# (Realistic) Modelisation of Water Waves, some applications to to visualization of liquid surfaces.

Damien Rohmer, Cédric Rousset
ETI 3 Image Processing, CPE Lyon

February 11, 2007

# Contents

# Approaches of the fluid simulation

| Physically based | Non physically based |
|---|---|

# Approaches of the fluid simulation

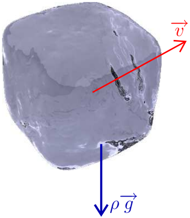
| Physically based | Non physically based |
|---|---|
| • Navier Stockes.<br>• EDP to solve. | |

# Approaches of the fluid simulation

| Physically based | Non physically based |
|---|---|
| • Navier Stockes.<br>• EDP to solve. | • Sinus functions.<br>• Noise functions. |

# Approaches of the fluid simulation

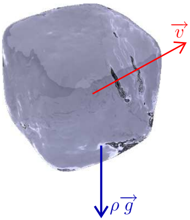
| Physically based | Non physically based |
| --- | --- |
| • Navier Stockes. | • Sinus functions. |
| • EDP to solve. | • Noise functions. |




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

# Approaches of the fluid simulation

| Physically based | Non physically based |
|---|---|
| • Navier Stockes.<br>• EDP to solve. | • Sinus functions.<br>• Noise functions. |




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

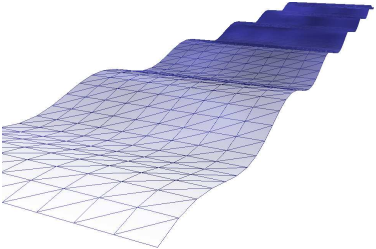
$$\sum f(\sin)(\mathbf{x}, t) + \text{noise}(\mathbf{x}, t)$$

# Sinusoïdal Case.

- Suppose a $N_x \times N_y$ mesh grid defined in $(x, y)$.
- Calculate $z(k_x, k_y) = \sum_i A_i \sin(\mathbf{k}_i \cdot \mathbf{x} - \omega_i t)$
- Draw $(x, y, z)$

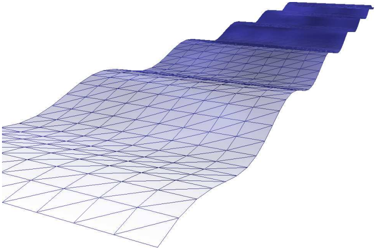
# Sinusoïdal Case.

- Suppose a $N_x \times N_y$ mesh grid defined in $(x, y)$.
- Calculate $z(k_x, k_y) = \sum_i A_i \sin(\mathbf{k}_i \cdot \mathbf{x} - \omega_i t)$
- Draw $(x, y, z)$

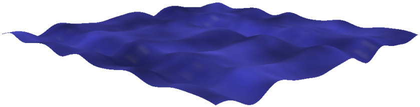


# Sinusoïdal Case.

- Suppose a $N_x \times N_y$ mesh grid defined in $(x, y)$.
- Calculate $z(k_x, k_y) = \sum_i A_i \sin(\mathbf{k}_i \cdot \mathbf{x} - \omega_i t)$
- Draw $(x, y, z)$



```
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);
```

# Yes But...

- **Unrealistic** propagation.
- Very **Periodical**
- How to set $A_i$ et $\omega_i$ which depend on $\mathbf{k}_i$.

# Yes But...

- **Unrealistic** propagation.
- Very **Periodical**
- How to set $A_i$ et $\omega_i$ which depend on $\mathbf{k}_i$.

## What are we solving?

- Take the linear case $\|\mathbf{k}_i\| = \mu\,\omega_i$.

# Yes But...

- **Unrealistic** propagation.
- Very **Periodical**
- How to set $A_i$ et $\omega_i$ which depend on $\mathbf{k}_i$.

## What are we solving?

- Take the linear case $\|\mathbf{k}_i\| = \mu\,\omega_i$.
- We are actually solving the **Wave Propagation** equation:

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

# Yes But...

- **Unrealistic** propagation.
- Very **Periodical**
- How to set $A_i$ et $\omega_i$ which depend on $\mathbf{k}_i$.

