Freestyle : Sculpting Meshes with Self-Adaptive Topology

Rapport Méthodes et Algorithmes

Étudiants :Charles Garibal, Maxime Robinot, Mathieu Dachy

Tuteur : Loic Barthe

10/11/2014

Table des matières

Introduction

Notre projet consiste en la réalisation d'une application de sculpture 3D en temps réel permettant une déformation arbitraire de la surface de l'objet et un changement de topologie transparent pour l'utilisateur. Ce dernier peut utiliser différents opérateurs de déformation et se concentrer sur l'objet qu'il sculpte tandis que le maillage « quasi-uniforme » se charge de séparer ou fusionner lorsque nécessaire.

Modification du maillage

Pour le projet nous allons utiliser un maillage dit « quasi-uniforme ». Ce maillage n'est valable que pour un maillage triangulaire et manifold. Il respecte quelques règles simples :

- La longueur maximale d'une arête est d_{detail} .
- La longueur minimale d'une arête est d : nous prendrons $d \leq d_{detail}/2$ (explication dans la partie Conversion en maillage quasi-uniforme).

Conversion en maillage quasi-uniforme

La première étape consiste à éliminer les arêtes de longueur inférieures à d.
Pour cela on itère sur toutes les arêtes. Lorsqu'une arête à une longueur inférieure à d, on fusionne les deux sommets de l'arête et on reconnecte le nouveau point correctement.

Pseudo-code :

```
Algorithme 1: 1ère conversion(élimination des arêtes trop petites)
Pour a∈Arêtes
    Si longueur(a) < d
        ns = (a.s1 + a.s2)/2
        Pour tout s∈{1−Anneau de s1}\{s2}
            m.ajout(new Arête(s, ns))
            m.suppr(m.arête(s, s1))
        FPour
        Pour tout s∈{1−Anneau de s2}\{s1}
            m.ajout(new Arête(s, ns))
            m.suppr(m.arête(s, s2))
        FPour
        m.ajout(ns)
        m.suppr(s1)
        m.suppr(s2)
    FSi
FPour
```

Remarque : Si une arête nouvellement créée est de longueur inférieure à d, l'itérateur repassera dessus car l'ajout se fait comme un push back.

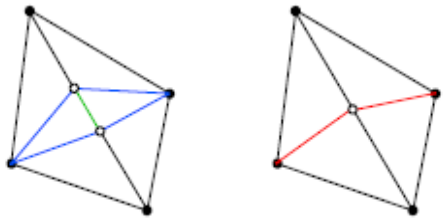


Illustration 1: Fusion

La seconde étape consiste à éliminer les arêtes de longueur supérieures à d_{detail}. Pour cela on itère sur toutes les arêtes à nouveau. Lorsqu'une arête à une longueur supérieure à d_{detail}, on crée un nouveau sommet au milieu de cette arête et on le connecte correctement.

Le choix d'un $d \leq d_{detail}/2$ s'explique par le fait que lorsque l'on va couper en deux les arêtes trop grandes, les arêtes résultantes auront une taille minimale de $d_{detail}/2 \geq d$. Nous n'aurons donc pas besoin de refaire la première étape.

Pseudo-code :

```
Algorithme 2 : 2ème conversion(découpage des arêtes trop grandes)
Pour a∈Arêtes
    Si longueur(a) > ddetail
        ns = (a.s1+a.s2)/2
        m.ajout(ns)
        m.ajout(new Arête(a.s1,ns))
        m.ajout(new Arête(a.s2,ns))
        m.ajout(new Arête(a.he1.next.s,ns))
        m.ajout(new Arête(a.he2.next.s,ns))
        m.suppr(a)
    FSi
FPour
```

Remarque : Comme précédemment, si une arête nouvellement créée est toujours de longueur supérieure à d_{detail}, l'itérateur repassera dessus.

