球面 Voronoi 图生成算法实现

摘要

Voronoi 图是计算几何领域研究最多、应用最广的一种传统技术之一。通过仔细研究 Voronoi 图的基本原理和相关算法，本文将传统的平面上的 Voronoi 图扩展到球面上，并同时将表达 Voronoi 图的数据结构 DCEL (Double Connected Edge List)和 Voronoi 生成算法 Sweepline 算法扩展到球面。最终，我们实现了一个产生球面 Voronoi 图的系统。该系统提供如下功能：

1. 实时的球面 Voronoi 图的生成演示（Sweepline 算法）
2. 动态增加新 Site、改变已有 Site，并实时更新球面 Voronoi 图结果
3. 易用、简洁的用户交互和界面

关键词： Voronoi 图 球面 Voronoi 图 SweepLine

1、问题背景

做为三维空间中一种最基本的临域表示方法，也由于球体和球面具有立方体和平面所没有的良好性质，球体在很多几何问题中被涉及，从而得到了广泛的研究。

Voronoi 图（Voronoi Diagram，简称 VD）是计算几何领域的一个重要问题，它有着很强的现实意义。在球面上的 Voronoi 图及与其对偶的 Delaunay 三角剖分是分析球面性质的重要工具。

这里定义球面上两点的距离为通过球面两个点之间最短路径的长度，也就是过两个点确定的大圆上较小的弧长。以这一距离定义导出的 Voronoi 图与平面上的 Voronoi 图性质类似，但是不会出现平面 Voronoi 图中必然出现的射线或直线边。例如，一组出现在“赤道”上的点所构成的 Voronoi 图是它们“经度”中值处的经线，它们会相交于“南北极”，而不会出现平面 Voronoi 图中的一组平行线。

另外，由于球面上的任意一点距离球心的距离是相等的这一性质，使得球面上的 Voronoi 图实际上和同样的点在三维空间中的 Voronoi 图是一一对应的。从而在求出球面上 Voronoi 图时已经求出了三维空间中的 Voronoi 图，形成了一种对三维空间的剖分，相信会对于天文计算等实际应用提供工具。

如果时间允许，还可以更进一步研究任意流形或网格模型表面的 Voronoi 图的生成。

2、算法及原理

2.1、基本单位的定义及性质

参考 Fortune 在文献 0 中描述的扫描线算法，把它应用于球面中。

定义 Site，为球面上任意一个点，由用户制定；定义 Vertex 为球面上的点，由算法生成；定义 Edge 为过两个 Vertex 的一段大圆弧，或者整个大圆；定义 Facet 为由 Edge 围成的一个区域，其中只可能有一个 Site。

数据结构上使用了 DCEL 结构，结构要求 Edge 表示一条有向半边，存储向关联的起始 Vertex、终止 Vertex、所在 Facet、下一条 Edge、上一条 Edge 等信息；Vertex 中存储一条相关 Edge，该 Edge 的起始 Vertex 就是该 Vertex；Facet 中存储一条相关 Edge，该 Edge 的所在 Facet 就是该 Facet。

另外，关于 Site，存储了球面上顶点的 x、y、z 三个坐标，做为一个向量，它的长度为球的半径，且不失一般性地定义为 1；关于 Edge，存储了它所在大圆的平面的法向量，通过这一法向量可以确定一个大圆，并规定法向量的方向可以由 Edge 方向根据右手定则确定。

2.2、扫描线和抛物线

要将其应用于球面上，还需要定义球面上的“扫描线”、“抛物线”。

通过比较分析，定义扫描线为一个小圆，该小圆所在的平面垂直与 Z 轴。如图 1 中黄色线条所示。扫描线从由 Z=－1 与球面相交成的小圆开始，移动到 Z=1 与球面相交成的小圆，之后再反向回扫。需要回扫的原因将在下一部分进行

