Subdivision de surfaces en temps réel sur CPU et GPU

GUITTENY Fabrice
IDIART Baptiste
LE GOFF Erwan
PONSONNET Olivier

Projet de Fin d'Etudes

31 Mars 2006

UNIVERSITÉ BORDEAUX 1 Sciences Technologies

Encadrants : Tamy Boubekeur - Pascal Guitton

Introduction

- Visualisation 3D de plus en plus réaliste
- Hardware graphique de plus en plus puissant

  → effectuer certains calculs sur le GPU

- Objectifs :
  - Utiliser cette puissance de calculs dans le cadre de la subdivision de surfaces en temps réel
  - Comparaison des implémentations de telles méthodes sur le CPU et sur le GPU

Plan

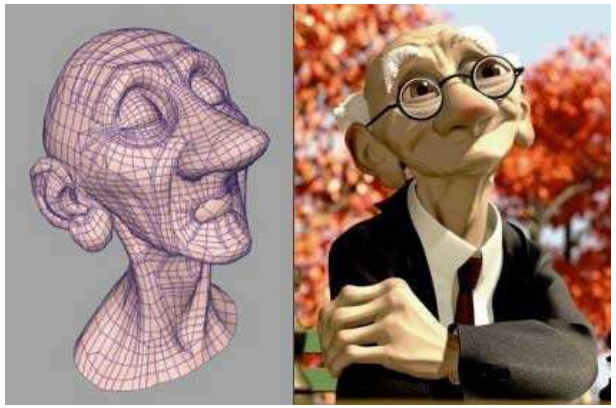
- Introduction au domaine d'application
  - Subdivision de surfaces
  - GPGPU
  - Extensions récentes d'OpenGL
- Descriptif du projet
  - Cahier des charges
  - Architecture du logiciel
  - Présentation de l'application
- Subdivision temps réel sur CPU
  - Méthode utilisée
  - Résultats obtenus
- Subdivision temps réel sur GPU
  - Méthode utilisée
  - Résultats obtenus

Introduction au domaine d’application

- **Subdivision de surfaces**
- GPGPU
- Extensions récentes d’OpenGL

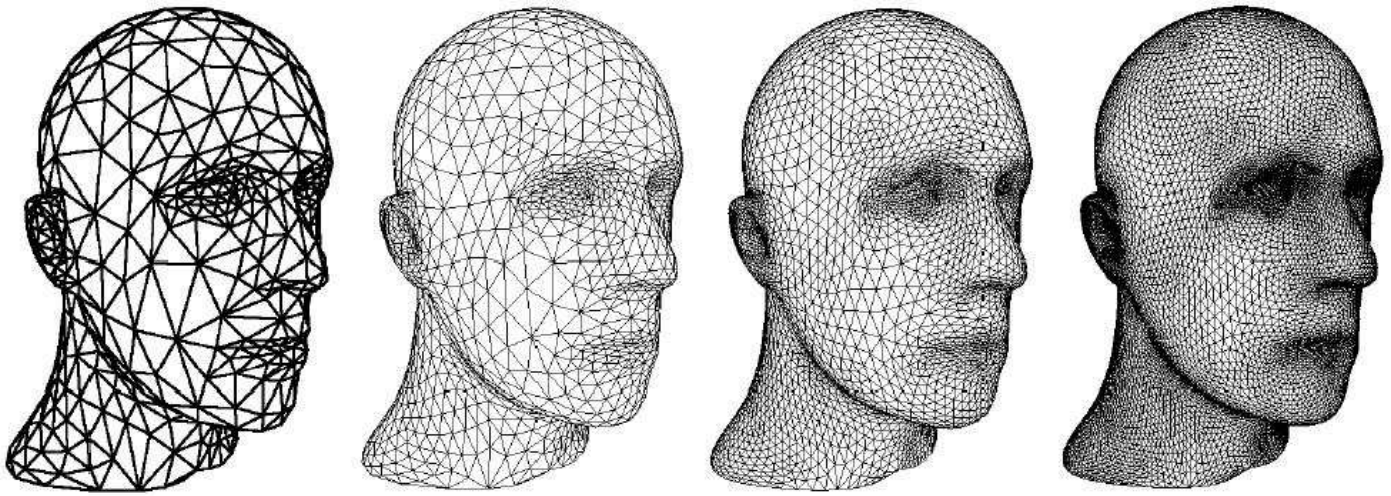
Subdivision de surfaces
Présentation

- Technique de modélisation géométrique permettant d'obtenir des surfaces lisses à moindre coût
- Mise en parallèle avec les surfaces paramétriques (NURBS…) ou les surfaces implicites équipotentielles
- Méthode la plus utilisée dans l'industrie actuellement



Subdivision de surfaces
Principe

- Principe général : on raffine une surface décrite par un maillage polygonal en augmentant récursivement la taille de ce maillage jusqu'à tendre vers une surface lisse appelée surface limite
- A chaque itération, un lissage est effectué sur la surface afin d'améliorer le rendu



Subdivision de surfaces
Différents schémas

- → Plusieurs techniques possibles avec des propriétés respectives selon :
  - Le type du maillage
  - La continuité de la surface générée
  - Le réalisme du rendu
  - Le temps de calcul nécessaire
  - L'adaptativité de la subdivision

- Les plus connus : Loop, Catmull-Clark, Butterfly, Doo-Sabin, Kobbelt, $\sqrt{3}$…

Subdivision de surfaces
Exemple

Maillage de contrôle

(1292 polygones)

Maillage raffiné après

2 passes de subdivision

(20672 polygones)

Subdivision de surfaces
Intérêts

- Représentation polygonale → topologie quelconque et rapidité du rendu matériel
- Modèle haute résolution d'un objet généré après des subdivisions successives
- Images de haute qualité
- Facilité d'utilisation et d'implémentation
- Faible coût (stockage du maillage…)
- Explicite
- Animation d'un objet facilitée
- Adaptativité pour certains schémas
- Technique présente dans la plupart des logiciels de modélisation (3DSMax, Maya, Blender, Lightwave…)

Subdivision de surfaces
Inconvénients

- Difficulté d'utilisation pour des applications temps réel
- Singularité de certains objets (arêtes vives, semi-vives notamment) complexifie les calculs

