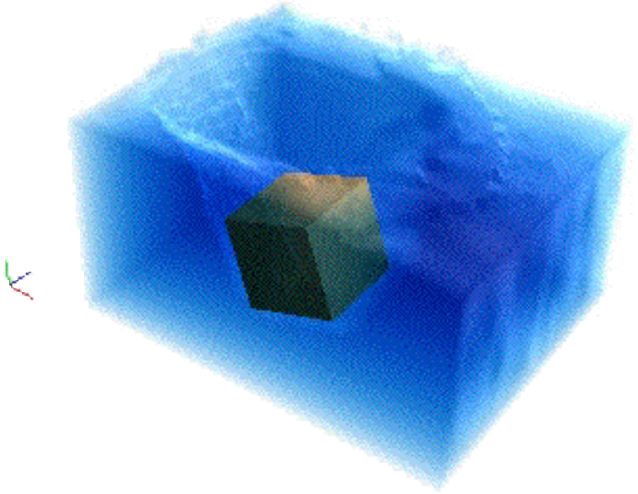
Rendu hardware d'eau avec éclairage local



Olivier Génevaux

IGG

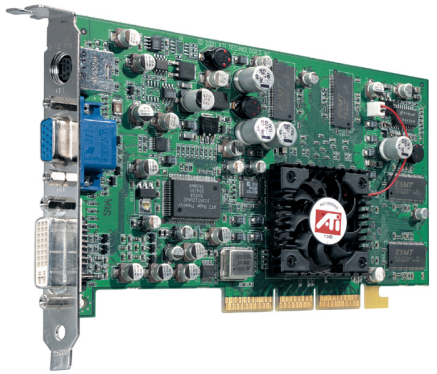
Simulation & Visualisation

LSiiT

UMR 7005 CNRS UL

Objectif et données

- Objectif:
  - Prototypage
  - Rendu rapide le plus réaliste possible
- Données:
  - Grille 3D stockant la présence du fluide
  - Obtenues par simulation numérique (Navier-Stokes)
- Matériel:
  - ATI Radeon 8500 (PS.1.4)

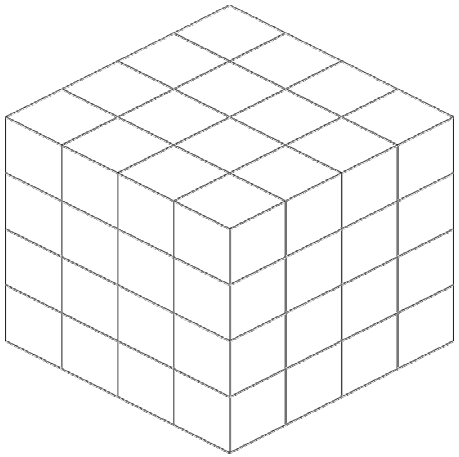


Problématique

Pas de surface polygonale

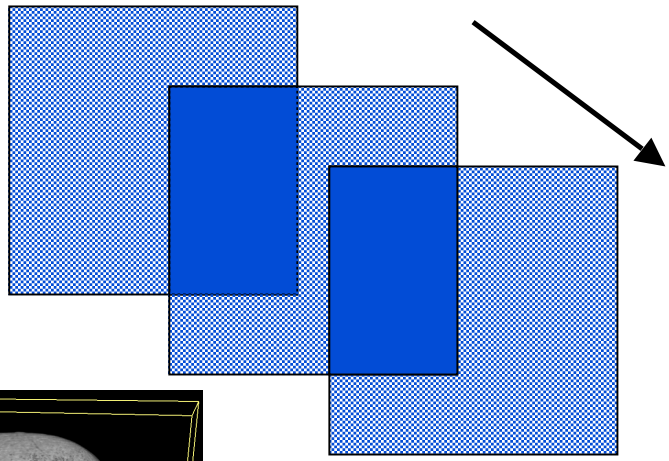
Rendu volumique

Eclairage de surface?



