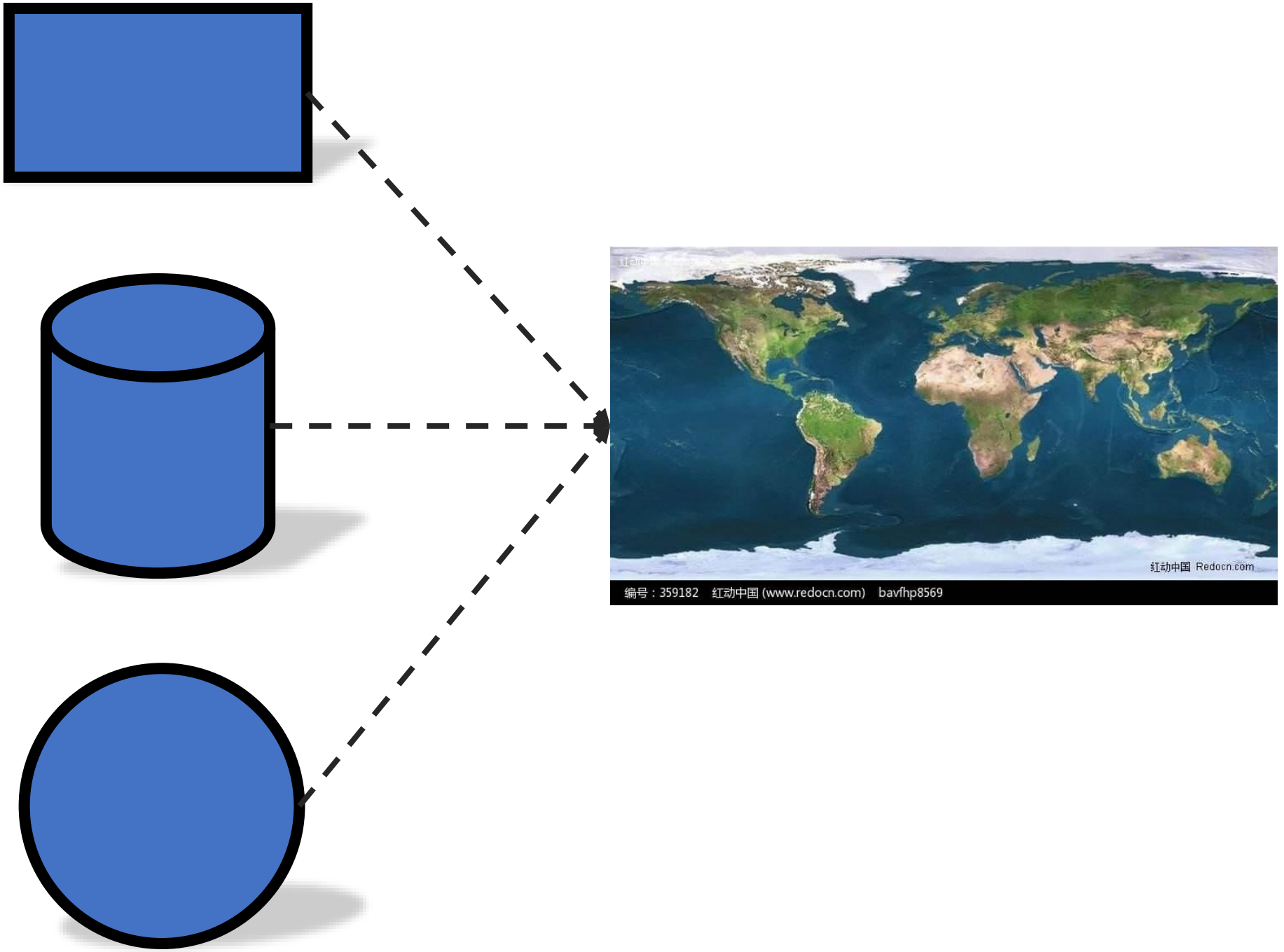


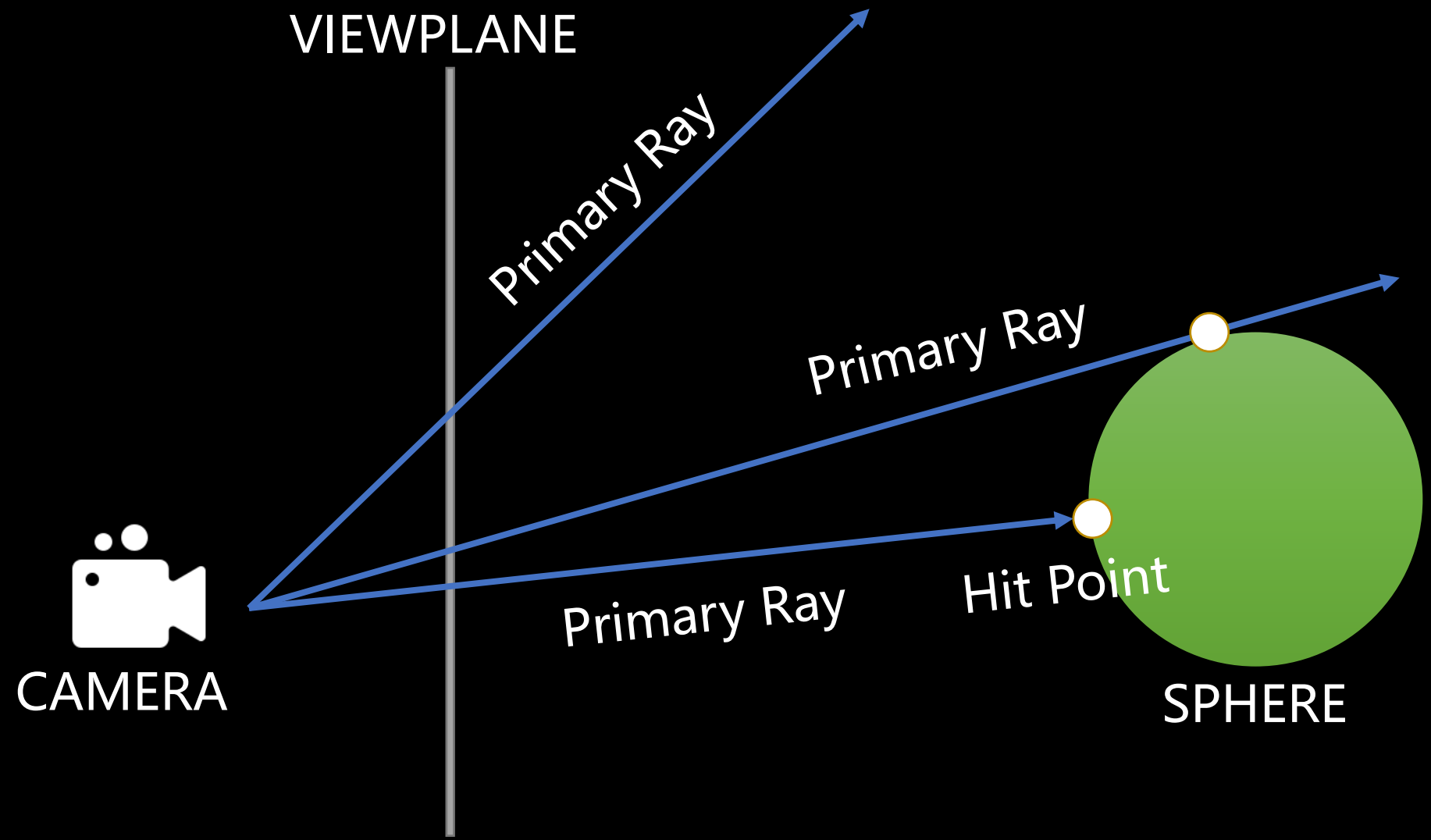
+

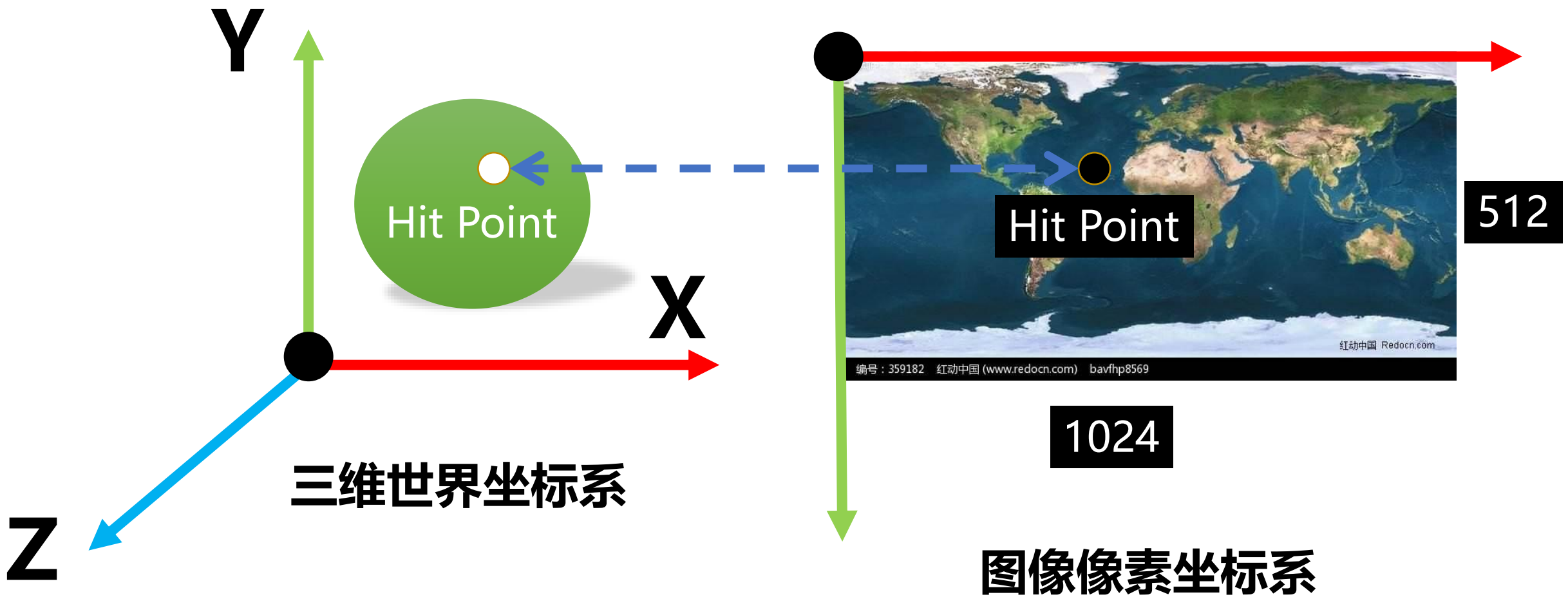

