

# FMIN313 Moteurs de jeux

## Génération de terrains

DUPÉRON Georges  
BONAVERO Yoann

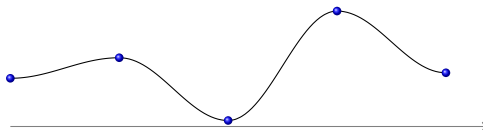
Université Montpellier II,  
Département informatique,  
Master 2 IFPRU  
Encadrants : F. Koriche et M. Moulis

Lundi 14 novembre 2011

# Plan

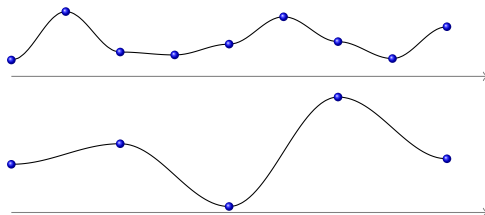
- 1 Génération
  - Perlin noise
  - Craters et Hills Algorithm
  - Érosion
  - Autres
  - Rivières
  - Démonstration
- 2 Rendu
  - Isosurfaces
  - Ray casting
- 3 Niveau de détail
  - ROAM
  - Geometry clipmaps
  - Notre algo
  - Streaming de scène

# Perlin noise



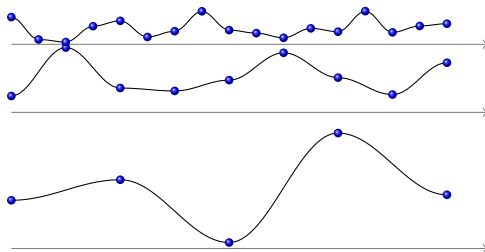
- Superposition d'octaves de bruit.

# Perlin noise



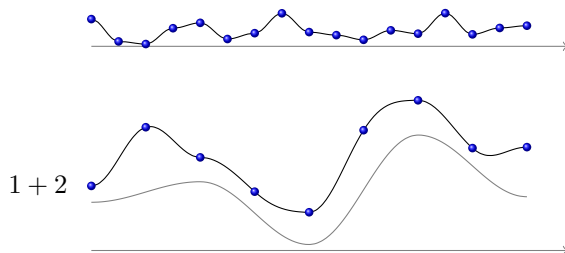
- Superposition d'octaves de bruit.

# Perlin noise



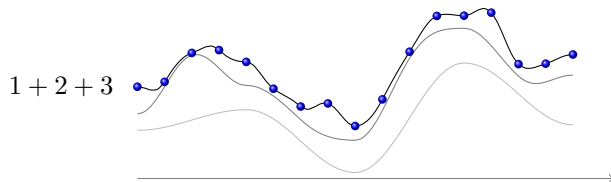
- Superposition d'octaves de bruit.

# Perlin noise



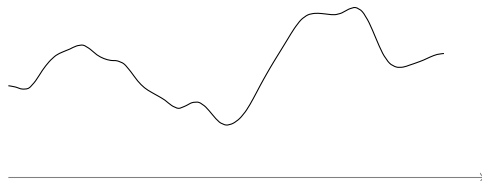
- Superposition d'octaves de bruit.

# Perlin noise



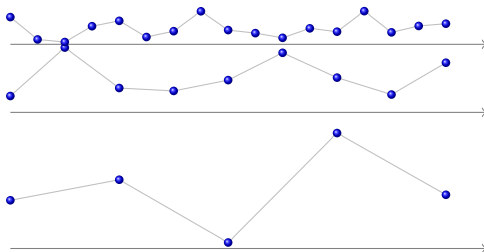
- Superposition d'octaves de bruit.

# Perlin noise



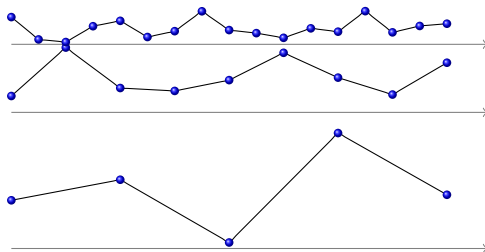
- Superposition d'octaves de bruit.