Introduction au domaine d’application

- Subdivision de surfaces
- **GPGPU**
- Extensions récentes d’OpenGL

GPGPU
Définition

- **G**eneral **P**urpose computing on **G**raphics **P**rocessor **U**nits

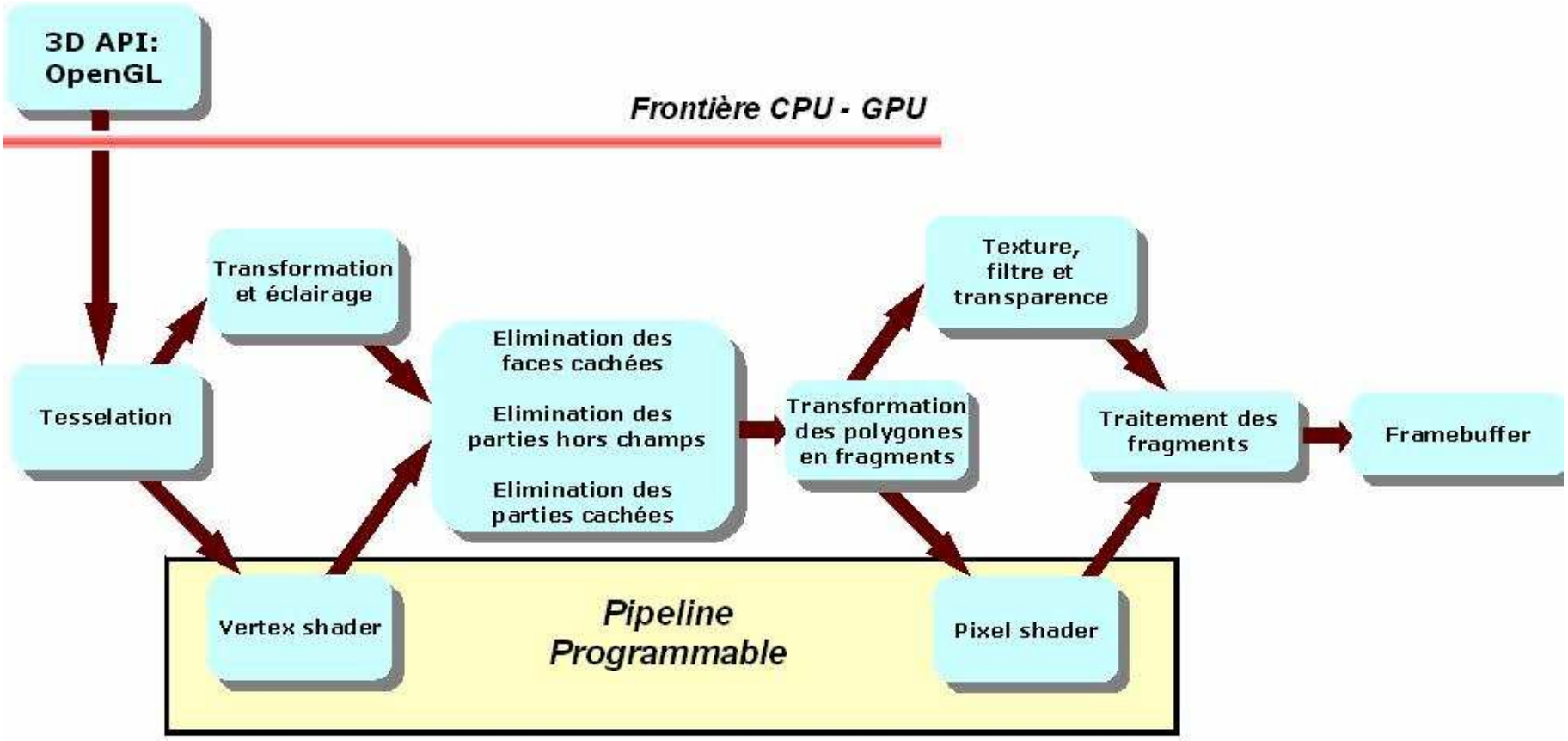
- Définition
  - Utilisation du GPU pour des tâches génériques

GPGPU
Quels intérêts ?

- Avantages
  - Parallélisme
  - Rapidité
  - En constante évolution

- Inconvénients
  - Programmation du GPU
  - Portabilité des applications difficile

GPGPU
Pipeline graphique



Introduction au domaine d’application

- Subdivision de surfaces
- GPGPU
- **Extensions récentes d’OpenGL**

Extensions récentes d'OpenGL

- Eviter les transferts GPU-CPU

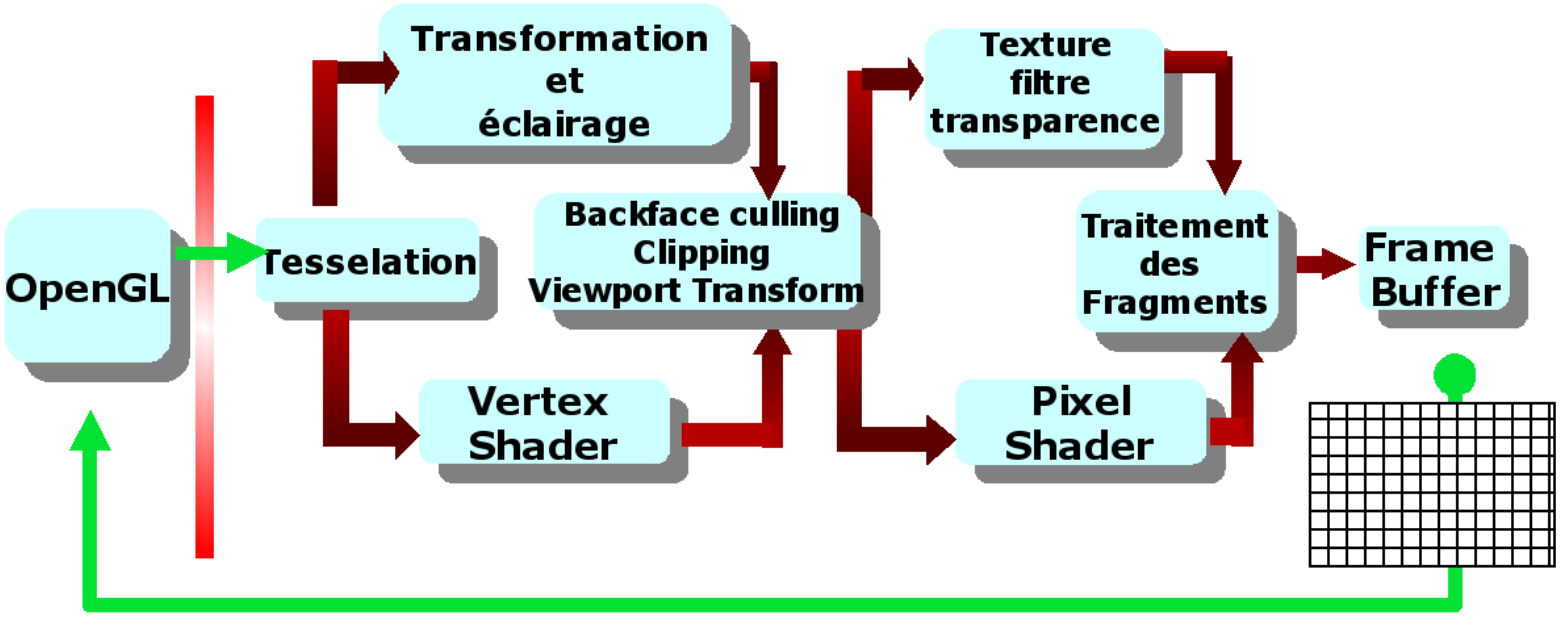
extensions récentes qui utilisent au maximum la mémoire graphique :

