Lorem Ipsum is simply dummy text of the printing and typesetting industry

0

1 webgl常用坐标系汇总

webgl常用坐标系汇总

坐标系名称	原点	轴向	备注
屏幕坐标系	浏览器左上角	Y轴竖直向下； X轴水平向右	
Canvas坐标系	具体浏览器 position:relative； top:rect.top; left:rect:left;	Y轴竖直向下； X轴水平向右	
纹理（UV）坐标系	左下：（0,0） 左上：（0,1） 右上：（1,1） 右下：（1,0）	y轴向上为正 x轴向右为正	
图片坐标系		y轴向下为正 x轴向右为正	
局部坐标系	一般是模型重心	自定义	以物体某一点为原点而建立的坐标，该坐标系仅对该物体适用，用来简化对物体各部分坐标的描述。
世界坐标系	画布中心	y轴向上为正 x轴向右为正	取决于外部插件比如glMatrix 或者GLM等
相机坐标系	相机重心	y轴垂直向上 z轴朝向视点	
裁减（投影）坐标系	同世界坐标系		是执行矩阵变换和透视投影之后，但在执行透视除法之前的坐标。超出裁剪空间的坐标会被丢弃。
规范化设备坐标系 NDC	屏幕中心	y轴向上为正[-1,1] x轴向右为正[-1,1] ；z轴向里[near,far]	webgl唯一定义的NDC坐标系，默认是左手坐标系

