Rapport ERDI

Cyril Colin
Voisin Dylan

Master 1 Informatique
Université d'Aix-Marseille

Décembre 2019

Résumé

Le bruit procédural phasé ou bruit de phase est une nouvelle méthode de générer des textures de manière procédurale. Ceci est souvent utilisé dans les rendus 3D pour fournir un haut niveau de détail tout en réduisant l'utilisation mémoire par rapport aux méthodes plus « traditionelles » qui utilisent des textures prégénérées chargée depuis le disque dur. Le bruit de phase permet de générer des textures présentant des crêtes comme l'écorce d'un arbre ou du sable. De plus, il peut également être utilisé en impression 3D où il peut générer des motifs de déposition de matière. Contrairement aux approches existantes, le bruit de phase produit des résultats avec moins d'artéfacts visuels et plus de contraste.

Abstract

"Procedural Phasor Noise" presents a novel way to generate textures in a procedural manner. This is quite often used in 3D renders to provide great details and avoid obvious repeating patterns all the while greatly reducing memory usage compared to the more "traditional" methods of using pre-computed textures loaded from disk. Procedural phasor noise is one such way to generate textures presenting ridges such as the bark of a tree or sand. Furthermore, it also presents

applications in the field of additive manufacturing where it can be used to generate material deposition patterns. In contrast to previous methods, the procedural phasor noise produces results with less artifacts and more contrast.

Table des matières

Introduction

Dans le monde de l'Infographie et plus particulièrement dans le domaine de la modélisation, les textures sont devenues indispensables pour associer un modèle et une image, afin de créer un objet réaliste, avec sa forme et ses motifs, apportés par les textures. Les domaines d'application sont nombreux, l'animation, les images de synthèses utilisées dans les films, ou encore les jeux vidéos. Plus les textures sont précises (et donc par conséquent lourde en terme d'espace disque), plus la qualité du rendu final sera réaliste.

Il se pose par conséquent une question de limitations, en effet, ces textures de très hautes qualités prennent certes de la place, mais également du temps à concevoir et à créer. Un autre problème peut également intervenir, même si l'on veut une texture très belle et que l'on peut y arriver, on aimerait dans l'idéal avoir plusieurs textures pour un type d'objet donné. Prenons l'exemple des jeux vidéos, il serait assez facheux de n'avoir que 3 ou 4 textures pour les arbres, donnant ainsi une diversité assez pauvre et une répétition qui peut vite lasser l'utilisateur, qu'importe la qualité de rendu de ces dites textures.

Avoir suffisamment de textures de bonne qualité prendrait beaucoup trop de temps et de place pour pouvoir être créé. Il faudrait donc palier à cela en utilisant d'autres procédés. Fort heureusement, ces procédés existent et se basent pour la plupart sur des particularités présentes dans la nature, et un point commun entre les objets visibles dans la nature est bien entendu le hasard. De plus, les objets et formes de même type ont des propriétés communes entre elles. De ce fait, nous sommes en mesure de créer des fonctions mathématiques basées sur l'aléa capables de s'adapter, moyennant certains paramètres, à plusieurs textures présentes dans la nature. Pour rendre le tout encore plus attractif, ceci se fait de manière procédurale.

Avec cette méthode, nous sommes donc capables de palier non seulement la qualité mais aussi la diversité des textures concevables, le tout de manière presque instantannée et sans prendre d'espace disque. Ce type de texture est plus communément appelé « textures procédurales » et permet la synthèse de textures comme le bois, le marbre, le granite, le métal...

Ces textures procédurales sont cependant générées avec des procédés différents selon le type de texture que l'on souhaite obtenir, ces procédés sont appelés les générateur de bruits, puisque basés sur l'aléa. On notera cependant que certaines méthodes telles que le texturage cellulaire ne se basent pas sur un générateur de bruit.

Dans ce rapport, nous allons donc voir différents types de générateurs de bruits, on

se limitera cependant aux générateurs de bruits gradients, par oppositions aux bruits de valeurs. Dans un premier temps, nous verrons l'exemple du bruit de Perlin, s'en suivra le bruit de Gabor qui a inspiré notre dernière partie qui se concentre autour du sujet d'étude : le « Phasor Noise ».