详细说明。如果将球面看成地球，Z=－1 为***北极***，Z=1 为***南极***，则扫描线就是不同 Z 所确定的***纬线***。

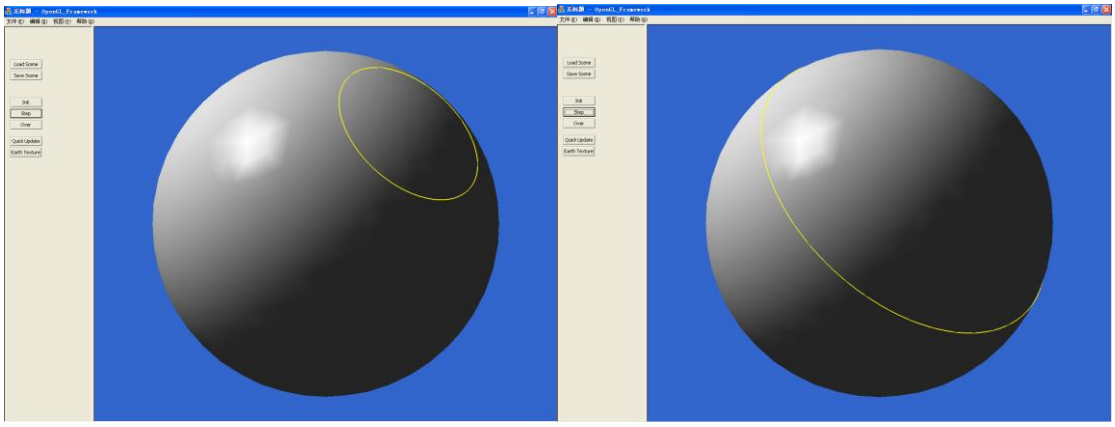


图 1

依照上述扫描线定义，定义与之相应的抛物线为，抛物线上任意一点到对应 Site 的球面距离与该点到扫描线上所有点的最短距离相等。如图 2 红色线条所示。注意到这里定义的抛物线是一条封闭曲线，而且不保证是一条平面曲线。为保证 Voronoi 图的中间结果的 Facet 的封闭性，将 Beach Line 看成一条特殊的 Edge。

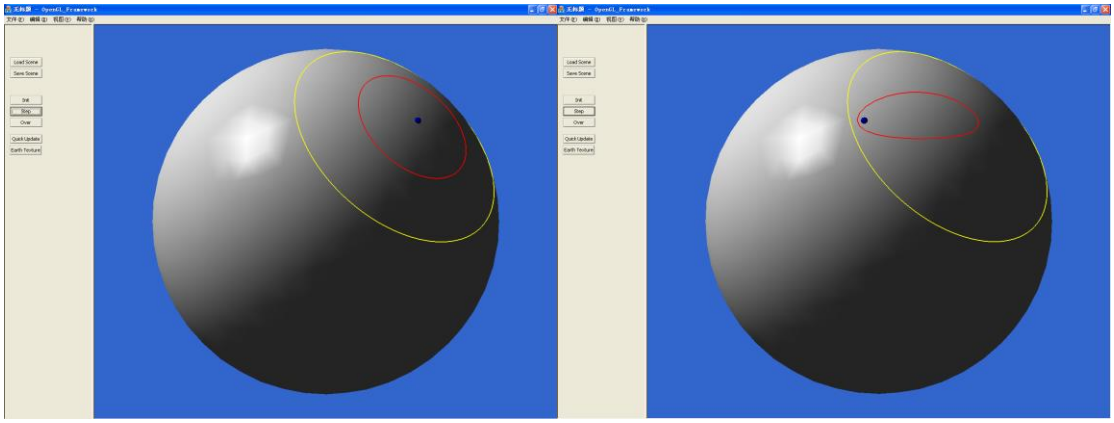


图 2

2.3、Site 事件

有了以上定义，按照 Fortune 的扫描线算法流程，每次推进扫描线，遇到两种事件时停下来处理，即 Site 事件和 Circle 事件。

Site 事件处理扫描线遇到一个 Site 的情况。需要将现有的 Beach Line 进行切分，生成两条新的 Beach Line。该过程向 Voronoi 图中加入了 2 个 Vertex，1 个 Facet，2 条 Edge/Beach Line，如图 3 所示。

