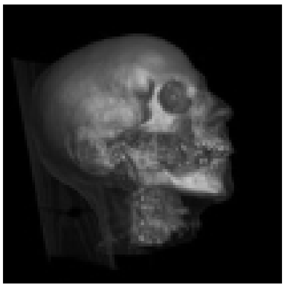
Rendu volumique

Nicolas Holzschuch
*i*MAGIS/GRAVIR IMAG
Nicolas.Holzschuch@imag.fr

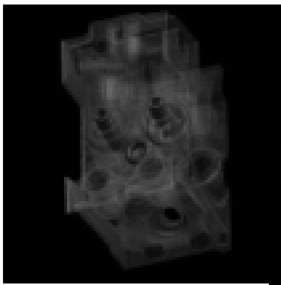
Rendu volumique

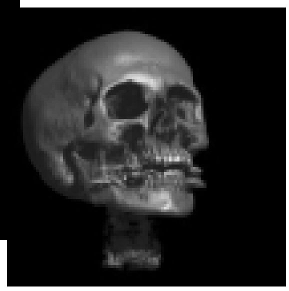
- Pourquoi faire ?
  - Parce qu'on a des données volumiques…
    - Simulations numériques
    - Scanners, données médicales
  - Visualiser et comprendre
- Qu'est-ce qu'on visualise ?
  - Ce qu'il faut pour comprendre
- *Comment* faire ?

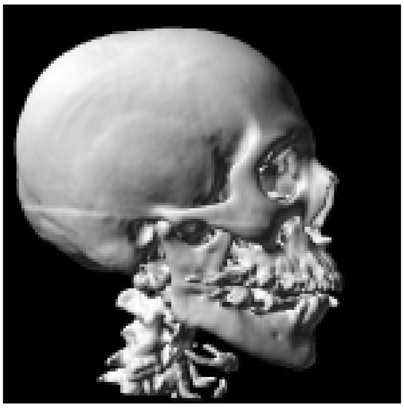
Exemples

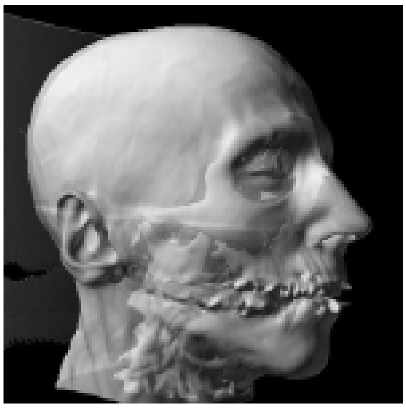


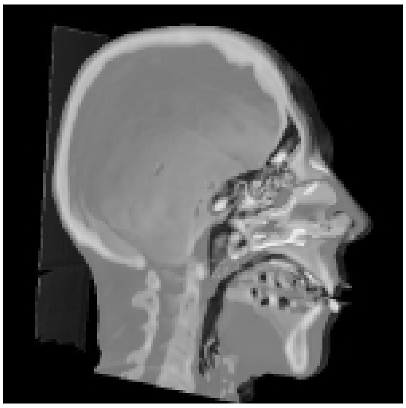
Exemples

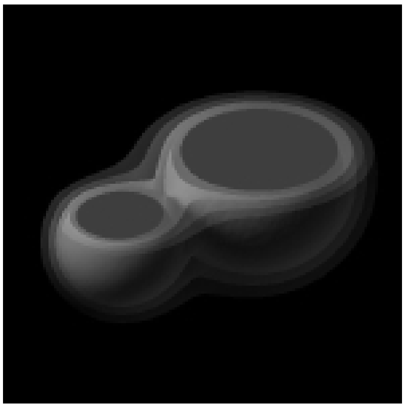












Plan

- Le traitement des données
- Les techniques standard, vue générale
  - Surfaces, pixels, volumes
- Les techniques standard, détails
  - Surfaces, pixels, volumes
- Utilisation du hardware
- Et le réalisme ?

