Construction de BVH sur le GPU pour le calcul de visibilité

Nicolas Said – Automne 2009

utbm
université de technologie
Belfort-Montbéliard

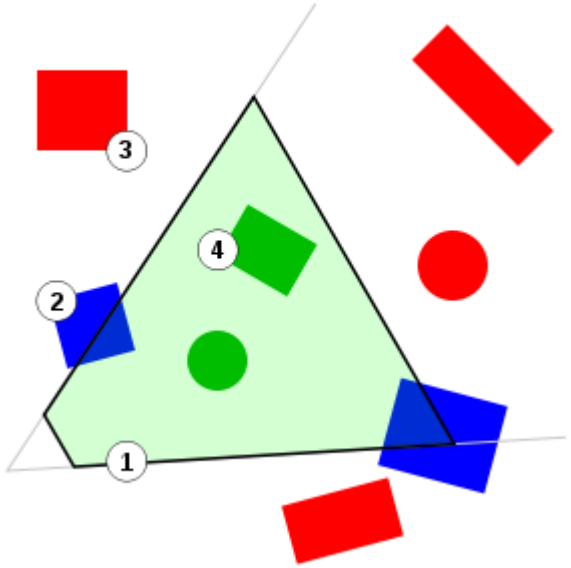
NVIDIA
CUDA

Plan de la présentation

- Rappel de la problématique
- Rappel concernant le travail précédent
- Notions de programmation GPGPU avec CUDA
- Présentation de l'algorithme de construction
- Présentation de l'implémentation
- Algorithme de Frustum Culling
- Résultats
- Démonstration

Rappel de la problématique

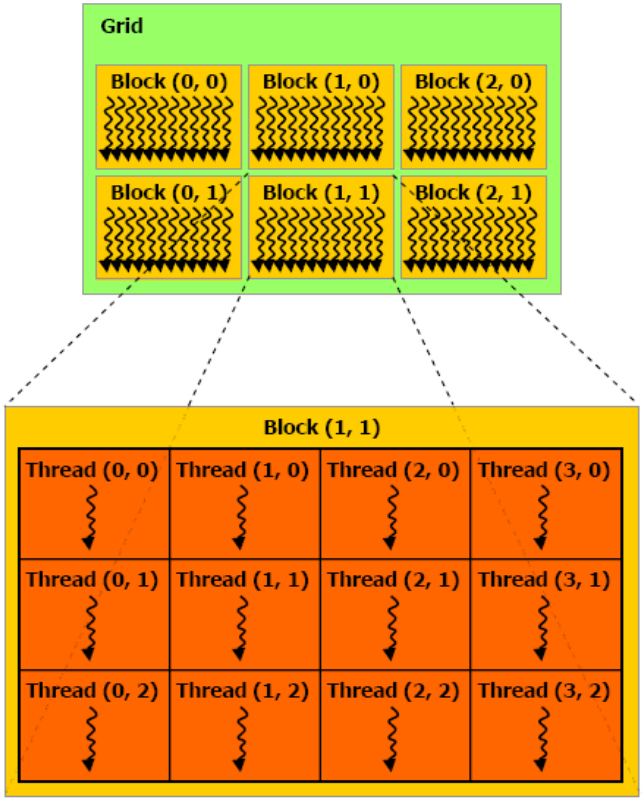
- Calcul de visibilité pour la simulation de piétons (entre autres)



Travail précédent

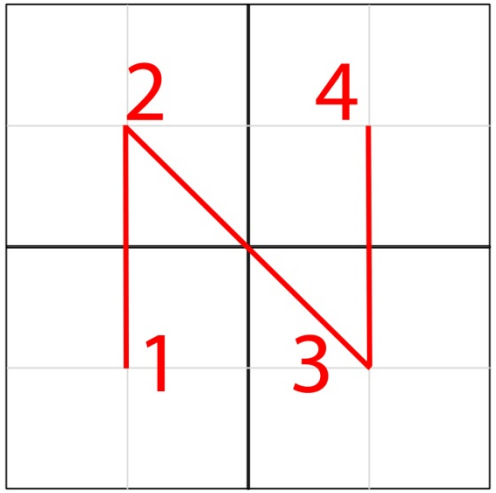
- Approche naïve (Bruteforce)