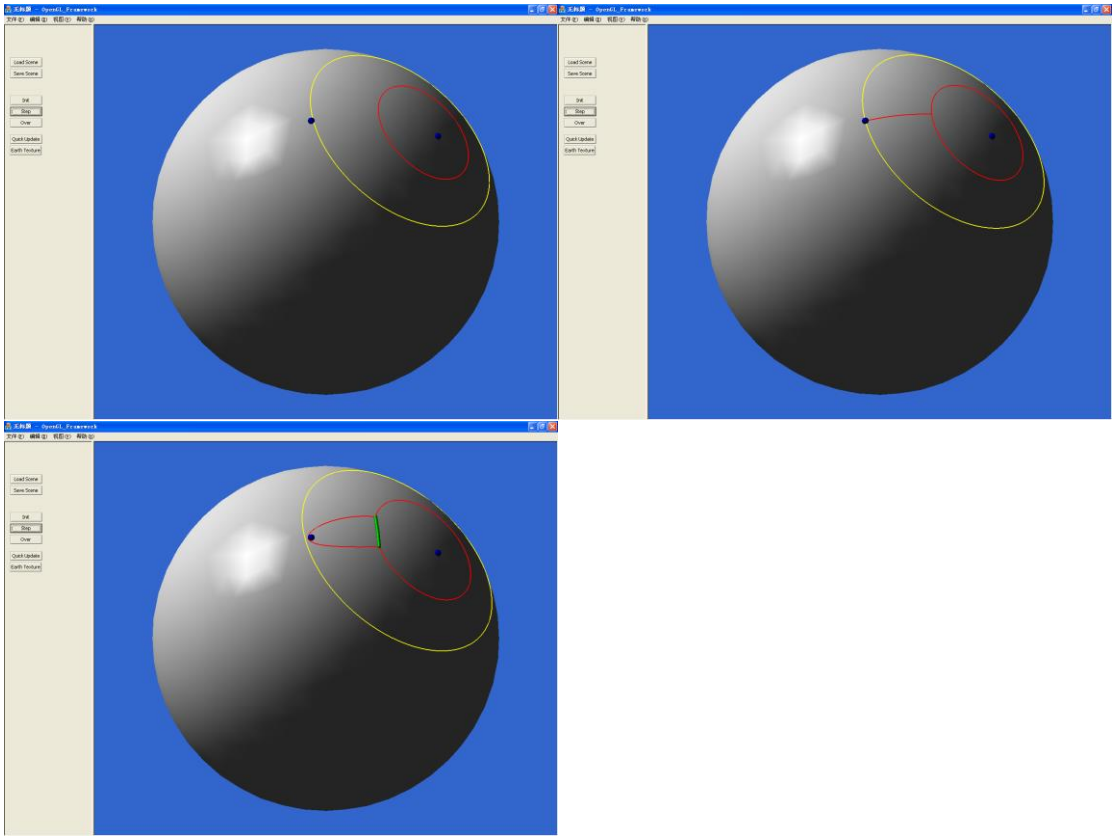


图 3

2.4、Circle 事件

Circle 事件是在 Site 事件和 Circle 事件的处理过程中预测生成的。在当前扫描线处，如果出现 3 个 Site 所共的圆中没有其它 Site 的情况，生成一个 Circle 事件。因为，如果任何一个 Site 都不在这 3 个 Site 所共的圆中时，该圆的圆心处一定有一个 Voronoi 图的 Vertex，这是需要进行一定处理的。

Circle 事件发生时，一定有一条 Beach Line 即将消失。需要将这条 Beach Line 删除掉，并将其起点和终点变成一个点，即小圆的圆心。该过程中 Voronoi 图的 Facet、Vertex、Edge/Beach Line 总数并没有变化，但是它们之间的连接关系发生了变化，而且一条 Beach Line 变成了一条 Edge，如图 4 所示。

