Projet COMELEC: Lancer du rayon

Jefferson FERREIRA

Halina MENDOZA

Le lancer de rayon dans l'actualité

Avantages

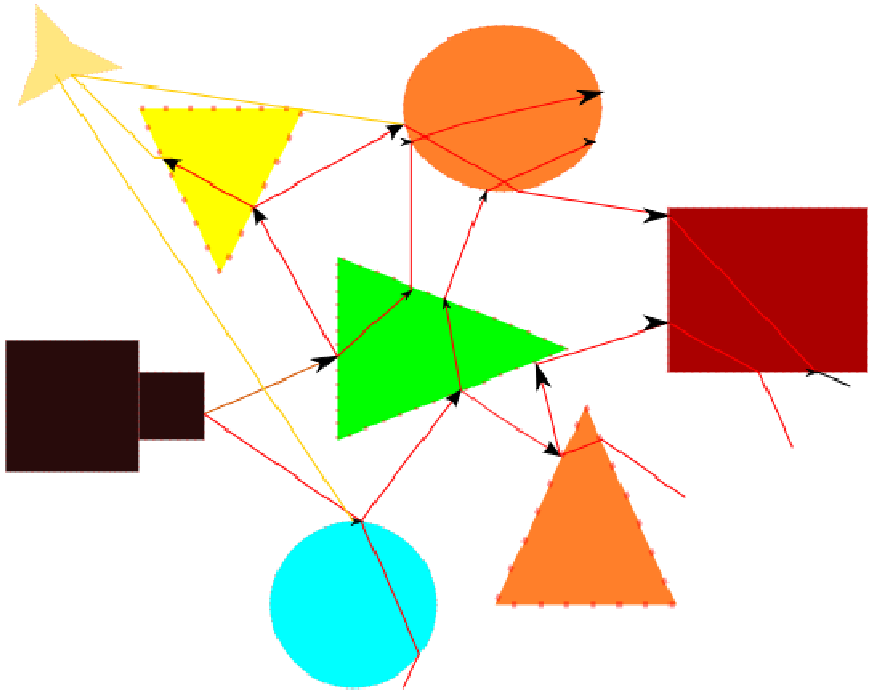
- Meilleure réalisme des images
- Facilité de simuler des effets des lumières et ombres.
- Usage: Filmes 3D

Désavantages

- Pas du matériel performant dédie en existence
- Non diffusion inter-réflexive entre surface
- Usage: Majorité des cartes graphiques

Principes physiques du lancer de rayon

Optique de la lumière	• Réflexion • Réfraction
Effets de l'illumination	• Illumination • Ombres
Primitives	• Sphère (test) • Triangle (base)



