Particules, imposteurs/billboarding

Exemple de système de particules

Chute libre de particules

- Représentation géométrique des particules: sphere
- Equation du mouvement $p(t) = \frac{1}{2}gt^2 + v_0 t + p_0$

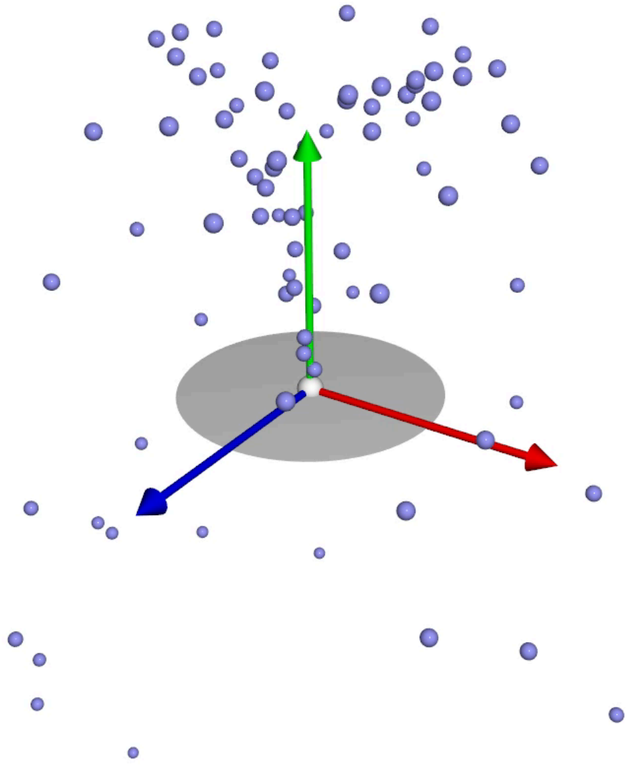
- Position initiale et vitesse peuvent avoir des composantes aléatoires
- Chaque particule peut avoir une durée de vie limitée

Q. *Quels sont les paramètres utilisés pour p_0 et v_0 dans cet example ?*

$p_0 = \ldots$

$v_0 = \ldots$

Note: Système de coordonnées (x, y, z) =(rouge, vert, bleu).



Rebond de sphères

Q. Quelle est l'équation du mouvement ?
(en prenant en compte un rebond)
$p(t) = ...$

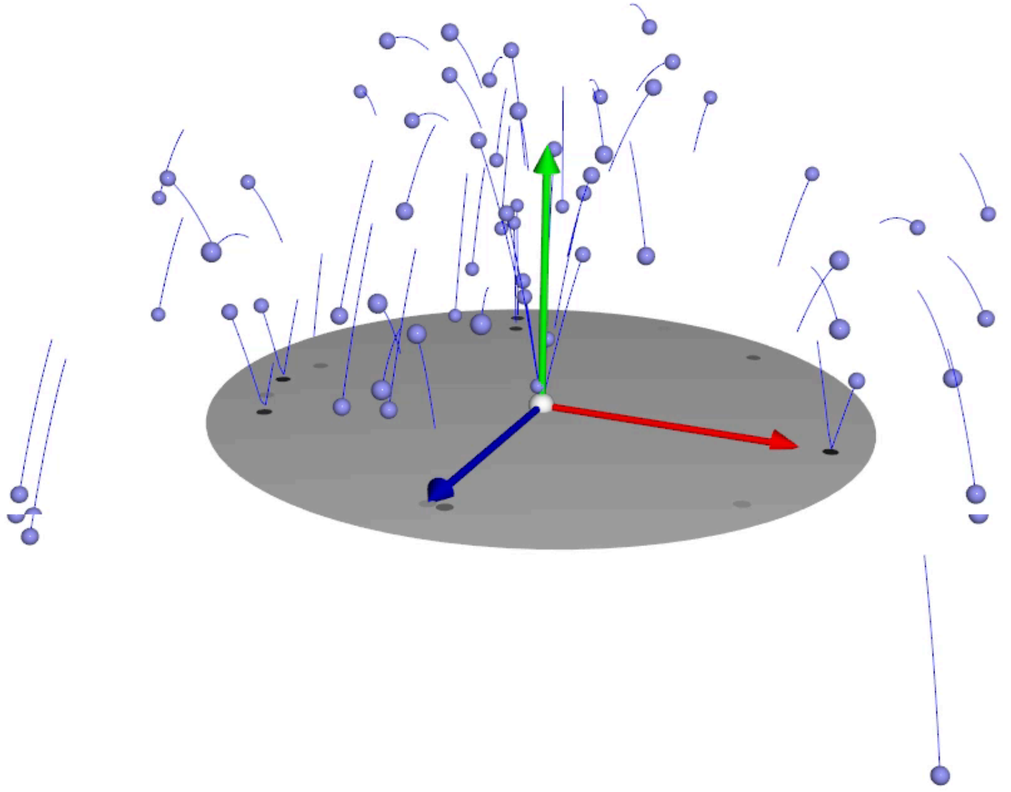
Aide:

- Considère une particule émise à $t = 0$

- A quel temps t_i, la particule touche le sol ?

- Quelle est la nouvelle vitesse après impact ?

- Exprimer l'équation du mouvement complète (définition par morceaux)



Mouvement générique

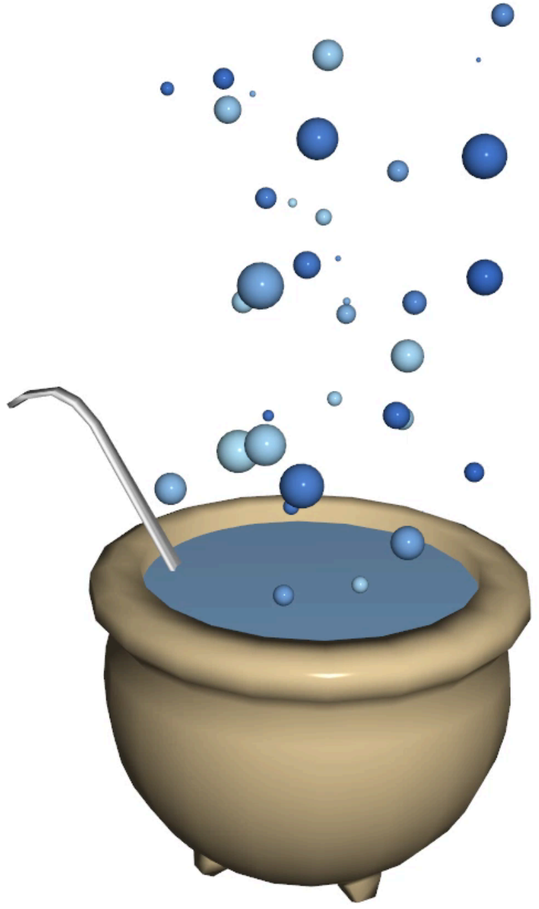
Trajectoires non restreintes à des mouvements physique.

Q. *Quels sont les paramètres (et valeurs initialles) associées à ces particules ?*

- position = ...
- radius = rand([0,0.1])
- ...

- Quelle est l'équation du mouvement d'une particule ?

- $p(t) = ...$



[Interactive view]

Billboards, Imposteurs, Sprites

Particules peuvent être affichées en tant qu'images

à la place de sphères

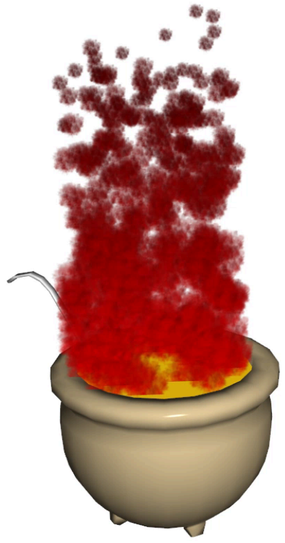
En pratique

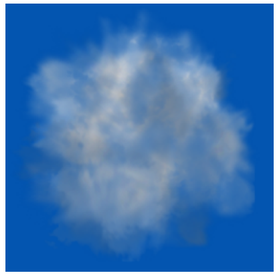
- Quand particule est affiché par un quadrangle.
- Une texture est appliqué sur le quadrangle.

- La texture peut contenir des parties transparents: La géométrie du quadrangle est invisible.