- Frame Buffer Object : FBO
- Pixel Buffer Object : PBO
- Vertex Buffer Object : VBO

Extensions récentes d'OpenGL

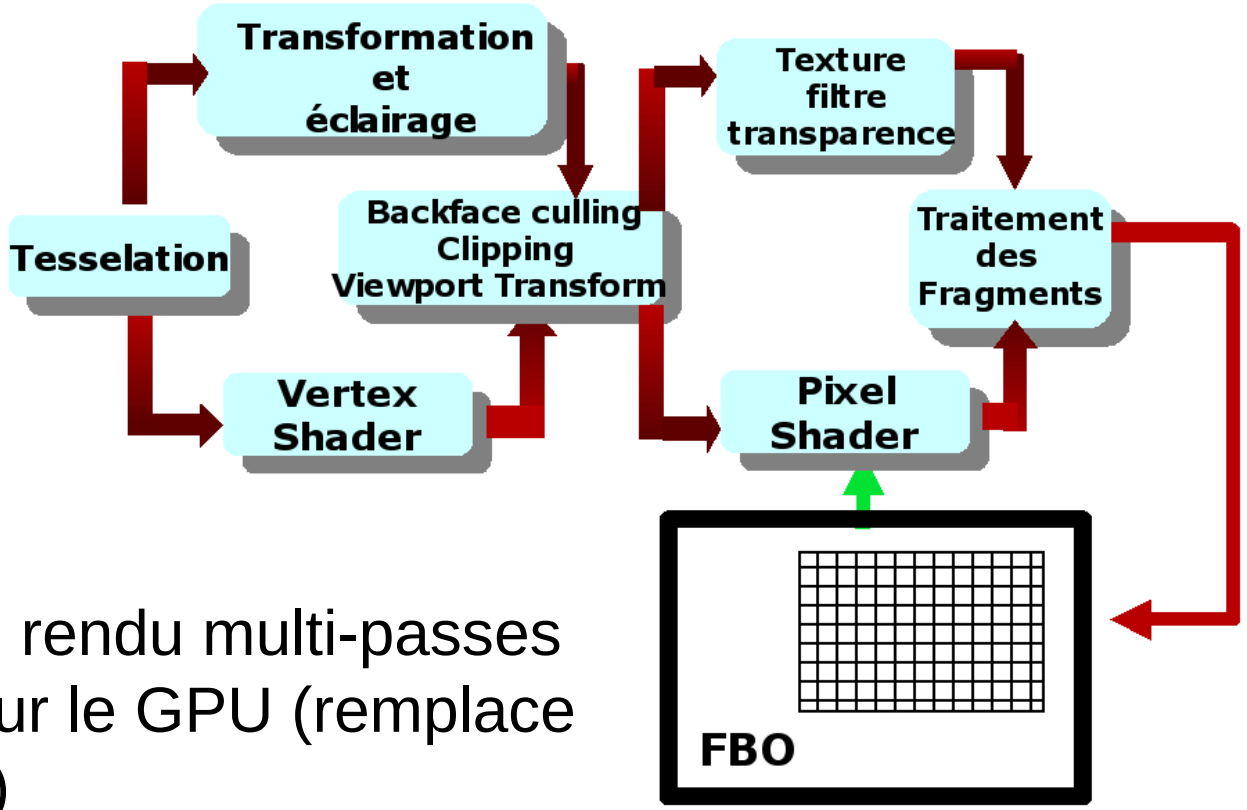
Render to Texture (1/2)

- Idée : copier le *framebuffer* dans une texture et utiliser celle-ci pour une nouvelle passe de rendu



Extensions récentes d'OpenGL

Render to Texture (2/2) : FBO

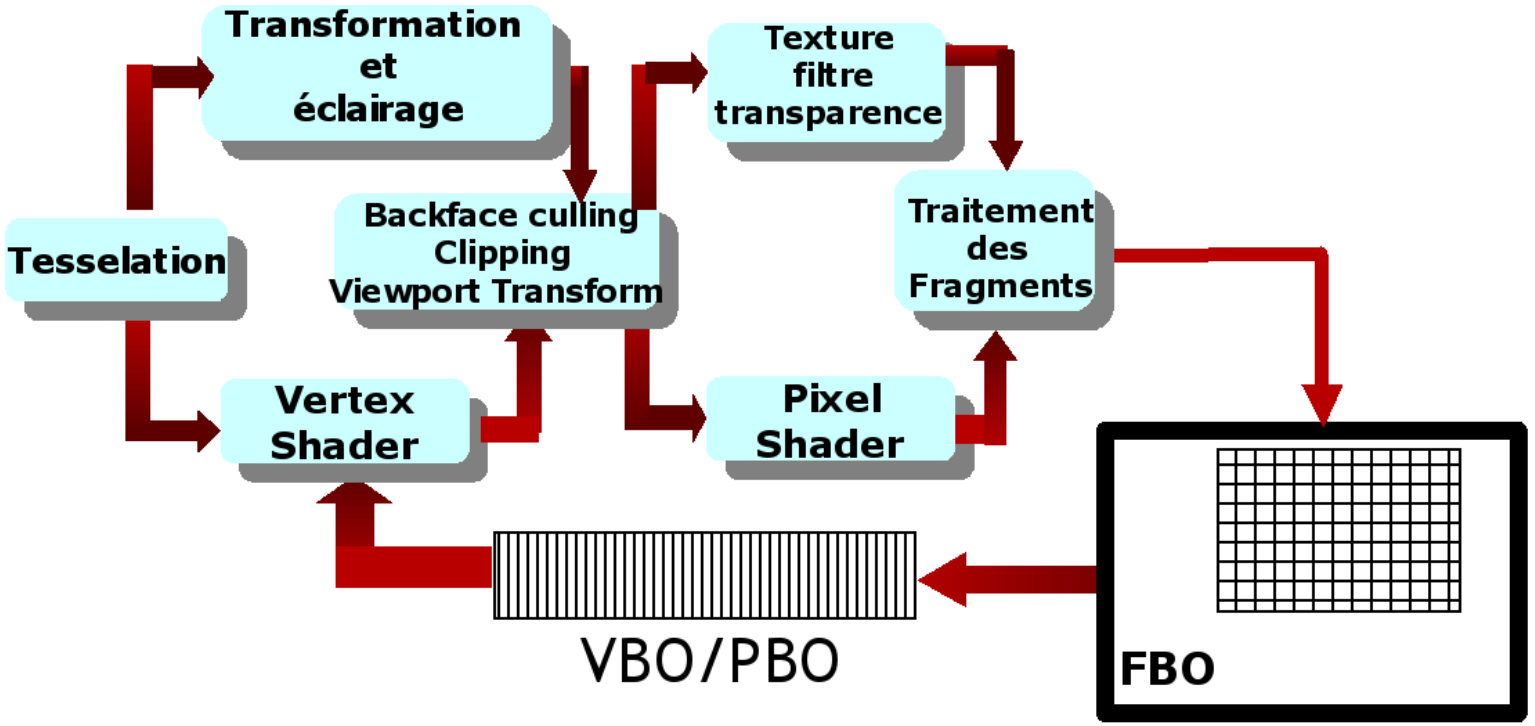


- Effectuer du rendu multi-passes en restant sur le GPU (remplace les *pbuffers*)

