PANDA3D ET HEIGHTMAP

2ohm

23 février 2016

Zeste de savoir

Table des matières

1 Introduction

Panda3D est un moteur de jeux 2D ou 3D pour python. Si il a l'air plutôt complet aux niveaux de ses fonctionnalités, sa documentation et les tutos qui en parlent sont eux plutôt rares.

Dans ce tutoriel, cher lecteur, nous allons voir comment créer une HeightMap et jouer avec. Il se tiendra donc en 3 parties : une premier dans laquelle nous nous mettrons rapidement d'accord sur le vocabulaire, une seconde pour créer la HeightMap et la dernière pour la transformer en terrain avec Panda3D.

Seront vu en cours de route quelques petites choses de génération procédurale, une peu de blabla sur le Perlin Noise et sûrement d'autres bricoles.

L'installation de Panda3D sort du cadre de ce tutoriel. Toute fois, elle ne devrait pas vous poser de problème, le cas échéant, vous trouverez de façon certaine de l'aide sur le forum !

Enfin, j'ai rassemblé dans une archive les bouts de codes présentés dans ce tutoriel. Profitez-en !

[[information]] | **[HS]** Il n'y a pas de paquets de la version stable de Panda3D pour Ubuntu 14.04. Par contre, vous trouverez tout votre bonheur en regardant du coté de la version de développement ;)

En route !

2 Tergiversations

Génération procédurale > La génération procédurale consiste à générer du contenu (mesh 3D, texture, musique, ...) à l'aide d'algorithme plutôt qu'en faisant le travail à la main. Dans les jeux vidéos, cela signifie que le contenu du jeu peut être créé par l'ordinateur du joueur au lieu d'être préparé en studio et intégré dans les fichiers du jeu. > > (* d'après en.wikipedia.org *)

Quelques exemples ? Minecraft bien sûr, dans lequel le terrain est entièrement généré par des algorithme. Il y a aussi le rendu des vagues à la surface de l'eau dans Morrowind ou encore une très bonne partie des décors du film Avatar ! La génération procédurale intervient souvent dès lors qu'il y a un grand terrain à créer, que l'on cherche à imiter la nature, que l'on est flemmard...

HeightMap > En infographie, une HeightMap est une image utilisée pour enregistrer des informations comme le relief d'un terrain pour les afficher dans en images de synthèse. > > (*d'après en.wikipedia.org*)

Pour décrire le relief d'un terrain, de quelles informations avez-vous besoin ? La façon la plus simple de faire est de noter pour chaque point (x, y) de ce terrain l'altitude à laquelle il s'élève. La carte qui associe position (x, y) à l'altitude s'appelle HeightMap. Elle prend souvent la forme d'une image en noir et blanc. Chaque pixel de l'image correspond à un point de l'espace, la couleur du pixel code pour l'altitude du point. Par convention, un point d'altitude maximale sera blanc, un point d'altitude minimale noire.