Lorem Ipsum is simply dummy text of the printing and typesetting industry

02

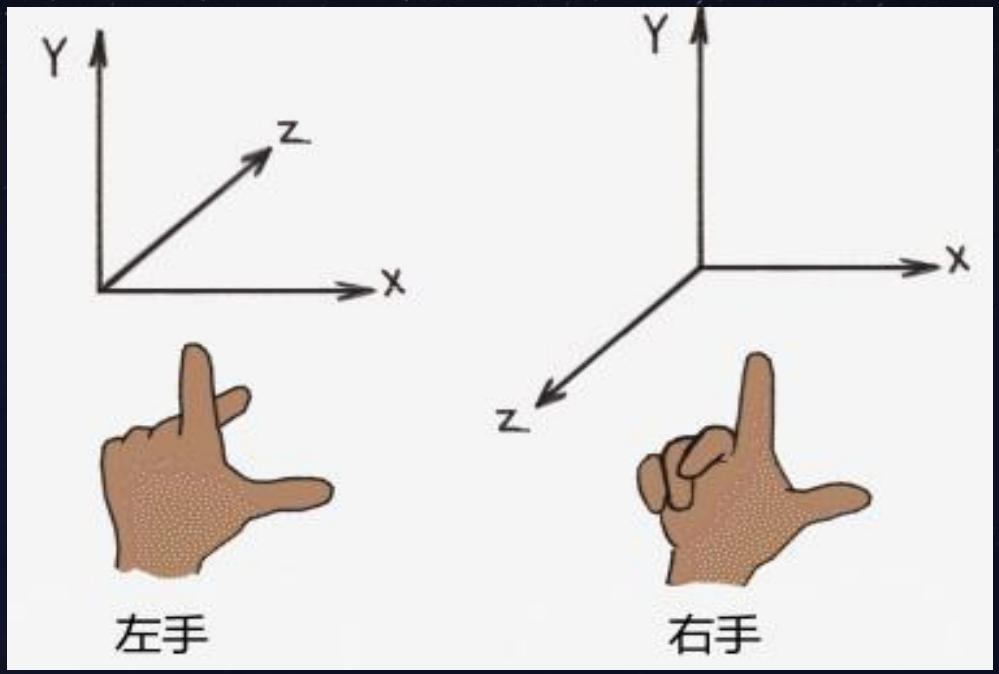
webgl是左手系还是右手系？

公式推导

什么是左手系？

什么又是右手系？

webgl是右手系或者左手系；
webgl世界坐标系、相机坐标系是右手，DNC是左手

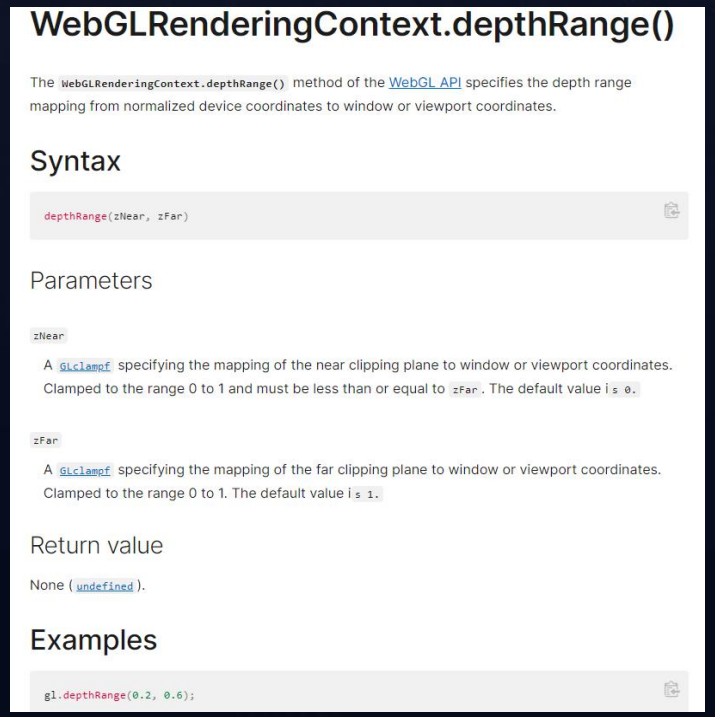


公式推导

webgl现在没有右手坐标系或者左手坐标系一说，并且在webgl API里面都没有世界坐标系一说，（是不是很玄幻，那为什么我们还来讲，因为方便大家去理解），众所周知webgl是opengl的阉割版，opengl之前确实因为设备或API限制确实是右手系，所以造成很多库现在也延续这种习惯，但是现在变得更加灵活。

外部提供的，需要看外部库使用的是什么坐标系

不管MVP是lh还是rh，最后转换成NDC坐标系即可



WebGLRenderingContext.depthRange()

The `WebGLRenderingContext.depthRange()` method of the WebGL API specifies the depth range mapping from normalized device coordinates to window or viewport coordinates.

Syntax

```
depthRange(zNear, zFar)
```

Parameters

`zNear`

A `GLclampf` specifying the mapping of the near clipping plane to window or viewport coordinates. Clamped to the range 0 to 1 and must be less than or equal to `zFar`. The default value is `0`.

`zFar`

A `GLclampf` specifying the mapping of the far clipping plane to window or viewport coordinates. Clamped to the range 0 to 1. The default value is `1`.

Return value

None (`undefined`).

Examples

```
gl.depthRange(0.2, 0.6);
```

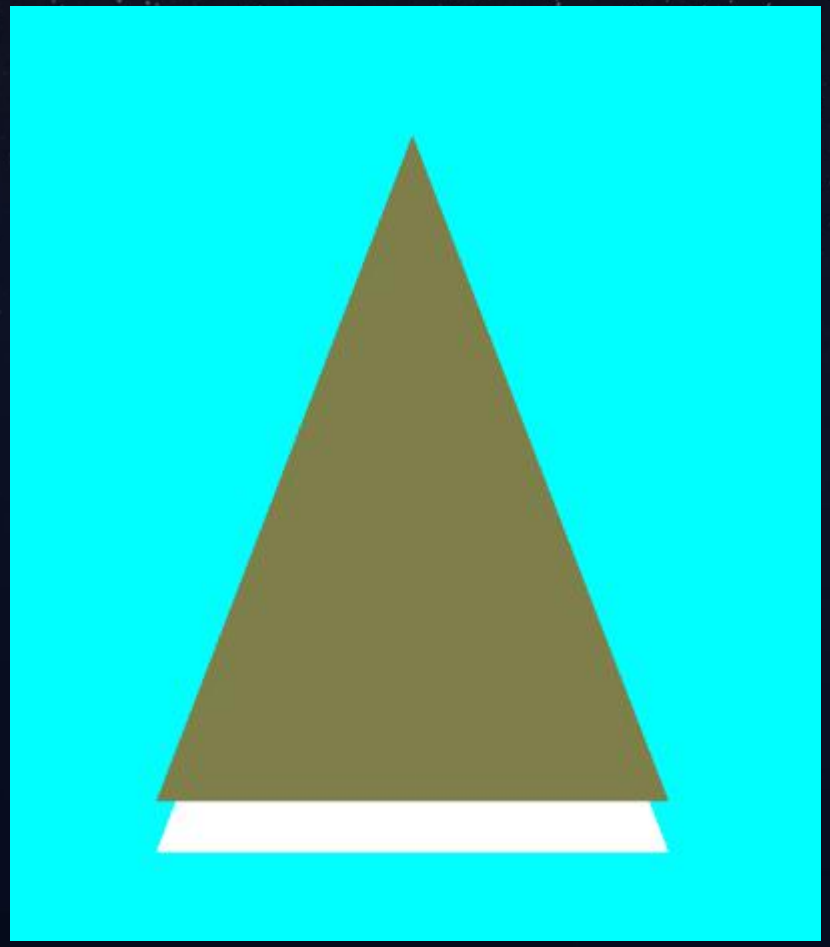
depthRange(zNear, zFar)