Extensions récentes d'OpenGL

Render to Vertex Array

- Combinaison des Frame Buffer Object avec les Pixel Buffer Object pour utiliser une texture comme maillage final

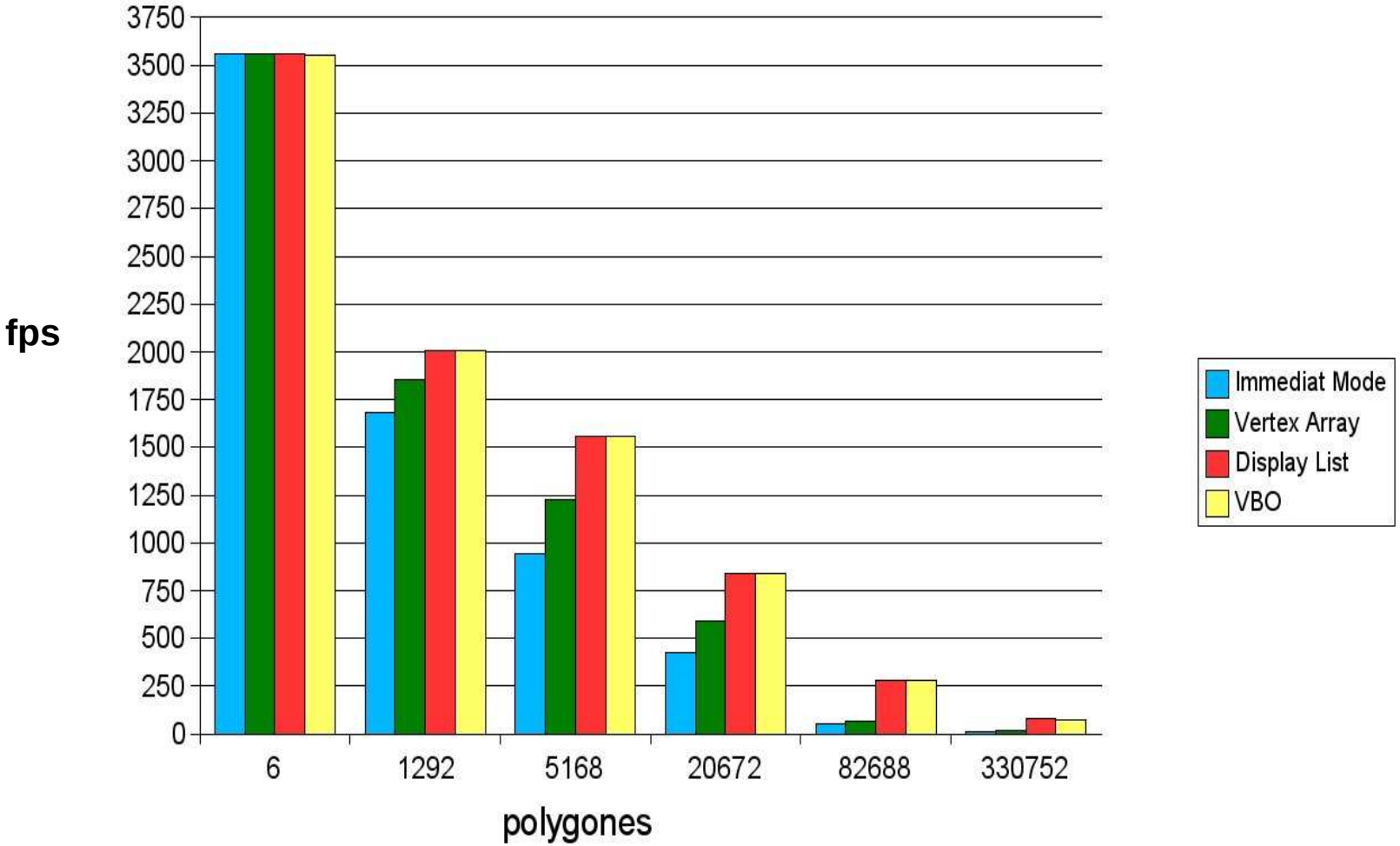


Extensions récentes d'OpenGL
Vertex Buffer Object (1/2)

- Utilisés à la fois dans l'application CPU et dans l'application GPU
- Parfaitement associés au PBO
- Chargé dans la mémoire graphique
  - Rapidité d'affichage
  - Structure pouvant être dynamique

Extensions récentes d'OpenGL
Vertex Buffer Object (2/2)

- Comparaison avec les autres méthodes d'affichage



Plan

- Introduction au domaine d'application
  - Subdivision de surfaces
  - GPGPU
  - Extensions récentes d'OpenGL
- **Descriptif du projet**
  - **Cahier des charges**
  - **Architecture du logiciel**
  - **Présentation de l'application**
- Subdivision temps réel sur CPU
  - Méthode utilisée
  - Résultats obtenus
- Subdivision temps réel sur GPU
  - Méthode utilisée
  - Résultats obtenus