Données de départ

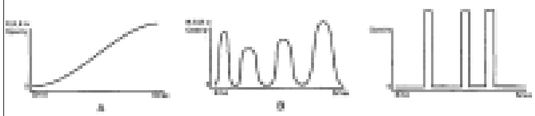
- Données volumiques
  - Champ volumique *f(x,y,z)*, voire *f(x,y,z,t)*
  - Densité, température, concentration,…
- Échantillonnage
  - Régulier ou irrégulier
  - Éventuellement insuffisant
    - Objets échantillonnés ? Hautes fréquences ?
    - Limite de Nyquist

Techniques générales

- Beaucoup d'information
  - Trop ?
- Éliminer certaines informations :
  - Cacher, masquer,
  - Rendre transparent,
  - Couper…
- Perception des surfaces :
  - Ombrage

Traitement des données

- Affichage : RGB + opacité
- Fonctions de transfert :
  - Permettent de souligner les détails, au choix



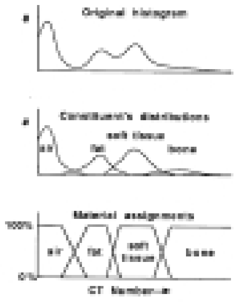
Interpolation des échantillons

- Interpolation linéaire entre les échantillons
  - Linéaire par axe, tri-linéaire au total :

$$S(x,y,z) = a_1 + a_2 x + a_3 y + a_4 z + a_5 xy + a_6 xz + a_7 yz + a_8 xyz$$

  - Calcul des coefficients : $[x_{ij}][a_j] = [S_i]$
  - Par rapport à la distance à l'œil : S est cubique

Techniques générales

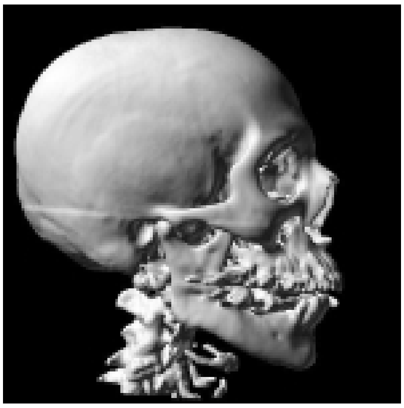


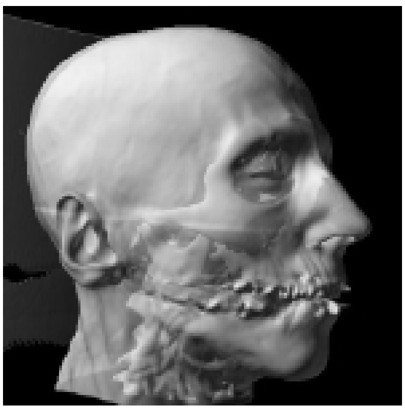
Techniques de visualisation

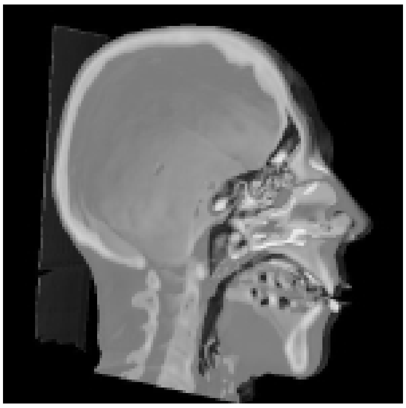
- Par surfaces
- Par pixels
- Par volume

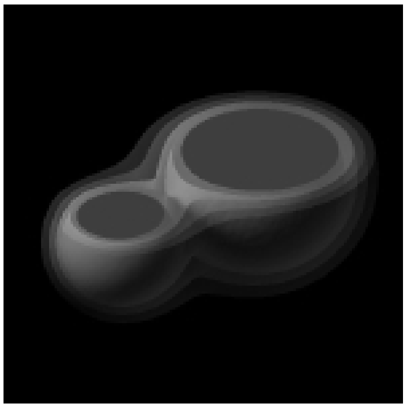
Techniques de visualisation

- Par surfaces :
  - Extraire les surfaces iso-potentielles
    - Marching-cube, octree…
  - Visualisation standard :
    - Z-buffer, ombrage, transparence…
  - Pré-traitement (un peu) long
- Par pixels
- Par volume









Techniques de visualisation

- Par surfaces
- Par pixels :
  - Lancer de rayon
  - Partie du volume de données intersectée par le rayon
  - Intégration/maximum/autres...
- Par volume

