

# Efficient Voxelization

Using Projected Optimal Scanline

François Palma  
Nicolas Marguerit  
Alexandre Réaubourg

January 4, 2022

# Sommaire

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

# État de l'art

# État de l'art

## Voxelization

### Exemple d'utilisation:

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

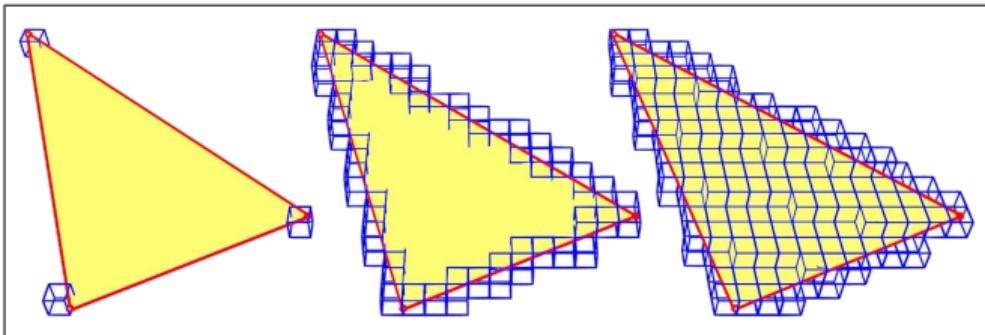
# Etude de l'article

# Etude de l'article

## 3D voxelization ? scanlines ?

# Etude de l'article

## Spécificité de l'algorithme



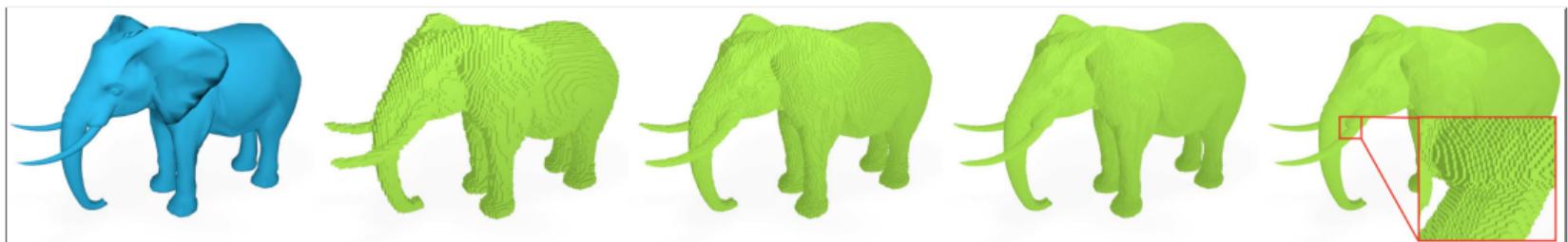
Etapes de la voxelization

```
Algorithm 1 Triangle Voxelization
1: function VOXELIZETRIANGLE( $v_0, v_1, v_2, n$ )
2:    $\{P_0, P_1, P_2\} \leftarrow \text{GETVOXEL}(v_0, v_1, v_2)$ 
3:    $i \leftarrow \text{DOMINANTAXISINDEX}(P_0, P_1, P_2)$ 
4:   SORTONAXIS( $P_0, P_1, P_2, i$ )
5:   MARKLINEILV( $P_0, P_1, Q_0$ )
6:   MARKLINEILV( $P_1, P_2, Q_1$ )
7:   MARKLINEILV( $P_0, P_2, Q_2$ )
8:    $Q_1 \leftarrow Q_0 \cup Q_1$ 
9:   FILLINTERIOR( $Q_1, Q_2, P_0, P_2, i$ )
10:
11: function MARKLINEILV( $P_0, P_1, Q$ )
12:    $\Delta P[0] \leftarrow \text{SIGN}(P_1[0] - P_0[0])$ 
13:    $\Delta P[1] \leftarrow \text{SIGN}(P_1[1] - P_0[1])$ 
14:    $\Delta P[2] \leftarrow \text{SIGN}(P_1[2] - P_0[2])$ 
15:    $L[0] \leftarrow M[0] \leftarrow |P_1[1] - P_0[1]| |P_1[2] - P_0[2]|$ 
16:    $L[1] \leftarrow M[1] \leftarrow |P_1[0] - P_0[0]| |P_1[2] - P_0[2]|$ 
17:    $L[2] \leftarrow M[2] \leftarrow |P_1[0] - P_0[0]| |P_1[1] - P_0[1]|$ 
18:    $P_{\text{current}} \leftarrow P_0$ 
19:   while  $P_{\text{current}} \neq P_1$  do
20:      $(L_{\min}, L_{\max}) \leftarrow \text{MIN}(L[0], L[1], L[2])$ 
21:      $P_{\text{current}}[L_{\text{index}}] \leftarrow P_{\text{current}}[L_{\text{index}}] + \Delta P[L_{\text{index}}]$ 
22:      $L \leftarrow L_{\min}$ 
23:      $L[L_{\text{index}}] \leftarrow 2M[L_{\text{index}}]$ 
24:     MARKVOXEL( $P_{\text{current}}$ )
25:      $Q_{\text{pushback}}(P_{\text{current}})$ 
26:
27: function FILLINTERIOR( $Q_1, Q_2, P_0, P_2, axis$ )
28:   for  $i = 0$  to  $P_2[axis] - P_0[axis]$  do
29:     slice  $\leftarrow P_0[axis] + i + 1/2$ 
30:      $Q_{1,\text{sub}} \leftarrow \text{GetSubSequence}(Q_1, slice)$ 
31:      $Q_{2,\text{sub}} \leftarrow \text{GetSubSequence}(Q_2, slice)$ 
32:     while  $Q_{1,\text{sub}} \neq \emptyset \text{ or } Q_{2,\text{sub}} \neq \emptyset$  do
33:        $P_{\text{start}} \leftarrow \text{GetNextInSlice}(Q_{1,\text{sub}})$ 
34:        $P_{\text{stop}} \leftarrow \text{GetNextInSlice}(Q_{2,\text{sub}})$ 
35:       MARKLINEILV( $P_{\text{start}}, P_{\text{stop}}$ )
```