Parcours des rayons réfléchis et réfractes dans des objets divers

Structures de données

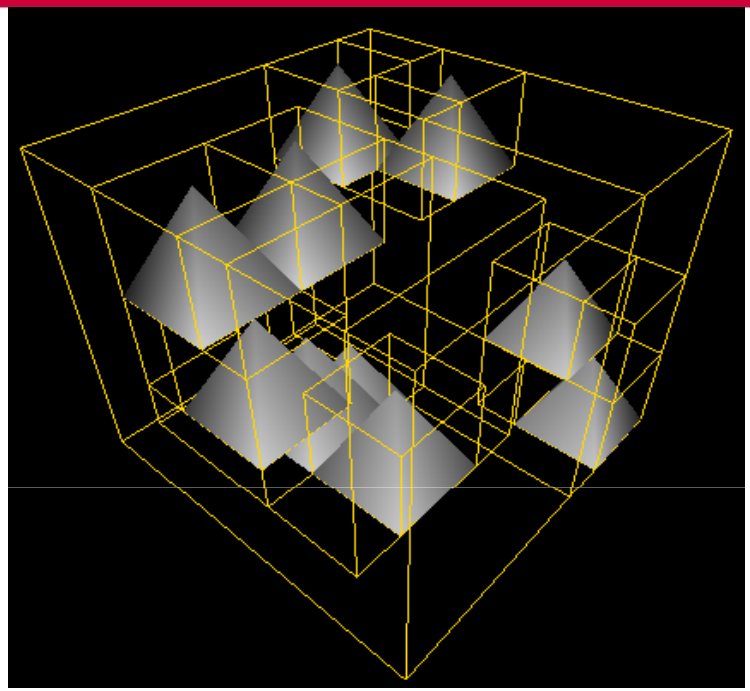
- **Problème:**

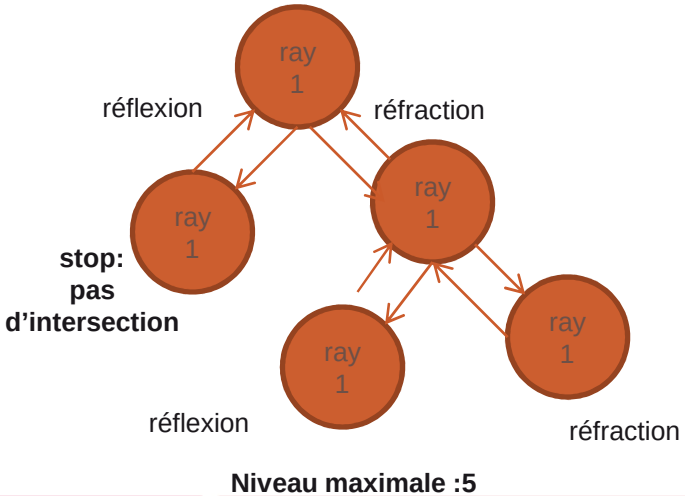
60K primitives – vérifier intersection, pour **chaque** rayon, avec **tous** les **primitives** Trop **lourd**!

- **Solution:**

Indexation Spatiale – Algorithme: *Surface Area Heuristic KD Tree.*

Composé par des **arbres binaires** qu'**indexent** les **primitives** par **régions**.





- Quand il y a un **cas d'intersection**, le rayon **incident génère** un rayon **réfléchi** et un rayon **réfracté**

- Les **arbres binaires** sont utilisés aussi pour traiter ces **rayons secondaires**

TELECOM ParisTech

Architecture proposé: Nombre d'opérations

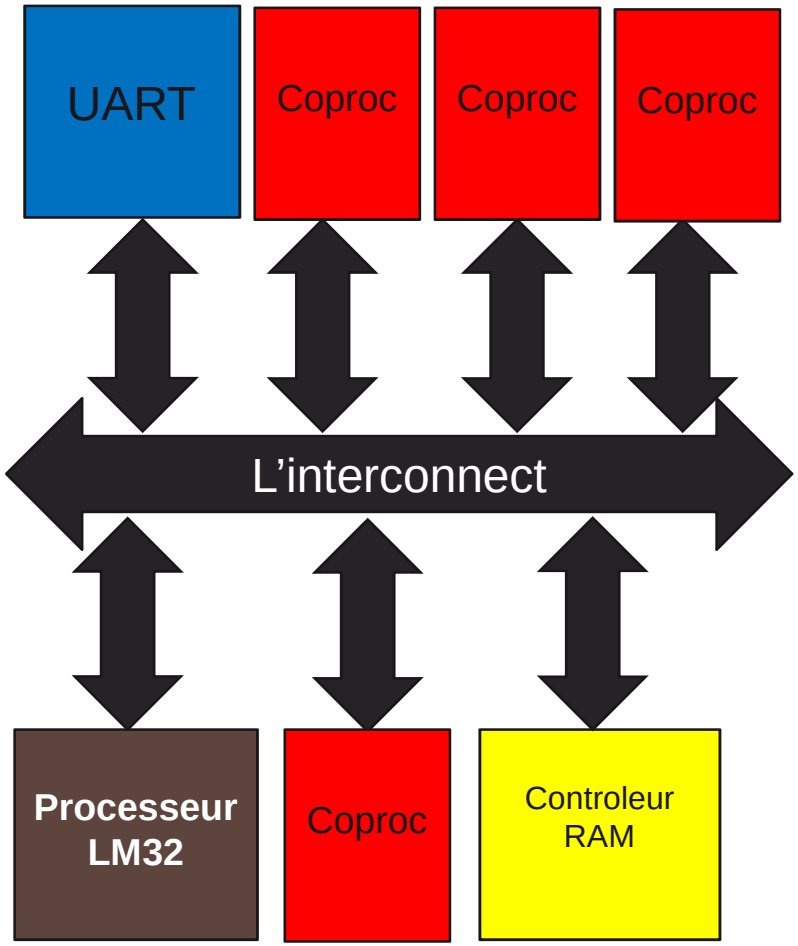
- Pour décider l'architecture il faut avoir une dimension du nombre d'opérations réalisées par le lancer de rayon

