学号	姓名	论文规范性（10）	问题分析与调研（30）	方案创新性（20）	实验结果分析与讨论（40）	结课论文总成绩（100）
21301119	陈锦璇	6	25	16	33	80

Loop 细分算法

摘要

本论文探究了 Loop 细分（Loop Subdivision）算法，该算法是一种广泛应用于计算机图形学领域的细分曲面技术。Loop 细分算法通过递归地细分三角形网格，使得网格逐渐趋于光滑曲面，最终生成具有更高细节的曲面表示。本文首先回顾了细分曲面的历史和发展背景，之后介绍了 Loop 细分的数学基础，包括顶点规则和边规则，详细解析了每个细分步骤的数学原理和几何意义。最后，通过对 Loop 细分算法的具体实现，本文讨论了其基本原理、操作步骤以及在几何处理中的应用。

在算法实现部分，本文对 Loop 细分算法的实现步骤进行了详细描述，包括顶点的插值计算、边的分割方法以及细分后网格的重构，以及算法的时间复杂度和空间复杂度。此外，我们对比了 Loop 细分与其他细分算法，如 Catmull-Clark 细分和 Doo-Sabin 细分，指出了它们各自的优缺点和适用场景。

关键词：Loop 细分，细分曲面，计算机图形学，三角形网格，曲面表示，算法优化，应用场景

目录

1 引言

曲面细分是一种将低细节度（Low-Poly）模型转化为高细节度（High-Poly）模型的过程。原始模型由少量多边形组成，这些多边形往往过于简略，难以表现出复杂的表面细节。通过曲面细分技术，我们可以根据预先定义的规则或程序算法，在不增加过多内存负担的前提下，动态地生成更细致的几何结构，使得模型表面更加平滑，线条过渡自然，同时保留原始模型的整体形状特征。

Loop 细分（Loop Subdivision）算法自从其提出以来，已经成为计算机图形学领域中一种重要的曲面细分技术。由 Charles Loop 在 1987 年提出，该算法通过递归细分三角形网格，使得初始网格逐渐趋向于光滑的曲面。

Loop 细分算法的理论基础源于细分曲面理论。[1]Loop 在其博士论文中提出了一种基于三角形网格的细分方法，该方法通过递归细分和插值生成光滑曲面。其核心思想是通过顶点和边的规则来细分网格，使得每次细分后新的顶点位置能够保持曲面的连续性和光滑性。

在游戏开发中，曲面细分技术被广泛应用以提高场景和角色的视觉质量，尤其是在开放世界游戏和高端图形模拟中，它能够在保证帧率的同时呈现丰富而真实的环境纹理。此外，在电影 CGI 制作、建筑设计可视化以及工业设计等领域，曲面细分也扮演着至关重要的角色，极大地提高了作品的艺术表现力和真实感。

2 相关工作介绍

细分曲面的概念可以追溯到 20 世纪 70 年代。最早的细分算法之一是由[2]George Chaikin 在 1974 年提出的 Chaikin 算法。该算法用于生成 Bezier 曲线，通过递归地细分线段生成光滑曲线。Chaikin 算法的成功激发了研究者对更高维度（如曲面）细分技术的兴趣。

1978 年，[3]Edwin Catmull 和 Jim Clark 提出了 Catmull-Clark 细分算法，这是细分曲面历史上的一个重要里程碑。该算法基于四边形网格，通过递归细分生成光滑曲面。Catmull-Clark 细分算法的特点是每次细分生成新的顶点，并通过特定的规则重新计算顶点位置，从而保持曲面的连续性和光滑性。

几乎在同一时间，[4]Daniel Doo 和 Malcolm Sabin 提出了 Doo-Sabin 细分算法。该算法类似于 Catmull-Clark 细分，但其操作对象是多边形网格，尤其是三角形网格和四边形网格。Doo-Sabin 算法通过对每个多边形面进行细分，生成新的顶点和面，从而实现曲面的光滑化。

1987 年，[5]Charles Loop 提出了一种专门针对三角形网格的细分算法，即 Loop 细分算法。该算法通过递归细分三角形网格，使得网格逐渐趋于光滑曲面。Loop 细分的核心思想是通过顶点规则和边规则来计算新顶点的位置，从而保持曲面的光滑性。

进入 20 世纪 90 年代，细分曲面技术得到了广泛的关注和应用。计算机图形学领域的研究者们进一步优化和扩展了细分算法，使其能够处理更复杂的几何形状和拓扑结构。例如，[6]Butterfly 细分算法和[7]√3 细分算法都是这一时期的重要成果。

