# Jeux video

Phenomènes naturels :

* Terrain
* Vegetation
* Fuides
* Atmosphère

Citées

* Routes

Chalenge :

* Scale (passage à l’echelle) peut passer de 1 cm à 1000 km (problème de data)
* Realisme / plausibilité (passage de bio/physique etc à l’informatique)
* Contrôle (faire ce qu’on veux ou l’on veux)

## Chap 1 : Modélisation de terrain

Terrain = macro et meso scale (+ 1mètre)

Micro scale = galets etc ..

Edition : Contrôle ++, Vitesse --, realisme ?

Simulation : Contrle --, realisme ++, lent

Generation : Crontrole +, rapide, realisme bof,

## Modèles :

|  |  |  |
| --- | --- | --- |
|  | Surfacique | Volumique |
| NoData | Z = f(x,y) | F(x,y,z) = 0  Donc avec la surfacique z-f(y,y) = 0 |
| Data | Une carte de data d’élévation  O(n²) | Les voxel O(n^3) |

*Int index(i,j){*

*return i\*my+j ;*

*}*

*P (i,j){*

*X = ax+bx-ax/nx-1 ;*

*Y = ay+by-ay/ny-1\*j ;*

*Z = z[index(i,j)] ;*

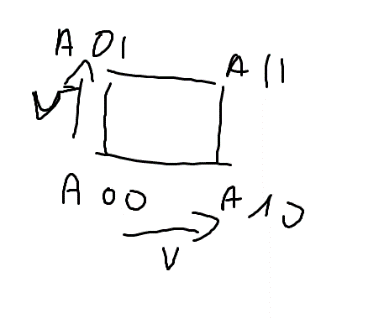
*}*

### Triangulaire C0:

Utiliser une interpoation triangulaire est utile lorsque l’on veut afficher qqch au dessus du terrain, on est forcement a la hauteur de l’affichage pour tout es poins données en paramètres.

(voir photo tel)

### Bilineraire C1 :

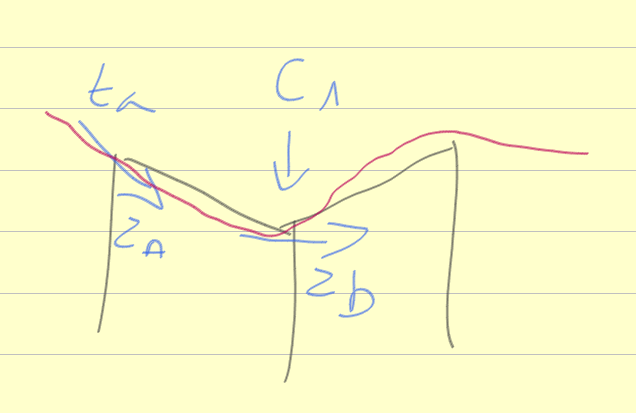


Lin(u,a,b) = (1-u)a+nb

Bilineaire(u,v,a\_)

(1-u) (1-v) \*a00 + (i-u) v \* a01 + u(1-v)\*a10 + uv\*a11

### Bicubique C1 partout :



Pour que ça soit joli sans utiliser le bicubique il faut faire la normale aux sommets au lieu de la normale par faces.

*Bool inside( x,y,z){*

*zt = p(x,y ) ;*

*return z<zt ;*

*}*

*Todo*

*class v2*

*class v3*

*class b3*

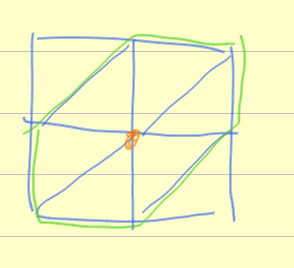
*class b2*

*Chass HF ( ::P(x,y) ::p(i,j) ::inside(x,y,z)*

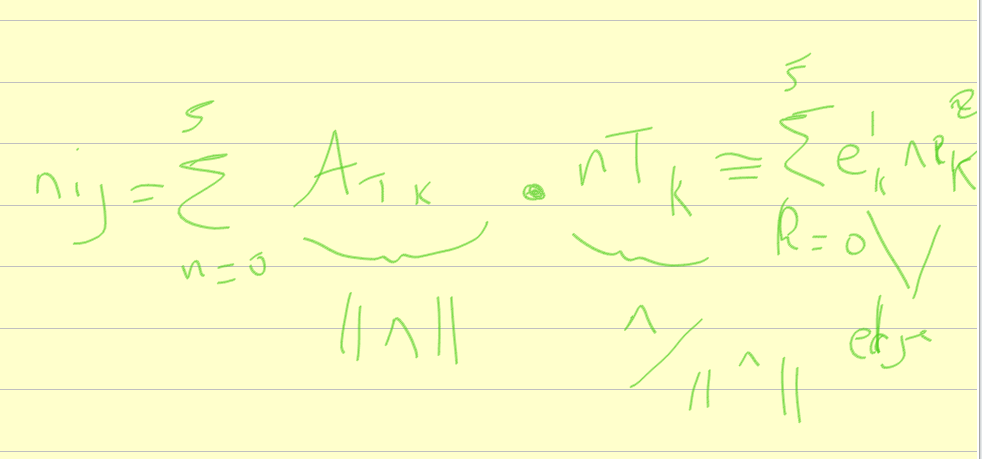
*::load (image ,v2 a,v2 b, float za, float zb) (float pour l’etirement)*

*:: saveObj(QS hing)*

*Normale au sommet :*



*Nij = somme k=0 à 5 de nTk et donc avec l’aire*



Il faut faire attention avec aussi au bord haut , droite , gauche , bas etc , et les point des cotées.