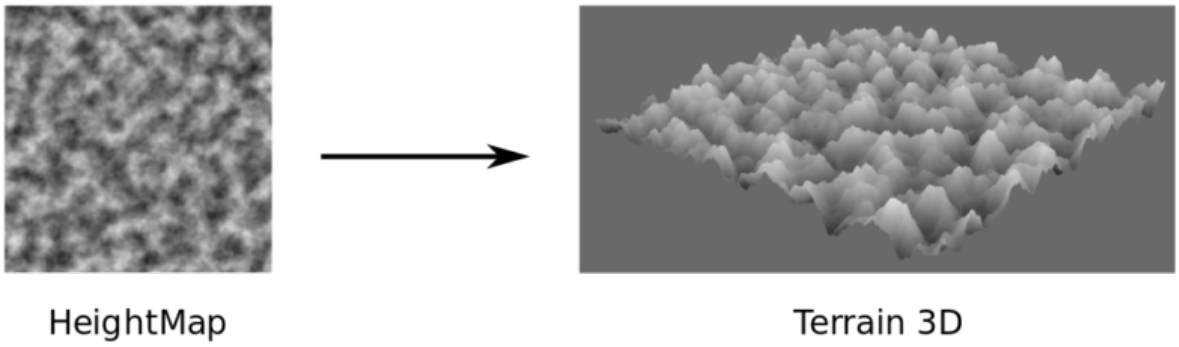


Figure 2.1 – Transformation d'une HeightMap en terrain 3D

Remarque. Vous remarquerez qu'une HeightMap ne permet pas de décrire les grottes et autres cavités.

3 Synthèse d'une HeightMap

La première étape que je vous propose est de fabriquer une image noir et blanc qui servira de HeightMap. Mais souvenez vous, cette image va coder pour le relief d'un terrain. Il faut donc réussir à générer une image qui produise ensuite un relief à l'allure naturelle.

3.1 Perlin Noise

Des chercheurs se sont amusés à trouver des méthodes pour fabriquer de telles images et ils ont abouti à plusieurs algorithmes. Nous utiliserons celui de M. Perlin, mais sachez qu'il existe aussi le Diamond Algorithm, le Midpoint Displacement et autres. Dans tous les cas, ces algorithmes ressemblent beaucoup à des générateurs de nombres aléatoires. Les nombres qu'ils produisent regardés les uns après les autres semblent totalement décorrélés : on dit qu'ils produisent du bruit.

[[information]] | En vrai de vrai, ce ne sont pas de véritables générateurs de nombres aléatoires car sinon votre HeightMap ressemblerait à l'écran d'une vieille TV cathodique dont on a oublié d'allumer le magnétoscope. Dans ce cas, le terrain généré à partir de cette HeightMap ressemblerait à une forêt de pics et pas du tout à un relief naturel.

M. Perlin a imaginé une fonction qui produit du bruit bien particulier : il est auto-similaire à différentes échelles. (C'est aussi un pouvoir que les fractales ont.) Pour clarifier les choses, prenons un exemple : la Bretagne.

Si vous demander à un Strasbourgeois à quoi ressemble la Bretagne, il vous décrit un morceau de la France très éloigné de chez lui qui ressemble vaguement à :

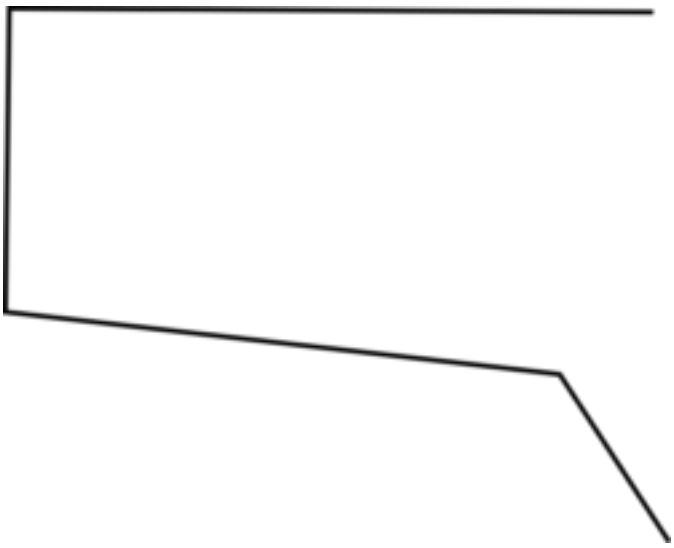


Figure 3.1 – La Bretagne vue par un strasbourgeois

Si maintenant vous demandez à un versaillais, il vous décrira avec passion, mais sans grand détails géographiques, les côtes bretonnes telles qu'il a pu les observer lors des régates auxquelles il ne manque pas de participer tous les étés.

