Synthèse d'image

10 avril 2015

1 Modèle

Modélisation : tte méthode permettant de réprésenter un objet dans le cadre 3D et de le manipuler. Un modèle, c'est :

- Un ensemble de données, en deux type
  - soit des données pour décrire la surface (2D). Les modèles surfaciques sont adaptés à la visualisation temps réel, car les cartes graphiques actuelles en sont capables
  - soit des données pour décrire le volume (3D).
- les algo de manipulation qui vont avec :
  - Les requêtes élémentaires : ($pt \in O$?, calculer l'intersection entre une droite δ et un objet σ, ...
  - Les ensembles : l'intersection entre deux objets existe-t-elle ?, déformation, animation...
  - conversion. Comme certaines opérations sont plus faciles avec certains modèles et beaucoup plus dur avec d'autres, il est nécessaire de pouvoir convertir les données d'un modèle à un autre.

2 modèle surfacique

- Surfaces paramétrés : $S = p(u, v) \quad (u, v) \in \omega$

xR
p(u, v) : $\omega -> R^3$
$(u, v) -> p(u, v)$
On travaille avec des pts de contrôle
exemples :

- $p(u, v) = uv0$

Souvent, on va travailler avec des ω simples $[O, 1]^2$ et p(u, v) simple ($R_n[u, v]$
On définit des morceaux (patchs) de surfaces 3D.
Utilisation de polynômes
Famille de patches : Bézier, Splines, Coons...
Ce sont des surfaces ayant de bonnes propriétés (polynomiales, donc contrôles de beaucoup de propriétes). Mais le problème vient du raccord entre plusieurs morceaux (préservation de continuité -> contraintes venant de quatres sources à satisfaire -> le degré du polynôme augmente pour absorber les contraintes -> on peut passer de degré 3 et 4 à des degrés 24.
Conversion d'une surface paramétré en un ensemble de triangle (maillage).

- Surfaces de subdivision : -> Aller lire ce que sont les surfaces de subdivision -> ultra utilisée dans l'industrie graphique, notemment animation. Pas de polynome, mais analyse spectrale. Passage d'un contrôle grossier à une surface lisse par un processus de subdivision. Gueri's Game Youtube. Subdivision : chaque triangle, à chaque itération, est coupé en quatre triangle, avec un raffinement récursif.
- Maillaige : Un maillage est constituée de deux types de données :
  - des données géométriques, qui permettent le déplacement
  - des données topologiques (sommets V, arêtes E, faces F, liés les un aux autres). Elles permettent de savoir qui est connecté à qui, ce qui est utile quand on change de topologie, c-a-d quand on change le nbre d'arete, de sommet ou de face.

Modèle eulérien : Relation d'Euler = $V + F - E = 2$
Les faces peuvent être n'importent quel type (triangle, carré, etc).
Structure minimale pour représenter un maillaige :

- Une structure pour stocker la géométrie : un tableau (QVector<Vector>)
- Une structure pour la topo minimale : un tableau (QVector<int> tcom) pour stocker des triplets, contenant les indices des points dans le tableau de la géométrie.

$\forall$ opération de mvmt solide (trslat, rotat, ..) => géom parcours
L'affichage est un parcours de la topologie (avec un parcours de la géométrie indexée par la topologie)
Lorsqu'il y a une déformation de maillage (mvmt, par exemple), on surmaille (on rajoute un nmbre de pts) à l'avance la zone destinée à une animation

non solide.
Pour gérer des maillages aux éléments vairables, on stocke la somme du nbre de point par mesh dans un autre tableau.
On a souvent besoin d'info supplémentaire : la normale au sommet (pour chaque sommet), des textures, ... Pour cela, soit on fait des tableaux supplémentaires de données, soit on entrelace les données.
Souvent, on aura besoin de normales au sommet d'un maillage. La normale à un triangle abc : $n = \frac{(b-a)\otimes(c-a)}{(b-a)\otimes(c-a)}$(on redivise par lui-même pour avoir un vecteur unitaire). tous les calculs d'illuminations prenent en compte la normale aux facettes.

3 Modèle surfacique

Modèles qui s'int au volummes, in/out (bool), type de matière(int).
Exemple de modèle volumique :

— Enumeration spatiales (voxels)
— Constructive Solide Geometry
— surfaces implictes -> nécessites beaucoup de boulot.

3.1 Voxel

Un voxel est un cube 3D (pixel) qui contient des informations (bool in/out, int type_mat, ...).
Le modèle voxel, c'est du Lego (je cite).

selon un abus de langage, un voxel, c'est une grille 3D (souvent régulière) : une box 3D, int n(subdivision régulière de la boite), un tableau de données de taille n^3 représentant.

Remarque : tableau 1D pour stocker 3D : index $= i + nj + n^2k$. $i = index\%n$. j $=((index - i)(n)\%n)$.

Inconvénients :

— cout possiblement très élevé (O(n^3)).
— données discrètes -> besoin de lisser

Avantages :

— in/out immédiat
— accès rapides aux datas
— structure de données très pratiques en simulation

Soit $p \in R^3$, comment savoir dans quel cube voxel je suis ?
$uvw = \frac{p_x - a_x}{b_x - a_x}$..... $\in [0,1]^3$
$ijk = E(uxn)$..... .

remarque : optimisation technique index $= i + nj + n^2k$ peut poser des problèmes d'optimisation. Car accèder à des voxels cote à cote selon i se fait avec un + 1 dans la mémoire, mais accéders à des vocels cote a cote selon j ou k se fait en + n. Et la mémoire n'aime pas. Donc il existe des techniques plus avancées pour gérer ce problème.