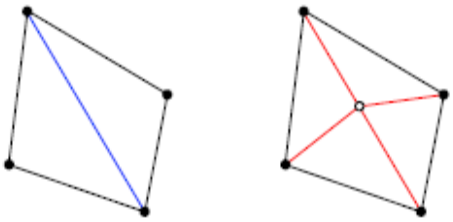


Illustration 2: Découpage

Open-Mesh

Il nous a été fortement conseillé d'utiliser Open-Mesh pour l'implémentation de notre chef d'œuvre. Ce qu'Open-Mesh va nous apporter :

- Une structure de données pour maillage complète et personnalisable (propriétés)
- Un framework de manipulation de maillage simple et efficace (Itérateurs et Circulateurs optimisés)

Open-Mesh fournit une structure de données du maillage en demi-arêtes.
Cette structure a les caractéristiques suivante :

- Chaque arête est décomposée en deux demi-arêtes.
- Chacune de ces demi-arête à un pointeur vers :
  - la demi-arête opposée
  - la demi-arête suivante
  - le sommet vers lequel elle pointe
  - la face à laquelle elle appartient
  - (optionnel) la demi-arête précédente
- Chaque sommet à pointeur sur une demi-arête partant de lui.
- Chaque face à pointeur sur une des demi-arêtes qui la compose.

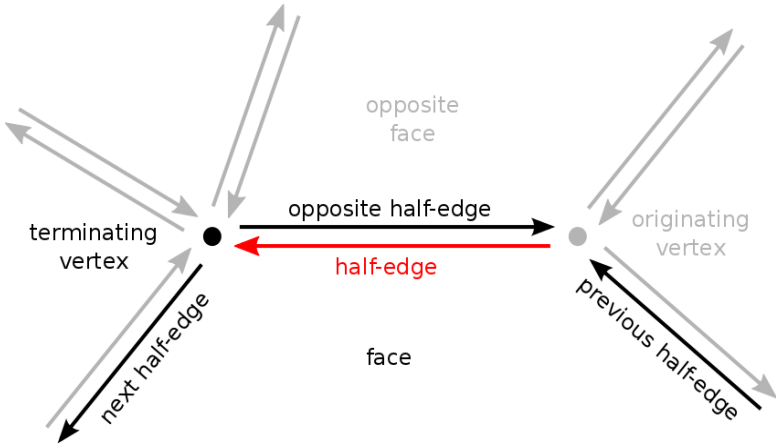


Illustration 3: Structure en demi-arête

C'est une structure qui est très pratique car elle permet plusieurs types de parcours du maillage. Par exemple, il est possible de trouver très facilement tout les sommets adjacents à un sommet en suivant une routine :

- le 1er sommet est celui pointé par la demi-arête
- ensuite on passe à la demi-arête opposée puis à la demi-arête suivante
- cette dernière pointe sur le second sommet
- on recommence : demi-arête opposée, demi-arête suivante, pointe sur un sommet adjacent
- jusqu'à retomber sur la 1ère demi-arête.

On peut également trouver tout les sommets d'une face, toutes les faces adjacentes à un sommet, etc … Open-Mesh fournit également des propriétés que l'on peut rajouter au maillage. Il est par exemple possible de rajouter une propriété « point » pour chacun des

sommets. Ainsi on peut ajouter un nouveau point dans cette propriété pour chaque sommet et ensuite changer les coordonnées des sommets par celles dans la propriété. Très utile pour appliquer une transformation aux sommets dépendante des sommets adjacents.
Pour nos besoins, cette structure sera très utile pour trouver efficacement les sommets qui sont dans la zone d'influence lors d'une transformation.

Picking

Il est indispensable d'implémenter une méthode de picking pour permettre à l'utilisateur de sélectionner un sommet ou une zone sur l'objet afin d'y appliquer des déformations.
Deux méthodes de picking s'offrent à nous : le raytracing et le colorpicking.

- Raytracing (lancé de rayon) : on lance un rayon au clic de l'utilisateur et on cherche la première intersection de ce rayon avec un maillage. Il est nécessaire d'avoir un picking précis au sommet près afin de poser correctement la sphère d'influence (voir dernière partie) pour les déformations. Cependant, c'est trop de précision demandé à l'utilisateur que de cliquer directement sur un sommet. Ainsi, les intersections seront calculées avec les faces (triangles) du maillage. Nous pourrons également implémenter une structure accélératrice (KdTree) pour ne pas tester des intersections avec absolument tout les triangles. Il nous est possible de récupérer le raytracing de l'UE ASDSI de l'année dernière qui effectue déjà des intersections rayon/triangle.
- Colorpicking : consiste à redessiner la scène avec une couleur unique pour chaque éléments de sélection et à lire la couleur du pixel du clic. Mais cette méthode ne se prête pas bien à une sélection aussi précise. En effet, en considérant que l'on affecte une couleur unique à chaque sommets du maillage, au moment du rendu il y aura interpolation des couleurs et il sera impossible de déterminer le sommet cliqué. De plus il est nécessaire que l'utilisateur clique exactement sur un sommet.