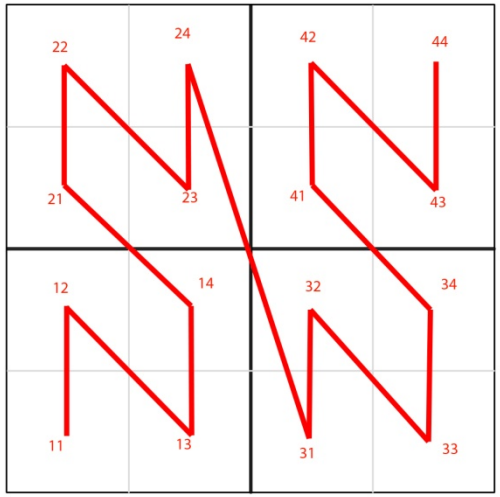
Technologie nVidia CUDA



Algorithme de construction : LBVH

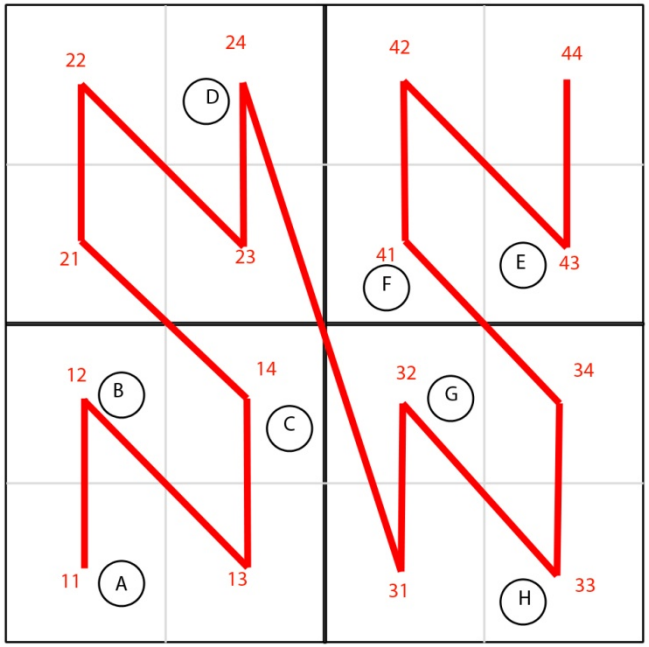
- Comparaison avec un algorithme de construction classique
- Inconvénients





LBVH : Implémentation

- Prenons un exemple



Primitive	Code niveau 1	Code niveau 2
A	1	1
B	1	2
C	1	4
D	2	4
E	4	3
F	4	1
G	3	2
H	3	3

LBVH : Implémentation

- Tri sur les codes (tri par base)

Primitive	Code niveau 1	Code niveau 2
A	1	1
B	1	2
C	1	4
D	2	4
E	4	3
F	4	1
G	3	2
H	3	3

Primitive	Code
A	11
B	12
C	14
D	24
G	32
H	33
F	41
E	43

LBVH : Implémentation

- Construction de la liste de « split »

Primitive	Code
A	11
B	12
C	14
D	24
G	32
H	33
F	41
E	43

Emplacement du split	Niveau du split
0	1
0	2
1	2
2	2
3	1
3	2
4	1
4	2
5	2
6	1
6	2
7	2
8	1
8	2

