

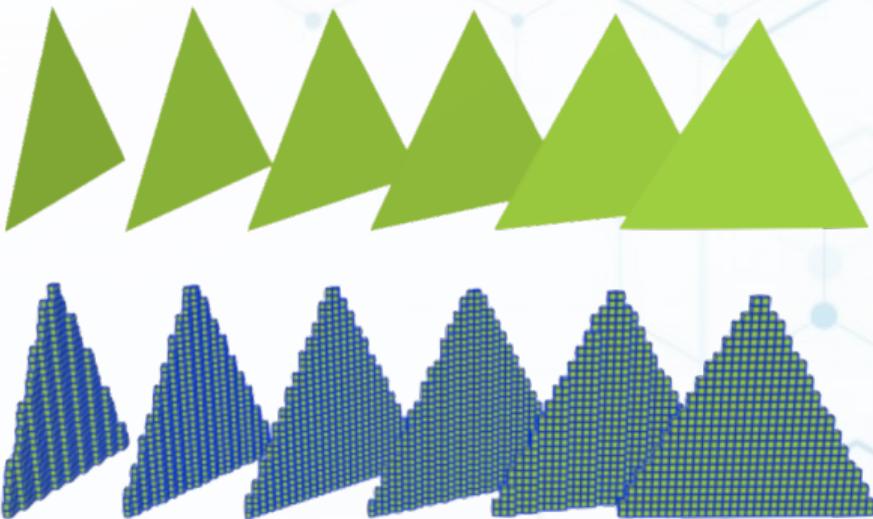
## Efficient Voxelization

Using Projected Optimal Scanline

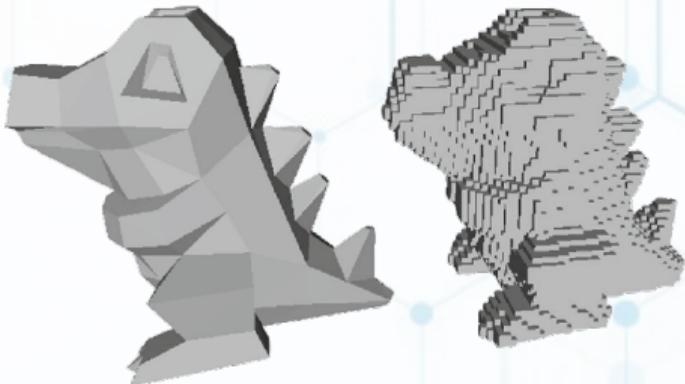
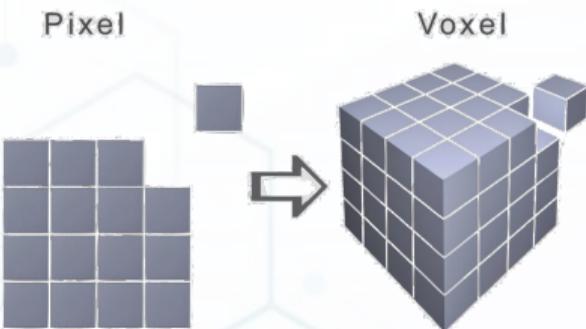
François Palma  
Nicolas Marguerit  
Alexandre Réaubourg