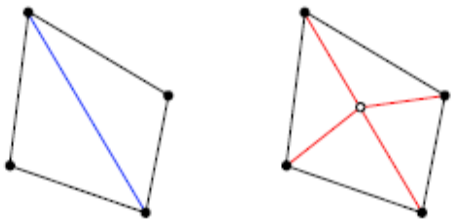


Illustration 4: Découpage

Evolution temporelle pendant une déformation

Nous allons montrer dans cette partie l'intérêt d'utiliser un maillage « quasi-uniforme » comme structure de données pour détecter une déformation de champ arbitraire qui est censée être connu avec une résolution égale à d_{detail}. Le paramètre $d_{thickness}$ doit être introduit pour la suite, il assure que le maillage restera manifold et qu'il ne s'intersecte pas tout seul.

Détecter une déformation spatiale au fil du temps

Soit un maillage « quasi-uniforme » M donné et un champ de déformation D, on utilise la contraction (« $d_{thickness}$ ») garantie par ces propriétés, en considérant la valeur du champ de

déformation aux sommets du maillage.

Cela élimine tout nécessité d'interpoler le champ dans l'espace environnant, même quand le champ dépend des propriétés caractéristiques à chaque sommet comme la normale.
Pour définir l'étape de déplacement maximum (« d_{move} »), utile pour détecter le mouvement de la surface en évolution, on a besoin de savoir la contraction minimum tolérée par le volume délimité par cette surface. Cette contraction peut être vu comme une caractéristique physique du matériau dans la structure virtuelle basée sur le maillage « quasi-uniforme ».

Définitions

$d_{thickness}$: Distance minimal entre deux sommets non adjacents. En-deçà, il y a fusion des deux sommets et réarrangement du maillage local.
d_{move} : L'étape de déplacement maximum permis pour un sommet est la valeur qui l'empêche de passer à travers une facette non incidente pendant le temps d'évolution du maillage.

Lemme

Soit un maillage quasi-uniforme M caractérisé par d_{detail} et dont la contraction n'excède pas $d_{thickness}$, l'étape de déplacement maximum d_{move} permis pour un sommet satisfait l'inégalité suivante :

$$4\,d_{move}^2 \leq d_{thickness}^2 - d_{detail}^2/3 \qquad (1)$$

Démonstration

d_{move} est déterminé quand un sommet Vni, non incident à la face F*max*, du maillage est initialement localisé à la même distance $d_{thickness}$ des 3 sommets de F*max*. Pour déterminer F*max*, il faut prendre une facette dont les 3 côtés sont égaux à d_{detail}.
L'inégalité (1) peut se déduire des propriétés de F*max* et de la configuration décrite précédemment.

Le sommet V*ni*, le centre de gravité G*max* de F*max* et un sommet Vfmax quelconque de F*max* forment un triangle rectangle et chaque triangle d'un maillage « quasi-uniforme » est équilatéral.

En appliquant le théorème de Pythagore et la propriété du barycentre d'un triangle équilatéral, on peut déterminer la longueur de l'hypoténuse :

$$d_{thickness}^2 \leq 4\,d_{move}^2 + d_{detail}/\sqrt{3}$$

Cette formule permet de retrouver l'équation (1).

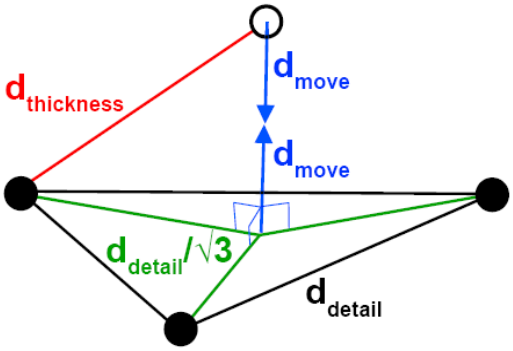


Illustration 5: Cas critique

Pour éviter qu'une arête soit renversée en raison de déplacement tangentiels de ces sommets (ce qui aurait pour effet que les normales des faces adjacentes pointent vers l'intérieur), il y a donc une autre contrainte à imposer :

$$d_{move} < d_{min}/2$$

où d_{min} étant la distance minimum entre deux sommets adjacents
d_{min} évolue dynamiquement durant la déformation du maillage, mais il a tendance à rester proche du paramètre *d* défini précédemment dans le maillage « quasi-uniforme ».
En pratique, choisir une valeur constante pour d_{move} dépendant de *d* plutôt que de d_{min} est suffisant.