Descriptif du projet
Cahier des charges

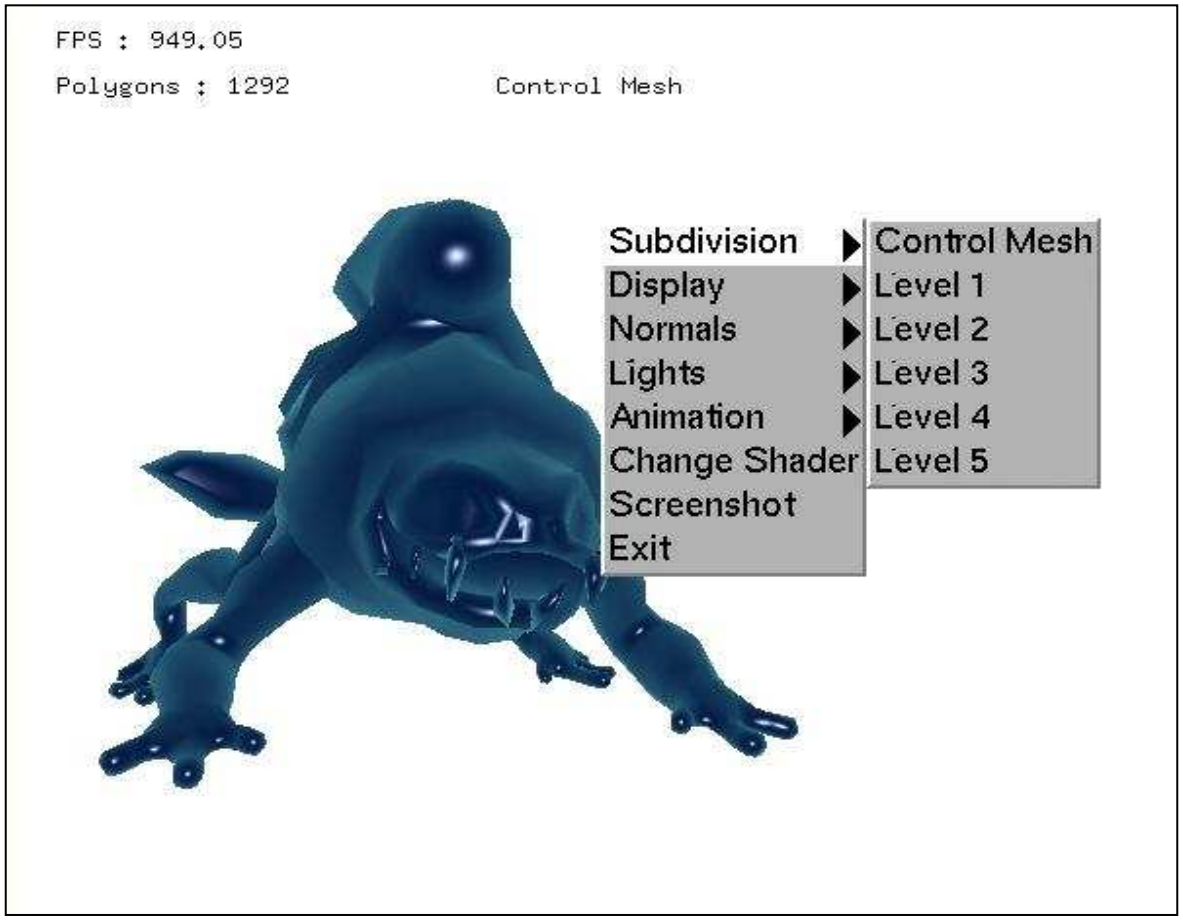
- Parser .obj
- Subdivision temps réel sur CPU
- Subdivision temps réel sur GPU
- Affichage des performances
- Comparaison des deux approches

Descriptif du projet
Architecture du logiciel

- Classe Vertex :
  - définit les sommets
- Classe Face :
  - liste d'indices de sommets
- Classe Model :
  - définit le maillage polygonal comme une liste de faces et une liste de sommets
- Classe CatmullRenderer :
  - effectue la subdivision CPU à partir d'un maillage

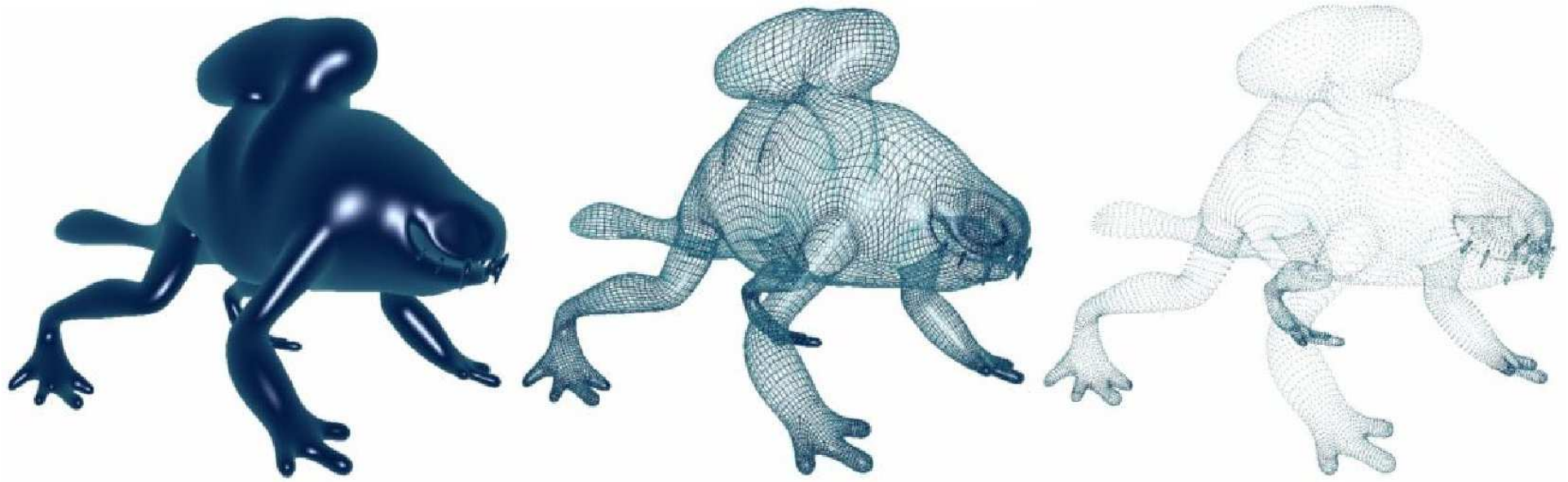
Descriptif du projet
Présentation de l'application (1/4)

- Interface du programme



Descriptif du projet
Présentation de l'application (2/4)

- 3 modes de visualisation :



Descriptif du projet
Présentation de l'application (3/4)

- Changement du niveau de subdivision

Descriptif du projet
Présentation de l'application (4/4)

- Changement de shaders

Plan

- Introduction au domaine d'application
  - Subdivision de surfaces
  - GPGPU
  - Extensions récentes d'OpenGL
- Présentation du projet
  - Cahier des charges
  - Architecture du logiciel
  - Présentation de l'application
- **Subdivision temps réel sur CPU**
  - **Méthode utilisée**
  - **Résultats obtenus**
- Subdivision temps réel sur GPU
  - Méthode utilisée
  - Résultats obtenus

Subdivision temps réel sur CPU
Méthode utilisée (1/4)

