

Efficient Voxelization

Using Projected Optimal Scanline

François Palma
Nicolas Marguerit
Alexandre Réaubourg

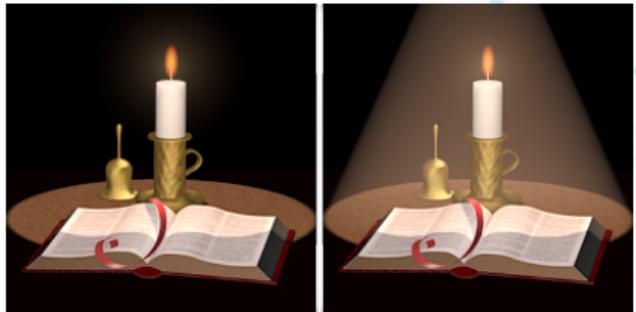
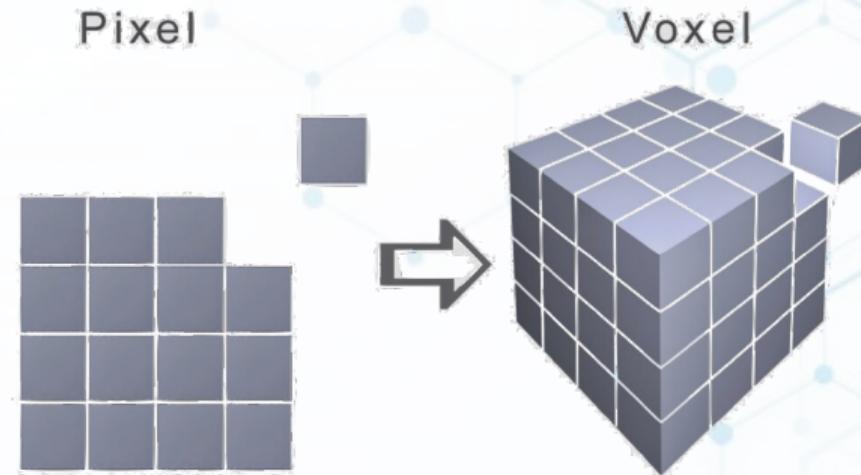
Sommaire

1. État de l'art
2. Etude de l'article
3. Prototypage
4. Résultats & Benchmark

Introduction

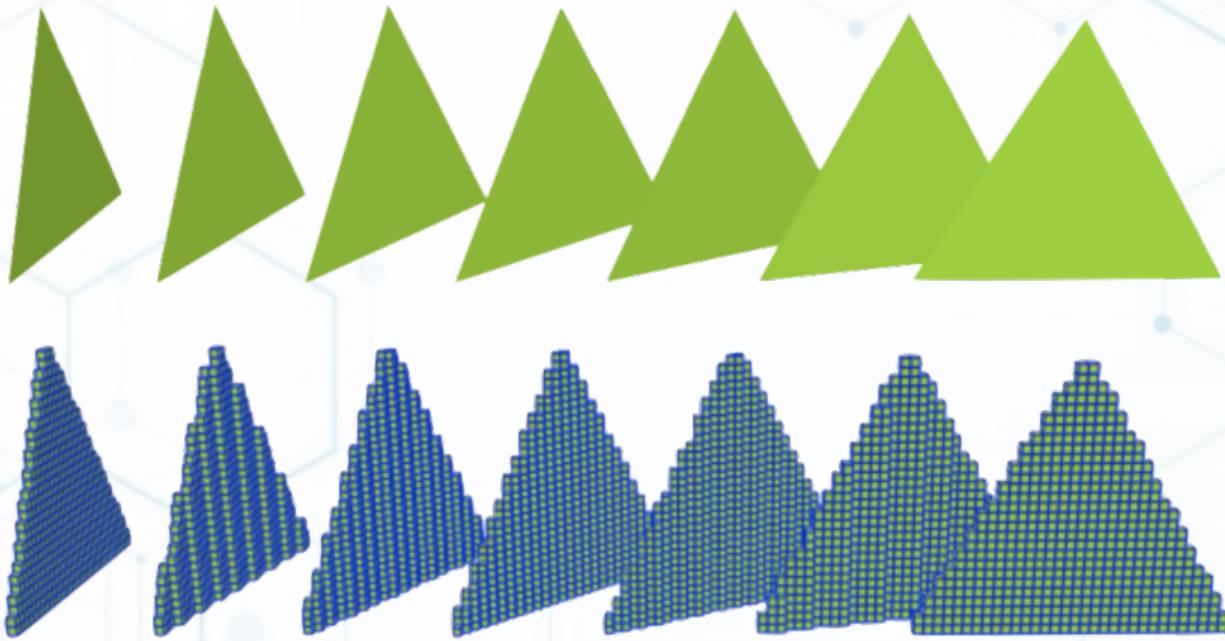
Les utilisations:

- ▶ Modélisation
- ▶ Simulation physique
 - ▶ De nouvelles simulations rendues possibles
- ▶ Eclairage volumétrique



