计算机图形学基础

网格简化 实验报告

陈键飞 2010011291

1. 实验目的

使用网格简化算法简化一个网格。即，将原来面片数较多的三角网格简化成为面片数较少的网格。并且使得模型尽量保持原样。

2. 实验原理

使用了边塌缩的网格简化算法，其原理如图 1。一般来说，每塌缩一条边，减少以它为一边的两个面片。

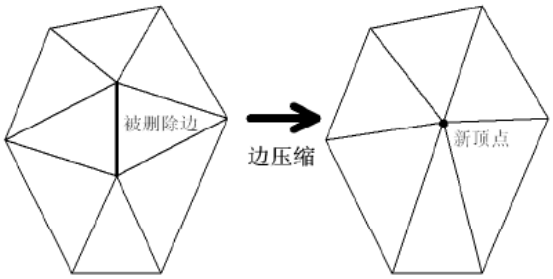

图 1：边塌缩原理

边塌缩方法需要考虑的两个问题是：

- 塌缩哪条边；
- 边塌缩后，原有的两个顶点移动到哪个位置。

本实验使用了[1]中使用的方法。使用一个二次型作为估价函数，每次选取代价最小的边塌缩。此外，还可以解得使代价最小的新位置。

更详细地说，将塌缩后的顶点 v 与原来包含它的面片的距离和的平方作为估价函数。设 v 的齐次坐标为 v，平面 p_i 的齐次坐标为 p_i。则两者的距离就是其点积。

$$\mathrm{Cost}(v) = \sum_{p_i} (v^T p_i)^2 = \sum_{p_i} v^T Q_i v = v^T Q v \qquad (1)$$

其中Q_i=$p_i {p_i}^T$，$Q = \sum Q_i$。这样，将v_1, v_2移到v的代价就是$v^T(Q(v_1) + Q(v_2))v$。

如果对代价函数求导，会发现使其取得最小的v是方程组

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} v = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (2)$$

的解。如果这个方程组无解，不妨就单纯地取$v = (v_1 + v_2)/2$。

3. 实验内容

算法

本实验对比了三种网格化简方法的效果：

1. 以边的长度作为估价函数；
2. 以(1)式作为估价函数，取$v = (v_1 + v_2)/2$ 作为新的顶点。
3. 以(1)式作为估价函数，取(2)式作为新的顶点。

可视化

本实验开发的工具支持三种使用方式：

1. 以动画的形式对比原型和 3 种方法的效果（图 3）。
2. 指定目标面片数，在应用程序中查看模型（图 2）。
3. 指定目标面片数，导出 obj 文件，使用其他软件查看模型（图 2）。

（a） (b)

图 2：使用 OpenGL 渲染的模型(a)，使用 3Ds max 查看生成的模型(b)

4. 实验效果

算法对比

对比三种方法的简化模型如图 3。同预料中一致，方法 1 因为优先简化短边，丢失了高频的细节特征；方法 2 保留了细节，但面片的位置有一定误差。方法 3 与原模型最为接近。

