《计算几何》大实验选题报告

吴伟杰 2006210812
许　磊 2006210786
金之星 2006210751

1 实验题目

简单多边形的三角剖分

2 实验目标

把简单多边形分解为单调多边形和对单调多边形进行三角剖分是计算几何中的经典问题；通过实现对应的两个算法，加深对扫描线算法的理解，体会其算法设计思想。

在实验的过程中，重点在于详细演示剖分的关键步骤，尽可能多地覆盖“分解”、“剖分”过程中会遇到的各种情况，清晰地展示剖分算法的执行过程。

3 总体描述

该实验分为两个部分，对应的两个算法都是基于平面扫描线的。扫描意味着排序，首先要有一个通用的 $n\log n$ 排序模块。

对于“简单多边形的单调分解”，主要集中在石钟乳和石笋的识别、helper 的确定、扫描过程中事件类型的判定及分类处理上。

对于“单调多边形的三角剖分”，主要工作是 SCC 条件的保证、事件类型的判断和处理。

在做到算法实现代码正确（完成剖分并把结果呈现出来）、效率合乎要求的基础上，争取能够对 helper 进行突出显示、单步执行、显示事件类型及栈状态、栈操作统计等。

另外，增强程序的健壮性：做一些常规的意外处理，如判断输入是不是简单多边形、是否退化情况等。

整体的数据结构和流程将参照第 3 部分讲义的 03.Tri.C.Monotone Polygons.ppt 和 03.Tri.D.Monotone Decomposition.ppt讲义。

考虑到对剖分细节的呈现，使用 Flash 比较理想，但由于 Flash 的逻辑是使用脚本来控制的，解释执行，从理论上讲会比 Java 还慢，另外由于不编译，编码过程中调试困难。如果通过一周的技术路线尝试，决定使用 Java，那么在显示方面就会有所折扣，如栈状态等可能就不做了。

4 日程安排

周次	日期	任务
11	2006.11.26~2006.12.02	尝试并确定技术路线
12~13	2006.12.03~2006.12.16	基本部分模块划分及分别实现
14~15 上	2006.12.17~2006.12.27	联合调试，保证算法正确性
15 下~16	2006.12.28~2007.01.03	加入显示增强部分

使用 Flash 开发算法演示程序小结

为什么用 Flash

Flash 由于画面精美、表现力强、制作方便、易于传播，在网页制作、影视宣传、产品展示、课件教程等邻域得到了广泛应用。

使用 Flash 来编写算法演示程序，主要考虑到 Flash 以下三方面的特点：

矢量图形

这是 Flash 最突出的优势。由于基本图形元素（曲线、平面区域等）的矢量化表示，使得 Flash 文件体积往往远小于位图表示方法生成的文件；更重要的是由于矢量表示以及播放器采用的反走样技术，在画面放大时，不会像位图那样出现令人不悦的锯齿。因此用 Flash 表现层次相对简单的场景，容易达到更好的显示效果。

从计算几何算法演示的角度，程序的功能的实现，往往依赖于对点、线、多边形、椭圆等几何元素的频繁操纵。Flash 本身“矢量至上”的设计初衷，使得我们在处理这些元素的移动、缩放、相互遮挡、不同透明度的混合时，要比使用 C++、Java 等语言的绘图 API 要轻松很多倍。另外，使用 Flash 还可以很简单地通过关键帧和补间动画地为矢量图形加上多种动态效果，极大地提升用户的观赏体验，而如果使用后者来实现就难上加难了。

交互性强

除了可以方便产生动画以外，Flash 在处理用户的键盘、鼠标输入时也给了开发者极大的方便。由于采用了类似 Java 的 Listener 设计模式，我们只需要对 Key、Mouse 这些全局对象或特定元件注册事件监听器，就可以截获这些事件。从而，我们可以把更多的精力投放在事件处理部分的代码编写，而不是事件与控件如何映射这样的繁琐而无技术含量的工作。

