

Travail de fin d'études réalisé en vue de l'obtention du  
diplôme de Bachelier en Techniques graphiques

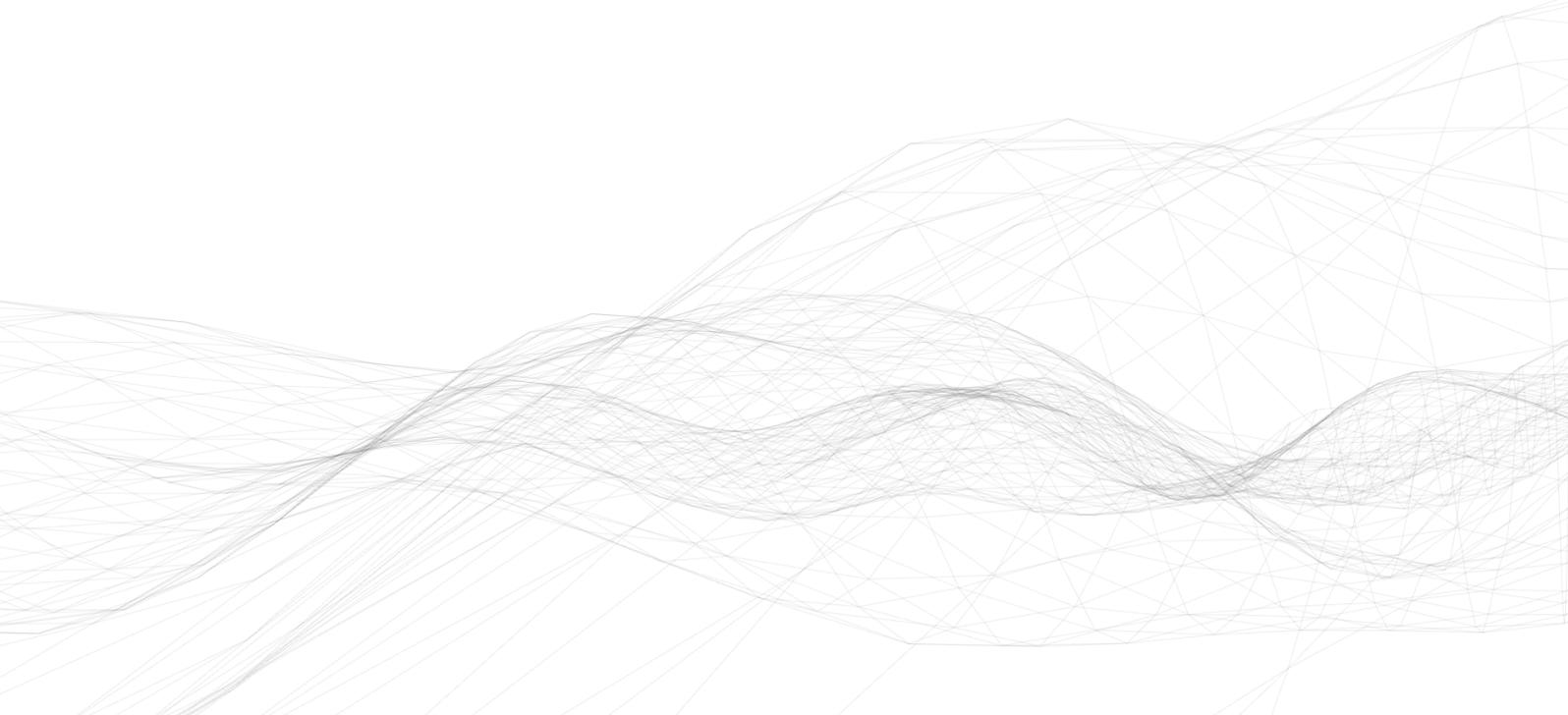
# LES USAGES DU BRUIT EN INFOGRAPHIE

```
var i = Math.floor(xin/s);  
var j = Math.floor(yin/s);  
var f0 = (3.0 - Math.sqrt(3.0))/6.0;  
var f1 = (1.0 - f0)*G2;  
var x0 = xin - x0;  
var y0 = yin - y0;  
var i1, j1;  
if(x0 > y0) {i1=1; j1=0;}  
else {i1=0; j1=1;}  
var x1 = x0 + i1 * G1;  
var y1 = y0 + j1 * G2;  
var t0 = 0.5 * (x0*x0 + y0*y0);  
if(t0 < 0) n0 = 0.0;  
else  
    t0 *= t0;  
n0 = t0 * t0 * this.dot(this.grad3[gi0], x0, y0);  
var t1 = 0.5 * (x1*x1 + y1*y1);  
if(t1 < 0) n1 = 0.0;  
else  
    t1 *= t1;  
n1 = t1 * t1 * this.dot(this.grad3[gi1], x1, y1);  
var t2 = 0.5 * (x2*x2 + y2*y2);  
if(t2 < 0) n2 = 0.0;  
else  
    t2 *= t2;
```

Promoteur interne :  
Mr Nicolas Sottiaux

Ludwig Dejonckheere

2020 - 2021



## REMERCIEMENTS

Je tiens à remercier avant tout Monsieur Nicolas Sottiaux, promoteur interne qui m'a assisté durant la réalisation de ce travail de fin d'études. Monsieur Sottiaux a réussi à m'orienter de la bonne manière et m'a redonné confiance lorsque j'ai pu douter de mes productions.

Je remercie également Monsieur Ivan Miller qui a su répondre à mes nombreuses questions concernant ce travail et qui m'a consacré un peu de son temps pour m'éclaircir sur mes interrogations.

Enfin, je remercie plus globalement la Haute École en Hainaut et la Section Techniques Graphiques, ainsi que le corps enseignant, qui m'ont fait découvrir de nouveaux horizons d'activités plaisants et passionnants.

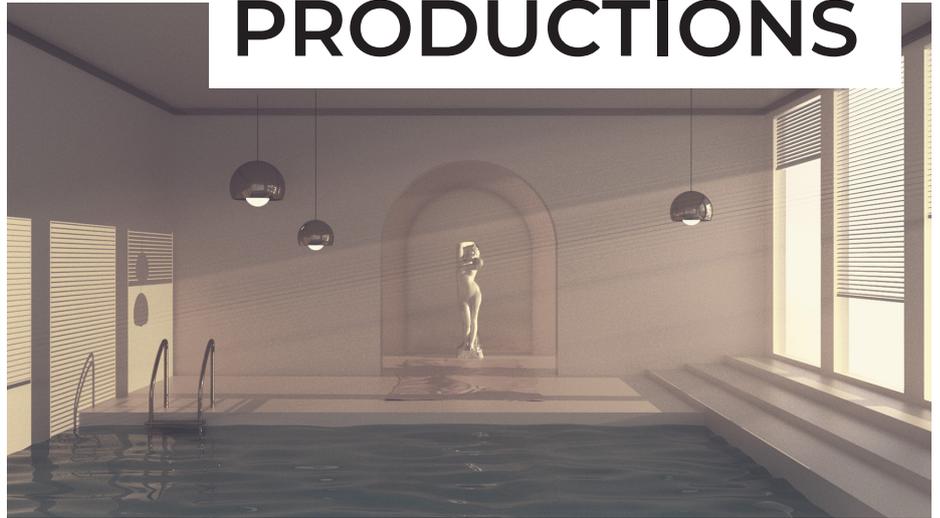


# INTRODUCTION

06

07

# OBJECTIFS ET PRODUCTIONS



# SOMMAIRE

## 10 CORPS DE TRAVAIL

PRÉAMBULE 11

Théorie 11

Logiciels 12

Paramètres du bruit dans Cinema4D 13

OUVRAGES PRATIQUES 18

Displacement 18

Texturing 33

Volume 39

Animation 41

Création du site 48

# RÉSULTATS PRATIQUES

## 62



**CONCLUSION**

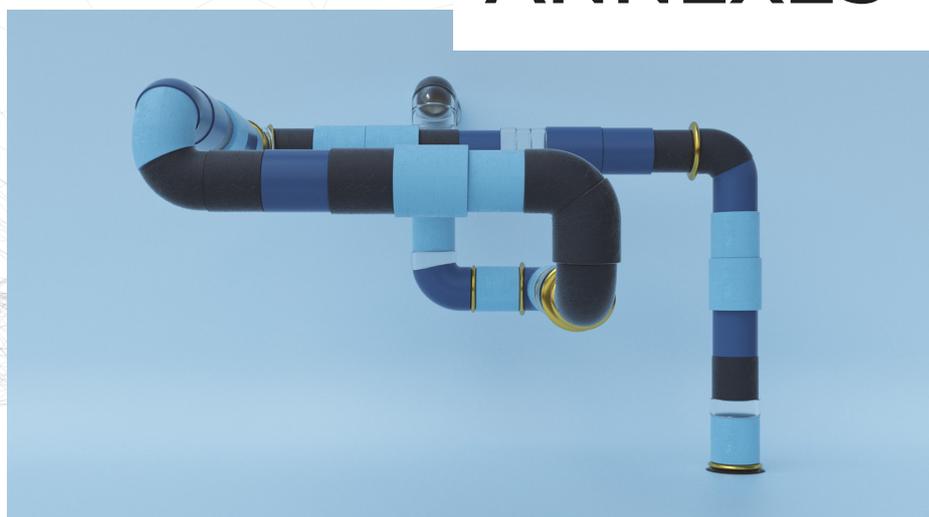
## 64

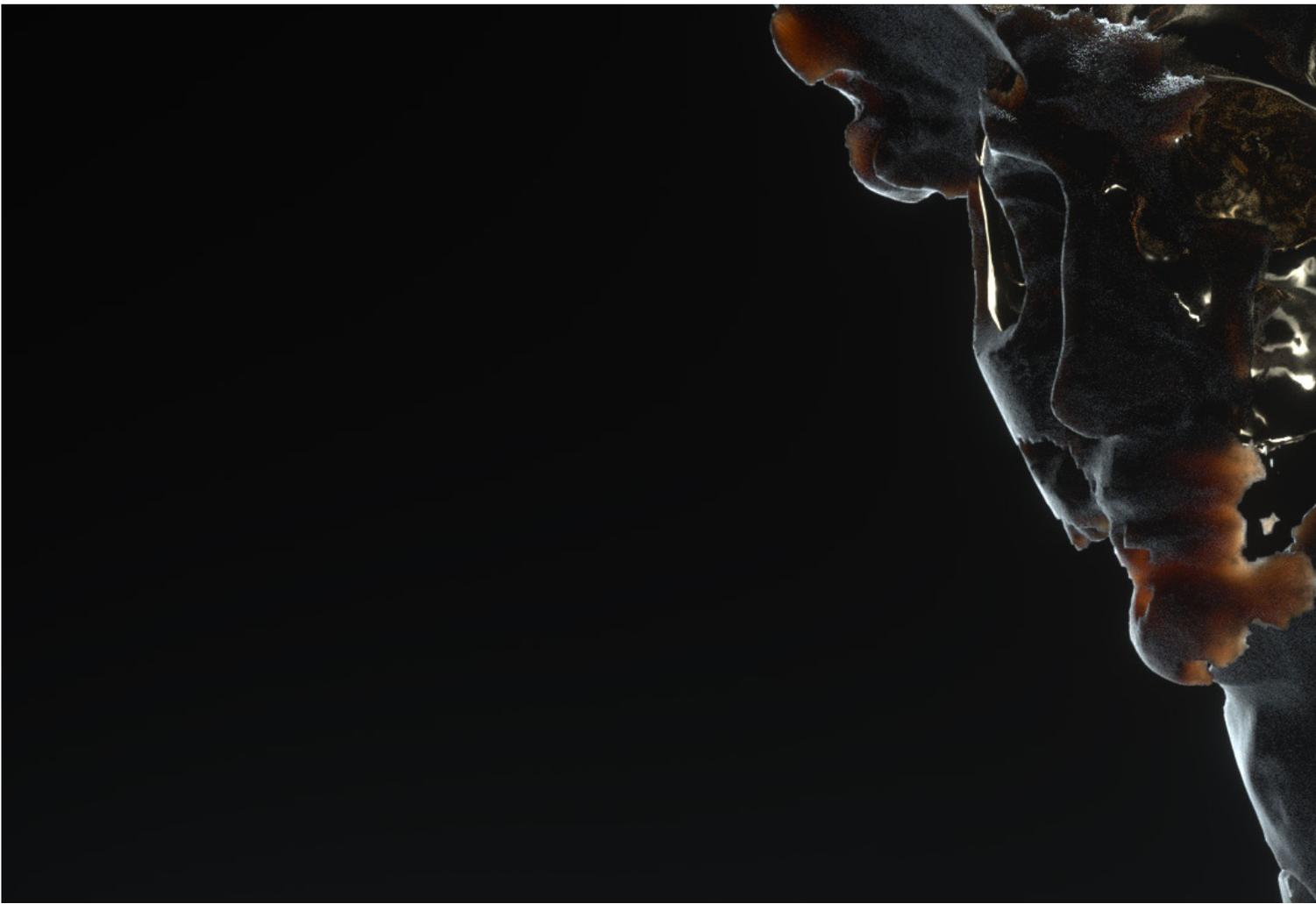
**BIBLIOGRAPHIE**

## 66

## 68

**ANNEXES**





# INTRODUCTION



L'infographie est un terme vaste qui regroupe énormément de compétences. On pense notamment à l'élaborations de logotypes, à la composition graphique, à la création de sites web ou encore de productions audiovisuelles.

Le domaine de la 3D, qui fait partie lui aussi du domaine infographique, est une matière dense qui évolue très vite. Il peut s'agir de dessins techniques et/ou industriels, de compositions graphiques ou encore d'animations. Qu'il s'agisse de motion design (« Motion Graphics » ou « MoGraph ») ou de réalisations pour le cinéma (effets spéciaux ou films d'animations 3D, par exemple).

Parmi toutes ces compétences abordant la 3D, un élément est désormais couramment utilisé pour diverses utilisations : le bruit.

Le bruit est relativement mal connu. Quand on l'évoque, on pense souvent à l'univers sonore. Lorsqu'on envisage le bruit visuel, on a alors tendance à penser au grain photographique. Dans le cadre de ce travail, nous nous concentrons sur un autre type de bruit : le bruit de Perlin (et ses dérivés).

# OBJECTIFS E PRODUCTION

L'objectif ici est d'expliquer et démontrer les différents usages du bruit dans le monde de la 3D. En commençant par le « déplacement », puis le « texturing » et enfin l'animation générée aléatoirement, en passant par la création de volumes organiques..

Si ces termes ne vous sont pas familiers, vous les comprendrez parfaitement en parcourant les différentes rubriques de ce rapport.

Enfin, un site spécifiquement dédié à la compréhension et l'utilisation du bruit en 3D est disponible. Ce dernier met en lumière les différentes compositions et œuvres animées élaborées durant ce travail de fin d'études, en incluant une partie pédagogique.

Ce site est avant tout destiné à un public familier à la 3D, plus spécifiquement à Cinema4D et Octane, et qui désire étendre ses connaissances en matière de composition 3D et de motion design.

**ET  
NS**





# **CORPS DE TRAVAIL**

# PRÉAMBULE

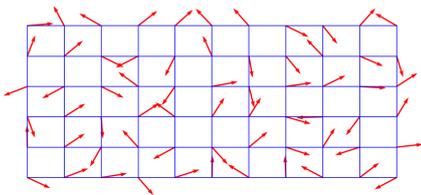
## THÉORIE

À l'origine, le bruit de Perlin est une texture procédurale développée par Ken Perlin en 1985, alors qu'il travaillait sur les effets spéciaux du film « Tron ». C'est un bruit de gradient, à ne pas confondre avec le bruit de valeur.

Sans entrer dans des détails trop techniques et en adoptant un langage vulgarisé, un bruit de gradient est un algorithme d'homogénéisation de plusieurs dégradés (= gradients). Ces derniers sont définis par des vecteurs, c'est-à-dire une sorte de flèche qui indique une direction et une intensité.

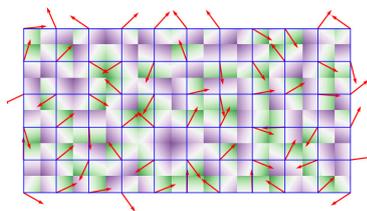
Il y a trois étapes dans la création d'un bruit de gradient :

### 01



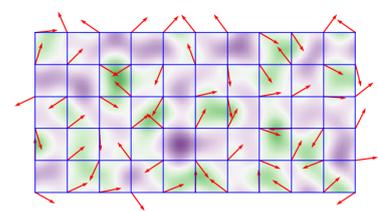
La première étape est la **définition d'une grille** à deux, trois, voire quatre dimensions. Prenons l'exemple d'une grille à deux dimensions. À chaque intersection des axes est attribué un vecteur de gradient aléatoire.

### 02



La seconde étape de l'algorithme est de calculer le **produit scalaire** de ces vecteurs (=produit des normes de 2 vecteurs par le cosinus de l'angle formé).

### 03



Enfin, la dernière étape est **l'interpolation** entre les produits scalaires calculés aux nœuds des cellules de la grille.

Le bruit apparaît alors concrètement.

Le bruit de Perlin génère une texture procédurale (c'est-à-dire définie par un algorithme) pseudo-aléatoire, dans le sens où elle fonctionne sur un système de « seed » (graine en français). Sans cette valeur, il ne peut pas y avoir de génération. Cette graine génère un signal qui est à l'origine du bruit. À valeur différente, génération du bruit différente et donc schéma différent.

Du fait que le bruit se base sur un algorithme mathématique, son grand avantage est qu'il est tout à fait manipulable et permet une infinité de possibilités.

Nous nous concentrerons ici sur les différents bruits que propose le logiciel Cinema4D qui ne compte pas moins de 25 bruits différents et modulables. Ces bruits sont des dérivés du bruit de Perlin : leurs algorithmes varient mais ils fonctionnent selon le même principe.



## LOGICIELS

Le premier logiciel utilisé ici est Cinema4D, un logiciel de création 3D développé par la firme Allemande Maxon en 1990. C'est un logiciel complet concernant la 3D puisqu'il comprend les outils de modélisations standards, des fonctionnalités de sculpture 3D, la création de matériaux, les outils nécessaires pour créer des rendus et enfin une bibliothèque exhaustive d'effets d'animations 3D : le « MoGraph ». Cinema4D est connu pour être adapté aux débutants se lançant dans le monde de l'animation 3D. Il est également tout à fait adapté et performant pour les motion designers professionnels.

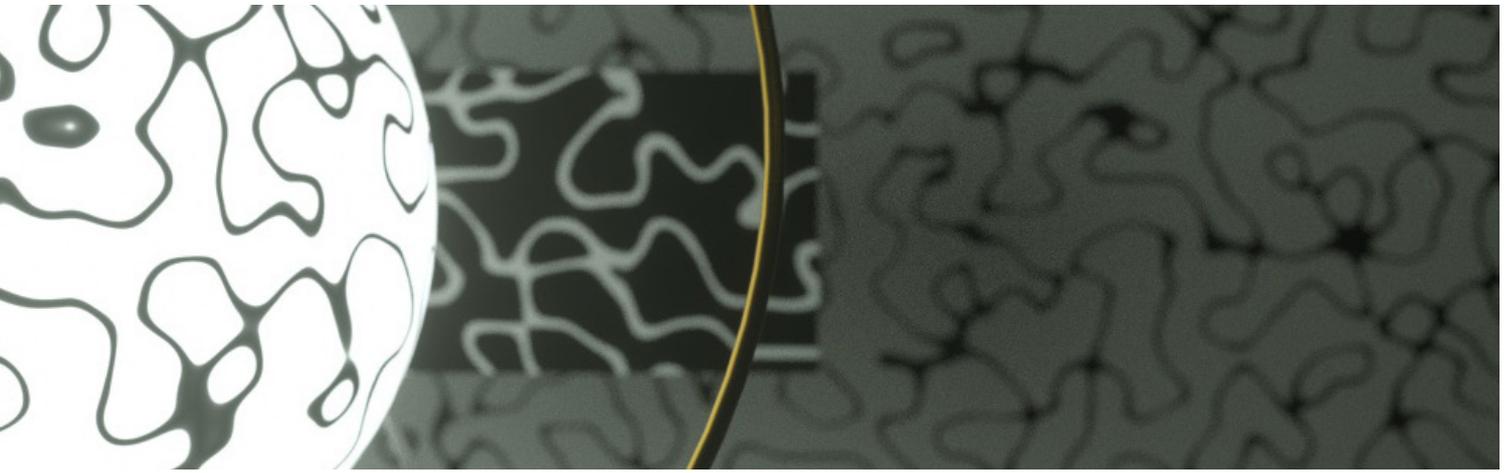
Le second logiciel est utilisé en tant que plugin de Cinema4D : Octane Render qui est un logiciel de rendu 3D développé en 2012 par la société néo-zélandaise OTOY. Il a plusieurs avantages qui font de lui un moteur de rendu très utilisé par les motion designers.

Premièrement, il s'agit d'un moteur de rendu en temps réel. Il n'y a donc pas besoin de lancer un rendu et de devoir attendre que celui-ci soit entièrement complété pour apprécier le résultat final. Ici, le rendu de la scène se fait directement dans une fenêtre séparée.

Deuxièmement, il utilise pleinement le processeur graphique (GPU), c'est-à-dire la puissance de calcul de la (des) carte(s) graphique(s) de l'ordinateur. Ce n'était pas le cas de la plupart des moteurs de rendu il y a encore quelques années. En effet, ces derniers s'appuyaient sur le processeur principal des ordinateurs, ce qui tendait à rendre le temps de rendu beaucoup plus long.

Ces deux facteurs sont déjà de bonnes raisons d'utiliser Octane, surtout si le processeur graphique est de dernière ou avant-dernière génération, étant donné leur puissance de calcul extraordinaire.

Enfin, parmi la palette d'avantages qu'offre Octane, il y a le mode « Pathtracing ». C'est une technique de « lancer de rayons » (ou « Ray Tracing » en anglais) qui détermine l'illumination de la scène 3D en fonction du rebond des rayons de lumières sur les différentes surfaces. L'image est d'abord un brouillard de pixels qui s'affine avec le temps. Cette technologie tend à créer des rendus très « réalistes ».

