Algorithmes de réduction de maillage

1) Suppression d’arête

Avec la formule du « Edge cost » (cout d’une arête)

On choisit de supprimer les arêtes qui ont le poids le plus faible jusqu’à obtention du nombre de polygone voulu. Lors de la suppression d’une arête [A,B], A vient fusionner avec B (A→ B). Ici nous différencions l’arête [A,B], dont la suppression implique A→ B), et [B, A] dont la suppression implique B→ A.

Le poids de chaque arête est calculé selon la formule du Edge cost exposée figure 1.

EQUATION 1. *The edge cost formula.*

$$\text{cost}(u,v) = \|u - v\| \times \max_{f \in Tu} \left\{ \min_{n \in Tuv} \left\{ \left(1 - f.normal \bullet n.normal\right) \div 2 \right\} \right\}$$

where *Tu* is the set of triangles that contain *u* and *Tuv* is the set of triangles that contain both *u* and *v*.

Équation 1: Calcul du poids de l'arête [U,V]

A chaque itération de l’algorithme, l’arête de poids le plus faible est supprimée.

Avantages	Inconvénients
❿ Calcul rapide, réalisable en temps réel ❿ Suppression d’arête sur les surfaces planes en priorité ❿ Possibilité de revenir en arrière dans la simplification si stockage des Vertex supprimés	❿ Le calcul du poids d’une arête ne prends pas en compte si elle est située dans une zone détallée ou non

Source : https://dev.gameres.com/program/visual/3d/PolygonReduction.pdf

2) Algorithmes de Visual Toolkit

2.a) VtkDecimate

L'algorithme vtkDecimate est basé sur différents états dans lesquels un Vertex peut être. Ceux-ci sont résumées figure 1. Pour chaque état on peut calculer une propriété caractéristique (qui sera par la suite notre critère pour l'omission d'un vertex) du Vertex : la Distance à une ligne, ou la Distance au plan. Ces propriété sont illustrées figure2.