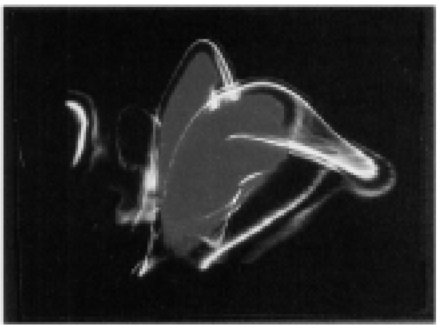
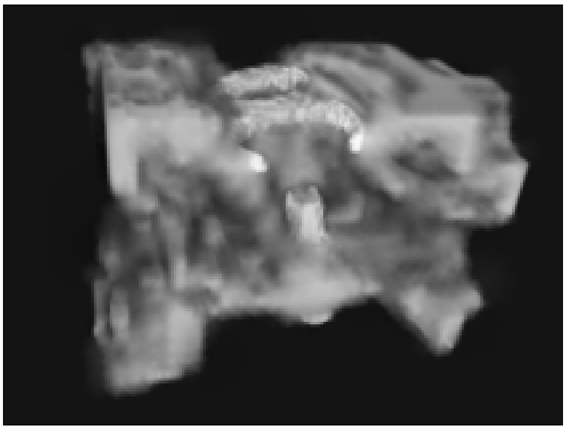


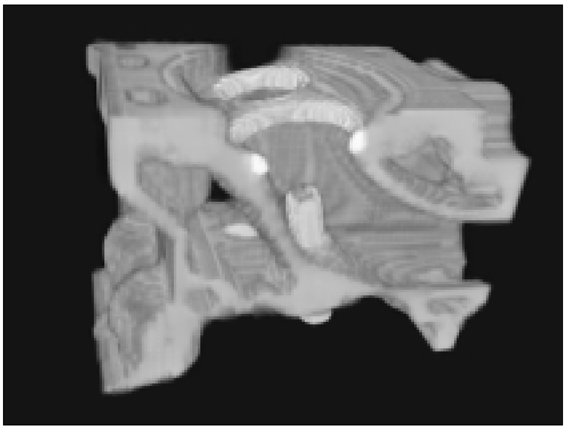
Figure 7. Rain water content of a severe storm simulation. (Data courtesy of Robert Wilhelmson, NCSA).

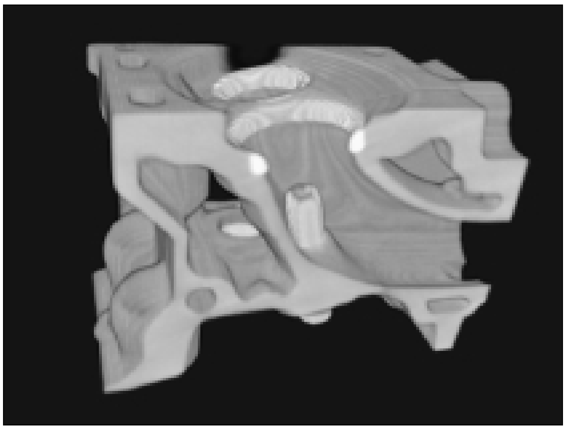
Figure 8. Rain water image with illumination from a single light source.

Techniques de visualisation

- Par surfaces
- Par pixel
- Par volume :
  - Projection des cellules du volume
    - Interpolation dans la projection
  - Intégration directe :
    - Valeur à un pixel = intégrale pour chaque échantillon
    - Projection orthographique :
      - Noyau indépendant de la position
    - Afficher la projection du noyau







Et maintenant, les détails

Extractions de surfaces

- Marching cubes (Lorensen, 1987)
  – Extraction de l'iso-surface et des normales
  – Échantillons ponctuels, organisés en cubes

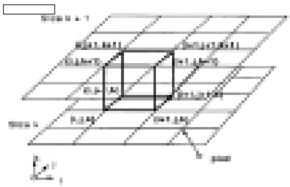


Figure 2. Marching Cube.