红动中国 Redocn.com
编号：359182 红动中国 (www.redocn.com) bavfhp8569

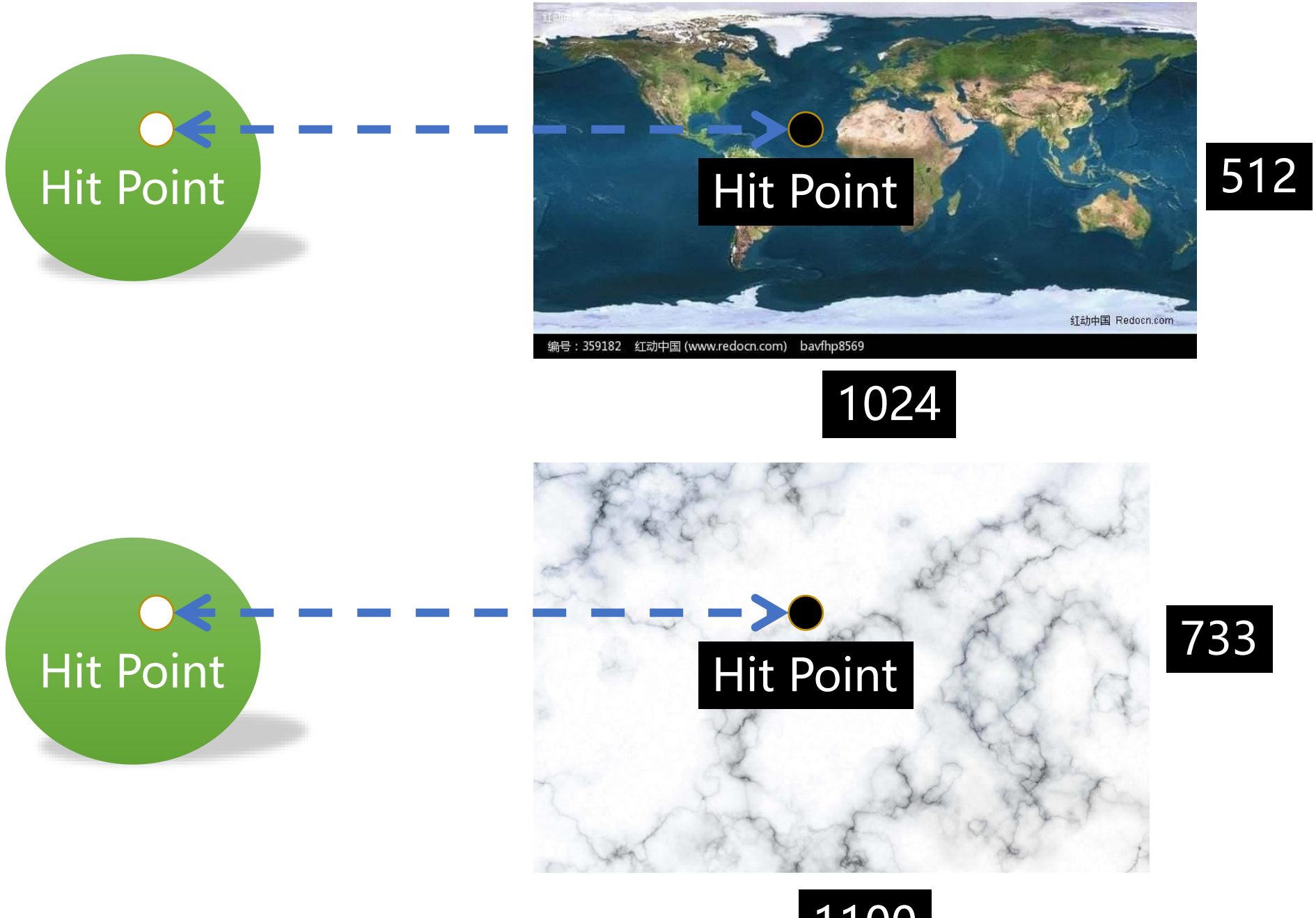
=