## What are we solving?

- Take the linear case $\|\mathbf{k}_i\| = \mu\,\omega_i$.
- We are actually solving the **Wave Propagation** equation:

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

- **This is not the water waves propagation!!**

# Lets add the Physic: Shallow water waves

Lets start to the beginning + Hypothesis

- Navier Stokes (free surface) :

# Lets add the Physic: Shallow water waves

Lets start to the beginning + Hypothesis

- Navier Stokes (free surface) :
- **Hypothesis 1:** Irrotationnal:

# Lets add the Physic: Shallow water waves

Lets start to the beginning + Hypothesis

- Navier Stokes (free surface) :
- **Hypothesis 1:** Irrotationnal:
- Uncompressible:

# Lets add the Physic: Shallow water waves

Lets start to the beginning + Hypothesis

- Navier Stokes (free surface) :
- **Hypothesis 1:** Irrotationnal:
- Uncompressible:
- **Hypothesis 2:** The depth does not count.

# Lets add the Physic: Shallow water waves

Lets start to the beginning + Hypothesis

- Navier Stokes (free surface) :
- **Hypothesis 1:** Irrotationnal:
- Uncompressible:
- **Hypothesis 2:** The depth does not count.

Mix and shake it

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

# Then throw some physics away

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

- Still too complex: Non linear!

# Then throw some physics away

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

- Still too complex: Non linear!
- Lets do some linearisation:

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

**Waring:** Hypothesis of small amplitudes.

## Solved Equation

The relation is now **straightforward**.

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

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

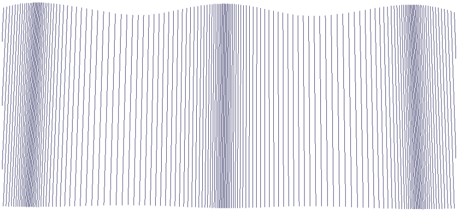
# Relation 1

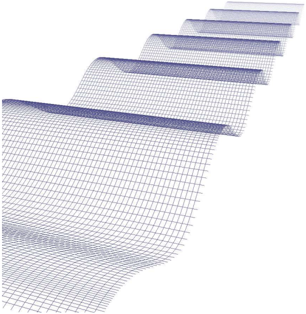
$$\begin{cases} \mathbf{X} = A \frac{\mathbf{k}}{\|\mathbf{k}\|} \sin(\mathbf{k} \cdot \mathbf{x} - \omega\, t) \\ z = A \cos(\mathbf{k} \cdot \mathbf{x} - \omega\, t) \end{cases}$$

- $z$ **is not** the single coordinate to vary.
- The water shape is **not a sinus**.
- **Trochoïdse** model: Gestner Swell (1802).
  [Fournier and Reeves, A simple models of ocean waves, 1986]
- Particles have **Circular Trajectory**.

# Relation 1 (Trochoïdes)

- Compression area (Physical wave propagation)





# Relation 2

Dispersion Relation:

- **Gravity Waves** Wave speed > Group speed

$$\omega = \sqrt{g\,\|\mathbf{k}\|}$$

$\Rightarrow$ **Small ridges** move **faster** than wave trains.

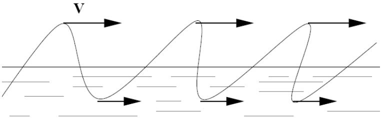
- We can take in account the **Capillarity Waves**

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

# Take the depth in account

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

- Water Depth $= \frac{\lambda}{h}$.
- The displacement speed depends on the height of the wave.
  $\Rightarrow$ Take care to the brake of the wave.
- Enable the modelisation of the beach.
- Amplitud depends on $\|\mathbf{k}\|$.



# OpenGL Implementation

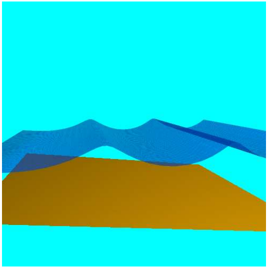
$$\begin{cases} \mathbf{X} = \sum A \frac{\mathbf{k}}{\|\mathbf{k}\|} \sin(\mathbf{k} \cdot \mathbf{x} - \omega\, t) \\ z = \sum A \cos(\mathbf{k} \cdot \mathbf{x} - \omega\, t) \end{cases}$$

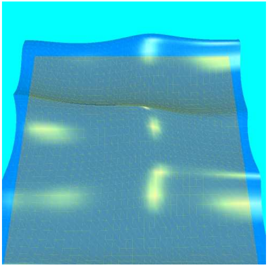
- Define for every vertex:
  Position+Normal+Depth
- The value of **k** completely define the wave.