© Lorensen, 1987

Marching cubes

- Sélection des cubes qui intersectent l'iso-surface :
  – Au moins un sommet dedans, au moins un sommet dehors
- Classification des cubes :
  – 8 sommets, 2 états (dedans, dehors) = 256 possibilités
  – Symétrie des 2 états : 128 possibilités
  – Symétries par rotation : 14 possibilités
  – Pour chaque possibilité, une seule triangulation possible
  – De 1 à 4 triangles par cube

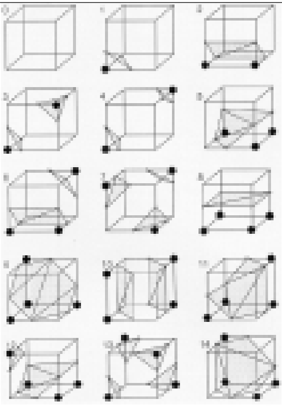


Figure 5. Triangulated Cubes.

© Lorensen, 1987

Marching cubes

- Indexation des cubes :
  - 8 bits par cube
  - Table donnant les arêtes intersectées
- Interpolation linéaire sur les arêtes pour calculer les points d'intersection

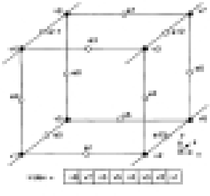


Figure 4. Cube Numbering

© Lorensen, 1987

Calcul des normales

- Les normales sont calculées indépendamment
  - Lisser la surface obtenue
  - Une normale par triangle
- Calcul :
  - Gradient de la fonction : normal à l'iso-surface
  - Calcul du gradient aux sommets du cube
  - Normalisation : normale aux sommets du cube
  - Interpolation linéaire : normales aux sommets des triangles
  - Interpolation linéaire : normales sur les triangles
- Besoin de 4 tranches en mémoire

Gradient aux sommets du cube

- Estimation par différences centrales :

$$G_x(i,j,k) = \frac{D(i+1,j,k) - D(i-1,j,k)}{\Delta x}$$
$$G_y(i,j,k) = \frac{D(i,j+1,k) - D(i,j-1,k)}{\Delta y}$$
$$G_z(i,j,k) = \frac{D(i,j,k+1) - D(i,j,k-1)}{\Delta z}$$

Marching-cubes : accélérations

- Cohérence du modèle :
  - On ré-utilise les calculs des cubes précédents
- Subdivision hiérarchique
  - Octrees
- Élimination de certaines ambiguïtés
  - Plus de cubes, moins d'erreurs

Marching-cubes : problèmes

- Beaucoup de polygones :
  - Même ordre de grandeur que le nombre de voxels
  - Stockage, affichage…
    - Parfois plus cher de stocker les polygones que de stocker le volume !
- Gros problèmes liés à la précision
  - Sous-échantillonnage
  - Artefacts

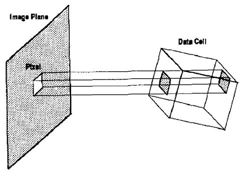
Rendu à base de pixels

- Lancer de rayons simplifié
  - Sans rebonds
  - Modèle simplifié de la lumière

$$I = I_a K_a + K_d \sum \left(\vec{n} \cdot L_j\right) I_j$$

  - Interpolation dans chaque cellule traversée
  - Calcul de l'opacité et de l'intensité
  - Arrêt quand opacité = 1

Rendu à base de pixels



$$I = \int_x \int_y \int_z \left[f(d) O(S) \left(K_a I_a + K_d \sum I_i (\vec{n} \cdot L_i) \right) + \left(1 - f(d)\right) bg \right] dxdydz$$

Rendu à base de pixels

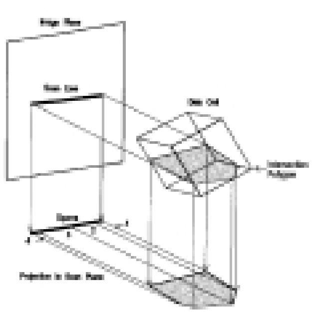
$$I = \int_x \int_y \int_z \left[f(d) O(S) \left(K_a I_a + K_d \sum I_i (\vec{n} \cdot L_i) \right) + \left(1 - f(d)\right) bg \right] dxdydz$$

- f : atténuation avec la distance
- O : opacité. Dépend de S (le champ)
- K_d : dépend de S.
- n : normale ; définie par le gradient de S
- bg : couleur du fond

