第一次实验作业， 3D体素实现划线算法

姓名 陈钧涛 **学号** 1552702

实验目的

直线段扫描转换是图形学课程的入门内容，包括：

- 确定最佳逼近于该直线的一组象素
- 按扫描线顺序，对这些象素进行写操作

课上关于直线段扫描转换共介绍了三种算法：

- 数值微分(DDA)算法
- 中点画线法
- Bresenham画线算法

而这次实验的目的就是将课上学习的2D划线算法推广到3D，加深对直线段扫描转换算法的理解

实验要求

1. 完成cube绘制
2. 选择一种2D划线算法，思考如何推广到3D
3. 使用1种的cube（体素）代替像素，绘制3D线段
4. unity/three.js OpenGL加分
5. 5月2日之前提交到tjgraphics2018@hotmail.com
6. 完成实验报告，最后写上参考文献/资料

实验原理

本次实验选用**three.js**完成，使用了orbitcontrols.js库用于控制相机位置，dat.gui.js库制作交互控件，增强实验作品的交互性。

函数名	作用
threeStart()	调用各个初始化函数
initThree()	初始化renderer
initCamera()	初始化相机
initScene()	初始化场景
initLight()	初始化光源 包括一个场景光和一个平行光
initObject()	初始化cube的形状和材质
initOribitControls()	初始化相机控制器
initGUI()	初始化交互界面
render()	渲染每一帧的内容
drawLineWith3DDDA()	使用3D DDA算法绘制直线
drawLineWith3DBresenham()	使用3D Bresenham算法绘制直线

3D DDA算法：

根据起点A坐标(x0, y0, z0)与终点B坐标(x1, y1, z1),计算向量AB(x1-x0, y1-y0, z1-z0)

根据为了防止直线不连续，我们以向量AB的最大分量为基准

例如假如x1-x0为最大分量 那么

dx = (x1-x0)/|x1-x0|

dy = (y1-y0)/|x1-x0|

dz = (z1-z0)/|x1-x0|

体素坐标(x, y, z) = (round(x+k * dx), round(y+k * dy), round(z+k * dz))

即每次增加(dx, dy, dz) 并四舍五入取整

3D Bresenham算法：

来自Matlab官方源码

```
function (X,Y,Z) = bresenham_line3d(P1, P2, precision)
```

```
if ~exist('precision','var') | isempty(precision) | round(precision) == 0
   precision = 0;
   P1 = round(P1);
   P2 = round(P2);
else
   precision = round(precision);
   P1 = round(P1*(10^precision));
   P2 = round(P2*(10^precision));
end

d = max(abs(P2-P1)+1);
X = zeros(1, d);
Y = zeros(1, d);
Z = zeros(1, d);

x1 = P1(1);
y1 = P1(2);
z1 = P1(3);

x2 = P2(1);
y2 = P2(2);
z2 = P2(3);

dx = x2 - x1;
dy = y2 - y1;
dz = z2 - z1;

ax = abs(dx)*2;
ay = abs(dy)*2;
az = abs(dz)*2;

sx = sign(dx);
sy = sign(dy);
sz = sign(dz);

x = x1;
y = y1;
z = z1;
idx = 1;

if(ax>=max(ay,az))          % x dominant
   yd = ay - ax/2;
   zd = az - ax/2;

   while(1)
      X(idx) = x;
```

