Représentations (réaliste) de vagues, Quelques applications à la visualisation de surface liquide.

Damien Rohmer, Cédric Rousset
ETI 3 Image, CPE Lyon

11 février 2007

Programme

Approches de la simulation de fluide

Approche Physique	Approche Non Physique

Approches de la simulation de fluide

Approche Physique	Approche Non Physique
• Navier Stockes. • Résolution d'EDP.	

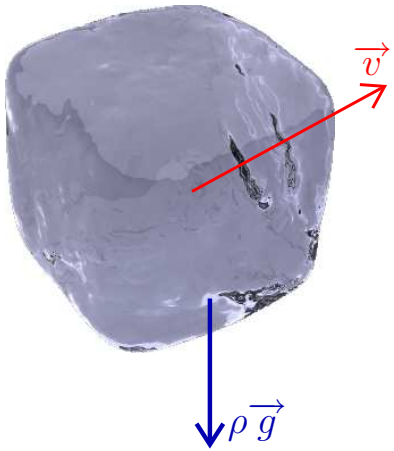
Approches de la simulation de fluide

Approche Physique	Approche Non Physique
• Navier Stockes. • Résolution d'EDP.	• Fonction Sinus. • Fonction de Bruit.

Approches de la simulation de fluide

Approche Physique

- Navier Stockes.
- Résolution d'EDP.



$$\begin{cases} \dfrac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\,\mathbf{v} = -\dfrac{\nabla p}{\rho} + \mu\nabla^2\mathbf{v} + \mathbf{f} \\ \nabla\cdot\mathbf{v} = 0 \end{cases}$$

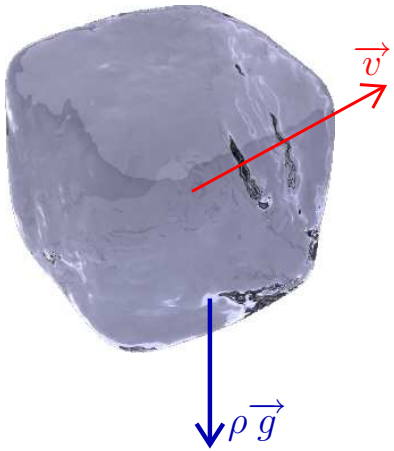
Approche Non Physique

- Fonction Sinus.
- Fonction de Bruit.

Approches de la simulation de fluide

Approche Physique

- Navier Stockes.
- Résolution d'EDP.



$$\begin{cases} \dfrac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\,\mathbf{v} = -\dfrac{\nabla p}{\rho} + \mu\nabla^2\mathbf{v} + \mathbf{f} \\ \nabla\cdot\mathbf{v} = 0 \end{cases}$$

Approche Non Physique

- Fonction Sinus.
- Fonction de Bruit.

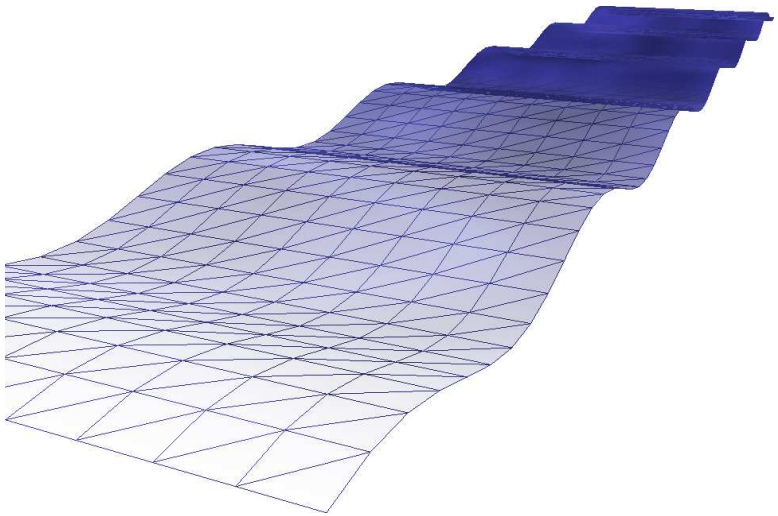
$$\sum f(\text{sin})(\mathbf{x}, t) + \text{noise}(\mathbf{x}, t)$$

Cas sinusoïdale.

- On considère une grille (x,y) fixe de $N_x \times N_y$ vertex.
- On calcul $z(k_x, k_y) = \sum_i A_i \sin(\mathbf{k}_i \cdot \mathbf{x} - \omega_i t)$
- On affiche (x, y, z)

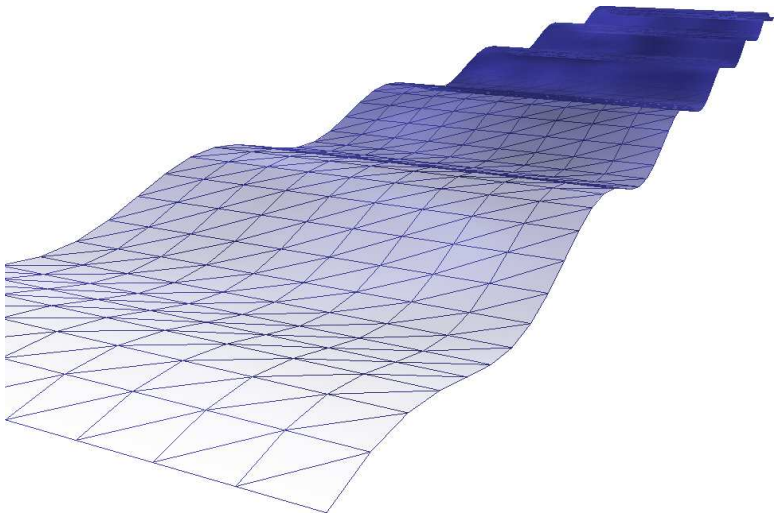
Cas sinusoïdale.

- On considère une grille (x,y) fixe de $N_x \times N_y$ vertex.
- On calcul $z(k_x, k_y) = \sum_i A_i \sin(\mathbf{k}_i \cdot \mathbf{x} - \omega_i t)$
- On affiche (x, y, z)

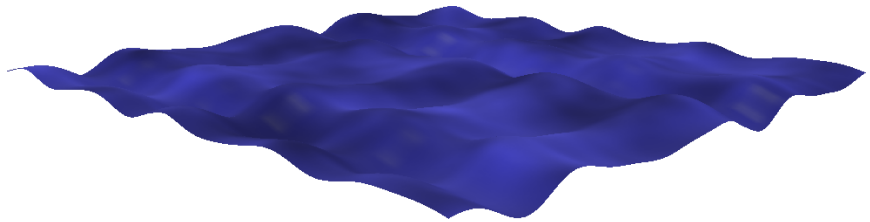


Cas sinusoïdale.

- On considère une grille (x,y) fixe de $N_x \times N_y$ vertex.
- On calcul $z(k_x, k_y) = \sum_i A_i \sin(\mathbf{k}_i \cdot \mathbf{x} - \omega_i t)$
- On affiche (x, y, z)



```
z = sin(3*x-t/7.0)*0.3;
```



```
z = sin(3*x-t/7.0)*0.1+sin(2*x-t/14)*0.2
   +sin(4*x-t/9.0)*0.05+sin(3*y+0.5*x-t/5.0)*0.03
   +sin(-4*y-0.5*x-t/6.4)*0.04+0.1*sin(2.4*y-t/15.0)
   +0.05*sin(2.6*y-t/14);
```

Oui Mais...