随着计算能力的提升和应用需求的增加，细分曲面技术在 21 世纪得到了进一步的发展。研究者们致力于提高算法的效率、鲁棒性和适用性。例如，[8]自适应细分技术和[9]并行计算方法的引入，使得细分算法能够处理更大规模的网格和更复杂的应用场景。

本文将探究 loop 细分的方法。

3 方法描述

1. 初始网格
开始时，需要一个初始的三角形网格。这些三角形可以是任意的拓扑结构，但必须是三角形。
2. 计算新顶点的位置
a. 顶点规则（Vertex Rule）
对每个现有顶点计算新顶点位置。新顶点的位置由现有顶点和其邻接顶点的位置加权平均计算。对于一个内点（非边界点），其位置计算公式为：

$$P'_i = \frac{1-n}{\beta} \cdot P_i + \beta \sum_{j=1}^{n} P_j$$

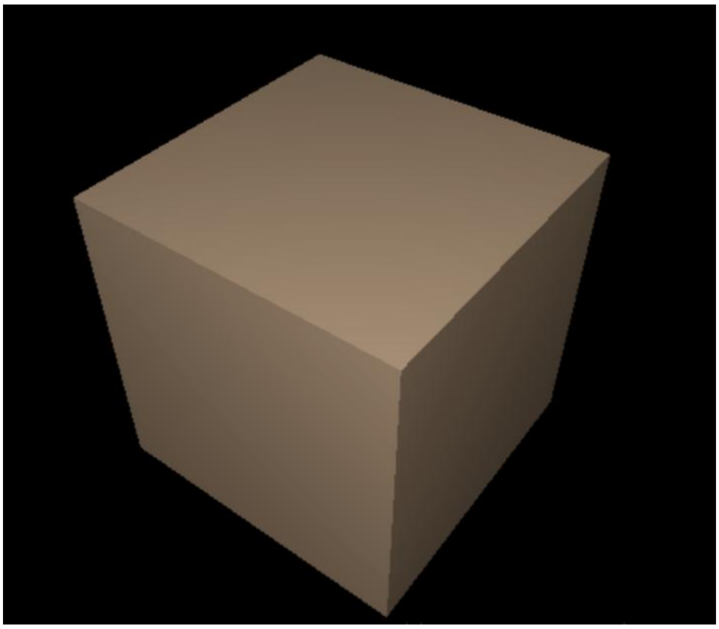
其中：
P'_i是新顶点的位置
P_i是旧顶点的位置
P_j是与旧顶点相邻的顶点的位置
n 是与旧顶点相邻的顶点数
β 是一个权重因子
b. 边规则（Edge Rule）
对每条边计算新顶点位置。新的边顶点位置是由边两端点及其相邻三角形的对角顶点的位置加权平均计算。公式为：
E' =3/8 (P_1 + P_2) + 1/8 (Q_1 + Q_2)
其中：
E' 是新边顶点的位置
P_1 和 P_2 是边的两个端点
Q_1 和 Q_2 是边相邻三角形的对角顶点
3. 生成新的三角形
通过新顶点的位置，生成新的三角形。每个旧三角形被细分为四个新的三角形，具体步骤如下：
- 每条旧三角形的边生成一个新的边顶点。
- 每个旧三角形的顶点通过顶点规则生成一个新顶点。
- 连接新的顶点和边顶点，形成新的三角形。
4. 递归细分
对生成的新网格重复上述步骤，直到达到所需的细分层次或曲面光滑度。