Impression d'une géométrie complexe

- Le quad peut faire face à la caméra en permanence (billboard)
- La texture peut être animée (sprite)
- La texture peut s'adapter au point de vue (imposteurs)

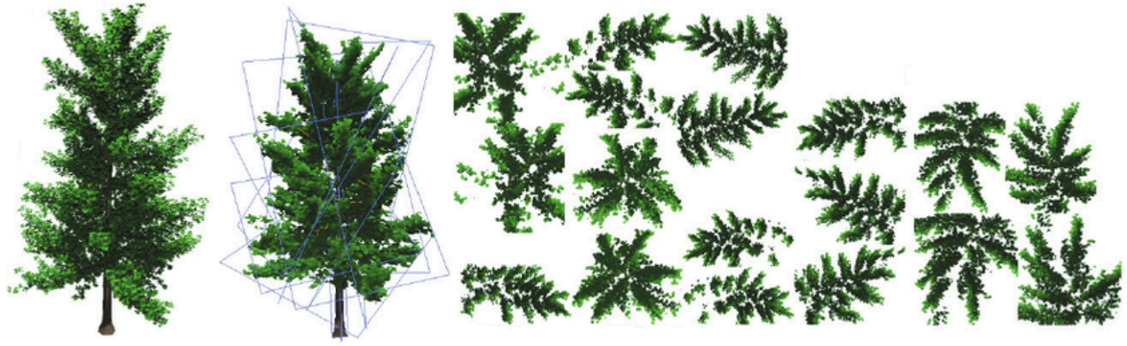


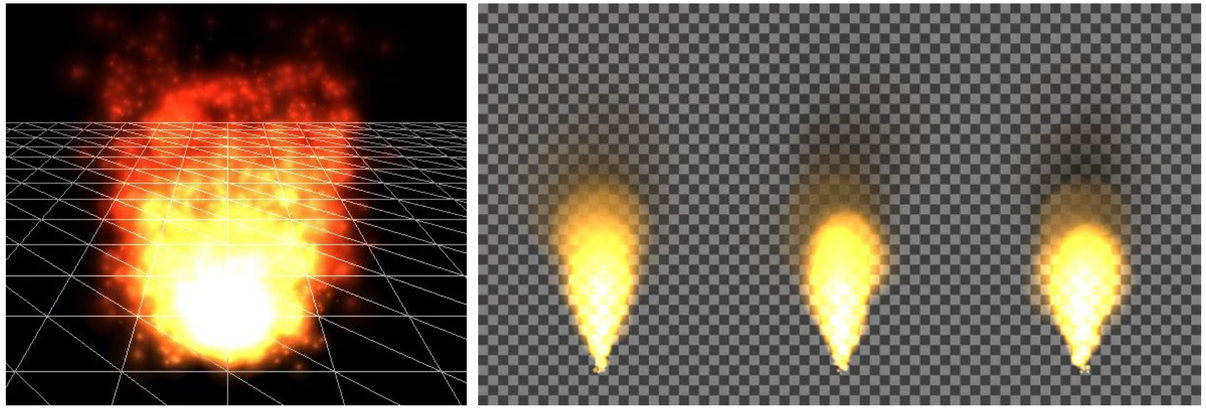


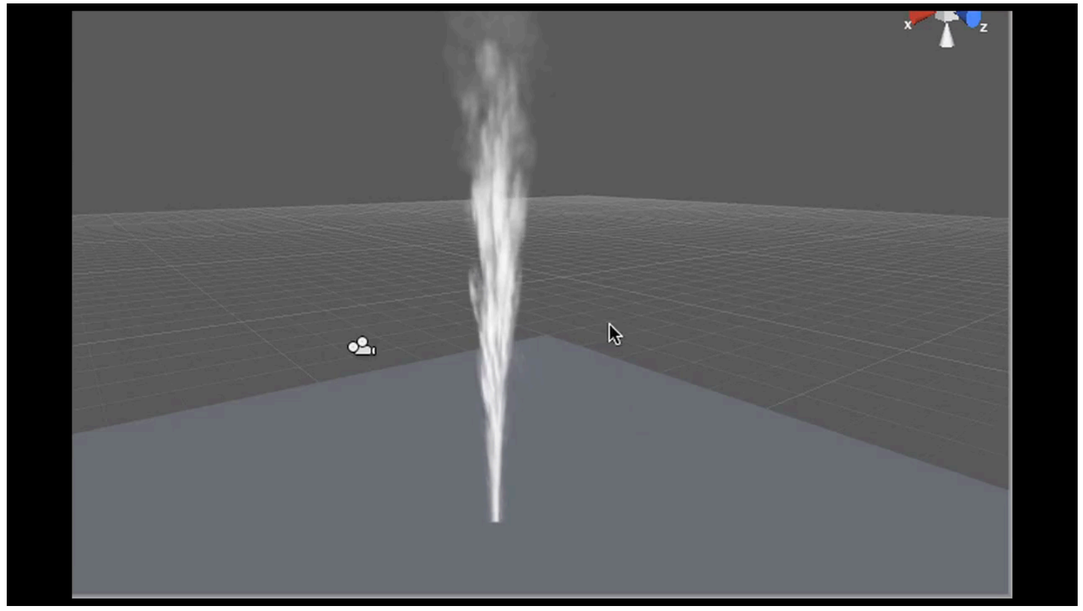
Utilisation des billboards

Utilisation classique pour les modèles complexes

Végétation, feu, fumée, etc.







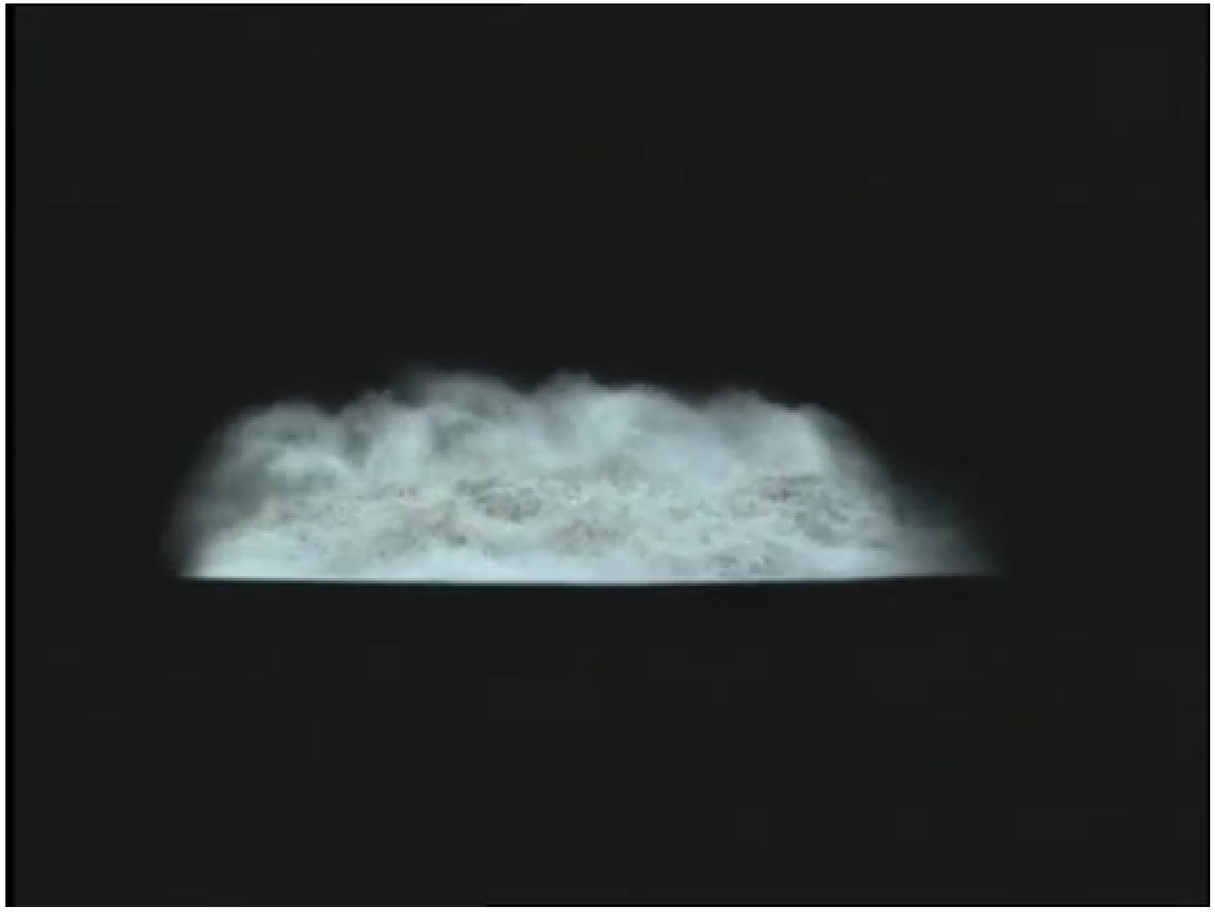
Exemple d'utilisation en production

Comment sont réalisés les *têtes de chevaux en eau* ?