- Propagation **non réaliste**
- Très **périodique**
- Comment régler les A_i et ω_i en fonction de $\mathbf{k}_i$.

Oui Mais...

- Propagation **non réaliste**
- Très **périodique**
- Comment régler les A_i et ω_i en fonction de $\mathbf{k}_i$.

Que résolvons nous ?

- Prenons le cas de $\|\mathbf{k}_i\| = \mu\,\omega_i$.

Oui Mais...

- Propagation **non réaliste**
- Très **périodique**
- Comment régler les A_i et ω_i en fonction de $\mathbf{k}_i$.

Que résolvons nous ?

- Prenons le cas de $\|\mathbf{k}_i\| = \mu\,\omega_i$.
- On résout en réalité l'équation de **propagation des ondes** :

$$\triangle z - \mu^2 \frac{\partial z}{\partial t^2} = 0$$

Oui Mais...

- Propagation **non réaliste**
- Très **périodique**
- Comment régler les A_i et ω_i en fonction de $\mathbf{k}_i$.

Que résolvons nous ?

- Prenons le cas de $\|\mathbf{k}_i\| = \mu\,\omega_i$.
- On résout en réalité l'équation de **propagation des ondes** :

$$\triangle z - \mu^2 \frac{\partial z}{\partial t^2} = 0$$

- **Ce n'est pas la propagation des fluides ! !**

Ajoutons la physique : Ondes de surfaces

Repartons à la base + Hypothèses :

- Navier Stokes (surface libre) :

Ajoutons la physique : Ondes de surfaces

Repartons à la base + Hypothèses :

- Navier Stokes (surface libre) :
- **Hypothèse 1 :** Irrotationnel :

Ajoutons la physique : Ondes de surfaces

Repartons à la base + Hypothèses :

- Navier Stokes (surface libre) :
- **Hypothèse 1 :** Irrotationnel :
- Incompressibilité :

Ajoutons la physique : Ondes de surfaces

Repartons à la base + Hypothèses :

- Navier Stokes (surface libre) :
- **Hypothèse 1 :** Irrotationnel :
- Incompressibilité :
- **Hypothèse 2 :** La profondeur ne rentre pas en compte

Ajoutons la physique : Ondes de surfaces

Repartons à la base + Hypothèses :

- Navier Stokes (surface libre) :
- **Hypothèse 1 :** Irrotationnel :
- Incompressibilité :
- **Hypothèse 2 :** La profondeur ne rentre pas en compte

Mélangons le tout

$$\Rightarrow \begin{cases} \dfrac{\partial z}{\partial t} + v_x \dfrac{\partial z}{\partial x} + v_y \dfrac{\partial z}{\partial y} = v_z \\ \\ \dfrac{\partial \phi}{\partial t} + \dfrac{1}{2}\mathbf{v}^2 + g\,z = 0 \end{cases}$$

Puis retirons en un peu ...

$$\frac{\partial z}{\partial t} + v_x \frac{\partial z}{\partial x} + v_y \frac{\partial z}{\partial y} = v_z \quad \text{et} \quad \frac{\partial \phi}{\partial t} + \frac{1}{2}\mathbf{v}^2 + g\,z = 0$$

- Toujours trop complexe : Non linéaire !

Puis retirons en un peu ...

$$\frac{\partial z}{\partial t} + v_x \frac{\partial z}{\partial x} + v_y \frac{\partial z}{\partial y} = v_z \quad \text{et} \quad \frac{\partial \phi}{\partial t} + \frac{1}{2}\mathbf{v}^2 + g\,z = 0$$

- Toujours trop complexe : Non linéaire !
- Linéarisons :

$$\Rightarrow \frac{\partial z}{\partial t} = v_z \quad \text{et} \quad \frac{\partial \phi}{\partial t} + g\,z = 0$$

Attention : Hypothèse de faible amplitude

Equation résolue

La relation est maintenant **simple**.

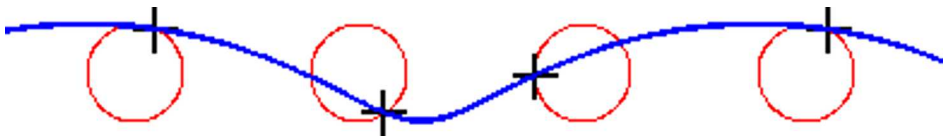
$$\frac{\partial z}{\partial t} = v_z \quad \text{et} \quad \frac{\partial \phi}{\partial t} + g\, z = 0$$

- On injecte $z = A \cos(\mathbf{k} \cdot \mathbf{x} - \omega\, t)$

Relation 1

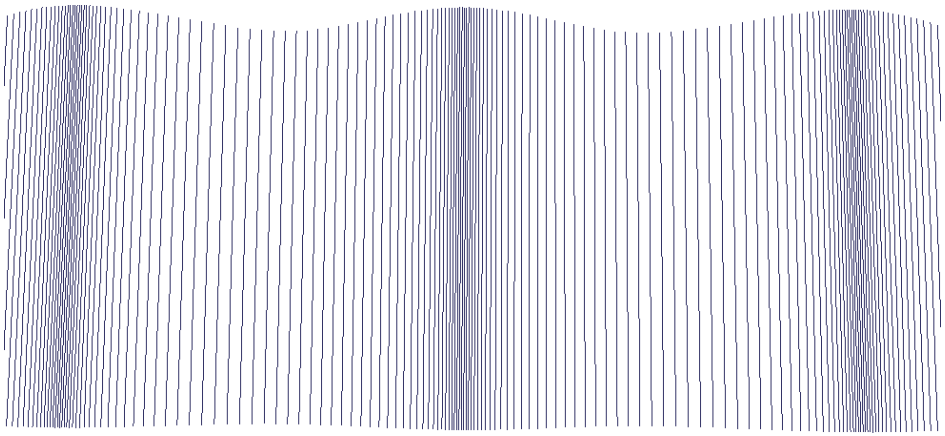
$$\begin{cases} \mathbf{X} = A \frac{\mathbf{k}}{\|\mathbf{k}\|} \sin(\mathbf{k} \cdot \mathbf{x} - \omega\, t) \\ z = A \cos(\mathbf{k} \cdot \mathbf{x} - \omega\, t) \end{cases}$$

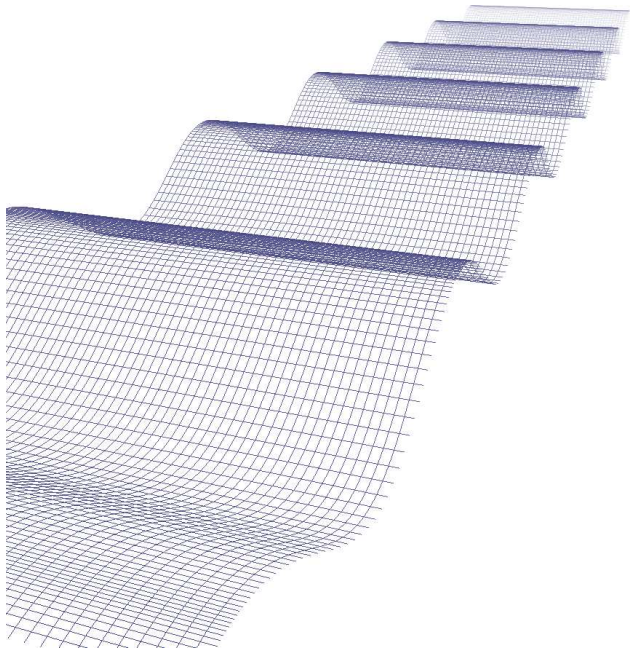
- z **n'est pas** la seule coordonnée à varier.
- La forme des vagues n'est donc **pas sinusoïdale**.
- Modèle de **trochoïdes** : Houle de Gestner (1802).
  [Fournier and Reeves, A simple models of ocean waves, 1986]
- **Trajectoire circulaire** des particules.