1 Bruit de Perlin

Dans cette partie, nous nous intéresseront au bruit de Perlin. Pour le contexte de la découverte de ce générateur de bruit, voici un extrait de l'article Wikipédia dédié :

« Le bruit de Perlin a été développé par Ken Perlin en 1985. À cette époque, après avoir travaillé sur les effets spéciaux de Tron pour MAGI en 1981, il cherchait à éviter le look « machinique ». Il commença donc par mettre au point une fonction pseudo-aléatoire de bruit qui remplit les trois dimensions de l'espace, avant d'inventer l'année suivante le premier langage de shading. Ses travaux sur les textures procédurales ont valu à Ken Perlin l'Academy Award for Technical Achievement en 1997. »

Une des particularité de sa méthode est que les détails du bruit généré sont de même taille, ce qui permet de combiner une texture générée d'une taille donnée avec d'autres échelles pour ajouter de la profondeur.

1.1 Principe

L'algorithme du bruit de Perlin se décompose en 3 parties que sont :

1. la définition de la grille.
2. le calcul du produit scalaire entre le vecteur gradiant et le vecteur distance.
3. l'interpolation entre ces valeurs.

Pour la définitionde la grille, il faut définir une grille à n dimensions. Attribuer pour chaque nœud un vecteur de gradient aléatoire de norme 1 et de dimension n.

Pour ce qui concerne le produit scalaire, nous faisons comme suit :

Soit un point de l'espace à n-dimensions envoyé à la fonction de bruit, l'étape consiste à déterminer dans quelle cellule de grille le point donné se situe. Pour chaque nœud-sommet de cette cellule, calculer le vecteur distance entre le point et le nœud-sommet. Puis calculer le produit scalaire entre le vecteur de gradient au nœud et le vecteur de distance. Cela conduit à l'échelle de complexité $O(2^n)$.

La dernière étape est l'interpolation entre les 2^n produits scalaires calculés aux nœuds de la cellule contenant le point d'argument. Cela a pour conséquence que la fonction de bruit renvoie 0 lorsqu'elle est évaluée sur les nœuds de la grille eux-mêmes.

L'interpolation est effectuée en utilisant une fonction dont la dérivée première (et éventuellement la dérivée seconde) est nulle aux 2^n nœuds de la grille. Cela a pour effet que le gradient de la fonction de bruit résultante à chaque nœud de grille coïncide avec le vecteur de gradient aléatoire précalculé.

Voici une version C++ de l'implantation du bruit de Perlin à 2 dimensions :

```
// Function to linearly interpolate between a0 and a1
// Weight w should be in the range [0.0, 1.0]
float lerp(float a0, float a1, float w) {
    return (1.0 - w)*a0 + w*a1;

    // as an alternative, this slightly faster equivalent
    // formula can be used:
    // return a0 + w*(a1 - a0);
}

// Computes the dot product of the distance and gradient
// vectors.
float dotGridGradient(int ix, int iy,
                      float x, float y) {

    // Precomputed (or otherwise) gradient vectors at each
    // grid node
    extern float Gradient[IYMAX][IXMAX][2];

    // Compute the distance vector
    float dx = x - (float)ix;
    float dy = y - (float)iy;

    // Compute the dot-product
    return (dx*Gradient[iy][ix][0] + dy*Gradient[iy][ix][1]);
}

// Compute Perlin noise at coordinates x, y
float perlin(float x, float y) {

    // Determine grid cell coordinates
    int x0 = int(x);
    int x1 = x0 + 1;
    int y0 = int(y);
```

```
    int y1 = y0 + 1;

    // Determine interpolation weights
    // Could also use higher order polynomial/s-curve here
    float sx = x - (float)x0;
    float sy = y - (float)y0;

    // Interpolate between grid point gradients
    float n0, n1, ix0, ix1, value;
    n0 = dotGridGradient(x0, y0, x, y);
    n1 = dotGridGradient(x1, y0, x, y);
    ix0 = lerp(n0, n1, sx);
    n0 = dotGridGradient(x0, y1, x, y);
    n1 = dotGridGradient(x1, y1, x, y);
    ix1 = lerp(n0, n1, sx);
    value = lerp(ix0, ix1, sy);

    return value;
}
```

1.2 Applications

Due à sa grande flexibilité, ce générateur de bruit peut être utilisé pour créer plusieurs types d'éléments comme les nuages, le feu, la fumée... Voici Quelques exemples d'images produites depuis cette méthode :