# Etude de l'article

## Points clés

- ▶ Plus c'est rapide mieux c'est
- ▶ Parfaitement compatible avec le multithreading
- ▶ Plus on a de triangles plus c'est rapide
- ▶ L'approximation des entiers réduit les trous dans la couverture



# Prototypage

# Prototypage

## Notre plan initial

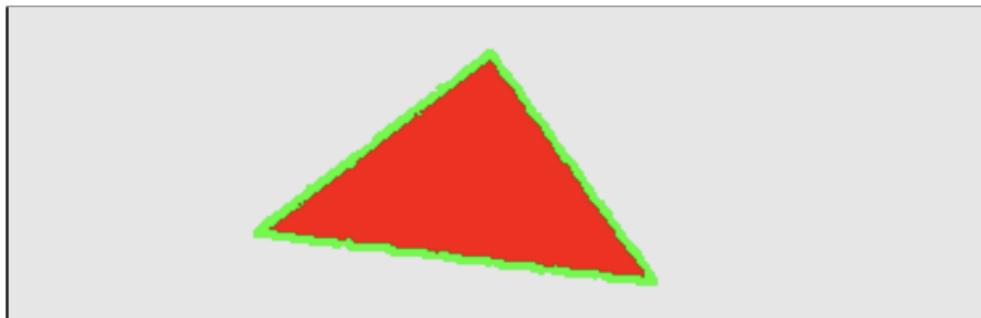
- ▶ Voxelization d'un triangle
  - ▶ Utiliser openGL
- 
- ▶ Nous pensions que tout était simple et clair



# Prototypage

## Premiers jours

- ▶ Reprise en main d'OpenGL
- ▶ Très satisfait du rendu du triangle à l'écran
- ▶ Voxelization des côtés se fait rapidement
- ▶ Parties qui ne sont pas claires dans l'algo

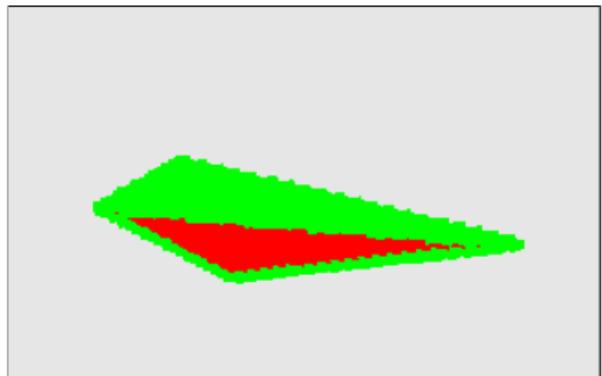


Voxelization des côtés

# Prototypage

## Difficultés rencontrées

- ▶ Essais et erreurs pour trouver ce qui ne fonctionne pas bien
- ▶ Essayer la méthode naïve et se rendre compte qu'elle marche mieux
- ▶ Réaliser que l'algo est incomplet



# Prototypage

Éclaircissement des zones d'ombres sur l'article

- ▶ Relecture minutieuse de l'article pour trouver où se trouve le problème
- ▶ Manque de détail dans les cas 2D

# Résultats & Benchmark

# Résultats & Benchmark

Quelques chiffres en comparaison

# Résultats & Benchmark

Démo

# Résultats & Benchmark

## Perspectives d'évolution

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

# Résultats & Benchmark

## Conclusion