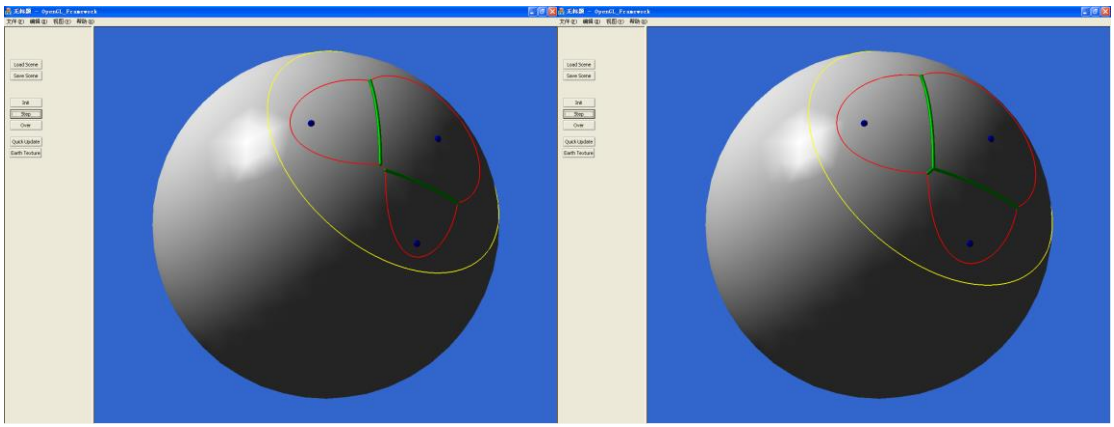


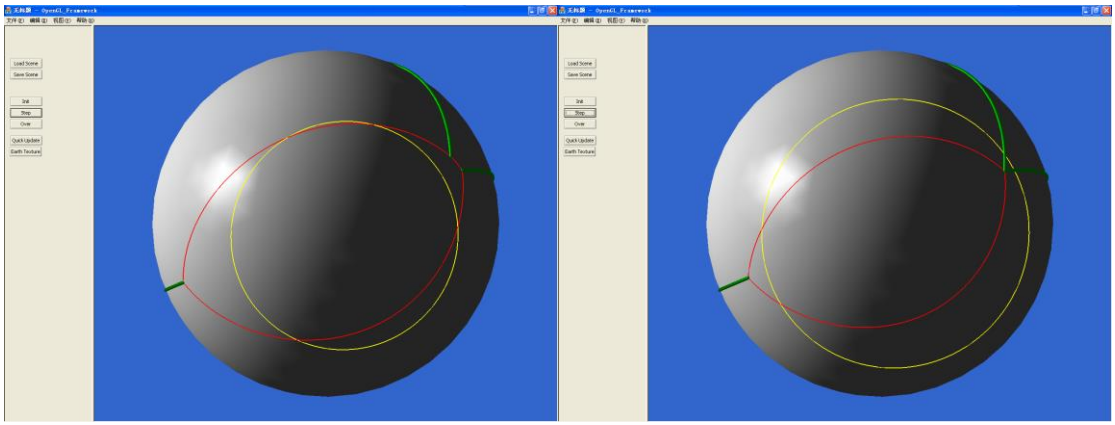
图 4

2.5、封口事件

注意到在扫描线扫到任意位置时，Beach Line 所围成的球面区域的一侧总是不存在已经扫描过的 Site 的。这个区域我们将它定义为一个特殊的 Facet，这个 Facet 没有对应的 Site，以后称之为 Beach Facet 这样做也可以保证扫描线前进的过程中的 Voronoi 图的中间结果的完整性。

与平面的情况不同，球面是一个封闭的曲面，因此在球面上也不存在无穷的射线。由于最终的 Voronoi 图不存在一个 Facet 是没有对应的 Site 的，这个特殊的 Facet 最终一定会消失。处理这个 Facet 消失的事件就是封口事件。

封口事件由 Site 事件和 Circle 事件处理结束后预测产生。产生的条件是 Site 事件和 Circle 事件全部处理完成，而且这时的 Beach Facet 必定只有两条 Beach Line 围成。所要做的就是去掉这两条 Beach Line，去掉相关的两个 Vertex，并去掉 Beach Facet。如图 5 所示。完成封口事件的图就是最终的 Voronoi 图。



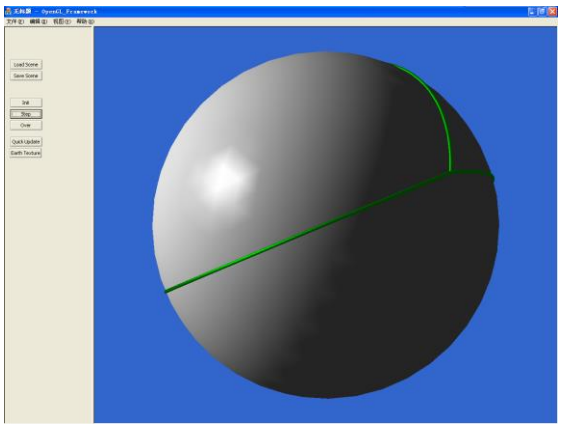


图 5

3、遇到的问题及解决对策

3.1、扫描线及抛物线定义的确定

与平面的情况不同，球面上没有一个坐标能够从无穷远扫过来，扫到无穷远而去的。要在球面上使用扫描线算法，最基本的问题是扫描线如何扫描。这时产生了两种想法：按经线扫描和按纬线扫描。