Rendu volumique simple

- Back to front blending
  - ↳ Effet d'absorption
  - ↳ Pas d'illumination



Ajout de l'éclairage local

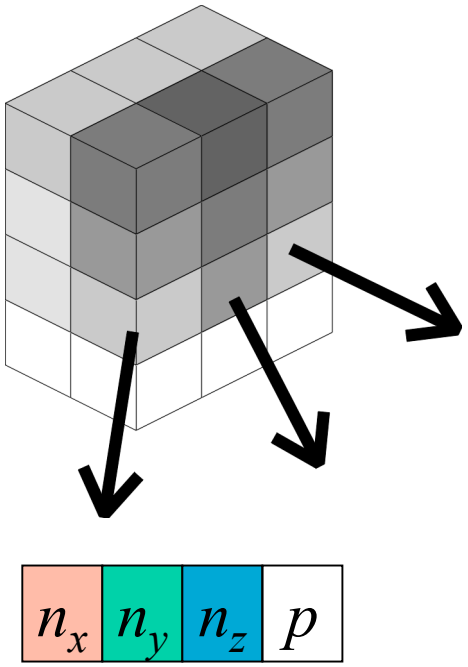
Nécessité d'une normale

- Calcul par gradient de la présence de fluide
- Valide sur la frontière du fluide
- Précalculé extrêmement rapidement

$$n = \nabla p = \left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z} \right)$$

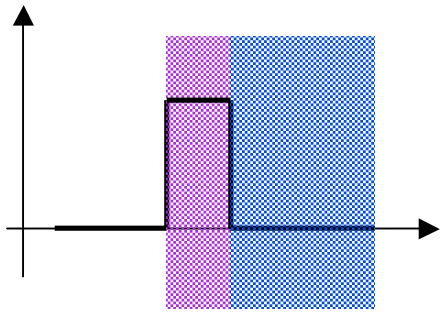
Données manipulées

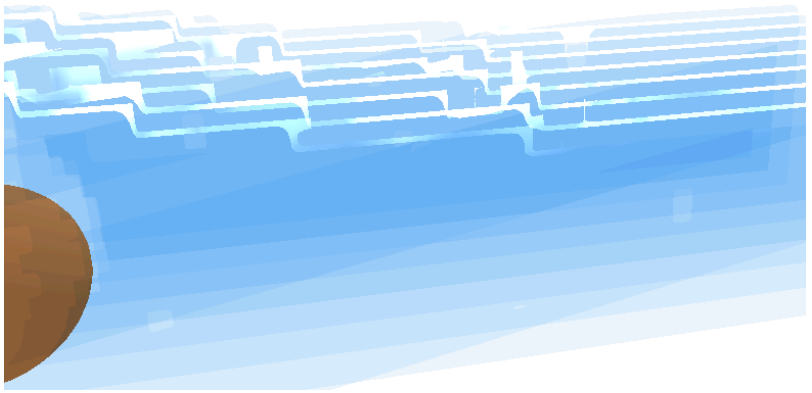
- Deux informations:
  - Présence fluide (retravaillée)
    - Scalaire
  - Normale
    - Vecteur 3D
- Stockage dans une seule texture 3D
  - Normale RGB
  - Présence A



Stratégie d'illumination locale

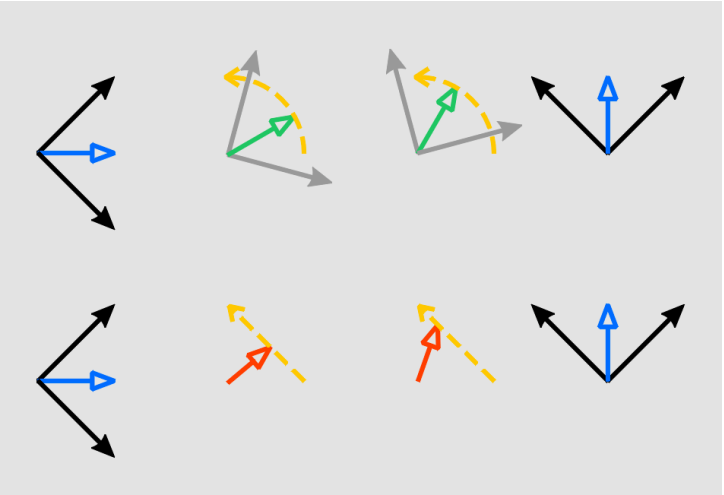
- Illumination de la frontière du liquide
  - Repérage grâce au facteur de présence