# Perlin noise



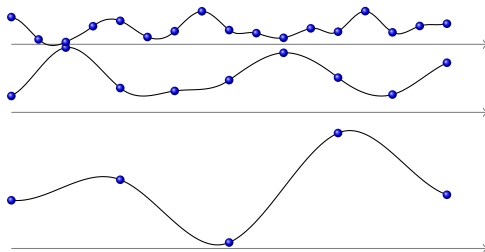
- Superposition d'octaves de bruit.
- Interpolation

# Perlin noise



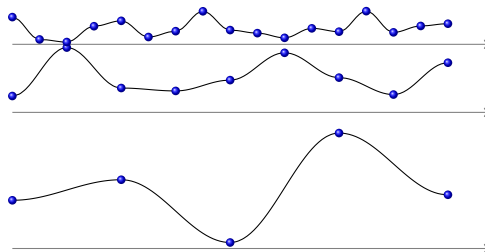
- Superposition d'octaves de bruit.
- Interpolation linéaire

# Perlin noise



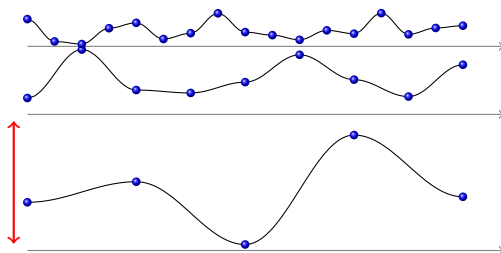
- Superposition d'octaves de bruit.
- Interpolation linéaire, cubique

# Perlin noise



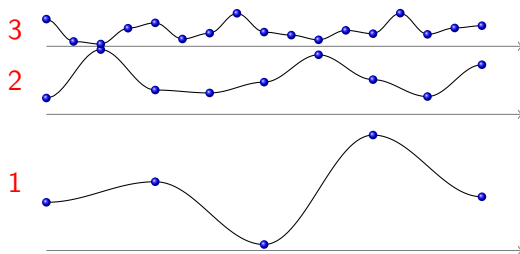
- Superposition d'octaves de bruit.
- Interpolation linéaire, cubique ou cosinusoidale.

# Perlin noise



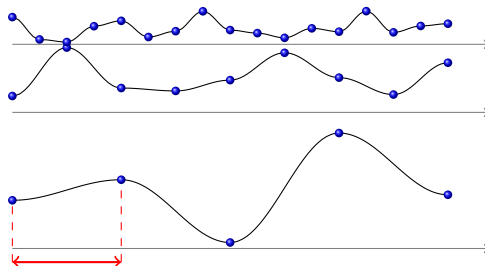
- Superposition d'octaves de bruit.
- Interpolation linéaire, cubique ou cosinusoidale.
- Amplitude

# Perlin noise



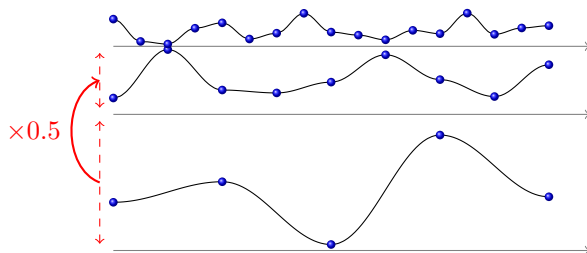
- Superposition d'octaves de bruit.
- Interpolation linéaire, cubique ou cosinusoidale.
- Amplitude, octaves

# Perlin noise



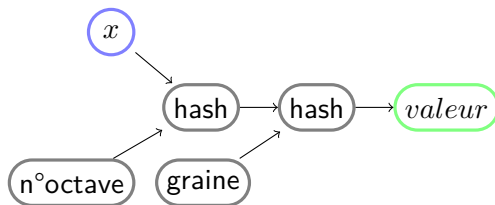
- Superposition d'octaves de bruit.
- Interpolation linéaire, cubique ou cosinusoidale.
- Amplitude, octaves, fréquence

# Perlin noise



- Superposition d'octaves de bruit.
- Interpolation linéaire, cubique ou cosinusoidale.
- Amplitude, octaves, fréquence, persistance.