按经线扫描指，如果将球面当作地球表面，扫描线为一条经线，自西向东从 180° 西扫描至 180° 东。这种做法会产生一个明显的问题，如果扫描线扫过将近一周才扫描到某一个 Site，该 Site 对现有 Voronoi 图将造成相当大的影响。即使扫描线定义为整个大圆而不是半个大圆情况也是一样。另外一种按经线扫描，将扫描线定义为两条经线的并，例如 45° 西和 45° 东的并，从 0° 东西开始扫描，到 180° 东西结束。上两种方案所出现的情况会有所好转，但由于扫描线没有一个统一的方程，使得抛物线定义变得更加复杂。

那么，我们最终放弃了按照经线进行扫描的想法，从而选择按照纬线进行扫描。下一个问题是抛物线如何定义。

抛物线的定义也有不同的选择。第一种方案考虑到球面的 Voronoi 图实际上是这些点在三维空间中的 Voronoi 图与球面的交线，从而按照纬线进行扫描，实际上就是按照 Z 轴坐标进行扫描。那么自然的，抛物线可以定义为，由 Site 和扫描线所在平面所确定的抛物面与球面的交线。但是这样的定义也带来了问题，

刚刚处理过 Site 事件的状态，该 Site 对应的抛物面与球面的交线只有一个点，并不与现有的 Beach Line 相交（这并不与三维扫描时该 Site 对应的抛物面会与三的 Beach Line 相交相矛盾），从而无法很好的推进 Beach Line。

最终选择的方案是将一切距离都在球面上计算，扫描线只是球面上的一个小圆，也就是一条纬线，抛物线是由球面距离相等所确定的曲线。其中，球面上的点到扫描线的距离，是该点到过该点的经线与扫描线的交点的距离。虽然通过简单的计算分析可以得出，这条曲线并不是一个小圆，因此不保证在一个平面上，也就很难想象出这条曲线确切的模样，但是对于一般的情况，它会与所有的经线有且仅有一个交点，从而可以求出它的参数方程。随后，我们也就可以保证，由多段这样的 Beach Line 可以将球面切成两个部分，正如平面上多条抛物线可以将平面切成两个部分一样。特殊情况是当扫描线通过 Site 时，抛物线是从 Site 到北极的一段大圆弧，这也不影响我们的结论。

抛物线的参数方程如下：

$$\begin{cases} x = \cos\theta\cos\varphi \\ y = \sin\theta\cos\varphi \\ z = \sin\varphi \end{cases}$$

θ 为参数，表示抛物线上该点所在经线的角度，φ 由下式确定：

$$\tan\varphi = \frac{\sqrt{1 - z_{scan}^2} - (x_f\cos\theta + y_f\sin\theta)}{z_f - z_{scan}}$$

$$\cos\varphi = 1/(1 + \tan^2\varphi)$$

$$\sin\varphi = \cos\varphi\tan\varphi$$

其中，(x_f, y_f, z_f) 为焦点坐标，也就是 Site 的坐标；z_{scan} 为扫描线的高度。

3.2、扫描线的回扫

刚刚确定了扫描线和抛物线，正在我们为它们很好的性质而高兴时，又发现一个问题。

举例说明，假设有一个 Site 在（0，0，－1），不难想象，按照上面的定义，扫描线沿着纬线扫描时，Beach Line 也是一条纬线在移动，但是当扫描线从北极移动到南极后，Beach Line 仅从北极移动到了赤道（Z=0 与球面交成的大圆）。

这样的情况并不是偶然的，Site 取在任意位置都会发现扫描线扫到南极后，Beach Line 并没有扫过整个球面。那么，就可能出现三个点的 Circle 事件，也就是一条 Beach Line 会消失的事件，直到扫描线扫到南极都没有机会进行处理。

这又是平面和球面进行扫描的一个区别。平面上，只要扫描线向无穷远扫描，Beach Line 一定会随着扫过整个平面。这样，不论 Voronoi 图的 Vertex 在何处，生成它的 Site 事件和 Circle 事件一定有机会进行。因此，相应的，在球面上的扫描线和抛物线的定义也应该满足这一点。