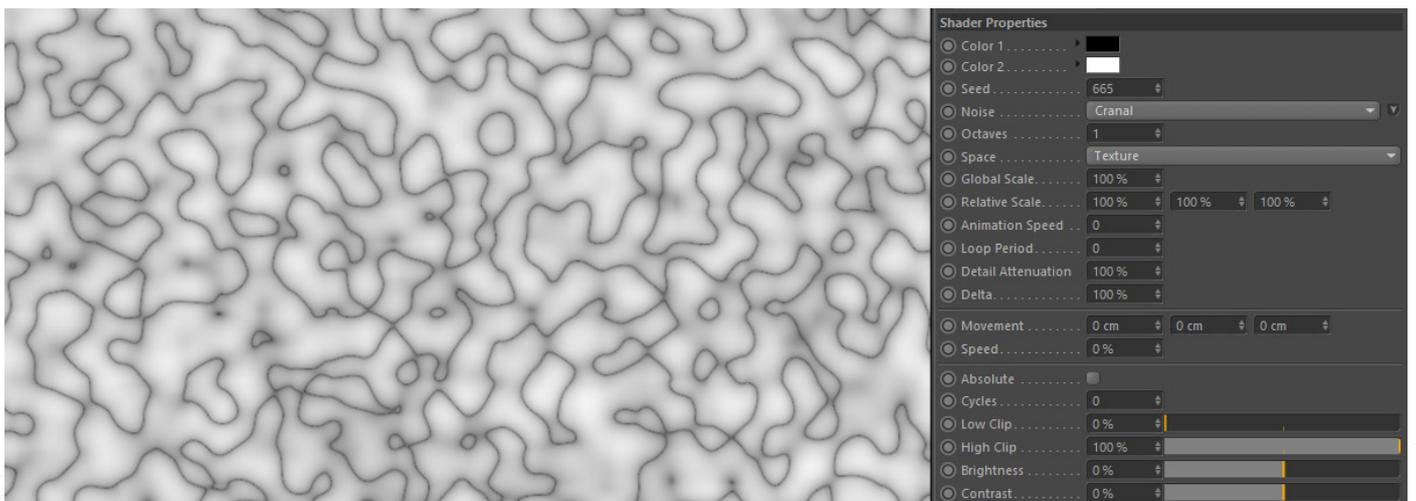


## PARAMÈTRES DU BRUIT DANS CINEMA4D

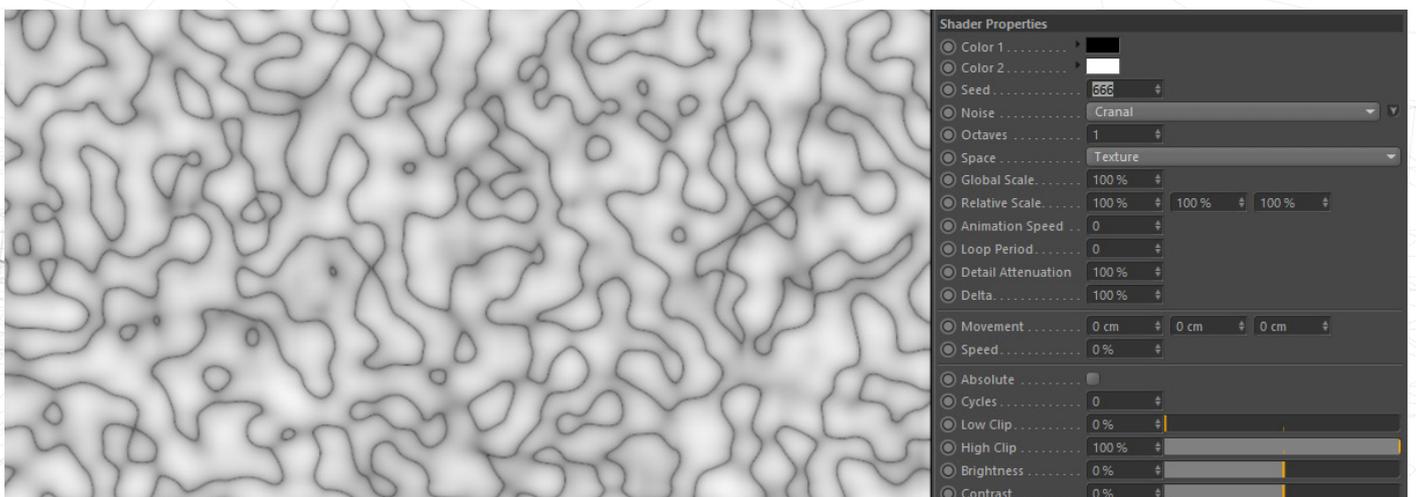
Avant de démontrer les applications du bruit, il est primordial de définir et comprendre les différents paramètres qui le composent. Ces paramètres sont les piliers qui définissent l'aspect même du bruit et qui caractérisent sa complexité.

Le bruit « Cranal » est ici pris comme exemple pour bien distinguer les nuances lorsqu'un paramètre est modifié.

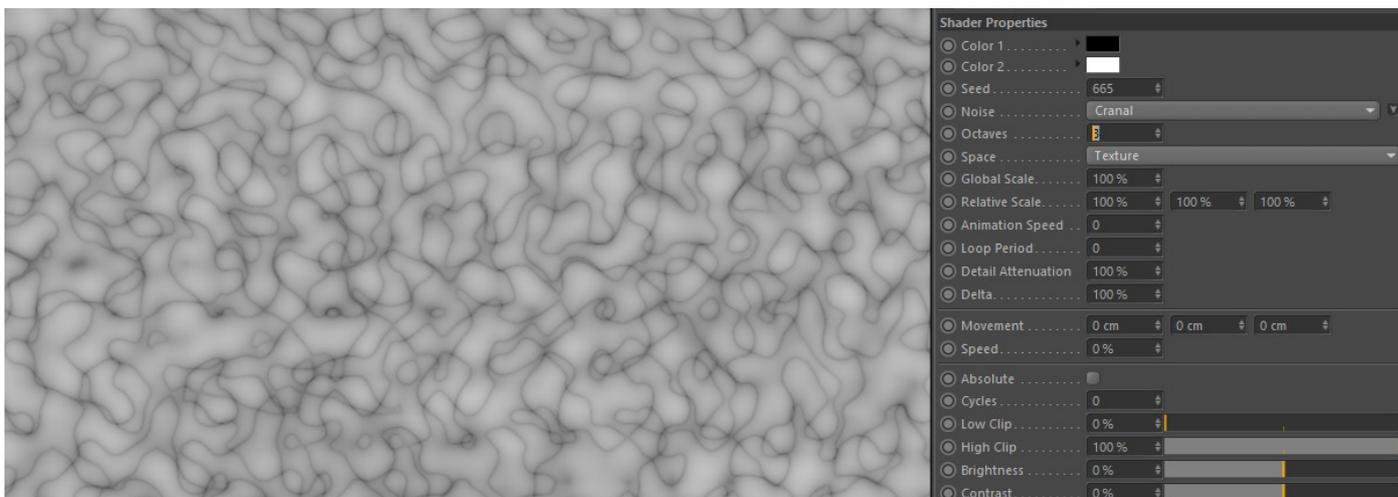
Voici l'aspect 2D qu'affiche le bruit par défaut, avec la fenêtre de paramètres se situant sur la droite.



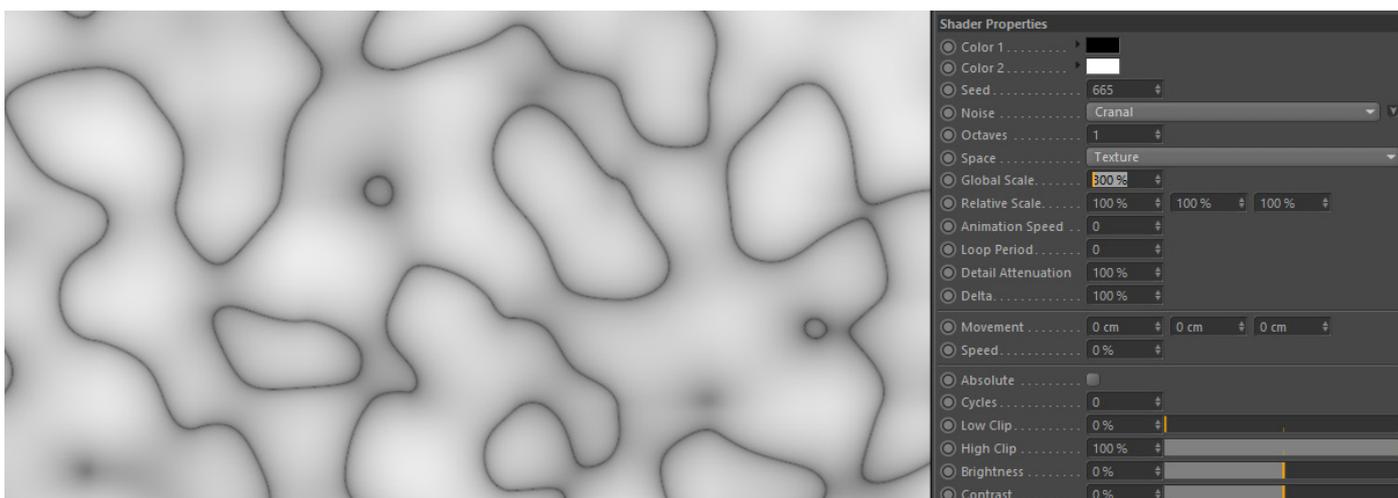
Outre les teintes des couleurs qui sont modifiables, le premier paramètre est la graine (« seed »). Comme énoncé en préambule dans la théorie, la graine est ce qui génère le bruit. Ce dernier présente une génération différente pour chaque graine.



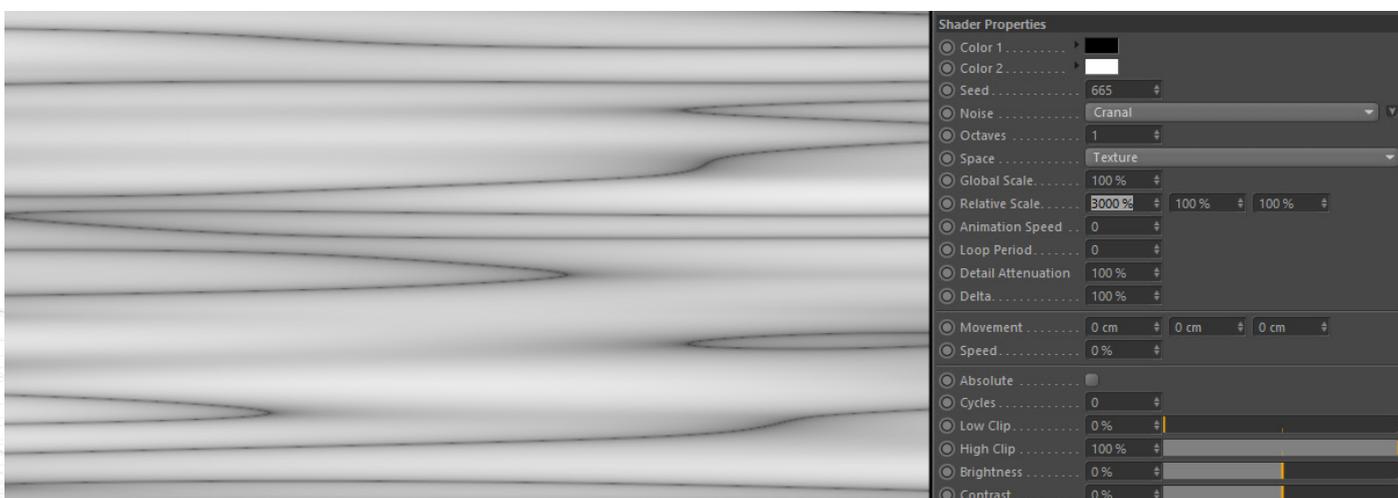
Le second levier sont les octaves (le « Noise » sera vu plus en détails). Elles jouent ici un rôle de complexification du bruit. Plus le nombre d'octaves est élevé, plus le bruit est détaillé, complexe.



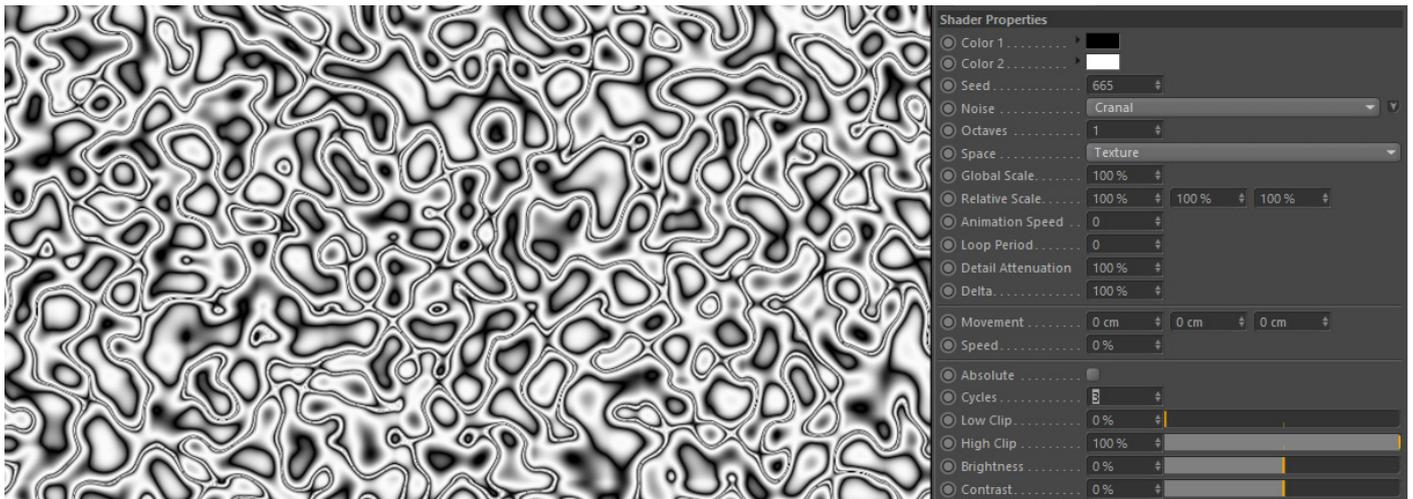
L'échelle globale est le paramètre suivant. Il ne joue ici que le rôle d'échelle standard. Le bruit peut être agrandi ou rétréci.



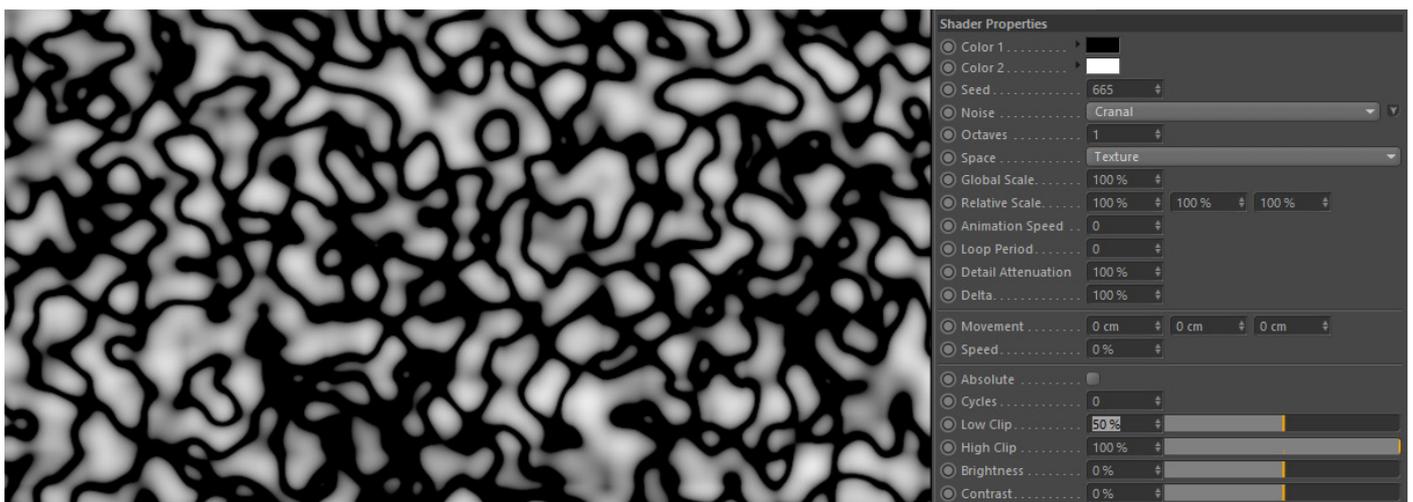
Les échelles relatives aux axes 3D que sont X, Y et Z suivent l'échelle globale. Puisqu'il s'agit ici d'une texture en deux dimensions, un des axes n'est pas effectif.



Le paramètre suivant intéressant est ici le cycle. Le cycle entraîne une sorte de répétition du bruit.



Viennent ensuite les paramètres de « Low Clip » et « High Clip » qui agissent un peu à la manière du gamma. En d'autres termes, le clip bas étend ici la zone noire et le clip haut étend la zone blanche. Si la valeur est poussée à son maximum, le bruit devient finalement soit complètement noir soit totalement blanc, en fonction du paramètre choisi.

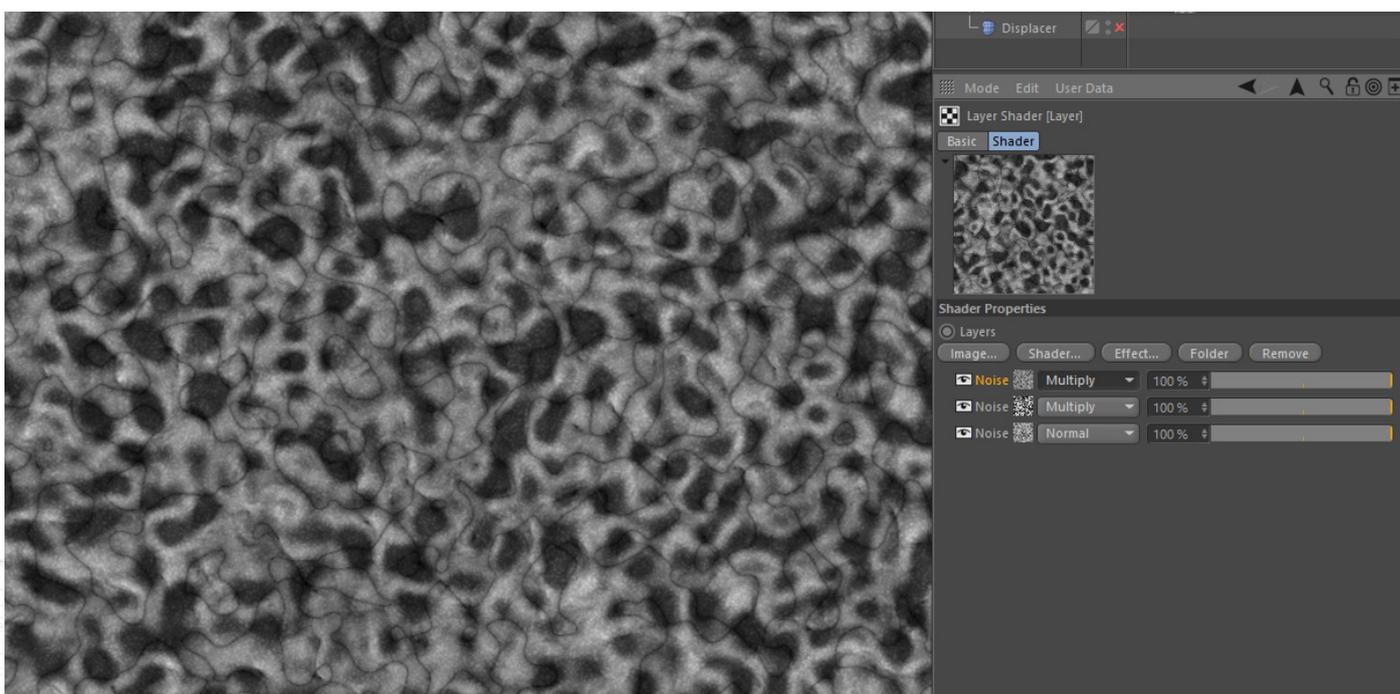
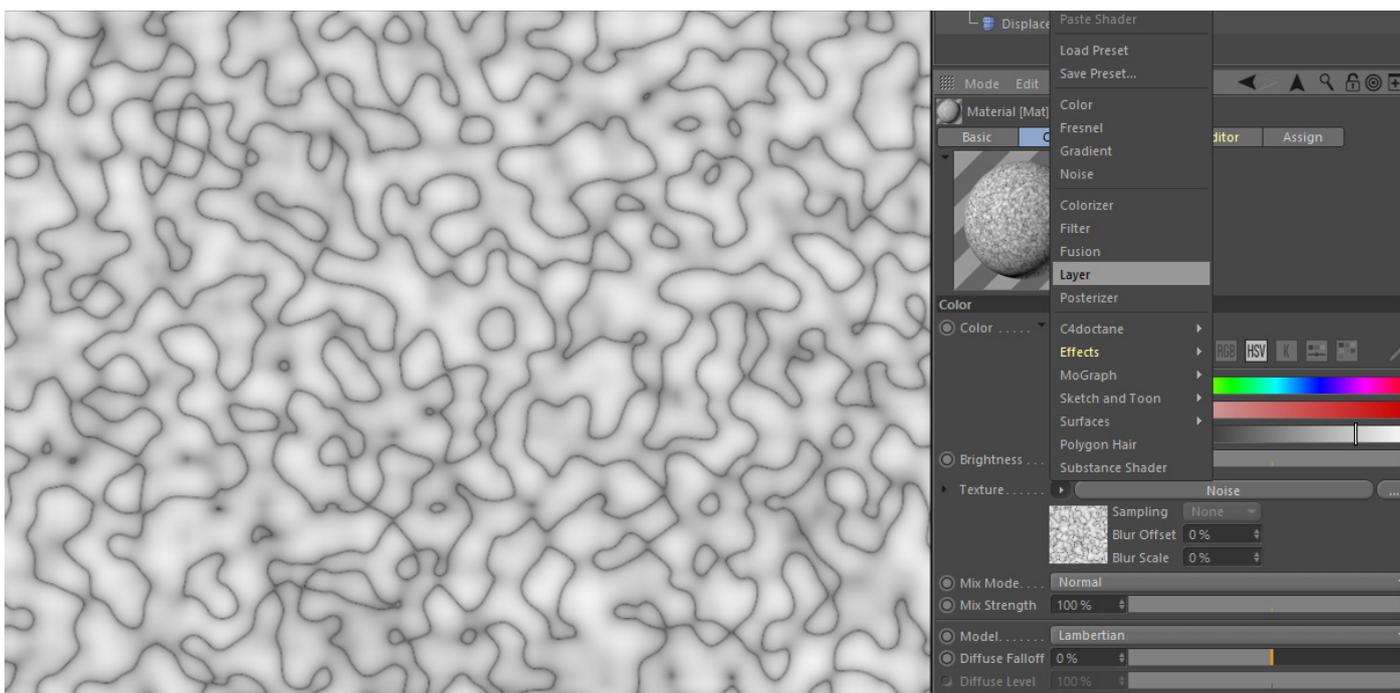