LBVH : Implémentation

- Tri de la liste de « split » par niveau de split

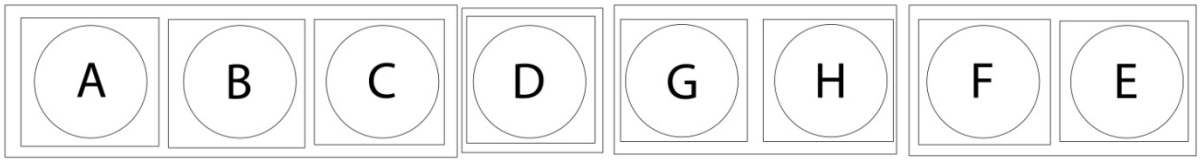
Emplacement du split	Niveau du split
0	1
0	2
1	2
2	2
3	1
3	2
4	1
4	2
5	2
6	1
6	2
7	2
8	1
8	2

Emplacement du split	Niveau du split
0	1
3	1
4	1
6	1
8	1
0	2
1	2
2	2
3	2
4	2
5	2
6	2
7	2
8	2

- Construction de la liste d'intervalles sur la liste de primitives

Emplacement du split	Niveau du split
0	1
3	1
4	1
6	1
8	1
0	2
1	2
2	2
3	2
4	2
5	2
6	2
7	2
8	2

Nœud	Niveau	Intervalle de primitives
0	1	[0 ;3[
1	1	[3 ;4[
2	1	[4 ;6[
3	1	[6 ;8[
4	2	[0 ;1[
5	2	[1 ;2[
6	2	[2 ;3[
7	2	[3 ;4[
8	2	[4 ;5[
9	2	[5 ;6[
10	2	[6 ;7[
11	2	[7 ;8[



LBVH : Implémentation

- Tri de la liste des nœuds par borne inf. des intervalles de primitive

Nœud	Niveau	Intervalle de primitives
0	1	[0 ;3[
1	1	[3 ;4[
2	1	[4 ;6[
3	1	[6 ;8[
4	2	[0 ;1[
5	2	[1 ;2[
6	2	[2 ;3[
7	2	[3 ;4[
8	2	[4 ;5[
9	2	[5 ;6[
10	2	[6 ;7[
11	2	[7 ;8[

Nœud	Niveau	Intervalle de primitives	Intervalle d'enfants
0	1	[0 ;3[	[4 ; ??]
4	2	[0 ;1[	
5	2	[1 ;2[	
6	2	[2 ;3[	
1	1	[3 ;4[	[7 ; ??]
7	2	[3 ;4[	
2	1	[4 ;6[	[8 ;??]
8	2	[4 ;5[	
9	2	[5 ;6[	
3	1	[6 ;8[	[10 ;??]
10	2	[6 ;7[	
11	2	[7 ;8[	

LBVH : Implémentation

- Tri de la liste des nœuds par borne sup. des intervalles de primitive

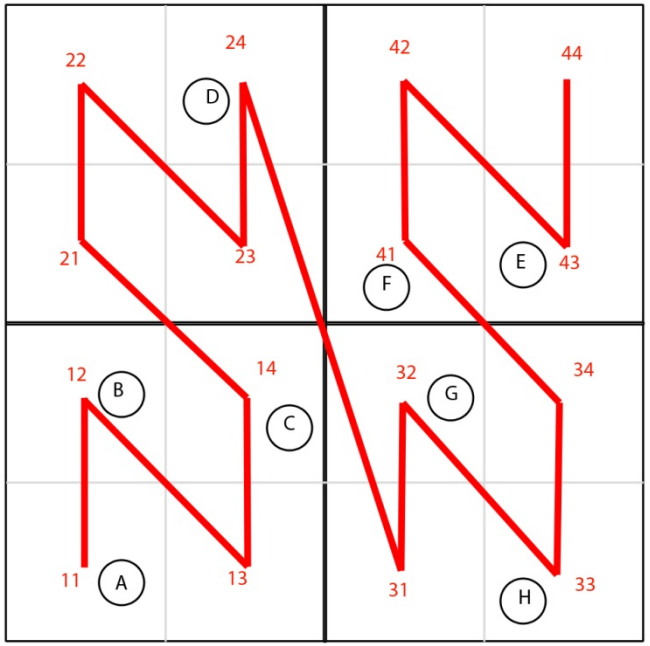
Nœud	Niveau	Intervalle de primitives
0	1	[0 ;3[
1	1	[3 ;4[
2	1	[4 ;6[
3	1	[6 ;8[
4	2	[0 ;1[
5	2	[1 ;2[
6	2	[2 ;3[
7	2	[3 ;4[
8	2	[4 ;5[
9	2	[5 ;6[
10	2	[6 ;7[
11	2	[7 ;8[