为此，还需要对上面的扫描线定义进行一点特殊调整。当扫描线扫到南极后，再从南极回扫，但回扫时球面上的点到扫描线的距离不再是简单的做经线相交得到，而是过该点的经线方向相反的经线与扫描线的交点与该点的距离，如图 6 所示。这一距离定义也就影响了抛物线的定义，但反而可以使抛物线自然的推进。对于本节开始的例子，当扫描线从南极再次扫回北极时，Beach Line 刚好从赤道扫到南极，扫过了整个球面。这正是我们希望的结果。

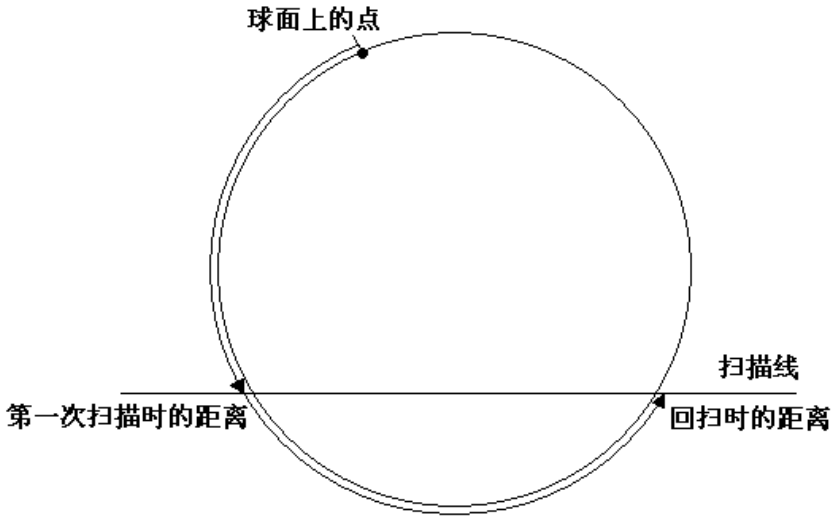


图 6

调整过的抛物线的参数方程大部分不变，需要修改的是将其中 $\tan\varphi$ 的式子修改为：$\tan\varphi = \dfrac{z'_{scan} - (x_f \cos\theta + y_f \sin\theta)}{z_f - z_{scan}}$，其中 $z'_{scan} = \pm\sqrt{1 - z_{scan}^2}$，第一次扫描时取正号，回扫时取负号。当 $z_f = z_{scan}$ 时，第一次扫描时是 Site 到北极较短的大圆弧，回扫时是该大圆其余的一段大圆弧。

3.3、多点共圆的情况

由于 Circle 事件是由三个 Site 共圆产生的，使得由 Circle 事件生成的 Vertex 只有 3 个相关的 Site，也就只有 3 条相关的 Edge。而对于多 Site 共圆的情况，结果应当是一个 Vertex 引出多条 Edge。如何形成这种结构呢？

对于比较一般的情况，会产生两个坐标相同的 Vertex，它们之间有一条长度为 0 的 Edge。因此，只要在算法运行结束后进行一下后处理，去掉所有长度为 0 的 Edge，并维护 DCEL 图的结构就可以了。

也有特殊情况会造成算法的错误，需要进行特殊处理。那就是第一次扫描线遇到一个 Site 时，就同时遇到多个 Site。在平面的情况不会在意这一点，它们之间的 Edge 是平行的。但在球面上，它们之间的 Edge 会交与北极，这就立即需要一个或多个 Circle 事件来处理。为此，对于执行位置相同的两种事件，设置 Circle 事件优先于 Site 事件执行。同时，特殊处理这种情况，令新加入的 Site 能够找到对应相交的 Beach Line，实际上，只要找到与之相邻的两个 Site 对应的 Beach Line 都是可以的。

3.4、长度为 0 的 Edge/Beach Line

在处理 Site 事件时，会产生起点和终点重合的 Edge 和 Beach Line，在处理 Circle 事件之前也会产生这样的 Beach Line，之后会产生这样的 Edge。看起来似乎不需要特殊处理，但在球上，还有其它情况与之类似，整个大圆的 Edge 和唯一的 Beach Line。