Relation 1 (Trochoïde)

- Zone de compression (Propagation d'une onde physique)





Relation 2

Relation de Dispersion :

- **Ondes de Gravitées** Vitesse d'onde $>$ Vitesse de groupe

$$\omega = \sqrt{g \left\| \mathbf{k} \right\|}$$

$\Rightarrow$ Les **petites crêtes** se déplacent **plus rapidement** que les trains d'ondes.

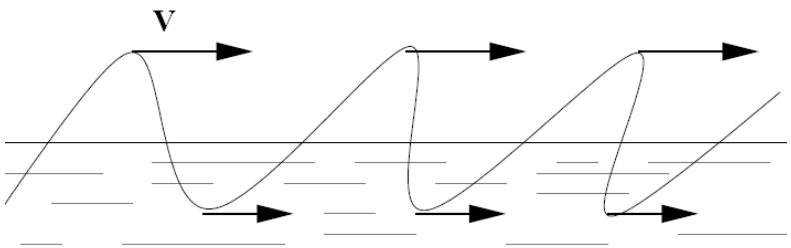
- On peux prendre en compte les **ondes de capillaritées**

$$\omega = \sqrt{g \left\| \mathbf{k} \right\| + \frac{t}{\rho} \left\| \mathbf{k} \right\|^3}$$

Prise en compte de la Profondeur

$$\omega = \sqrt{g\|\mathbf{k}\| \tanh(\|\mathbf{k}\| h)}$$

- Profondeur d'eau $= \frac{\lambda}{h}$.
- La vitesse de déplacement dépend de la hauteur de la vague.
  $\Rightarrow$ Attention à la brisure de vague.
- Permet de modéliser l'approche d'une plage.
- L'amplitude dépend de $\|\mathbf{k}\|$.



Implémentation OpenGL

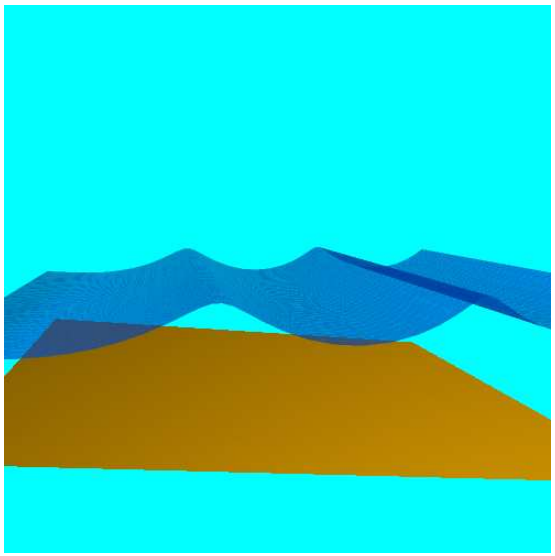
$$\begin{cases} \mathbf{X} = \sum A \frac{\mathbf{k}}{\|\mathbf{k}\|} \sin(\mathbf{k} \cdot \mathbf{x} - \omega t) \\ z = \sum A \cos(\mathbf{k} \cdot \mathbf{x} - \omega t) \end{cases}$$

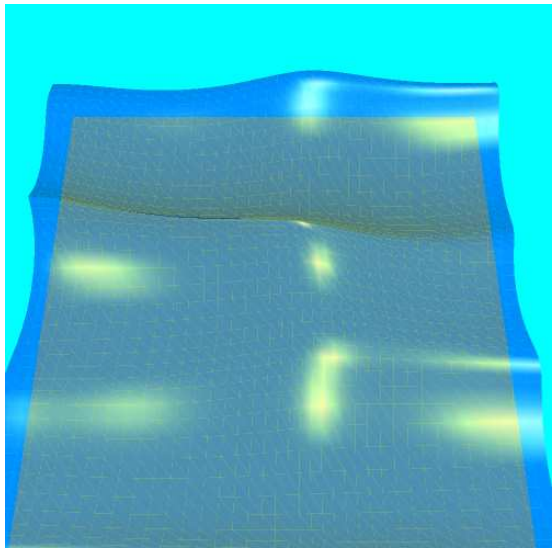
- On défini pour chaque vertex : Position+Normal+Distance au sol
- La valeur de $\mathbf{k}$ défini entièrement l'onde.

Implémentation OpenGL : Paramètres

Paramètres de :

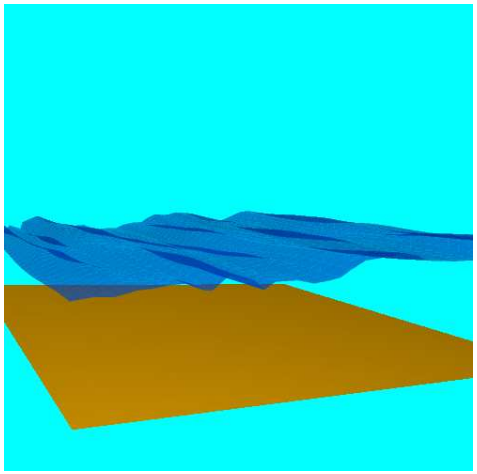
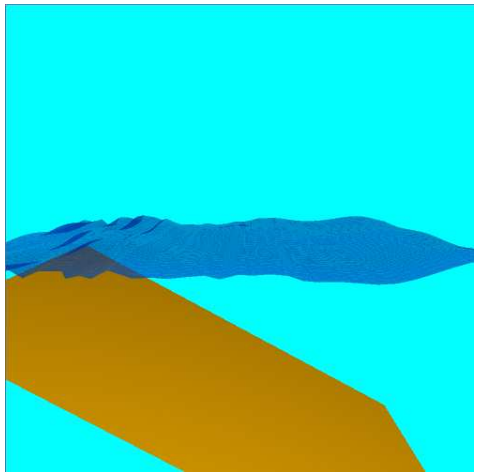
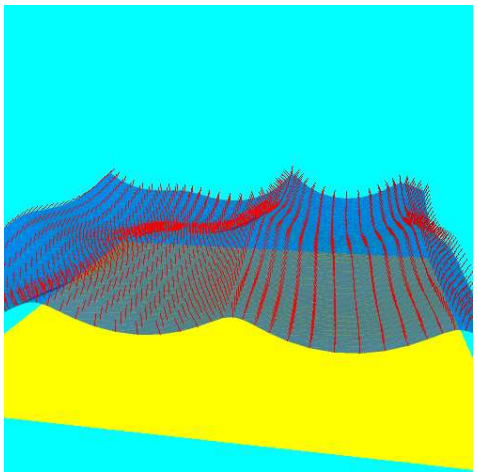
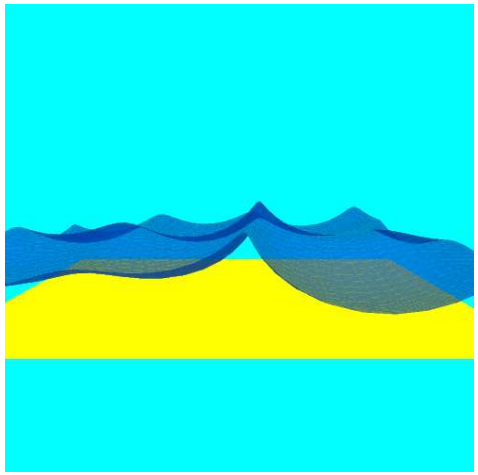
- Nombre d'ondes
- Amplitude
- Hauteur du sol