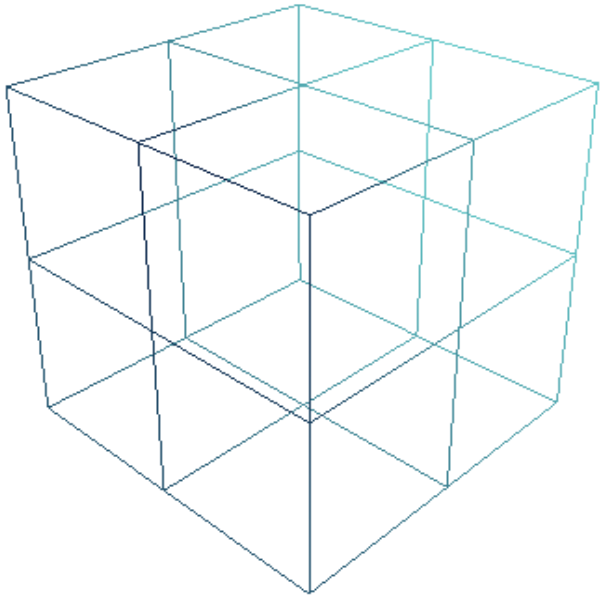
- Tiré des travaux de Joe Warren et Scott Schaefer

- But : effectuer une subdivision de type Catmull-Clark en temps-réel sur CPU

- Se déroule en deux phases

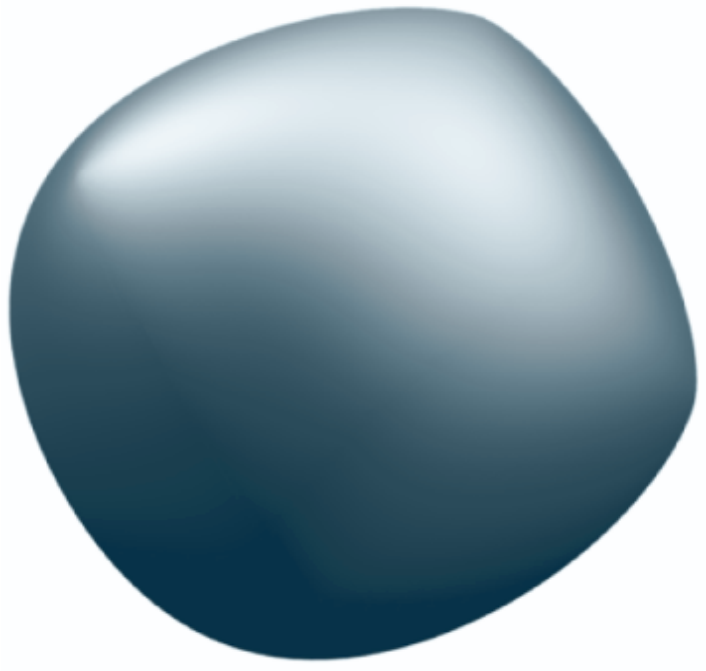
Subdivision temps réel sur CPU
Méthode utilisée (2/4)

- Première phase : subdivision linéaire du maillage
  - Exemple avec un cube :



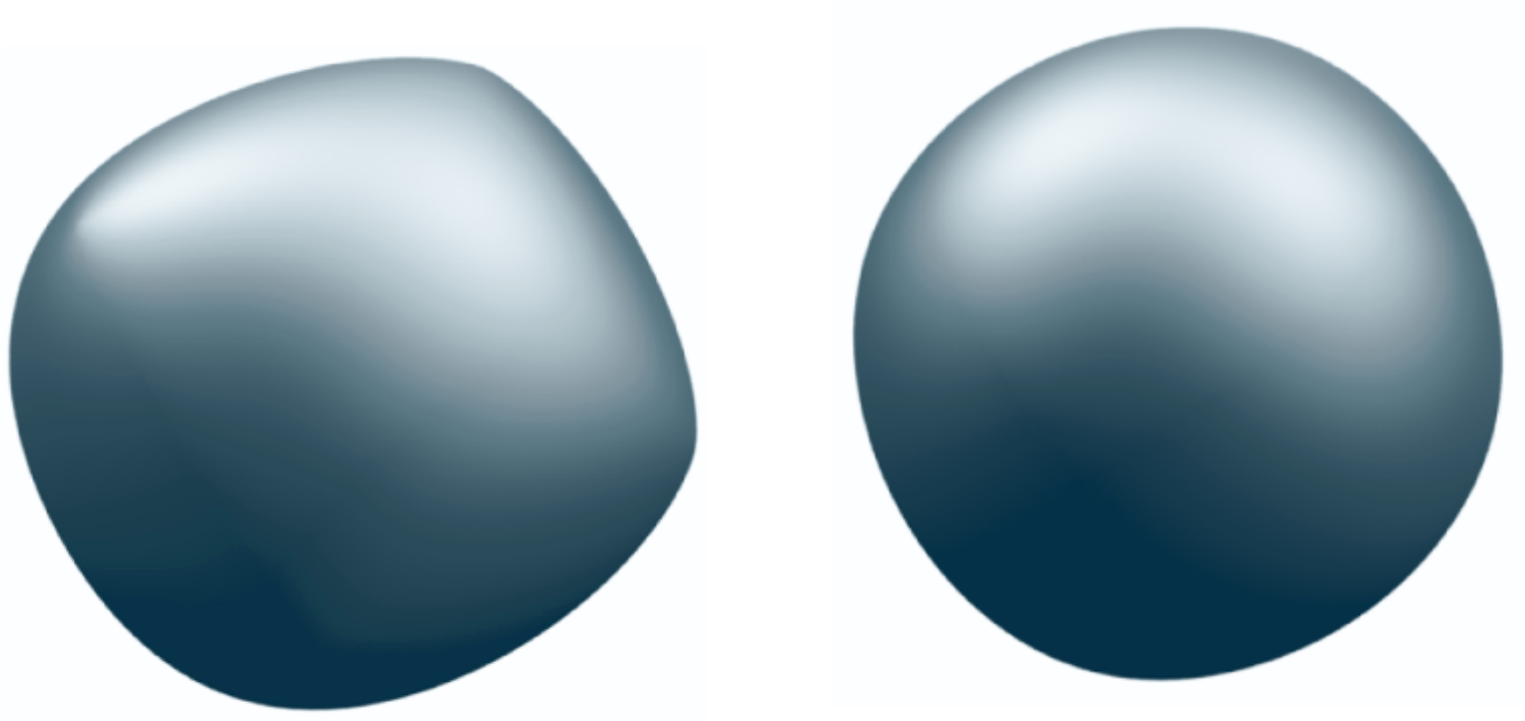
Subdivision temps réel sur CPU
Méthode utilisée (3/4)

- Deuxième phase : lissage par “moyennage”
  - Exemple avec un cube :