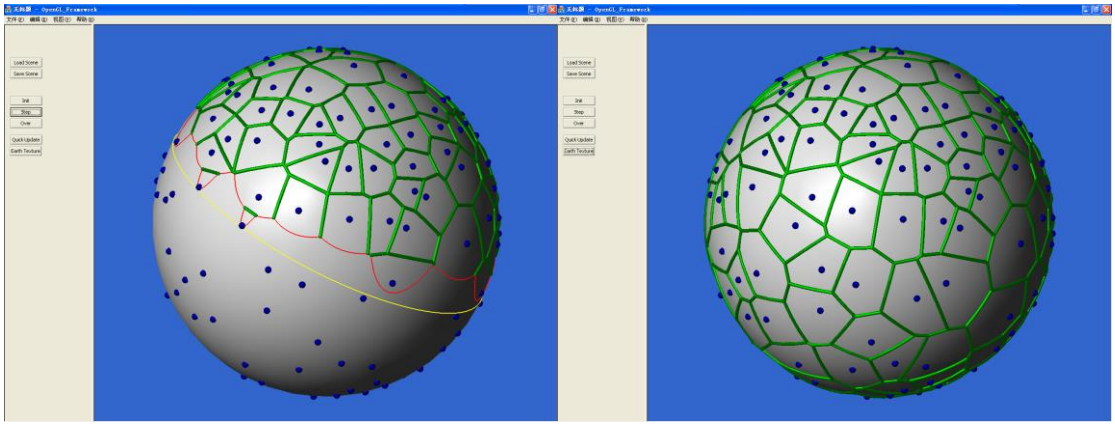
对于整个大圆的 Edge，是只有两个 Site 的情况下产生的。为了绘图方便，我们也给它设定了起点和终点，而这两个点是重合的，如何区别它于边长为 0 的 Edge 呢？为了区别，我们设定大圆的 Edge 的起点和终点指向的是同一个 Vertex，也就是说，只有两个 Site 的情况会得出一个有 2 个 Facet、1 条 Edge、1 个 Vertex 的 Voronoi 图，而不是 2 个 Vertex 或 0 个 Vertex，并在封口事件中做了相应判断和处理。这纯属是为了区别，并没有什么可细究的。

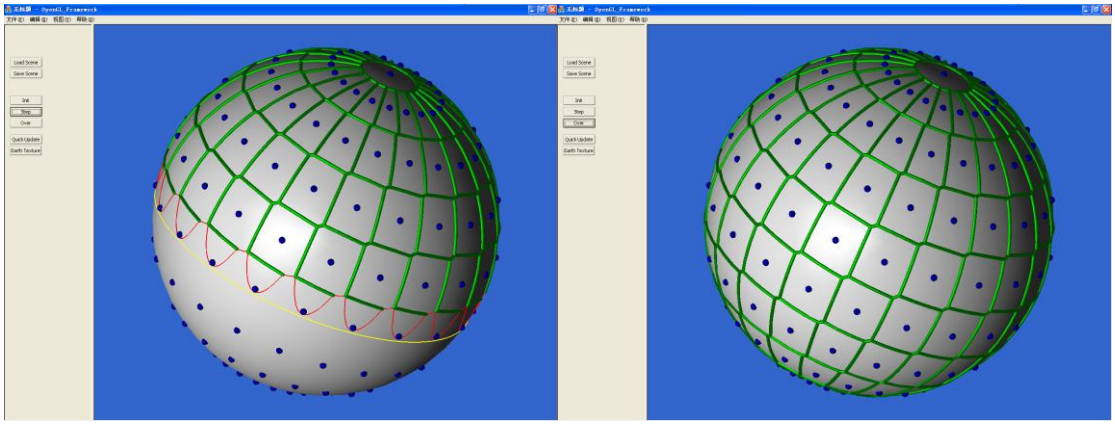
对于唯一的 Beach Line 的情况，是当扫描线刚刚扫过一个 Site，还没有遇到下一个 Site 的情况下产生的。同样为了表示它，我们设定了它的起点和终点是重

合的，如何区别它于 Site 事件生成的 Beach Line，以及由于 Beach Line 推进而即将消失的 Beach Line 呢？区别它们无非有两个用途，一个是不要在 Site 事件求交点的时候混淆，一个是在绘制的时候不要混淆。对于求交点时，只要多判断当前是否只有唯一的 Beach Line 就可以了；对于绘制时，也是同样道理，同时实际上即将消失的 Beach Line 不绘制也不会有影响，那么只要判断当前扫描线是否在其对应 Site 处就可以了。