红动中国 Redocn.com
编号：359182 红动中国 (www.redocn.com) bavfhp8569


VIEWPLANE
Primary Ray
Primary Ray
Primary Ray
Hit Point
CAMERA
SPHERE

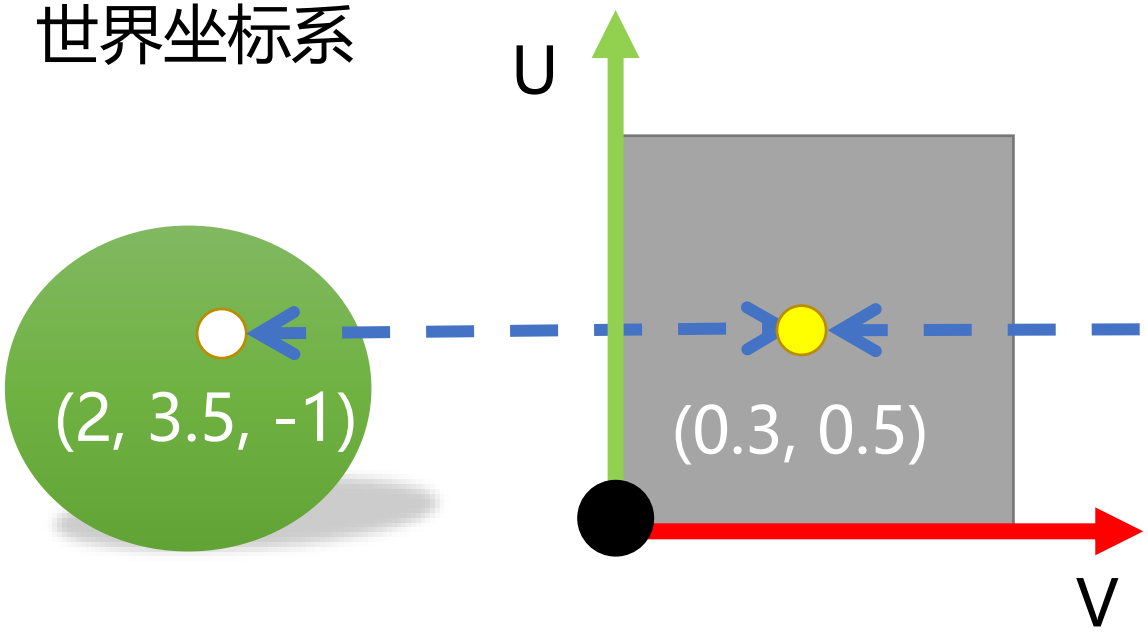

Y
X
Z
Hit Point
三维世界坐标系
Hit Point
512
1024
图像像素坐标系


Hit Point
Hit Point
512
红动中国 Redocn.com
编号：359182 红动中国 (www.redocn.com) bavfhp8569
1024
Hit Point
Hit Point
733
1100

UV 纹理坐标系

UV 坐标系[0,1]