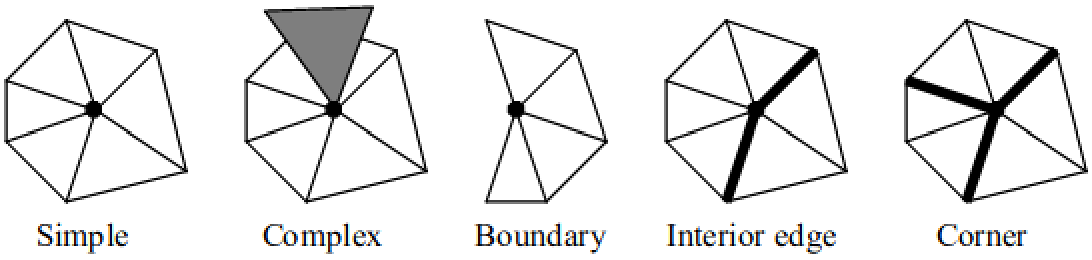


Figure 1: vertex topology

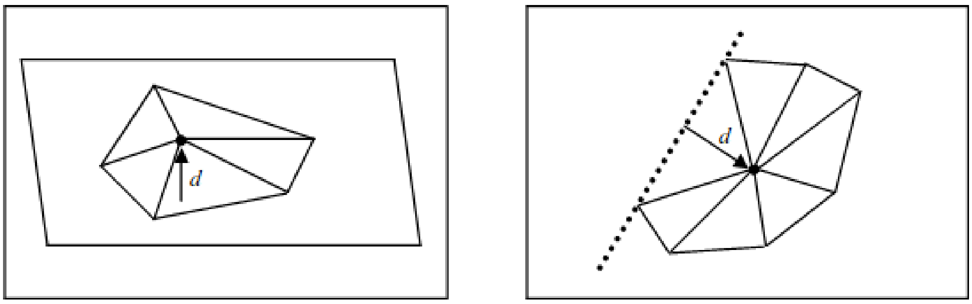


Figure 2: Error Metrics

L’algorithme est présenté figure 3.

```
while (reduction() < TargetReduction)
{
  for (i = 0; i < n; i++) //Pass thru n vertices
  {
    if (!VertexAlreadyRemoved(i))
    {
      c = ClassifyVertex(i); //Charac geo and topo
      if ((c == simple)&&(DistToPlane(i) < d1))
        candidateForDeletion = 1;
      else if ((c == corner)&&(DistToPlane(i) < d1))
        candidateForDeletion = 1;
      else if ((c == boundary)&&(DistToLine(i) < d2))
        candidateForDeletion = 1;
      else if ((c == int_edge)&&(DistToLine(i) < d2))
        candidateForDeletion = 1;
      else
        candidateForDeletion = 0; //Complex vertex
      if (candidateForDeletion == 1)
      {
        if ((CanTriangulate(i))&&(GlobErrorOk(i))
        {
          RemoveVertexAndIncidentTriangles(i);
          TriangulateHole(i);
        } //Line 21
      } //Line 19
    } //Line 6
  } //Line 4
} //Line 2
```

Figure 3: Algorithme de vtkDecimate

2.b) vtkQuadricDecimation

The VTK quadric decimation process consists usually of three steps:

1. Vertex pairs selection
2. Compute the error = the cost of decimation when a given vertex is removed
3. Triangulation: after a vertex has been removed, the resulting hole has to be triangulated

QEM (Quadric Error Metrics) pair selection

- Choisir un pair de nœuds (v1,v2).
- Soit v1, v2 une arrête.
- Soit |v1-v2| < T avec T défini par l'utilisateur « threshold »
- T=0 ➔ simple edge contracting algorithm
- T positif ➔ capable de connecter les parties non connectées de notre maillage
- Une fois le pair est contacté en un seul nœud v, on remplace v1 et v2 par v dans tout le reste des pairs.

QEM error approximation

Introduire "the quadric Qv" pour pouvoir sélectionner le meilleur pair à contracter pendant une itération. Qv s'agit d'une matrice qu'on associe à chaque vertex v, Qv égale donc la somme des matrices Kp définie en dessous qui représente le plan contenant un triangle du maillage et le vertex v.

- Associating 4x4 matrix $\mathbf{Q}_v$ with each vertex **v**
  - Construct plane p: $ax+by+cz+d = 0, a^2+b^2+c^2=1$ for each triangle incident to vertex **v**
  - Compute fundamental error quadric matrix $\mathbf{K}_p$ $$\mathbf{K}_p = \mathbf{p}\mathbf{p}^T = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$
  - $\mathbf{Q}_v$ is then sum of all $\mathbf{K}_p$
- Error (cost) at vertex v is $\Delta(\mathbf{v})=\mathbf{v}^T\mathbf{Q}_v\mathbf{v}$
- After contraction $(\mathbf{v}_1,\mathbf{v}_2) \rightarrow \mathbf{v}$, the new error matrix is $\mathbf{Q}_v=\mathbf{Q}_{v_1}+\mathbf{Q}_{v_2}$
- After contraction $(\mathbf{v}_1,\mathbf{v}_2) \rightarrow \mathbf{v}$, the position of v is such that it minimizes $\Delta(\mathbf{v})$, e.g. $\mathbf{v}=\mathbf{Q}_v^{-1}\mathbf{0}^T$
- Cost of contraction $(\mathbf{v}_1,\mathbf{v}_2) \rightarrow \mathbf{v}$ is $\Delta(\mathbf{v})=\mathbf{v}^T\mathbf{Q}_v\mathbf{v}$

The same here. How we compute the quadric Qv for a giving plane:

Garland and Heckberts' simplification algorithm is based on the iterative contraction of vertex pairs, which can be written as $(v_i, v_j) \rightarrow \overline{v}$. The order of edge collapse depends on the error which quantifies the distance of point to the faces.

Let v be a vertex in three-dimensional space which is $[v_x, v_y, v_z, 1]^T$. And p represents a plane in three-dimensional space, whose equation is

$$ax + by + cz + d = 0, \tag{1}$$

with $a^2 + b^2 + c^2 = 1$, and we denote $p = [a, b, c, d]^T$.

The distance from vertex v to the plane p is

$$\begin{aligned} D^2 &= \left(p^T v\right)^2 = \left(p^T v\right)^T \cdot \left(p^T v\right) = v^T \left(pp^T\right) v \\ &= v^T K_p v, \end{aligned} \tag{2}$$

where K_p represents a 4×4 matrix as defined in the following formula:

$$K_p = pp^T = \begin{vmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{vmatrix}. \tag{3}$$

Garland and Heckbert [2] associate a set of planes with each vertex. The error of each vertex is defined as the sum of squared distances to all planes. This vertex belongs to

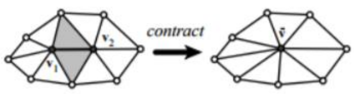
$$\begin{aligned} \Delta(v) &= \sum_{p \in \text{planes}(v)} D^2(v) = \sum_{p \in \text{planes}(v)} v^T \left(K_p\right) v \\ &= v^T \left(\sum_{p \in \text{planes}(v)} K_p \right) v, \end{aligned} \tag{4}$$

where planes (v) represents all the triangles that meet at that vertex.