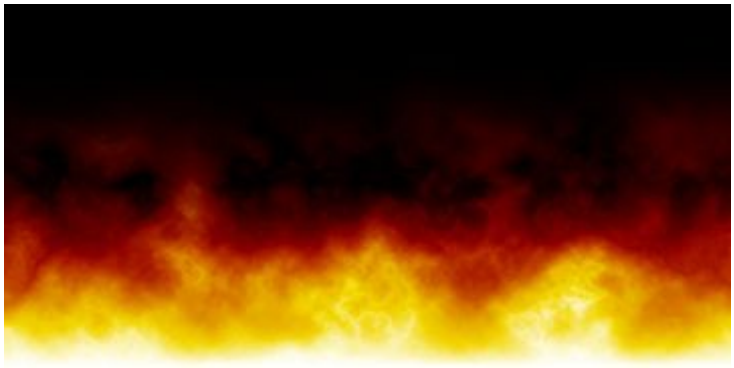


Figure 1 : Feu généré à partir du bruit de Perlin

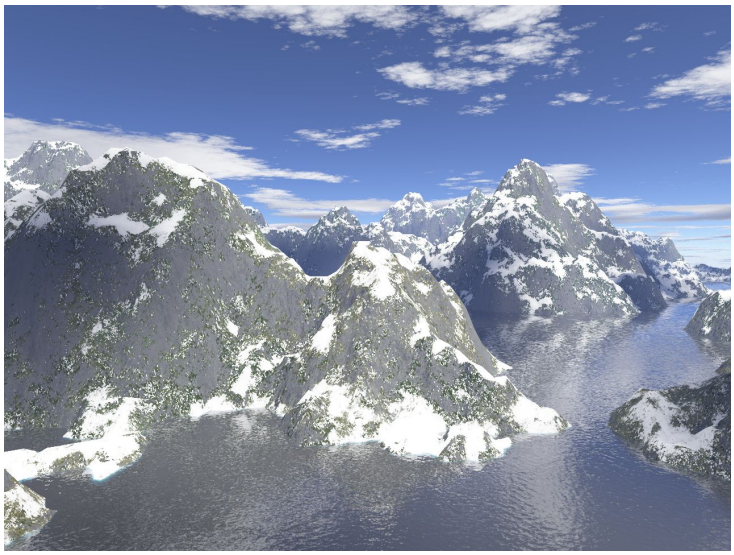


Figure 2 : Terrain généré à partir du bruit de Perlin

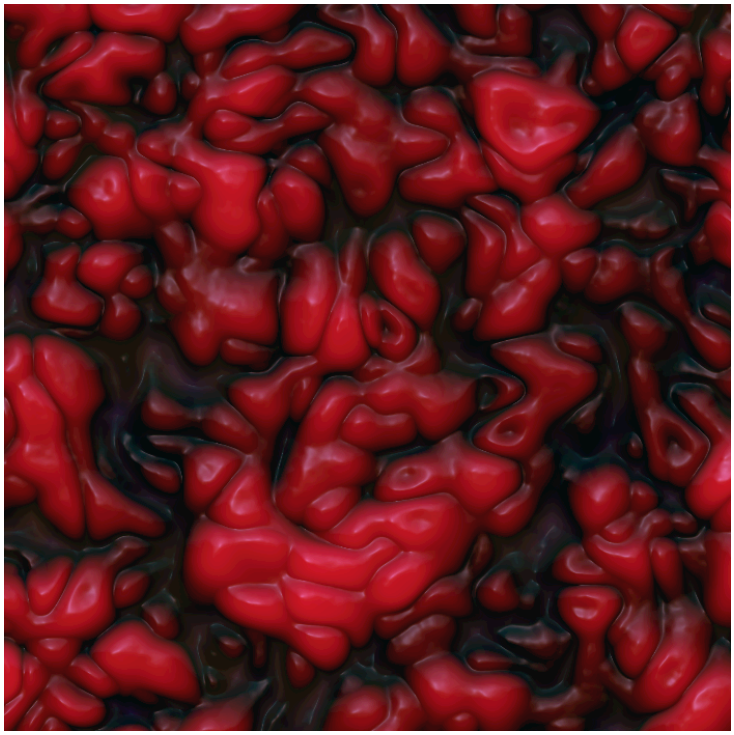


Figure 3 : Surface organique générée à partir du bruit de Perlin et travaillée avec application texture et lumière

2 Bruit de phase

Malgré la grande modularité du bruit de Perlin, celui-ci reste assez peu contrôlable et nécessite très souvent d'être recombiné à plusieurs reprises afin d'avoir l'effet souhaité. Un autre reproche fait aux générateurs de bruit traditionnels est le fait de mettre un seuil maximum aux valeurs générées par le bruit, résultant de pertes de contraste ainsi qu'une limitation de contrôles.