The Lord of the Rings

Exemple d'utilisation en production

Comment sont réalisés les *têtes de chevaux en eau* ?



The Lord of the Rings

Exemple d'utilisation en production

Making-Of complet

The Lord of the Rings

Implementation - transparence en OpenGL

Effet de transparence non trivial avec le rendu par projection/rasterisation.

Pas de solution parfaite

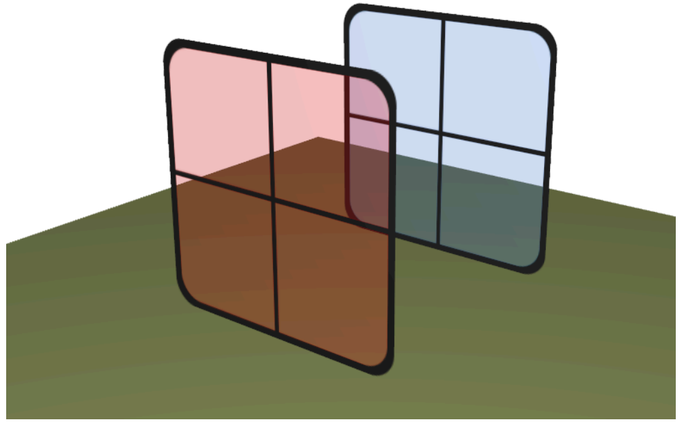
$\Rightarrow$ Possibilité de mélanger les couleurs pour chaque pixel en utilisant le canal α (rgba)

Nécessité d'activer explicitement le mode de *blending* en OpenGL

```
glEnable(GL_BLEND);
glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
 ... Draw Calls ...
glDisable(GL_BLEND);
```

- Plusieurs mode de mélange sont possibles pour différents effets (see glBlendFunc)
- For transparency: *(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)*

  $\Rightarrow$ newColor = α currentColor + $(1 - \alpha)$ previousColor



Note:

L'**ordre** des appels à draw est important (previousColor/currentColor)

Le buffer de profondeur ne peut pas être utilisé pour des objets semi-transparents.

Implementation - transparence in OpenGL

Approche générale:

- 1. Draw all **non-transparent** objects
- 2.a Activate color blending, desactivate depth buffer write
- 2.b **Sort** transparent objects by depth to the camera

$$p_{proj} = Proj \times ModelView \times p$$

$$z_{depth} = p_{proj}.z / p_{proj}.w$$

- 3. **Draw transparent objects** from furthest to closest

```
 ... Draw all non-transparent objects ...
glEnable(GL_BLEND); // Color Blending
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
glDepthMask(false); // do not write on depth buffer
 ... Sort transparent objects by depth ...
 ... Draw all transparent objects from furthest to closest ...
glDisable(GL_BLEND);
// do not forget to re-activate depth buffer write
glDepthMask(true);
```

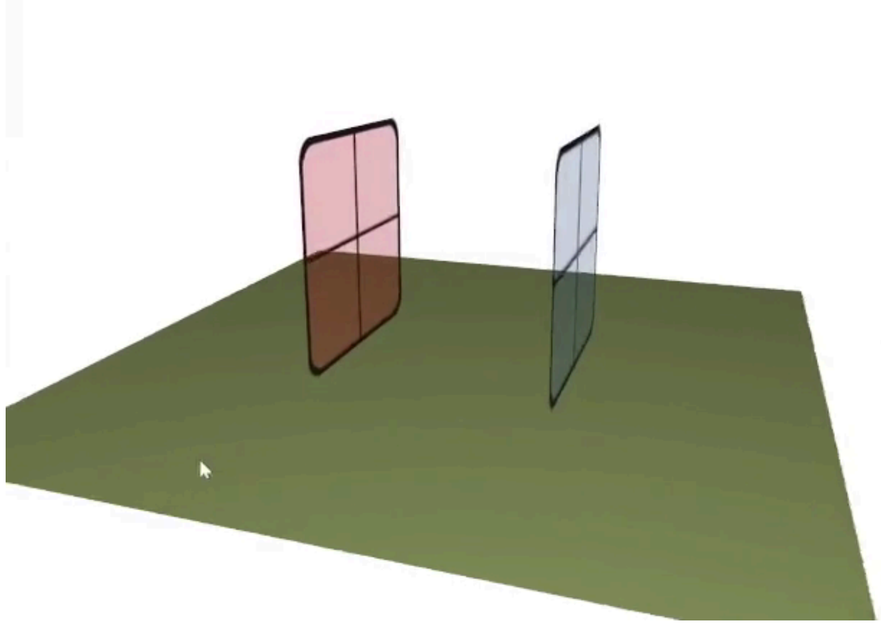
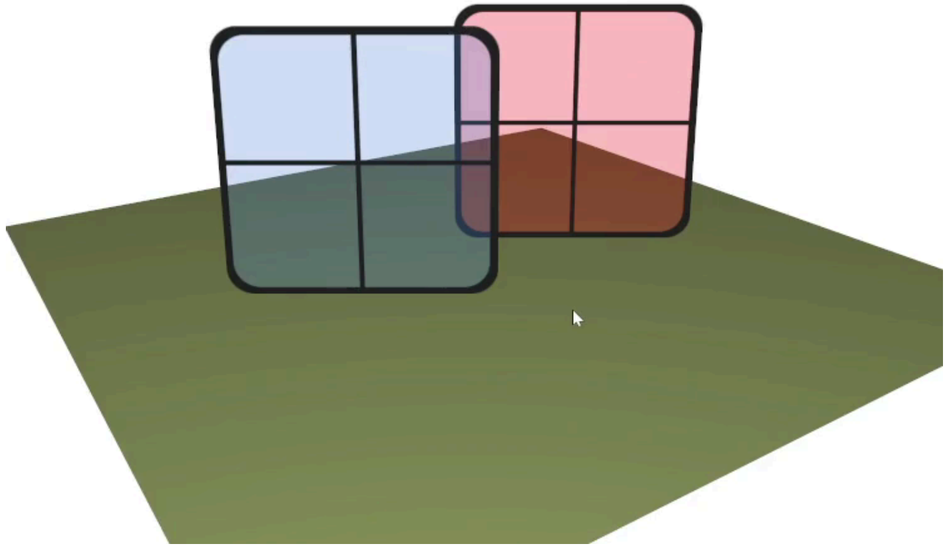


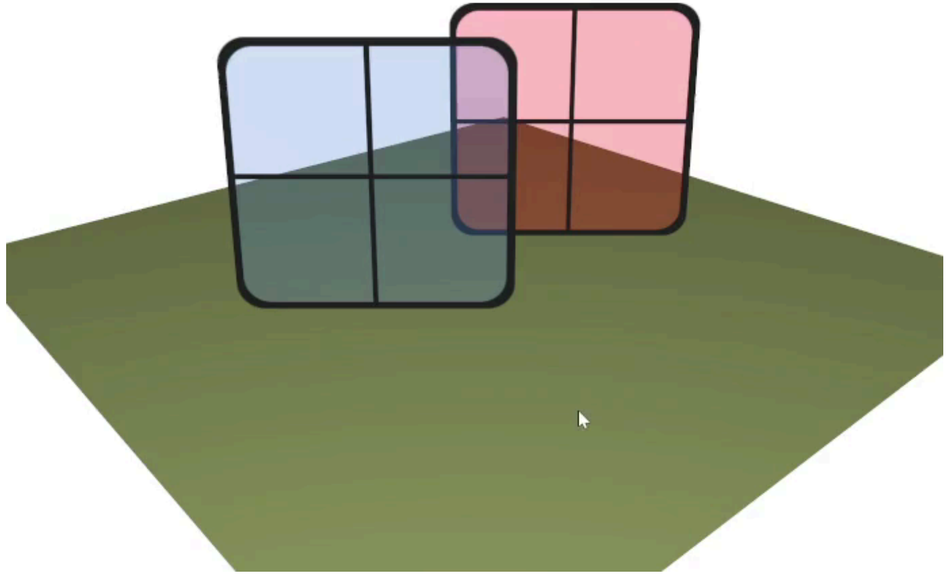
Image Blending - erreur classique

Garder l'écriture dans le buffer de profondeur (sans tris)



Problème visibles

Pas de tris, mais écriture désactivé du depth-buffer



Note:

OK pour des sprits flous (ex. fumée)
où l'ordre des éléments est difficile à distinguer.
Evite le cout du tri.