# Perlin noise

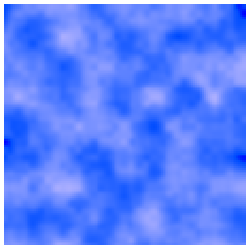


- Superposition d'octaves de bruit.
- Interpolation linéaire, cubique ou cosinusoidale.
- Amplitude, octaves, fréquence, persistance.
- Hash de coordonnées.

# Perlin noise (Variations)

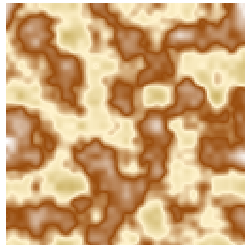
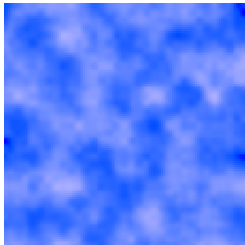
- Bruit  $n$ D et voxels : cavernes

# Perlin noise (Variations)



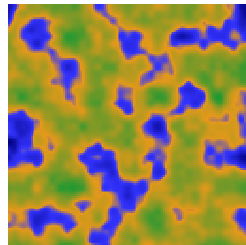
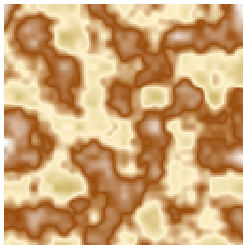
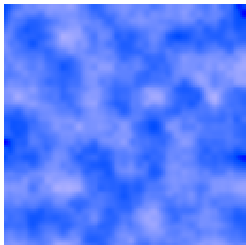
- Bruit  $n$ D et voxels : cavernes, nuages

# Perlin noise (Variations)



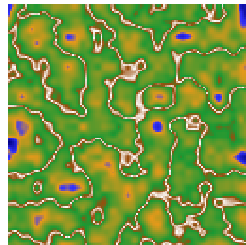
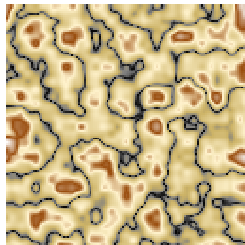
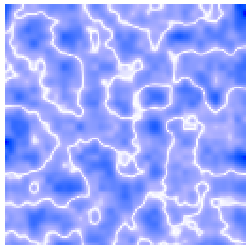
- Bruit  $n$ D et voxels : cavernes, nuages, textures

# Perlin noise (Variations)



- Bruit  $n$ D et voxels : cavernes, nuages, textures, terrains.

# Perlin noise (Variations)



- Bruit  $n$ D et voxels : cavernes, nuages, textures, terrains.
- Ridged Perlin Noise.

# Perlin noise (Variations)



- Bruit  $n$ D et voxels : cavernes, nuages, textures, terrains.
- Ridged Perlin Noise.
- Midpoint displacement.

# Perlin noise (Variations)



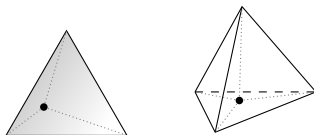
- Bruit  $n$ D et voxels : cavernes, nuages, textures, terrains.
- Ridged Perlin Noise.
- Midpoint displacement.

# Perlin noise (Variations)



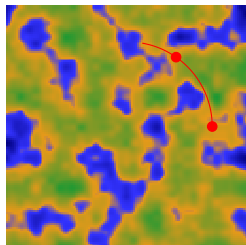
- Bruit  $n$ D et voxels : cavernes, nuages, textures, terrains.
- Ridged Perlin Noise.
- Midpoint displacement.

# Perlin noise (Variations)



- Simplex noise.  $O(d^2)$  au lieu de  $O(2^d)$ .

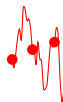
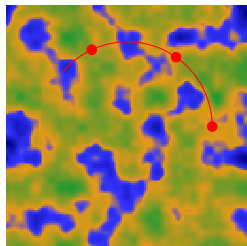
# Perlin noise (Variations)



- Simplex noise.  $O(d^2)$  au lieu de  $O(2^d)$ .
- Bruit répétable 1D : Cercle dans un espace 2D.  
Bruit répétable  $nD$  : hypercercle dans un espace  $2nD$ .