-> E.Haine T.Möller, *Real time Rendering*, 3 eme édition, AK Peters.
Fundamental of CG, P. Shirley, A.K Petes.

3.2 CSG

définir un objet par un arbre de construction :

— Noeuds : opérateur booléen (Union, intersection, différence)

— Feuilles : primitives géométriques (celles pour lesquelles on peut déterminer si on est dedans (volumme))

exemple : si on fait l'union d'un cylindre et d'un cube, on obtient une masse, et si on y enlève un cylindre couché, on obtient un marteau.

On fait du complexe par assemblage simple.

Requête fondamentale sur un mdoèle volumique : bool inside(pt) -> est-ce qu'un pt est à l'intérieur de mon modèle ?

exemple d'implémentation :

```
N::inside(p) = 0
-> Sphère::inside(p)
-> U::inside(p)
```

Tout noeud doit savoir si un point est à l'intérieur :

```
class N {
public : bool inside(v) const = 0;
}
class S : public N{
protected :
V c;
d r;
public :
bool Inside(v) const;
}
class U : public N {
protected :
N *a; //Data sous arbre
N *b;
public :
bool Inside(v) const;
}

bool S::Inside(p) const {
return (p-c)(p-c) < r?true:false;
```

```
}

bool U :: Inside(p) const {
return a->inside(P) || b->Inside(p);
}
```

- Cube : six tests
- Sphère d(p, c)
- Cylindre : d(p, δ)
- Tore : d(p, C)
- Cone : complexe

Tester si je suis à l'intérieur d'un maillage est extrêment compliqué. Il faut que le maillage soit fermé, cohérent, et 2-variété (Manifold). On utilise le théorème de Jordan, en tirant une demi-droite partant du point. Si le nbre d'intersection avec le maillage est pair, le point est en dehors, sinon, il est dedans. Maillage : :Inside(p) -> Triangle : :Inside(p).

4 Texture

4.1 Texture and (Shading ou illumination globale)

Texture : Carac de surface/volumme du matériau, lui donnant sa couleur propre.
Illumination globale/Shading : interraction entre la lumière, les matériaux et l'oeil. -> Constitura le chap III

Hypothèse : la couleur est représenté dans un système RougeVertBleu : un certain espace de couleur peut-être approximé par un triplet de valeur $\in [0,1]^3$. $[0,0,0]$ représente le noir, $[1,1,1]$ le blanc.

Problème :
être capable de définir la couleur $c(p) \forall p \in Obj$. Ce qui nous intéresse plus précisément, c'est la couleur des points à la surface de l'objet : $c(p) \forall p \in Surf(O)$ -> Texture surfacique. Il existe aussi la texture volumique.

Évidement, les texture volumiques sont plus destinées aux modèles volumiques, mais les textures surfacique peuvent être utilisés avec les deux modèles.

Critère de comparaison de méthode de texture :

— rapidité de calcul
— mémoire nécessaire
— réalisme -> critère subjective
— facilité d'usage/de contrôle

4.2 Placage de texture

Soit un modèle géométrique G (surface maillée). On cherche une correspondance entre tous les points de la surface (en 3D) et les points de ce que serait cette surface en 2D. On travail dans le domaine de la texture $\omega = [0,1]^2$, qui contiendra une image de résolution n*n.
Pour trouver la correspondance, on travail avec des coordonnées liés à chacun des triangles du modèle géométrique. On rajoute, en prenant par exemple un triangle abc, les coordonnées des points dans la texture -> $a, (u_a, v_a); b(u_b, v_b); c, (u_c, v_c)$.

Pour un point p ($\forall p \in G$, $p \in T_{(abc)} \in \omega$), soit ($\alpha_{p,p}$) les coordonnées barycentrique de p dans T. $q \in T \in \Theta$ avec les mêmes (α,).

Problème :
Coment découper la peau et la déplier ?

Cas simples : formes géométriques élémentaire :

— C'est simple pour un plan
— Pour une sphère, on utilise les coordonnées polaires des points : $v = \frac{C+\frac{\pi}{2}}{\pi}$ et $v = \frac{\theta}{2\pi}$.

Cas réels :
C'est fait à la main : dépliage des (u, v). Il existe des outils d'aide.

Critique :

— temps de calcul : temps réel, (GPU), le calcul des (u, v) est fait en amont (préprocessing)
— coût mémoire : taille d'une image n*n -> coûteux, car $n \approx 1024$, soit 4 Mo
— réalisme : Non applicable
— contrôle : placement très précis -> bcp d'opérations à faire à la main pour être très précis

4.3 Textures volumiques

c(p) va être une fonction math $\mathbb{R}^2 \to \mathbb{R}^4$
$p(x, y, z) \to (R(x, y, z), V(x, y, z), B(x, y, z), \alpha(x, y, z))$ void Mesh : :append(const Mesh mToAppend)

Critique :

— temps de calcul : plus long, mais désormais fait par le GPU Shader -> temps réel désormais
— coût en mémoire : une fonction, donc une texture réprésente quelques ko.
— réalisme : gros inconvénients : seules quelques textures se prêtent bien à leur éxécution sous la forme d'une fonction