Nœud	Niveau	Intervalle de primitives	Intervalle d'enfants
4	2	[0 ;1[	
5	2	[1 ;2[	
0	1	[0 ;3[	[4 ; 6]
6	2	[2 ;3[	
1	1	[3 ;4[	[7 ; 7]
7	2	[3 ;4[	
8	2	[4 ;5[	
2	1	[4 ;6[	[8 ;9]
9	2	[5 ;6[	
10	2	[6 ;7[	
3	1	[6 ;8[	[10 ;11]
11	2	[7 ;8[	

LBVH : Implémentation

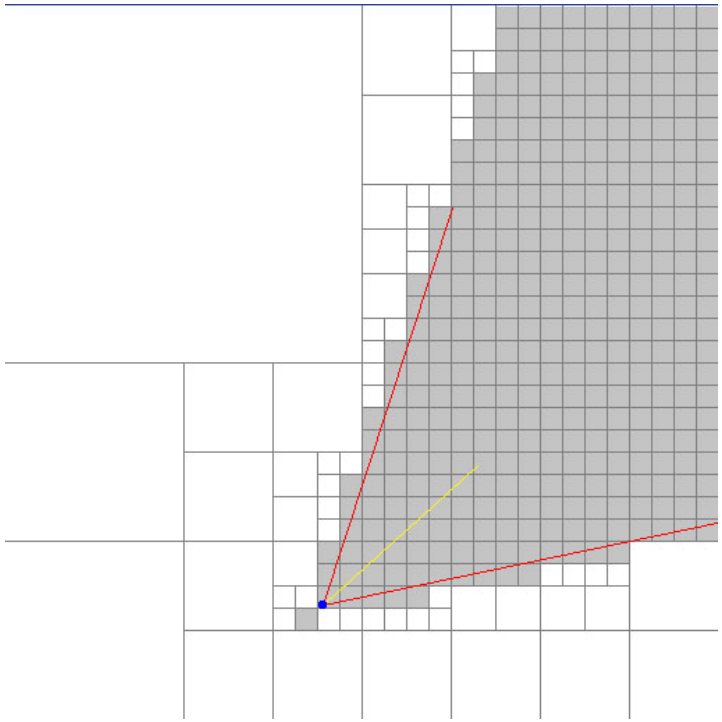
- Tri de la liste des nœuds par identifiant

Nœud	Niveau	Intervalle de primitives	Intervalle d'enfants
0	1	[0 ;3[	[4 ; 6]
1	1	[3 ;4[	[7 ; 7]
2	1	[4 ;6[	[8 ;9]
3	1	[6 ;8[	[10 ;11]
4	2	[0 ;1[	
5	2	[1 ;2[	
6	2	[2 ;3[	
7	2	[3 ;4[	
8	2	[4 ;5[	
9	2	[5 ;6[	
10	2	[6 ;7[	
11	2	[7 ;8[	