<http://www.gamedev.net/blog/33/entry-2138456-seamless-noise/>

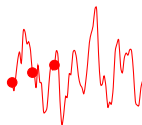
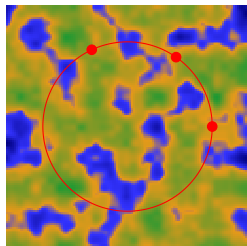
# Perlin noise (Variations)



- Simplex noise.  $O(d^2)$  au lieu de  $O(2^d)$ .
- Bruit répétable 1D : Cercle dans un espace 2D.  
Bruit répétable  $nD$  : hypercercle dans un espace  $2nD$ .

<http://www.gamedev.net/blog/33/entry-2138456-seamless-noise/>

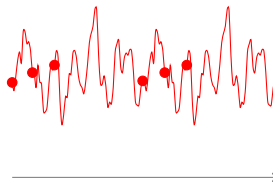
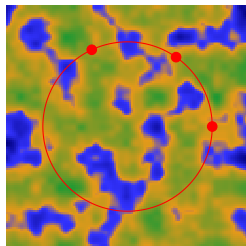
# Perlin noise (Variations)



- Simplex noise.  $O(d^2)$  au lieu de  $O(2^d)$ .
- Bruit répétable 1D : Cercle dans un espace 2D.  
Bruit répétable  $nD$  : hypercercle dans un espace  $2nD$ .

<http://www.gamedev.net/blog/33/entry-2138456-seamless-noise/>

# Perlin noise (Variations)



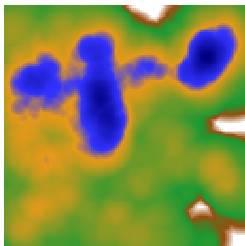
- Simplex noise.  $O(d^2)$  au lieu de  $O(2^d)$ .
- Bruit répétable 1D : Cercle dans un espace 2D.  
Bruit répétable  $nD$  : hypercercle dans un espace  $2nD$ .

<http://www.gamedev.net/blog/33/entry-2138456-seamless-noise/>

# Craters et Hills Algorithm

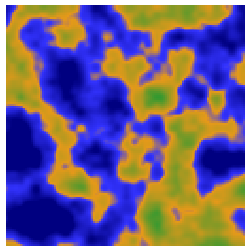
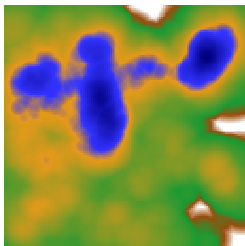
- Craters
  - Soustraire des cercles au terrain.  
( $z = z - f(\text{distance au centre})$ )

# Craters et Hills Algorithm



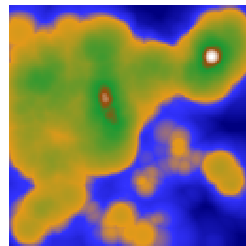
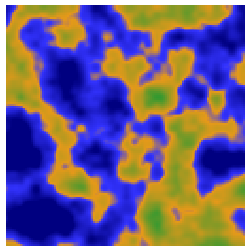
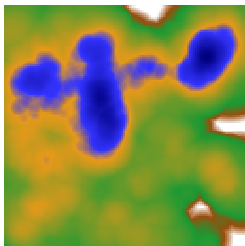
- Craters
  - Soustraire des cercles au terrain.  
( $z = z - f(\text{distance au centre})$ )
  - Sur un terrain nu.

# Craters et Hills Algorithm



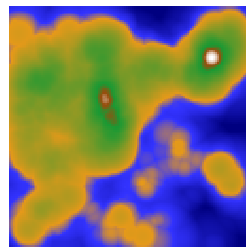
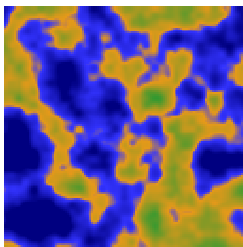
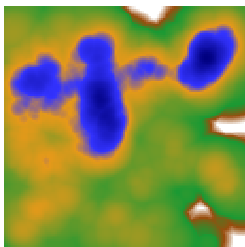
- Craters
  - Soustraire des cercles au terrain.  
( $z = z - f(\text{distance au centre})$ )
  - Sur un terrain nu.
  - Sur un terrain existant.