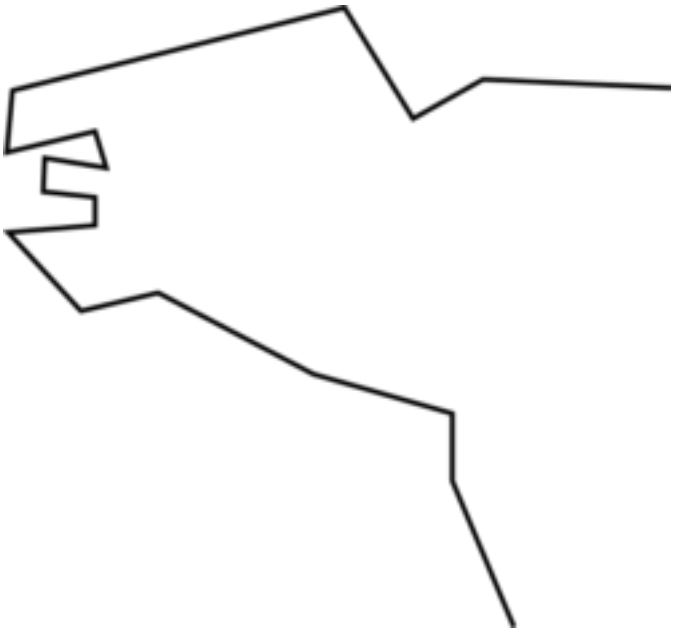


Figure 3.2 – La Bretagne vue par un versaillais

Maintenant, si vous demandiez bien fort au vieux pécheur sourd comme un pôt qui profite du beau temps sur le port, il vous parlera de chaque petites plages et de chaque criques secrètement nichées loin des touristes.

Figure 3.3 – La Bretagne vue par un breton

Est-ce que vous commencez à comprendre ? La Bretagne peut être décrite à différentes échelles et chacune apporte son lot d'informations, mais à chaque échelle, la géométrie reste la même.

Le Perlin Noise se construit de la même façon. Une première fréquence (appelée octave) en décrit les grands mouvements.

Et puis une seconde octave plus élevée apporte des perturbations.

Figure 3.4 – Première octave

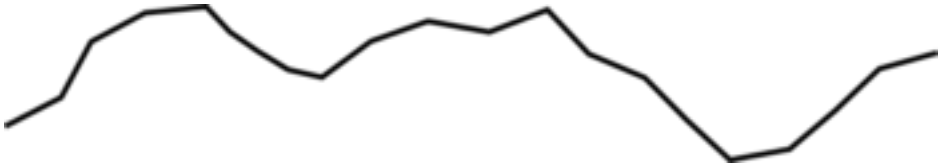


Figure 3.5 – Deuxième octave

Et puis encore une autre, encore plus élevée.

Et cætera… Il est possible de continuer ainsi aussi longtemps qu'on le souhaite !

C'est donc en utilisant le Perlin Noise et sa propriété d'auto-similarité que nous allons pouvoir construire une HeightMap vraisemblable. Le bruit à basse fréquence donnera la forme générale du terrain (collines, montages, pleine, creux, …) alors que le bruit à haute fréquence créera des irrégularités locales.

Avant de passer à la suite :

1. ☒ J'ai compris ce qu'était la génération procédurale ?
2. ☒ J'ai compris le concept de HeightMap ?
3. ☒ J'ai installé Panda3D ?

3.2 Perlin & Python

Nous n'allons pas écrire l'algorithme pour fabriquer le bruit de Perlin, mais plutôt utiliser une bibliothèque : Noise. (informations supplémentaire en annexe)

[[information]] | Pour obtenir de la documentation de Noise, chargez le paquet dans l'invite de commandes de python et utilisez `help(noise)`.