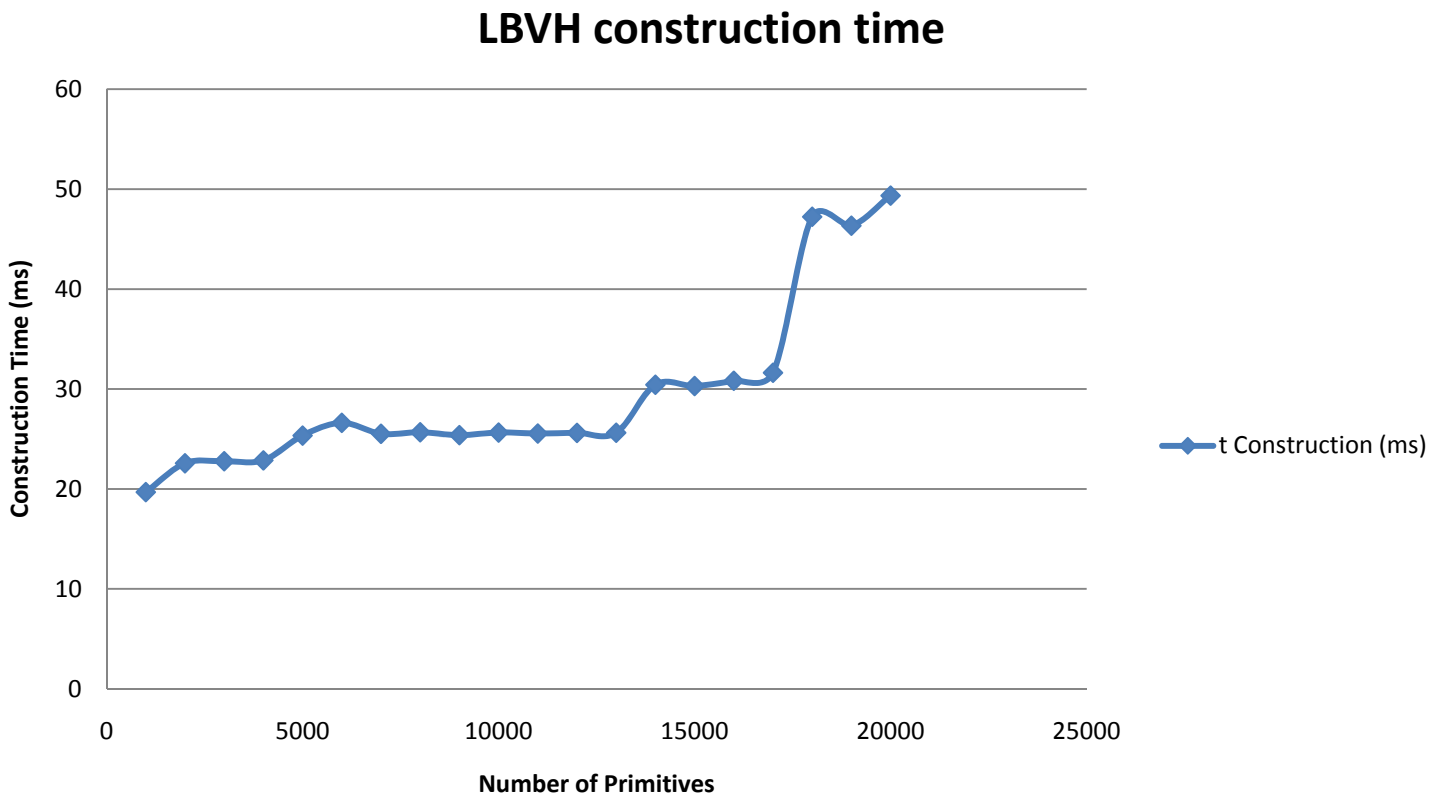
Frustum Culling

- Utilise la structure BVH (quadtree)