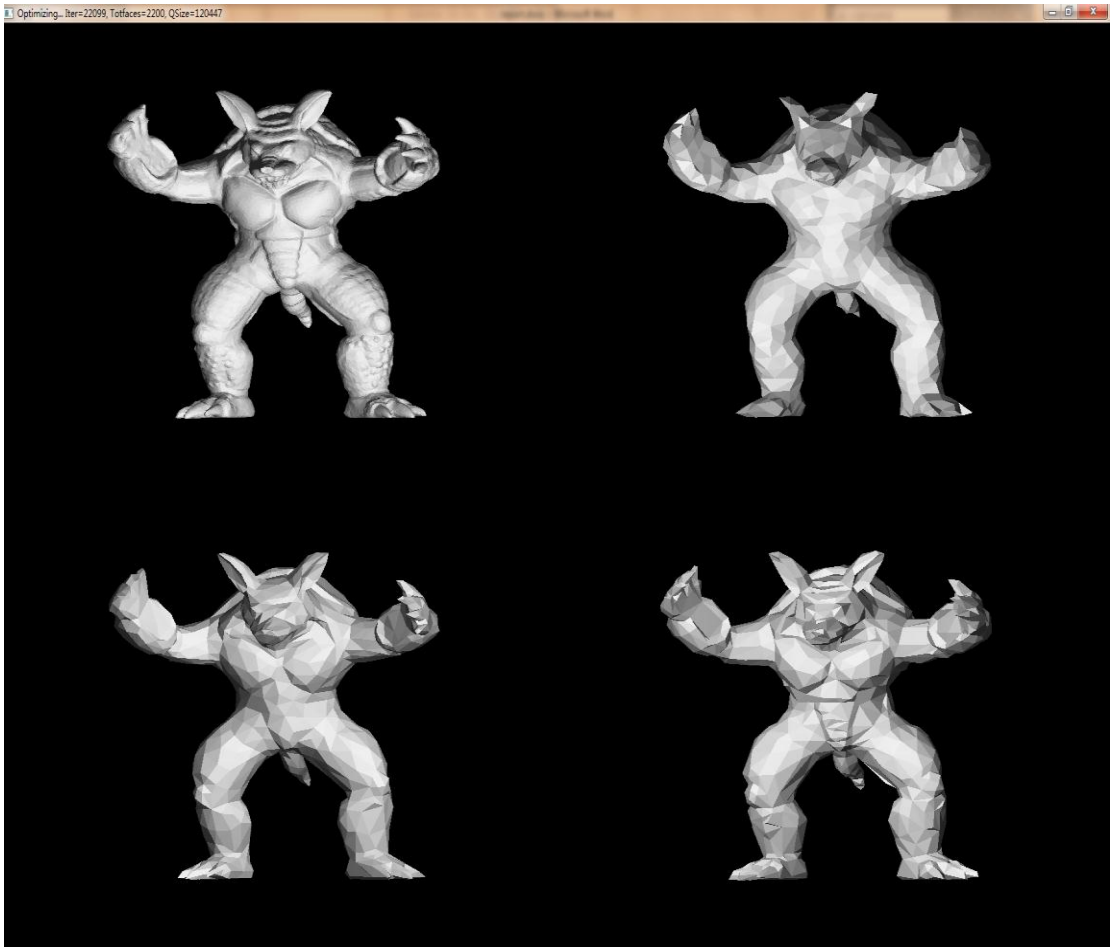

图 3：将 46398 个面片的 Arma 模型简化到 2200 个面片。左上、右上、左下、右下分别为为原模型、方法 1、方法 2、方法 3

面片数对模型质量的影响

不同面片数的效果对比如图 4、图 5。可以发现在这个大小下 20%的面片数与原模型除了光滑程度差异不大。如果查看缩小后的模型，1%的面片数仍然可以看清物体的大致形状。