Noise propose une fonction très chouette qui répond au doux nom de `snoise2()`. *s* par ce qu'elle utilise la version 2 de la fonction de Perlin (dite *simplex*) et 2 parce qu'elle travaille en 2D. `snoise2()` accepte de nombreux paramètres, nous n'utiliserons que les 3 premiers.

```
snoise2(X, Y, nb_octaves)
```

Figure 3.6 – Troisième octave

- `X` -> abscisse du pixel
- `Y` -> ordonnée du pixel
- `nb_octaves` -> nombre d'octaves dans la génération du bruit de Perlin

Après avoir installé Noise, faites vous plaisir, et jouez avec les exemples ! (Cela à un double intérêt : vous pourrez vérifier que la librairie est bien installée, et puis c'est rigolo :p)

C'est fait ? Alors, est-ce que vous avez repéré le fichier d'exemple qui va particulièrement nous intéresser ? Dans les exemples, il y a en effet un programme qui génère du bruit de Perlin 2D et qui l'enregistre dans une image. Ouvrons le. (C'est `2dtextures.py` pour ceux qui n'ont pas fait joujou avec les exemples.)

Le code est très simple. Une image est crée et la couleur de chaque pixel est donné par `snoise2()`. Vous remarquerez en particulier que l'auteur du programme divise `x` et `y` par le produit d'une fréquence et du nombre d'octaves. Cette fréquence est un facteur sur lequel vous allez pouvoir jouer pour modifier l'échelle de votre terrain.

Allez hop, on met les mains dans le cambouis et chacun crée sa HeightMap ! Pour la générer, utilisez directement `2dtextures.py` ou bien la version légèrement modifiée que je vous propose (`mapGenerator.py`). Dans les 2 cas, la syntaxes est la même :

```
python 2dtextures.py [nom_du_fichier_de_sortie] [nb_octaves]
python mapGenerator.py [nom_du_fichier_de_sortie] [nb_octaves]
```

Histoire de personnaliser votre HeightMap, voici les paramètres sur lesquels vous pouvez jouer :

- `freq`
- `nb_octaves`

Pouvez-vous me dire quelle influence ils ont sur l'image générée ?

3.2.1 Influence de `freq` et de `nb_octaves`

C'est bon, vous avez fait vos expériences ? Super ! Alors, sommes nous arrivés aux mêmes conclusions ?

J'ai trouvé qu'en diminuant `freq`, l'image devient très bruitée. Au contraire, lorsque `freq` est grand, l'image devient toute lisse.

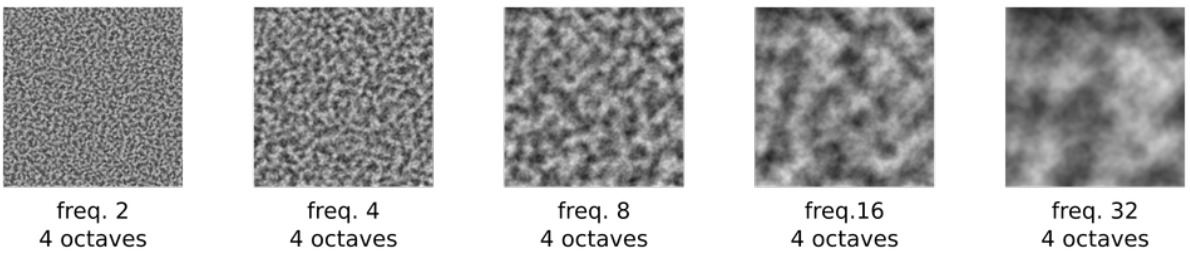


Figure 3.7 – Lorsque la fréquence diminue, le Perlin Noise est beaucoup plus granuleux.

J'ai l'impression que ce que l'auteur du programme à appelé fréquence soit en réalité une ... période. (ie l'inverse d'une fréquence) Cette fréquence, qui n'en est pas une, fixe la plus petite période des perturbations. Ou autrement dit, les variations spatiales les plus courtes ne seront pas plus petites que la fréquence.