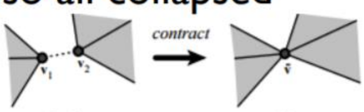
When an edge is contracted, the resulting quadric error is $\Delta(\overline{v}) = \Delta(v_i) + \Delta(v_j)$, where $\Delta(\overline{v})$ is the error for new vertex $\Delta(\overline{v})$.

QEM the sum of the squared distance from the vertex to each plane.

- 1. Compute the Q matrices for all the initial vertices.
- 2. Select all valid pairs.
- 3. Compute the optimal contraction target v for each valid pair $(\mathbf{v_1}, \mathbf{v_2})$. The error $\mathbf{v}^T(\mathbf{Q_{v_1}}+\mathbf{Q_{v_2}})\mathbf{v}$ of this target vertex becomes the cost of contracting that pair.
- 4. Place all the pairs in a heap keyed on cost with the minimum cost pair at the top.
- 5. Iteratively remove the pair $(\mathbf{v_1}, \mathbf{v_2})$ of least cost from the heap, contract this pair, and update the costs of all valid pairs involving $\mathbf{v_1}, \mathbf{v_2}$. Remove also all collapsed triangles.



Before After



Before After

On fait tout d'abord choisir un pair de vertices valides, ce sont les seuls à considérer lors du parcours de l'algorithme, et donc pour chaque pair, on calcule « the cost » défini par la forme quadratique Delta(v)=vT.Q.v , avec Q défini comme la somme des deux formes quadratiques Q1 et Q2 associées au pair (v1,v2). La nouvelle position de v est calculée en minimisant Delta(v).
Les étapes de cet algorithme sont résumées comme suit :

- Calculer les matrices quadratiques pour tous les nœuds (vertices) (the sum of quadric forms of planes representing all the triangles that meet at that vertex)
- Sélectionner tous les pairs valides.
- Calculer l'optimal résultat de la contrainte sur v (the new vertex) donc l'erreur Delta(v) de ce « target vertex » est notre « cost » sur quoi on se basera pour la suppression des nœuds d'une façon prioritaire.
- Placer tous les pairs avec chacun (chaque pair) leur cost dans un ordre croissant.
- Supprimer les pairs d'une façon itérative après avoir la remplacer par le nouveau vertex et mettre à jour tous les pairs candidats contenant soit v1 ou v2 par le nouveau vertex v.

The algorithm

The vtkQuadricDecimation algorithm begins by calculating the fundamental error quadric for each triangle (the corresponding plane, we name it for example Qp) (lines 1-2) and then we initialize the quadric Qv of each vertex to the sum of the quadrics of its associated triangles (sum Qp) (lines 3-4) . Next the algorithm computes the target vertex vk and cost for each edge (vi,vj) and inserts the result into a priority queue (lines 5−9). In accordance with the above discussion, the cost of collapsing an edge (vi,vj) to vk becomes vkT.(Qi+Qj).vk so the target vertex vk is chosen in a way that minimizes this error. In the lines 10−15, the algorithm simplifies the input mesh by iteratively collapsing the edge with the least cost (line 12) until some termination condition is met (line 10). After each edge collapse to a target vertex vk , it is necessary to update target vertices (line 13) and costs (line 14) for each edge associated with vk.

```
for (i = 0; i < N_triangles; i++)
  CalcFundErrorMatrix(i);
for (i = 0; i < N_vertices; i++)
  InitializeErrorMatrix(i);
for (i = 0; i < N_edges; i++)
{
  ComputeTargetVertex(i);
  CalcCostAndInsertIntoPriorityQueue(i);
}
while (reduction() < TargetReduction)
{
  CollapseEdgeWithLeastCost();
  UpdateNecessaryTargetVertices();
  RecomputeNecessaryCostAndInsertIntoPrioQueue();
}
```