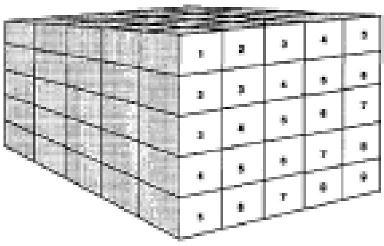
Intégration

- Calcul de l'intégrale cellule par cellule
- Valeurs aux coins de la projection du pixel sur la cellule (par interpolation)
- Dans une cellule :
  - Intégration pas à pas (I et O)
  - Arrêt quand O=1
- Si cellule se projette entièrement dans un pixel :
  - Valeurs moyennes pour la cellule,
  - Multiplié par le rapport aire projetée/aire du pixel

Optimisations

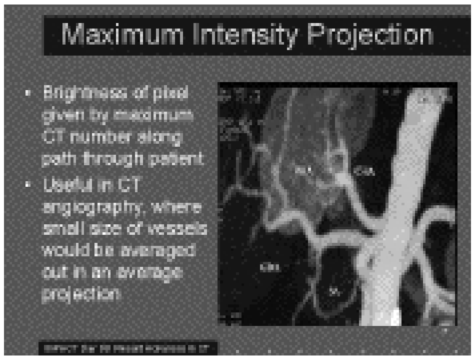


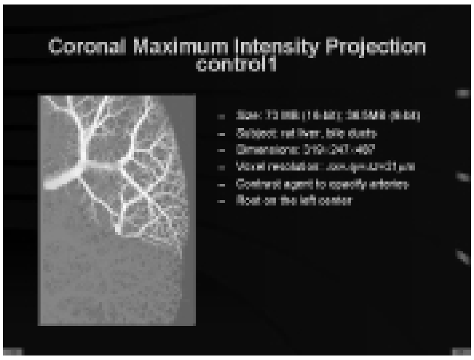
Parallélisation

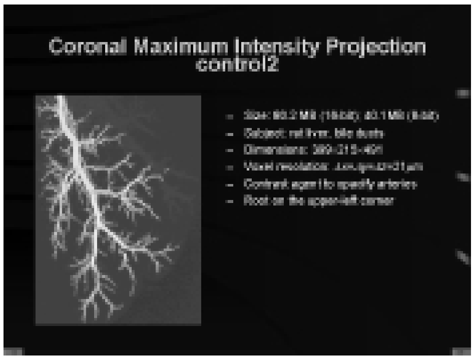


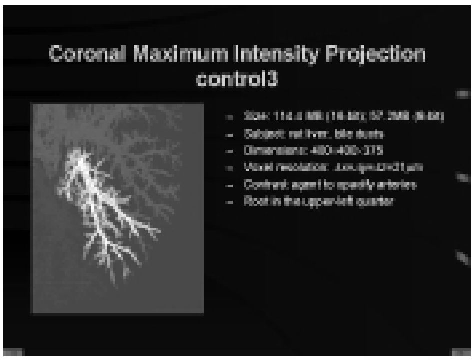
Traitement dans l'ordre. Les cellules de même numéro ne se recouvrent pas, donc elles peuvent être traitées simultanément.

Cas particulier : MIP









Par le volume : *coherent projection*

- Projection des cellules :
  - Même projection pour toutes les cellules
  - Rendu par polygones (de 1 à 7 pg par cellule)
  - Calcul des valeurs aux sommets des polygones
  - Interpolation entre les sommets
    - Faite par la librairie graphique
- Projection, *puis* interpolation
- Approximatif, mais rapide

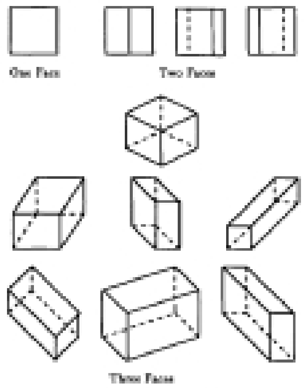


Figure 1: Cell Projections

© Wilhelms 1991

Par le volume : *splatting*

- Extraire la contribution de chaque cellule
  - Chaque cellule contribue à différents pixels