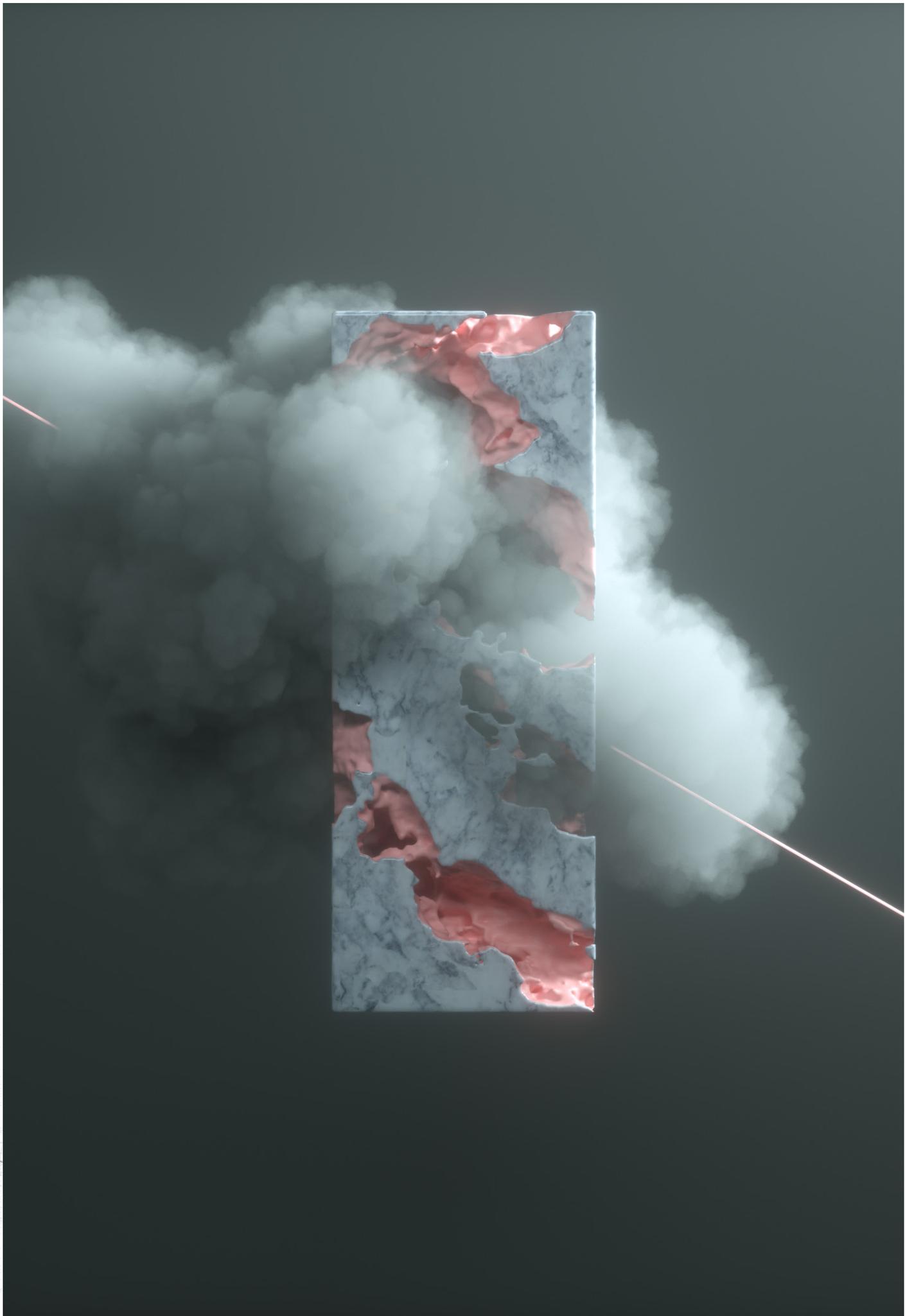
Enfin, les derniers leviers utiles sont ici la luminosité et le contraste, avec la particularité de pouvoir encoder des valeurs négatives, à la manière de Photoshop.



Une fois que tous ces facteurs sont déterminés, le bruit se présente alors comme une palette infinie de formes générées aléatoirement, modulables et surprenantes.

À noter qu'il ne s'agit ici que de simples démonstrations des paramètres et ce, sur un seul bruit parmi une gamme de 25 bruits différents proposés par Cinema4D. De plus, tout comme au sein du logiciel Photoshop, ces bruits peuvent être utilisés sous forme de calques qui sont alors additionnables avec des modes de fusions.





# OUVRAGES PRATIQUES

## DISPLACEMENT

### Qu'est-ce que le déplacement ?

Le « déplacement », littéralement « déplacement » en français, est l'action de générer du volume sur base d'une texture 2D. Ce volume prend forme en fonction de tons noir et blanc que composent la texture. Plus la teinte de la texture se rapproche du blanc, plus le déplacement est à son maximum. A contrario, plus la teinte se rapproche du noir, plus le déplacement ne prend pas effet.

Il peut s'agir simplement d'une couleur unie, d'un dégradé, d'une image, etc. Dans ce cas-ci, il est question d'appliquer un bruit en texture.

### Distinction entre Cinema4D et Octane

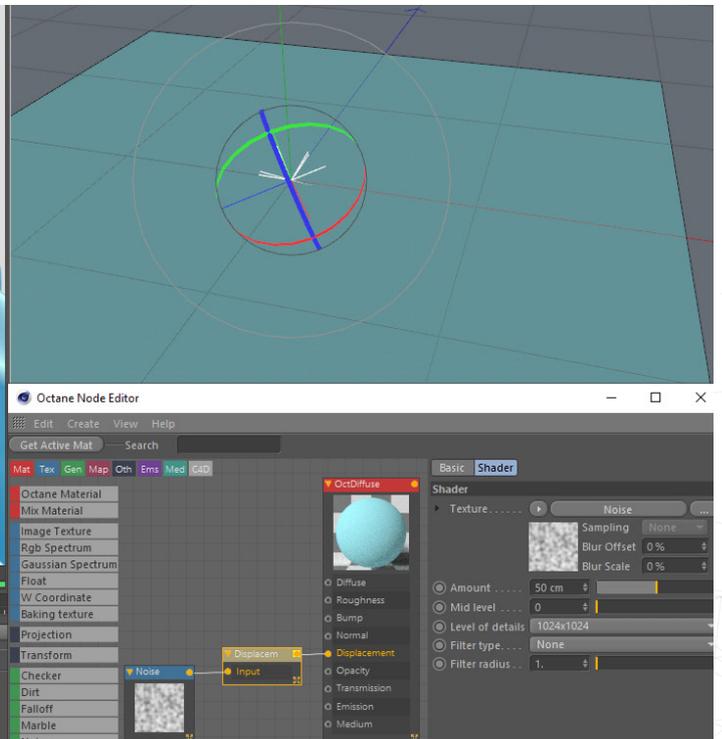
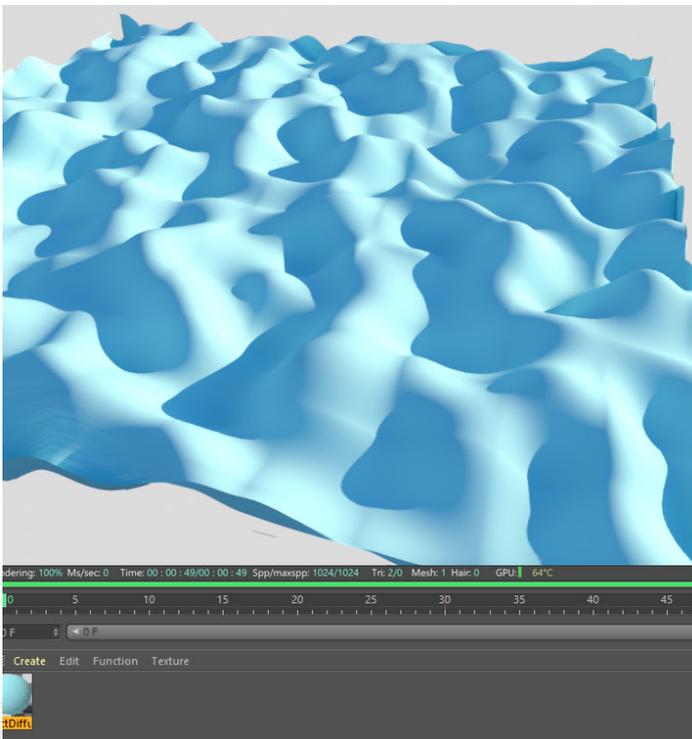
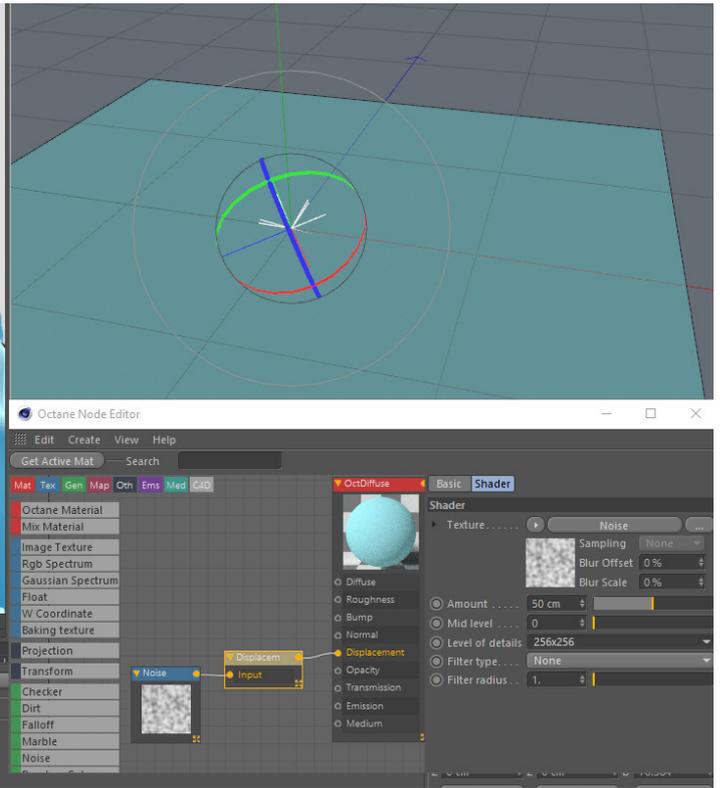
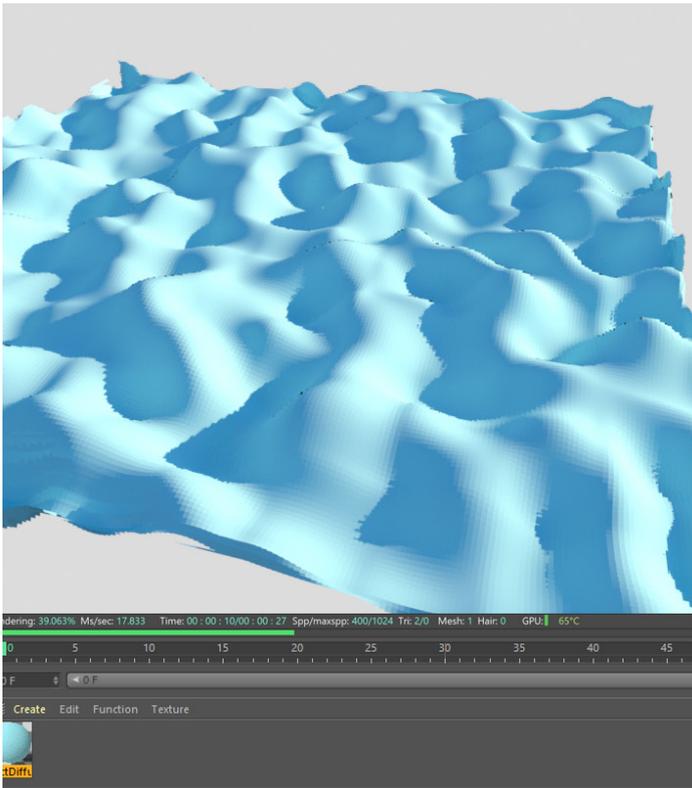
Cinema4D propose son lot de bruits, la plupart du temps utilisés dans un déformeur de géométrie (« déplacer ») lorsqu'il s'agit de lui donner un volume aléatoire en fonction du bruit. Cependant, le bruit de Cinema4D est aussi utilisable lors de la création d'un matériau Octane.

La différence ici est que si la déformation est établie directement dans le matériau via le calque de déformation, c'est la mémoire graphique qui travaille, et qui tend à ralentir le logiciel.

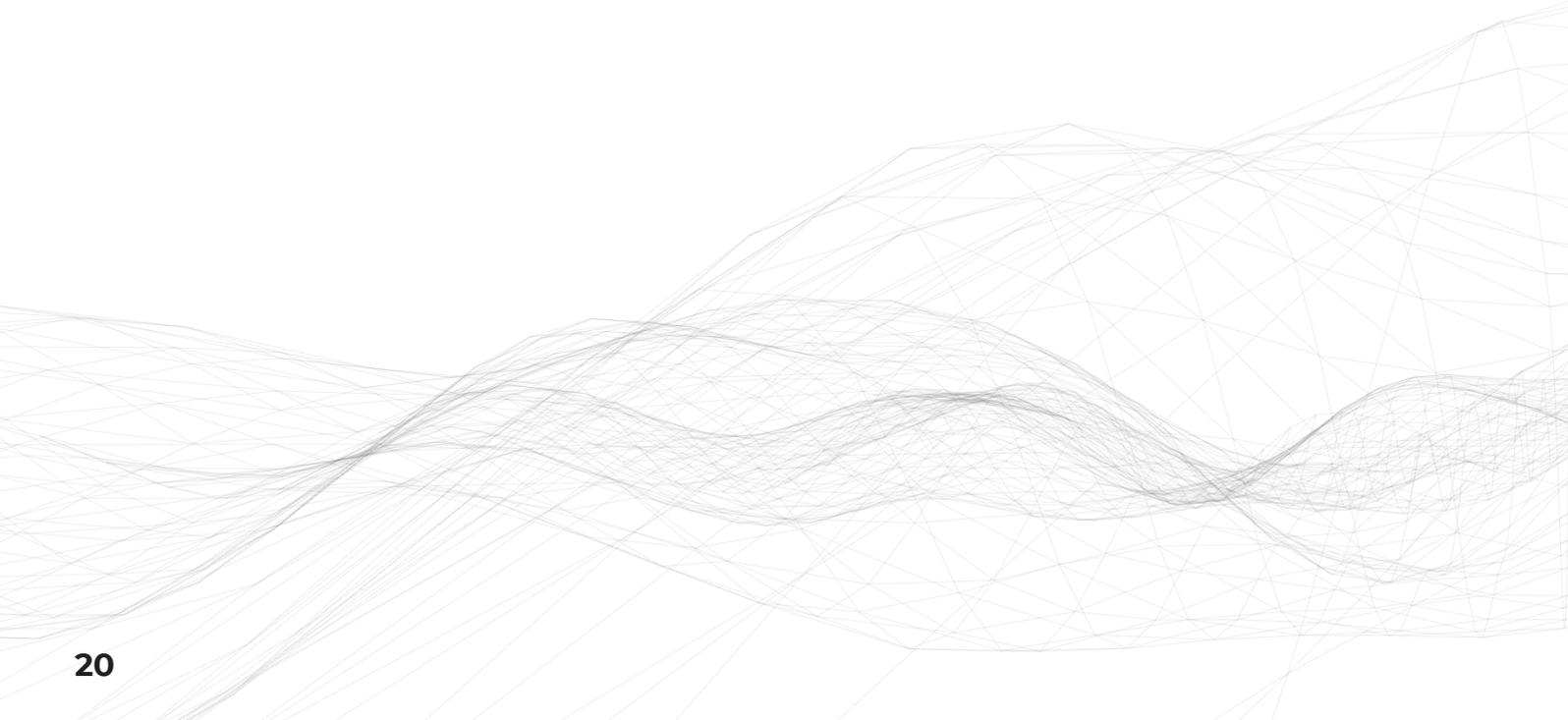
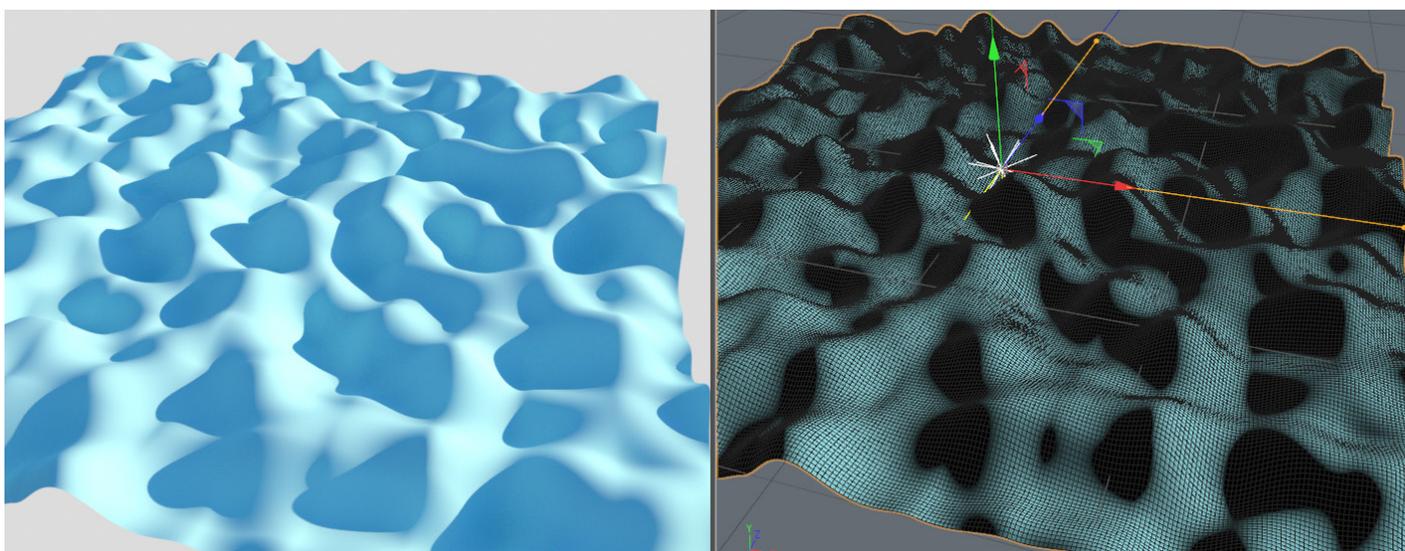
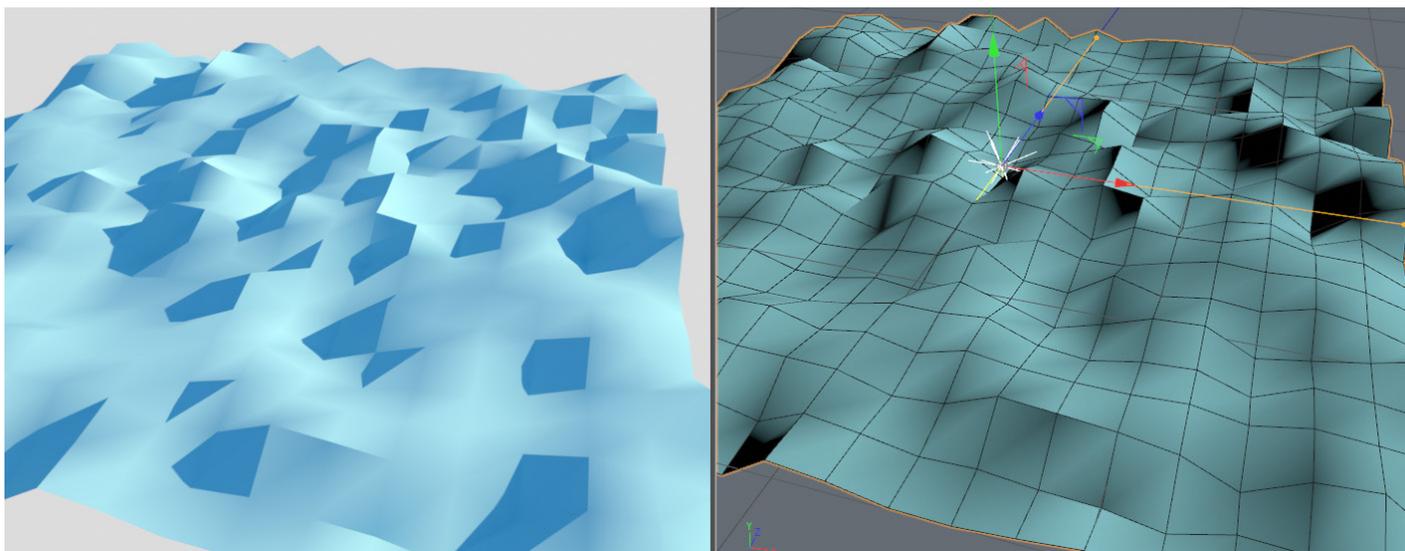
De plus, le déplacement est considéré comme matériau et non comme une simple géométrie. Si ce même matériau est appliqué à différents objets 3D, alors ils subiront tous la déformation suivant le matériau.