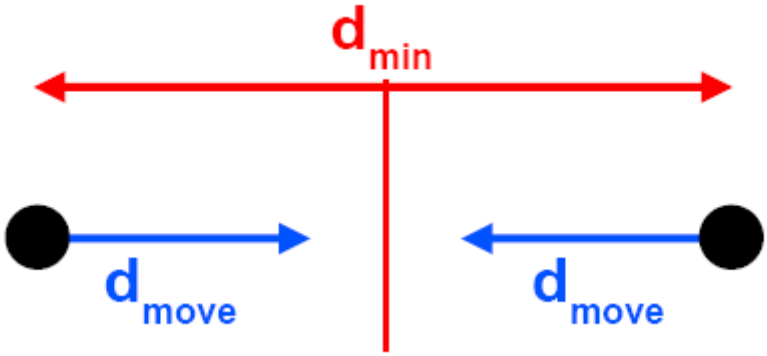


Illustration 6: Relation entre d_{min} *et* d_{move}

Lemme

Soit un maillage M « quasi-uniforme » caractérisé par d_{detail} et $d_{thickness}$, l'application d'un champs de déplacement satisfaisant les contraintes précédemment énumérées garantit qu'il n'y aura pas d'intersections ni d'inversions locales.

Changement de topologie : fusions et scissions

Le **chevauchement** local de deux parties de la surface non adjacentes est détecté avant qu'il n'arrive en réalisant une détection de collisions entre des sphères de diamètres $d_{thickness}$

centrées sur les sommets concernés.

Les changements de topologies sont déclenchés dès que deux sommets non adjacents sont à une distance l'un de l'autre inférieure à $d_{thickness}$. La **fusion** entre le voisinage de deux sommets est effectué en connectant leurs premiers anneaux. Les arêtes nouvellement crées sont appelées *connecting edges.*

Pseudo-code :

***Algorithme* 3 :** *Fusion des sommets non adjacents*
Pour tout $s_o \in$ *Sommets Observés*
  Pour tout $t \in$ *Pas de Temps*
    Pour tout $s_{na} \in$ *Sommets Non Adjacents*
      Si $distance(s_{na}, s_o) < dthickness$
        Pour tout $s \in \{1 - Anneau\ de\ s_o\}$
          s_{fusion}
          $minimum = grande_{distance}$
          Pour tout $s_2 \in \{1 - Anneau\ de\ s_{na}\}$
            Si $distance(s, s_2) < minimum$
              $s_{fusion} = s_2$
              $minimum = distance(s, s_2)$
            FSi
          FPour
          Tantque $distance(s, s_{fusion}) > dthickness$
            $[xpos, ypos] = interpolation(s, s_{fusion})$
            $s_{new} = m.ajout(new\ Sommet(xpos, ypos))$
            $m.ajout(new\ Arête(s, s_{new}))$
            $s = s_{new}$
          FTantque
          $m.ajout(new\ Arête(s, s_{fusion}))$
          $\{1 - Anneau\ de\ s_{na}\} \setminus \{s_{fusion}\}$
          $m.suppr(m.arête(s_o, s_{fusion}))$
          Si $m.sommet(s_o).arêtes = 0$
            $m.suppr(s_o)$
          FSi
        FPour
      FSi
    FPour
  FPour
FPour

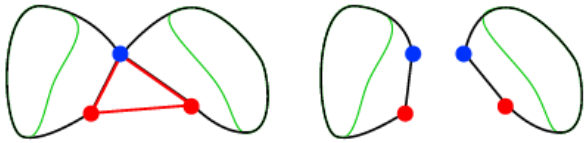
Le principal problème du processus de fusion est que les arêtes nouvellement crées peuvent avoir une longueur supérieures à d_{detail} . Cependant, ces arêtes ne seront pas modifiées avant le prochain mouvement. Ces sommets sont considérés comme des sommets protégés pendant certains pas de temps, le temps que la structure soit plus stable.

La destruction des arêtes et la fusion des sommets impliquent un nettoyage supplémentaire du voisinage qui élimine toutes les parties fines tel que les paires de triangles qui ont tout

leur sommet en commun et localement sépare la surface lorsque deux arêtes coïncident.