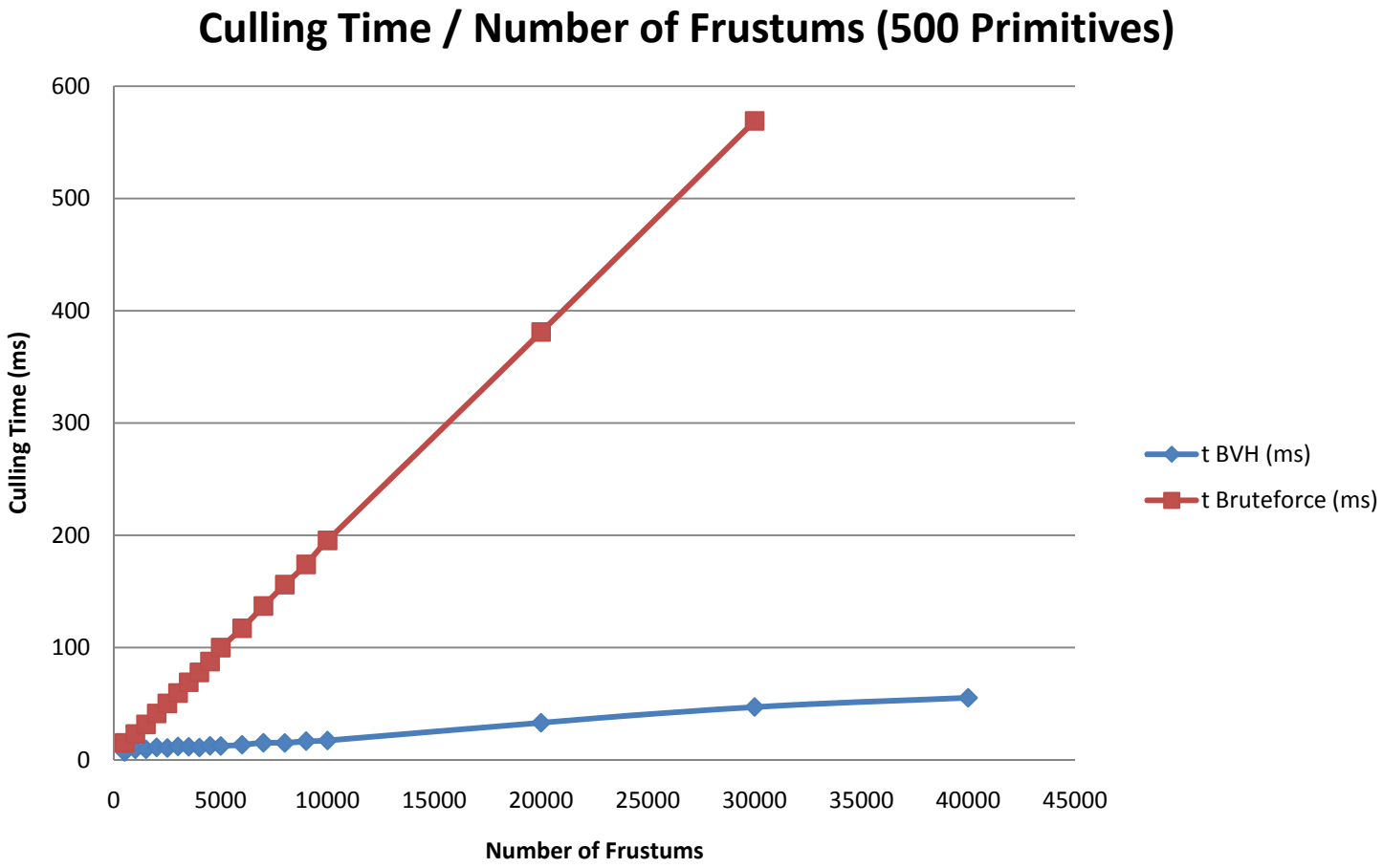
Résultats

- Temps de construction



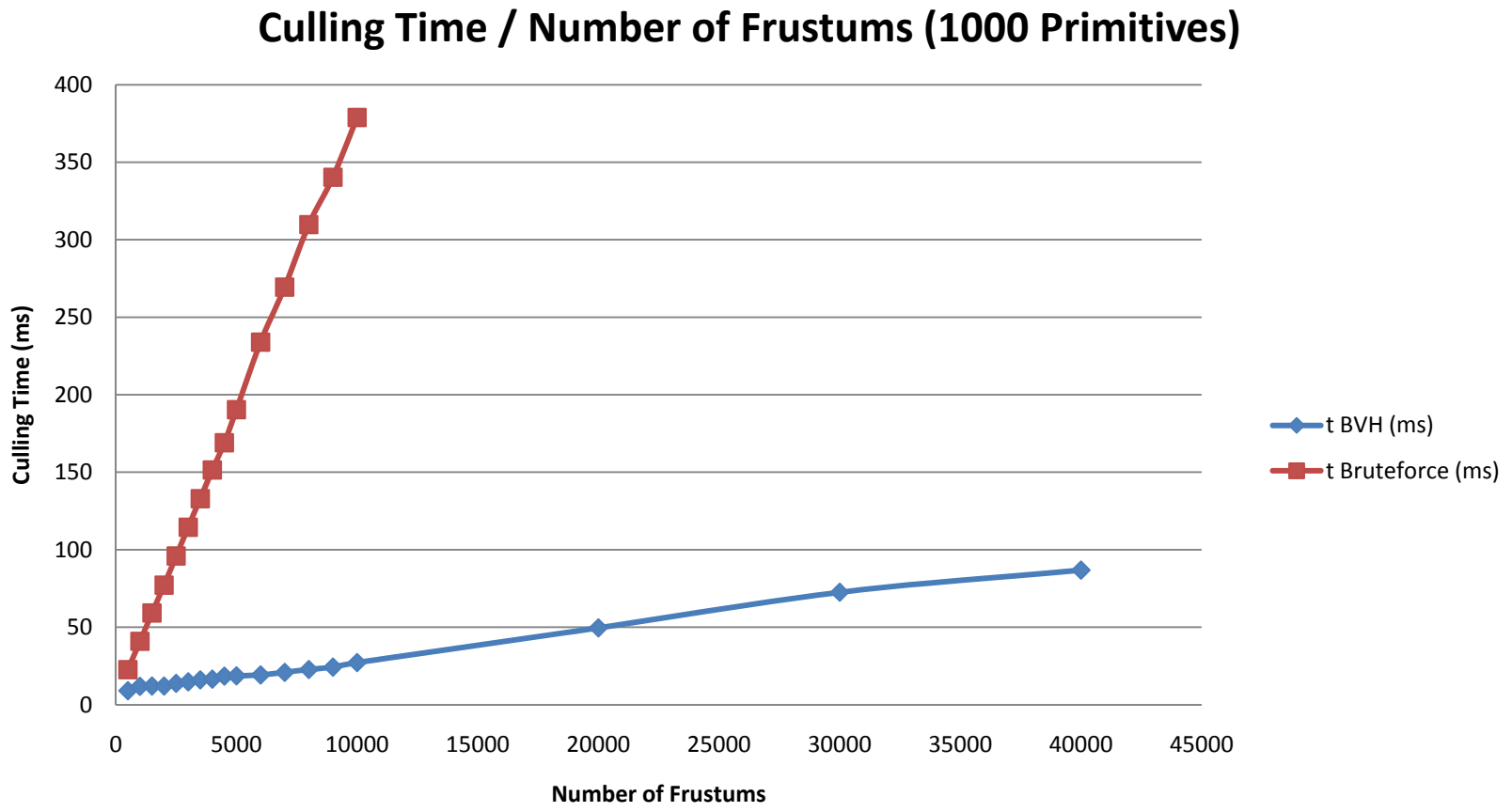