RUN

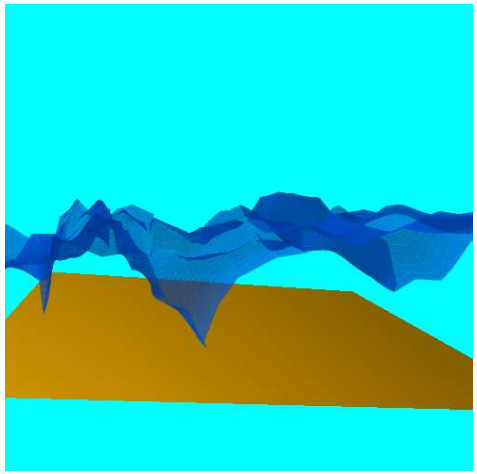
Implémentation OpenGL

Limitations de la méthode

Limitations dues aux **approximations** du modèle

- **Brisure de vague** (recouvrement)
- **Profondeur** trop faible ($< \lambda$)



Autres paramètres influants

- Vitesse du **vent**.
- Effet des **courants**

D'autres modèles physiques existent (Fournier et Reeves, Houle de Biesel, Bruit . . .)

Laissons la physique

Laissons la physique

- **Idée 1 :** La nature ressemble à du bruit.

Laissons la physique

- **Idée 1 :** La nature ressemble à du bruit.
- **Idée 2 :** Le nature est de type fractal.

Laissons la physique

- **Idée 1 :** La nature ressemble à du bruit.
- **Idée 2 :** Le nature est de type fractal.

$\Rightarrow$ **Bruit de Perlin.**
[Ken Perlin, Hypertexture, 1989]

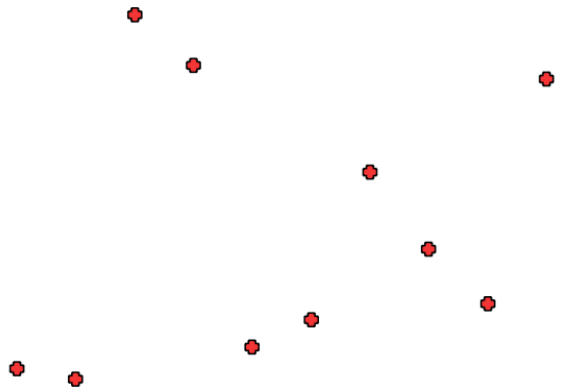
- Bruit (Pseudo Aléatoire) **continu** et de **nature fractale**.

Le bruit de Perlin : Comment ça marche

- On considère $f : n \mapsto f(n)$ avec $n \in \mathbb{Z}$
- On construit

$$\gamma : \begin{cases} \mathbb{R} & \longrightarrow [0,1] \\ x & \mapsto \gamma(x) \end{cases}$$

en **interpolant** les valeurs $f(n)$.

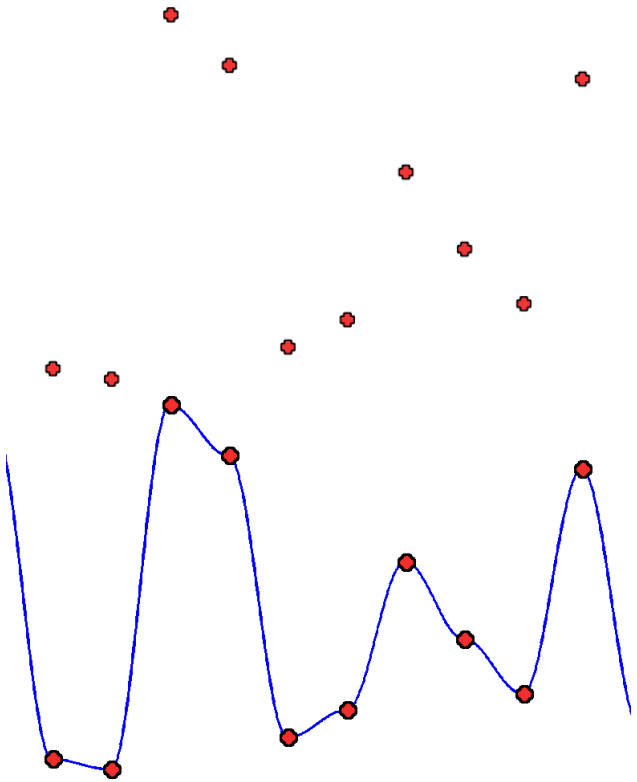


Le bruit de Perlin : Comment ça marche

- On considère $f : n \mapsto f(n)$ avec $n \in \mathbb{Z}$
- On construit

$$\gamma : \begin{cases} \mathbb{R} & \longrightarrow [0,1] \\ x & \mapsto \gamma(x) \end{cases}$$

en **interpolant** les valeurs $f(n)$.

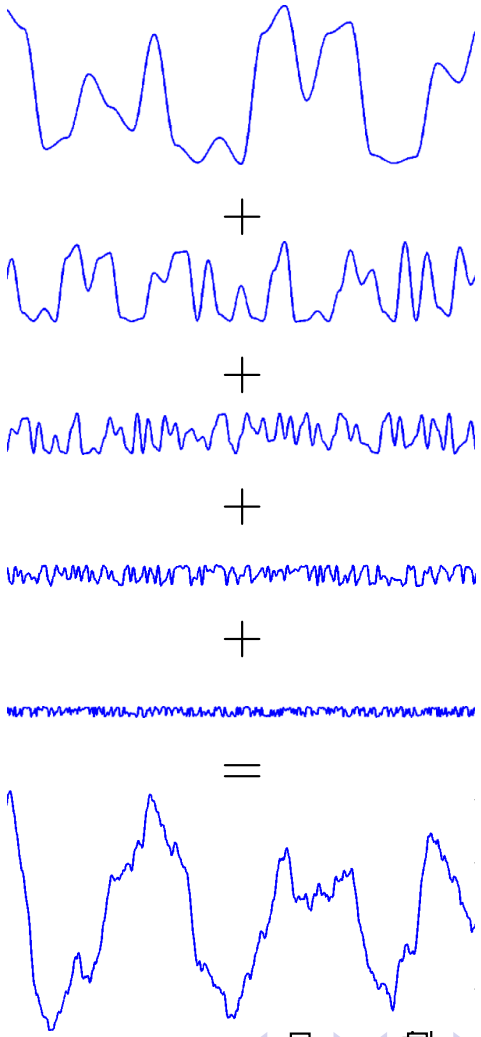


Le Bruit de Perlin : L'aspect Fractale

- Trop lisse ?