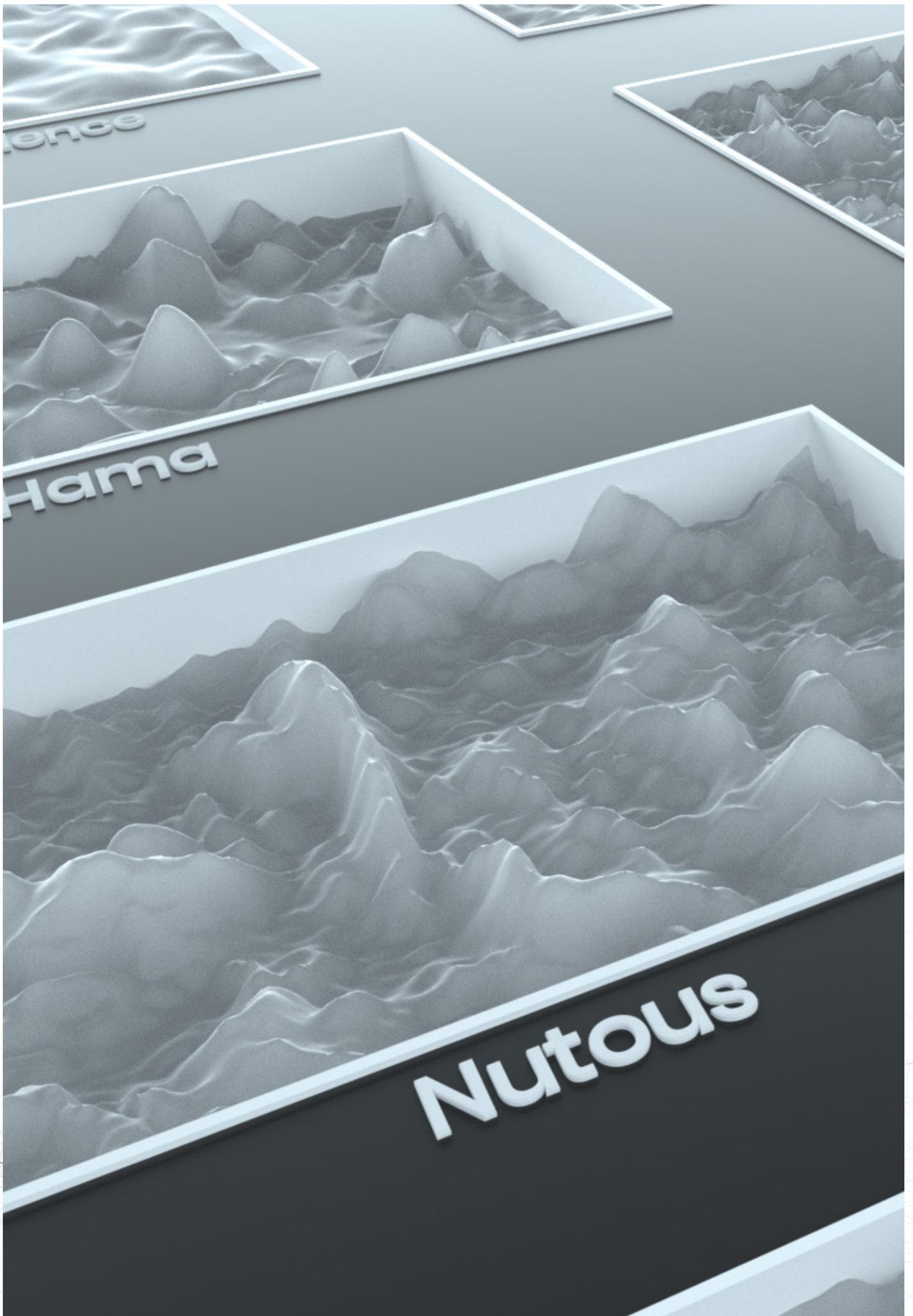
Néanmoins, la déformation sera de qualité supérieure si le bruit est inséré directement dans le matériau. Ceci s'explique par le fait que plusieurs définitions du bruit sont proposées au sein du matériau. Il peut très bien être de moindre qualité en 256x256 px, comme être très précis en 8192x8192 px. Plus la définition est élevée, plus le matériau occupera de la mémoire graphique et plus le logiciel sera lent. À noter que comme il est question ici de matériau, la géométrie propre à l'objet ne change pas, c'est bien dans le rendu que la déformation a lieu.





D'un autre côté, si la déformation est établie par un « displacer » de Cinema4D, le détail sera défini par les subdivisions que compose la géométrie. Plus cette dernière présente de subdivisions, plus le bruit est affiné et net.



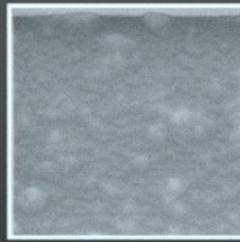




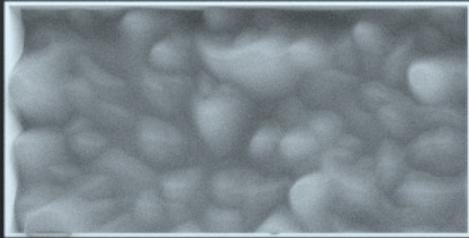
*Box Noise*



*Blistered Turbulence*



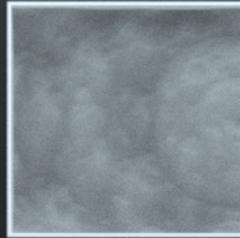
*Bu*



*Dents*



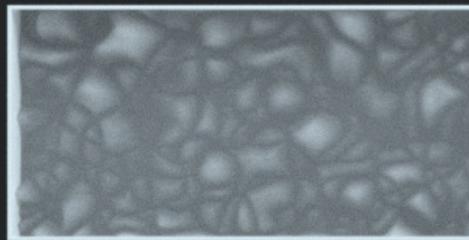
*Displaced Turbulence*



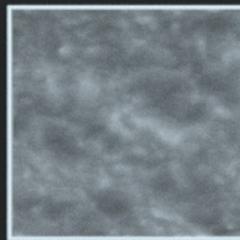
*Elec*



*Gaseous*



*Hama*



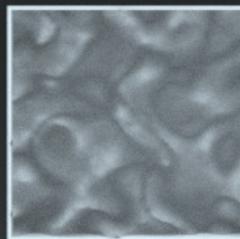
*Lu*



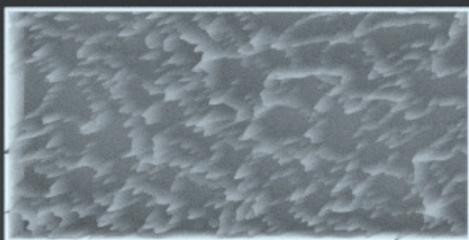
*Noise*



*Nutous*



*Ob*



*Sema*



*Stupl*

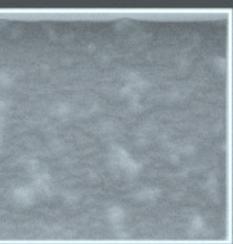


*Turbu*

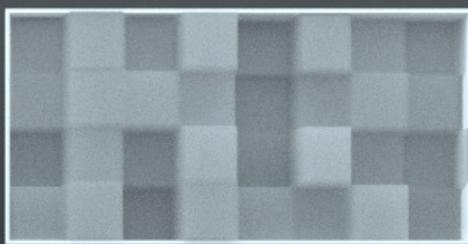
## Nuancier

Annexes p.68

La première chose à entreprendre pour bien visualiser les différents bruits de Cinema4D est de réaliser un nuancier de ceux-ci. C'est une façon concrète de représenter visuellement la palette de bruits qu'offre le logiciel.



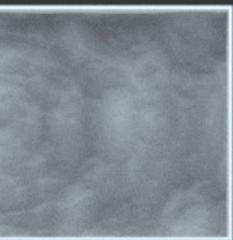
ya



Cell Noise



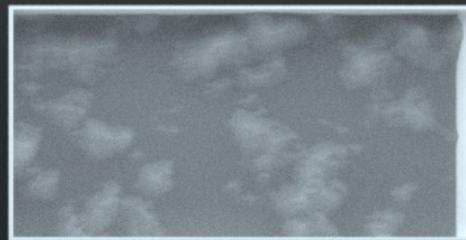
Cranal



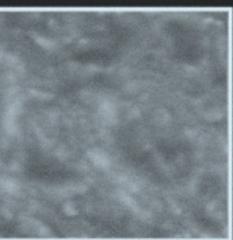
etric



FBM



Fire



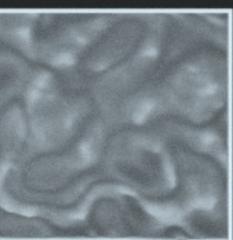
ka



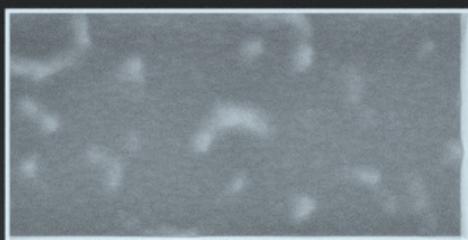
Mod Noise



Naki



er



Pezo



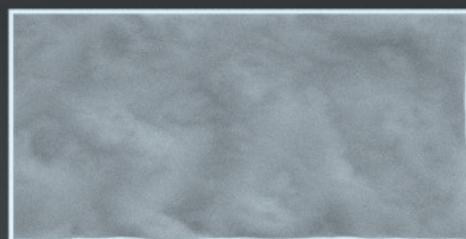
Poxo



lence



VL Noise

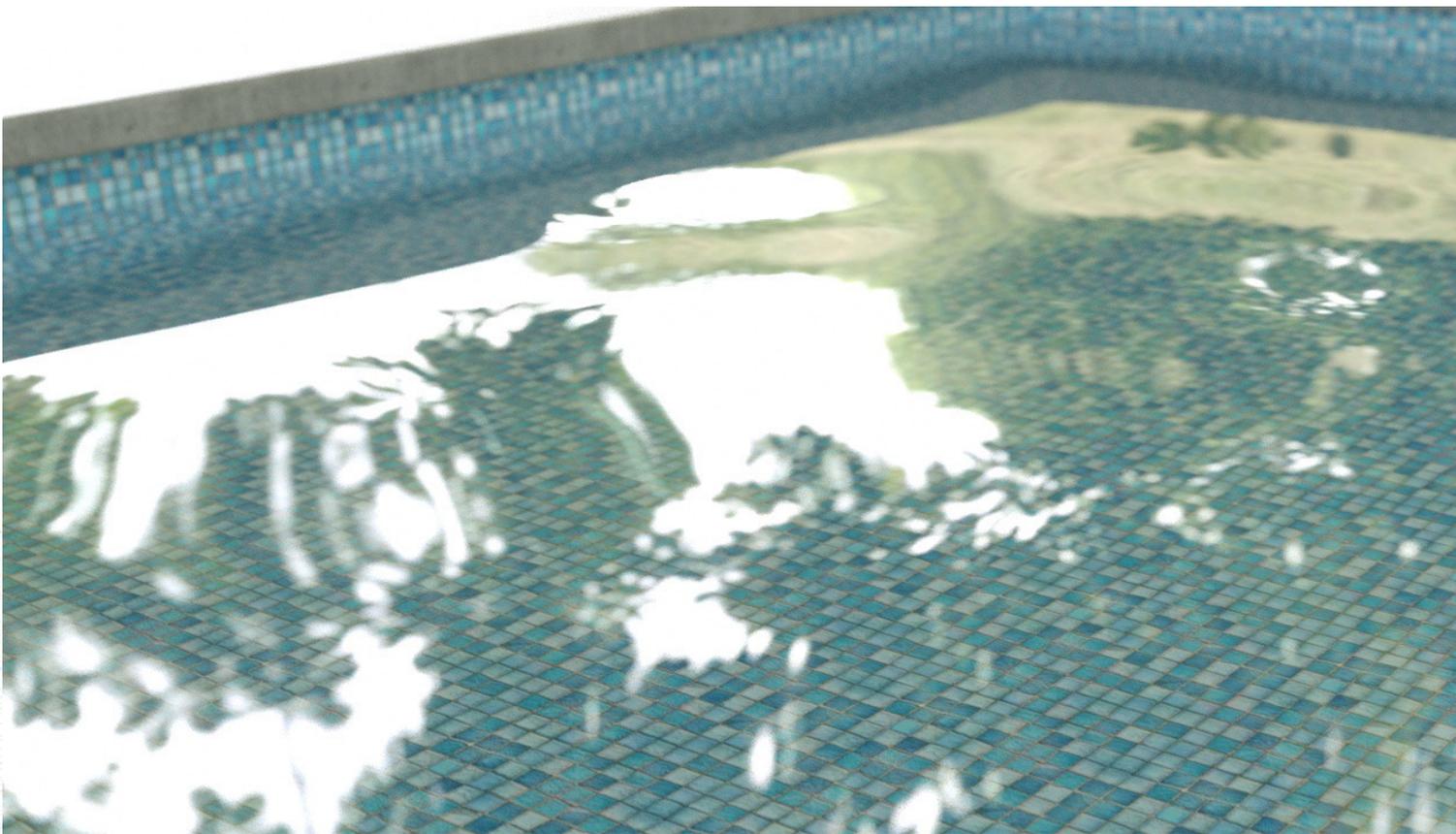


Wavy Turbulence

Si certains bruits présentent des similitudes, c'est dans leurs mouvements qu'ils se distinguent. En effet, le « Cell Noise » semble ne pas être si différent du « Mod Noise » aux premiers abords. C'est seulement en voyant leurs mouvements générés aléatoirement que la nuance est perçue.

Animation Cell Noise : [https://www.ludwigdejonckheere.com/noise/assets/nuancier/cell\\_noise.mp4](https://www.ludwigdejonckheere.com/noise/assets/nuancier/cell_noise.mp4)

Animation Mod Noise : [https://www.ludwigdejonckheere.com/noise/assets/nuancier/mod\\_noise.mp4](https://www.ludwigdejonckheere.com/noise/assets/nuancier/mod_noise.mp4)

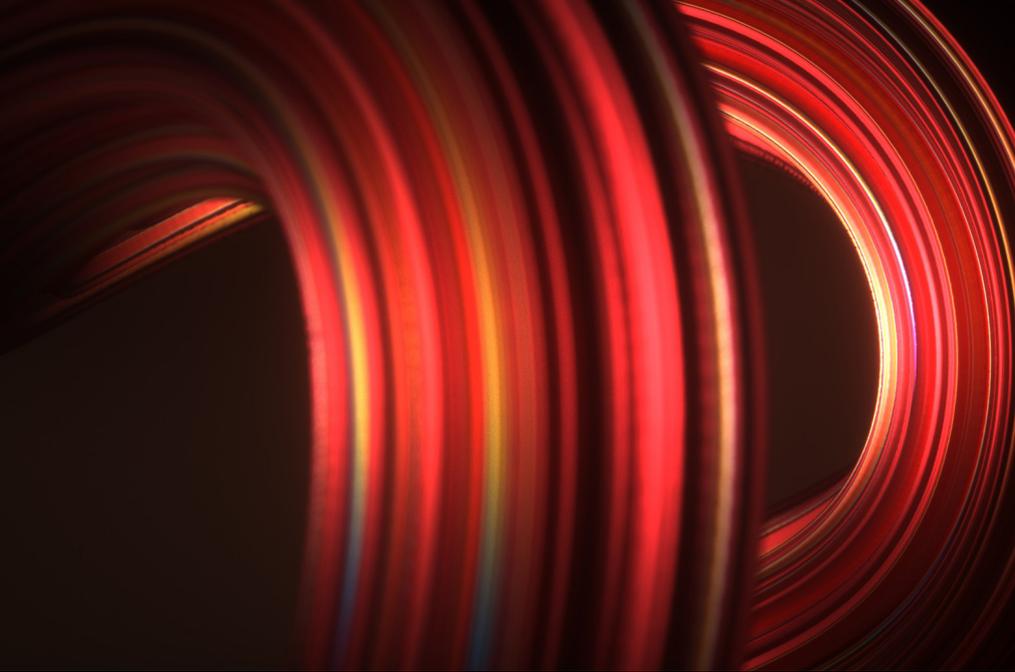




## **Rendu d'eau**

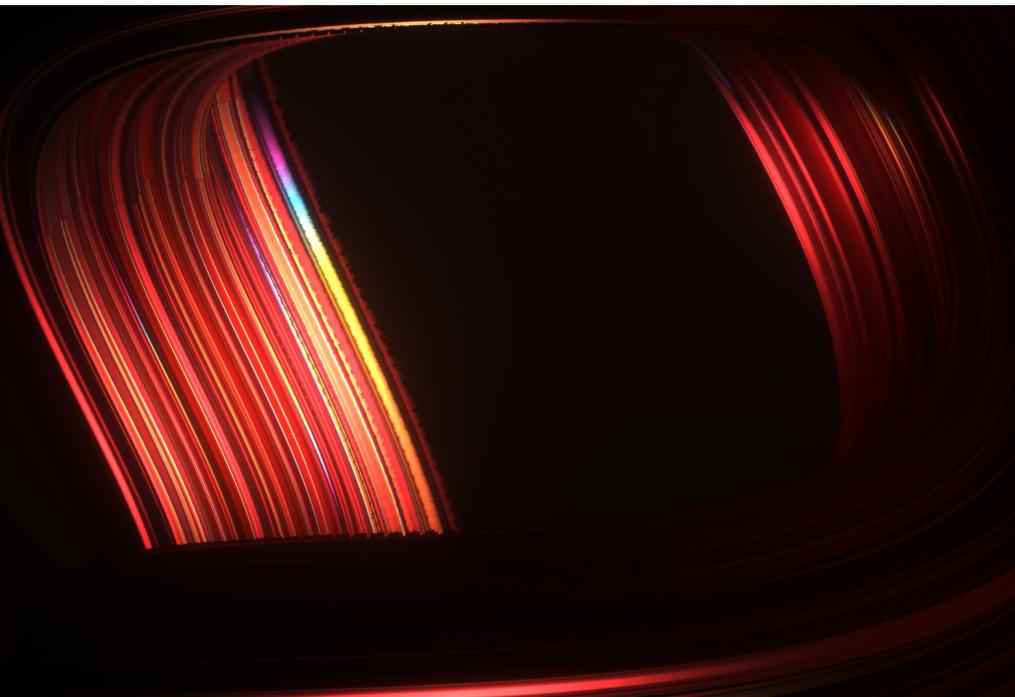
Annexes p.70

Un des usages utiles avec un bruit en tant que déformeur est le rendu de l'eau. En effet, il est très simple et rapide de créer une surface simulant un relief d'étendue d'eau, telle une piscine. Cet usage peut s'avérer intéressant dans le rendu architectural ou encore d'environnements naturels. De plus, cette méthode fonctionne aussi s'il s'agit d'une animation, en modifiant alors le paramètre de vitesse du bruit.



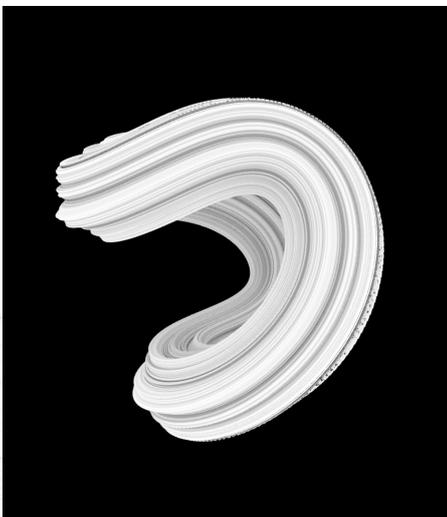
## Échelle Relative

Annexes p.71

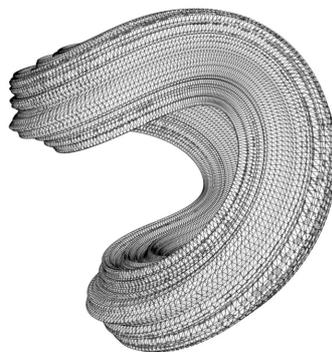


Lorsque le bruit est agrandi ou rétréci de manière relative selon les axes, il apparaît alors distendu et écrasé. Cet aspect rend alors le bruit comme étant allongé et présente des lignes droites. Ceci permet alors de créer des sillons le long d'un objet 3D.

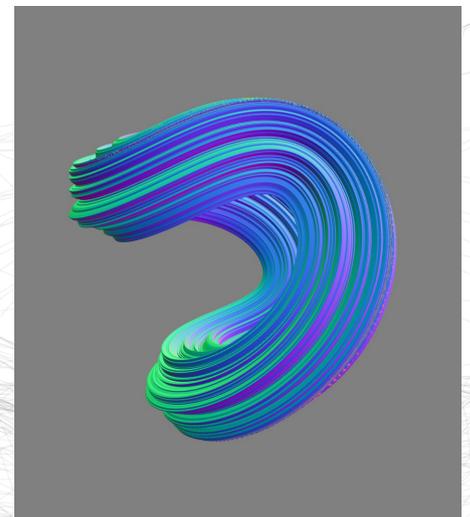
Dans cet exemple, il s'agit d'un torus, une sorte d'anneau, où un « bend » lui est appliqué premièrement. Le bruit distendu lui est ensuite appliqué pour lui donner cet aspect irrégulier sur sa longueur. Pour obtenir cet aspect, l'échelle relative à l'axe Y a été augmentée de 10 000%.



