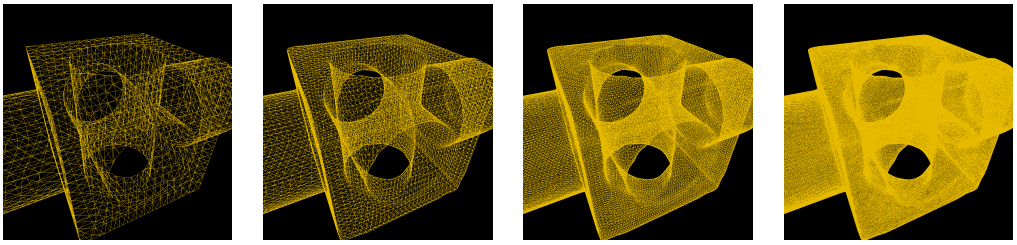
Lab2-Report

王想 2100013146

1 Task 1: Loop Mesh Subdivision

在每一次迭代中进行如下操作: 首先建立 DCEL 查询链表并判断是否建立成功，之后遍历每一个三角面片，对于该三角面片的三个顶点，根据邻居顶点的位置与该顶点原位置的加权和，分别求出这三个点的新的坐标。相应的公式为 $pos\_new = (1 - n * u) * pos\_old + u * \sum_{i=1}^{n} old\_pos\_neighbor\_i$, 当邻居的个数 $n == 3$ 时取 $u = \frac{3}{16}$, 否则取 $u = \frac{3}{8n}$。对于该三角面片的三条边，分别产生新的顶点，新的顶点坐标取决于包含该边的两个三角面片的这 4 个顶点，相应的公式为 $pos = \frac{1}{8}(pos\_1 + pos\_2) + \frac{3}{8}(pos\_3 + pos\_4)$, pos_1 与 pos_2 分别是与该边相对的两个顶点的原始坐标，pos_3 与 pos_4 分别是该边上的两个顶点的原始坐标。最后，按照逆时针的顺序依次把得到的新的 4 个三角面片的顶点索引放入 mesh 中的 Indices 中。需要注意的是，这里的索引必须是逆时针的顺序，并且需要对顶点进行判重，如果一个顶点已经出现过，放入的索引必须与先前放入时的索引相同，否则在下一轮迭代中建立 DCEL 链表会出错。并且，每一次迭代都需要清空原有的 Indices，即清除原先连接的边。



(a) **origin** (b) **iterations = 1** (c) **terations = 2** (d) **terations = 3**

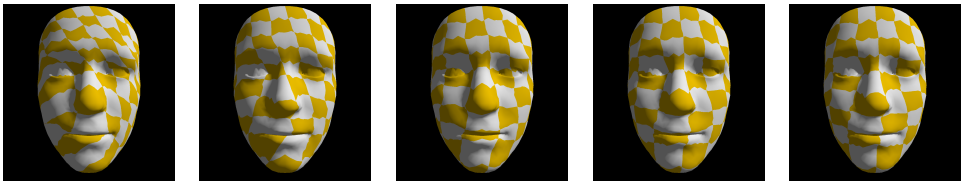
图 1: **Mesh Subdivision**

2 Task 2: Spring-Mass Mesh Parameterization

首先建立 DCEL 查询链表并判断是否建立成功，之后遍历每一个顶点，判断是否为边界点，如果是边界点，将其索引存入 $boundary$ 便于后续查找，并将对应的 TexCoord 初始化为 $(0.5 * normalized\_pos.x + 0.5, 0.5 * normalized\_pos.y + 0.5)$，即初始化为圆形边界。为了保证所有点的 TexCoord 坐标范围均是 $[0, 1] \times [0, 1]$，这里圆的圆心坐标是 $(0.5, 0.5)$，半径为 0.5。如果不是边界点，其 TexCoord 初始化为 $(0, 0)$。接着，构建矩阵方程 $\mathrm{AX} = \overline{\mathrm{X}}$。这里

$$A_{ij} = \begin{cases} 1, if\ i == j \\ -\lambda_{ij}, else\ if\ p_j\ is\ neighbor\ of\ p_i \\ 0, otherwise \end{cases}$$

X_i 是点 i 的 (u,v) 坐标，$\overline{X_i} = \sum_{j=1}^{k} \lambda_{ij} * p_j$, p_j 是点 i 的邻居中的边界点的 (u,v) 坐标，依据 $tutorial$，这里 $\lambda_{ij} = \frac{1}{n}$。最后，使用 Jacobi 迭代求解矩阵方程 $\mathrm{AU} = \overline{\mathrm{U}}$ 和 $\mathrm{AV} = \overline{\mathrm{V}}$, 迭代足够多步 (大概 800 步到 900 步) 之后坐标收敛，点 i 的 TexCoord 即为 (U_i, V_i)。