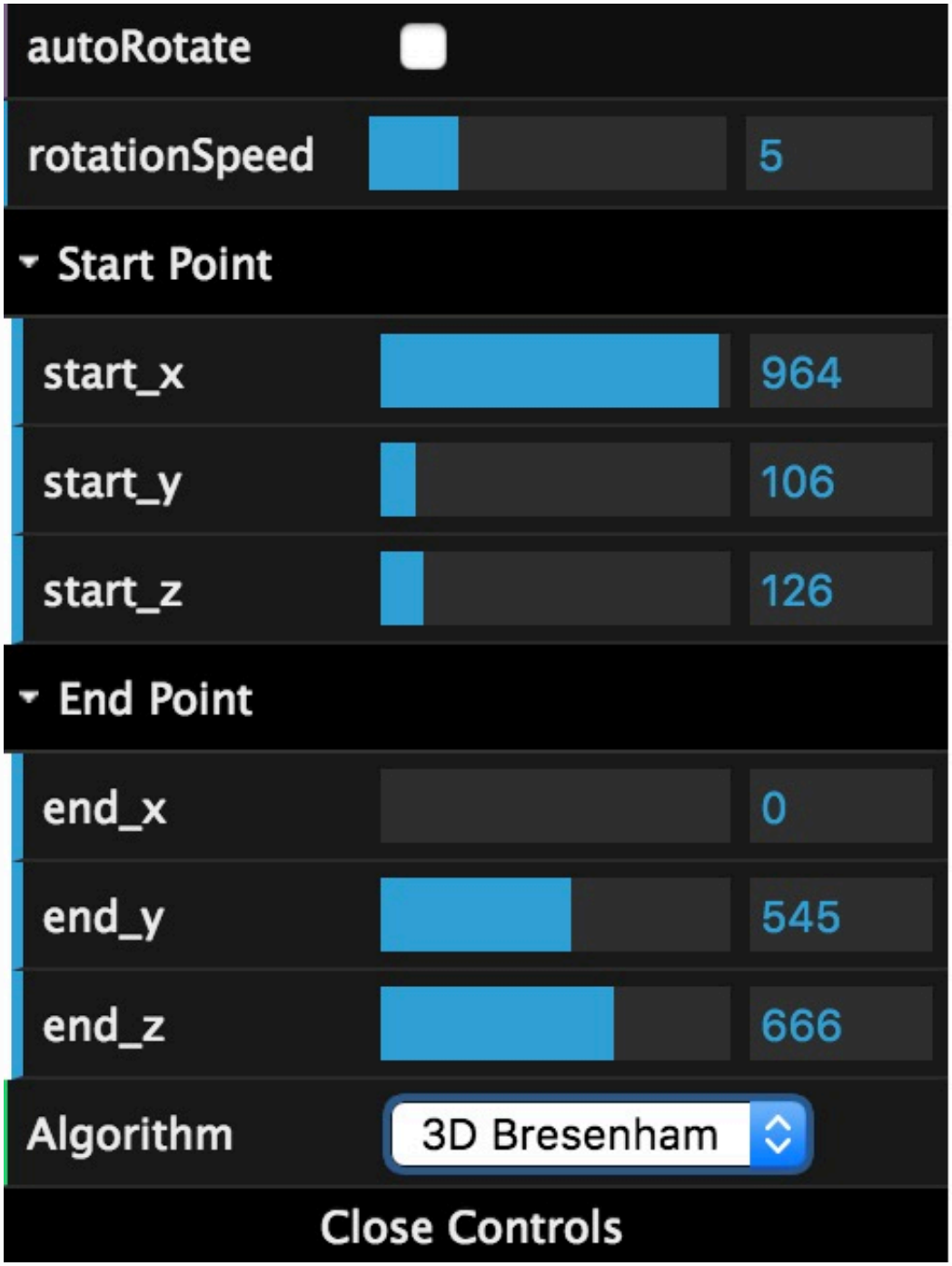
```
    Y(idx) = y;
    Z(idx) = z;
    idx = idx + 1;

    if(x == x2)        % end
      break;
    end

    if(yd >= 0)        % move along y
      y = y + sy;
      yd = yd - ax;
    end

    if(zd >= 0)        % move along z
      z = z + sz;
      zd = zd - ax;
    end

    x  = x  + sx;      % move along x
    yd = yd + ay;
    zd = zd + az;
  end
elseif(ay>=max(ax,az))       % y dominant
  xd = ax - ay/2;
  zd = az - ay/2;

  while(1)
    X(idx) = x;
    Y(idx) = y;
    Z(idx) = z;
    idx = idx + 1;

    if(y == y2)        % end
      break;
    end

    if(xd >= 0)        % move along x
      x = x + sx;
      xd = xd - ay;
    end

    if(zd >= 0)        % move along z
      z = z + sz;
      zd = zd - ay;
    end

    y  = y  + sy;      % move along y
```

```
    xd = xd + ax;
    zd = zd + az;
  end
elseif(az>=max(ax,ay))       % z dominant
  xd = ax - az/2;
  yd = ay - az/2;

  while(1)
    X(idx) = x;
    Y(idx) = y;
    Z(idx) = z;
    idx = idx + 1;

    if(z == z2)        % end
      break;
    end

    if(xd >= 0)        % move along x
      x = x + sx;
      xd = xd - az;
    end

    if(yd >= 0)        % move along y
      y = y + sy;
      yd = yd - az;
    end

    z  = z  + sz;      % move along z
    xd = xd + ax;
    yd = yd + ay;
  end
end

if precision ~= 0
  X = X/(10^precision);
  Y = Y/(10^precision);
  Z = Z/(10^precision);
end

return;                % bresenham_line3d
```