4 实验设置

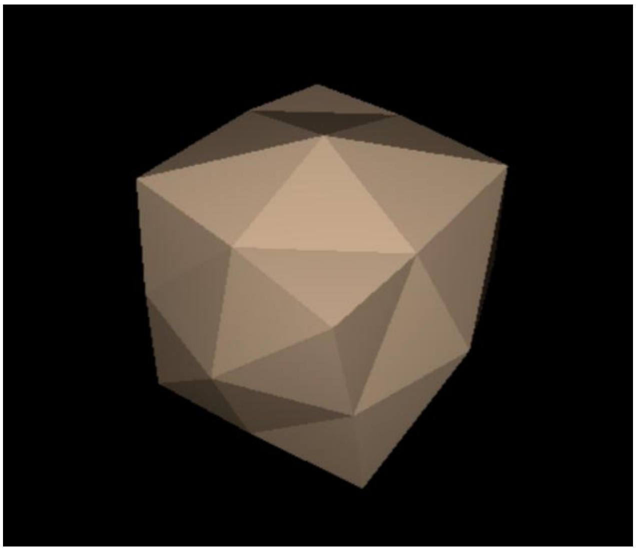
实验环境：Windows 11、C++、OpenGL 4.3+环境。

5 实验结果与分析

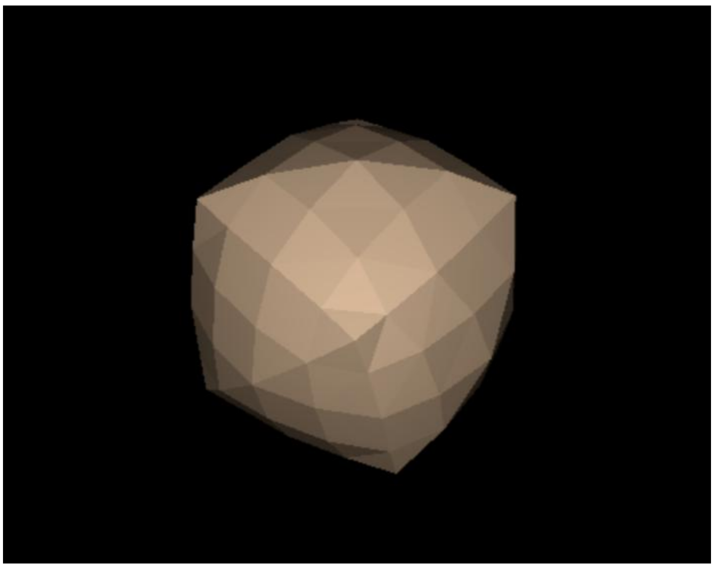
5.1 实现效果
（1）立方体原始模型



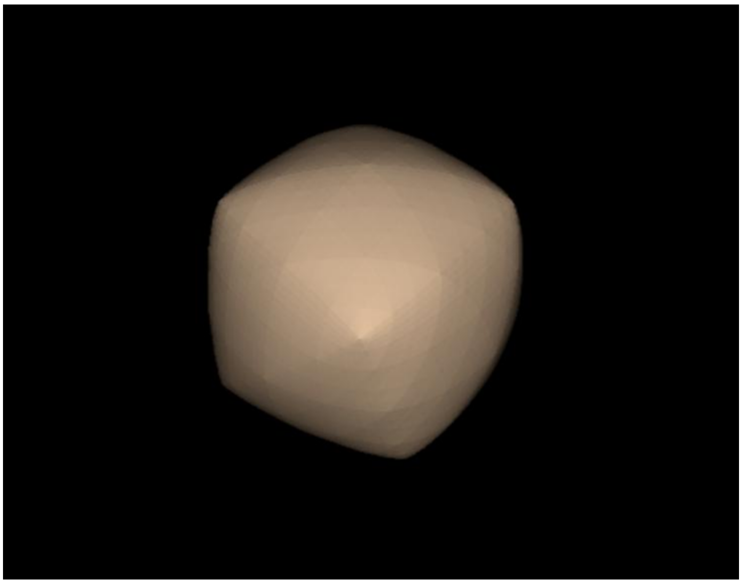
（2）细分一次效果



（3）细分 2 次效果

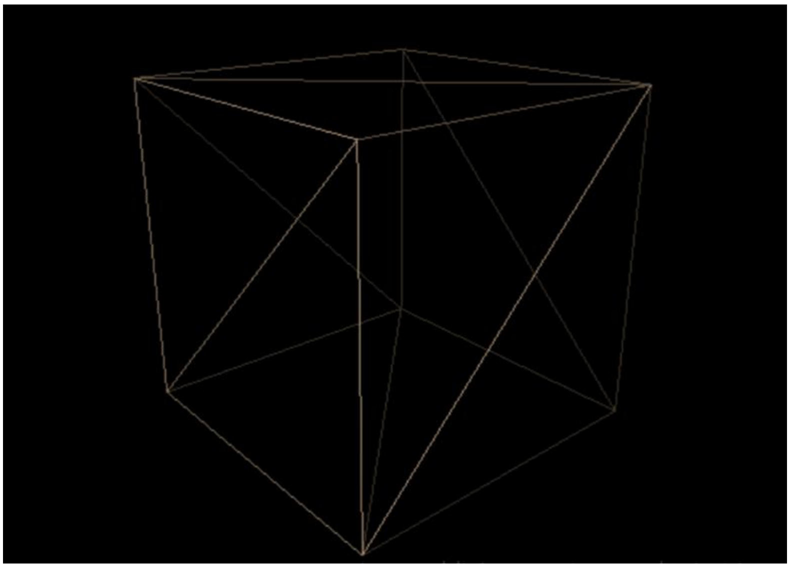


（4）细分 5 次效果

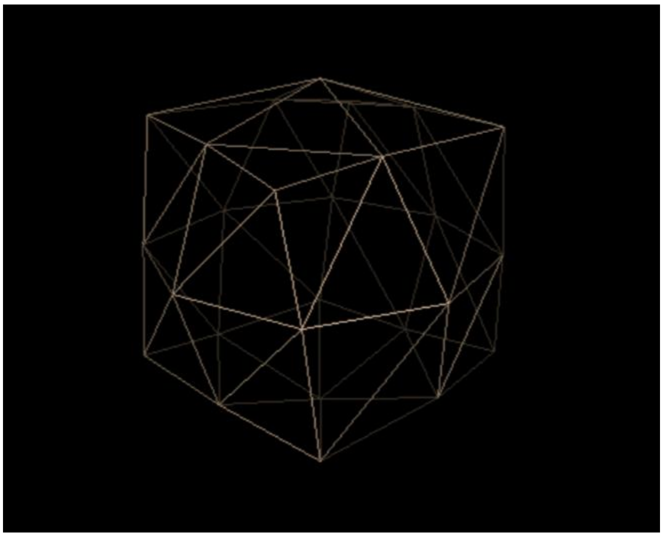


5.2 网格化的效果

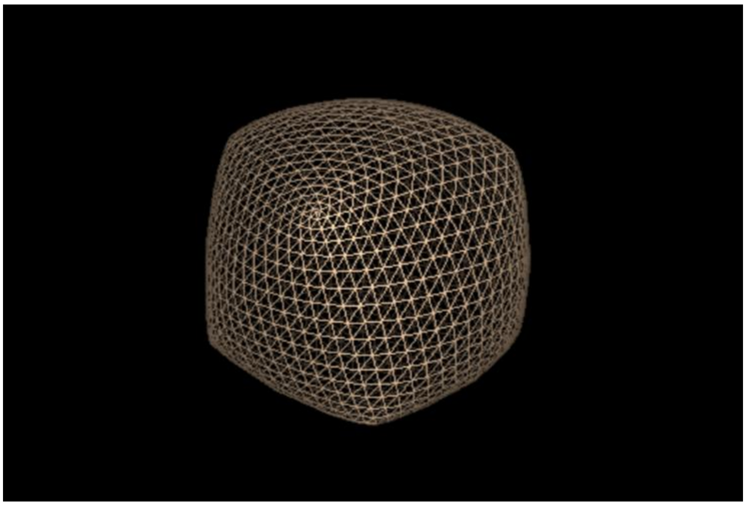
（1）原始模型



（2）细分一次效果



（2）细分五次效果



5.3 时间复杂度和空间复杂度计算

（1）时间复杂度

Loop 细分算法的时间复杂度主要取决于每次细分操作所需的计算量，以及细分操作执行的次数。

A. 顶点规则计算：对于每个顶点，算法需要计算与其相邻的顶点的加权平均位置。假设一个网格有 V 个顶点，每个顶点平均有 k 个邻居，那么计算所有顶点的新位置需要 O(V*k) 的时间。由于 k 是一个常数，可以将这部分的时间复杂度简化为 O(V)。

B. 边规则计算：对于每条边，算法需要计算新顶点的位置。假设网格有 E 条边，由于每个顶点都与多条边相连，因此 E 通常与 V 成正比。计算所有边的新顶点需要 \(O(E) \) 的时间，与 O(V) 相同。

C. 生成新的三角形：每次细分操作会将每个三角形细分成四个新的三角形。假设初始网格有 F 个三角形，那么细分后的新网格会有 4F 个三角形。生成新的三角形需要处理所有的边和顶点，因此这部分的时间复杂度也是 O(V)。

综上所述，单次细分操作的时间复杂度为 O(V)。

由于细分操作是递归进行的，每次细分会增加网格中的顶点数和三角形数。假设初始网格有 V_0 个顶点和 F_0 个三角形，经过 n 次细分后，顶点数和三角形数将会显著增加。通常情况下，每次细分顶点数和三角形数大约会增加一个常数因子。因此，总体时间复杂度