Implementation - transparence en OpenGL

Pour les imposteurs avec des fragments entièrement opaques/transparents:

discard peut être utilisé sur les fragments transparents

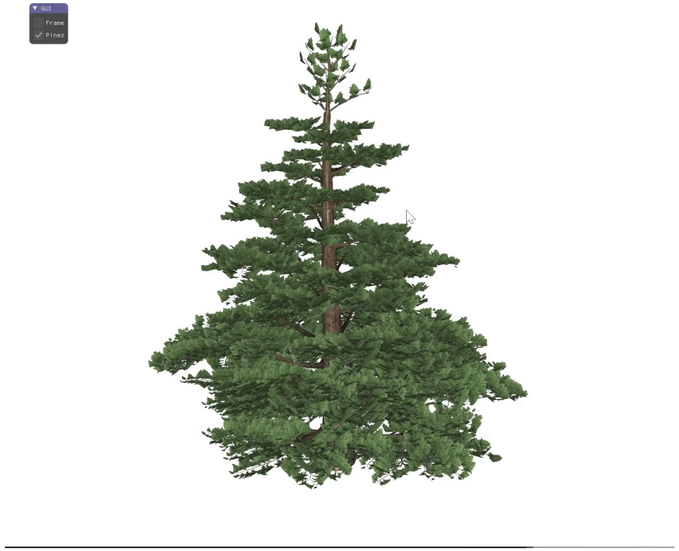
discard = Arrête l'exécution du fragment shader (rien n'est écrit)

(+) Générique, standard

pas besoin de trier les formes, le depth-buffer fonctionne

(-) Possibles artefacts d'aliasing

(-) Ne peut pas être utilisé sur des objets semi-transparents



Fragment shader:

```
float limit = 0.6;
if( texture.a < limit) {
  discard;
}
```

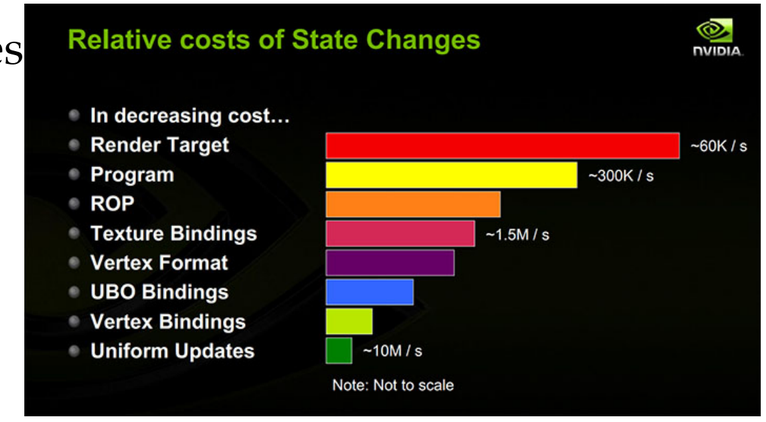
Instancing

Les système de particules nécessitent l'affichage de nombreuses formes

De multiples appels à glDrawElements peuvent être coûteux

chaque appel nécessite une synchronisation CPU-GPU

```
// Slow for large N
for(int k=0; k<N; ++k)
   someOperator(k)
   draw(shape) // ... glDrawElements(...)
```



[NVIDIA]

Solution: **Instancing**

glDrawArrays → glDrawArraysInstanced

glDrawElements → glDrawElementsInstanced

In the C++ code

```
// 1 unique call, N is a parameter
drawInstanced(shape, N) // ... glDrawElementsInstanced(...,N)
// No more explicit loop
```

In the Shader:

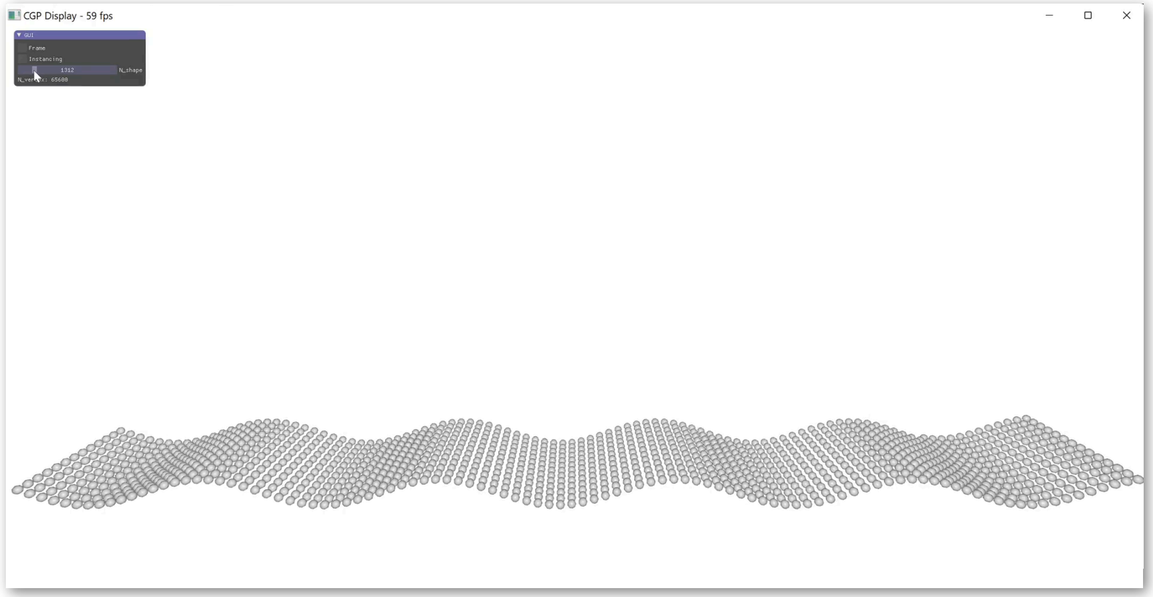
```
// ...
someOperator(gl_InstanceID)
// ...
```

(+) Full capability of your GPU

(-) Require a specific shader

Instancing Demo

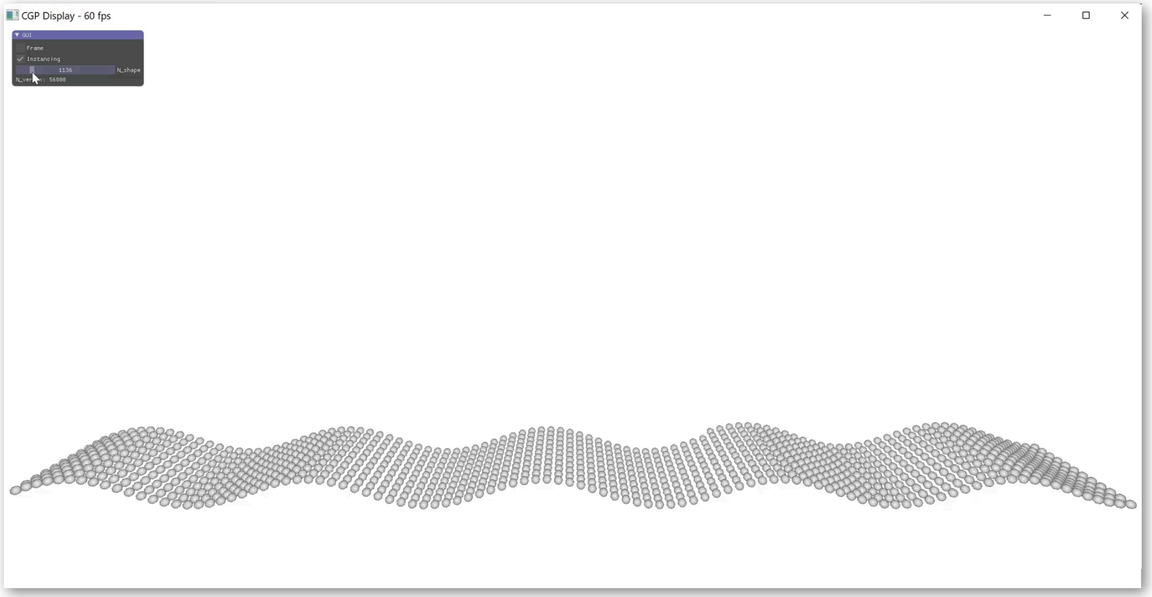
Sans Instancing



Lent pour 15,000 particles

#vertices=750,000

Avec Instancing



Affiche 450,000 particles à 60fps (dépend du GPU)

