Projet extrusion

Géométrie pour la 3D

L3 S6 informatique

On cherche à extruder un polygone plan suivant n'importe quelle direction. La révolution est un cas particulier d'extrusion.

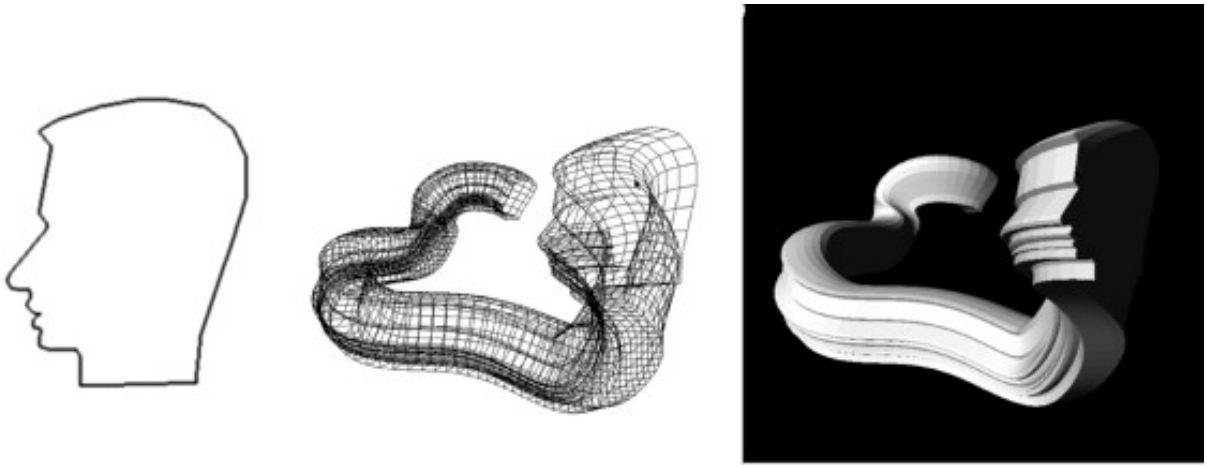

Figure 1. Un polygone plan (à gauche) est extrudé suivant les lignes d'un champ vectoriel aléatoire (bruit de Perlin) il en résulte une forme complexe visualisée en filaire (au milieu) et en ombré (à droite).

Pour ce faire, nous allons partir du canevas Extrusion.tgz que vous pourrez télécharger depuis Moodle.

1. Nouveaux traitements sur les vecteurs

Pour commencer il nous faudra quatre fonctions supplémentaires dans le fichier Vector.c. En voici les profils :

```
double V_decompose(Vector p, Vector u);
// Si (u,v,w) est un repère orthonormé de l'espace, chaque vecteur p
// possède des coordonnées dans ce repère. Cette fonction retourne
// la coordonnée selon u du vecteur p.

Vector V_recompose(double x, double y, double z,
                   Vector u, Vector v, Vector w);
// Si (u,v,w) est un repère orthonormé de l'espace, cette fonction
// retourne le vecteur dont les coordonnées dans cet espace sont (x,y,z)

void V_uxUyFromUz(Vector uz, Vector *ux, Vector *uy);
// A partir d'un vecteur uz, cette fonction retourne deux vecteurs
// ux et uy tels que (ux, uy, uz) soit un repère orthonormé direct
// et que les vecteurs (uy,uz) soient dans le même plan que les
// vecteurs ((0,1,0),uz) sauf si uz est lui-même colinéaire à (0,1,0).

Vector V_rotate(Vector p, Vector centre, Vector v1, Vector v2);
// Cette fonction opère sur le point p une rotation autour de centre
// et qui transforme le vecteur v1 en vecteur v2.
```

Pour la fonction V_decompose, nul besoin de changement de repère. Cette fonction nécessite au plus une ligne de code. Pour la fonction V_uxUyFromUz, pensez à ce qu'on peut faire avec un produit vectoriel.
Enfin la fonction V_rotate met en œuvre les fonctions précédentes. L'idée est de construire deux

repères orthonormés contenant respectivement v1 et v2 : (ux,uy,v1) et (vx,vy,v2). Pour cela, on peut utiliser V_uxUyFromUz. La rotation qui nous intéresse est celle qui transforme le premier repère en le deuxième repère. Pour avoir l'image du point p par cette rotation, on calcule les coordonnées (x,y,z) de p dans le premier repère (V_decompose). L'image de p est le point qui a les mêmes coordonnées (x,y,z), mais dans le deuxième repère. On peut obtenir ce vecteur avec V_recompose.

2. Nouveaux traitements sur les polygones

A présent, nous cherchons à réaliser des transformations sur des polygones plans : translation et rotation autour du centre. Voici les profils des fonctions à écrire :

```
Vector P_center(Polygon *P) ;
// Cette fonction retourne le centre d'un polygone plan, c'est-a-dire
// la moyenne des positions de ses points.

Vector P_normal(Polygon *P);
// On suppose que P est un polygone plan. Cette fonction retourne
// un vecteur unitaire normal à ce polygone.

void P_translate(Polygon *P, Vector trans);
// Translate tous les sommets de P d'un vecteur trans.

void P_rotate(Polygon *P, Vector normal);
// Faire tourner le polygone plan P autour de son centre de façon à
// ce que son vecteur normal soit égal à normal. (Cf fonction V_rotate)
```

N'oubliez pas de tester vos fonctions.

3. Nouvelles fonctions traitant les maillages.

Pour obtenir des formes de révolution, il faut tourner les polygones plans autour de l'axe Y. Dans notre cas, nous allons utiliser un champ vectoriel aléatoire appelé *bruit de Perlin*. Vous n'aurez pas à l'écrire. Vous trouverez dans le module Perlin.c la fonction suivante :

```
Vector PRLN_vectorNoise(Vector p);
```

Pour réaliser une extrusion, la procédure est la même que pour une révolution. Au lieu de tourner les polygones plans autour de l'axe Y, on calcule le bruit de Perlin v_noise au centre du polygone courant. A chaque pas de l'extrusion, le polygone suivant s'obtient à partir du polygone courant en le translatant de v_noise (avec la fonction P_translate) et en tournant ce polygone de façon à ce que son vecteur normal soit colinéaire à v_noise (avec la fonction P_rotate).

```
void M_perlinExtrude(Mesh *M, Polygon *p, int nb_slices);
```

Ce projet est à réaliser de façon individuelle. A la séance de TP du 6/12, chacun d'entre vous me montrera l'état d'avancement de son projet. Je vous demanderai peut-être de m'expliquer certaines parties du code ou d'en modifier certains aspects. Ensuite avant le lundi 9/12 à 23h59 vous rendrez dans la rubrique Moodle prévue à cette effet une archive contenant :

1. votre code ;
2. une capture d'écran montrant le résultat que vous avez obtenu ;
3. un fichier LISEZ-MOI.txt spécifiant, sans rien omettre, toutes les opérations (frappes de touches, clics, etc) à réaliser pour obtenir l'image obtenue (2/). Notez bien qu'il ne s'agit aucunement d'un mode d'emploi montrant tout ce qu'on pourrait faire, mais d'une procédure à suivre pour obtenir l'image que vous avez montrée au 2/.