$$I = \iiint \sum S(x,y,z)K(x-u,y-v,z-w)\,dudvdw$$

$$I(x,y,z) = \sum_{D \in volume} S(D)K(x-D_x,y-D_y,z-D_z)$$

$$I(x,y,z) = \sum_{D \in volume} contribution_D(x,y,z)$$

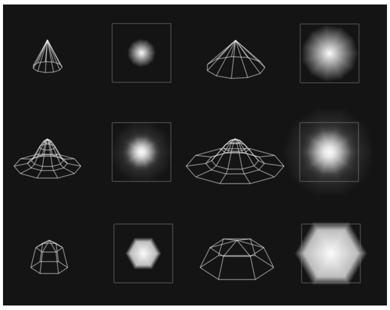
footprint d'une cellule

$$contribution_D(x,y) = \int_{-\infty}^{+\infty} S(D)K(x-D_x,y-D_y,w)\,dw$$

$$contribution_D(x,y) = S(D)\int_{-\infty}^{+\infty} K(x-D_x,y-D_y,w)\,dw$$

$$fp(x,y) = \int_{-\infty}^{+\infty} K(x,y,w)\,dw$$

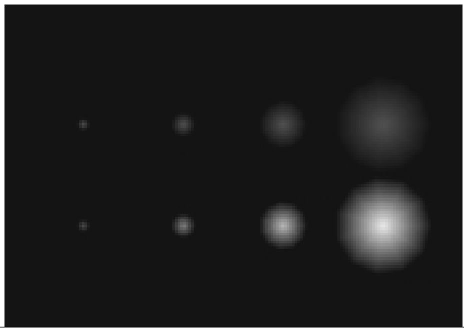
- Empreinte de la cellule sur chaque pixel
  - *(x,y)* déplacement par rapport à la projection du centre
- Noyau approximé, pré-échantillonné



© Laur, 1991

- Noyaux Gaussiens approximés par des polygones
  - Utilise le hardware
  - Qualité de l'approximation variable

Splatting hiérarchique

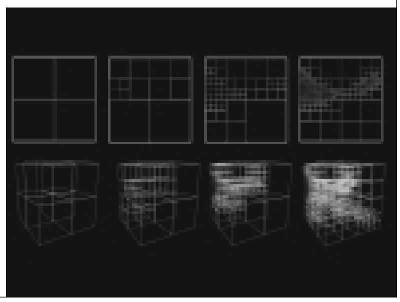


© Laur, 1991

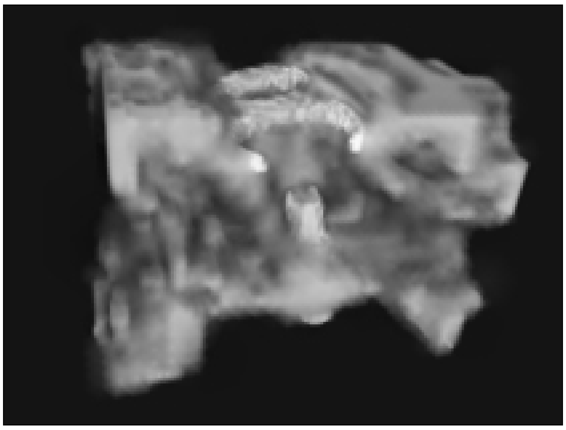
Décomposition hiérarchique

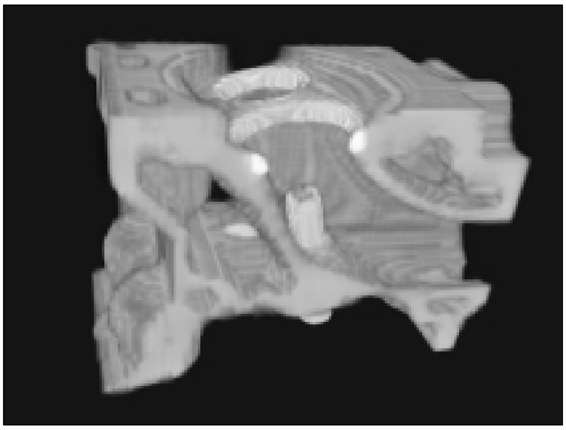
•Un noyau pour chaque cellule

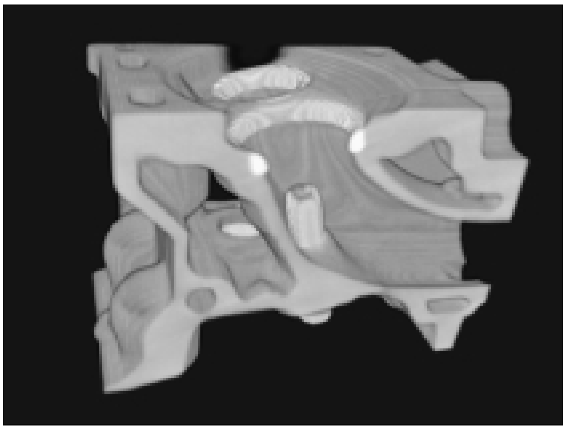
•Décomposition adaptable selon la précision/la rapidité souhaitée