# Craters et Hills Algorithm



- Craters
  - Soustraire des cercles au terrain.  
( $z = z - f(\text{distance au centre})$ )
  - Sur un terrain nu.
  - Sur un terrain existant.
- Hills Algorithm : ajouter des cercles.

# Craters et Hills Algorithm



- Craters
  - Soustraire des cercles au terrain.  
( $z = z - f(\text{distance au centre})$ )
  - Sur un terrain nu.
  - Sur un terrain existant.
- Hills Algorithm : ajouter des cercles.
- Stockage des cercles dans un arbre (BSP, Quadtree, LOD, ...).

# Érosion

- Déplacement de sédiments.

# Érosion

- Déplacement de sédiments.
- Taux en fonction de la pente, dureté de la roche, végétation.

# Érosion

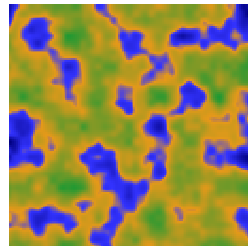
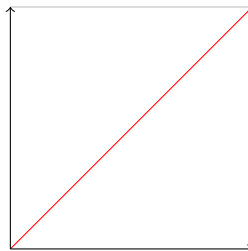
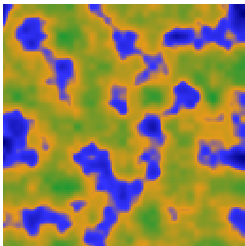
- Déplacement de sédiments.
- Taux en fonction de la pente, dureté de la roche, végétation.
- Carte de circulation des eaux.

# Érosion

- Déplacement de sédiments.
- Taux en fonction de la pente, dureté de la roche, végétation.
- Carte de circulation des eaux.
- Pas temps-réel.

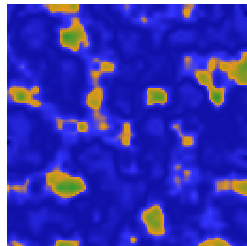
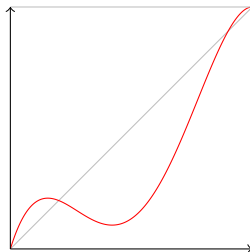
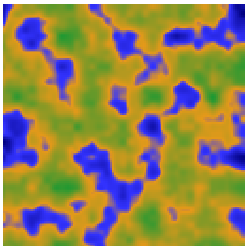
# Érosion

- Déplacement de sédiments.
- Taux en fonction de la pente, dureté de la roche, végétation.
- Carte de circulation des eaux.
- Pas temps-réel.
- Approximation : modification de la distribution des hauteurs.



# Érosion

- Déplacement de sédiments.
- Taux en fonction de la pente, dureté de la roche, végétation.
- Carte de circulation des eaux.
- Pas temps-réel.
- Approximation : modification de la distribution des hauteurs.



# Autres méthodes

- Chaînage d'algorithmes de bruit :
  - Ajout de couleurs, climats, végétation, relief. . .
  - Altération du comportement d'un algo.

<http://www.gamedev.net/blog/33/entry-2249260-procedural-islands-redux/>

# Autres méthodes

- Chaînage d'algorithmes de bruit :
  - Ajout de couleurs, climats, végétation, relief. . .
  - Altération du comportement d'un algo.

<http://www.gamedev.net/blog/33/entry-2249260-procedural-islands-redux/>

- Cartes polygonales.

<http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/>

# Autres méthodes

- Chaînage d'algorithmes de bruit :
  - Ajout de couleurs, climats, végétation, relief. . .
  - Altération du comportement d'un algo.

<http://www.gamedev.net/blog/33/entry-2249260-procedural-islands-redux/>

- Cartes polygonales.

<http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/>

- Intégration de formes dans le terrain.

# Rivières

- Pathfinding. <http://www.umbrarumregnum.net/articles/creating-rivers>

# Rivières

- Pathfinding. <http://www.umbrarumregnum.net/articles/creating-rivers>
- Affinage du tracé en fonction du LOD.