*Ambient Occlusion*



*Wireframe*



*Normals*

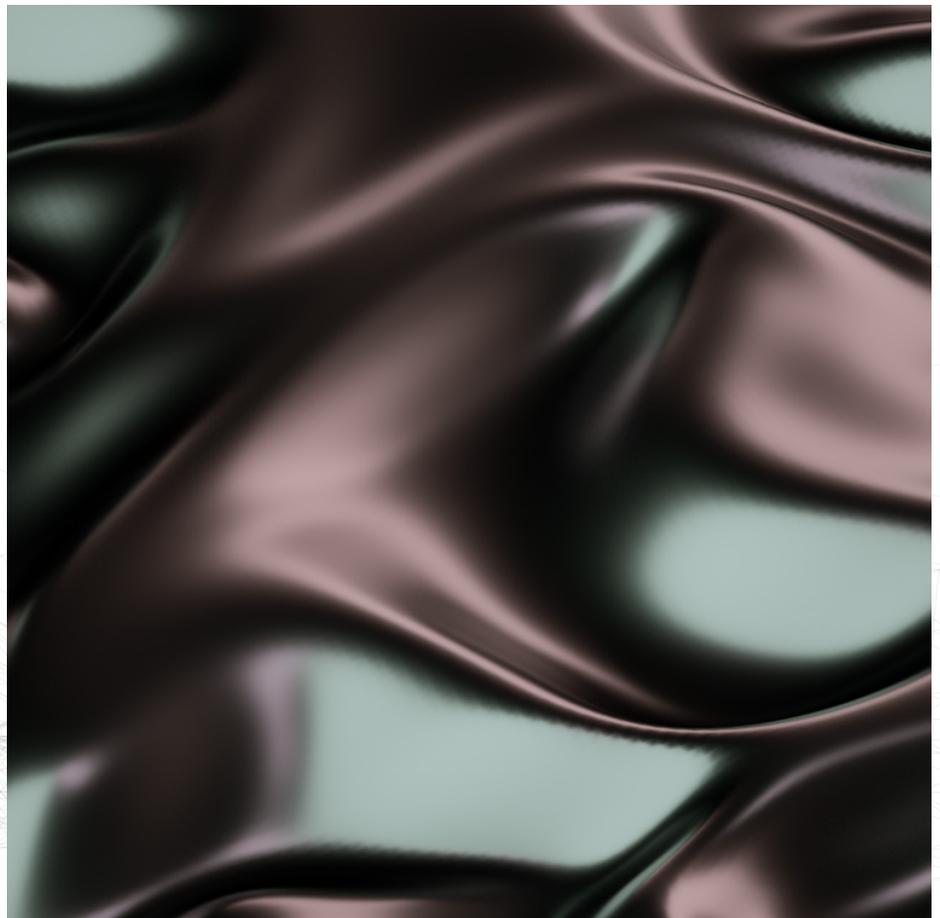
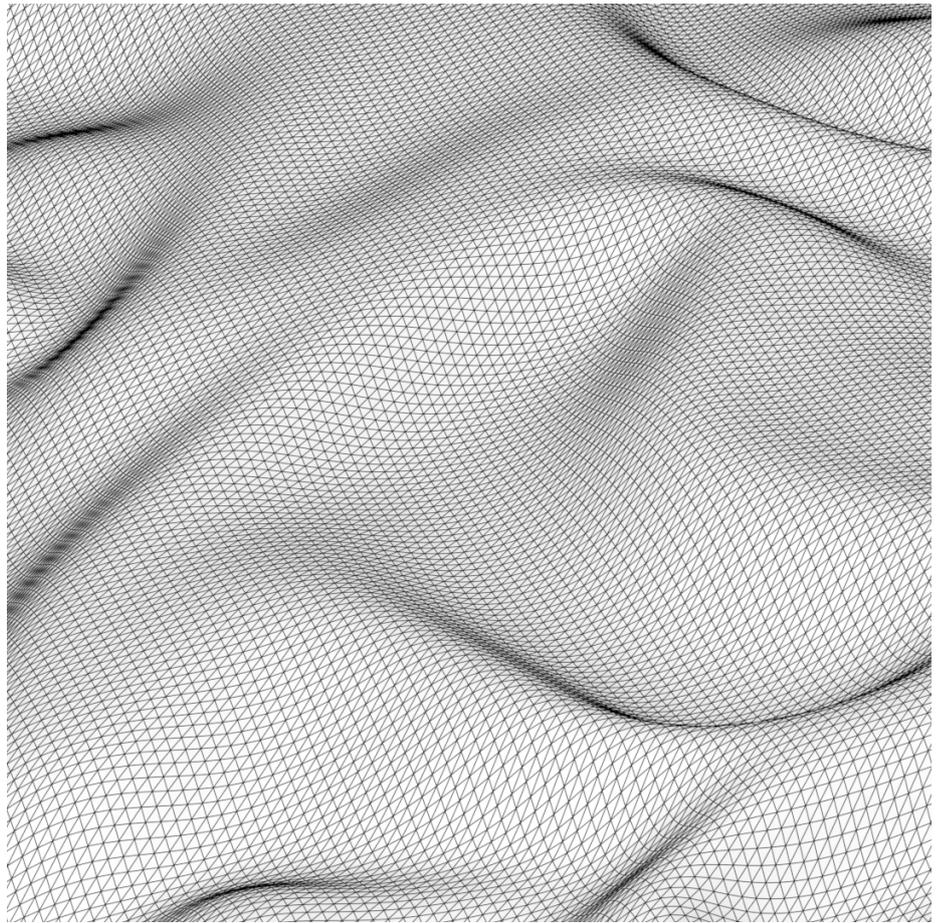
## Iridescence

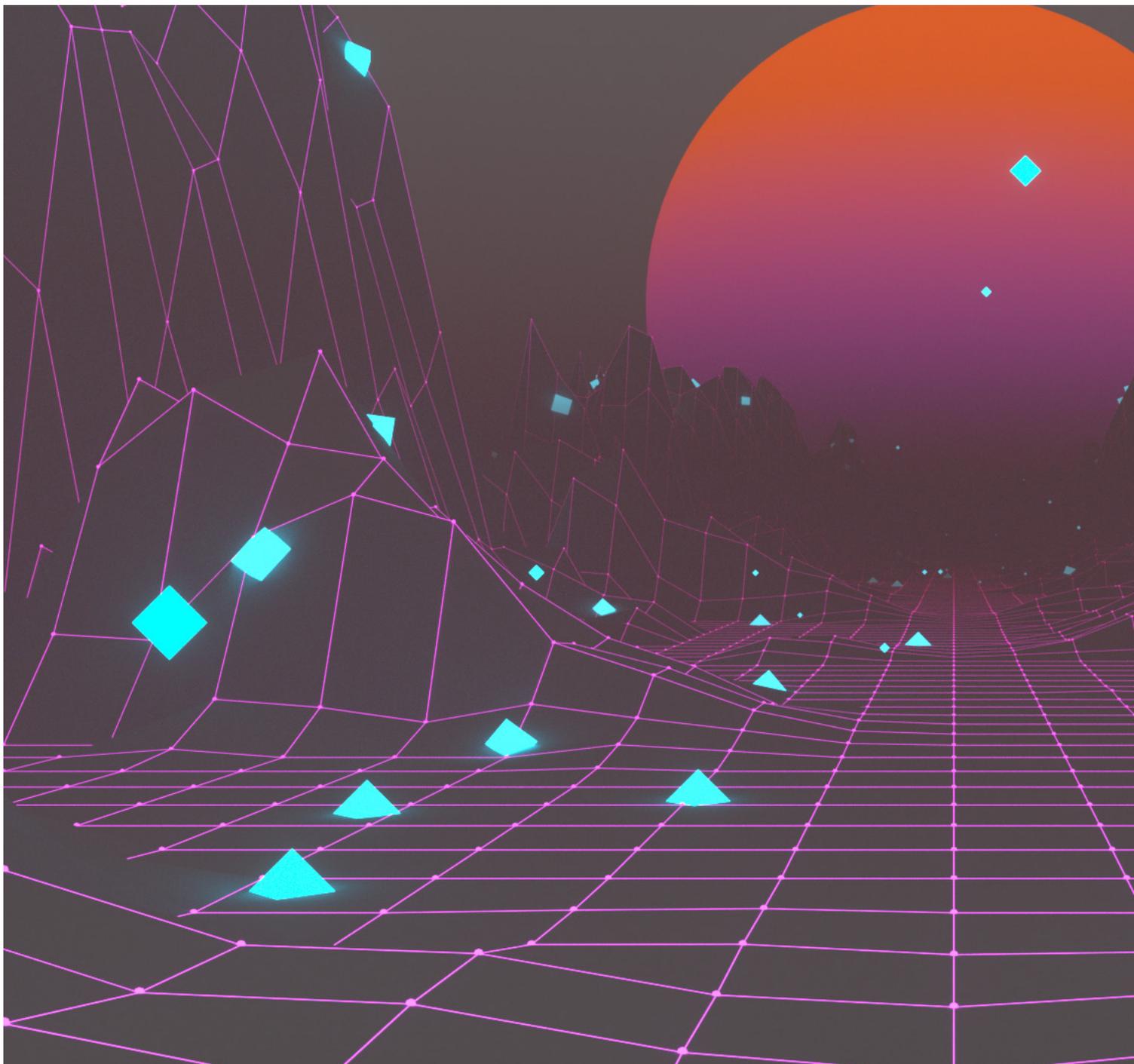
Annexes p.72

Une autre application, similaire à celle du rendu d'eau, est d'utiliser le bruit en déplacer de manière animée sur une surface plane. En lui attribuant un matériau réfléchissant à la manière du métal, la surface va réfléchir les lumières de différentes couleurs en prenant un aspect bariolé.

Pour permettre une géométrie fluide sans artéfacts, la surface plane contient un nombre important de subdivisions.

Animation Iridescence :  
<https://www.ludwigdejonckheere.com/noise/assets/iridescence/iridescent.mp4>

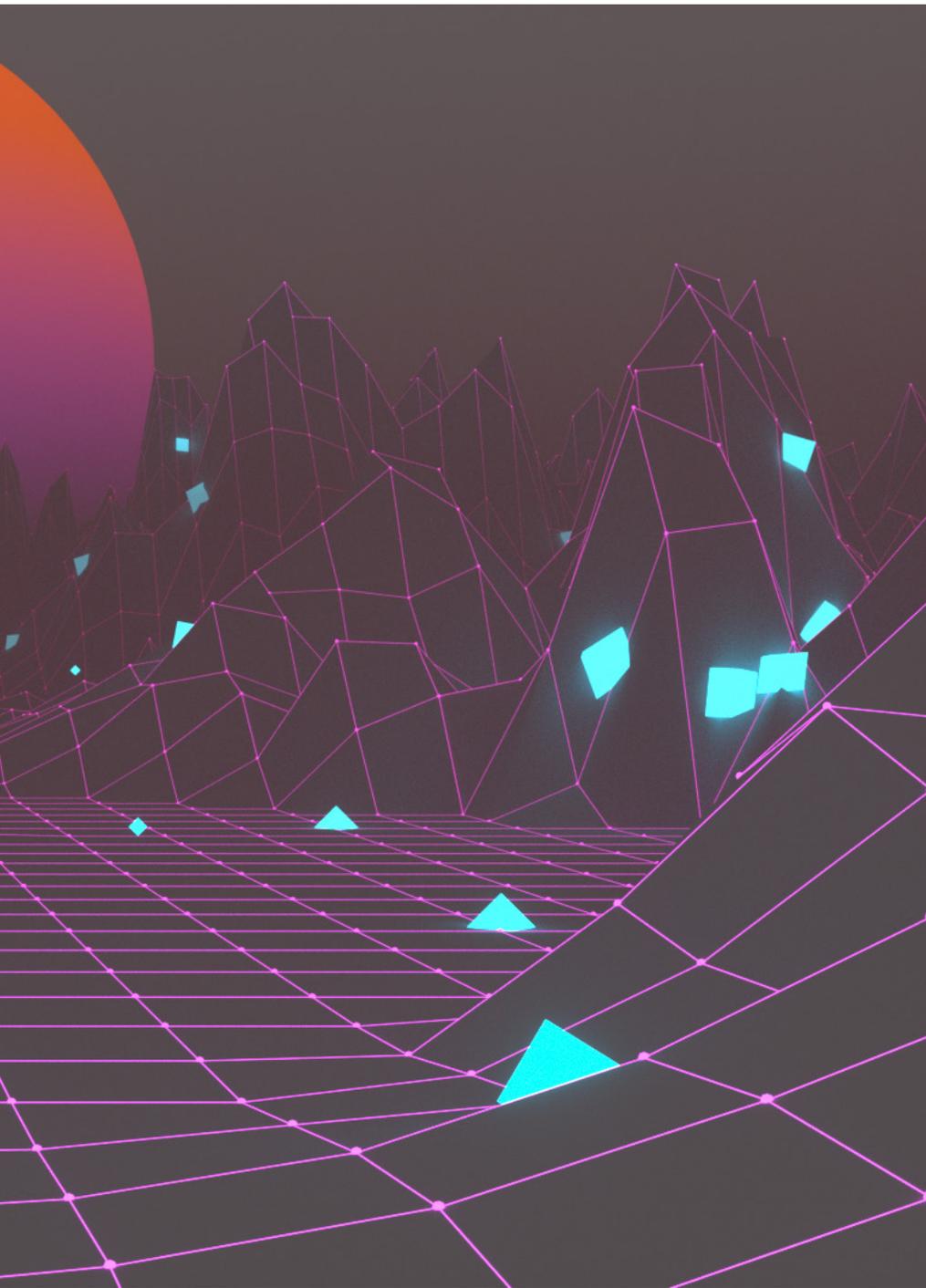




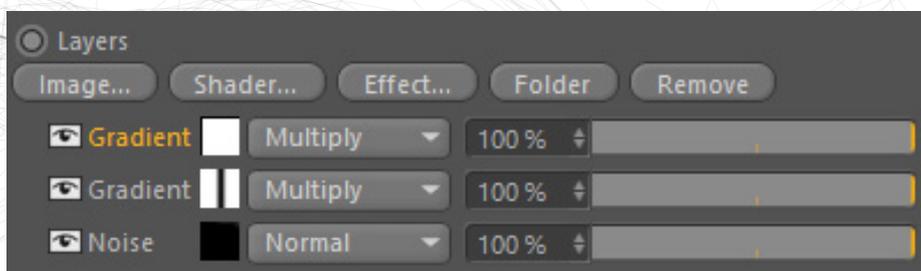
## Rétro Terrain

Annexes p.72

Le choix d'un faible nombre de subdivisions dans la géométrie est aussi un choix esthétique qui peut dans cet exemple reproduire ce look « Vaporwave ». Lorsque la surface contient peu de subdivisions, le bruit est directement moins détaillé et donne alors ce résultat de bosses anguleuses.



Afin de ne pas rendre le déplacement effectif au centre de la composition et ainsi créer un « chemin » rectiligne, le bruit est ici appliqué en tant que calque, sur lequel s'ajoute un dégradé noir au milieu et blanc sur les côtés, en mode de fusion « multiply ». De cette manière, le milieu est toujours de couleur noire et le déplacement n'affecte donc pas la géométrie.



Animation Retro Terrain :  
<https://www.ludwigdejonckheere.com/noise/assets/retro.mp4>

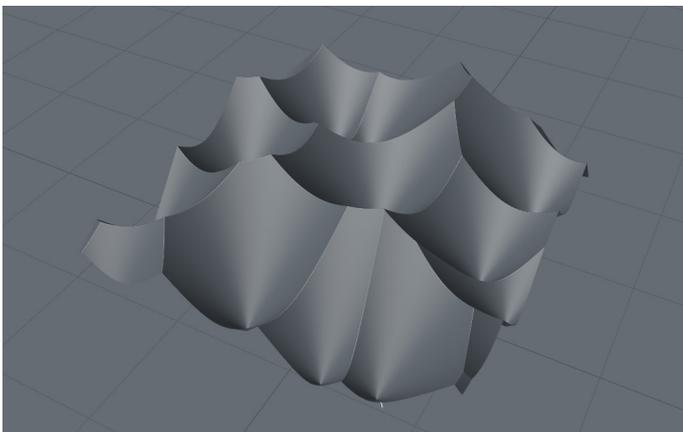
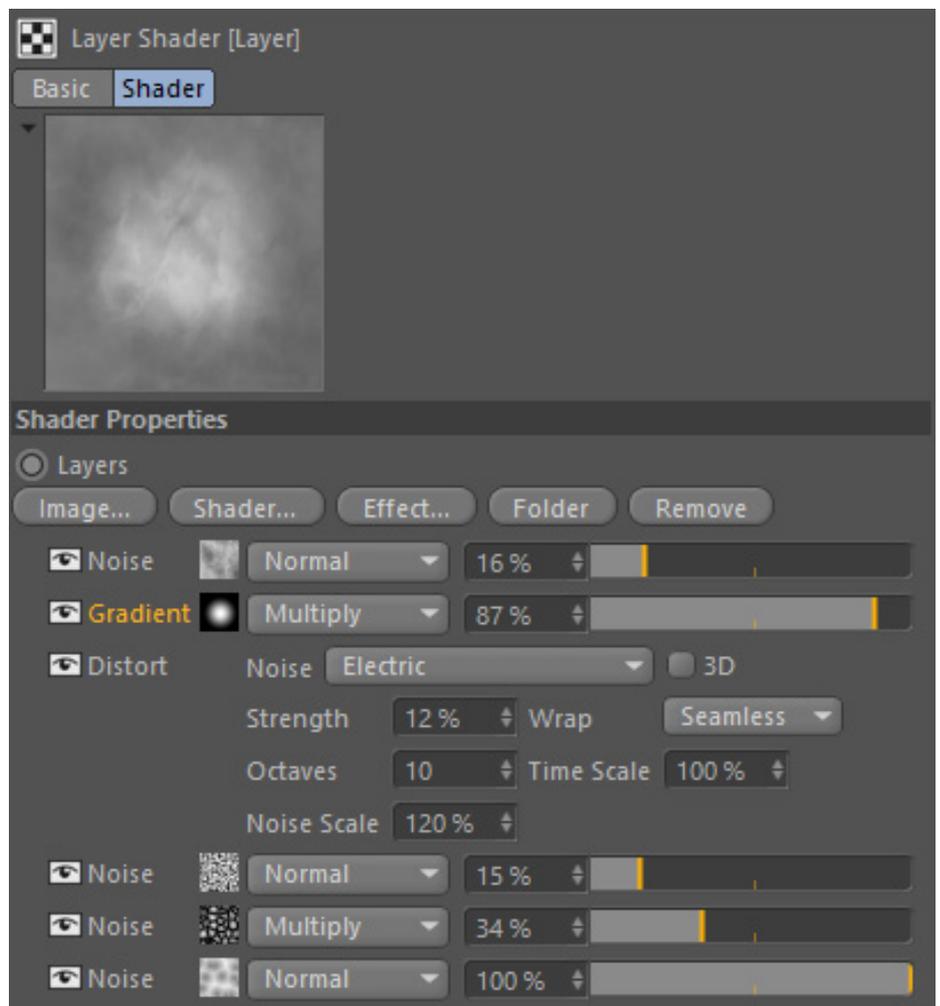
## Montagne

Annexes p.73

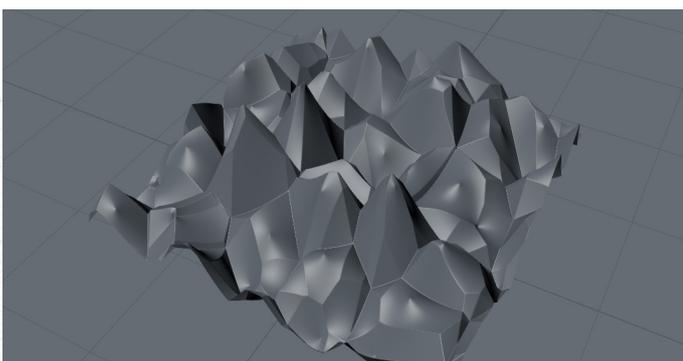
Pour ce dernier exemple d'application de bruit dans le « déplacement », l'objectif est de créer une chaîne de montagnes. Il existe un outil de création de terrain au sein de Cinema4D mais le but ici est de créer ces montagnes avec le déplacement.

Ceci consiste donc en une multitude de couches dans le déplacer qui agiront plus ou moins sur la géométrie.

Comme ces calques fonctionnent à la manière de ceux d'Adobe Photoshop, il faut commencer la lecture par le bas.



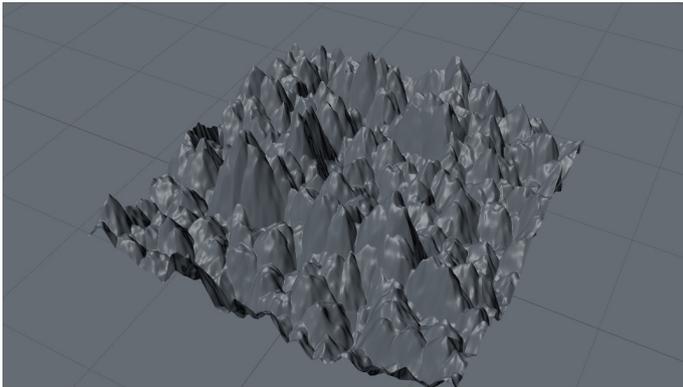
Dans un premier temps, il faut générer une géométrie de « gros œuvres » qui pose le volume global de la montagne. C'est pourquoi l'échelle de ce bruit est agrandie par rapport aux autres (ici 400%).



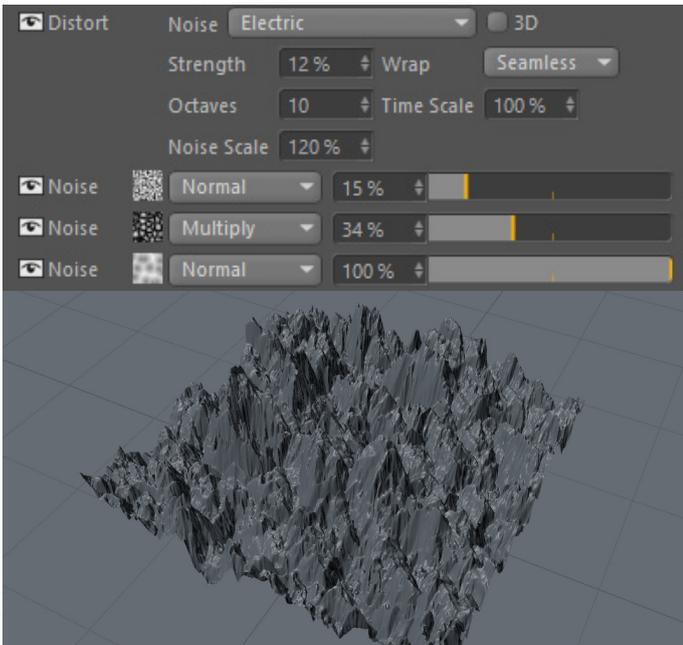
L'étape suivante est d'ajouter une couche de détails moyens qui garde la forme du premier volume et qui intègre un relief un peu plus détaillé. L'échelle de ce bruit est légèrement plus petite pour permettre de ne pas rentrer directement dans le détail (250%).

S'ajoute à ceci le fait que ce calque est en opacité moindre (ici 35%), en mode de fusion « multiply ».

La troisième couche, quant à elle, génère du détail plus fin et présente déjà une forme plus organique. Cette couche est laissée en mode normal, avec une faible opacité (15%).



Afin de venir maintenant creuser dans le détail, un effet est rajouté en mode « multiply ». Cet effet est une distorsion selon un bruit. Cette couche rajoute de la complexité et de l'aléatoire à la géométrie.