© Laur, 1991







Utilisation du hardware

- Textures 3D :
  – Construire une texture 3D à partir des données
  – Utiliser la texture 3D pour l'affichage
    - Plans parallèles semi-transparents texturés
      – De l'arrière vers l'avant
    - C'est la carte qui fait tout
      – Conversion de coordonnées, extraction de textures...
    - On peut rajouter de l'ombrage

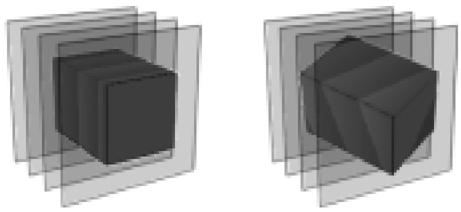
Construction d'une texture 3D

- La texture contient une intégration des données entre deux plans
  – Dépend de l'intervalle Δ entre deux plans
  – Cet intervalle doit être connu à l'avance
- Pour chaque cellule de données :
  – Calculer opacité et couleur
  E = couleur ponctuelle (*cf.* fonction de transfert)
  O_1 = opacité ponctuelle (*cf.* fonction de transfert)

$$\alpha = \ln\left(\frac{1}{1-O_1}\right) \qquad C = \frac{1-e^{-\alpha\Delta}}{\alpha\Delta}E \qquad O = 1-e^{-\alpha\Delta}$$

  – Stocker (C,O) dans texture 3D (R,G,B,A)

Affichage



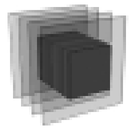
© Wilson, 1994

- « Bloc » de textures
- Tranches qui coupent le bloc
- Coordonnées des tranches ?

Affichage

- La texture tourne, les tranches restent fixe
  - Le bloc des tranches doit pouvoir contenir toutes les rotations du cube de texture
  - Conversion entre coord. (x,y,z) (fixes) et coordonnées de textures (s,t,r) (variables)
  - La carte fait l'interpolation

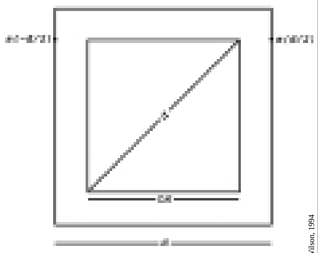
Cas particulier simple



- Cube ($n_x\Delta_x$, $n_y\Delta_y$, $n_z\Delta_z$)
  - n_x nbre échantillons
  - Δ_x intervalle d'éch.
  - s varie de 0 à 1sur le cube

$$s\left(-\tfrac{d}{2}\right)=\frac{1}{2}\left(1-\frac{d}{n_x\Delta_x}\right)$$

$$s\left(\tfrac{d}{2}\right)=\frac{1}{2}\left(1+\frac{d}{n_x\Delta_x}\right)$$



Cas particulier simple...

- Pb : les textures doivent avoir une taille 2^n
  - $N_x=2^k > n_x$
  - $L_x=N_x\Delta_x$
  - s varie maintenant de 0 à n_x / N_x

$$s(x)=\frac{x+\frac{1}{2}n_x\Delta_x}{N_x\Delta_x}$$

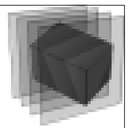
Cas particulier simple

- D : matrice de changement d'échelle d
- S : matrice de changement d'échelle (d/L_x, d/L_y, d/L_z)
- T : translation de ($n_x /2N_x$, $n_y /2N_y$, $n_z /2N_z$)

$$(s,t,r)=TSD^{-1}(x,y,z)$$

- Texture de taille (N_x, N_y, N_z)