# Rivières

- Pathfinding. <http://www.umbrarumregnum.net/articles/creating-rivers>
- Affinage du tracé en fonction du LOD.
- Tracé arbitraire.

# Rivières

- Pathfinding. <http://www.umbrarumregnum.net/articles/creating-rivers>
- Affinage du tracé en fonction du LOD.
- Tracé arbitraire.
- Intégration dans le terrain.

# Démonstration

## World machine

<http://www.world-machine.com/>

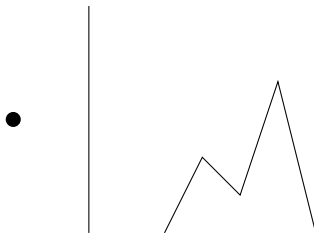
# Plan

- 1 Génération
  - Perlin noise
  - Craters et Hills Algorithm
  - Érosion
  - Autres
  - Rivières
  - Démonstration
- 2 Rendu
  - Isosurfaces
  - Ray casting
- 3 Niveau de détail
  - ROAM
  - Geometry clipmaps
  - Notre algo
  - Streaming de scène

# Isosurfaces

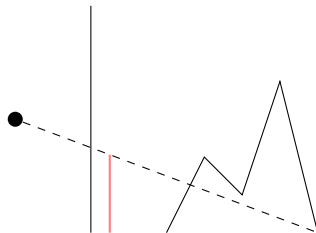
- Metaballs. `/usr/lib/xscreensaver/metaballs`
- Surface 2D d'un bruit 3D.
- Simplification de nuages.
- Surface de l'eau.

# Ray casting



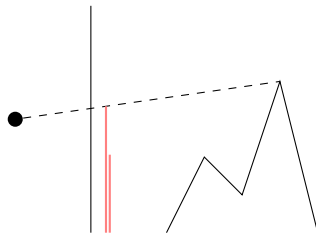
- Très simple, très petit code.

# Ray casting



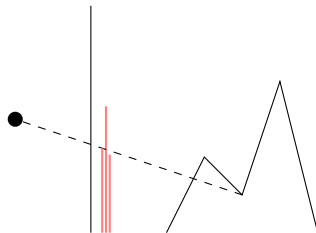
- Très simple, très petit code.

# Ray casting



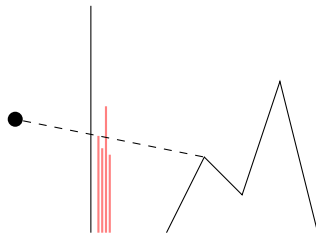
- Très simple, très petit code.

# Ray casting



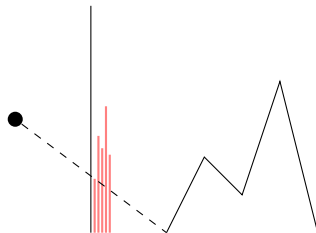
- Très simple, très petit code.

# Ray casting



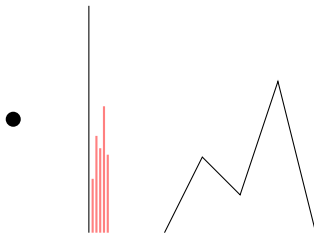
- Très simple, très petit code.

# Ray casting



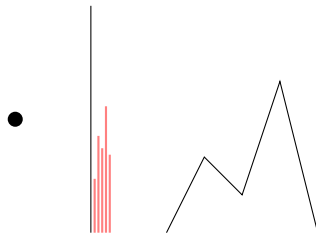
- Très simple, très petit code.

# Ray casting



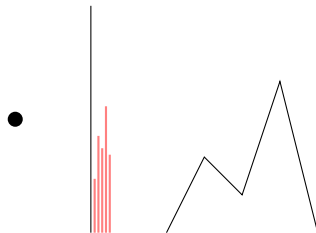
- Très simple, très petit code.

# Ray casting



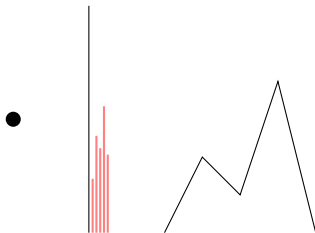
- Très simple, très petit code.
- Sampling.