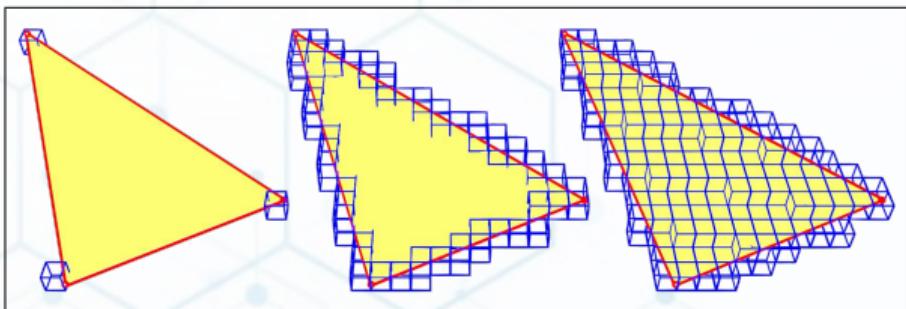
État de l'art

SS10 & Pan11



Etude de l'article

Spécificité de l'algorithme



Etapes de la voxelization

Triangle Voxelization

Just a regular block

MarkLineLV

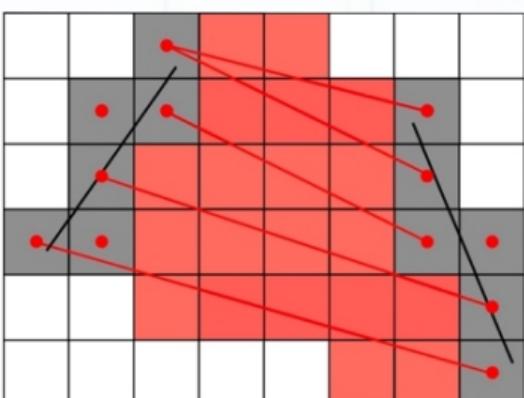
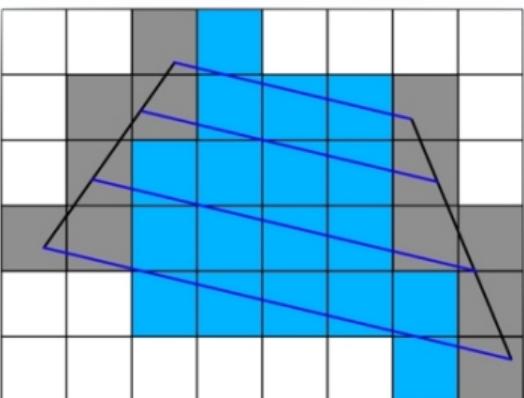
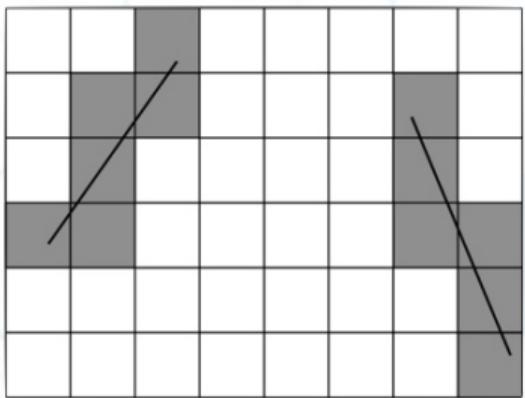
Just a regular block

FillInterior

Just a regular block

Etude de l'article

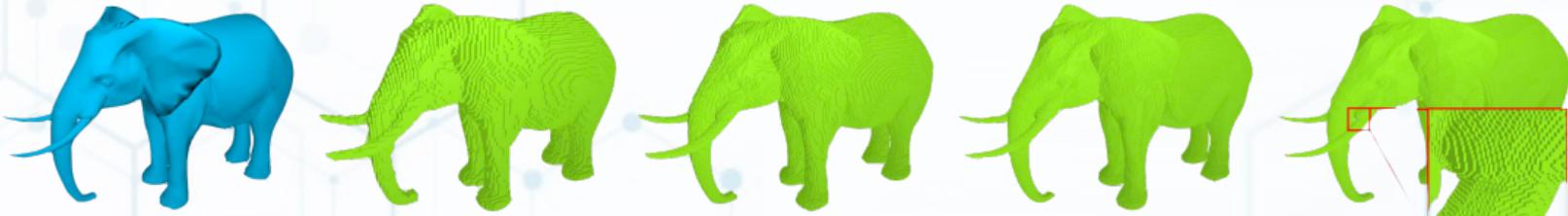
3D voxelization ? scanlines ?



