三维变形作业报告

陈伟业

一、算法思路：

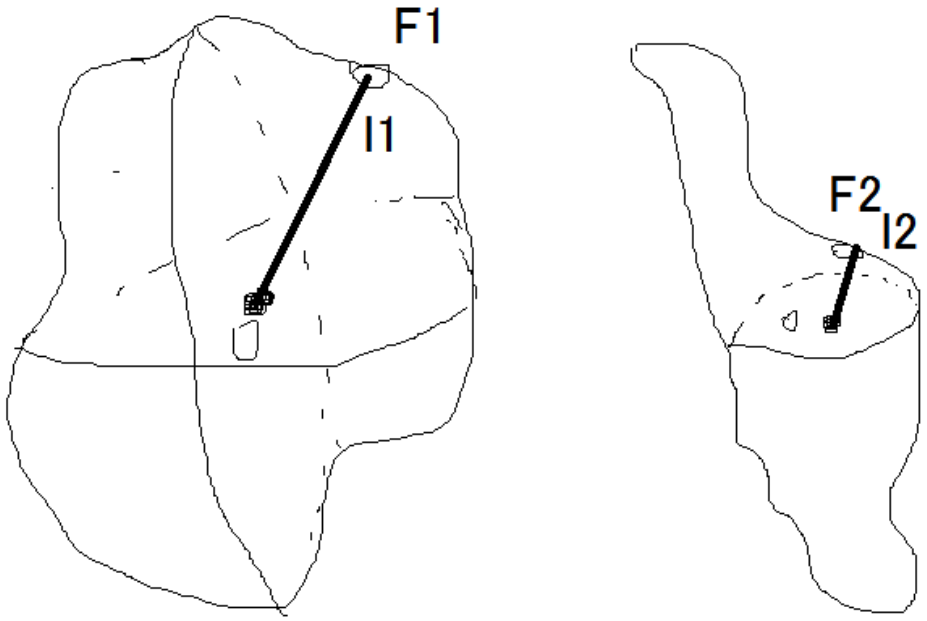
变形映射：将物体 A 的面映射到物体 B 相对中心的的相同位置上。

变形过程：平滑插值变形。

粒子效果：物体周围散射着粒子束，其中粒子束是由许多粒子构成的。

二、算法描述：

变形映射： 将物体的每个面当做一个抽象变形图中的结点（不同于实际顶点），建立每个结点到结点的映射（边）。



Match(F1,F2)=cos(intersect(I1,I2))

物体 A 面集：{FaceA}，物体 B 面集：{FaceB}

首先，在面数量少的物体上增加一些三个点重合的面积为 0 的面，将 A 物体与 B 物体的面的数量补至相同。

对每一对 Fa 和 Fb，建立边，其边权值为 a*|OF1-OF2|-Match(Fa, Fb)，其中 Match 为 Fa 面重心与 A 物体重心的连线 和 Fb 面重心与 B 物体**重心**连线 的夹角的余弦值，a 为加权系数。

随后，在建立的图中使用贪心算法，每次选择权值最小的边加入结果集，在面集中去掉这些边相关的面，进行迭代，直到结果集大小为物体顶点总数，所得结果集即为映射结果。

变形过程：

P(x)=alpha*Src(x)+(1-alpha)*Dst(x);

Where alpha=x*x*(3-2x),x in [0,1]

此即为平滑插值法的定义，其中 P（x）是面 x 当前的位置。

粒子效果：

每个粒子束从某一个粒子，朝一个特定方向发射不同速度、加速度的粒子。物体的周围随机均匀分布着这些粒子束。

三、算法实现：

变形映射：

```
Set FaceMap;
Array MappingResult;
Balance();// 使两个物体面数相同
For Fa in FaceA
   For Fb in FaceB
         FaceMap.insert(Edge(Fa,Fb,Match(Fa,Fb));
FaceMap.sort();
While (FaceMap.size() < number of vertices)
Begin
   Edge e= FaceMap.pop();
```

```
If (e.Fa in FaceA && e.Fb in FaceB)
Begin
       FaceA.erase(e.Fa)
       FaceB.erase(e.Fb)
       mappingResult.add(e)
end
else
    continue
end
```

变形过程：

```
while (currStep < moveStep)
Begin
  double x = currStep*1.0 / moveStep;
  double alpha = 1-x*x*(3 - 2 * x);
  currV1=src1*alpha+dst1*(1-alpha);
  currV2=src2*alpha+dst2*(1-alpha);
   currV3=src3*alpha+dst3*(1-alpha);

  GlTriangle(currV1,currV2,currV3);
  currStep++;
end
```

粒子特效：

```
For loop in MaxLoop
Begin
    GlDraw;
    Update x,y,z
    Update speedx,speedy,speedz;
    If (fade)
    Begin
        Reset Particle
    end
end
```

四、编程环境：

Windows 10 + Visual Studio 2015

语言：C++； Opengl 库：GLUT 3.7

五、程序运行说明：

开始时，在命令行输入obj文件的相对路径，变形路径：1->2。按回车开始绘制。

鼠标左键拖动：改变视角方向。

鼠标右键单击：进行一次变形。

按键q：拉远视角。

按键 w：拉近视角。

六、程序结果分析：

本程序可呈现较为规整的变形动画，虽然由于面到面的映射的实现方式，中间的面碎片化还是可以看得出来，但相对于上一个变形版本，视觉效果已经有了很大的提升。

程序的运行效率不是很高，时间复杂度为 O(F^2)，当计算面数为 7000 的模型的变形映射时，时间往往为 20 秒左右。但在处理 1000 个面以下的模型时，等待几乎不会发生，还在可以接受的程度之内。

粒子特效的实现由于需要关闭深度测试，在物体背面的粒子还是可以被看到，在视觉效果上有所欠佳，还有改进的空间。

总的来说，变形效果虽然不是非常平滑，但在视觉观察上，变形的过程还是比较清晰的，大体上达到了预期的目标。