4、测试结果

对算法进行了大量结点的压力测试，多顶点在同一个小圆上等特殊情况，都可以测试通过。





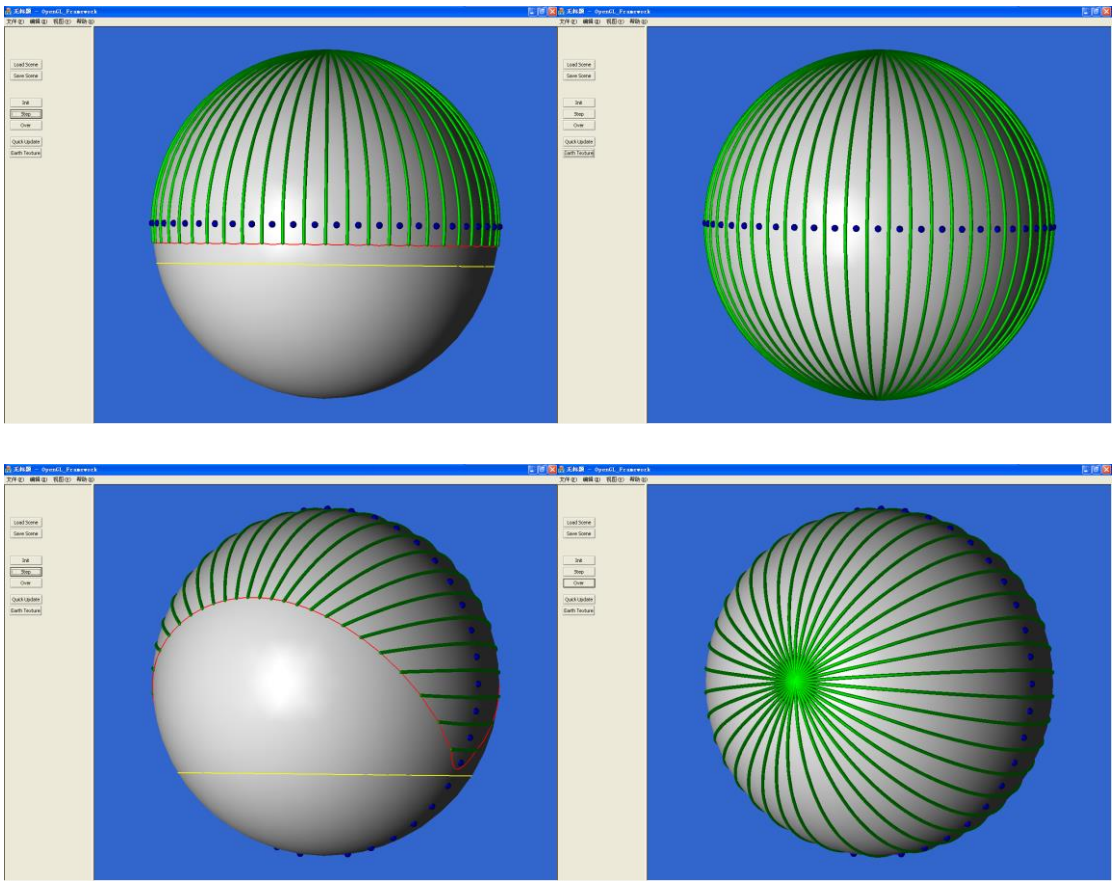


图 7

5、算法局限和待解决问题

由算法中扫描线和抛物线的定义决定，不能有顶点出现在南极，也就是（0，0，1）的位置。否则，这个结点的 Site 事件时，整个球面的所有点到扫描线的距离和到 Site 的距离都相等，使得抛物线变得无意义，从而导致错误。为了避免这种情况的发生，可以对该 Site 的坐标进行一点扰动，也可以旋转坐标系，使得没有结点在（0，0，1）处出现。目前，为了方便，仅仅不允许用户将 Site 设定在（0，0，1）处。

为了方便实现，Site 事件队列和 Circle 事件队列是采用双向循环链表进行组织的，因此，算法并不是 $O(n\log n)$ 的算法，应该是一个 $O(n^2)$ 的算法。要实现 $O(n\log n)$ 的算法，需要使用二叉平衡树对 Circle 事件队列进行组织，但会大大增加处理的复杂性，对于 Site 不多的情况反而可能降低效率，并没有最终实现。

6、参考文献

[1] S. J. Fortune. **A sweepline algorithm for Voronoi diagrams, Algorithmica**, 2:153-174, 1987.

[2] Junhui.Deng. **Computational Geometry Course 2006, Lecture Notes**. Chapter 4, Chapter5.