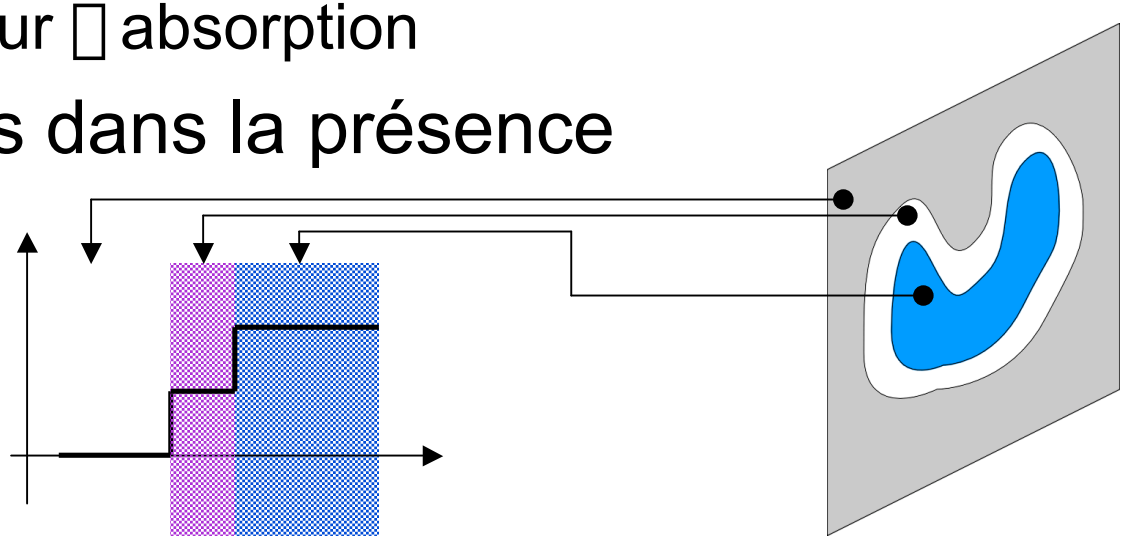
Modèle d'illumination

- Lambert
  - Limitations hardware
- Orientation source / pixel
  - Interpolation d'une orientation déduite aux sommets au lieu d'un changement de repère



Algorithme

- 3 cas
  - Extérieur pas de rendu
  - Frontière illumination locale + absorption
  - Intérieur absorption
- Encodés dans la présence



Blending pour compositer les tranches

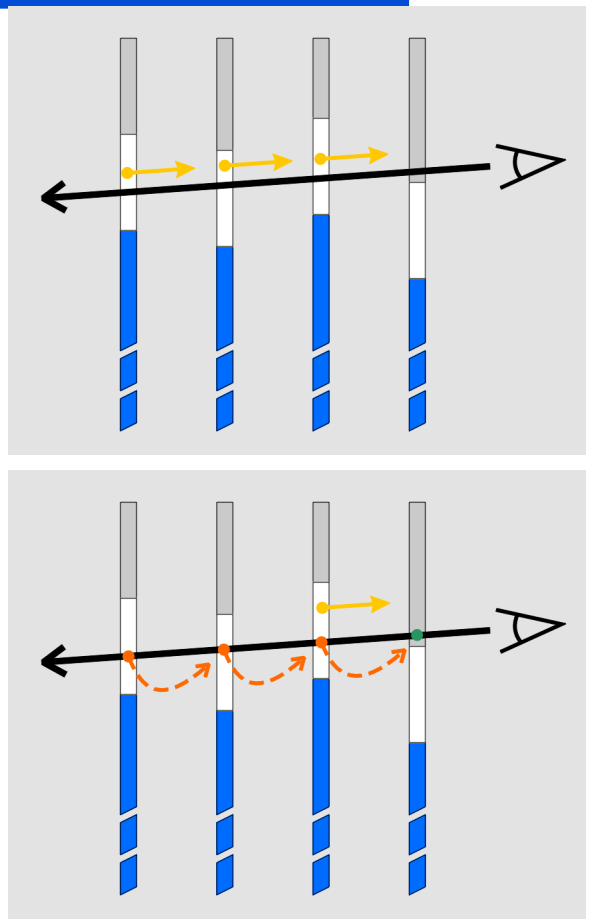
- Blending additif détourné
  - Couleur constante
  - Alpha encode l'illumination