# Sommaire

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

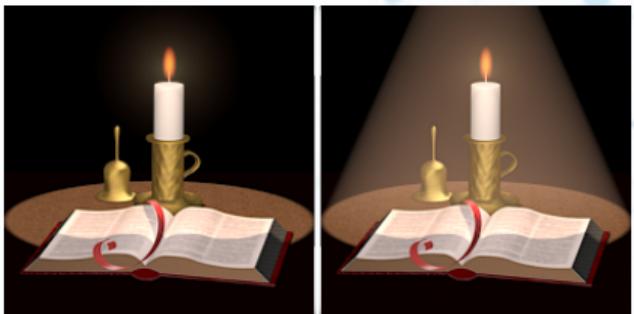


# Introduction

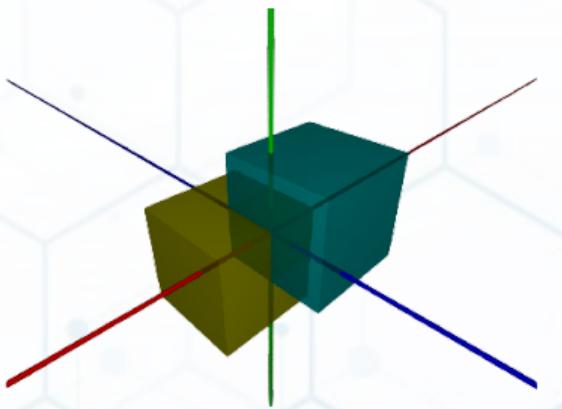


## Les utilisations:

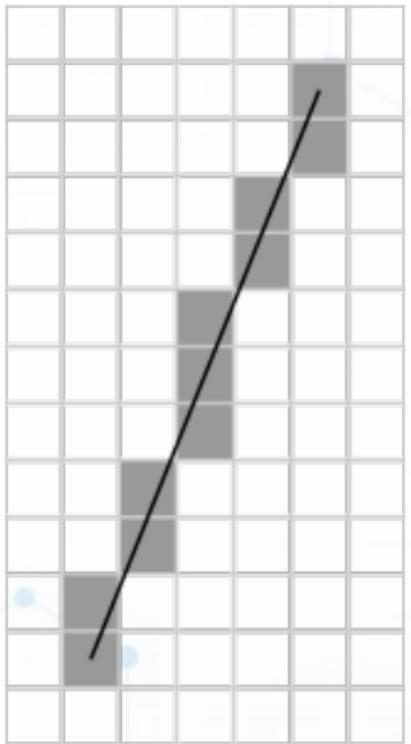
- ▶ Modélisation
- ▶ Simulation physique
  - ▶ De nouvelles simulations rendues possibles
- ▶ Eclairage volumétrique



# État de l'art



SAT



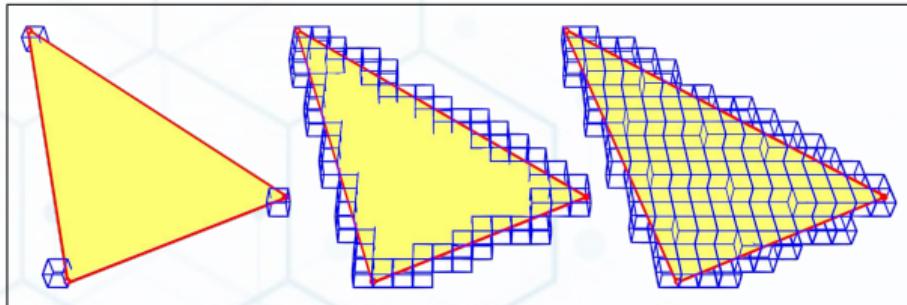
Bresenham



Couverture & Tunnel

# Etude de l'article

## Spécificité de l'algorithme



Etapes de la voxelization

**MarkLineLV**

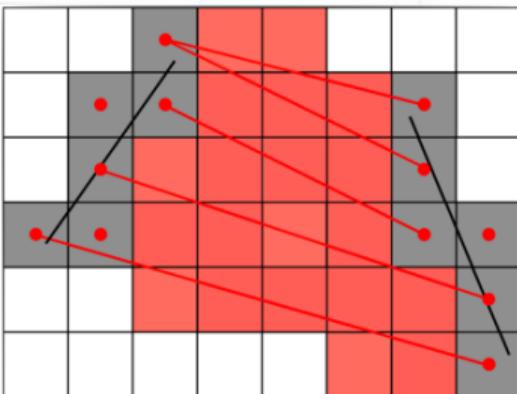
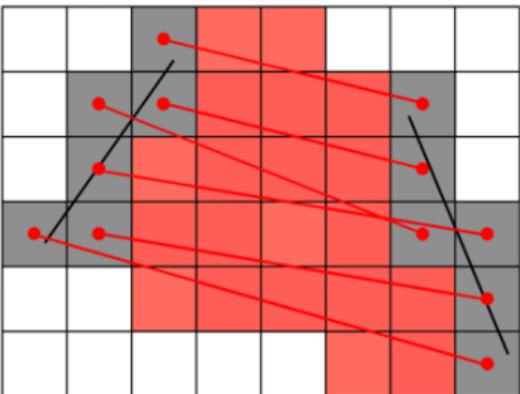
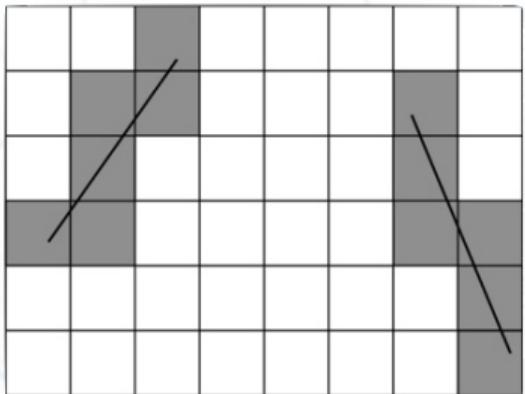
Voxelization des côtés