J'ai remarqué que plus `nb_octaves` est grand, plus l'image générée est détaillée, granuleuse et plus le motif générale est grand ; et c'est tout à fait normal. Les fréquences des plus grandes octaves sont diminuées de telle sorte que la dernière octave ajoutée ait une fréquence égale à freq (qui reste constant).

Figure 3.8 – Lorsque le nombre d'octaves augmente, le Perlin Noise devient plus régulier.

[[information]] | | **[Aller plus loin]** La génération d'un Perlin Noise est contrôlée par deux autres paramètres : la *persistance* et la *lacunarité*. La documentation de noise vous expliquera comment les faire intervenir dans `snoise2()`.

Pour la suite du tutoriel, vous avez besoin d'une heighmap, alors prenez le temps d'en fabriquer une à votre goût !

Si vous utilisez le matériel de ce tutoriel, merci de l'appeler map et de la placer dans le dossier … map !

```
python mapGenerator.py map/map [nb_d'octaves]
```

Et puis, vous pouvez aussi en changer la taille. Tachez de ne pas non plus en faire une d'une taille trop démente !

Important : c'est mieux si la taille de votre HeightMap est de la forme $2^n + 1$.

[[information]] | **[Aller plus loin]** Intrigué par le bruit de Perlin ? Le site suivant vous aidera a en apprendre plus. | http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

Avant de passer à la suite :

1. ☒ J'ai installé la librairire Noise ?
2. ☒ J'ai généré ma HeightMap à moi ?

4 Au tour de Panda3D

Vous allez voir, générer un terrain à partir d'une HeightMap dans Panda3D, c'est du gâteau !

4.1 GeoMipTerrain

Tout d'abord, on commence par créer un objet GeoMipTerrain, on lui attribue ensuite la HeightMap que l'on vient de créer et finalement, on l'accroche au render. Cela donne :

```
## Prépare le terrain
self.terrain = GeoMipTerrain("map_test")
self.terrain.setHeightfield("map/map")
## Attache le terrain au render
self.terrain.getRoot().reparentTo(render)
```

Il manque une toute dernière étape et ce sera (presque) fini : générer le terrain.

```
## Génère le terrain
self.terrain.setBruteforce(True)
self.terrain.generate()
```

[[information]] | GeoMipTerrain permet de générer un terrain avec différent niveaux de détails. Cette fonctionnalité est très pratique pour les très grands terrains. Ici, on ne va pas mettre en place ce système, on force donc panda3D a rendre notre détails en mettant la gomme sur les détails, c'est le rôle de `setBruteforce(True)`.

Si vous vous contentez de ça, vous remarquerez vite que votre monde est un peu petit et plat. Un petit coup de scale, et tout ira mieux ! Vous pouvez aussi en profiter pour placer votre terrain là où ça vous convient.

```
## Positionne et modifie l'échelle du terrain
terrainScale = 10
self.terrain.getRoot().setScale(terrainScale, terrainScale, terrainScale*40)
self.terrain.getRoot().setPos(-128, -128, 0)
```

PS. Pensez à faire ces transformations avant de générer votre terrain.

Et voilà, nous y sommes ! Si vous lancez maintenant votre programme, vous devriez voir s'afficher sous vous yeux ébahis … une grosse masse blanche.

Pas de panique !