因此，使用 Flash 开发做出的演示程序，对用户更加友好，功能接口更加直观，没有帮助文档也可以很快上手，极大地激发用户使用欲望。

易于发布

Flash 发布的文件中会选择性地嵌入矢量字体，甚至可以嵌入源文件中没有显式包含而在运行时可能用到的文字对应的字体，并保持其矢量方式表示。即使目标环境中没有这样的字体，或字体平滑效果没有开启，都不会影响其最终的显示效果。

Flash 动画的最常用发布形式是嵌在网页中，在装有 Flash 插件的浏览器上即可播放，这一点需求在 MacOS、Linux 各种发行版、Windows 这些主流桌面操作系统及大部分嵌入式系

统中往往都已经事先满足，即使没有其部署成本也远远低于 Java 运行时环境。

此外，我们还可以把制作好的影片导出成可以在 MacOS 和 Windows 平台上独立运行的程序，文件体积增大只有 1.5 兆字节左右。

Flash 的若干局限

当然，Flash 也并不是全是优点，在侧重表现力的同时，在其它方面的功能就有所折扣。对于算法演示程序的需求，主要不足体现在：

读文件有限

Flash 给开发人员的文件读取接口比较单纯，只能读取表单形式(a=1&b=2)或 XML 的这样的文本内容。如果要导入数据，就只能采取上述两种格式。

另外，远程网页方式运行的 Flash 影片，默认只能读取网络文件（即远程 URL 形式）；本地网页和程序运行的 Flash 影片，默认只能读取本地的文件。

虽然通过 Flash 影片可以向服务器上传文件，但是这个过程对程序员是透明的，我们无法获得其内容，甚至在用户通过文件浏览对话框选中文件之后，连它的完整路径都不能获得。所以需要用户手动输入文件的完整路径。

不能写文件

出于安全因素，使用 Flash 程序不能写文件，因此对实验结果的批量输出不甚方便，通常我们把结果格式通过文本框显示，对于大量数据难以胜任了。

执行效率低

Flash 中的逻辑代码是解释执行的，因此理论上讲会比 Java 还慢，特别是当算法部分的复杂度较高或处理数据量巨大时。但是，在演示的算法复杂度和数据量一般时，由于本身对图形的快速处理，这种效率低的缺欠很难察觉，至少看起来会比 Java 程序更灵敏。

编码环境弱

Flash 并不提供类似于 Xcode、VisualStudio 那样的集成开发环境，可能把相关的文件组织成一个工程。演示界面部分源文件及其用到的类或接口的定义文件需要开始者自行维护其组织关系。

Flash 对于代码的提示功能也比较简单，也不是很智能。

Flash 不具备一般程序员所习惯和期待的调试环境，特别是对于通过外部文件定义的类，断点是无效的，只能通过 trace 语句输出来观察程序运行时的内部数据。

Flash 源文件的内部组织结构

一个典型的 Flash 源文件(后缀通常为.fla)的组织结构，在舞台场景里，通过时间线、层在时间和空间两个维度展开。

时间——帧

Flash 通过在时间线上不同的关键帧的展现，还表现影片的时序逻辑。开发者在不同的关键帧安排不同的元素或改变其相互关系，来向用户传达不同的信息。

整个场景可以具有多个帧，场景的某些元素内部也可以有多个帧。这样我们可以尽可能地封装变化，管理行为，充分发挥面向对象设计方法的优势。

默认情况下，帧是按照其编辑时的先后顺序依次呈现给观看者的，但是在使用了脚本之后，可以做到帧间的跳转，从而实现相对复杂的逻辑。

帧可能有名字，这样方便开发者通过其名字来获取并进行操作，如跳转。

空间：

z 轴——层

Flash 通过层来确定构成元素在观看者视线方向上的排列顺序，这一点与其它平面设计软件如 Photoshop 中对元素遮挡关系的处理是类似的。

处于上方的层内的元素会遮住下方的层中的元素，具有透明度的元素按照 Alpha 值来混合。通过使用蒙板层，还可以特定的层中的元素只显示出特定区域的部分（在后续小节中介绍）。

xOy 平面——元件

Flash 中的元件（Symbol），分布在某个特定的层上，在 xOy 平面上进行分布。它们类似于影片中的各种角色，我们通过对它们的控制来完成演示。元件主要分为三种类型：