**FillInterior**

Voxelization de l'intérieur

# Etude de l'article

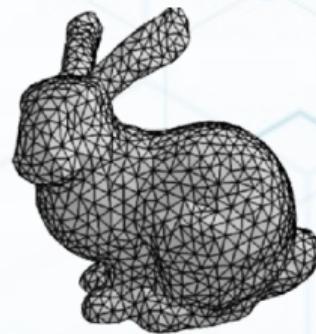
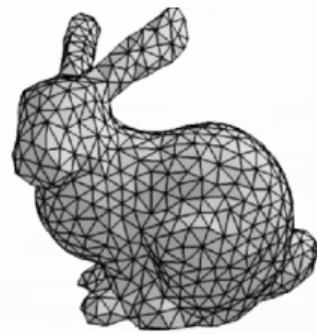
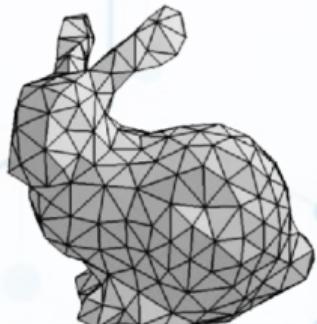
## 3D voxelization ? scanlines ?



# 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 des entiers permet l'optimisation des temps de calcul

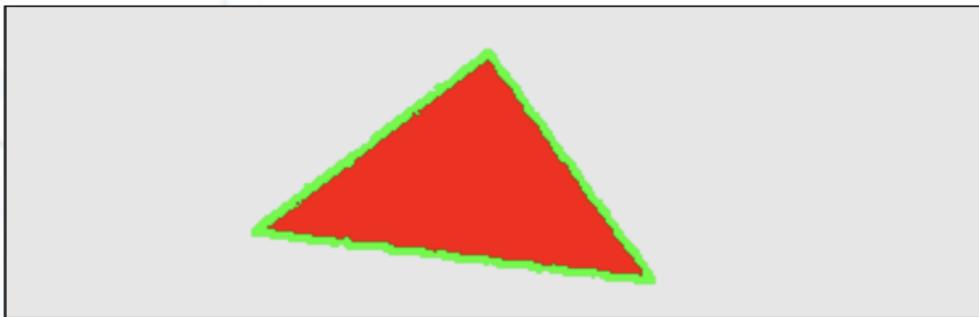


# Prototypage

## Début



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

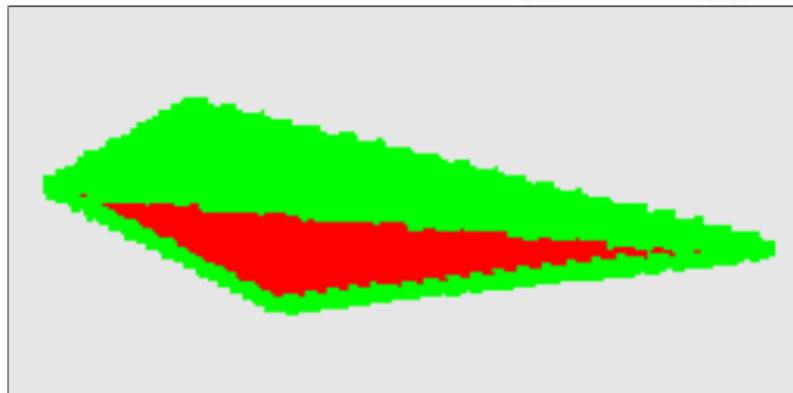


Voxelization des côtés

# 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

```
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

	100 triangles	1000 triangles	10000 triangles	couverture en %
Scanline Optimales	2.316	20.114	209.718	92.11
Scanline Naïves	4.35	45.584	444.665	100

*Et ensuite place à la démo !*

# Résultats & Benchmark

## Perspectives d'évolution

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



# Conclusion

- ▶ Implementation réussie
  - ▶ Mais avec notre interprétation pour les cas 2D
- ▶ Présentation en LaTeX