$$d = \square_s \cdot s + c_s \cdot d$$

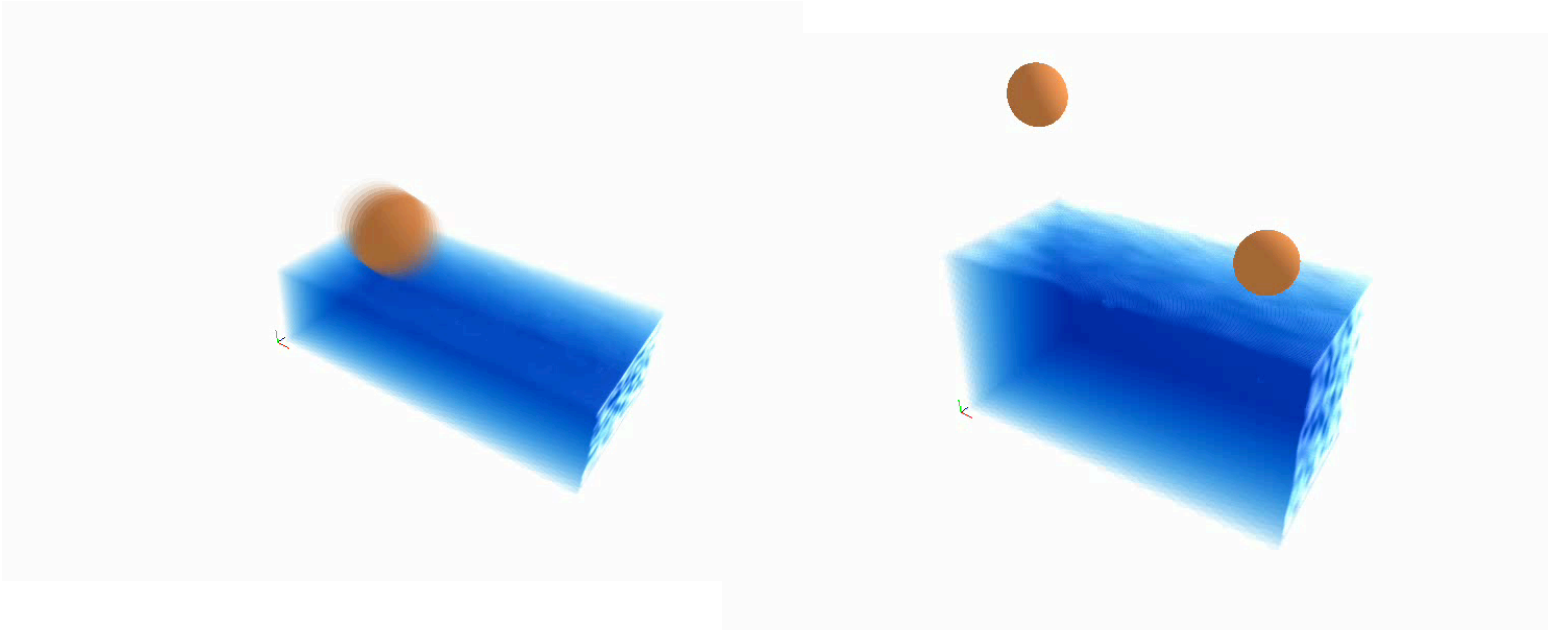
- Limitation
  - Lumière blanche (1 composante stockable)
  - Précision du frame buffer
    - 1/n slice absorbante

Problème des angles rasants

- Accumulation de l'illumination locale
  - Due à l'épaisseur de l'illumination
- Suppression en modulant l'illumination selon l'angle de vue
  - Double évaluation de la présence du fluide



Résultats



Limitations

- Illumination locale très approchée
  - Pas d'éclairage spéculaire
- Faible résolution des données
- Précision du frame buffer
- Lent si beaucoup de slices

Conclusion

- Objectif atteint: outil de prototypage
  - Bonne vision de la scène
  - Coût modeste en regard d'un vrai rendu réaliste
    - 4 - 5 sec par image (qualité max)
    - 20 sec par image pour une simulation de bonne taille
    - Coût quasi linéaire / qualité désirée

Questions

Code

```
// volumetric object ps

// t2                    -> light direction

ps.1.4

def c2, 0.0, 0.0, 0.0, 0.0
def c3, 0.0, 0.0, 0.0, 0.0          // ambient
def c4, 0.0, 0.3, 0.0, 0.0          // [texkill bias, surface depth bias, 0, 0]
def c6, 0.0, 0.0, 0.0, 0.8          // light intensity

texld r0, t0.xyz                    // tex sampling (real point)
texld r5, t3.xyz                    // tex sampling (forward point)
texcrd r4.xyz, t2                   // acquire interpolated light direction

mov r1, r0_bx2.a                    // expand presence
sub r2, r0_bx2.a, c4.r              // bias 4 texkill
mov r5, r5_bx2.a                    // expand presence
```

Code

```
phase

texkill r2                          // kill rendering if outside fluid

dp3_sat r2.a, r0_bx2, r4            // lambert lighting
sub_sat r2.r, r1.r, r5.r            // density gradient along ray

mul r0.a, r2.r, r2.a                // attenuate according 2 density gradient

mad r0.a, r0.a, c6.a, c3.a          // light (adjust intensity & ambient)

add r1.r, r1.r, c4.g                // surface bias
cnd r0.a, r1.r, c2, r0.a            // check if pixel is on surface

mov r0.rgb, c1                      // output absorbtion color
```