# 作业1报告《2D图形动画和交互》

**课题名称： 2D图形动画和交互**

课题负责人名（学号）： 2016141503045

同组成员名单（角色）： 朱佳旻

指导教师： 陈蓉

**评阅成绩：**

评阅意见：

提交报告时间：2019 年 10 月 9 日

**2D图形动画和交互**

计算机科学与技术 专业

**学生** 朱佳旻 **指导老师** 陈蓉

**[摘要]** 利用webGL技术，使用koch曲线，绘制分形雪花图案，并处理着色和交互。学习GPU编程的原理和着色器编程语言的开发方法。学习JavaScript的事件处理和基本API操作，通过解析几何等方式完成koch曲线的数学迭代算法。通过Vue.js和MDUI，实现响应式的现代web交互界面。

**关键词**：计算机图形学 webGL 分形 Koch曲线

利用koch曲线，绘制雪花形状。

1. 概述

作业要求绘制雪花形状，并能够形成颜色渐变等效果。同时，还要处理平移和旋转等问题。在实现以上要求的基础上，兼具美观性能。

1. 设计（分析问题，设计解决问题的总体框图和思路）

查找资料可知，雪花形状的分形曲线称为koch曲线。Koch曲线的主要思想是：

从一条直线段开始，将线段中间三分之一部分用等边三角形的两条边代替，形成具有5个结点的图形(图1)；在新的图形中，又将图中每一直线段中间的三分之一部分都用一等边三角形的两条边代替，再次形成新的图形，以此递归。

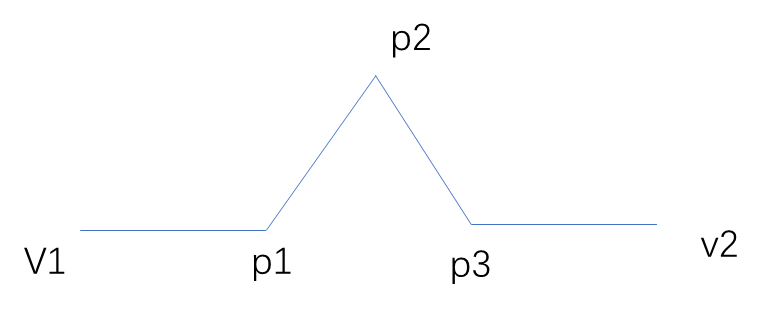


图 1

由于分形曲线具有的递归性质，算法也应该同样使用递归方法实现。算法的主要难点在于：

1. 求出以两个已知点为底的等边三角形的第三个顶点的坐标
2. 判断该点位于图形内部还是图形外部（因为雪花图案中所有三角形均向外拓展）

对于第一个难点，令图1中p1点坐标为，p3点坐标为，所求的p2点坐标为，则可以通过解析几何的方法算得：

对于第二个难点，有两种解决方法，第一种是通过解析几何方法，给定三角形三个点和新的点，判断该点是否在三角形内部。第二种方法是观察分形生成的三角形形状可以发现，在相邻的两次迭代中，三角形的朝向会反转一次，如图2所示，较细的线标出的是新生成的三角形。

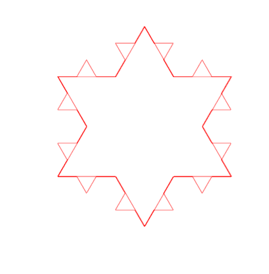


图 2

利用这一特点也可以确定第三点的位置。这里我选用的是第二种方案。

在绘图的实现上，可以采用如图3所示的流程图，具体来说就是首先设计着色器程序，设置表示中心点坐标和旋转角度的变量，以及顶点和颜色数据。然后进入递归程序，递归生成图形所需的所有顶点，发送顶点数据给着色器并绘图即可。

处理交互的方法主要采用了js的事件机制和Vue.js的绑定和侦听器。例如，当鼠标点击绘图区域时，js捕获到鼠标点击事件，进入对应的回调函数，计算出坐标并更新Vue.js的对象属性。该对象属性绑定有侦听器，当其值改变时，会自动将新的坐标发送给着色器。该对象的属性同时还双向绑定至控制面板的坐标输入框中，修改输入框也会触发侦听器并发送新的坐标数据。