Cas général



- Point de vue modifié par une rotation R
  - On inverse la rotation, et on est ramené au cas précédent :

$$(s,t,r)=TSD^{-1}R^{-1}(x,y,z)$$

  - D commute avec R

Cas général

$$(s,t,r)=TSR^{-1}D^{-1}(x,y,z)$$

- Évaluer aux coins du cube : (±1/2d, ±1/2d, ±1/2d)
- Égale évaluer TSR^{-1} sur (± 1/2,±1/2,±1/2)
- Transformation faite par la *texture matrix* (OpenGL)
- Rotation, puis échelle, puis translation

Considérations matérielles

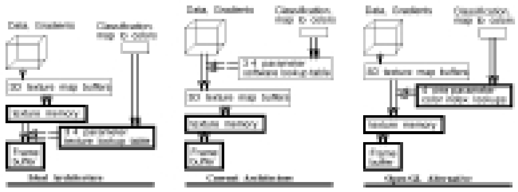
- Nombre optimal de plans :
  - Au moins 1 plan par échantillon
  - Donc d/Δ_z
    - 110 plans pour 64^3
  - Qualité proportionnelle au nombre de plans
  - Temps de rendu aussi
- Interpolation tri-linéaire
  - Remplace plus-proche-voisin (par défaut)
  - Plus lent (50 %), mais meilleure qualité, plus lisse

Considérations matérielles

- Précision des calculs :
  - 8 bits par canal ou 12 bits ?
  - 2 fois plus rapide, 50 % de coût mémoire en moins
  - Problèmes d'arrondi :
    - Erreurs d'arrondi au maximum de 1/2*1/256
    - Devient problématique si on a plusieurs centaines de tranches
    - Plus visible avec des données homogènes
- Mémoire :
  - 2 Mo de texture 3D : 64^3, 4 canaux, 12 bits

Ombrage des surfaces

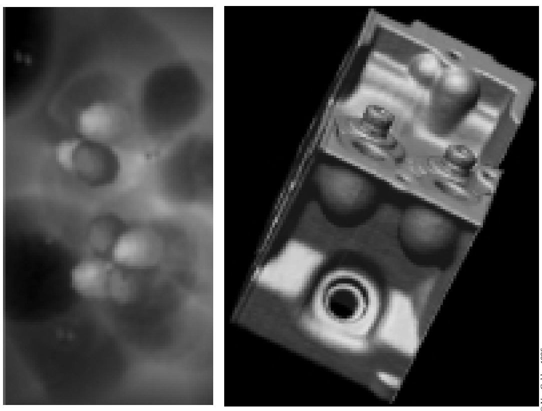
- Identification des cellules proches d'une frontière de matériau
- Calcul du gradient
  - Discrétisé, *g* valeurs possible
- Table des couleurs :
  - *m* matériaux, *g* valeurs de gradient
  - $(m+1)g$ entrées
- Ordre des opérations :
  - Interpoler les données ou les couleurs ?



Légende :

Une fois — software

À chaque image — hardware



Bibliographie

- Marching cubes :
  - Lorensen, W. E. et Cline, H. E., Marching cubes: a high resolution 3D surface construction algorithm, *Computer Graphics*, 21(4) (*Siggraph '87*), p.163-169
- Rendu à base de pixels :
  - Upson, C. et Keeler, M., V-Buffer : Visible Volume Rendering, *Computer Graphics*, 22(4) (*Siggraph '88*), p. 59-64
  - Drebin, R.A., Carpenter, L. et Hanrahan, P., Volume Rendering, *Computer Graphics*, 22(4) (*Siggraph '88*), p. 65-74
- *Coherent Projection* :
  - Wilhelms, J. et Van Gelder, A., A Coherent Projection Approach for Direct Volume Rendering, *Computer Graphics*, 25(4) (*Siggraph '91*), p. 275-284
- *Splatting* :
  - Westover, L., Footprint Evaluation for Volume Rendering, *Computer Graphics*, 24(4) (*Siggraph '90*), p. 367-376
  - Laur, D. et Hanrahan, P., Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering, *Computer Graphics*, 25(4) (*Siggraph '91*), p. 285-288
- Textures 3D :
  - Wilson, O., Van Gelder, A. et Wilhelms, J., Direct Volume Rendering via 3D Textures, Tech. Report de l'Université de Californie-Santa Cruz (UCSC-CRL-94-19)
  - Van Gelder, A. et Kwansik, K., Direct Volume Rendering with Shading via 3D Textures, *ACM Symposium on Volume Visualisation '96*, p. 23-30.