世界坐标系

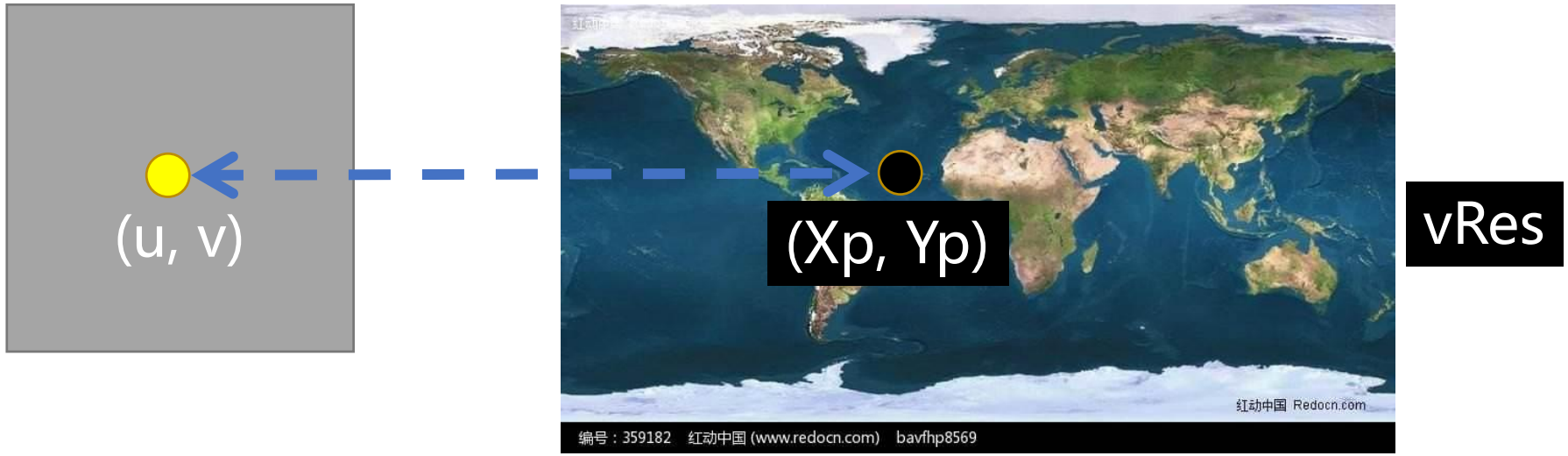


图像坐标系

UV 纹理坐标系

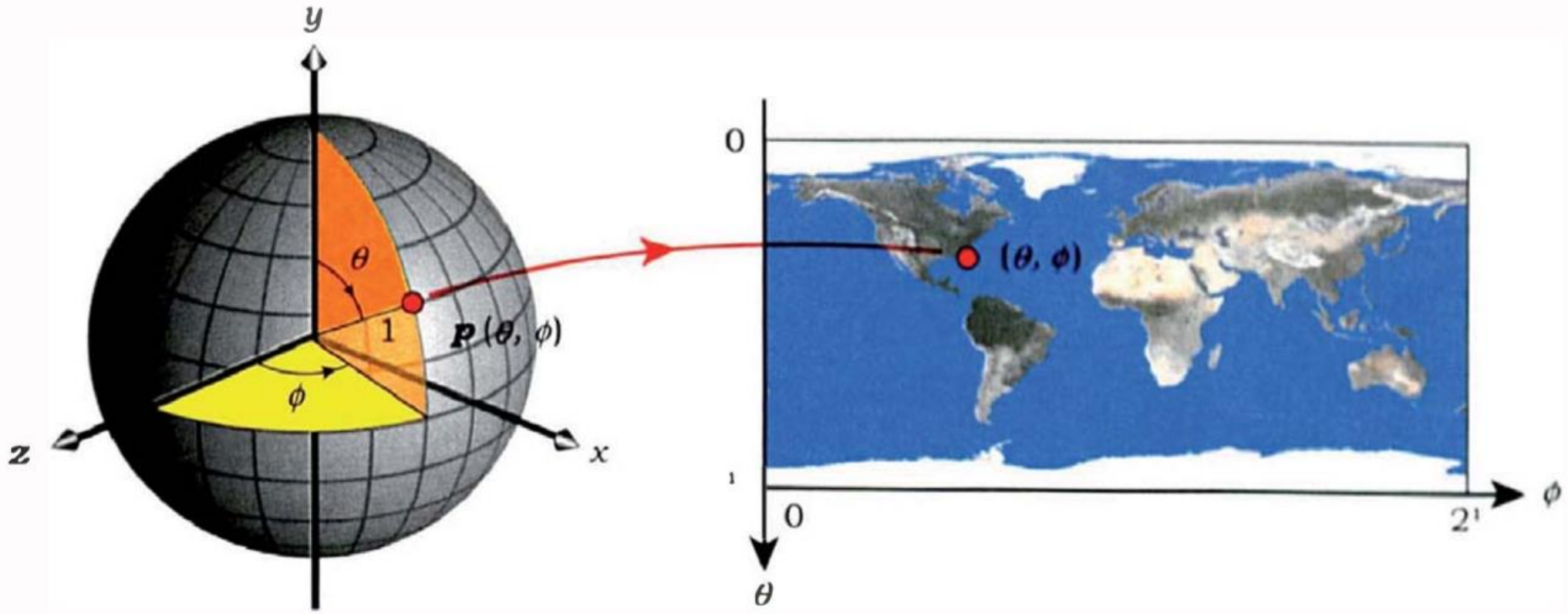
UV 坐标系(0,1)