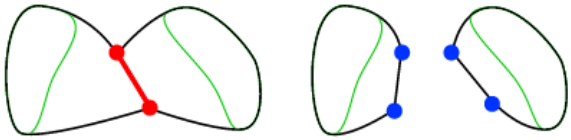
Dans ce cas-ci également, la propriété $d_{thickness}$ est assouplie temporairement pour les sommets qui sont séparés durant ce processus. Cette routine de nettoyage est la clé pour maintenir un maillage manifold.

Cas n°1 :



Les deux triangles coïncidant sont effacés et le sommet bleu qui est en commun pour les deux surfaces est séparé en deux, ce qui a pour effet de diviser la surface en deux également.

Cas n°2 :



Chaque sommet qui est incident à des arêtes coïncidentes entre deux surfaces sont séparés en deux sommets bleus. La surface est séparée en deux parties.

Algorithme 4 : *Nettoyage des parties fines*
Pour tout $f \in Faces$
  Pour tout $s \in f.sommets$
    Pour tout $f_2 \in f.faces_{adjacentes}$
      Si $\exists s_2 \in f_2.sommets = s$
        Pour tout $f_3 \in Faces, s_2 \in f_{3.\,sommets}$
          Pour tout $s_4 \in f_{3.}\ sommets$
            Si $adjacent(s_{2,} s_4)$
              $s_{adjacents} <- s_4$
            FSi
          FPour
        FPour
      FSi
      Si $s_{adjacents}.length = 2$
        $m.suppr(m.arête(s_{adjacents}[1], s_{adjacents}[2]))$
      Sinon
        $m.suppr(m.arête(s_{2,} s_{adjacents}))$
      FSi
    FPour
  FPour
FPour

Outils d'édition

Nous allons maintenant décrire les concepts et algorithmes qui permettront à l'utilisateur d'appliquer des déformations sur le maillage. Ces déformations sont au nombre de trois, selon leur type, elles suivent différentes règles mathématiques et sont garantes de certaines propriétés géométriques. Ces dernières peuvent être rangées dans deux catégories.

La première, ne fait pas dépendre la déformation de la géométrie locale du maillage. Cette déformation préserve le volume du maillage, l'empêche de s'auto-intersecter et garantit une continuité C^1. La seconde catégorie, contrairement à la première, s'appuie sur les caractéristiques du maillage pour faire évoluer la surface et permet notamment au maillage de changer de topologie.

D'une manière générale, l'utilisateur pourra opérer sur le maillage grâce à un outil de visualisation que nous appellerons la sphère d'influence. Cette sphère sera centrée sur le sommet visé par l'utilisateur et éventuellement, projetée sur la surface pour plus de lisibilité. Lorsque l'utilisateur initiera un mouvement de déformation, celui ci sera décomposée en autant de sous actions élémentaires que nécessaires pour obtenir des déformations uniformes. Un autre paramètre, d_{move}, est donc introduit pour fixer le déplacement maximum que peut effectuer un sommet lors d'une déformation.

Le concept mathématique caché derrière les déformations de maillage est celui de « champ vectoriel ». Un champ vectoriel est une fonction qui permet d'associer à un point de l'espace, un vecteur qui, dans notre cas, décrira le déplacement de chacun des sommets.

Dans cette partie nous ne traiterons pas du problème de changement de topologie qui s'opère lorsque deux sommets se retrouvent trop proches, ce dernier a été traité dans la partie précédente et et les algorithmes que nous allons aborder ne dépendent pas de son fonctionnement. Dans les parties qui suivent je parlerai de sc comme étant le sommet cliqué par l'utilisateur (sommet considéré).

Outil « étendre »

Le premier outil dont nous allons vous parler fait partie de la première catégorie décrite plus haut. C'est un simple outil permettant de tirer/étendre une partie du maillage. Pour imager son fonctionnement, il permet de faire suivre la matière lorsqu'on tire dessus.

Ici, la fonction de champ est définie par partie, chaque sommet évoluera différemment en fonction de sa distance au sommet traité par l'utilisateur. On distingue alors trois régions : interne, intermédiaire et externe et deux distances de seuil pour les départager : R_o et R_i. Ces valeurs peuvent être vues comme étant les rayons des sphères centrées sur le sommet considéré.

Pour pouvoir orienter la déformation, nous devons construire une base orthonormée, d'où le choix d'un vecteur, v. Ce vecteur peut être la normale du sommet considéré ou le vecteur entre la caméra et ce point, il n'est pas nécessaire de le savoir maintenant.