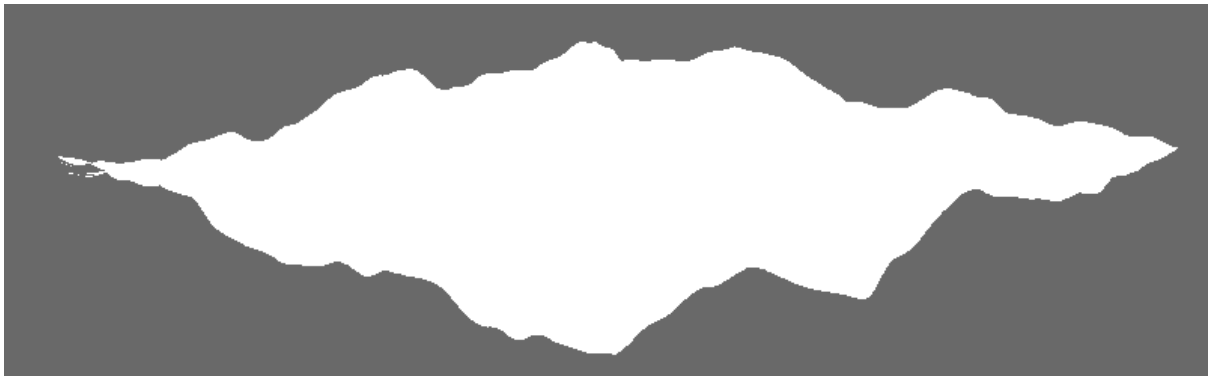


Figure 4.1 – La grosse masse blanche ...

4.2 Texture

Comme nous n'avons pas mis de système d'éclairage en place, il n'y a pas d'ombres pour relever le relief du terrain. Dans un jeu classique, il faudrait bien sur ajouter des lampes, je vais toute fois vous montrer ici autre chose : nous allons appliquer à notre terrain la HeightMap comme texture.

```
## Applique une texture
tex = loader.loadTexture('map/map')
self.terrain.getRoot().setTexture(tex, 1)
self.terrain.getRoot().setTwoSided(True)
```

La première ligne charge la texture, la seconde l'applique et la troisième est une petite fioriture de ma part : grâce à elle, notre terrain est texturé qu'on le regarde du dessus, ou du dessous !

4.3 Conclusion

Le code minimal final qui permet de charger une HeightMap ressemble donc à :

```
## -----------------------------------------------------------
# showMap
## Ouvre une HeightMap avec Panda3D et GeoMipTerrain.
# -----------------------------------------------------------
## Current: showMap.py   2014-08-13  by 2ohm
# -----------------------------------------------------------
from panda3d.core import *
from direct.showbase.ShowBase import ShowBase

class MyApp(ShowBase):

    def __init__(self):
        ShowBase.__init__(self)

        # Prépare le terrain
        self.terrain = GeoMipTerrain("map_test")
        self.terrain.setHeightfield("map/map")
```

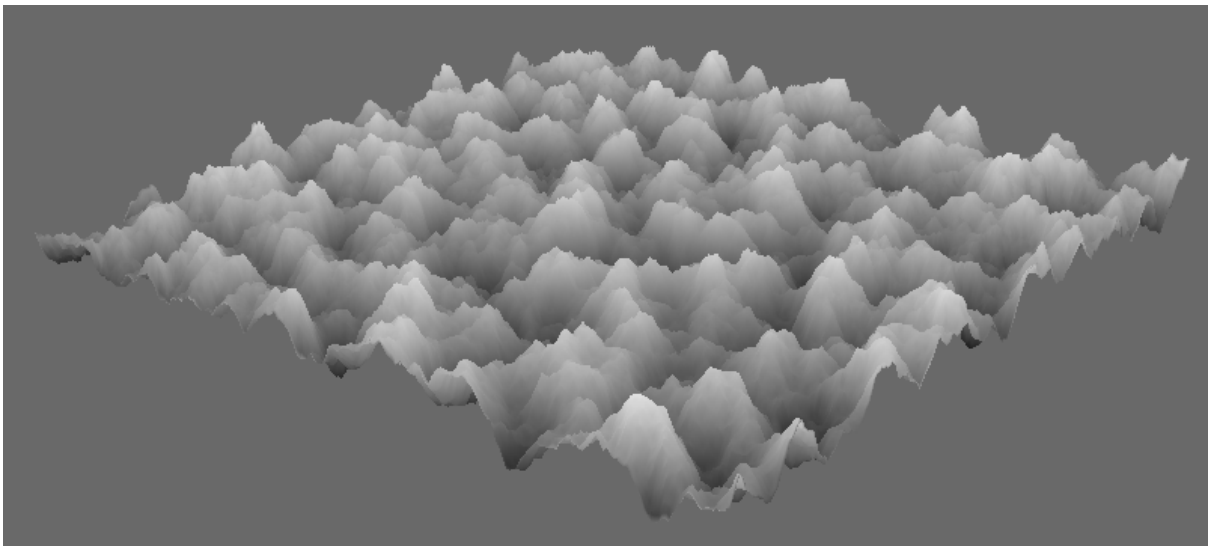
```
        # Attache le terrain au render
        self.terrain.getRoot().reparentTo(render)
        # Positionne et modifie l'échelle du terrain
        terrainScale = 10
        self.terrain.getRoot().setScale(terrainScale,
                                        terrainScale,
                                        terrainScale*40)
        # Génère le terrain
        self.terrain.setBruteforce(True)
        self.terrain.generate()

        # Applique une texture
        tex = loader.loadTexture('map/map')
        self.terrain.getRoot().setTexture(tex, 1)
        self.terrain.getRoot().setTwoSided(True)

app = MyApp()
app.run()
```