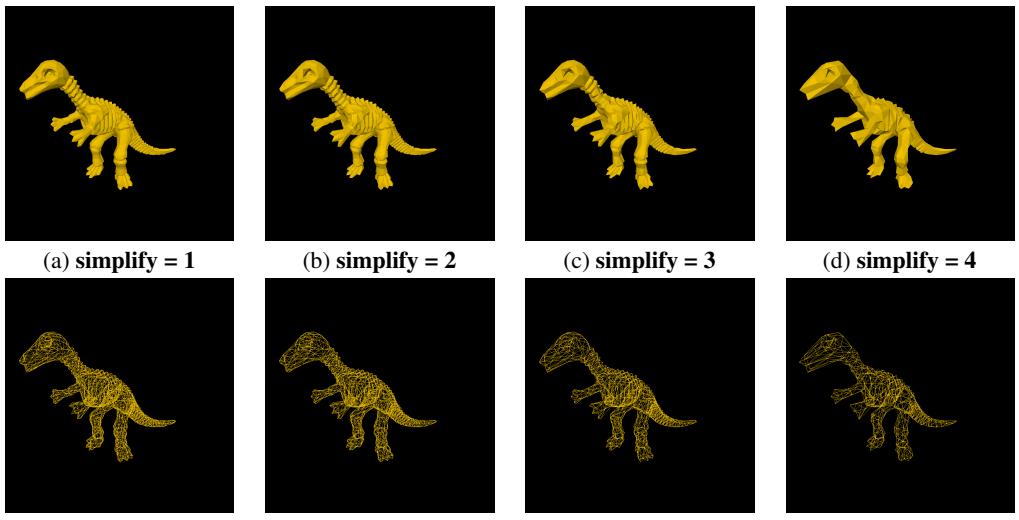


(a) **iterations = 200** (b) **iter = 400** (c) **iter = 600** (d) **iter = 800** (e) **iter = 1000**

图 2: **Mesh Parameterization**

3 Task 3: Mesh Simplification

首先建立 DCEL 查询链表并判断是否建立成功，之后遍历每一个顶点，对每一个顶点求解相应的二次代价矩阵，$mathbfQ = \sum_{i=1}^{n} p_i p_i^T$, 这里 $p_i = (a, b, c, d)^T$, 对应该顶点所在的面 i 的平面方程 $ax + by + cz + d = 0$, 并且约束 $a^2 + b^2 + c^2 = 1$。与此同时，选择所有与该顶点的邻居组成顶点对，存入 $pairs$ 便于后续查找。若采用距离约束，则还需要找到距离该点距离小于 $valid_pair_threshold$ 的顶点，组成顶点对存入 $pairs$。考虑到可能产生重复，在存入之前进行判重，避免重复计算。接着，对于每一对顶点对，算出将他们合并的代价 $error = v^T(\mathrm{Q}_1 + \mathrm{Q}_2)v$，$\mathrm{Q}_1$ 与 Q_2 分别是顶点 v1 与 v2 的二次代价矩阵，v 是使得代价最小化的顶点位置，通常由 $(\mathrm{Q}_1 + \mathrm{Q}_2)^{-1}$ 的第四列给出，若不可逆则选取 v1 或 v2 或二者中点。并存入 $errors$，顺序与顶点对 $pairs$ 一一对应。最后，迭代地选取最小合并代价对应的顶点对进行合并 (如果代价一样就选择顶点对距离最小的)，并且更新所有涉及到这两个顶点的 $pair$ 与 $error$，更新 $Indices$ 中的索引，迭代到剩余点的数目不超过原先点数目的 $simplification\_radio$ 倍。具体来说，这里更新是以 v1 的坐标设置为合并点 v 的坐标，也就是更新后的 v1 就是合并后的顶点，$pair$ 与 $Indices$ 是把所有涉及到 v2 的顶点对或索引序列中 v2 对应的索引全部更新为 v1 的索引，$error$ 是把所有涉及到 v1 与 v2 的顶点对的合并代价全部更新为与 v1 的合并代价, 这样修改后 $pair$ 中可能出现顶点对两顶点索引相同的情况，所以在前面寻找代价最小的顶点对时需要提前判断顶点对是否合法。



(a) **simplify = 1** (b) **simplify = 2** (c) **simplify = 3** (d) **simplify = 4**