***Algorithme* 5 :** *Tirer un point du maillage*
Function $sweep(Mesh\ m, Sommet\ sc, Vect\ v, double\ R_i, double\ R_o)$
$u=\{1,1,(-v.x-v.y)/v.z\}$
$w=v\times u$
Pour $s\in m$
  $s_{sc/s}=s-sc$
  $r=\|s_{sc/s}\|$
  Si $r\le R_i$
    $vd=u\times(w\cdot s_{sc/s})$
  SinonSi $r\le R_o$
    $x=(r-R_i)/(R_o-R_i)$
    $vd=u\times((w\cdot s_{sc/s})\times(3x^4-4x^3+1))$
  Sinon
    $vd=\vec{0}$
  Fsi
  $s.move(vd)$
FPour

Outil « bomber/creuser »

Cet outil, comme son nom l'indique permet d'effectuer deux déformations possibles. Chacune d'elle à la particularité de favoriser le changement de genre. Contrairement à la fonction de champ précédente qui ne dépendait pas de la géométrie du maillage, cet outil utilise les normales des sommets pour déterminer la direction de leurs déplacements. Lorsque l'utilisateur veut « creuser », les sommets sont poussés dans le sens inverse de la direction des normales, l'inverse se produit pour bomber le maillage. Le paramètre associé, ϵ , vaut donc respectivement -1 ou 1.

Comme précédemment, la longueur du déplacement est pondérée par la distance du sommet traité au sommet considéré pour effectuer la déformation. Le déplacement s'annule si cette distance est trop importante, c'est à dire, si le point est en dehors du cercle d'influence de rayon R . Un autre paramètre important, n , contrôle les coefficients des polynôme de la fonction de « blending » et doit être supérieur à 2. Il permet notamment de paramétrer le lissage des jonctions entre les points qui se déplacent et les autres.

***Algorithme* 6 :** *Bomber/Creuser un maillage*
Function $infldefl(Mesh\ m, Sommet\ sc, Double\ R, Integer\ \epsilon, Integer\ n)$
Pour $s\in m$
  $s_{sc/s}=s-sc$
  $r=\|s_{sc/s}\|$
  Si $r\le R$
    $x=r/R$
    $vd=\epsilon\times((n-1)x^n-nx^{n-1}+1)\times s.getNormal()$
  Sinon
    $vd=\vec{0}$
  Fsi
  $s.move(vd)$
FPour

Outil « torsion »

Ce dernier outil est similaire au précédent puisqu'il fait partie de la même catégorie et utilise la même fonction polynomiale de paramétrage d'évolution de la surface. La dite torsion s'effectue autour de la normale du point cliqué par l'utilisateur, encore une fois, cette rotation d'un point est pondéré par sa distance qui le sépare de la sphère d'influence.

Algorithme 7 : *Torsion d'un maillage*
Function $twist(Mesh\ m,\ Sommet\ sc,\ Double\ R,\ Integer\ n,\ Double\ A_0)$
Pour $s \in m$
  $s_{sc/s} = s - sc$
  $r = \|s_{sc/s}\|$
  $N = sc.getNormal()$
  Si $r \leq R$
    $A = A_0 \times ((n-1)r^n - nr^{n-1} + 1)$
    $vd = (s_{sc/s} - (s_{sc/s} \cdot N) \cdot N)(1 - \cos A) + N \times s_{sc/s} \cdot \sin A$
  Sinon
    $vd = \vec{0}$
  Fsi
  $s.move(vd)$
FPour

Conclusion

Nous venons de présenter le fonctionnement des modules que nous allons ajouter au moteur graphique 3D existant. Ce moteur est celui que nous avons utilisé en M1 lors de l'UE ASDSI. Ces modules permettront à l'utilisateur de sculpter un maillage de manière intuitive grâce à un outil que nous avons appelé « sphère influente ».

Le premier module permet d'obtenir un maillage « quasi-uniforme » quelque soit le maillage en entrée grâce au framework OpenMesh. Au cours des déformations, il peut s'opérer un changement de topologie qui est géré par un second module. Enfin, trois outils seront mis à la disposition de l'utilisateur pour lui permettre de sculpter l'objet.

Les principales difficultés qui nous envisageons de rencontrer sont la mise en relation des modules, l'intégration d'OpenMesh et la gestion de projet.

Index des illustrations