#vertices=22,000,000

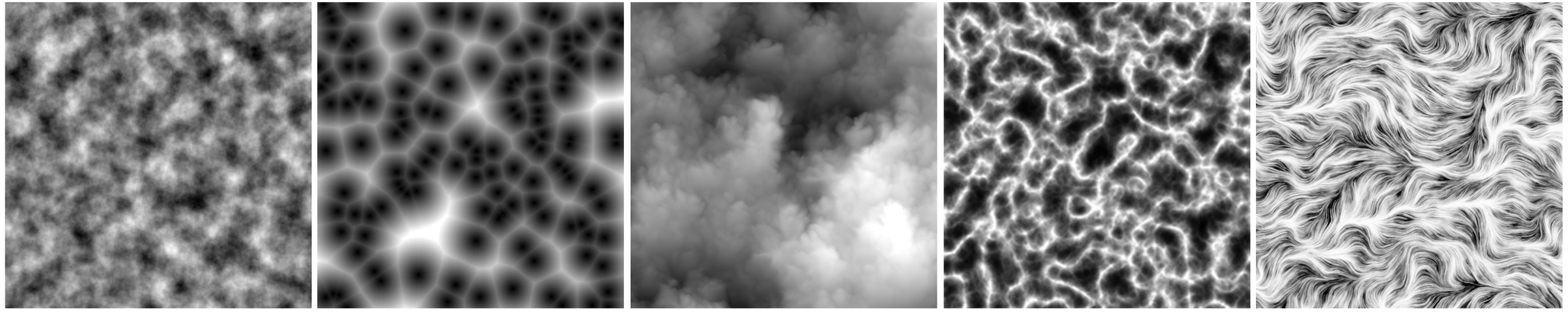
Animations procédurales

Noise function - Perlin Noise

Bruit Procéduraux

Qu'est ce qu'un bruit procédural:

- Fonction avec une structure/pattern visible (bande de fréquence limitée)
- Sans périodicité visible
- Déterministe (Même résultat pour une entrée donnée)



Exemples de textures 2D procédurales

Création d'un bruit procédural

Ex. Fonction continue $f(x)$ avec bande fréquentielle limitée.

Pour des valeurs entières: $f(n) =$ pseudo-aléatoire, déterministe

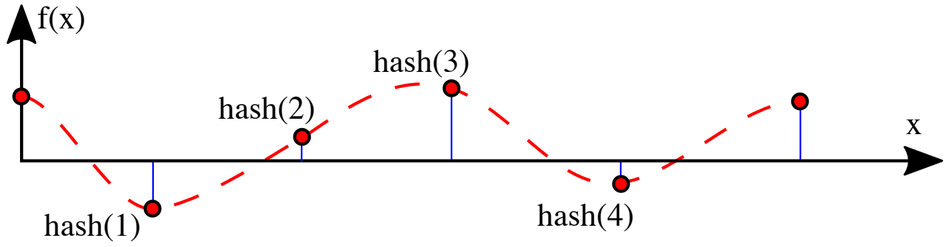
ex. Fonction de hashage: float hash(float n) { return fract(sin(n) * 1e4); }

Interpole les valeurs entières avec une courbe lisse (Smooth step, polynômes cubiques, etc.)

Propriétés:

- Fonction continue
- Apparence non périodique
- Fréquence limitée autour de 1 Hz

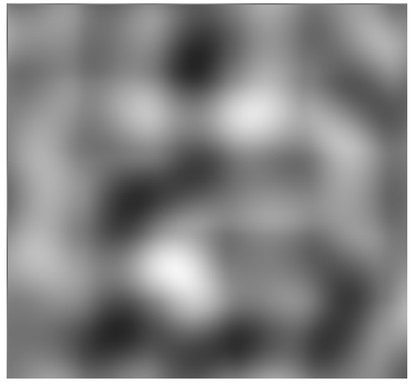
Peut être calculé en 2D, 3D, etc.



Algorithme simple - ex. en 1D

Pour un x donné, $n =$floor(x)

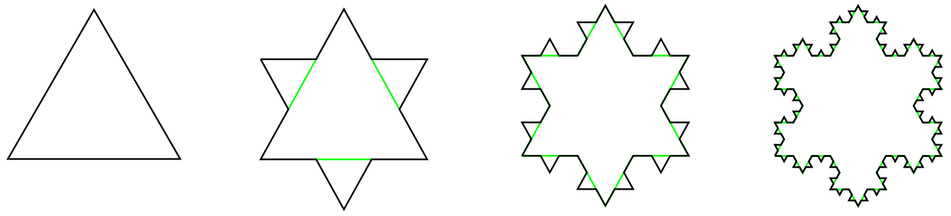
- Évaluer la fonction de hashage en n, $n+1$
  ($(n \pm i)$ pour des échantillonnages supplémentaires)
- Calculer la valeur interpolée $f(x)$ à la position x



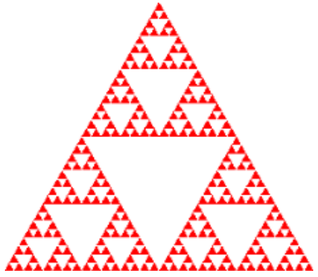
Ajout de détails à haute fréquence: Notion de fractale

Idée: Ajouter récursivement des détails auto-similaires

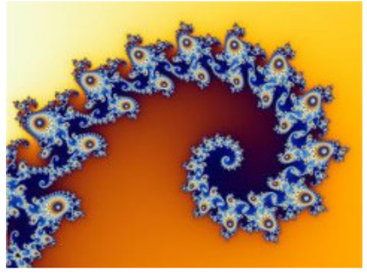
- Règle simple ⇒ formes complexes
- Peut ressembler à des détails naturels complexes



Koch Snowflake



Sierpinski triangle, Shell:Oliva Porphyria



Standard fractales (Mandelbrots, Sierpinski, racines de Newton, etc) difficiles à contrôler. Rarement utilisées directement en CG.

Bruit de Perlin

Premier bruit procédural proposé par

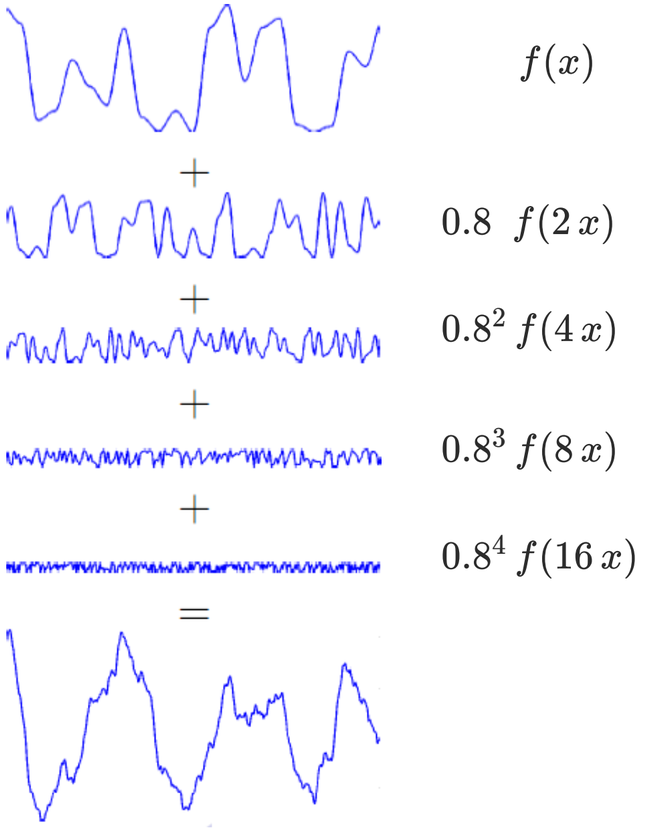
[Ken Perlin, An Image Synthesizer. SIGGRAPH 85]

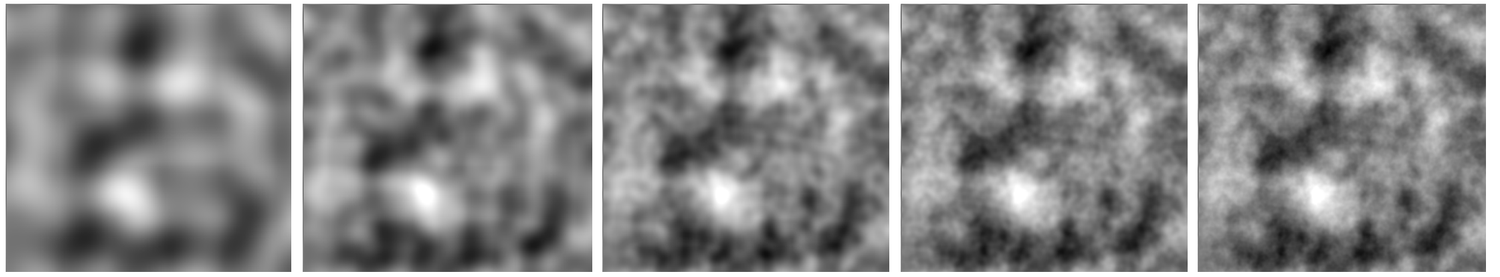
Idée: Somme de fonctions pseudo-aléatoires avec fréquences croissantes et amplitudes décroissantes