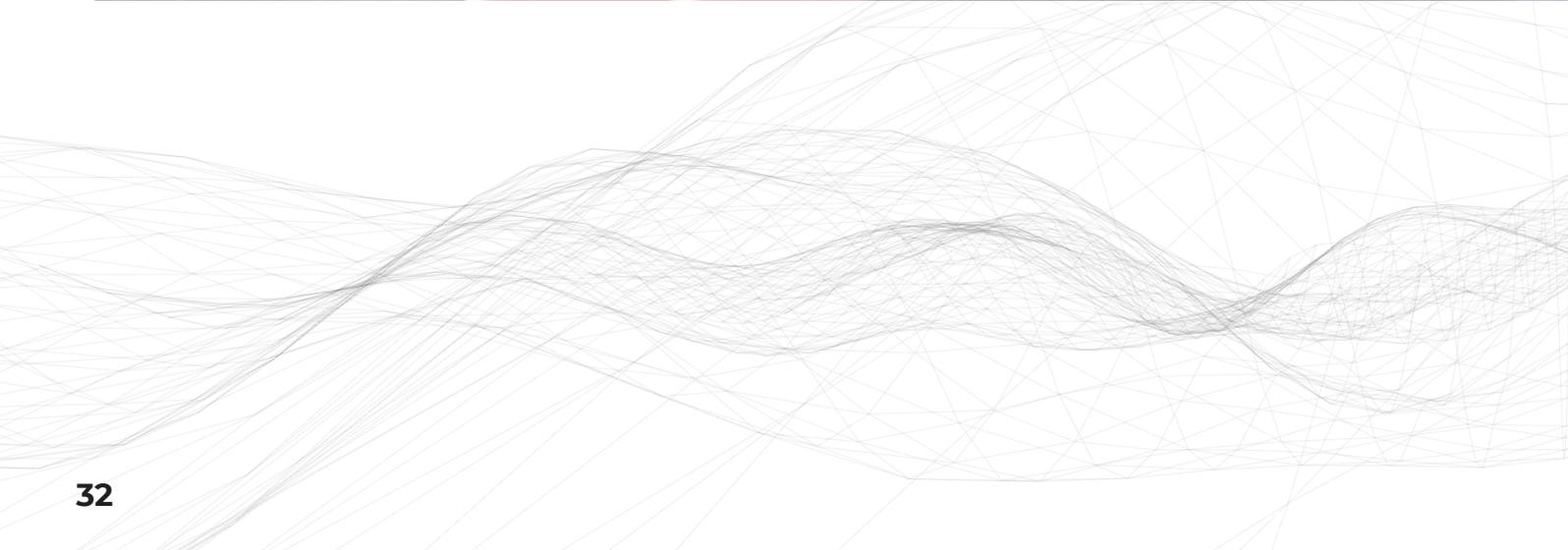


Enfin, pour donner à voir la topologie d'une montagne et non pas d'un terrain vague, un dégradé est rajouté sur le tout, afin que le déplacer ne prenne pas effet sur les bords.



La forme de la montagne est quasiment terminée, un dernier bruit est rajouté en texture fine, en mode normal, à 16% d'opacité.







# TEXTURING

## Qu'est-ce que le texturing ?

Le « texturing » se réfère à la création de matériaux. Un matériau est constitué de plusieurs caractéristiques permettant de recréer quasiment tous les matériaux existant dans le réel, qu'il s'agisse de verre, de pierre, de plastique, etc.

Dans ce contexte, le texturing se fait via l'éditeur de matériaux d'Octane, le moteur de rendu et non via l'éditeur de matériaux standards de Cinema4D.

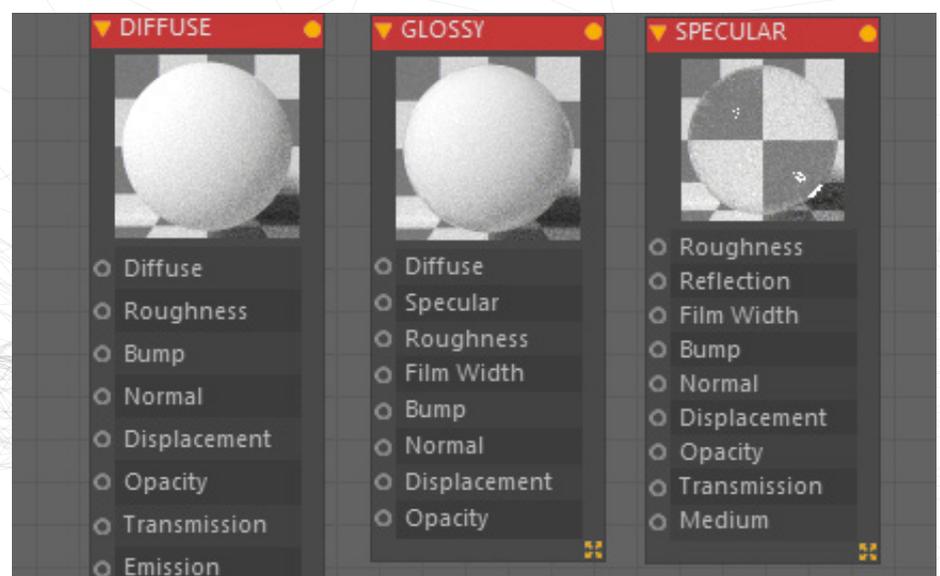
Parmi ces paramètres, il y a :

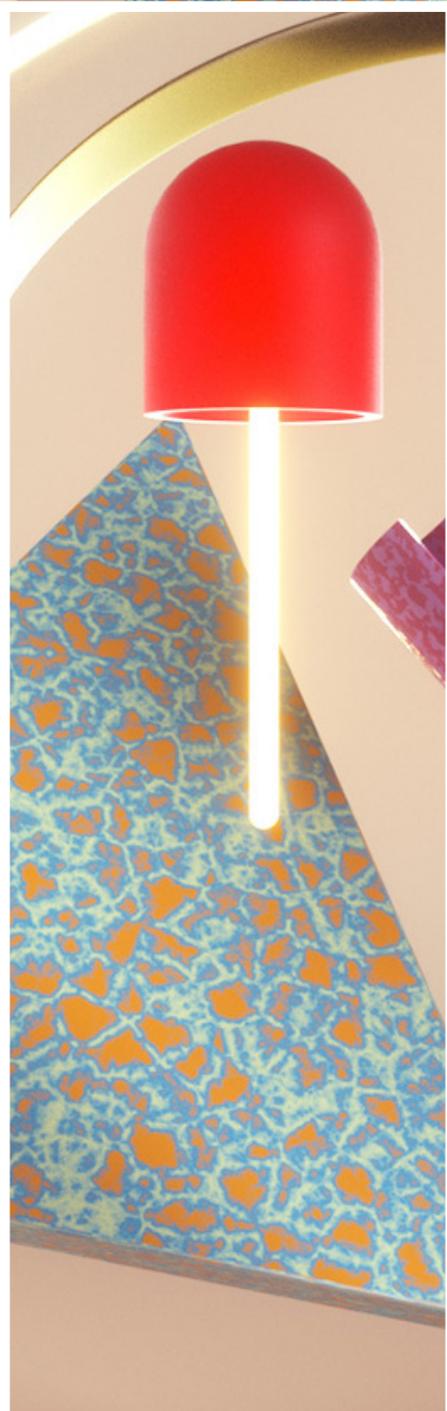
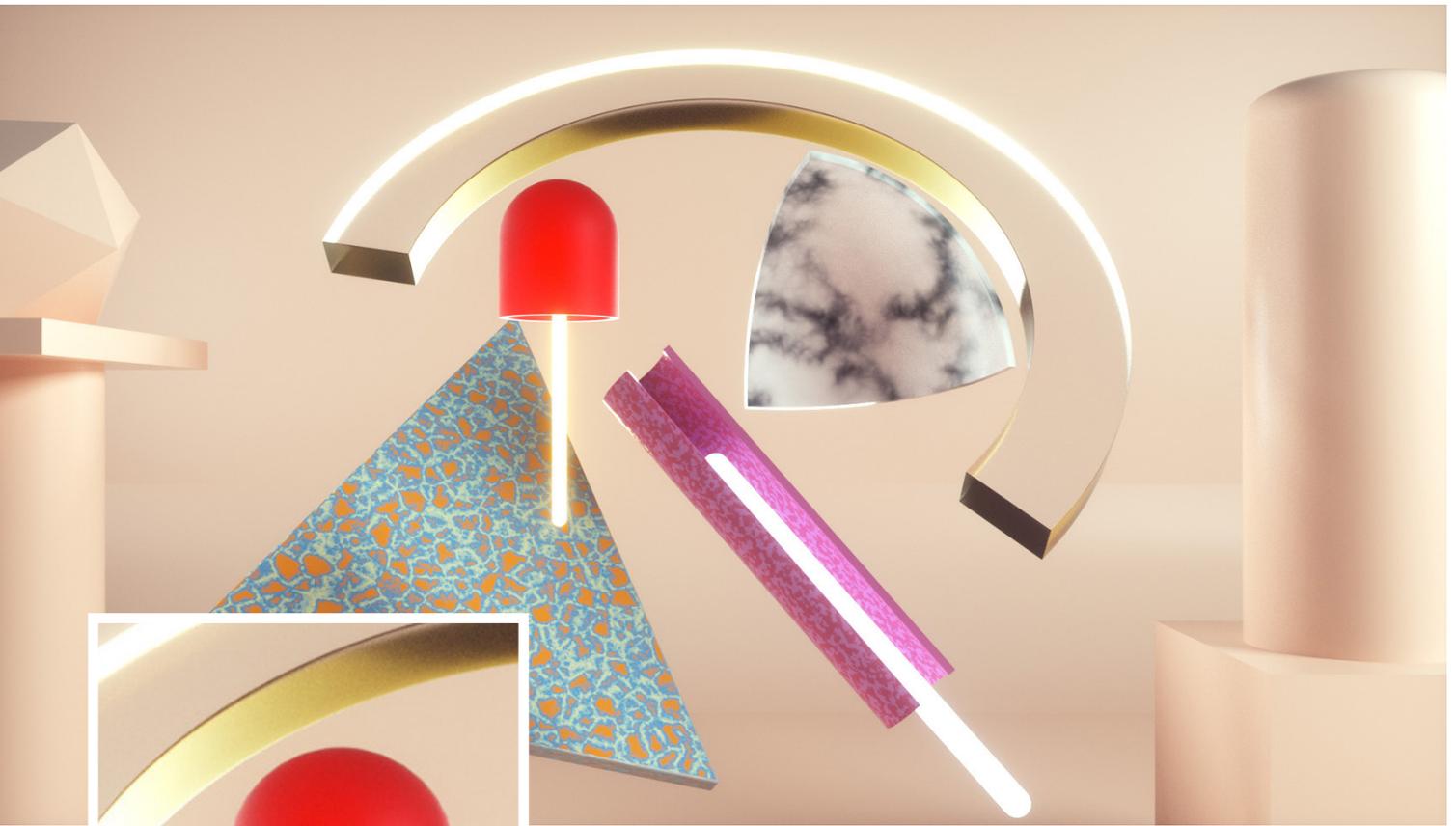
- « **diffuse** » : la couleur générale du matériau, que ce soit une couleur unie, un dégradé, un bruit, etc.
- « **roughness** » : la rugosité du matériau. Une faible rugosité rend la surface brillante, tandis qu'une forte rugosité la rend plus mat.
- « **bump** » et « **normal** » : ces deux paramètres simulent un relief depuis une texture. L'apparence du matériau change et peut donner cette illusion alors que la géométrie est inchangée.
- « **displacement** » : paramètre relatif au chapitre précédent, mais il s'agit ici du matériau, et non pas de l'objet 3D. Cependant, contrairement au « bump » et « normal », la géométrie est modifiée.
- « **opacity** » : tout simplement l'opacité du matériau.
- « **emission** » : le matériau devient une source lumineuse.

Il y a bien d'autres paramètres qui varient selon le matériau, mais ceux-ci sont les plus importants pour la suite.

Chaque paramètre décrit ci-dessus peut se voir attribuer une texture et, en l'occurrence, un bruit.

En ce qui concerne le bruit en lui-même, il est question, dans la plupart des cas, du bruit d'Octane et non de Cinema4D. C'est presque similaire, mais les paramètres sont présentés différemment.

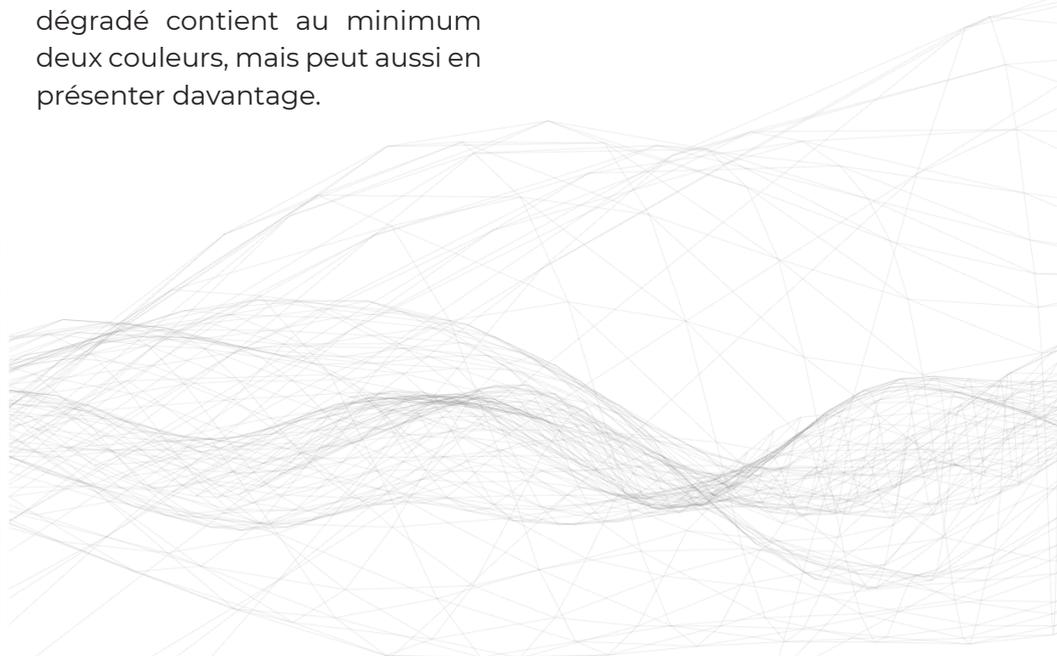




## Diffuse

Annexes p.74

En commençant par la caractéristique « diffuse » du matériau, le bruit agit directement sur la couleur de la texture. Le bruit est basiquement une nuance de noir et blanc, mais en lui associant un dégradé d'Octane, le bruit présente alors des variantes de couleurs. Ce dégradé contient au minimum deux couleurs, mais peut aussi en présenter davantage.





## Roughness

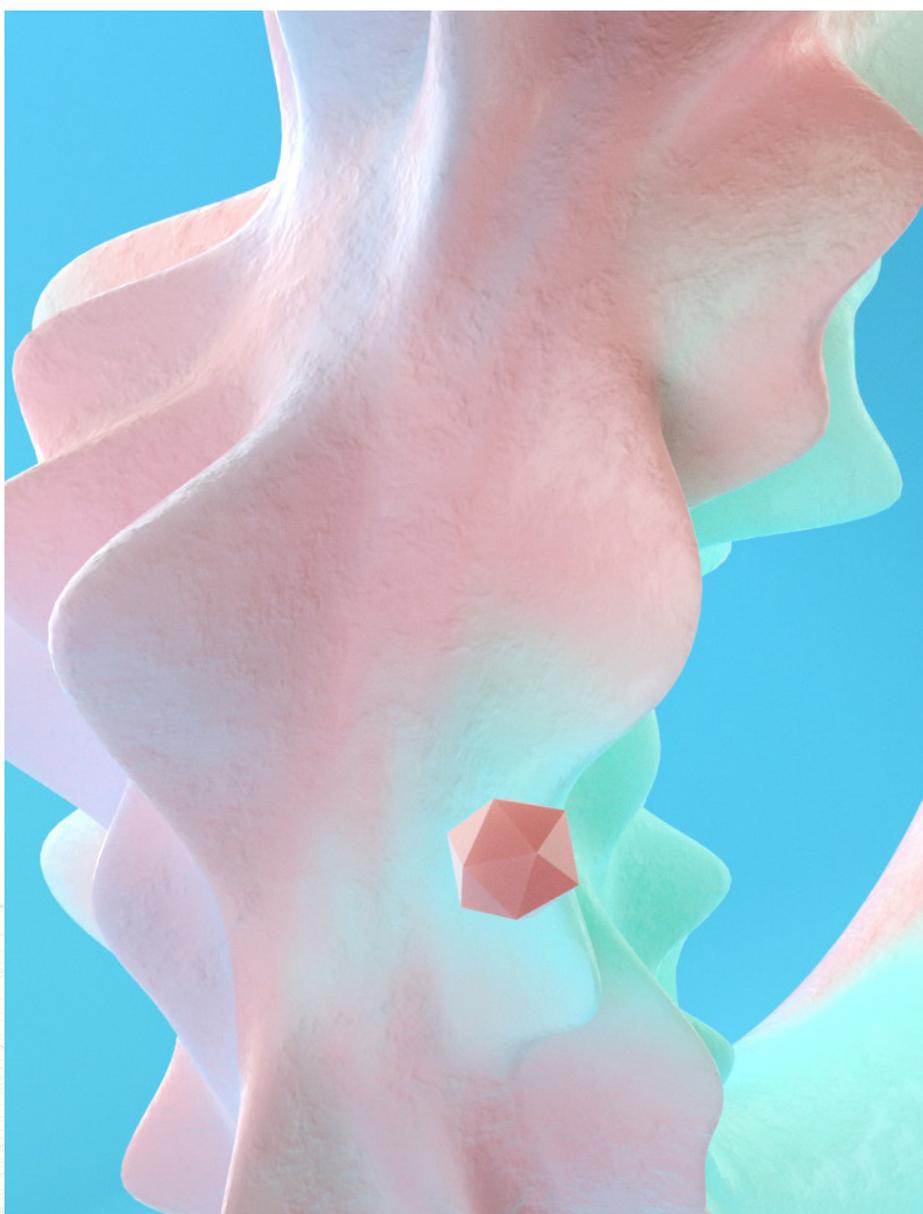
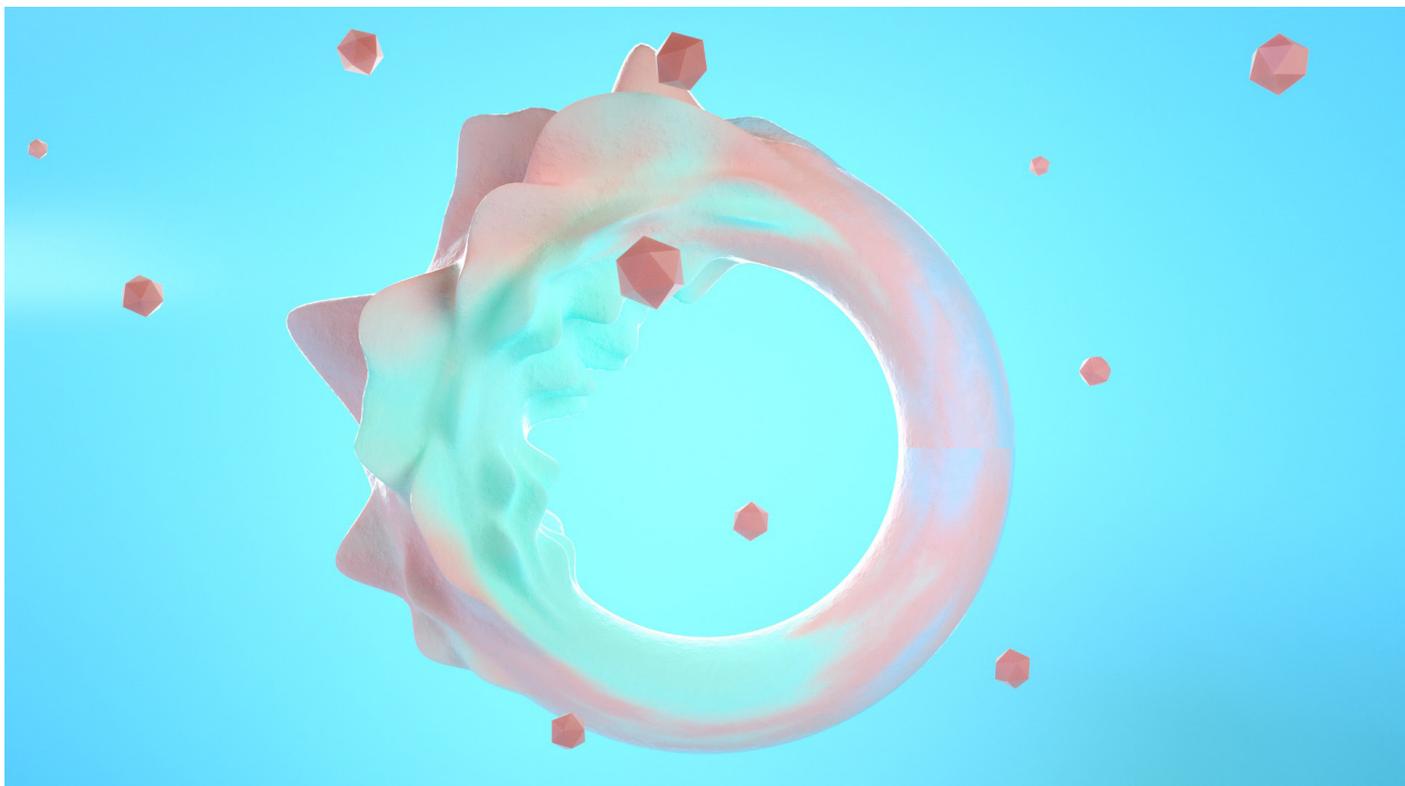
Annexes p.74

La rugosité d'un matériau va drastiquement changer son comportement vis-à-vis des reflets de lumière. Plus le matériau est lisse, plus les reflets seront définis et nets. Si celui-ci est rugueux, les reflets seront alors plus atténués et doux.

Ce paramètre peut varier avec une valeur allant de 0 à 1, 0 étant la valeur la plus lisse et 1 la valeur la plus rugueuse. Mais la rugosité peut aussi être définie en fonction d'une texture, et donc un bruit. Les nuances sombres et noires de ce bruit sont associées à l'aspect lisse et les nuances claires et blanches sont alors rugueuses.