Vous devriez obtenir quelque chose qui ressemble à :

->



<-

Magique non !?

Avant de passer à la suite :

— ☒ J'ai réussi à générer un terrain depuis ma HeightMap et à l'afficher.

5 Annexe - installation de Noise

5.1 Installer Noise sous linux :

1. Commencez par télécharger la librairie : .
2. Décompressez l'archive dans un dossier temporaire.
3. Exécutez l'installateur avec : `python setup.py install --user`

5.2 Installer Noise sous windows :

Je ne connais pas la procédure à suivre sous Windows. N'hésitez pas à me l'indiquer si vous la connaissez.

5.3 Tester son installation de Noise :

Avant de vous lancer dans la suite du tutoriel, vérifié que l'installation de Noise s'est déroulée sans problème.

1. Lancer l'interpréteur python.
2. Si la commande `import noise` ne retourne pas d'erreur, tout est bon !
3. La documentation est accessible avec `help(noise)`.

6 Conclusion

Nous voilà arrivé à la fin de ce mini-tuto. À la lecture de celui-ci, vous devriez avoir appris ce qu'est une HeightMap et comment la transformer en terrain avec Panda3D. N'est-ce pas magnifique !?

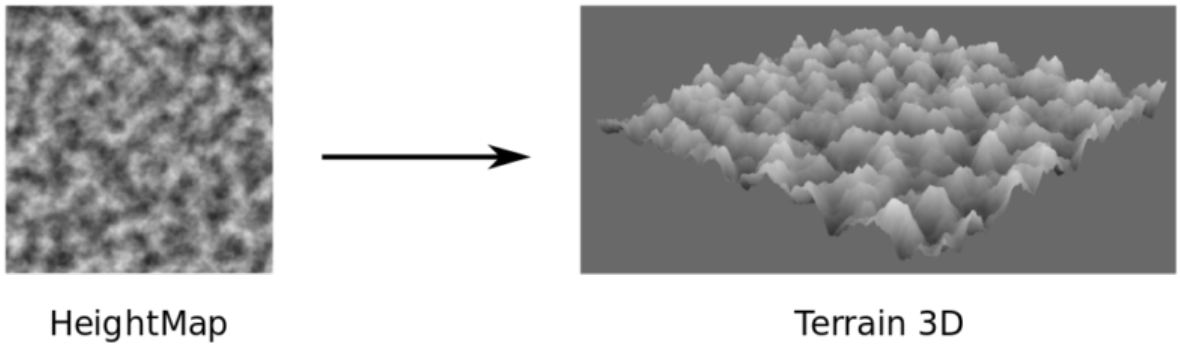


Figure 6.1 – Ce que vous savez maintenant faire ...

Maintenant, c'est à vous de jouer !

-> ~2ohm ->