f : Smooth pseudo-random function

$$P(x) = \sum_{k=0}^{N} \alpha^k f(\omega^k\ x)$$

- N number of Octave
- α persistency ($1/\alpha$ attenuation)
- ω frequency gain

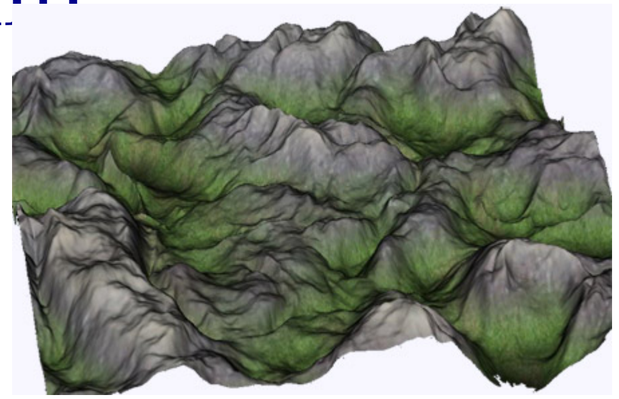




Utilisation du bruit de Perlin

Directe: $z = P(x, y)$

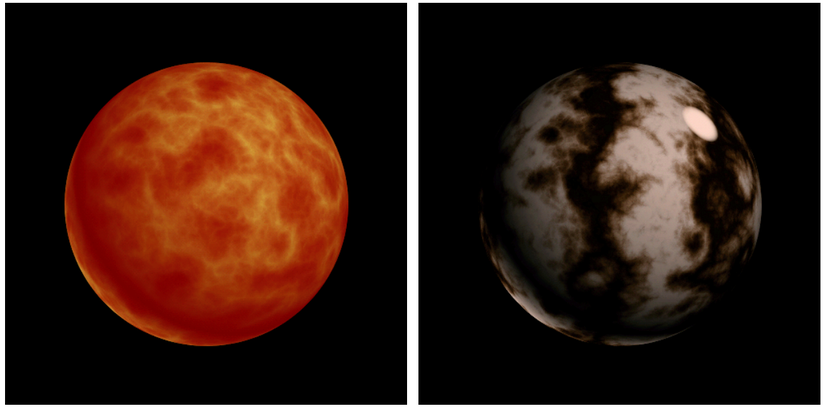
Apparence de montagne



Textures procédurales

Crête (/Ridge): $\sum_k \alpha^k |f(\omega^k x)|$

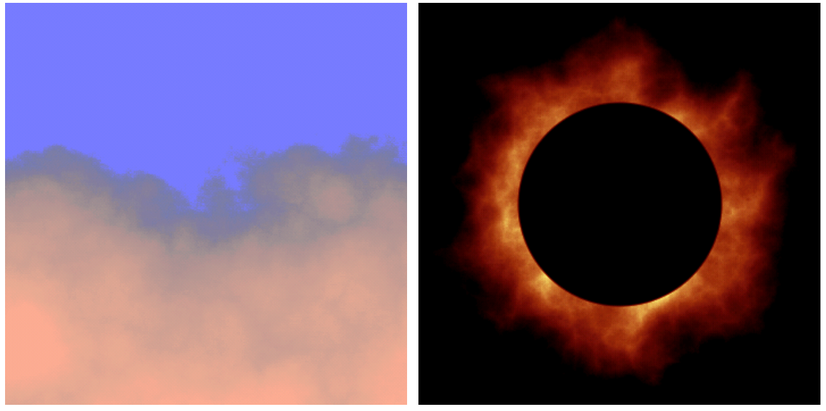
Marbre: $\sin\left(x + \sum_k \alpha^k |f(\omega^k x)|\right)$



Textures animées

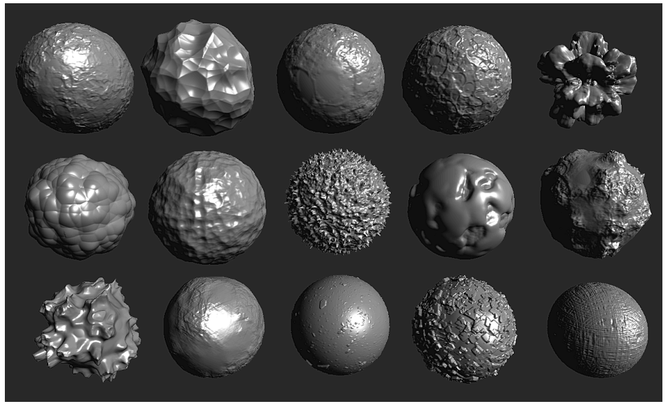
Translation: $P(x, y + t)$

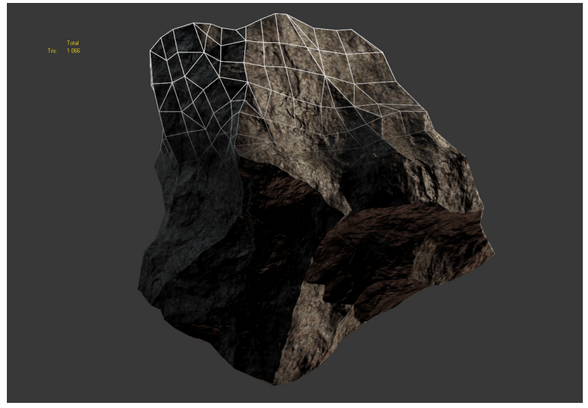
Evolution lisse: $P(x, y, t)$

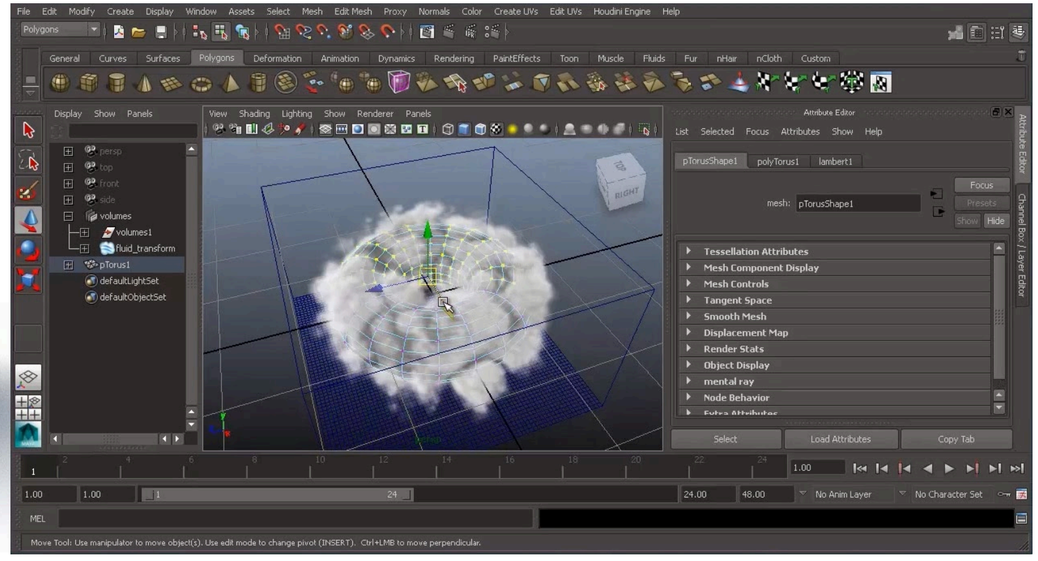


Applications du bruit de Perlin

Pour quasiment toutes les formes complexes ...







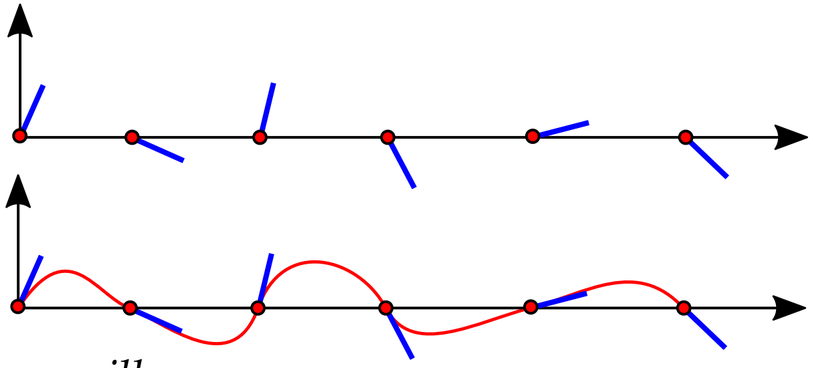
Look at Shader Toy + Noise example

Bruit de Perlin: Améliorations

Gradient Noise

Utilisation de gradients pseudo-aléatoires au lieu de positions

Contrôle de la fréquence amélioré: Oscillation à période 1



Simplex Noise

Interpolation dans des polytopes simples à la place d'une interpolation sur grille.