Une bonne application du bruit dans le paramètre de rugosité est frappante lorsqu'il s'agit de matériaux spéculaires, comme du verre par exemple.

Dans cet exemple, il est question d'une forme géométrique, l'icosaèdre, auquel un matériau similaire à du verre est appliqué. Au niveau de la rugosité est attaché une texture de bruit, qui donne alors un aspect gelé ou sali au verre.



## Bump

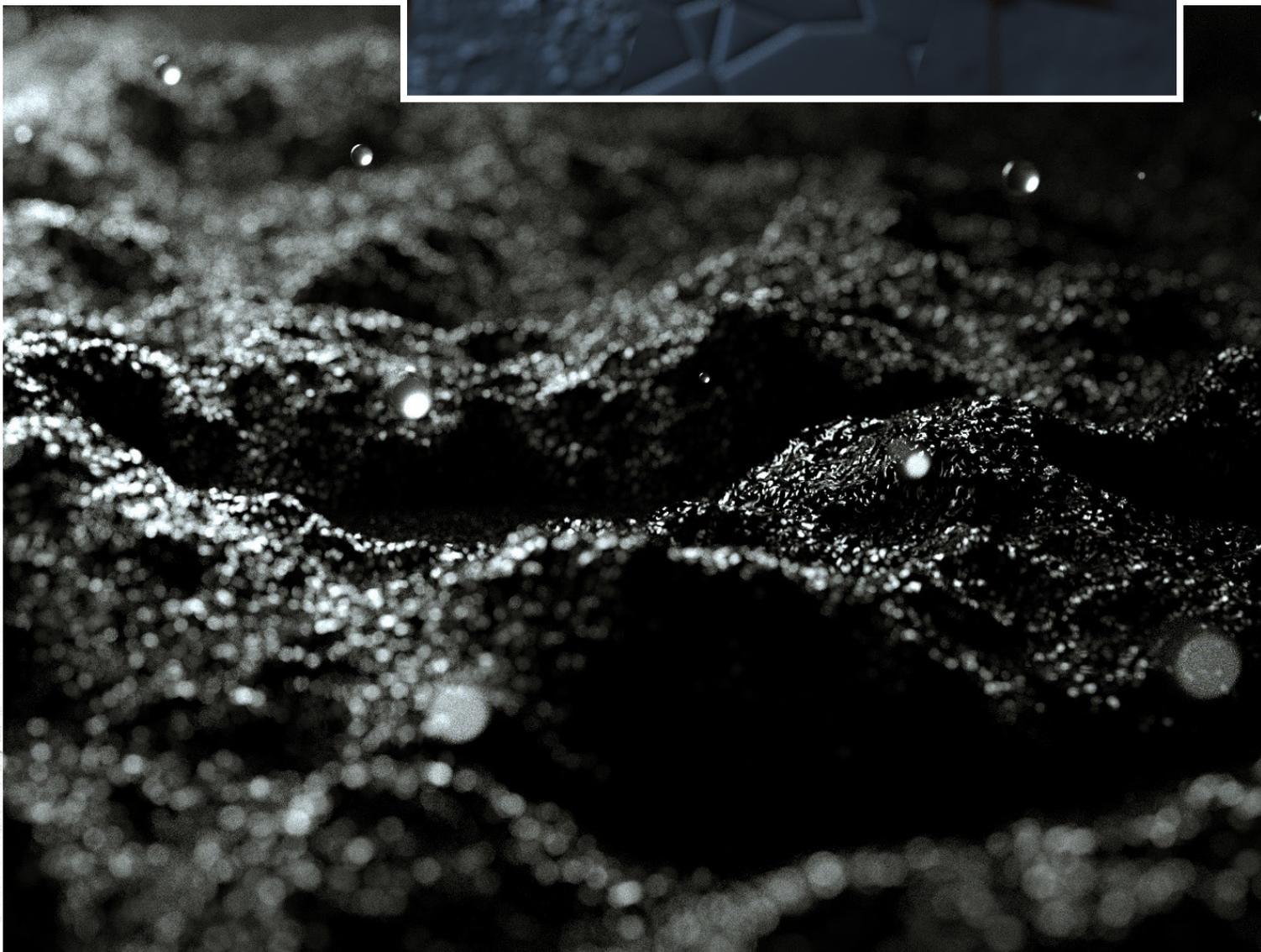
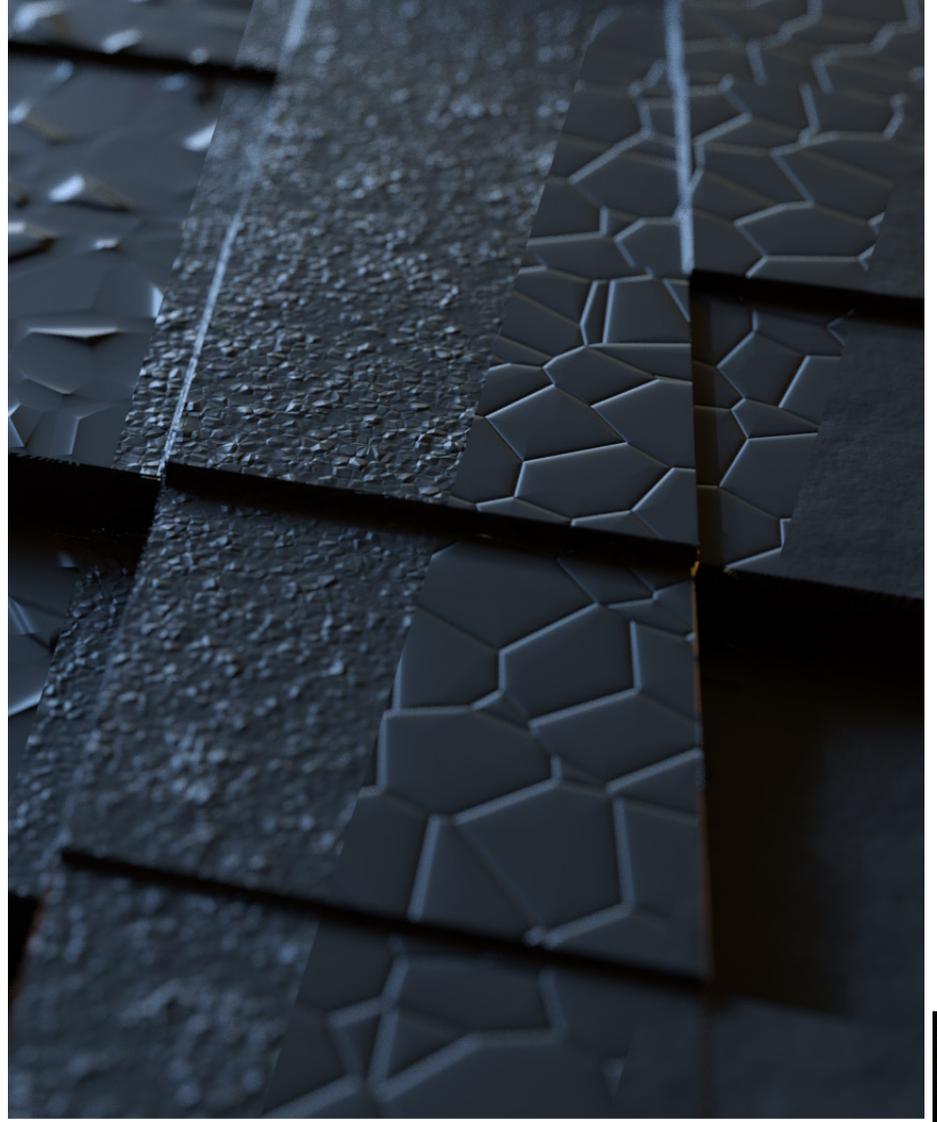
Annexes p.75

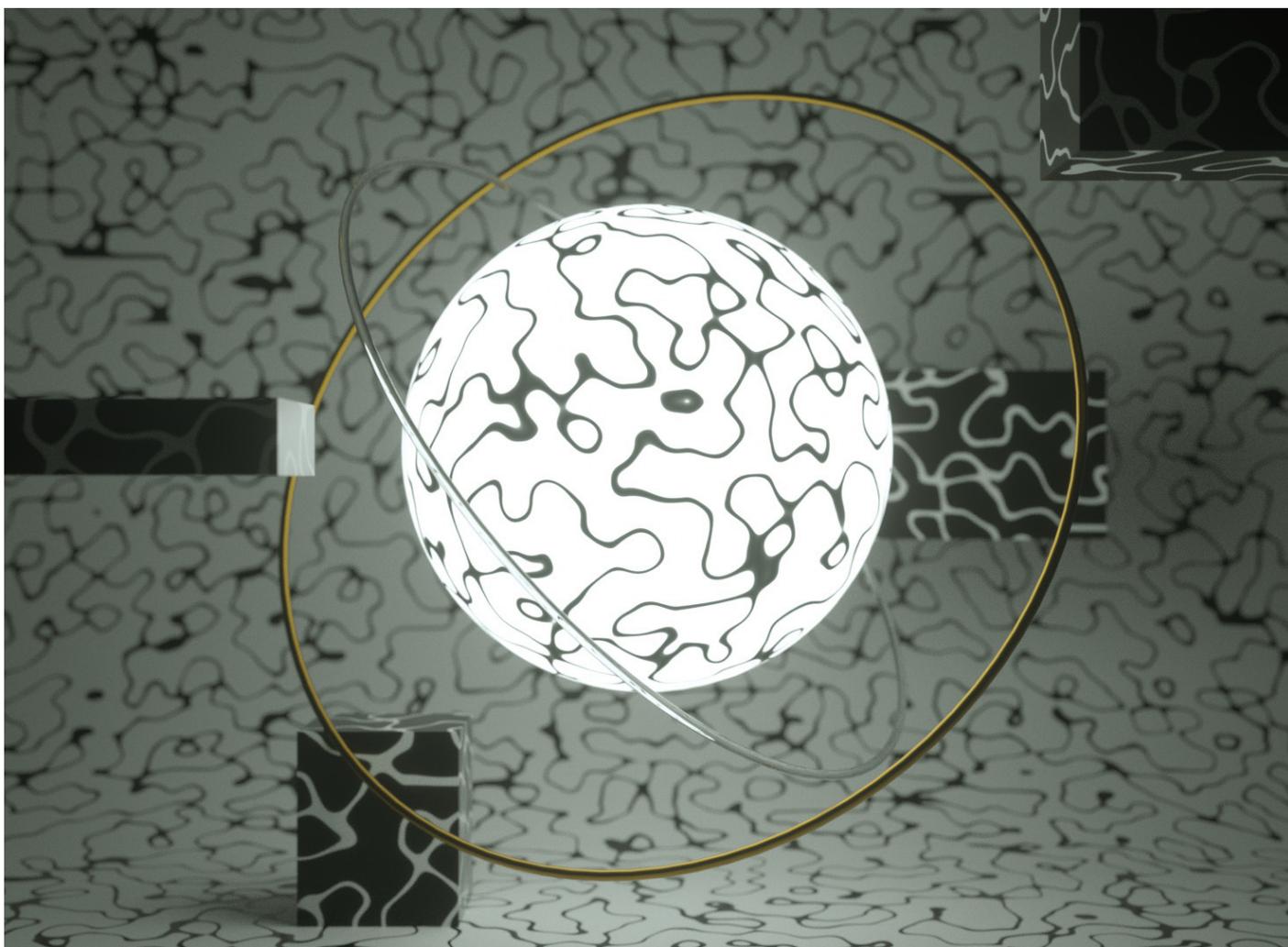
Le paramètre de « bump » consiste à simuler un relief, au même titre que les « normals ». Cependant, le paramètre de normals prend ici en compte un autre type de texture contenant des variances de couleurs selon les axes X, Y et Z. Un bruit est donc difficilement applicable. Mais ce n'est pas le cas du « bump », qui s'adapte aux nuances noires et blanches.

Ce paramètre est assez similaire au « displacement », à la différence que la géométrie ici reste inchangée, le relief n'agit pas véritablement sur la topologie.

Dans cet exemple de torus, le détail de la texture est assez fin et est généré via un bruit dans le « bump ». Ceci rend alors le matériau plus intéressant et riche.

Il existe également une possibilité de rajouter du détail supplémentaire à plus petite échelle sur des surfaces qui présentent déjà un « déplacement ». En mixant alors ces deux paramètres, on obtient un terrain complexe et organique.





## Emission

Annexes p.78

Enfin, le dernier paramètre de texture présenté est le canal « emission » ou émission en français. Il s'agit plus vulgairement de rendre une surface lumineuse. Comme pour les paramètres présentés précédemment, un bruit peut très bien s'insérer en tant que source de lumière.

Dans cet exemple, c'est le bruit cranal qui est source de lumière, où la partie blanche devient alors émettrice.

Animation Emission :  
<https://www.ludwigdejonckheere.com/noise/assets/emission.mp4>

# VOLUME

## Que sont le Volume Builder et le Volume Mesher ?

Ces deux fonctionnalités sont apparues dans la version R20 de Cinema4D et fonctionnent ensemble.

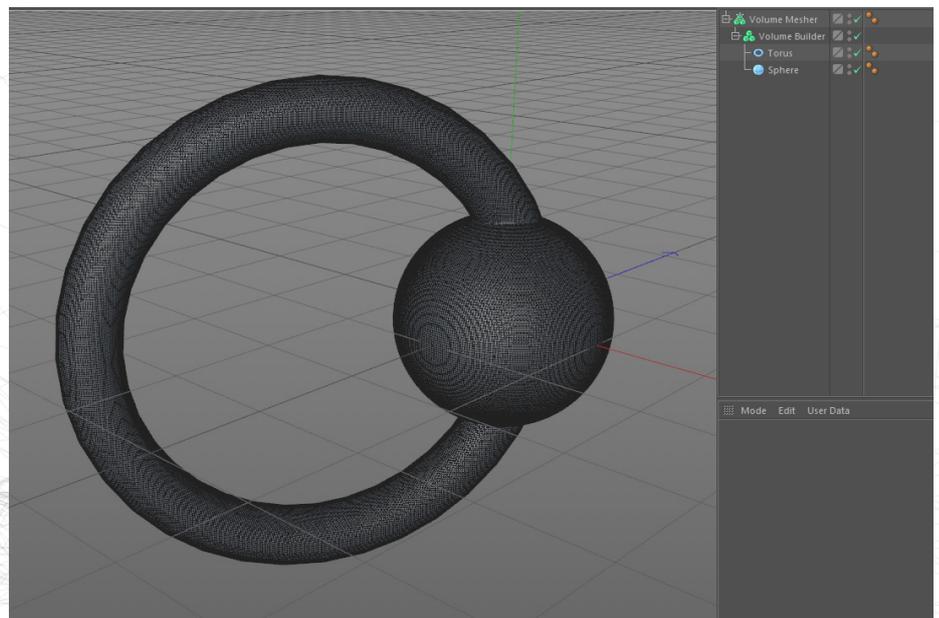
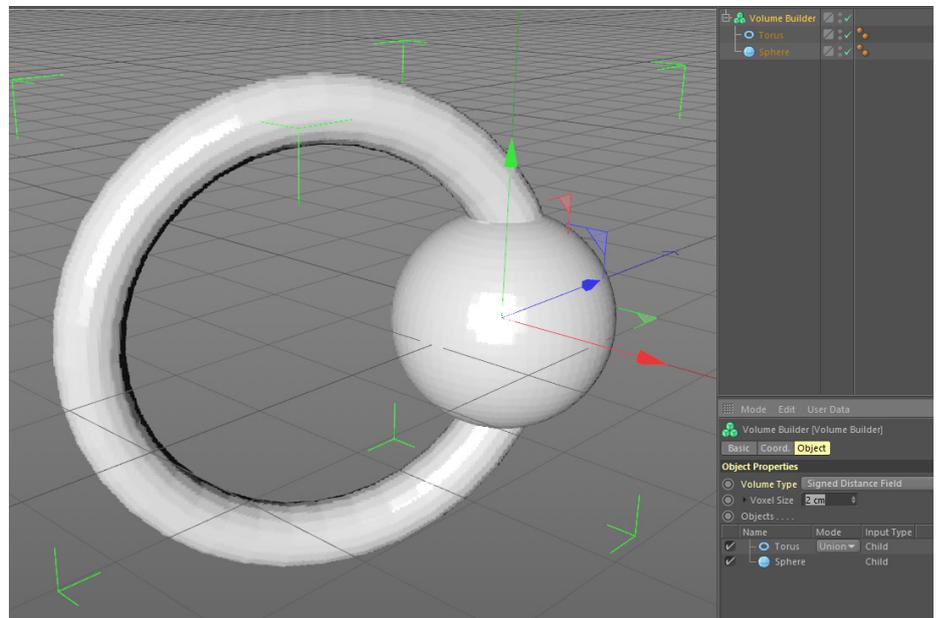
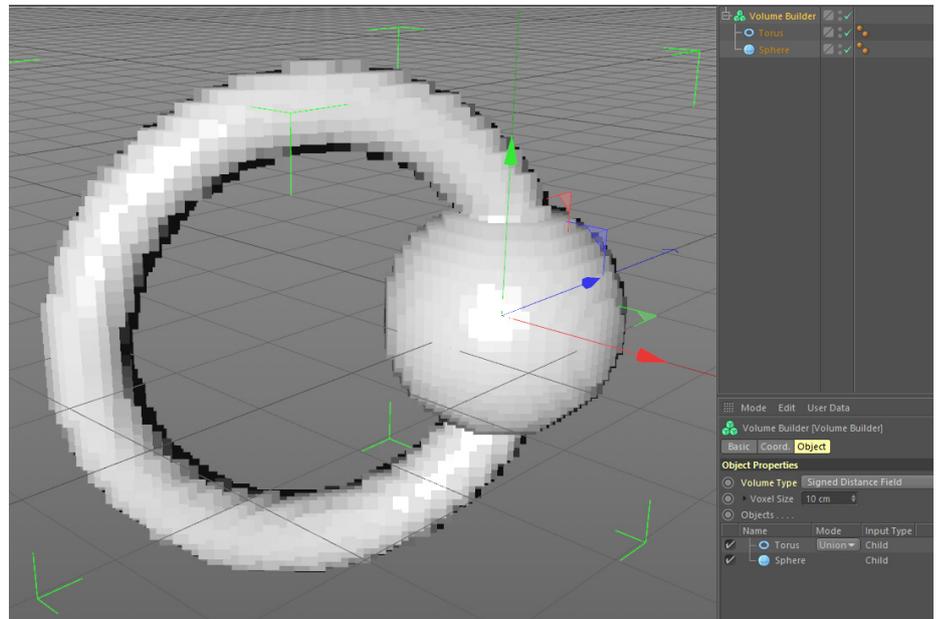
Premièrement est appliqué le « volume builder », sur un/des objet(s) 3D. Le « volume builder » transforme alors la géométrie en un ensemble de « voxels », une sorte de pixels 3D. Ces derniers prennent l'apparence de petits cubes blancs. La taille des pixels 3D peut varier, rendant ainsi le résultat final plus ou moins détaillé.

Si la taille des petits cubes est petite, le résultat est plus précis. À l'inverse si les « voxels » sont de plus grande taille, le détail est perdu.

Attention cependant, plus les pixels 3D sont petits, plus le logiciel demande de ressources, et donc ralenti, voire se ferme ou ne répond plus.

Ce « volume builder » va donc transformer une forme géométrique en un ensemble de pixels 3D. Sur un objet 3D simple comme une sphère ou un cube, l'utilité est nulle. Si plusieurs objets complexes sont chevauchés et entremêlés, alors cette fonctionnalité devient intéressante.

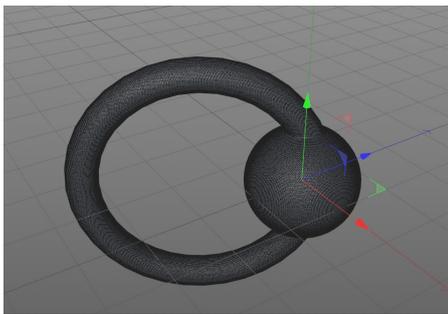
En effet, les diverses formes géométriques vont fusionner et se souder les unes aux autres, pour créer une forme globale.



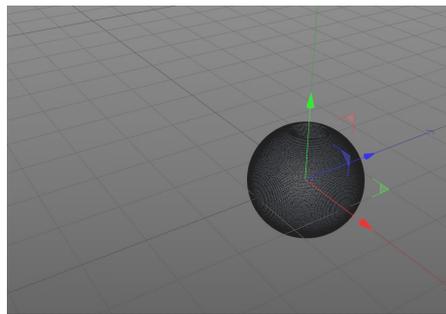
À cette étape, la forme n'apparaît encore que comme un ensemble de pixels 3D agglutinés. C'est alors qu'intervient le « volume mesher » qui va retransformer ces cubes en une vraie forme géométrique composée de polygones.

Plusieurs modes sont proposés avec ces nouvelles fonctionnalités, un peu à l'image de l'outil « booléen », un outil très connu dans le monde de la 3D.

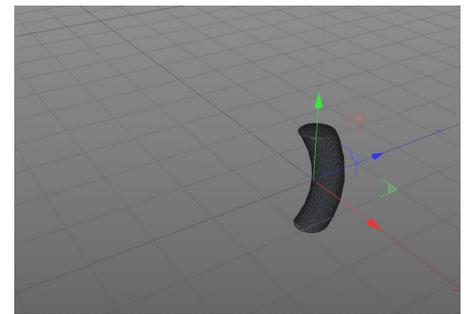
Basiquement, ces modes sont au nombre de 3 :



**Union** : permet d'unir plusieurs objets 3D pour n'en former qu'un.

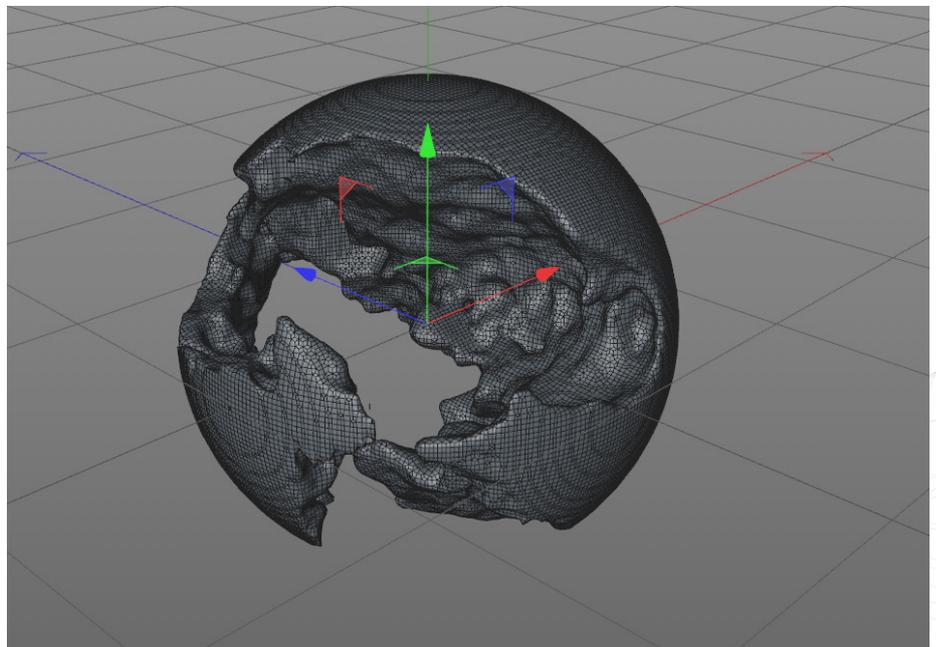


**Soustraction** : creuse la forme d'un objet en fonction de la forme des autres objets. L'ordre des objets est ici prise en compte.