### Layers fields :

* Bonnes en 2001
* Modéliser différentes couche en matériaux

Avec le HF précédemment definit



Hight field deviens un scalar field

*Class SF (s[], a, b, nx, ny)*

*Class HF : public SF (rien de plus sauf fonctions)*

*Class LayerField{*

*SF br ;*

*SF sable ;*

*SF Water ;*

*}*

*Class NoiseF{*

*double a[*

*double b[*

*double H(x,y)*

*}*

*void HF(NF,n,b)*

### 3) Modèle par Fo

Perlin / huffert/ musgrave

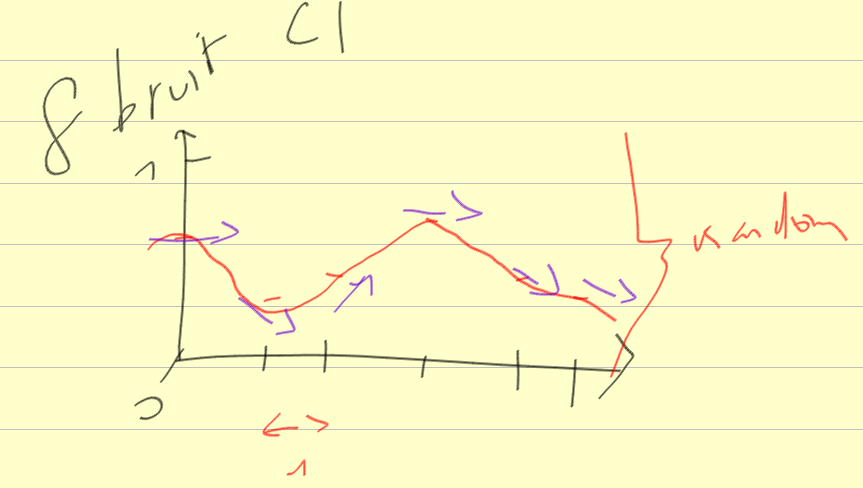
Interpol

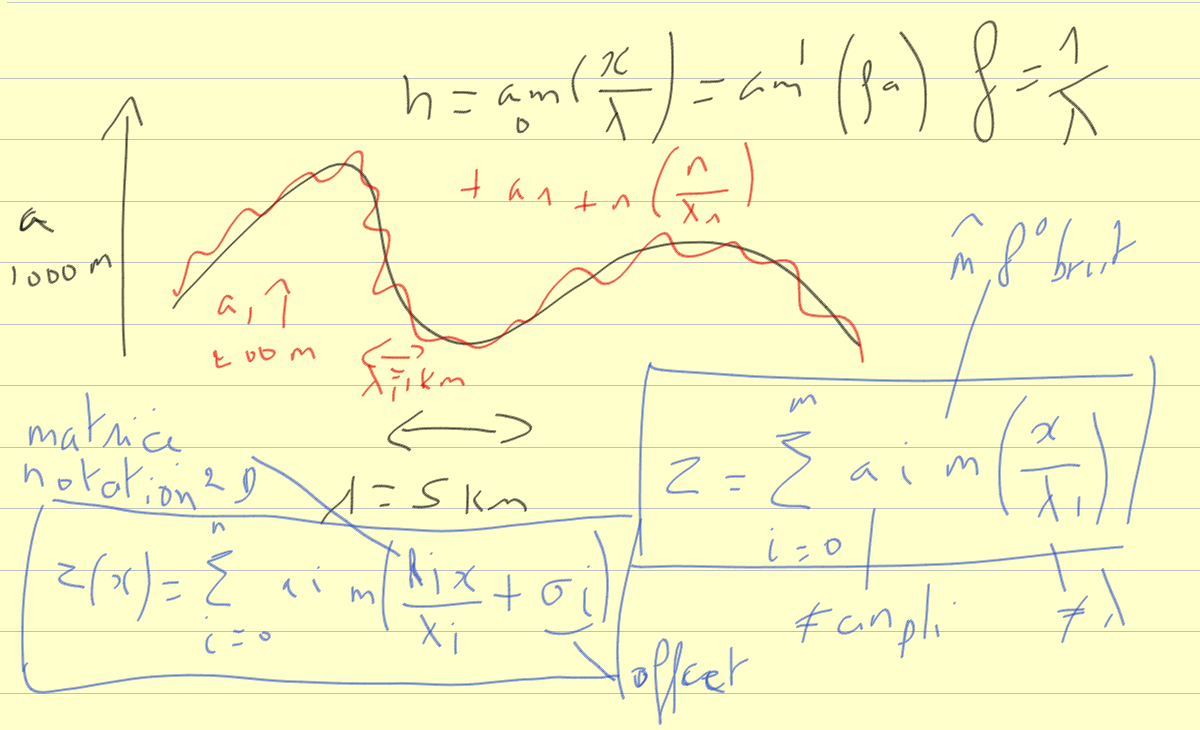
|  |  |  |
| --- | --- | --- |
|  | Valeur (hi fixe, derive =0) | « derive » = random , hi = 0 |
| Grille | Value noise (99% internet) | Perlin |
| Simplex | Simpling Noise | Simpling Noise |
| Aleatoire | Gabor Noise | Gabor noise |

