3D体素实现画线算法

一．Cube绘制

我使用了A-Frame，A-Frame 是一个用来构建虚拟现实（VR）应用的网页开发框架。由WebVR的发起人[Mozilla VR](https://mozvr.com) 团队所开发，是当下用来开发WebVR内容最强大最易用的技术方案。A-Frame基于HTML，容易上手。但是A-Frame不仅仅是一个3D场景渲染引擎或者一个标记语言。其核心思想是基于Three.js来提供一个声明式、可扩展以及组件化的编程结构。可以使用js自由地添加元素。

<a-scene>

<a-camera position="0 1.8 0" cursor="fuse: true; maxDistance: 500; timeout: 500"></a-camera>

<!-- <a-box position="-1 0.5 -3" rotation="0 0 0" width="0.1" height="0.1" depth="0.1" color="#4CC3D9"></a-box> -->

<a-sky src="https://c1.staticflickr.com/8/7376/16218590470\_468084c950\_h.jpg"></a-sky>

</a-scene>

我用一个长宽高为0.1的a-box来代表一个体素。在js中使用如下代码添加体素：

var str=""+round2(x)/10+" "+round2(y)/10+" "+round2(z)/10;

var box=document.createElement('a-box')

box.setAttribute('position',str)

//box.setAttribute('position', {x: 1, y: 2: z: -3});

box.setAttribute('rotation',"0 0 0")

box.setAttribute('width', '0.1')

box.setAttribute('height', '0.1')

box.setAttribute('depth', '0.1')

box.setAttribute('color', '#4CC3D9')

scene.appendChild(box)

二、数值微分(DDA)法推广到3D

1.根据绘制的起点和终点坐标，求得x,y,z哪个方向增量绝对值最大分别用index=0，1，2代表，选取该方向（x，y，或z）作为绘制时保持+1的方向（自变量）。

2.用flag表示自变量方向增量正负情况，即可知道绘制时是x（或y、z）递增还是递减（true或者false）。同时获得两个k值（以该方向为自变量，另外两个方向为因变量，如dy/dx，dz/dx）

3.循环绘制cube。先判断自变量是哪个方向，再判断自变量是增加还是减小，每一个因变量的下一个值是上一个值+或-k\*自变量。绘制时对两个因变量取整。例：if(index==0)

{

if(flag)

{

if(x>=x1) break;

x=x+1;

y=y+k1;z=z+k2;

}

else

{

if(x<=x1) break;

x=x-1;y=y-k1;z=z-k2;

}

}

三、中点画线法推广到3D

1. 根据绘制的起点和终点坐标，求得x,y,z哪个方向增量绝对值最大分别用index=0，1，2代表，选取该方向（x，y，或z）作为绘制时保持+1的方向（自变量）。同时求出两个因变量增加方向是否和自变量一致，分别用sign1，sign2表示（0代表不增加，1代表增加方向一致，-1代表相反）。

2. 用flag表示自变量方向增量正负情况，即可知道绘制时是x（或y、z）递增还是递减（true或者false）。同时获得两个k值（以该方向为自变量，另外两个方向为因变量，如dy/dx，dz/dx）

3.如果flag为false，直接交换起点和终点，使绘制时自变量保持增长。（之前dda绘制时略微麻烦，这样操作更加简洁）。

4.计算两个因变量对应的a，b，例：a=(y0-y1); b=(x1-x0);a1=(z0-z1);b1=b;

5.根据两个因变量与自变量增加方向是否一致，分别求得，d，d1，d2，自增相同：d=2\*a+b;d1=2\*a;d2=2\* (a+b); ，自增相反：d=2\*a-b;d1=2\*a;d2=2\* (a-b);

6.循环绘制cube。当因变量增加方向与自变量一致时每个因变量在自变量+1时是否+1的判别方法与2D相同。当相反时，判别式d情况相反。例子：

if(sign1>0)

{

if (d<0) { x=x+sign1; d+=d2; }

else { d+=d1;}

}

else{

if (d>0) { x=x+sign1; d+=d2; }

else { d+=d1;}

}

四、Bresenham画线算法推广到3D

前三步与中点画线法几乎相同，第三步增加了dx，dy，dz取反操作。

1. 根据绘制的起点和终点坐标，求得x,y,z哪个方向增量绝对值最大分别用index=0，1，2代表，选取该方向（x，y，或z）作为绘制时保持+1的方向（自变量）。同时求出两个因变量增加方向是否和自变量一致，分别用sign1，sign2表示（0代表不增加，1代表增加方向一致，-1代表相反）。

2. 用flag表示自变量方向增量正负情况，即可知道绘制时是x（或y、z）递增还是递减（true或者false）。同时获得两个k值（以该方向为自变量，另外两个方向为因变量，如dy/dx，dz/dx）

3.如果flag为false，直接交换起点和终点，使绘制时自变量保持增长。（之前dda绘制时略微麻烦，这样操作更加简洁）。同时把自变量和因变量从起点到终点的增量dx，dy，dz取为原来的相反数，后面的计算会用到。

4.如果因变量增加方向与自变量相反，因变量增量取反，保证每一次因变量增量为正值，例：dy=-dy，因为后面的计算把因变量增加方向为正方形和负方向进行了统一，这样就又简化了操作，相比中点画线法使用不同的判别式。

5. 循环绘制cube。两个因变量是否+1与2D算法一样。在绘制时根据因变量增加方向是否一致，需要对因变量做一个关于对称变换。例：2\*y0-y，因为计算时保证因变量增加方向是和自变量一致（由4进行），后来绘制时需要对称变换。

五、效果。直接运行vr.html,可以实现三种算法绘制cube，并能在页面上任意输入起点和终点坐标，并选择一种方法绘制。

六、注意事项。在web中展示时可能需要鼠标拖动，因为默认的照相机方向可能没有对准绘制的cube。可以用键盘方向键控制场景和元素的相对位置。用了一个全景图作背景，更加能凸显立体的效果，所以作为代码的一部分交上去了。

七、参考资料。

<http://www.techbrood.com/aframe>

<https://zhuanlan.zhihu.com/p/25518498>