# OpenGL Implementation: Parameters

Parameters for:

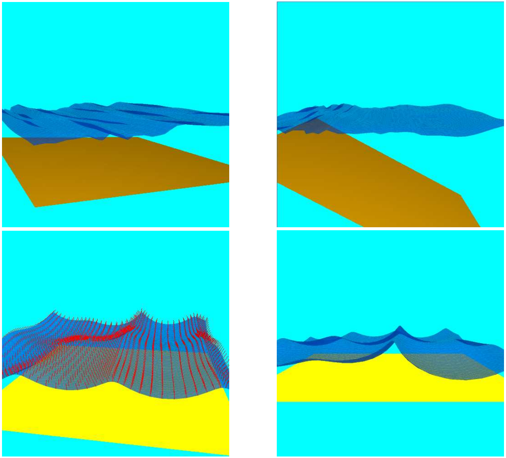
- Wave number
- Amplitude
- Depth





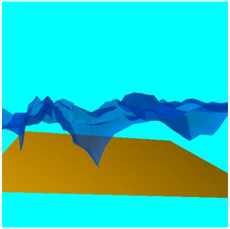
RUN

# OpenGL Implementation



# Limits

Limitations due to the **approximations** of the model



- **Wave breaking** (multi-valued function)
- **Depth** $(< \lambda)$

Other parameters to take in account

- **Wind** speed.
- **Current** effect.

Other physical model exist (Fournier et Reeves, Houle de Biesel, Bruit . . . )

# Let the physics far away

# Let the physics far away

- **Idea 1:** The nature looks like noise.

# Let the physics far away

- **Idea 1:** The nature looks like noise.
- **Idea 2:** The nature is like fractal noise.

# Let the physics far away

- **Idea 1:** The nature looks like noise.
- **Idea 2:** The nature is like fractal noise.

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

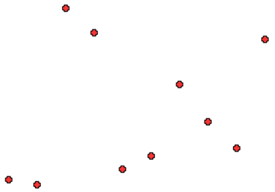
- **Continuous** and **Fractal** (Pseudo Random) Noise.

# Perlin Noise: How does it works

- Lets take $f : n \mapsto f(n)$ with $n \in \mathbb{Z}$
- We build

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

by **interpolating** the values of $f(n)$.

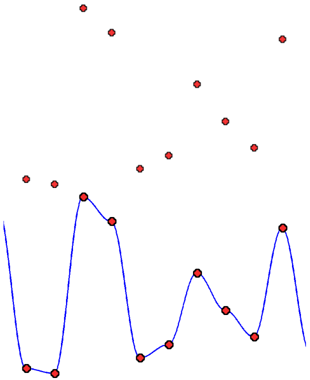


# Perlin Noise: How does it works

- Lets take $f : n \mapsto f(n)$ with $n \in \mathbb{Z}$
- We build

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

by **interpolating** the values of $f(n)$.

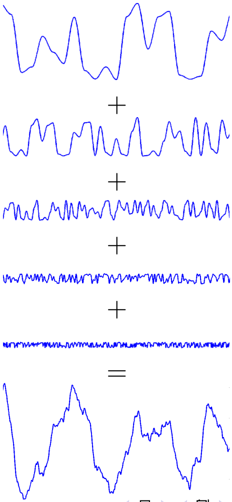


# Perlin Noise: Fractal Behavior

- Troo smooth?

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

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



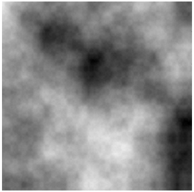
# Perlin Noise: (Pseudo) code (c)

```
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);
}
```

# Perlin Noise: Applications I

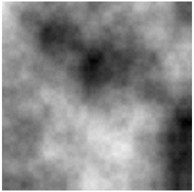
# Perlin Noise: Applications I

- Textures (Gimp)

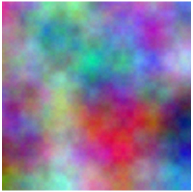


# Perlin Noise: Applications I

- Textures (Gimp)



- Colored Textures (Gimp again)