Operations	Accès au mémoire	Comparaisons	+	-	*	/
Prendre premier paquet des triangles	17	234	56	28	48	6
Charger Triangles (20 paquets – 40 Triangles)	800	0	0	0	0	0
Hit Triangles	0	7	9	15	15	3
Intersection ensemble de triangles (40 Triangles/paquet)	0	361	400	600	1080	120
Vérifier réflexion	0	363	400	600	1080	120
Vérifier réfraction	0	363	400	600	1080	120
Prendre prochain paquet de triangles	17	203	50	18	42	6

- **Architecture :**

1) Processeur – responsable de localiser et donner le premier paquet pour le coprocesseur. Il gère la exécution du coprocesseur.
2) Coprocesseur – responsable du calcul d'intensité lumineuse d'un pont d'image

Architecture proposé: Vision globale



Parallélisme: Chaque **coprocesseur** traite l'**intensité** lumineuse d'un point **différent** de l'image

Pour cela, il faut que le **processeur**:

1) **Calcule** l'intersection du rayon venant du camera avec l'espace indexé
2) **Fournisse** pour chacun des coprocesseurs un rayon (utilisé pour calculer l'intersection) et l'**adresse** du **premier paquet** de primitives.

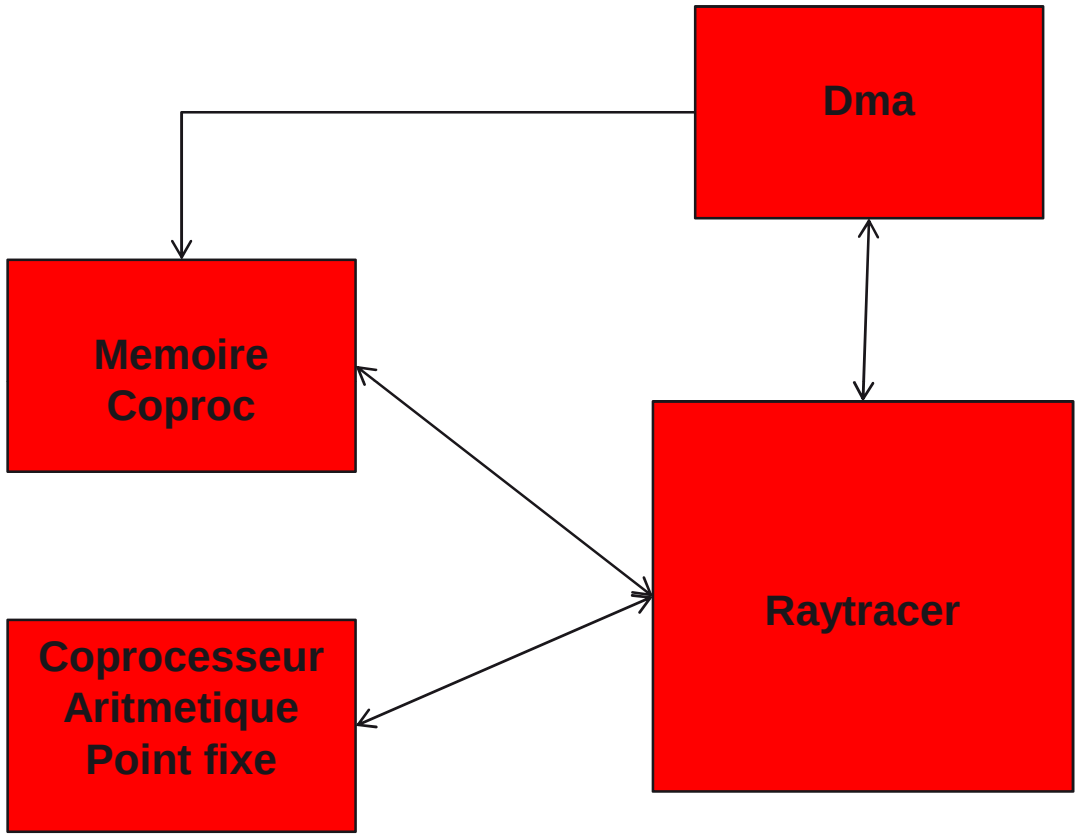
Pour quoi le parallélisme avec **4** coprocesseurs?