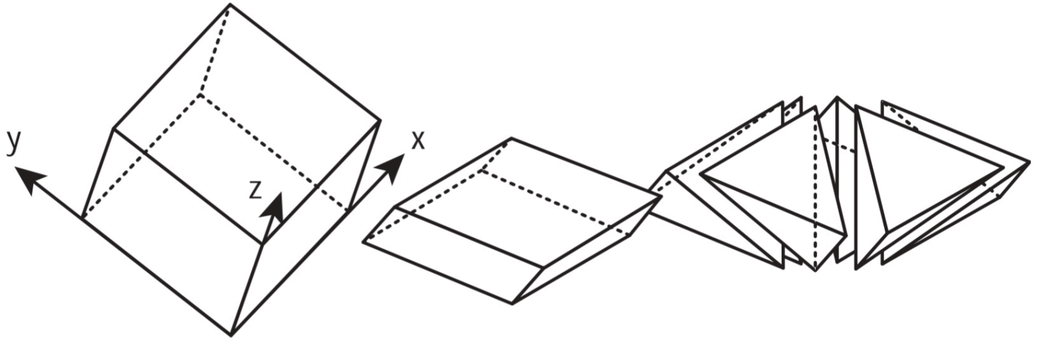
Evite les artefacts de direction de grille

Plus rapide pour des dimensions élevées

[Stefan Gustavson Simplex noise demystified]

Pour aller plus loin:

[A Lagae et al., STAR in Procedural Noise Functions. EG 2010]



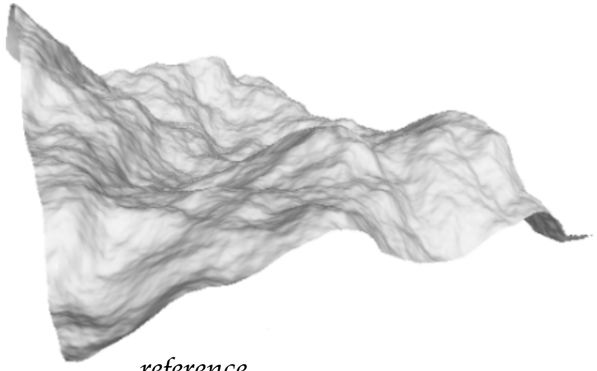
Bruit de Perlin: Terrain

On considère la surface S définit par: $S(u,v) = \begin{cases} x(u,v) = u \\ y(u,v) = v \\ z(u,v) = h\,P(s(u+o), s(v+o)) \end{cases}$

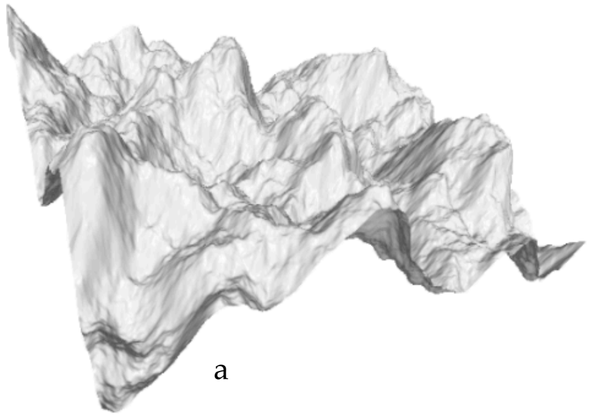
Le bruit de Perlin

$$P(u,v) = \sum_{k=0}^{N} \alpha^k \, f(2^k u, 2^k v)$$

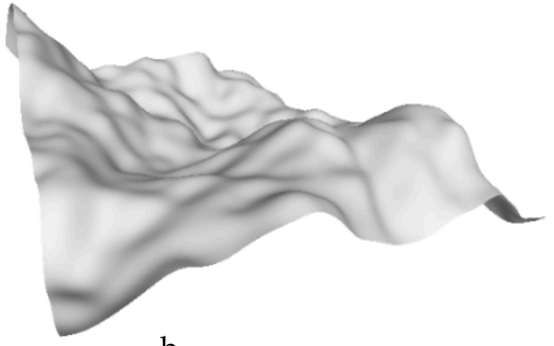
$N = 9$
$\alpha = 0.4$
$h = 0.3$
$s = 1$
$o = 0$



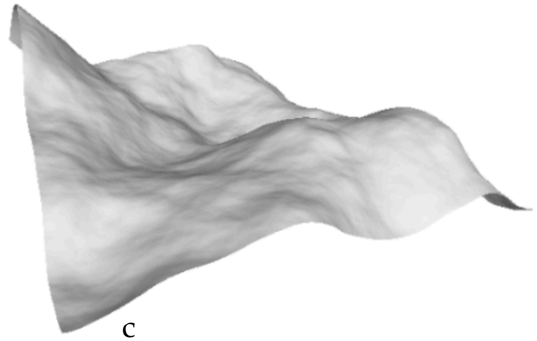
reference



a

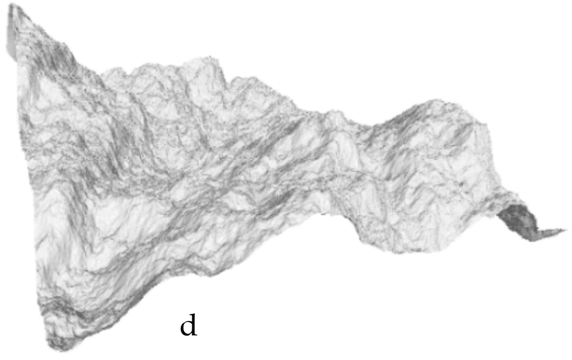


b

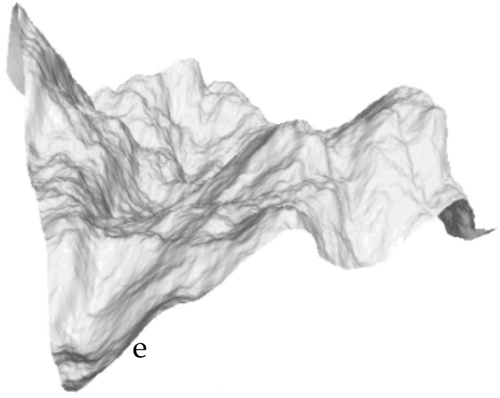


c

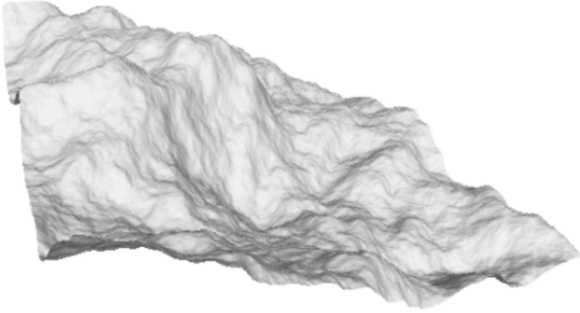
Q. Quels paramètres correspondent aux terrains (a,b,c,d,e,f) ?



