Textures : un peu de maths

Nicolas Holzschuch
iMAGIS/GRAVIR-IMAG

*i*MAGIS is a joint project of CNRS - INPG - INRIA - UJF

iMAGIS

Plaquage de textures

- On part d'une fonction 2D
  - la « texture »
- On veut établir une correspondance entre les coordonnées de la texture et les coordonnées de l'objet

iMAGIS

Algorithmes de plaquage

- Forward Mapping (basé sur la texture) :
  - **pour tout** (u,v)
    - **trouver**(x,y) = f(u,v)
    - image[x][y] = texture[u][v]
- Backward Mapping (basé sur l'écran) :
  - **pour tout**(x,y)
    - **trouver**(u,v) = $f^{-1}(x,y)$
    - image[x][y] = texture[u][v]

iMAGIS

Algorithmes de plaquage

- Algorithme en deux passes :
  - **pour tout** (u,v)
    - **trouver** x(u,v)
    - tmp[x][v] = texture[u][v]
  - **pour tout** (x,v)
    - **trouver** y(x,v)
    - image[x][y] = tmp[x][v]

iMAGIS

Quel algorithme ?

- basé sur la texture :
  - Peut laisser des trous à l'écran
  - Mais pas d'inversions à faire
  - Applications temps-réel
- Basé sur l'écran :
  - Il faut inverser f
  - Intégré dans les cartes graphiques
- Deux passes :
  - Toujours 1D
  - Plus de coût mémoire

iMAGIS

Coordonnées

2D texture space *(u,v)*
↓ paramétrisation (T)
3D object space
↓ trasnformation du modèle (M)
3D world space
↓ transformation de caméra (V)
3D eye space
↓ perspective (P)
2D pixel space *(x,y)*

iMAGIS

Paramétrisation des objets

- Conversion de la texture à l'objet
- Direct pour les triangles :
  - Et linéaire aussi : $x = au+bv+c$
- Pour les autres objets :
  - non-linéaire en général
  - On préfère subdiviser en triangles
- Cas particulier : les parallélogrammes :

  $(x,y,z,w) = \mathbf{M}(u,v,1)$ (linéaire)

Backward mapping

- Toutes les transformations sont linéaires :

  $(x,y,w) = PVMT(u,v,1)$
- L'inverse est aussi linéaire :

  $(u,v,w) = (PVMT)^{-1}(x,y,1)$
- Linéaire, mais homogène :

  $u=(ax+by+c)/(gx+hy+i)$

  $v=(dx+ey+f)/(gx+hy+i)$

Projection Orthographique

- La matrice de projection orthographique est affine :
  - $w'=w=1$
- L'inverse est affine aussi :
  - $w=w'=1$
  - On peut faire du texture mapping avec seulement deux additions par pixel

Deux additions par pixel

- $u = (a/i)\,x + (b/i)\,y + (c/i)$
- $v = (d/i)\,x + (e/i)\,y + (f/i)$
- Algorithme :

```
for all y
    u = (a/i) xmin + (b/i) y + (c/i)
    v = (d/i) xmin + (e/i) y + (f/i)
    for all x
      u += (a/i); v += (d/i)
      image[x][y] = texture[u][v]
```

Perspective

- Forme Générique :

  $u=(ax+by+c)/(gx+hy+i)$

  $v=(dx+ey+f)/(gx+hy+i)$
- 3 additions, 2 divisions par pixel
- Peut être optimisé pour les cas particuliers :
  - Polygones horizontaux/verticaux (Doom)

3 additions, 2 divisions

- Algorithme :

```
for all y
    unum = a xmin + b y + c
    vnum = d xmin + e y + f
    den = g xmin + h y + i
    for all x
        image[x][y] = texture[unum/den][vnum/den]
        unum += a
        vnum += b
        den += g
```