Nous allons à présent voir un générateur de bruit qui a l'avantage de véritablement être paramétrable, il s'agit du sujet d'étude : le « Phasor Noise », on conviendra de parler de bruit de phase par la suite.

Le bruit de phase s'appuie sur un autre bruit qu'est le bruit de Gabor et a pour but de l'améliorer, tout en laissant plus de liberté pour la génération.

2.1 Principe

Le bruit de phase s'appuie sur le bruit de Gabor pour 2 de ses qualités : l'évaluation spatiale du contenu du bruit ainsi que le fait de pouvoir appliquer le bruit en 2D et 3D et sur des surface ayant des champs de tangeantes.

Un des défauts majeur du bruit de Gabor est la perte locale de contraste et des fluctuations d'orientations des sinusoïdes générées due à la non séparation de l'intensité et de la phase.

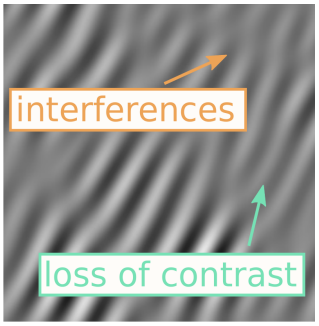


Figure 4 :

Mise en valeur des problèmes du bruit de Gabor

Contrairement à d'autres méthodes, les bruits de Gabor et de phase ne génèrent pas de champ scalaire mais un champ de phase, destiné à être modifié par une fonction périodique dans le cas du bruit de phase.

L'avantage du bruit de phase est que cette fonction permet de définir le profil des motifs à générer. Ce motif suivra cette fonction sans fluctuation. Le bruit de phase est obtenu via la phase instantannée des oscillations du bruit de Gabor, on sépare la phase et l'intensité pour éviter les problèmes du bruit de Gabor. Ainsi, les pertes locales de contraste sont corrigées et l'image résultante est plus nette que celle obtenu via la méthode de Gabor.

Le bruit de Gabor est déjà paramétrable de base, on peut ainsi changer la fréquence (l'échelle du motif) et l'orientation des sinusoïdes.

Le bruit de phase réécrit le bruit de Gabor en une sinusoïde. On a ainsi :

$$\text{PhasorNoise}(x) = \phi(x)$$

$\phi(x)$ étant la phase instantannée du bruit de Gabor modifié. On définit la sinusoïde du bruit de phase comme suit :

$$\text{PhasorSinewave}(x) = \sin(\text{PhasorNoise}(x))$$

On peut également contrôler le profil des oscillations via une fonction f définie sur 2π :

$$\text{ProfiledNoise}(x) = f(\text{PhasorNoise}(x))$$

Cependant, 2 types de problèmes persistent dans la méthode du bruit de phase. Ces 2 problèmes surviennent lorsque la phase tend vers 0. Premièrement, il y a des points

pour lesquels la bande de la sinusoïde apparaît, ce qui donnt une forme de « Y » car 2 lignes fusionnent (ou se détachent selon le sens). Deuxièmement, il y a des composantes du champ de phases où les valeurs à leur frontière sont en opposition, cela cause une inversion entre les lignes des sinusoïdes et les « blancs ». Ce deuxième type de problème est plus flagrant et donc plus problématique. L'apparition de ces problèmes est impactée par un paramètre b qu'est la bande passante du bruit de Gabor, plus sa valeur est basse, plus le signal sera sinusoïdal et moins ces anomalies apparaitront.

Cependant, une correction à ces problème à été apportée en 2016 par Nyret et Heitz, mais ce procédé n'est pas faisable procéduralement, la correction de ces problèmes est donc un travail de recherche à venir.