Perlin :



Value noise :



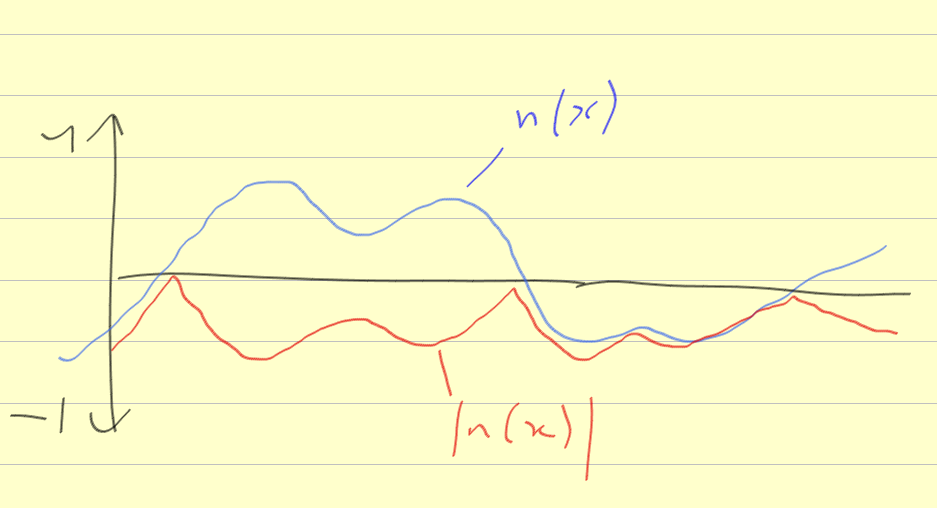


Ai = a/2^i =/- e

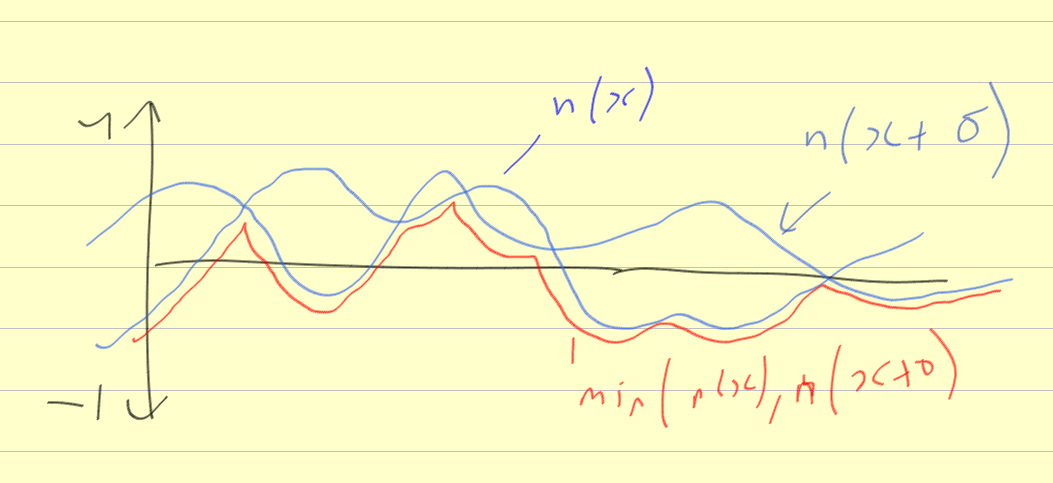
Lambda i = Lambda/2^i +/- e

### Amelioration :

Autres fonctions que n

Ridge noise r(x) = |n(x)| (car n appartient à -1 1) 

Por avoir des pic pas a la même hauteur



Mélange à fonction de support compact

Cours sur les surfaces implicites => Blend

### Hybride HF + fontion :



1 HF z

2 carte alpha

3 fonction noice

H = z(x,y) + alpha(x,y) m(x,y)