Lorem Ipsum is simply dummy text of the printing and typesetting industry

02

左右手系实例认证

实例



```
▶WebGL: INVALID_OPERATION: depthRange: zNear > zFar
Uncaught Error: http://127.0.0.1:5502/lib/gl-lint.js:3110: error in depthRange(0.9, 0.1): INVALID_OPERATION
    at reportError (gl-lint.js:2197:15)
    at reportFunctionError (gl-lint.js:3000:7)
    at ctx.<computed> [as depthRange] (gl-lint.js:3110:11)
    at Object.depthRange (gl-debug.js:208:38)
    at initBuffer (模型变换.html:85:16)
    at main (模型变换.html:36:13)
    at onload (模型变换.html:98:23)
```

公式推导

世界坐标=>屏幕坐标

局部坐标=>世界坐标=>观察坐标=>裁剪坐标=>标准设备坐标=>屏幕坐标

MVP v = screen

```
function worldToScreen(worldIn) {
    var world = new Float32Array(4);
    var screen = new Float32Array(4);
    var mvp = mat4.create();
    world[0] = worldIn[0];
    world[1] = worldIn[1];
    world[2] = worldIn[1];
    world[3] = 1.0;
    //! ????mvp????
    mat4.multiply(projectMat, viewMat, mvp);
    //! ??mvp * world
    mat4.multiplyVec4(mvp, world, screen);

    if (screen[3] == 0.0) {
        return false;
    }
    //! ?????? ??
    screen[0] /= screen[3];
    screen[1] /= screen[3];
    screen[2] /= screen[3];

    // map to range 0 - 1
    screen[0] = screen[0] * 0.5 + 0.5;
    screen[1] = screen[1] * 0.5 + 0.5;
    screen[2] = screen[2] * 0.5 + 0.5;

    // map to viewport
    screen[0] = screen[0] * viewPortW;
    screen[1] = viewPortH - (screen[1] * viewPortH);
    return screen;
}
```

公式推导

屏幕坐标=>世界坐标

局部坐标=>世界坐标=>观察坐标=>裁剪坐标=>标准设备坐标=>屏幕坐标

MVP v = screen

$v = MVP^{-1}screen$

```
function screenToWorld(screen) {
    debugger
    var v = new Float32Array(4);
    var world = new Float32Array(4);
    v[0] = screen[0];
    v[1] = screen[1];
    v[2] = screen[2];
    v[3] = 1.0;
    // map from viewport to 0 - 1
    v[0] = (v[0]) / viewPortW;
    v[1] = (viewPortH - v[1]) / viewPortH;
    //v[1] = (v[1] - _viewPort.Y) / _viewPort.Height;
    // map to range -1 to 1
    v[0] = v[0] * 2.0 - 1.0;
    v[1] = v[1] * 2.0 - 1.0;
    v[2] = v[2] * 2.0 - 1.0;


    var mvp = mat4.create();
    var mvpInvert = mat4.create();
    //! ����mvp����
    mat4.multiply(projectMat, viewMat, mvp);
    inverseEx(mvp, mvpInvert);
    world[0] = v[0];
    world[1] = v[1];
    world[2] = v[2];
    world[3] = v[3];
    multiply(mvpInvert, v, world);
    if (world[3] == 0.0) {
        return world;
    }
    world[0] /= world[3];
    world[1] /= world[3];
    world[2] /= world[3];
    console.log(world);
    return world;
}
```