实验结果

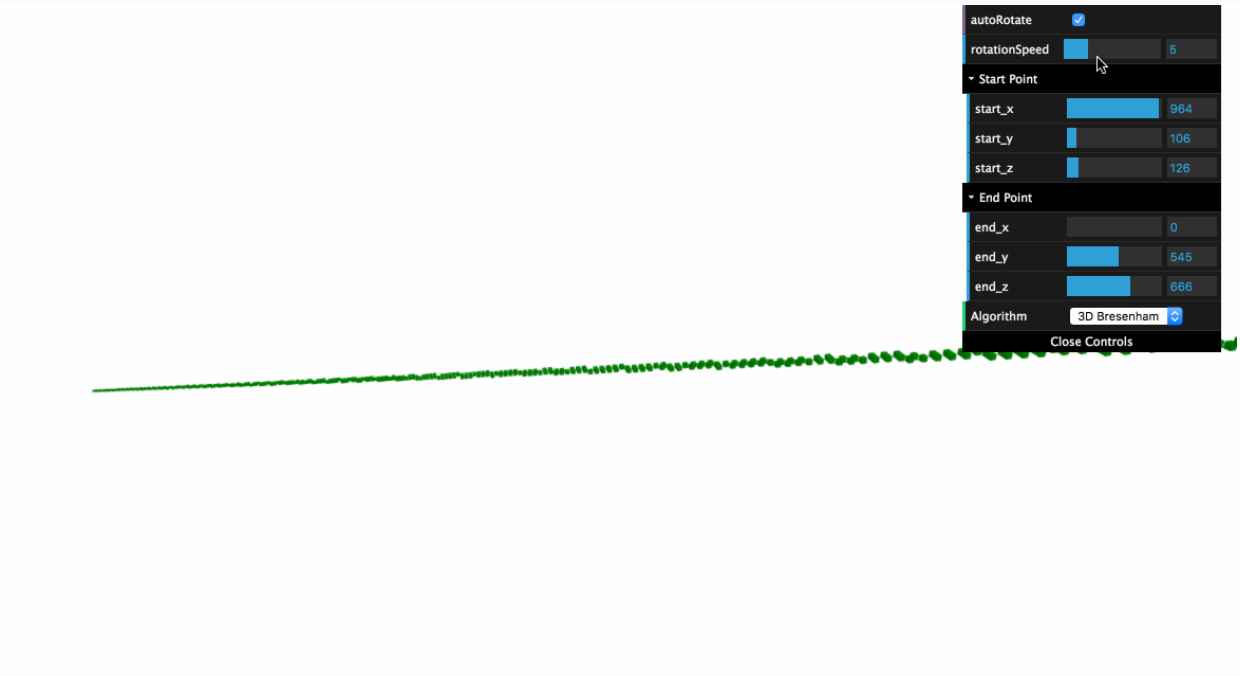
本实验以一个web的形式呈现，使用方法如下

1. 直接运行index.html 或者访问http://closecv.com/line3d
2. 使用控制面板调节参数，是否自动旋转，转速，起点终点坐标等



3. 使用鼠标拖拽转动相机，鼠标滚轮控制距离
4. 可以选择不同算法，拉近距离可以比较不同算法的区别

效果如图



参考文献

- 3D Bresenham's line generation *https://ww2.mathworks.cn/matlabcentral/fileexchange/21057-3d-bresenham-s-line-generation?focused=5102923&tab=function*
- Algorithm for 3DDDA *http://euklid.mi.uni-koeln.de/c/mirror/www.cs.curtin.edu.au/units/cg351-551/notes/lect12p1.html*