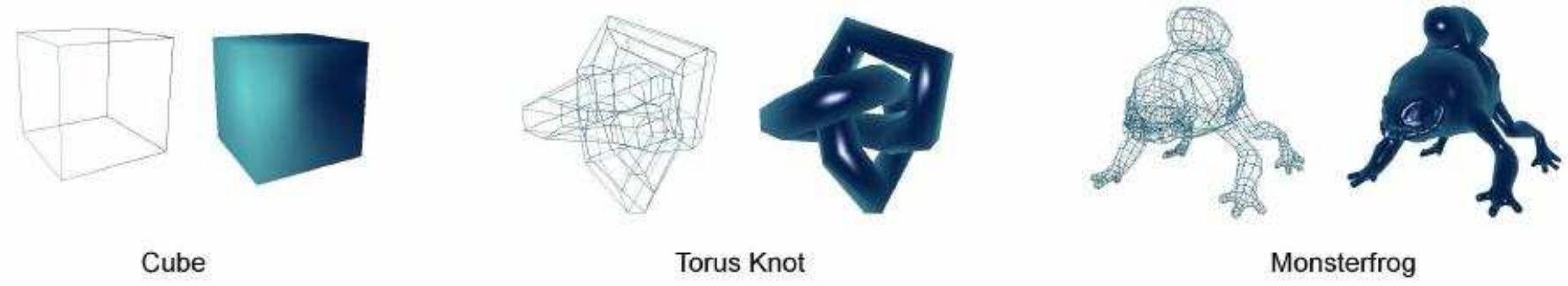


Subdivision temps réel sur CPU
Méthode utilisée (4/4)

- Correction de la deuxième passe
  - Exemple avec un cube :



Subdivision temps réel sur CPU
Résultats obtenus



	Control Mesh	Niveau 1	Niveau 2	Niveau 3	Niveau 4	Niveau 5
Cube	**3271,3** 6	**960,1** 24	**276,2** 96	**68,7** 384	**16,7** 1536	**3,6** 6144
Torus Knot	**1529,4** 96	**90,4** 384	**17,2** 1536	**3,7** 6144	**0,9** 24576	**0,2** 98304
Monsterfrog	**1257,7** 1292	**5,8** 5168	**1,1** 20672	**0,3** 82688	**0,07** 330752	**0,02** 1323008

FPS

Nombre de polygones

Subdivision temps réel sur CPU
Conclusion

- Subdivision en temps réel sur CPU → pas de garantie d'un taux de FPS raisonnable pour un niveau de subdivision élevé

- Pourtant, un modèle relativement simple et optimisé (structure *hashmap* de la STL, schéma non adaptatif, pas de gestion des singularités…)

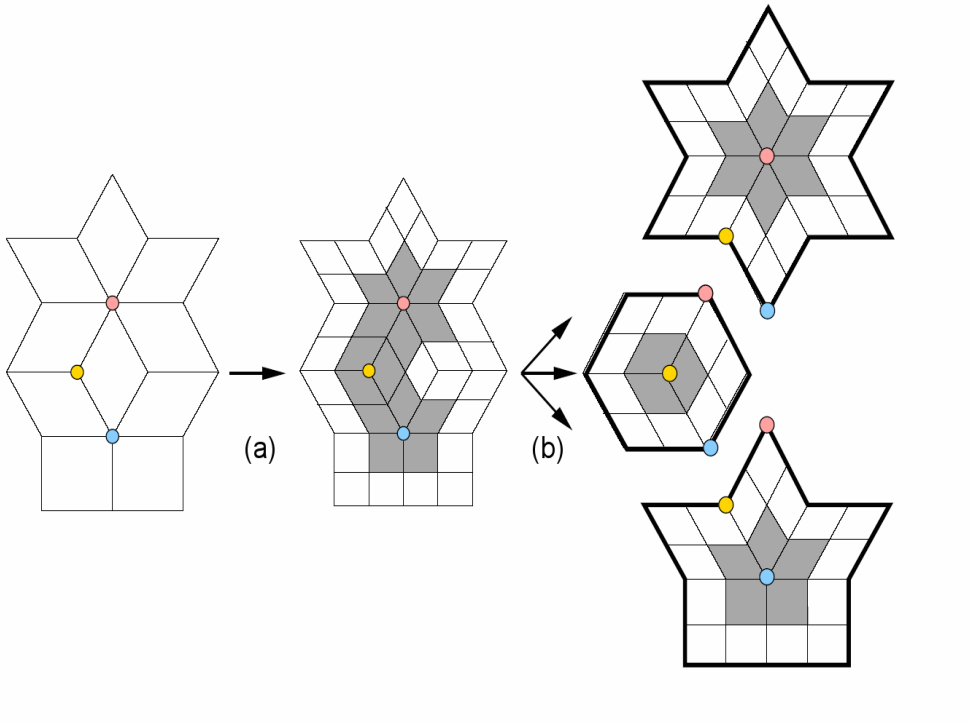
  → Nécessité de trouver d'autres solutions pour satisfaire au critère du temps réel (effectuer certains calculs sur le GPU notamment)

Plan

- Introduction au domaine d'application
  - Subdivision de surfaces
  - GPGPU
  - Extensions récentes d'OpenGL
- Présentation du projet
  - Cahier des charges
  - Architecture du logiciel
  - Présentation de l'application
- Subdivision temps réel sur CPU
  - Méthode utilisée
  - Résultats obtenus
- **Subdivision temps réel sur GPU**
  - **Méthode utilisée**
  - **Résultats obtenus**

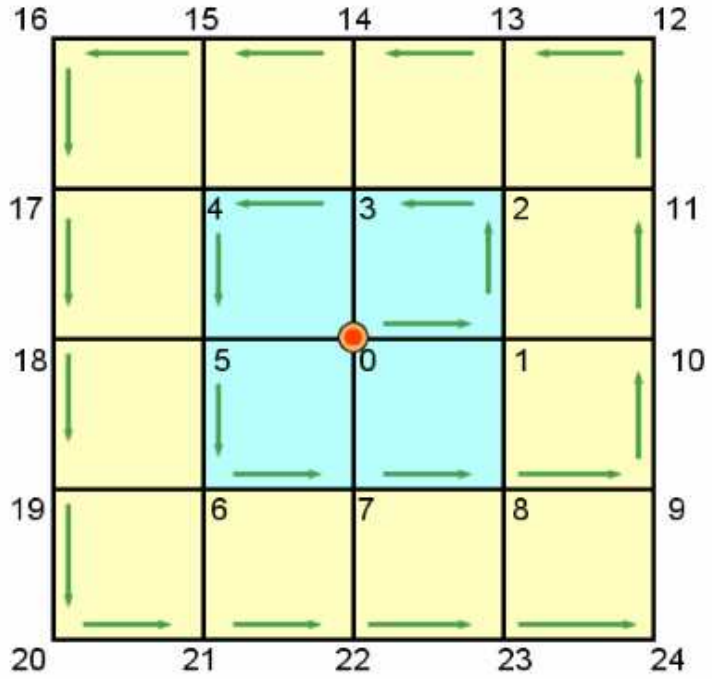
Subdivision temps réel sur GPU
Méthode utilisée (1/5)