图像坐标系



$$Xp = (hRes - 1) * u;$$

$$Yp = (vRes - 1) * v;$$

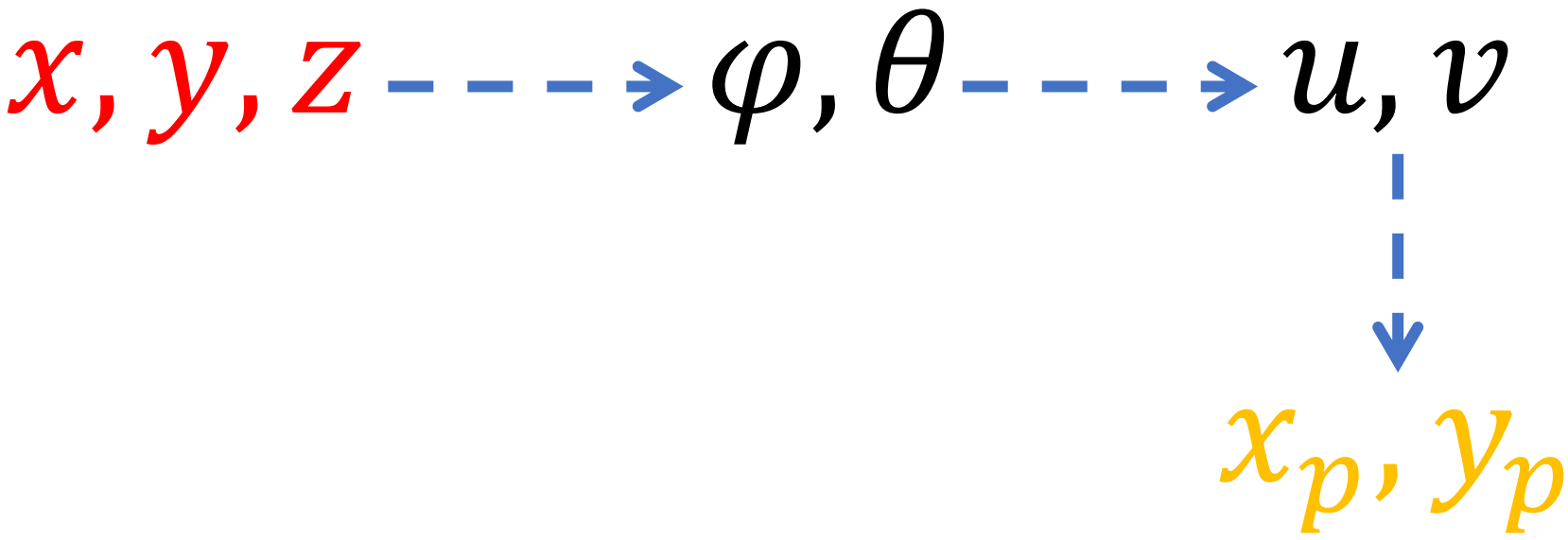


$$\varphi = tan^{-1}\left(\frac{x}{Z}\right); \quad \theta = cos^{-1}(y);$$

$$\varphi = tan^{-1}(\frac{x}{z}); \quad \theta = cos^{-1}(y);$$

$$u = \varphi/2\pi; \quad v = 1 - \theta/\pi;$$

UV 纹理坐标系



```
2 个引用
class Texture
{
    //用作纹理的图片
    Bitmap bmp = new Bitmap(100,100);

    //纹理的分辨率
    int hres = 100;
    int vres = 100;
```

```
    //设置用作纹理的图片
    2 个引用
    public Bitmap Bmp
    {
        get
        {
            return bmp;
        }

        set
        {
            bmp = value;

            hres = bmp.Width;
            vres = bmp.Height;
        }
    }
```

```
//依据击中点三维坐标，得到图片坐标
1 个引用
public void getTextCoordinate(Point3D pHit,
    out int row, out  int column)
{
    double theta = Math.Acos(pHit.Y);
    double phi = Math.Atan2(pHit.X, pHit.Z);

    if (phi < 0)
    {
        phi += 2.0 * Math.PI;
    }

    double u = phi / (2.0 * Math.PI);
    double v = 1.0 - theta / Math.PI;

    column = (int)((hres - 1) * u);
    row = (int)((vres - 1) * v);

}
```

```
//得到贴图上的具体的颜色值
1 个引用
public Color getColor(ShadeRec sr)
{
    int row;
    int column;


    getTextCoordinate(sr.HitPoint, out row, out column);

    return bmp.GetPixel(column, vres - row-1);
}
```

```
//纹理对象
Texture texture = new Texture();


//选择某张图片做纹理，并进行渲染
1 个引用
private void button1_Click(object sender, EventArgs e)
{
    OpenFileDialog fd = new OpenFileDialog();
    fd.ShowDialog();
    texture.Bmp = new Bitmap(fd.FileName);
    Render();
}
```

```
Light pointLight = new Light();