2.2 Libertés de contrôle et exemples

Puisque le bruit de phase est issu du bruit de Gabor, nous pouvons faire varier le nombre de bi-lobes et obtenir différents types de rendu :

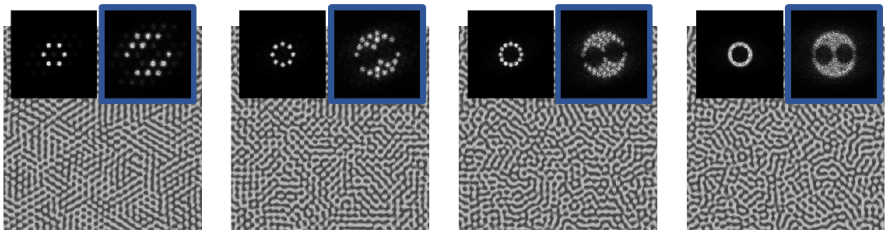


Figure 5 : Bruit de phase avec 3, 4, 6 et 32 bi-lobes avec orientations équitablement réparties sur π

Le bruit de phase permet également de contrôler les champs de phases. Nous pouvons bien entendu combiner le bruit de phase avec un PWM afin de le transformer, et on peut également agir sur les orientations. Nous pouvons voir tout cela sur la figure 6.

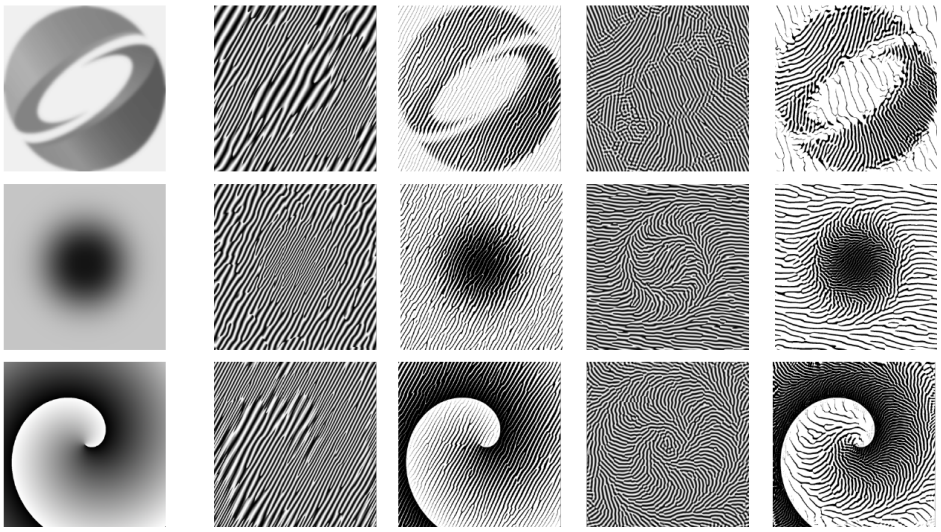


Figure 6 : **Colonnes, de gauche à droite** :contrôle des champs, contrôle de fréquences, contrôle PWM sur la largeur, contrôle d'orientation, et tout confondus

3 Apports du bruit de phase

3.1 Avantages

Les méthodes de générations procédurales permettent d'obtenir des motifs avec un haut niveau de détail et sans répétition évidente, tout en diminuant le coût en mémoire par rapport à d'autres méthodes.

Le bruit de phase en particulier et par rapport aux autres bruits propose un contraste élevé et une multitude de contrôles qui permettent de l'adapter à ses besoins. En effet, tandis que d'autres bruits comme le bruit de Perlin nécessitent souvent plus de travail pour parvenir à un résultat convenable, tel que la superposition de différents bruits à différentes résolutions ou bien l'application de filtres, le bruit de phase donne un résultat plus convainquant pour moins de travail.

Cependant il est peut-être moins versatile, étant donné qu'il génère un champs de lignes.

Il reste idéal pour toutes les applications qui ont besoin de tels motifs, on en retrouve principalement deux.

D'une part pour la génération de textures à des fin de génération d'image par ordinateur, notamment pour des objets qui comportent des « fissures » régulières comme les troncs d'arbre, du sable, de la terre craquelée... Cette utilisation est d'autant plus pertinente car le bruit de phase peut sans-problème être implémenté en tant que shader de pixel pour s'exécuter en temps réel à moindre coût, et même s'adapter à d'éventuels changements de géométrie.