可以表示为： O(V_0 *4^n)

（2）空间复杂度

Loop 细分算法的空间复杂度取决于存储网格顶点、边和三角形所需的空间。

A. 顶点：初始网格有 V_0 个顶点，经过 n 次细分后，顶点数大约会增加到 V_0*4^n。

B. 边：初始网格有 E_0 条边，经过 n 次细分后，边数大约会增加到 E_0*(4^n)。

C. 三角形：初始网格有 F_0 个三角形，经过 n 次细分后，三角形数大约会增加到 F_0*4^n。

综上所述，经过 n 次细分后，所需的空间大约为：O(4^n * (V_0 + E_0 + F_0))

由于 E_0 和 F_0 与 V_0 是线性相关的，可以进一步简化为：O(4^n * V_0)

5.4 Loop 细分、Catmull-Clark 细分和 Doo-Sabin 细分的对比

特性	Loop 细分	Catmull-Clark 细分	Doo-Sabin 细分
适用网格类型	三角形网格	四边形网格（任意多边形网格）	任意多边形网格
细分规则	顶点规则、边规则	面规则、边规则、顶点规则	面规则、边规则、顶点规则
生成新面	每个三角形细分成四个三角形	每个面细分成四个新面	每个旧顶点、边和面生成新面
曲面光滑度	高	非常高	高
算法复杂度	低	高	中等
应用场景	游戏中的动态网格处理	高质量动画和电影模型	处理任意拓扑结构的多边形网格

6 结论

本实验探究了 Loop 细分（Loop Subdivision）算法，并对其在计算机图形学中的应用进行了详细分析。通过实验，得出以下结论：

1. 细分效果：Loop 细分算法能够有效地递归细分三角形网格，使其逐渐趋于光滑曲面，生成高细节的曲面表示。这使得该算法特别适用于需要处理动态网格的应用场景，如计算机游戏和实时渲染。

2. 算法实现：在实现过程中，我们详细描述了顶点插值计算、边的分割方法以及细分后网格的重构步骤。实验表明，Loop 细分算法具有较低的时间复杂度和空间复杂度，适合高效处理大规模三角形网格。

3. 比较分析：通过与 Catmull-Clark 细分和 Doo-Sabin 细分的对比，实验指出：

Loop 细分适用于任意三角形网格，生成的曲面连续且光滑，计算效率高。

Catmull-Clark 细分适用于四边形网格（也能处理任意多边形网格），生成的曲面质量更高，适用于高质量动画和电影模型，但算法复杂度较高。

Doo-Sabin 细分适用于任意多边形网格，具有较高的拓扑结构处理灵活性，但生成的曲面相对不如 Catmull-Clark 细分光滑，适合处理需要任意拓扑结构的多边形网格。

综上所述，Loop 细分算法因其高效性和广泛适用性，在需要快速处理和渲染的应用中表现出色。尽管与其他细分算法相比，其生成的曲面质量可能略逊一筹，但在实际应用中，Loop 细分仍然是一种具有重要价值的细分曲面技术。

参考文献

[1] Loop, C. “Smooth Subdivision Surfaces Based on Triangles.” Master’s thesis, University of Utah, 1987.

[2] Chaikin, G. "An Algorithm for High-Speed Curve Generation." Computer Graphics and Image Processing, 3(4), 346-349, 1974.

[3] Catmull, E., & Clark, J. "Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes." Computer-Aided Design, 10(6), 350-355, 1978.

[4] Doo, D., & Sabin, M. "Behaviour of Recursive Division Surfaces Near Extraordinary Points." Computer-Aided Design, 10(6), 356-360, 1978.

[5] Loop, C. "Smooth Subdivision Surfaces Based on Triangles." Master's thesis, University of Utah, 1987.

[6] Dyn, N., Levin, D., & Gregory, J. A. "A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control." ACM Transactions on Graphics (TOG), 9(2), 160-169, 1990.

[7] Kobbelt, L. " √3-Subdivision." ACM Transactions on Graphics (TOG), 16(4), 422-439, 1997.

[8] Zorin, D., & Schröder, P. "A Unified Framework for Primal/Dual Quadrilateral Subdivision Schemes." Computer Graphics Forum, 17(2), 1998.

[9] Burkhart, A., & Hamann, B. "Level-of-Detail Generation of Subdivision Surfaces." The Visual Computer, 18(1), 2002.