Etude de l'article

Points clés

- ▶ Optimisation en temps
- ▶ Parfaitement compatible avec le multithreading
- ▶ Plus on a de triangles plus c'est relativement rapide
- ▶ la discréétisation par des entiers réduit les trous dans la couverture



Prototypage

Plan initial

- ▶ Voxelization d'un triangle
- ▶ Utiliser OpenGL
- ▶ Prototypage 2D

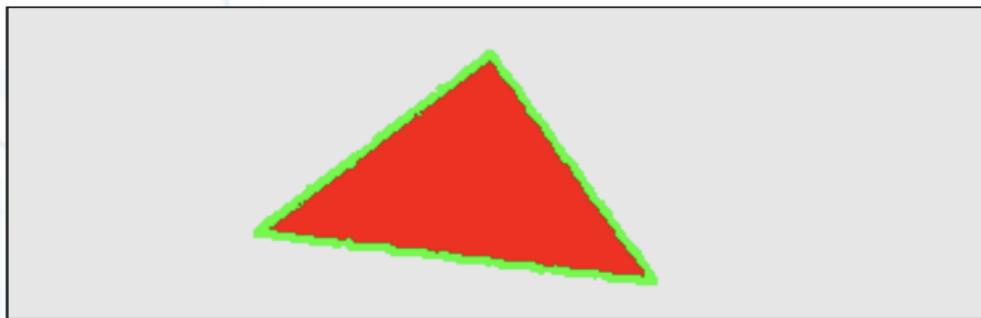
Tout semblait simple et clair



Prototypage

Premiers résultats

- ▶ Reprise en main d'OpenGL
- ▶ Rendu du triangle à l'écran très satisfaisante
- ▶ Voxelization des côtés se fait rapidement

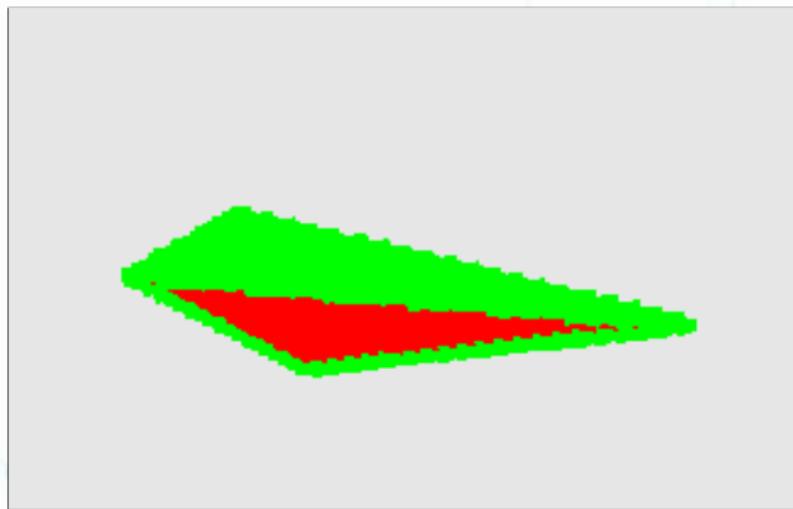


Voxelization des côtés

Prototypage

Difficultés rencontrées

- ▶ Apparition du problème de la pyramide
- ▶ La méthode naïve marche étonnamment mieux avec une couverture parfaite



Problème de la pyramide

Prototypage

Éclaircissement sur les zones d'ombres

- ▶ Manque de détail dans les cas 2D
- ▶ Choix d'utilisation de Bresenham dans les cas 2D
- ▶ Des parties de l'algorithme ne sont pas détaillées(mettre image avec encadrés rouges)

```
function FILLINTERIOR( $Q_1, Q_2, P_0, P_2, axis$ )
    for  $i = 0$  to  $P_2[axis] - P_0[axis]$  do
        slice  $\leftarrow P_0[axis] + i + 1/2$ 
         $Q_{1sub} \leftarrow \text{GETSUBSEQUENCE}(Q_1, slice)$ 
         $Q_{2sub} \leftarrow \text{GETSUBSEQUENCE}(Q_2, slice)$ 
        while  $Q_{1sub} \neq \emptyset$  OR  $Q_{2sub} \neq \emptyset$  do
             $P_{start} \leftarrow \text{GETNEXTINSLICE}(Q_{1sub})$ 
             $P_{stop} \leftarrow \text{GETNEXTINSLICE}(Q_{2sub})$ 
            MARKLINEILV( $P_{start}, P_{stop}$ )
```

Résultats & Benchmark

Quelques chiffres en comparaison

	Velocity	Angle	Vertical force
	U [m/s]	α [°]	F_z [N]
2D simulation	9	2	9.23
3D simulation	10.0	3	15.039
Experiment A	11.31	2.5	13.2
Experiment B	11.26	2.7	12.6
Experiment C	11.33	2.47	13.6

Et ensuite place à la démo !

Résultats & Benchmark

Perspectives d'évolution

- ▶ Utiliser CUDA
- ▶ Changer de langage
- ▶ Passer sur du multithread



Résultats & Benchmark

Conclusion

Implémentation de l'algo de voxelization de l'article en OpenGL
réussi mais avec notre interprétation pour le problème des cas 2D