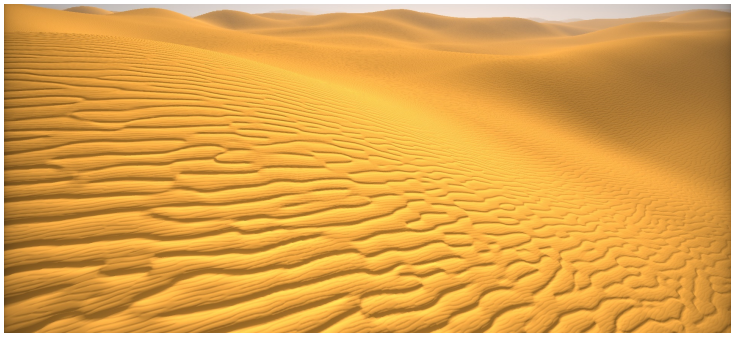


Figure 7 : Une texture de sable

(a) Une texture fissurée

(b) Une texture de tronc d'arbre, dont les propriétés diffèrent au sommet

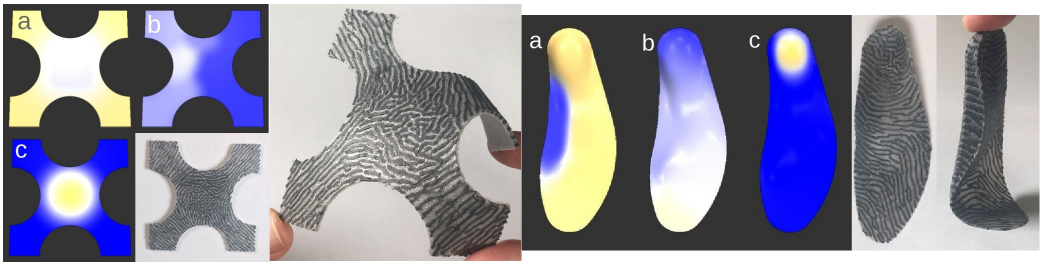
Figure 8 : Démonstrations des capacités du bruit de phase

3.2 Utilités en fabrication additive

Le bruit de phase peut également être utilisé dans le cadre de la fabrication additive. C'est un procédé qui consiste à concevoir un objet ou une pièce mécanique en déposant des couches de matériaux succesives. Cela peut être du plastique fondu à l'aide d'une tête chauffante comme c'est le cas dans les imprimantes 3D que l'on peut retrouver dans le commerce, mais aussi des métaux sous forme de poudre qui est fondue à l'aide d'un LASER.

Cela permet de produire des pièces difficiles à produire avec les méthodes classiques d'usinage par exemple, en utilisant moins de matière première (pas de chutes). On peut également avoir un contrôle plus fin sur la quantité et éventuellement le mélange de différents matériaux.

Dans ce contexte, le bruit de phase peut servir à produire des motifs à fin de construire des pièces flexibles mais résistantes, voire même de contrôler les degrés de liberté (les manières dont on peut tordre l'objet).



(a) Une surface souple en forme de croix (b) Une semelle dûre au talon, souple à la plante

Figure 9 : Démonstration des applications à l'impression 3D

4 Conclusion

Pour conclure, nous savons désormais que les générateurs de bruits ne sont pas utilisés que pour faire de la génération de texture procédurale, mais aussi pour créer des formes, ou encore être employés dans la synthèse additive. Les générateurs de bruits sont nombreux et s'appuient sur des procédés différents. Ainsi le bruit de Perlin peut s'appliquer, au niveau des textures, à ce qui s'apparente à un amas de particules plus ou moins dense comme le feu et la fumée, même si le rendu n'est pas véritablement agréable et comparable avec ce qui se fait en matière de flammes de nos jours. En revanche il peut convenir en génération de terrain tridimensionnel procédural, même si là encore, il y a des méthodes plus adaptées. Concernant le bruit de phase, nous avons vu des choses qui nous ont assez impressionnés avec notamment l'application avec les empreintes digitales qui étaient étonnamment réalistes et détaillées, l'apect des textures torsadées générées (la corde du bracelet présentée dans le sujet d'étude) mais aussi son utilité dans la synthèse additive avec les propriétés des objets ainsi créés. Nous avons donc appris qu'il existait des domaines d'applications assez vastes pour les générateurs de bruit et que chacun d'eux avait ses avantages, ses limites et ses particularités.

5 Bibliographie

Thibault Tricard, Semyon Efremov, Cédric Zanni, Fabrice Neyret, Jonàs Martínez, et al.. Procedural Phasor Noise. ACM Transactions on Graphics, Association for Computing Machinery, In press, pp.1- 13. 10.11453306346.3322990. hal-02118508

Jérémy Cochoy, Introduction au bruit de Perlin, 10 février 2011, `http://www.cochoy.fr/pdfs/perlin-noise/perlin_noise.pdf`

`https://en.wikipedia.org/wiki/Perlin_noise`

`https://fr.wikipedia.org/wiki/Bruit_de_Perlin`