图 3

1. 实现（阐述基于设计而实现的平台，关键技术和主要代码）

主要使用了HTML网页和webGL平台。使用的技术包括：

* webGL
* HTML/css/javascript
* MDUI
* Vue.js

主要的代码如下：

|  |
| --- |
| function koch(start, end, depth, position){      if(depth >= MAX\_DEPTH) return;      var p1 = mix(start, end,0.333333) //找到线段的1/3点      var p2 = mix(start, end,0.666667) //找到线段的2/3点      var dx = p1[0]-p2[0] //计算x1-x2      var dy = p1[1]-p2[1] //计算y1-y2      var k;  //k表示斜率的符号      if(Math.abs(dx)<=0.0001||Math.abs(dy)<=0.0001) k=0;      else if(dx<0&&dy<0||dx>0&&dy>0) k=1;      else k=-1;      if(k<0){          if(position>0){              if(dy<0){                  dy=-dy;                  dx=-dx;              }          }else{              if(dy>0){                  dy=-dy;                  dx=-dx;              }          }      }else if(k>0){          if(position>0){              if(dy>0){                  dy=-dy;                  dx=-dx;              }          }else{              if(dy<0){                  dy=-dy;                  dx=-dx;              }          }      }else{          if(position>0){              if(dx<0){                  dx=-dx;                  dy=-dy;              }          }else{              if(dx>0){                  dx=-dx;                  dy=-dy;              }          }      }      var new\_x = 0.5\*(p1[0]+p2[0]+1.73205\*dy) //计算新的点的x坐标      var new\_y = 0.5\*(p1[1]+p2[1]-1.73205\*dx) //计算新的点的y坐标      var new\_p = [new\_x, new\_y] //构造新的点        koch(start,p1, depth+1, position);      points.push(p1);      colorsOfVertexs.push(c1);      koch(p1, new\_p, depth+1, -position);      points.push(new\_p);      colorsOfVertexs.push(c2);      koch(new\_p, p2, depth+1, -position);      points.push(p2);      colorsOfVertexs.push(c3);      koch(p2, end, depth+1, position);  } |

上面是koch曲线绘制函数的代码。

|  |
| --- |
| angle = new Vue({          el:"#Rotation\_Angle",          data:{              angle:theta,              step:0.1          },          watch:{              angle: function(){                  if(typeof(this.angle)==typeof('123')) this.angle=parseFloat(this.angle);                  gl.uniform1f(thetaLoc, angle.angle);              },              step: function(){                  if(typeof(this.step)==typeof('123')) this.step=parseFloat(this.step);              }          }      });      XY = new Vue({          el:"#XY\_input",          data:{              centerX:centerX,              centerY:centerY          },          watch:{              centerX: function(){                  if(typeof(this.centerX)==typeof('123')) this.centerX=parseFloat(this.centerX);                  gl.uniform1f(centerXLoc, this.centerX);              },              centerY: function(){                  if(typeof(this.centerY)==typeof('123')) this.centerY=parseFloat(this.centerY);                  gl.uniform1f(centerYLoc, this.centerY);              }          }      }); |

上面是Vue对象的初始化代码。

|  |
| --- |
| function render()  {      gl.clear( gl.COLOR\_BUFFER\_BIT );      //\*\*\*\*\*\*\*增加代码\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*      if(animflag)//如果旋转控制按钮由切换，需要发送旋转角度给shader      {          if(typeof(angle.angle)==typeof('123')) angle.angle=parseFloat(angle.angle);          angle.angle += angle.step ;        };        if(sliderchangeflag)//如果slider值有变化需要发送Gasket2D 新初始顶点属性数据给shader      {          gl.bindBuffer( gl.ARRAY\_BUFFER, bufferId );          gl.bufferData( gl.ARRAY\_BUFFER, flatten(points), gl.STATIC\_DRAW );          gl.bindBuffer( gl.ARRAY\_BUFFER, colorbufferId );          gl.bufferData( gl.ARRAY\_BUFFER, flatten(colorsOfVertexs), gl.STATIC\_DRAW );      }        /\*      if(centerchageflag)//如果鼠标重新点击了中心，需要把新中心传递给shader      {          gl.uniform1f(centerXLoc, centerX);          gl.uniform1f(centerYLoc, centerY);      }      \*/      //gl.drawArrays( gl.TRIANGLE\_STRIP, 0, points.length );      //gl.drawArrays( gl.LINE\_LOOP, 0, points.length );      gl.drawArrays( gl.TRIANGLE\_FAN, 0, points.length );      sliderchangeflag=false;      centerchageflag=false;      //\*\*\*\*\*\*\*增加代码\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*      requestAnimFrame(render);  } |

以上是渲染函数的代码。

图片包含 屏幕截图

描述已自动生成

图片包含 屏幕截图

描述已自动生成

以上是最终成果展示。

1. 总结（总结现有的优点和不足，个人的心得体会和改进方向）

目前的优点是界面美观、响应式设计、可调节参数较多。

目前的不足是：在填充雪花这一环节有一些问题。由于koch本质上是曲线而不是图形，因此和上课讲的gasket不一样，比较难使用gl.TRIANGLE绘图，可以使用gl.LINE\_LOOP绘制环线，但是比较不明显。因此我使用的是gl.TRIANGLE\_FAN来填充，但会造成内凹的部分线条被覆盖的问题。目前还有待改进。

心得体会：图形学的编程方式和传统的编程方式确实存在很大的差别。传统方式的编程应该是不断计算出新的点的坐标，然后逐个在屏幕上显示出来。而图形学的方式则是CPU只控制旋转角度和中心位置，由GPU计算点的坐标。这样做，减轻了CPU的计算压力，也避免了每次都要向GPU发送顶点数据的IO压力，并且能大幅度提高并行性，高效利用GPU的硬件资源。

改进方向：可以进一步改进填充颜色的方式，避免上述问题的出现。

# 参考文献

[1] Edward Angel, Dave Shreiner.交互式计算机图形学——基于WebGL的自顶向下方法（第七版）.电子工业出版社.2016

[2] 晨心. matlab绘制peano（皮亚诺）曲线和koch（科赫曲线，雪花曲线）分形曲线. https://segmentfault.com/a/1190000015066537,2018

[3] MDN web docs. https://developer.mozilla.org/zh-CN/,2019

[4] MDUI. https://www.mdui.org/,2019

[5] Vue.js. https://cn.vuejs.org/,2019