$$\Rightarrow \gamma_N(x) = \sum_{k=0}^{k=N} \frac{\gamma(a^k\, x)}{b^k}$$

  - N : octaves
  - a : frequence
  - 1/b : persitence



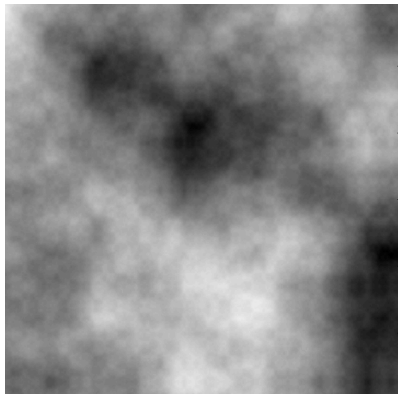
Le Bruit de Perlin : Le (pseudo) code (c)

```
float get_perlin(float (x,y,z),int octave,float persistence,float frequency)
{
 for(k=0:octave)
    (x,y,z) *= frequency^k;
    noise += interpolate_noise_3D(x,y,z)*persistence^k;
}


float interpolate_noise_1D(float x)
{
  x_0 =floor(x); x_1 =ceil (x); u = frac(x);
  return noise(x_0)*u+noise(x_1)*(1-u);
}


float noise_3D(int n1)
{
  //mess up
  n = ((n<<13)*5245465+rand_octave*23)*n*32412;
  //take abs
  n = n&0x7FFFFFFF;
  //between [0,1]
  return (n%432435136)/(432435136);
}
```

Le Bruit de Perlin : Applications I

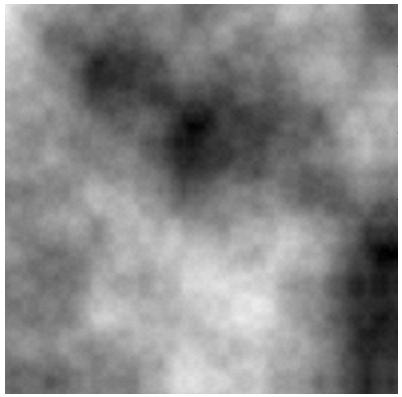
Le Bruit de Perlin : Applications I

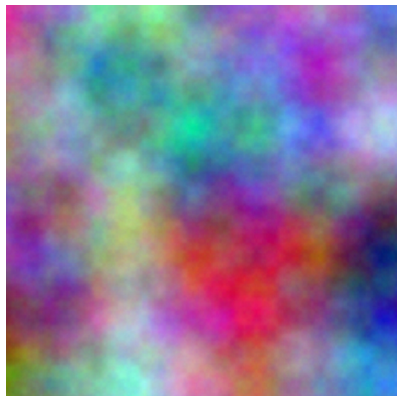
- Textures (Gimp)



Le Bruit de Perlin : Applications I

- Textures (Gimp)
- Textures en couleurs (Gimp toujours)



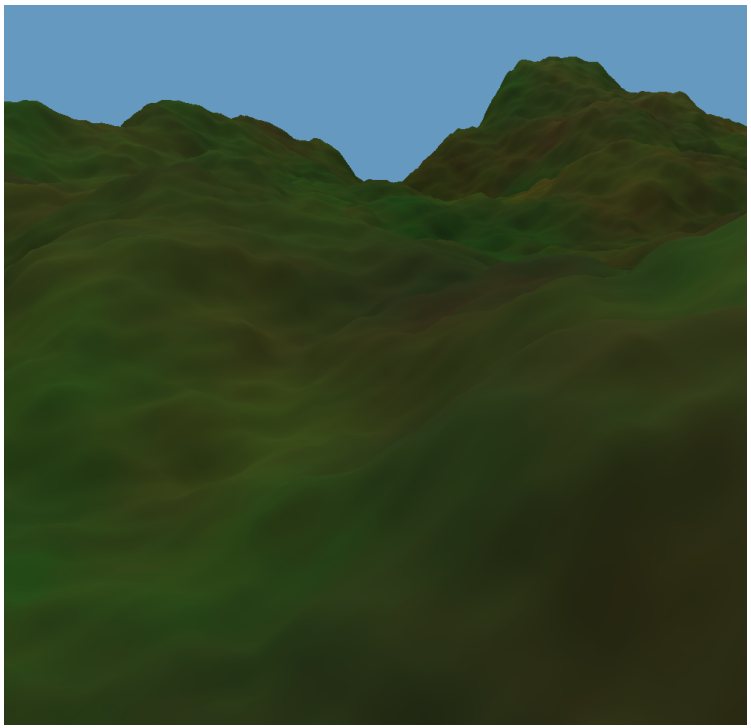


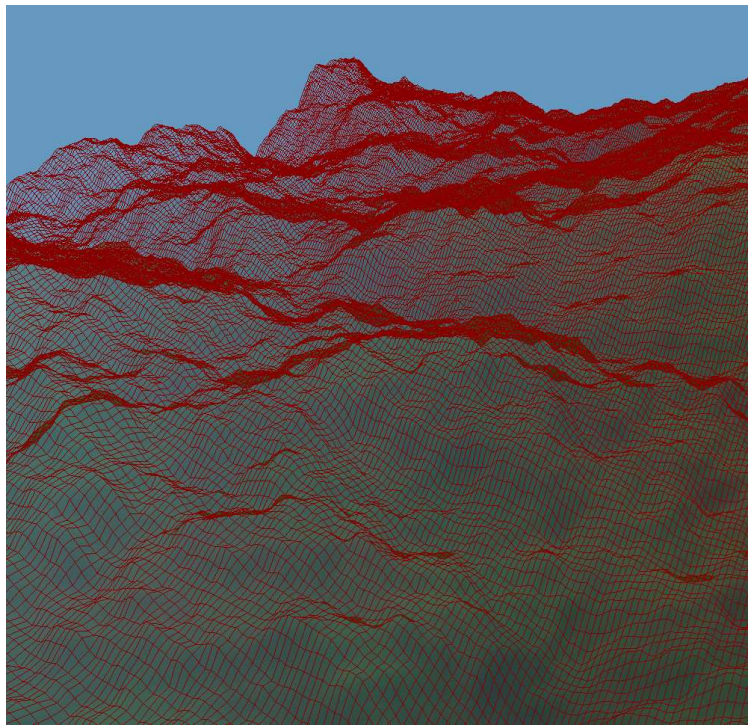
Le Bruit de Perlin : Applications II

- Jolies Montagnes (Terragen)

Le Bruit de Perlin : Applications II

- Jolies Montagnes (Terragen)





```
z = 0.3*noise.get_perlin(x,y,0,6,1/2.0,2.0);
```

Le Bruit de Perlin : Applications III (fin)

- Feu, cheveux, formes quelquonques, particules d'eau, ...