Total **d'opérations élémentaires** réalisés (**1 clock**) par un coprocesseur: 203000

L'accès de **mémoire** est réalisée dans **2 clocks:** 49600

Alors le bus peut loger **sans perte abrupte de performance 4** coprocesseurs

Architecture Coprocesseur



Capacité de la mémoire du coprocesseur:

- 2 Nœuds kdTree – 110

N1 – avec les données de la racine

N2 – avec les données du paquet de triangles courant (une feuille de la kdTree)

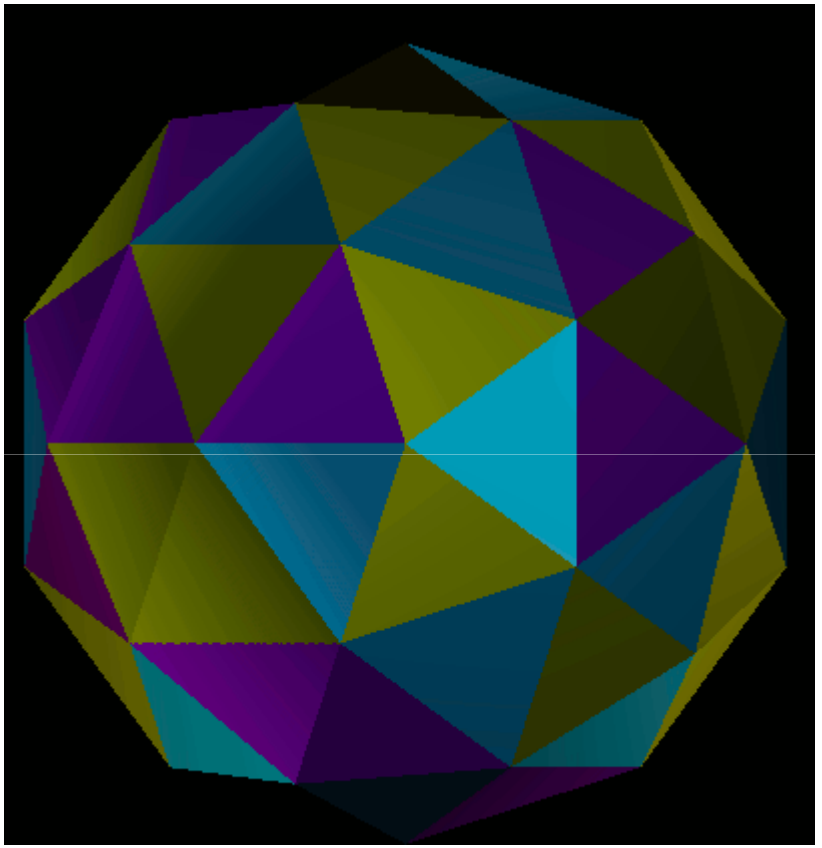
- l'adresse du paquet courant – 4
- l'adresse pour stocker l'intensité lumineuse – 4
- Un vecteur pour jusqu'à 512 triangles – 15872
- Une structure pour le triangle plus proche – 31
- 1 – arbre binaire réflexion-réfraction – 1736
- 1 – vecteur pour les matérielles – 384
- 1 – vecteur pour les lumières - 384

Total: = 18.09 Kbytes

Conclusions

- Projet intéressant car, il est possible d'utiliser les techniques de projet de SoC dans une area inconnue par les intégrants du groupe (génération d'images en 3D)

- Il y a des bugs encore: le lancer de rayon arrive a fournir des images mais il faut vérifier quelques pendances.



Exemple d'image générée