图形

这是最简单的一种元件，只有一帧，适合组织静态的图形或文字，它们的内部没有行为的变化，也不能响应事件。对于它的不再实例，开发者无法在代码中区分和引用。

影片剪辑

这是 Flash 中最常用的元件，也是功能最强的。可以有多个帧，可以监听事件，并具有一系列供开发者使用的属性和方法。

它对应于 MovieClip 这个内置类型，每个实例可以有一个名称，或在编辑过程中由开发者输入，或在动态创建时在代码中指定。通过它们的名称可以获得对这个影片剪缉的引用，通过其接口方法或公共字段来改变其行为。

按钮

它是一种特殊化的影片剪辑，有 4 个帧，分别对应按钮的 4 种状态，可以注册事件处理函数，和一般应用程序中的按钮起相同的作用，只不过具有更强的定制性。我们也可以通过 MovieClip 来实现自定义按钮。

编程语言 ActionScript 简介

在 Flash 中进行逻辑控制主要采用 ActionScript，它的代码可以写在关键帧中，也可以保存在.fla 文件的外部，类的定义只能写在外部。

它基于由 JavaScript 抽象出的 ECMA-262 标准，因此与 JavaScript 在很多方面具有相似性，但是同时也具有 ECMA-262 没有规定的功能，主要特征如下：

脚本语言

与 JavaScript 类似，它是一种弱类型的语言，以 var 的方式来声明变量。由它编写的代码是解释执行的。它与 JavaScript 的不同主要有：

ActionScript 不支持特定于浏览器的对象，例如 Document、Window 和 Anchor。

ActionScript 没有为所有 JavaScript 内置对象都提供支持。

ActionScript 不支持某些 JavaScript 语法构造，例如语句标签。

在 ActionScript 中，eval()函数只能执行变量引用。

ActionScript 不支持使用 RegExp 对象的正则表达式。

面向对象

ActionScript 是一种支持面向对象的脚本语言，可以用 class package_path.ClassName {…} 这样的方式来定义类，类的定义必须放在与之同名后缀为.as 的文件中，并支持以包的方式来组织类的结构。通过 import package_path. ClassName 使用特定的类。这与 Java 的作法非常相似。

ActionScript 支持继承,比如我们可以通过定义一个 MovieClip 的子类来封装特定的功能。当然我们也可以通过改变内置类的 prototype 的方法来做到这一点，这种做法往往用于对特定的类进行功能的临时扩展。如为 MovieClip 类添加画虚线的功能。

支持强类型

虽然，脚本通常都只有弱类型，但是 ActionScript 也允许我们使用强类型，并在编译过程中提供类型匹配检查。使用强类型的方式定义一个类，其内容可能如下：

```
Class net.polarray.cg.algorithm.Point {
    var _idx:Number = -1;
    var _x:Number;
    var _y:Number;
    function toString():String {
        return “Point[ _idx: ” + _idx + “, _x: ” + _x + “ _y: “ + _y + “]”;
    }
    fuction Point(idx:Number, x:Number, y:Number) { //  构造函数
        /* TODO */
    }
}
```

惯例与技巧

下面介绍一下我们小组在制作这个 Flash “三角剖分” 演示程序中的关键技术。

影片剪辑的基本操作

把演示中主要的元素制成影片剪辑，最重要的是 “顶点”，即标志为"Vertex"的元件。

这个影片剪辑具有如下几个关键帧：grow, still, pushed, popped, twinkle, role, delete。

帧名	表示状态
grow	用户点击后，顶点产生，显示膨胀并复原的效果
still	顶点的常态，即静止
pushed	表示顶点被压栈的状态，显示由某一高度落入栈中的效果
popped	表示顶点被弹出的状态，显示由栈中弹出并消失的效果
twinkle	表示多边形即将闭合的状态，顶点闪烁

role	表示顶点在扫描线当前状态下所处的角色，如 S、T、C，或 Helper，显示绿光圈
delete	表示顶点不再是一个 Helper，从数据结构中删除，显示红光圈