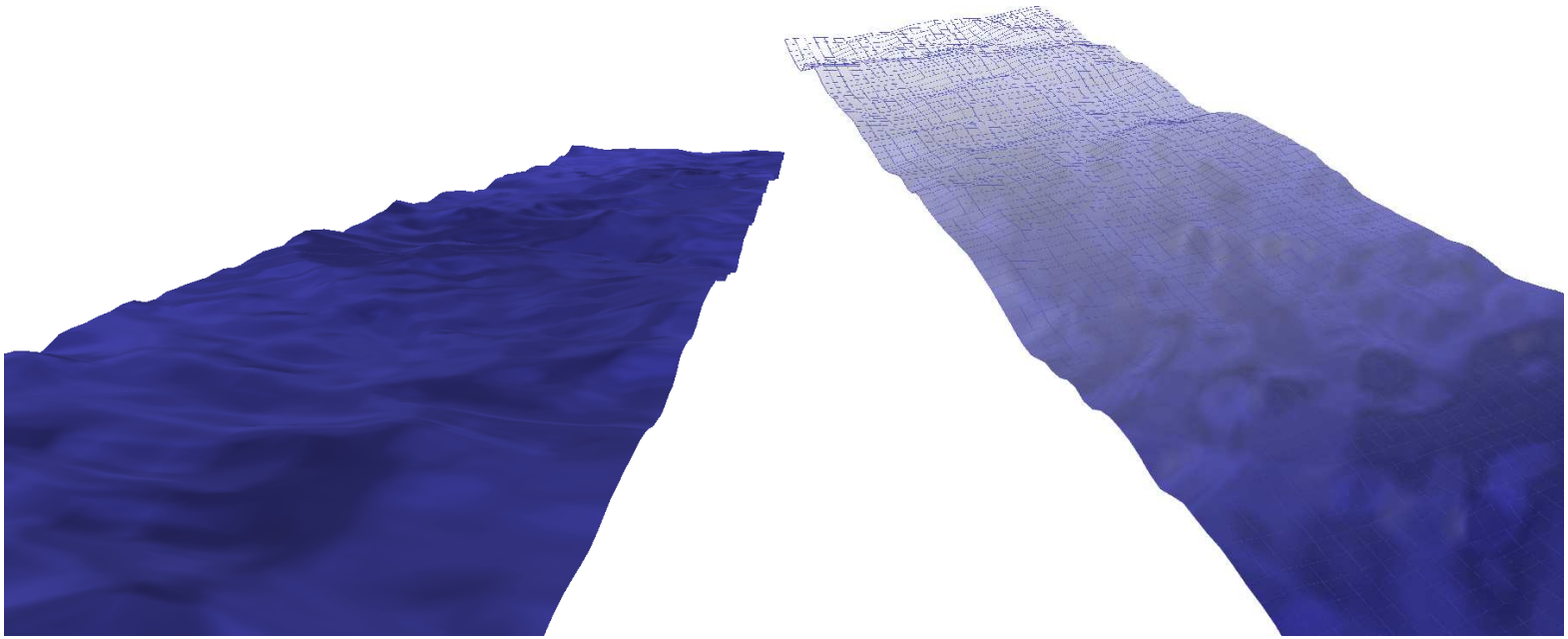
Le Bruit de Perlin : Applications III (fin)

- Feu, cheveux, formes quelquonques, particules d'eau, ...
- **Et de l'eau !**

Le Bruit de Perlin : Applications III (fin)

- Feu, cheveux, formes quelquonques, particules d'eau, ...
- **Et de l'eau !**
  Par exemple :

$$z = A\,\gamma(x, y, 0, \text{octave}) \quad \left(+\sin(\mathbf{k}\cdot\mathbf{x} - \omega\, t)\right)$$



```
z = 0.03*sin(3*x-t/7)+0.1*noise.get_perlin(2*x-t/20,2*y+0.05*cos(t/10),t/40-x/10+y/25,2.1/1.5,2);
```

Ridged Perlin

- **Problème :** Les crêtes sont lisses.

Ridged Perlin

- **Problème :** Les crêtes sont lisses.
- Augmentation de *octave* $\Rightarrow$ allure de montagnes.

Ridged Perlin

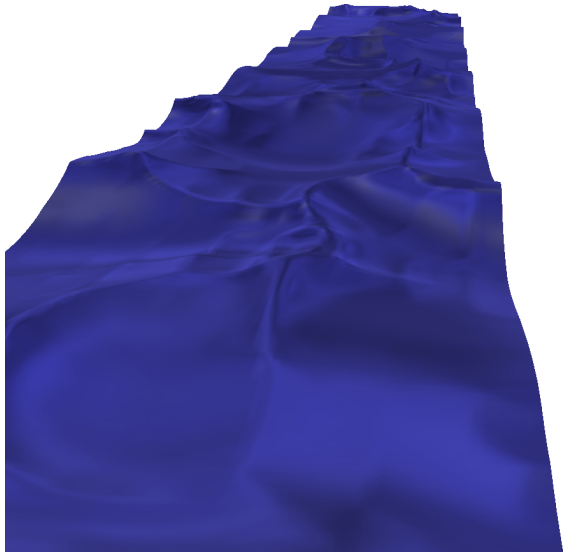
- **Problème :** Les crêtes sont lisses.
- Augmentation de *octave* $\Rightarrow$ allure de montagnes.
- **Solution :** Ridged (multifractal) Perlin μ.
  [code source Pov-Ray]

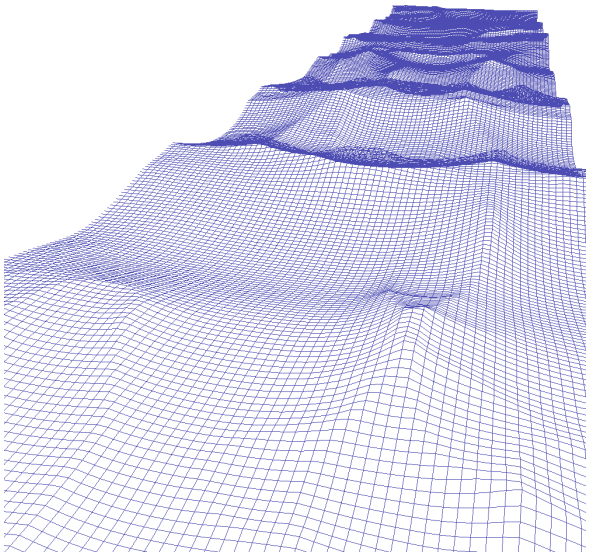
$$\begin{cases} \mu_p(\mathbf{x}) = \mu(\mathbf{x}) + \sum_{k=1}^{N} \omega_k(\mathbf{x}) a^{-k\,H}\, \mu(a^k\,\mathbf{x}) \\ \\ \begin{cases} \omega_k(\mathbf{x}) = \min(\max(\alpha\, \omega_{k-1}(\mathbf{x})\, \mu(a^{k-1}\mathbf{x}), 0), 1) \\ \omega_0(\mathbf{x}) = 1; \end{cases} \end{cases}$$

Avec

- N : nombre d'octave, a : multiplicateur en fréquence
- H : exposant (règle l'aspect lisse $\in [0, 1]$)
- α : Paramètre de coupure