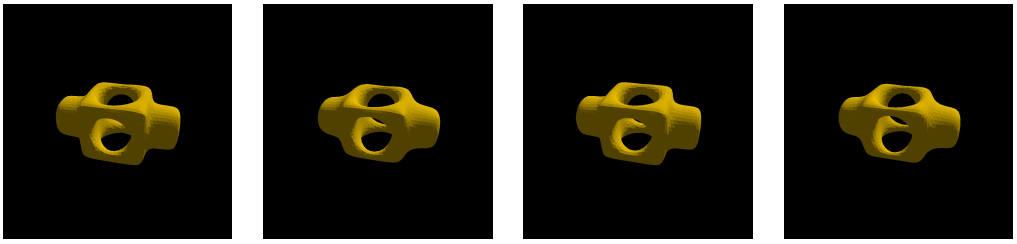
(e) **simplify = 1** (f) **simplify = 2** (g) **simplify = 3** (h) **simplify = 4**

图 3: **Mesh Simplification**

4 Task 4: Mesh Smoothing

首先建立 DCEL 查询链表并判断是否建立成功，之后迭代地进行顶点位置的更新，每一次迭代中，遍历每一个顶点，求得邻居顶点的与原先顶点的加权和作为新的顶点，顶点 i 相

应的公式为 $pos\_new = (1-\lambda) * pos\_old + \lambda * \sum_{j=1}^{n} w_{ij} * old\_pos\_neighbor\_j$, 其中使用 Uniform Laplacian 时 $w_{ij} = 1$, 使用 Cotangent Laplacian 时 $w_{ij} = cot\alpha_{ij} + cot\beta_{ij}$. 需要注意的是，由于 α, β 角度可能很小，导致这里的 cot 值可能会出现无穷大，需要进行特殊判断处理异常情况。

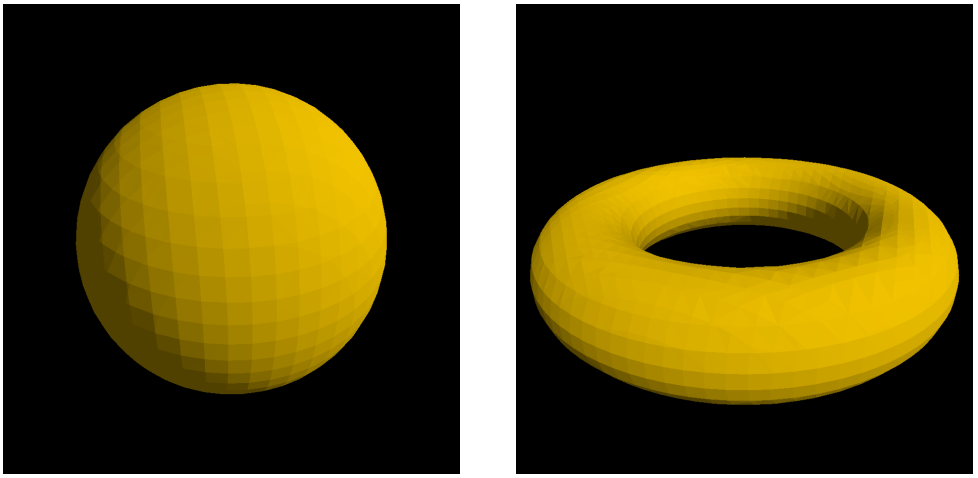


(a) **iter=5,Uniform** (b) **iter=10,Uniform** (c) **iter=5,Cotangent** (d) **iter=10,Cotangent**

图 4: **Mesh Smoothing**

5 Task 5: Marching Cubes

立方体在空间内分别沿三个坐标轴滑动，每滑到一个位置就计算 8 个顶点对应的函数值，函数值为负就把该顶点位设置为 1，以此得到一个索引 $index$，根据索引 $index$ 在 $EdgeStateTable$ 中找到相应立方体的边的状态 e，e 的第 i 位为 1 则代表第 i 条边上有顶点，此时根据边的两个顶点的函数值做线性插值得到该点的坐标与法向，插值公式为 $pos = pos_0 + dx * unit[j >> 2] * rate$(法向同理插值)，其中,$rate = sdf(pos_0)/(sdf(pos_0) - sdf(pos_1))$, pos_0 与 pos_1 是边的起点与终点的坐标。确定隐式面与该边的交点坐标。然后根据 $index$ 在 $EdgeOrdsTable$ 中找到相应的三角面片的连接方式，将插值的三角面片的顶点索引依次放入 $Indices$ 中完成三角面片的连接。



(a) **resolution = 35** (b) **resolution = 35**

图 5: **Marching Cubes**