当某个顶点需要改变状态时，便调用 MovieClip 的 gotoAndPlay 方法，跳到特定的帧中。gotoAndPlay 是 Flash 中控制帧的跳转最常用的方法，它的参数就是帧的名称。与之相似的一个方法是 gotoAndStop，这个方法会把剪辑停到指定的帧，相对使用得较少。

使用 gotoAndPlay 要注意保证该 MovieClip 当前并没有播放参数所指定的帧，如果两次调用 gotoAndPlay 到同一目标帧，相隔时间太短，以致到第二次调用时，目标帧仍然处于播放状态，这样会导致该剪辑代码崩溃，将从第一帧开始循环播放，脚本不再起作用，从而导致程序的异常行为。

对 MovieClip 的操作主要通过访问以下属性达到：

_x, _y

每次用户在多边形输入区点击后，需要在点击处新生成一个 Vertex 剪辑的实例，并把它放到点击位置。即把该实例的(_x, _y)改变为(_xmouse, _ymouse)。

_width, _height

在表示栈等数据结构时，需要维护栈中的元素并把它们放到合适的位置，主要通过获取元素对应的剪辑的宽或高（即_width 或_height）来计算新加入的元素应该放在什么位置。

_xscale, _yscale

由于 Flash 主要采用矢量表示，在用户改变容器大小后，工作区的所有元素会同比放大，有效的分辨率并没有增加。为了允许用户充分利用屏幕的区域，使输入区可以缩放，满足不同输入习惯。这个用 Flash 实现非常容易，只要修改整个画面对应的 MovieClip 的 x, y 方向的缩放比例即可。原始大小，_xscale 和_yscale 取 100,若缩为一半则分别取 50。

_rotation

这个属性使开发人员控制影片剪辑的旋转角，角度制，在本程序中没有用到。

_alpha

修改 MovieClip 的透明度。例如，在把简单多边形分成若干个单调多边形，每个单调多边形的区域用不同透明度的黑色来填充，这样就把它们区别开了；然后需要用户选择其中一个进行三角剖分。这时，当用户把鼠标入到某个单调多边形上时，就增加该多边形填充区所在 MovieClip 的透明度，从而达到对 MouseOver 事件的响应。_alpha 的值为 0 到 100，0 为全透明，100 为完全不透明。

_visible

修改这个属性值可以切换元件的显示与隐藏，例如在用户输入多边形时，就把扫描线隐藏，而在单步演示时，再把它显示出来。

动态创建影片剪辑

在程序运行过程中，需要大量动态创建影片剪辑，而不是仅限于操纵已有元件。主要用到了：

duplicateMovieClip

复制已经存在的剪辑，需要在编辑之初有该剪辑一个静态的实例。

attachMovie

指定剪辑在元件库中的唯一标志，需要设置该剪辑的 Linkage 属性，使之导出到 ActionScript 中，可以使用。它不需要原来的影片中已经存在这样的实例。

createEmptyMovieClip

创建一个空的剪辑，多用来组织动态创建的同类元件。如在本程序中，动态创建了 vertices_mc, edges_mc, labels_mc, monoinner_mc 等，分别盛放顶点、边、边的名称、单调多边形的填充区等不同类型的元件。

上述方法都需要指定新创建的影片剪辑所处的深度，如果指定的深度已经有其它剪辑了，它们会被新的剪辑抹掉。为了避免这种情况发生，这个深度值通过 MovieClip 的 getNextHighestDepth 方法来给出。

相交检测——hitTest 与_hitArea

在编写代码时，经常需要判断某个点是不是位于特定的区域上，特别是需要判断它有没有在某个影片剪辑的范围内。如在用户输入点集时，要保证不同的点之间要保持一定的距离，而不能重合或粘连，这时就需要调用 MovieClip 的 hitTest 方法，检查新的点的坐标是不是位于已有顶点剪辑范围内。这时需要注意作为参数传进去的坐标不是相对该剪辑的，而是相对最多层的场景的，需要通过 localToGlobal 来转换。