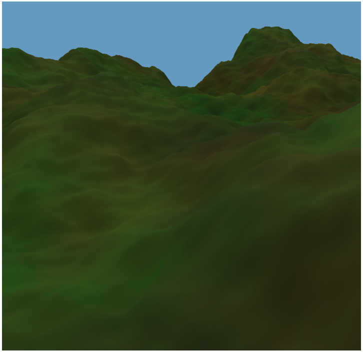


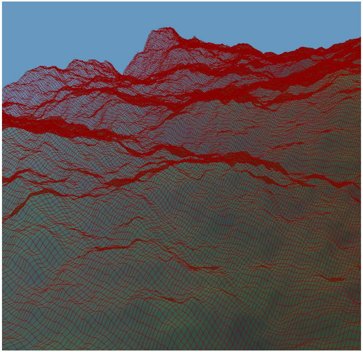
# Perlin Noise: Applications II

- Nice Mountains (Terragen)

# Perlin Noise: Applications II

- Nice Mountains (Terragen)





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

# Perlin Noise: Applications III (end)

- Fire, Hairs, any shapes, water particles, . . .

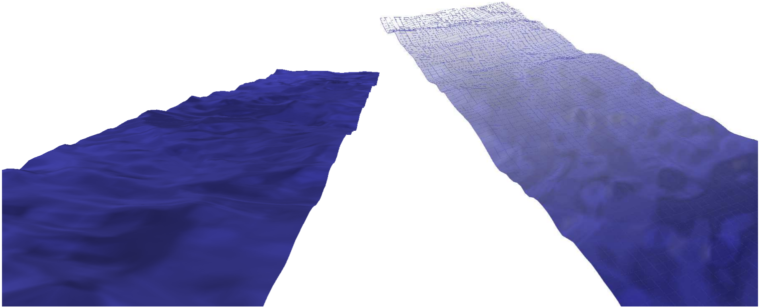
# Perlin Noise: Applications III (end)

- Fire, Hairs, any shapes, water particles, . . .
- **And the WATER!**

# Perlin Noise: Applications III (end)

- Fire, Hairs, any shapes, water particles, ...
- **And the WATER!**
  for instance:

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



```
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

- **Problem:** Ridged are too smooth.

# Ridged Perlin

- **Problem:** Ridged are too smooth.
- Increase of *octave* number $\Rightarrow$ Mountain shapes.

# Ridged Perlin

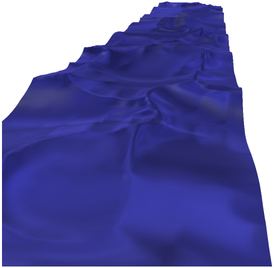
- **Problem:** Ridged are too smooth.
- Increase of *octave* number $\Rightarrow$ Mountain shapes.
- **Solution:** Ridged (multifractal) Perlin $\mu$.
  [source code Pov-Ray]

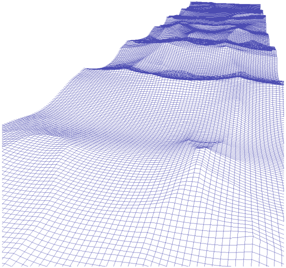
$$\begin{cases} \mu_p(\mathbf{x}) = \mu(\mathbf{x}) + \sum_{k=1}^{N} \omega_k(\mathbf{x}) a^{-k\,H}\, \mu(a^k\,\mathbf{x}) \\ \begin{cases} \omega_k(\mathbf{x}) = \min(\max(\alpha\,\omega_{k-1}(\mathbf{x})\,\mu(a^{k-1}\mathbf{x}),0),1) \\ \omega_0(\mathbf{x}) = 1; \end{cases} \end{cases}$$

With

- N: Octave number, a: frequency multiplier
- H: exponant (smoothing aspect $\in [0,1]$)
- $\alpha$: Cut parameter (multifractal behavior)

# Ridged Perlin: Implementation

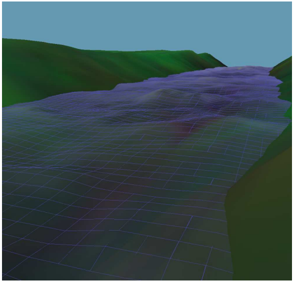


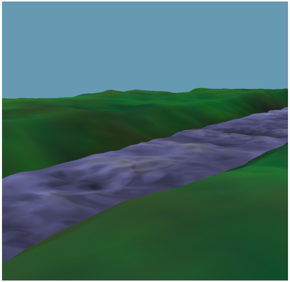


# To the End

- We can merge together:
  [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 to Ray Tracing

- Quality limitations
- Grid limitations (how to draw fractals?)

# Introduction to Ray Tracing

- Quality limitations
- Grid limitations (how to draw fractals?)

**Idea:** Ray Tracing
[Pov-Ray]
[thanks to Christoph Hormann, Gilles Tran and Ben Weston]

# Introduction to Ray Tracing

- Quality limitations
- Grid limitations (how to draw fractals?)

**Idea:** Ray Tracing
[Pov-Ray]
[thanks to Christoph Hormann, Gilles Tran and Ben Weston]

- No resolution limitations
- Physical effects: (refraction, reflection, (caustics), . . . )

# Introduction to Ray Tracing

- Quality limitations
- Grid limitations (how to draw fractals?)

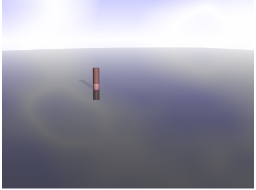
**Idea:** Ray Tracing
[Pov-Ray]
[thanks to Christoph Hormann, Gilles Tran and Ben Weston]

- No resolution limitations
- Physical effects: (refraction, reflection, (caustics), ...)

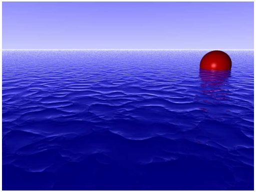
**But it's so Slow!!**

## Starting point

- Start with a reflective plane surface (and a sky).



Easy solution (but false): Bump Mapping (with ridged Perlin)



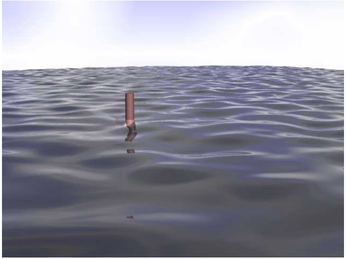
# Other solution: Implicit Function + Perlin

- Render the isosurface defined by

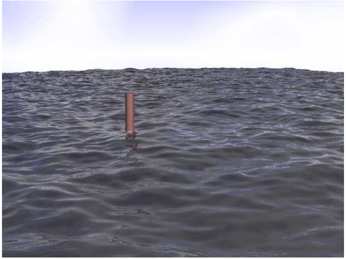
$$\left\{(x,y,z)\in\mathbb{R}^3 \middle| \phi(x,y,z)=0\right\} \quad , \quad \phi = y + A\,\gamma_N + B\,\mu$$

- Simple Perlin: un small lake.

anim



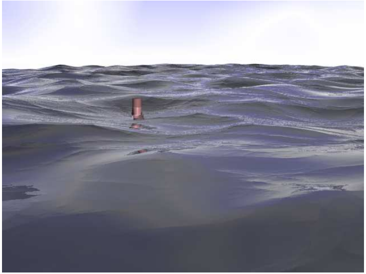
`y-0.5+f_noise3d(2*x,0,2*z)/10`



`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`

# Implicit Function: Ridged Perlin

- And the Sea is coming.

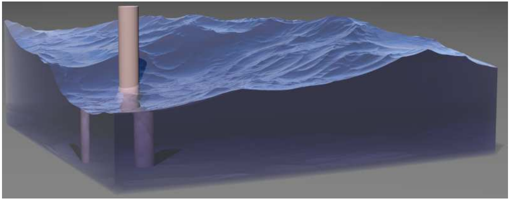


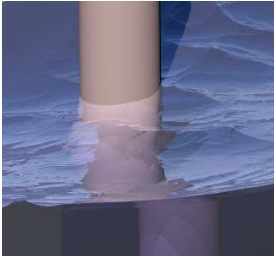
```
y-0.5
 -(f_ridged_mf(x/4,0,z/4,0.8,3,6,1,0.8,3)
 -0.8)/10-f_noise3d(x/1.3,0,z/1.3)/2.5
```
anim

```
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
```

# Implicit Function: Ridged Perlin

- A cut through the surface:





# Conclusion and future works

- Visualization depends on the applications:

| | **Fixed Picture** | **Animation** |
|---|---|---|
| **Speed** | Simple Noise | Sinus+Noise |
| **Realism** | Fractal Noise | Equations + Fractal Noise |

- **Tricks** enable to get **photorealistic** aspects.
- **Physic** enable to get a correct **dynamic**.
- **Interactions** need to solve **EDP**.
- We need to come back to the full Navier-Stockes equation ...

# Conclusion

... but it will be for the next time.

**Thank you for your attention.**