# Ray casting



- Très simple, très petit code.
- Sampling.
- Très lent.

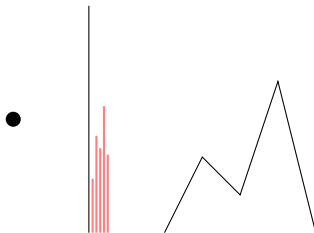
# Ray casting



- Très simple, très petit code.
- Sampling.
- Très lent.
- Démonstration.

<http://forum.osdev.org/viewtopic.php?p=170625#p170625>

# Ray casting



- Très simple, très petit code.
- Sampling.
- Très lent.
- Démonstration.

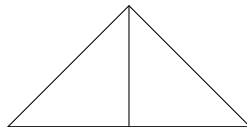
<http://forum.osdev.org/viewtopic.php?p=170625#p170625>

- Monte Carlo.

# Plan

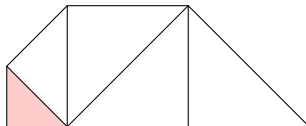
- 1 Génération
  - Perlin noise
  - Craters et Hills Algorithm
  - Érosion
  - Autres
  - Rivières
  - Démonstration
- 2 Rendu
  - Isosurfaces
  - Ray casting
- 3 Niveau de détail
  - ROAM
  - Geometry clipmaps
  - Notre algo
  - Streaming de scène

# ROAM



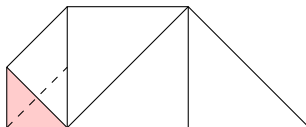
- Triangle bintree.

# ROAM



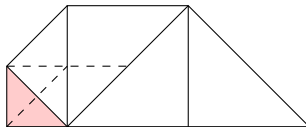
- Triangle bintree.
- Split et merge. CLOD.

# ROAM



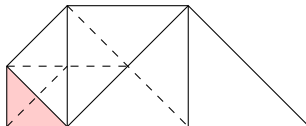
- Triangle bintree.
- Split et merge. CLOD.

# ROAM



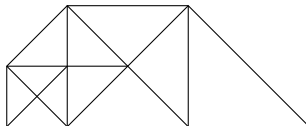
- Triangle bintree.
- Split et merge. CLOD.

# ROAM



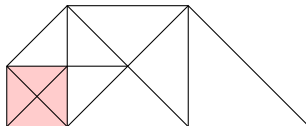
- Triangle bintree.
- Split et merge. CLOD.

# ROAM



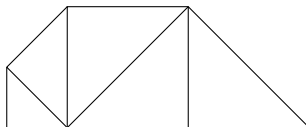
- Triangle bintree.
- Split et merge. CLOD.

# ROAM



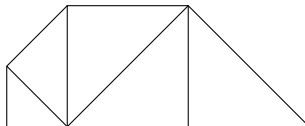
- Triangle bintree.
- Split et merge. CLOD.

# ROAM



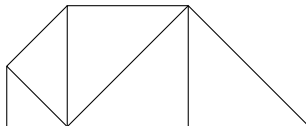
- Triangle bintree.
- Split et merge. CLOD.
- Défaut maximal visible à l'écran.

# ROAM



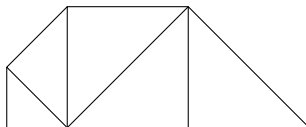
- Triangle bintree.
- Split et merge. CLOD.
- Défaut maximal visible à l'écran.
- Split queue et Merge queue.

# ROAM



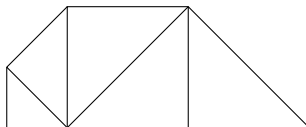
- Triangle bintree.
- Split et merge. CLOD.
- Défaut maximal visible à l'écran.
- Split queue et Merge queue.
- Frustum culling : utilisation de drapeaux.

# ROAM



- Triangle bintree.
- Split et merge. CLOD.
- Défaut maximal visible à l'écran.
- Split queue et Merge queue.
- Frustum culling : utilisation de drapeaux.
- Améliorations :
  - Triangle stripping.
  - Geomorphing.
  - Calcul différé des priorités dans les queues.
  - Temps réel.