d

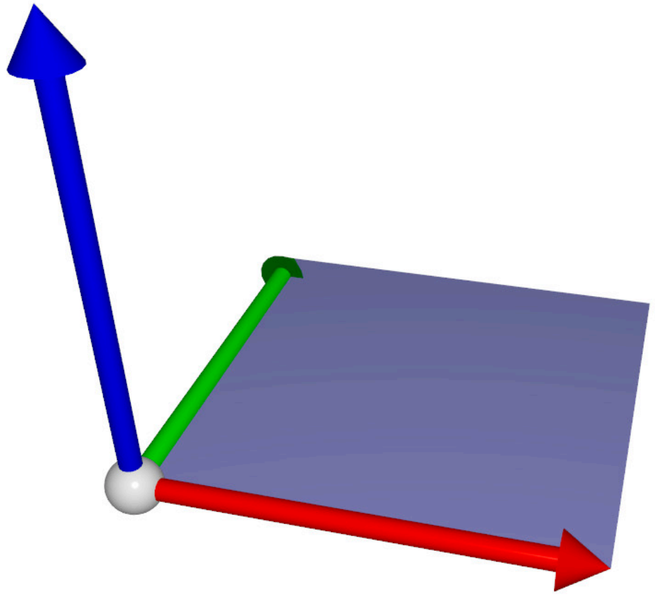


e



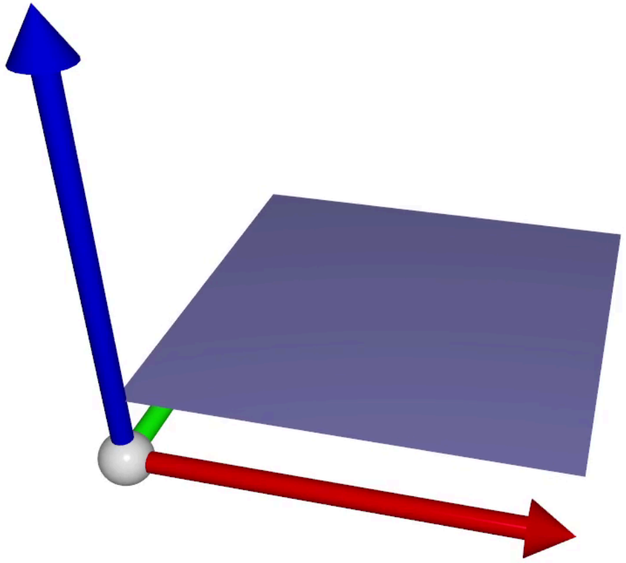
f

Perlin Noise - Animation



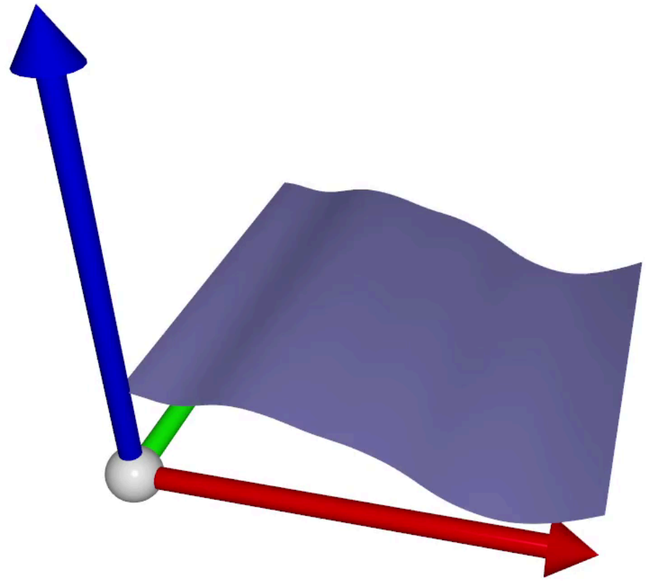
Fonction de la surface de référence

$(u, v) \in [0, 1]^2\,,\ S(u, v) = (u, v, 0)$



Bruit de perlin dépendant du temps

$S(u, v, t) = (u, v, P(t))$

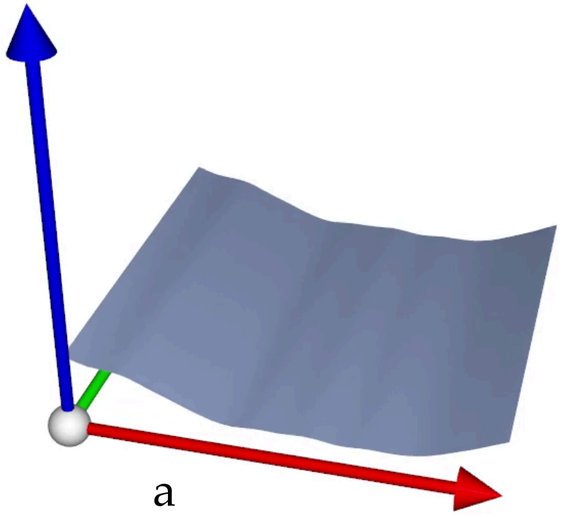


Dépendence à l'espace et au temps

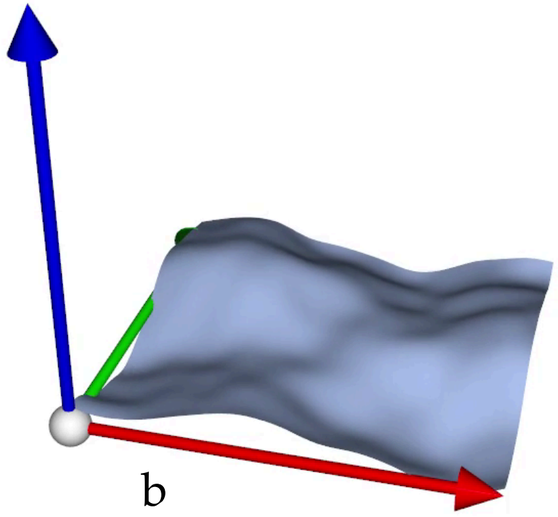
$S(u, v, t) = (u, v, P(u, t))$

Bruit de Perlin - Animation

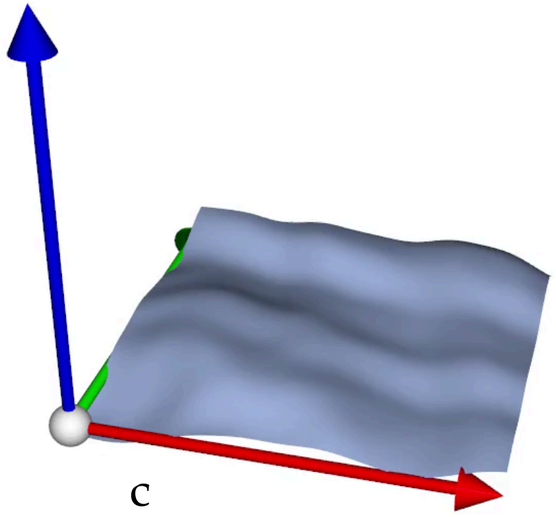
Surface de référence: $(u, v) \in [0, 1]^2$, $f(u, v) = (u, v, 0)$
Q. Comment générer les animations suivantes ? $S(u, v, t) = ...$



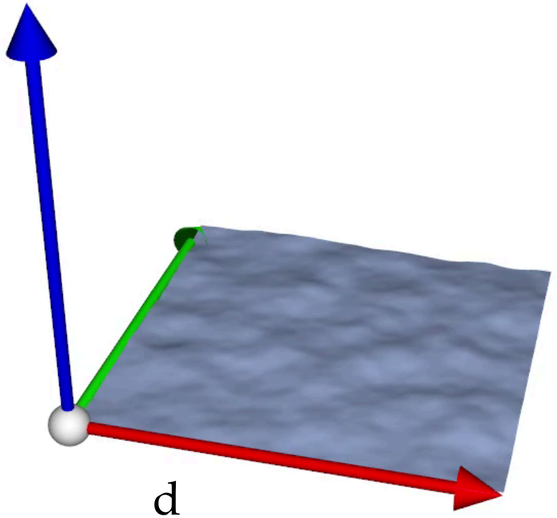
a



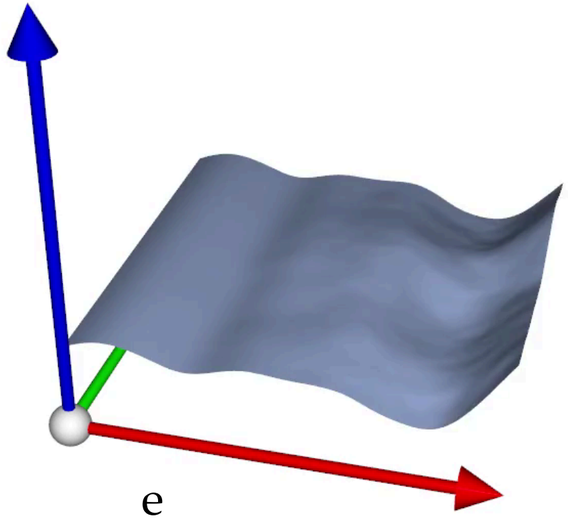
b



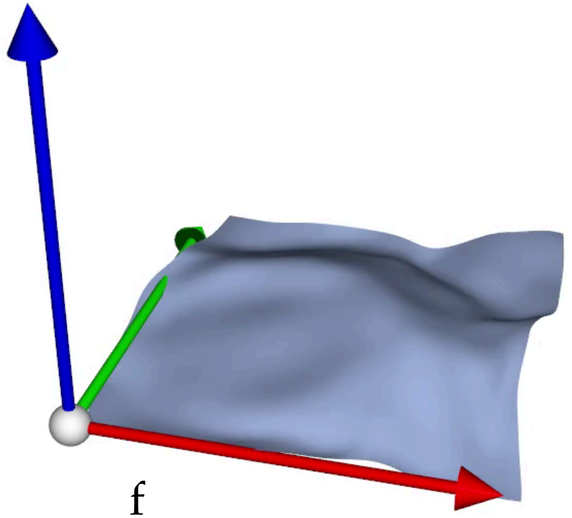
c



d



e



f