pointLight.LightColor = new SColor(1, 1, 1);
pointLight.Position = new Point3D(3,2, 2);


//小球
Sphere sphere = new Sphere(new Point3D(0, 0, 0), 1);//球体位置
sphere.Mat = new Material(0.2, 0.5, 0.3, 500,
    new SColor(1, 1, 1), true);
word.Add(sphere);

//底部大球
Sphere sphere3 = new Sphere(new Point3D(0, -101, -1), 100);//球体位置
sphere3.Mat = new Material(0.2, 0.8, 0.3, 50,
    new SColor(1,1, 0));
word.Add(sphere3);



Point3D eye = new Point3D(0, 0, 4); //观察点位置



//成像平面上的每个点的位置
Point3D p = new Point3D(-2 + step * i, 1 - step * j, 2);
```

让球体动起来

$$(x'\ y'\ z'\ 1)=(x\ y\ z\ 1)\begin{bmatrix}\cos\beta & 0 & -\sin\beta & 0\\ 0 & 1 & 0 & 0\\ \sin\beta & 0 & \cos\beta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

绕Y轴

```
//对球体进行旋转
double theta = Math.PI * trackBar1.Value / 180.0;
double x = sr.HitPoint.X * Math.Cos(theta) + sr.HitPoint.Z * Math.Sin(theta);
double z = -sr.HitPoint.X * Math.Sin(theta) + sr.HitPoint.Z * Math.Cos(theta);
```

```
//镜面反射指数
double ks = 0.3;
double ns = 100;
```

```
//求反射方向
Vector3D R = 2 * normal * (normal * rayLight) - rayLight;
R.Normalize();

primaryRay.Direction.Normalize();

SColor color = Ia * ka + Ip * kd * (rayLight * normal) +
   Ip * ks * (Math.Pow(R * primaryRay.Direction, ns));

bmp.SetPixel(i, j, color.ToRGB255Color());
```

光照 纹理 融合

```
//纹理颜色
Color tColor = texture.getColor(sr);

//光照颜色
Color lColor = color3.ToRGB255Color();

//融合权重
double t = 0.3;

//融合
Color result = Color.FromArgb(
    (int)(tColor.R * t + lColor.R * (1 - t)),
    (int)(tColor.G * t + lColor.G * (1 - t)),
    (int)(tColor.B * t + lColor.B * (1 - t))
    );

//设置颜色
bmp.SetPixel(i, j, result);
}
```