# ROAM

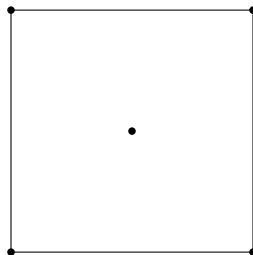


- Triangle bintree.
- Split et merge. CLOD.
- Défaut maximal visible à l'écran.
- Split queue et Merge queue.
- Frustum culling : utilisation de drapeaux.
- Améliorations :
  - Triangle stripping.
  - Geomorphing.
  - Calcul différé des priorités dans les queues.
  - Temps réel.
- $\tilde{O}$ (Nb triangles mis à jour)

# Geometry clipmaps

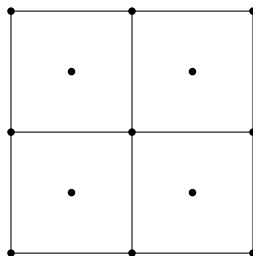
- Carrés concentriques avec des LOD différents.
- Comme le mipmapping de textures.
- Utilisation forte du GPU.
- Displacement shader.
- Remplissage à la jointure des LOD.

# Notre algorithme



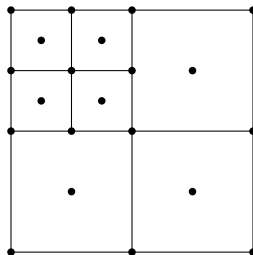
- Quadtree de carrés.
- Triangle fans.
- LOD en fonction de la distance.
- Mise à jour de quelques branches seulement.
- Temps réel.
- $\tilde{O}$ (Nb carrés mis à jour)

# Notre algorithme



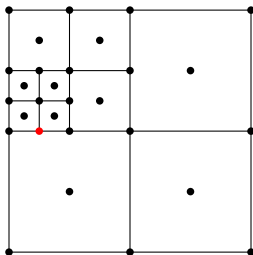
- Quadtree de carrés.
- Triangle fans.
- LOD en fonction de la distance.
- Mise à jour de quelques branches seulement.
- Temps réel.
- $\tilde{O}(\text{Nb carrés mis à jour})$

# Notre algorithme



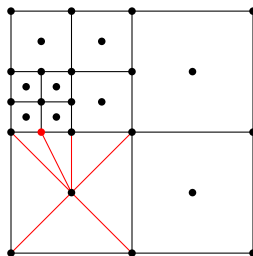
- Quadtree de carrés.
- Triangle fans.
- LOD en fonction de la distance.
- Mise à jour de quelques branches seulement.
- Temps réel.
- $\tilde{O}(\text{Nb carrés mis à jour})$

# Notre algorithme



- Quadtree de carrés.
- Triangle fans.
- LOD en fonction de la distance.
- Mise à jour de quelques branches seulement.
- Temps réel.
- $\tilde{O}(\text{Nb carrés mis à jour})$

# Notre algorithme



- Quadtree de carrés.
- Triangle fans.
- LOD en fonction de la distance.
- Mise à jour de quelques branches seulement.
- Temps réel.
- $\tilde{O}(\text{Nb carrés mis à jour})$

# Streaming de scène

- Modèle client/serveur.
- Tiles avec LOD maximal.
- Qualité progressive des tiles.
- Geometry clipmaps.
- Démonstration. `/usr/lib/xscreensaver/crackberg`

# Sources

- Perlin noise. [http://freespace.virgin.net/hugo.elias/models/m\\_perlin.htm](http://freespace.virgin.net/hugo.elias/models/m_perlin.htm)
- <http://www.gamasutra.com>
- <http://vterrain.org>
- <http://world-machine.com>
- Algorithmes de bruit. <http://www.sluniverse.com/php/vb/project-development/34994-automatically-generated-terrain-map.html>
- Composition d'algorithmes de bruit.  
<http://www.gamedev.net/blog/33/entry-2249260-procedural-islands-redux/>
- Création de cartes polygonales.  
<http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/>
- Pathfinding pour créer des rivières.  
<http://www.umbrarumregnum.net/articles/creating-rivers>

# À propos

- Utilise pgf/tikz 2.10 .
- Graine aléatoire : 0.99661 .