Ridged Perlin : L'implémentation

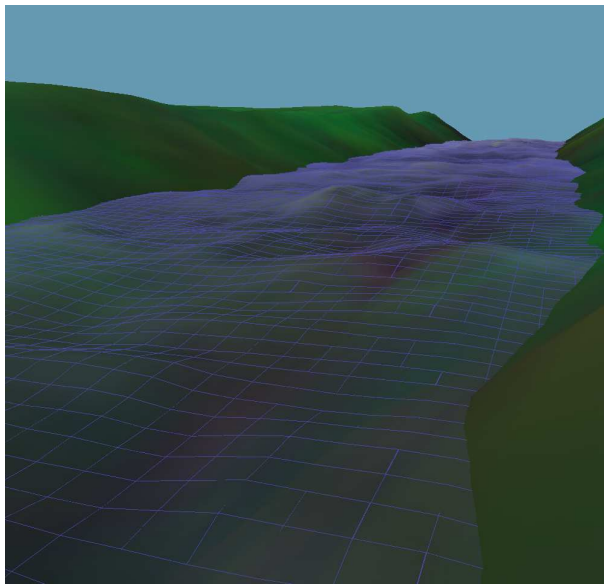


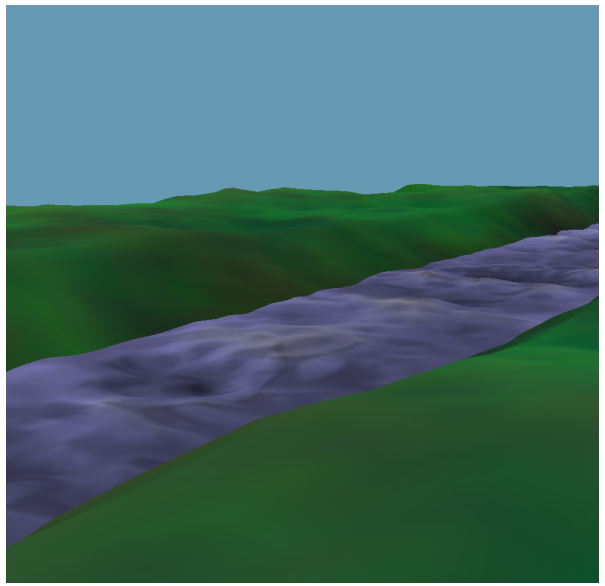


Au Final

- On peux combiner :
  [THON et al., A simple model for realistic running waters, 2000]

$$z = A \sum_i \sin(\mathbf{k} \cdot \mathbf{x} - \omega\, t) + B\, \gamma_N(\mathbf{x}) + C\, \mu(\mathbf{x})$$





Try it !

Introduction du Ray Tracing

- Limitation de la qualité du rendu
- Limitation de la grille (rendu de fractal...)

Introduction du Ray Tracing

- Limitation de la qualité du rendu
- Limitation de la grille (rendu de fractal...)

Idée : Utiliser le ray tracing
[Pov-Ray]
[merci à Christoph Hormann, Gilles Tran et Ben Weston]

Introduction du Ray Tracing

- Limitation de la qualité du rendu
- Limitation de la grille (rendu de fractal...)

Idée : Utiliser le ray tracing
[Pov-Ray]
[merci à Christoph Hormann, Gilles Tran et Ben Weston]

- Pas de limitations de résolution
- Effet physiques : (refraction, reflexion, (caustics), ...)

Introduction du Ray Tracing

- Limitation de la qualité du rendu
- Limitation de la grille (rendu de fractal...)

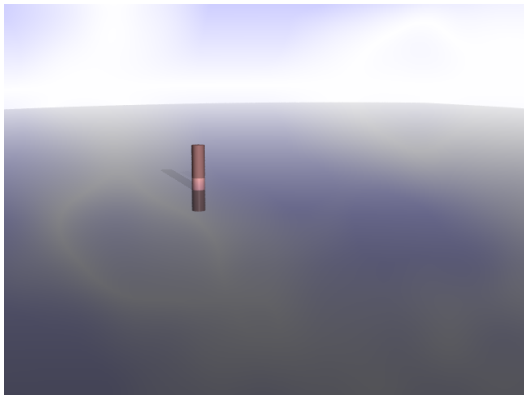
Idée : Utiliser le ray tracing
[Pov-Ray]
[merci à Christoph Hormann, Gilles Tran et Ben Weston]

- Pas de limitations de résolution
- Effet physiques : (refraction, reflexion, (caustics), . . .)

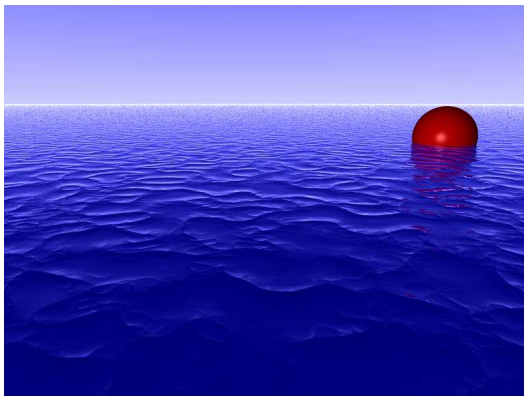
Mais c'est très long ! !

Point de départ

- On part d'une surface plate réflechissante (et d'un ciel).



Solution de facilité : Bump Mapping (avec ridged Perlin)



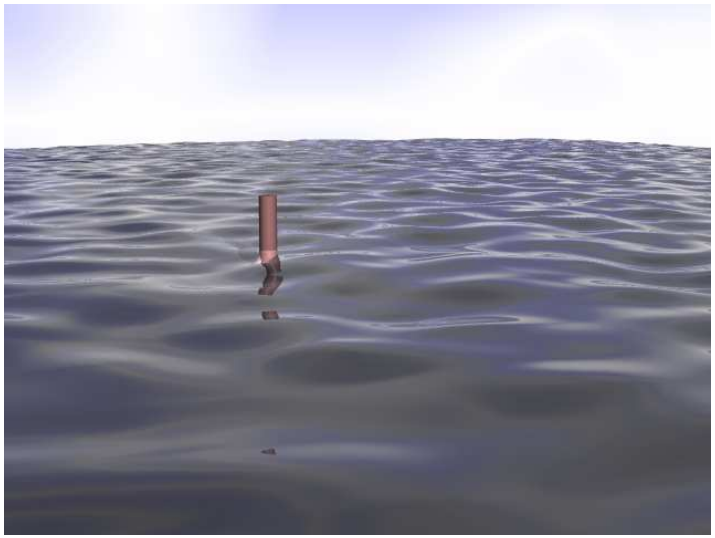
Autre facon : Fonction Implicite + Perlin

- On trace l'isosurface définie par

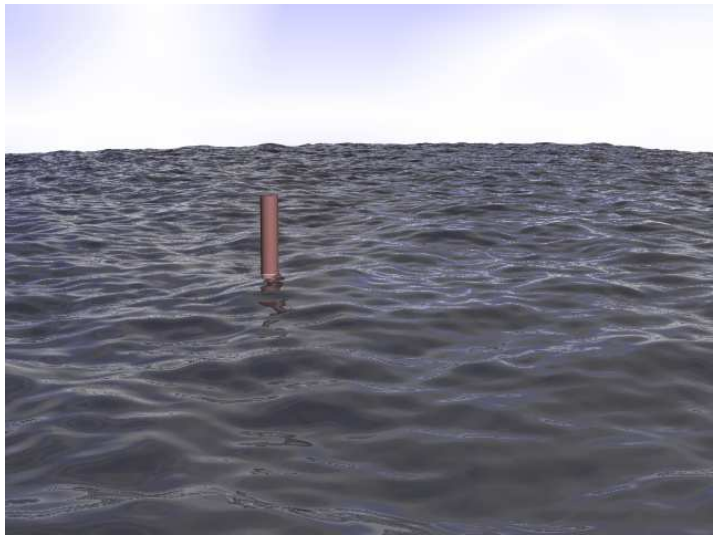
$$\left\{(x,y,z) \in \mathbb{R}^3 \middle| \phi(x,y,z) = 0\right\} \quad , \quad \phi = y + A\,\gamma_N + B\,\mu$$