该方法也可以接受 MovieClip 作为参数，判断两个剪辑是否相碰。

最后一个 Boolean 型的参数用于指定是按照剪辑实际所覆盖的区域来判断还是通过其最小包围矩形（区域一般大于前者）来判断是否相碰。

指定一个剪辑的_hitArea 为其它剪辑，允许我们改变默认方法的行为，适应特定需要。

使用 trace 调试

由于没有一个完善的调试环境，如果我们需要观察程序运行特定位置某变量的值，常用的做法是使用 trace 语句输出。在 Flash 的编辑模式下击 Ctrl+Enter，导出并调试影片，会看到输出。以外部发布模式运行不会看到输出。

有时，为了方便观察自定义的类的信息，推荐的作法是在该类中重载其最上层基类 Object 的 ***toString*** 方法，这样就可以通过 trace 语句得到我们关心的信息，否则我们只得到类似 “[Object Object]” 这样的输出。当然可以编写特定的 output 之类的方法再调用该方法，但这样做没有充分利用面向对象带来的便利。

flags 层与 scripts 层

在需要控制帧间跳转的剪辑内部，常见的作法是在顶层建两个空的层（不含有任何元件），分别命名为 flags 和 scripts，作用就是指定帧的名称和盛放逻辑代码。这是一种剪辑内部清晰组织的一种好办法，在开发社区中被广泛认可。

补间动画

对于矢量图形的移动、简单变形、透明度变化（淡入淡出），可以通过设定同一个元件在几个关键位置（帧）的属性，在这些帧中使用补间动画，可以做出简单的动画。本程序中的顶点膨胀、入栈、出栈、闪烁等动态效果就是通过补间动画实现的。

补间动画分为形状和动感两种，分别应用于基本形状和元件，如果匹配不当会没有效果。

读取参数

对于简单的参数的读取，通常采用 LoadVars 类，接收的是形如 a=0&b=1 这样的表单格式。这个类本意是读取服务器的影响的，但是同样也可以读取本地文件，只要内容符合要求即可。

对于复杂的数据类型，可以用 XML 来描述，在代码中使用 XML 类来加载。

蒙板

一个层中的元素所能呈现的区域由另一个层中的元件的轮廓决定，后面的层就是蒙板。这个概念与 Photoshop 中的蒙板相似，但是不提供不同透明度的蒙板，只能完全显示或完全不显示。在一个层的名称标签上击右键，选择 mask，即可把一个变通层转化为蒙板。这种技术在处理扫描线水平运动的动画时使用到，用来指定扫描线的可见范围。

动态画实线及擦除、清空

在用户输入多边形和显示剖分结果时，需要动态画线。

如果不填充区域，只要调用 MovieClip 的 moveTo 和 lineTo 方法，且上一次 lineTo 的位置默认为下一次 lineTo 的起点。如果是画拆线，只需要在最开始调用一次 moveTo。在画线前可以通过 lineStyle 设置线的粗细、颜色、透明度等。

如果需要填充区域，在画线开始之前，调用 beginFill 设定颜色等填充选项，完毕后调用 endFill 即可。

动态画虚线

由于 MovieClip 没有提供画虚线的接口，通常需要在其原型上进行扩展，国外有人写了一段代码可以直接复制到目标剪辑的第一帧的脚本层中使用，放在本 PDF 文件的附件（需要 Adobe Reader 7 以上打开）中。

数组的使用

ActionScript 中的 Array 是一种重要的数据结构，除了用作数组，可以当作栈、队列、链表来使用，或者其它特殊数据结构。其重要方法包括：push(压栈或入队), pop(弹栈), shift(出队), unshift(加入队首), splice(删除一段长度可为零的连续元素并同时插入新元素，完成链表的删除插入操作)。