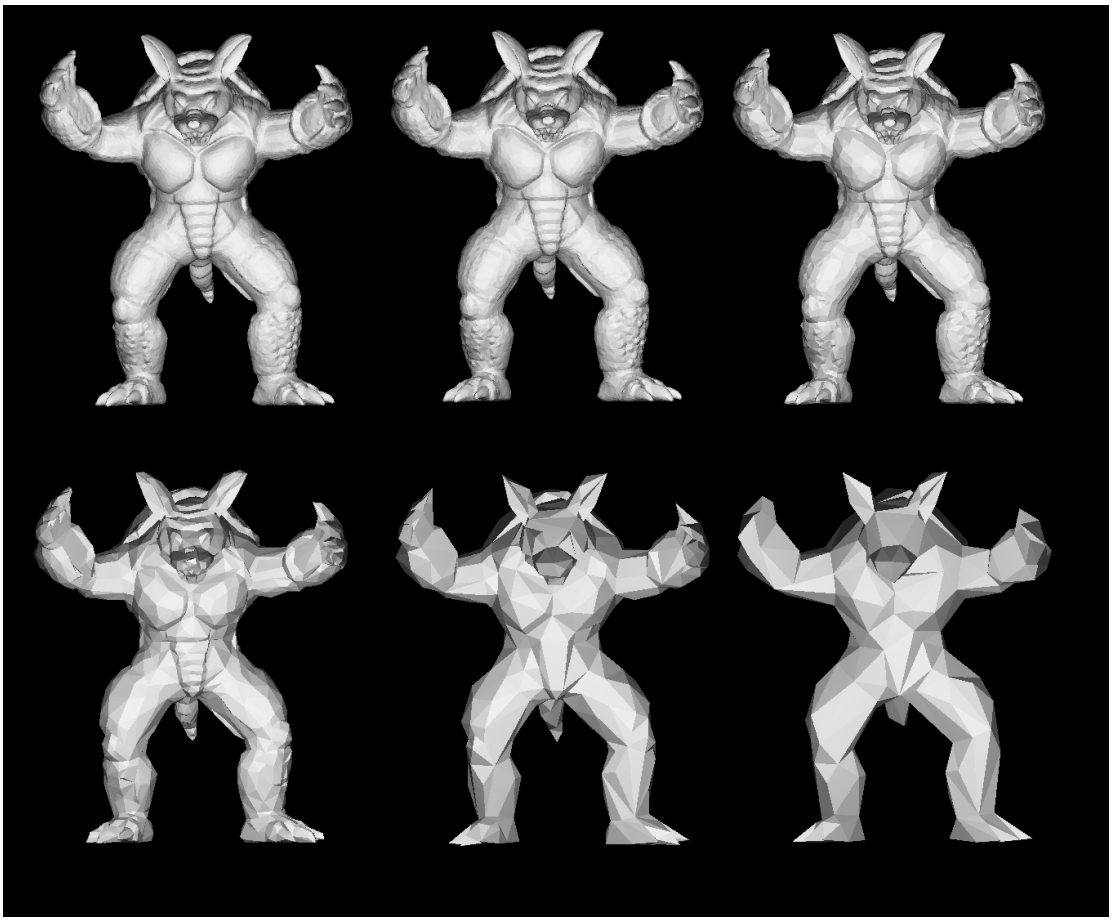

图 4：Arma 模型，依次为 100%、50%、20%、7%、2%、1%的面片数，最后一个模型共 464 个面片。

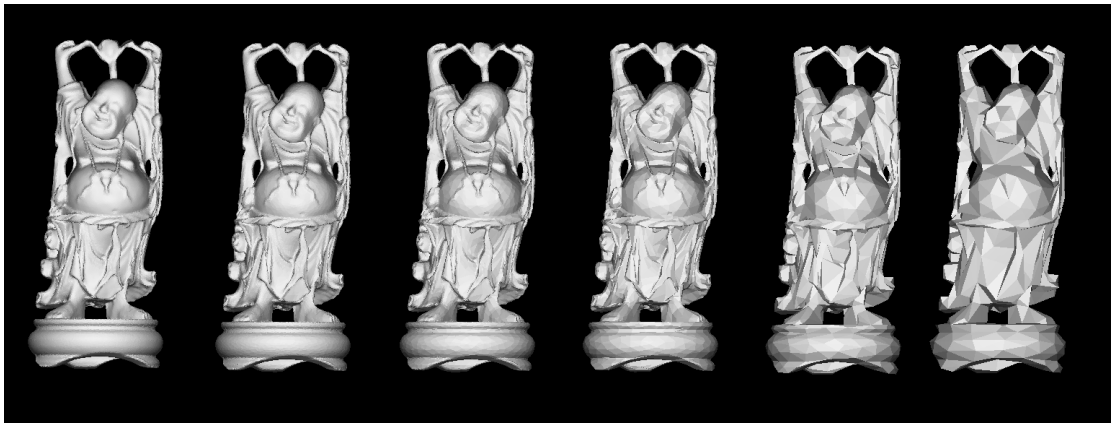

图 5：Buddha 模型，依次为 100%、50%、20%、7%、2%、1%的面片数，最后一个模型共 1230 个面片。

图 6：缩小后的 Buddha 模型，六个模型差别不大

5. 实验总结

本次实验基本完成了预定目标。网格简化是强有力的解决存储空间和计算资源的工具。1%的面片在远距离看与原图形差别并不大，在渲染时可以大大节省计算开销。

6. 参考文献

[1] Michael Garland and Paul S. Heckbert. 1997. Surface simplification using quadric error metrics. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH '97).