- Perlin simple : un petit lac.

anim



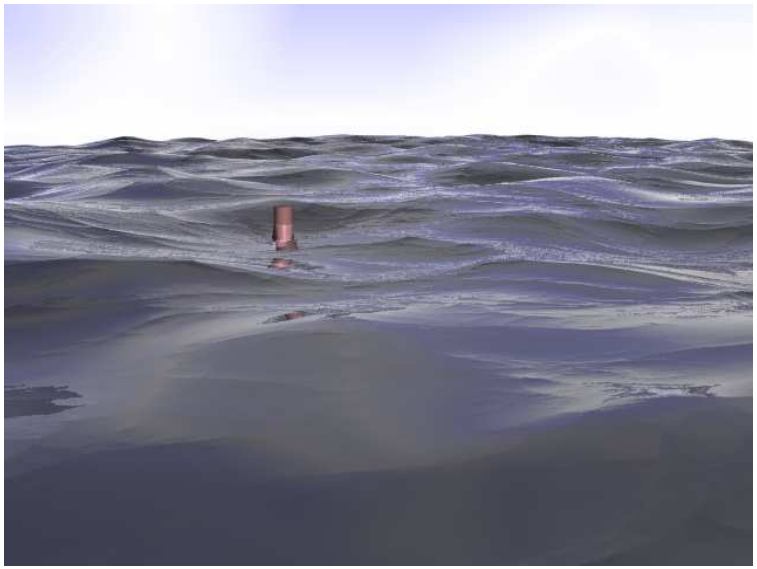
y-0.5+f_noise3d(2*x,0,2*z)/10



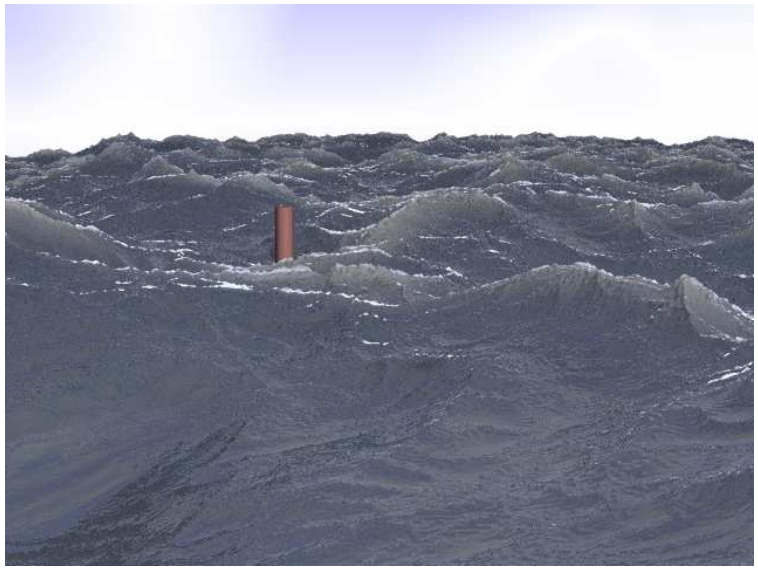
y-0.5+f_noise3d(x,0,z)/5+f_noise3d(2*x,0,2*z)/10
+f_noise3d(4*x,0,4*z)/20+f_noise3d(8*x,0,8*z)/40

Fonction Implicite : Ridged Perlin

- Et voila la mer



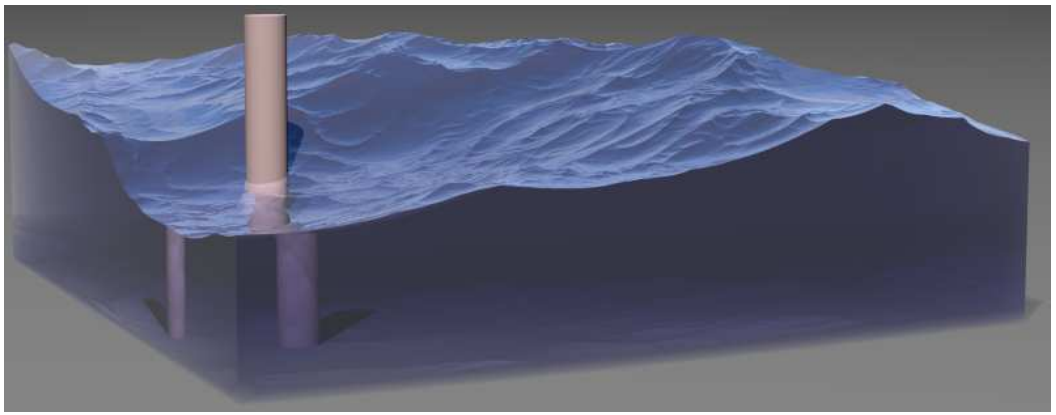
```
y-0.5
 -(f_ridged_mf(x/4,0,z/4,0.8,3,6,1,0.8,3)
 -0.8)/10-f_noise3d(x/1.3,0,z/1.3)/2.5
anim
```

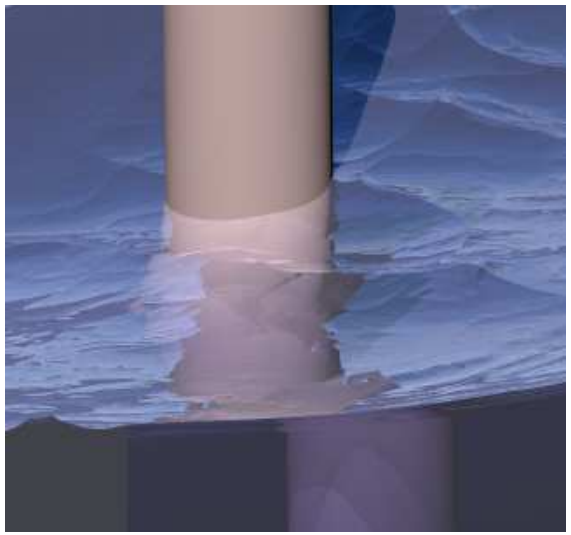


```
y-0.5
 -(f_ridged_mf(x/4,0,z/4,0.6,5,6,1,0.8,3)-1.5)/2
 -f_noise3d(x/1.3,0,z/1.3)/1.5
```

Fonction Implicite : Ridged Perlin

- Une coupe au travers de la surface :





Conclusion et possibilitées de poursuite

- Les rendus dépendent des applications

	Image Fixe	**Animation**
Rapidité	Bruit simple	Sinus+Bruit
Réalisme	Bruit fractal	Equations + Bruit fractal

- Les **astuces** permettent d'avoir des aspects **photoréalistes**.
- La **physique** permet d'avoir un **comportement animé** correct.
- Les **interactions** forcent l'utilisation des **EDP**.
- Reprenons Navier-Stockes à la base ...

Conclusion

... mais ceci sera pour la prochaine fois.

Merci de votre attention.