Résultats

- Temps de calcul de visibilité



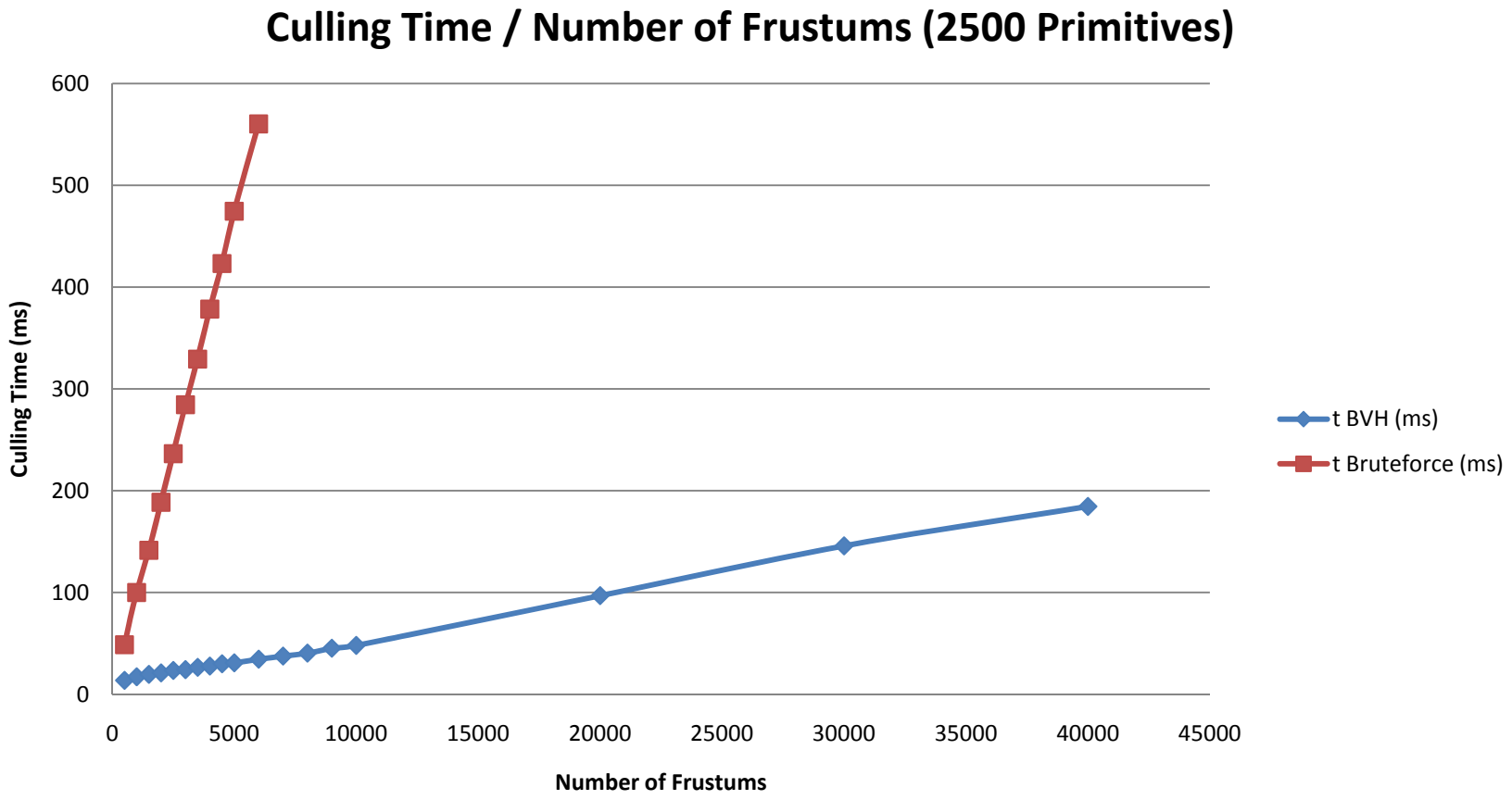
Résultats

- Temps de calcul de visibilité



Résultats

- Temps de calcul de visibilité



Démonstration