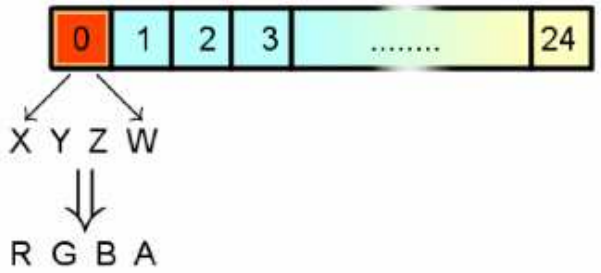
- Précalculs effectués sur le CPU
  - Découpage en *fragment meshes*



Subdivision temps réel sur GPU
Méthode utilisée (2/5)

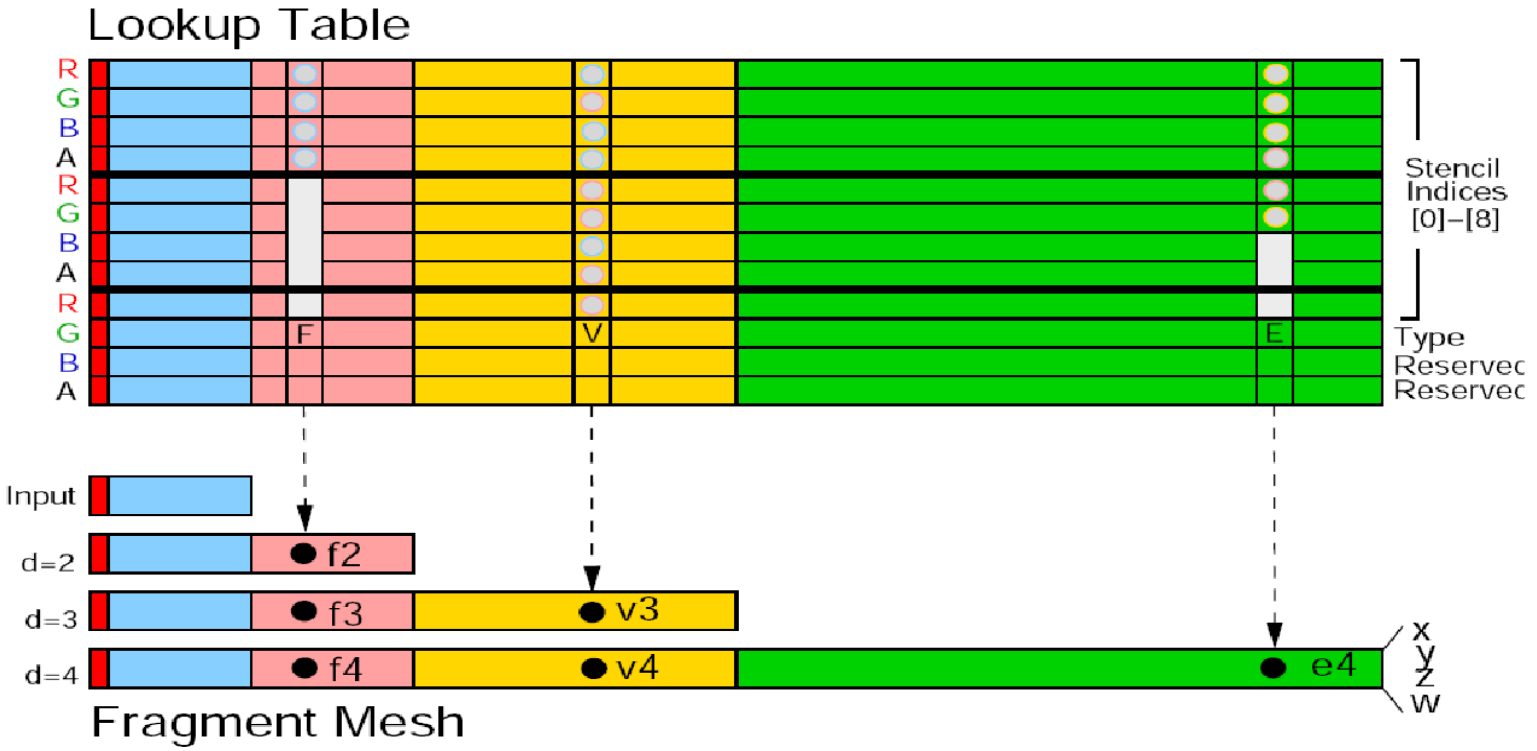
- Précalculs effectués sur le CPU
  - Création d'une *patch-texture* associée à chaque *fragment mesh*





Subdivision temps réel sur GPU
Méthode utilisée (3/5)

- Précalculs effectués sur le CPU
  - Création d'une *lookup table* par valence



Subdivision temps réel sur GPU
Méthode utilisée (4/5)

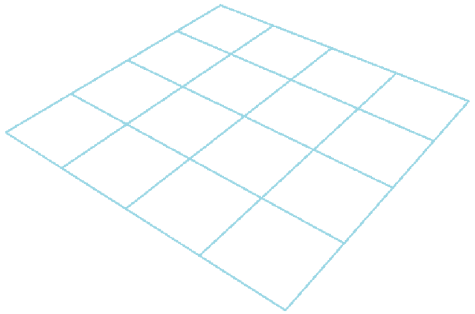
- Tâches effectuées sur le GPU
  - Initialisation du rendu à la taille de la *patch-texture* du niveau suivant (n+1) avec un quad lui aussi à la taille du rendu
  - Calcul de chaque pixel à l'aide d'un *fragment program* prenant en paramètre la *patch-texture* de niveau n et la *lookup table*

Subdivision temps réel sur GPU
Méthode utilisée (5/5)

- Tâches effectuées sur le GPU
  - Réalisation du nombre d'itérations souhaitées (nombre de subdivisions)
  - Récupération de la *patch-texture* finale et conversion de RGBA en XYZW du *framebuffer* dans le VBO
  - Affichage des quads obtenus

Subdivision temps réel sur GPU
Résultats obtenus

- Test effectué sur un unique *fragment mesh* de valence 4 avec un niveau de subdivision



	Nombre de Polygones		FPS
	Niveau 0	Niveau 1	
CPU	9	36	743,5
GPU	16	36	1410,2

Bilan (1/2)

- Travail réalisé :
  - application CPU opérationnelle
  - application GPU en partie réalisée
    - découpage en *fragment meshes*
    - étude du rendu multi-passes à l'aide des *shaders* et du *render to vertex array*
- Ce qu'il reste à faire :
  - génération automatique de la *lookup table*
  - *extension à l'ensemble d'un objet*

Bilan (2/2)

- Extensions possibles :
  - plusieurs schémas de subdivisions
  - combinaison avec des techniques de *displacement mapping* et *normal mapping*
  - interface plus élaborée

Questions ?