**Intersection** : ne prend uniquement en compte que la partie intersectée entre les différents objets 3D.

Dans le cadre du bruit, et concernant le « volume builder » et « volume mesher », l'idée est de creuser un objet 3D grâce à un champ aléatoire, qui est le bruit. En utilisant le mode soustraction, la forme est creusée par le bruit paramétré auparavant.

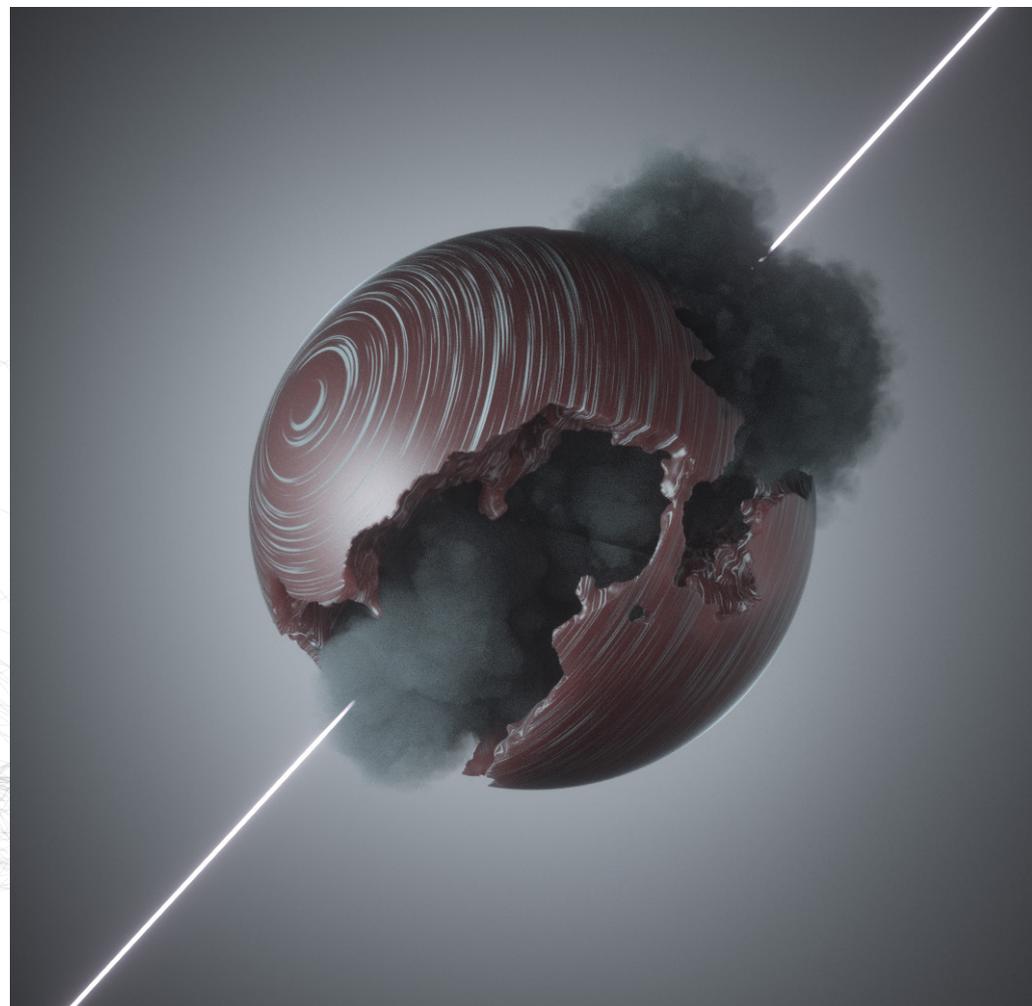
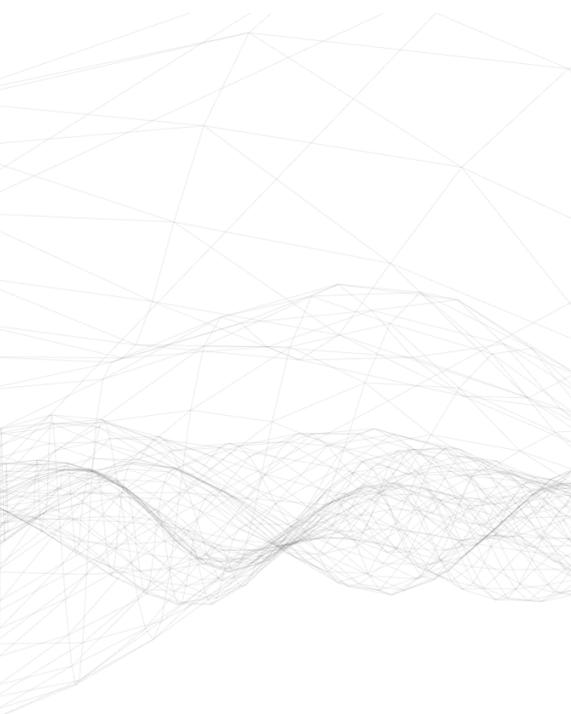
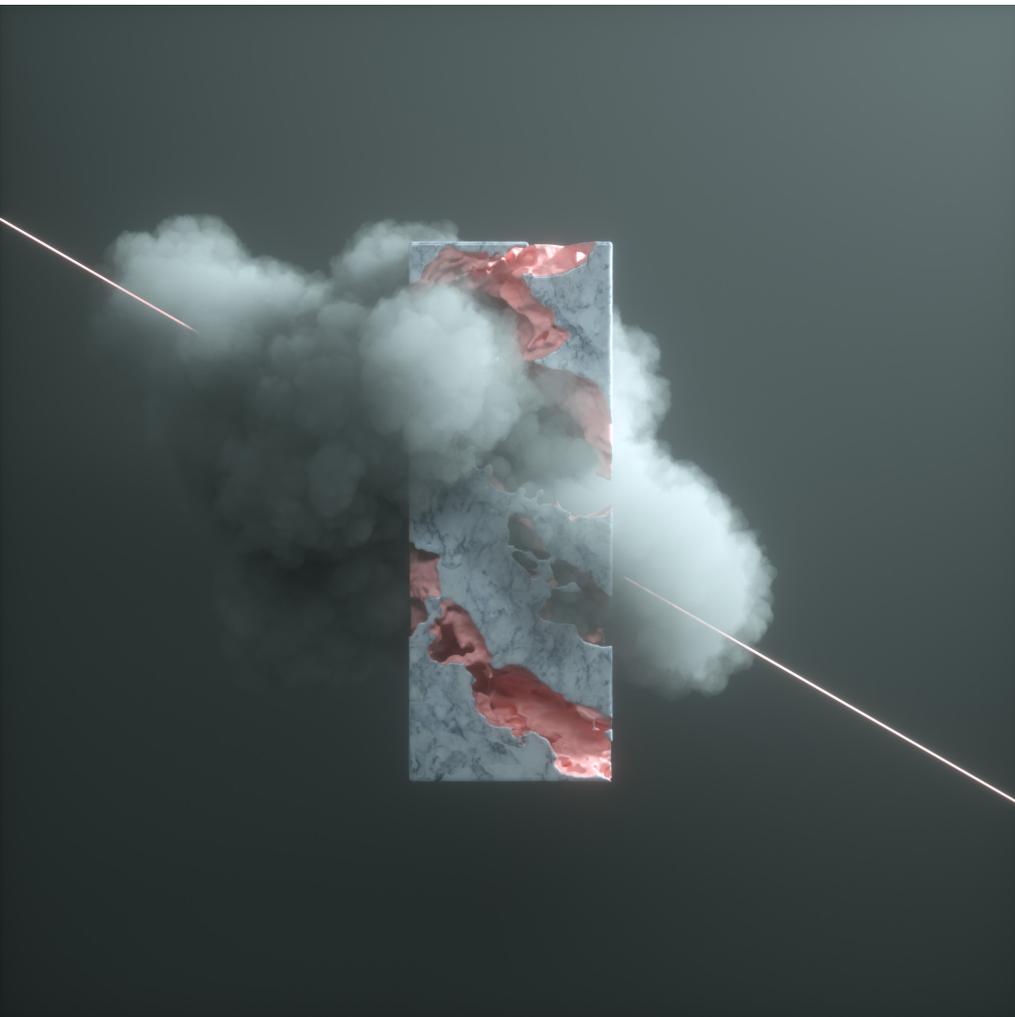


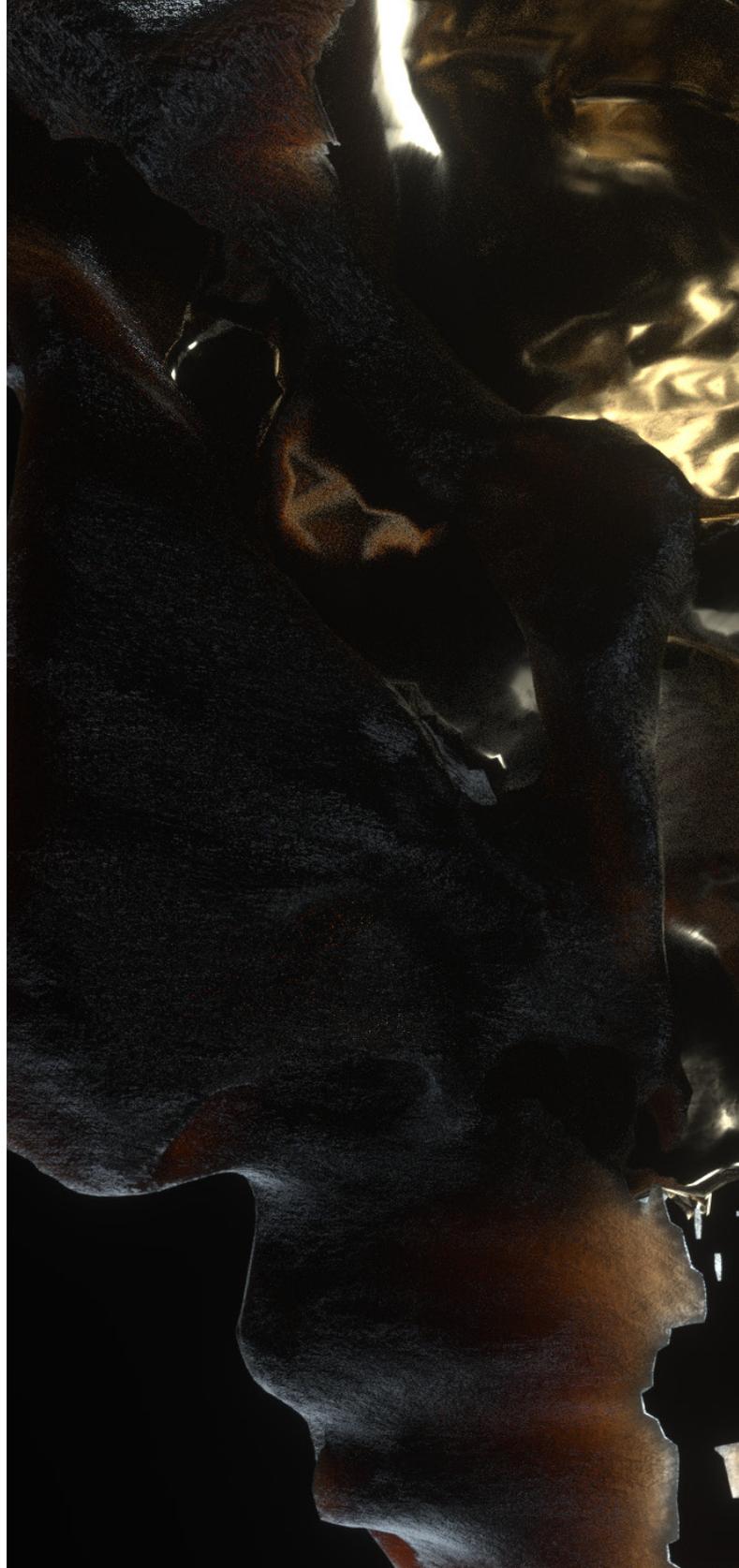
## Marble & Ball

Annexes p.79

Dans ces exemples, l'idée est d'utiliser des formes simples pour ainsi y creuser à l'aide du bruit. Le résultat donne alors quelque chose d'organique et aléatoire, avec des aspérités.

Pour donner l'illusion que le centre de la forme est différent de sa surface, il faut sélectionner tous les polygones qui sont « à l'intérieur » de cette dernière, et ensuite attribuer un nouveau matériau.





## David

Annexes p.80

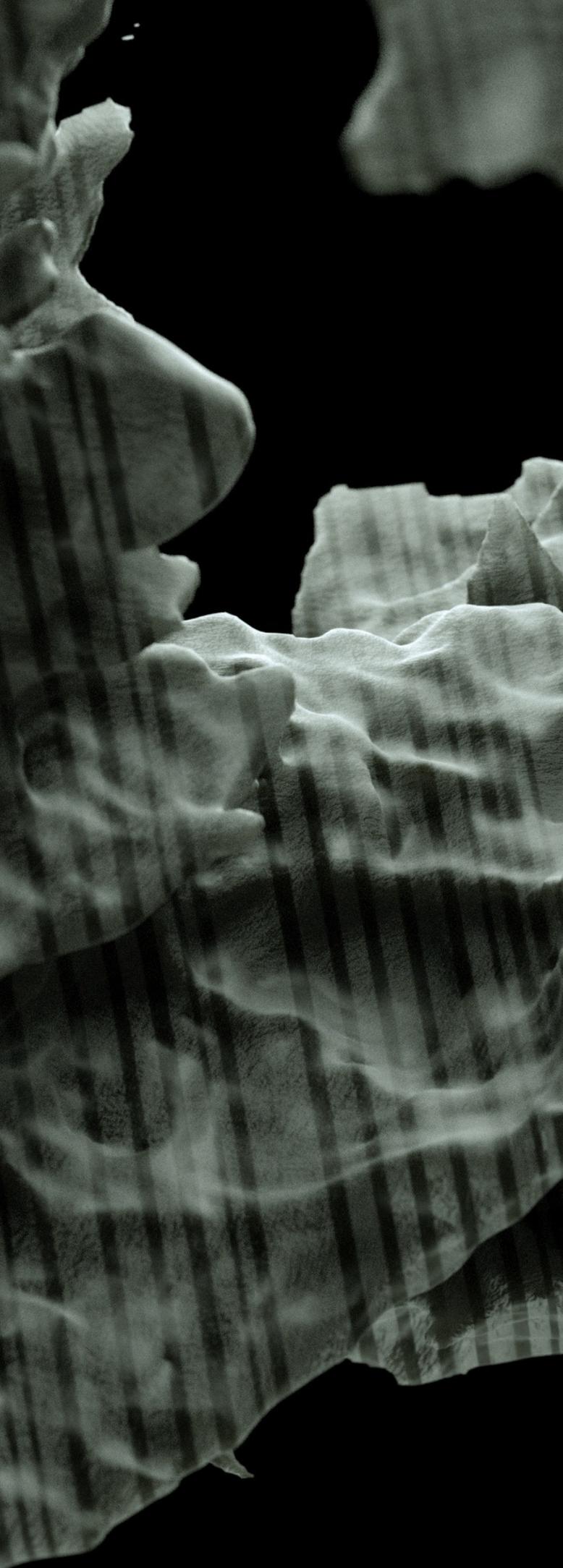
Dans cet exemple, il n'est plus question d'objets 3D simples, mais d'appliquer cette méthode sur des formes plus complexes. La démarche reste similaire, mais il s'agit ici d'une statue scannée en 3D (la tête de David par Michel-Ange).

Lorsque la tête est creusée, l'intérieur est visible et laisse alors apparaître un crâne en or.



## ANIMATIONS ALÉATOIRES

Pour ce dernier chapitre concernant la 3D, le principe est d'animer des objets de manière aléatoire, sur base d'un bruit. Ce bruit n'est donc pas visible concrètement comme les exemples précédents, mais agit plutôt comme un champ de valeurs quelconques qui affecte la position, la rotation, l'échelle, etc.

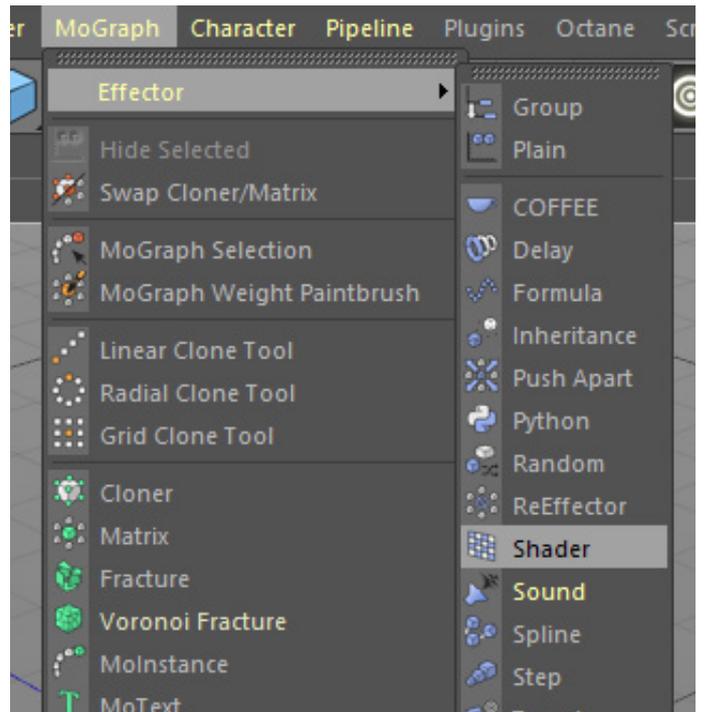


## HUD

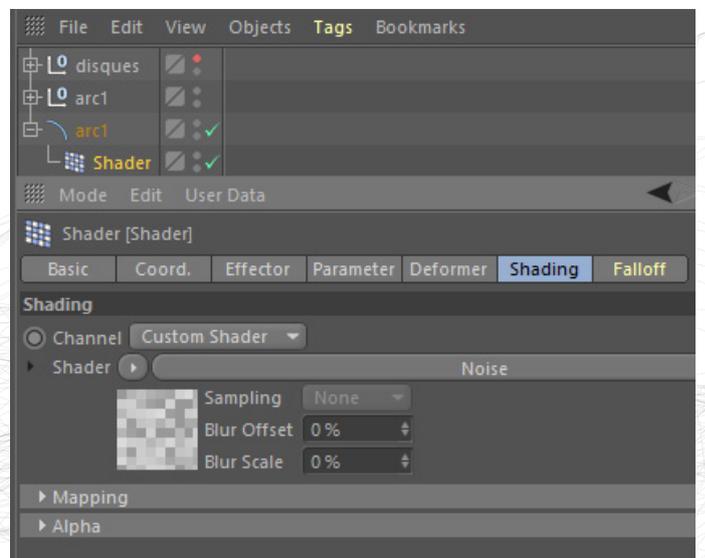
Annexes p.81

Pour cet exemple, l'animation est décomposée en plusieurs parties, qui seront unies en un projet final regroupant tous ces éléments.

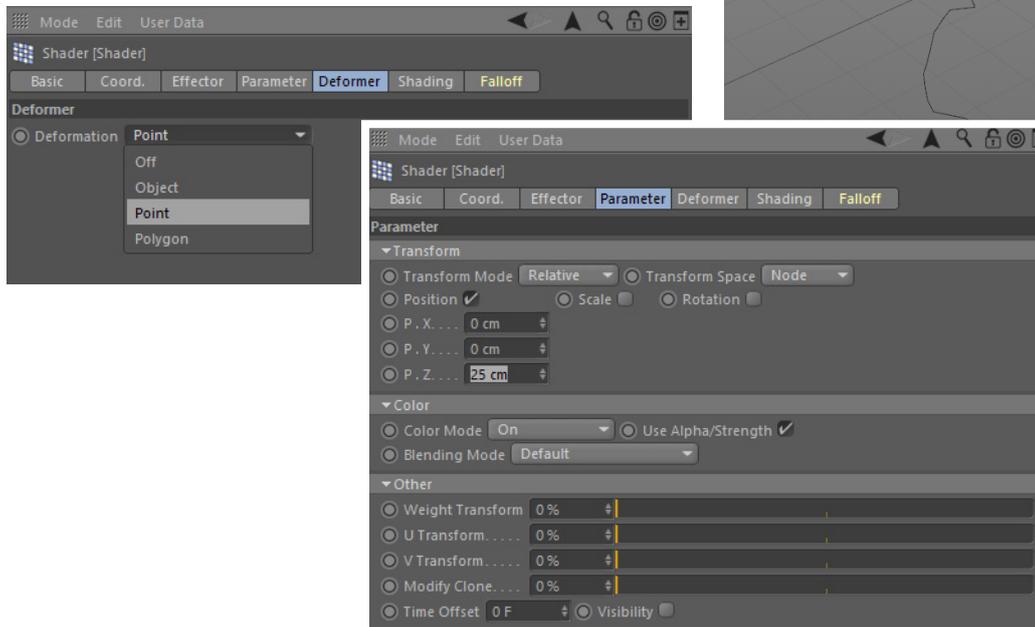
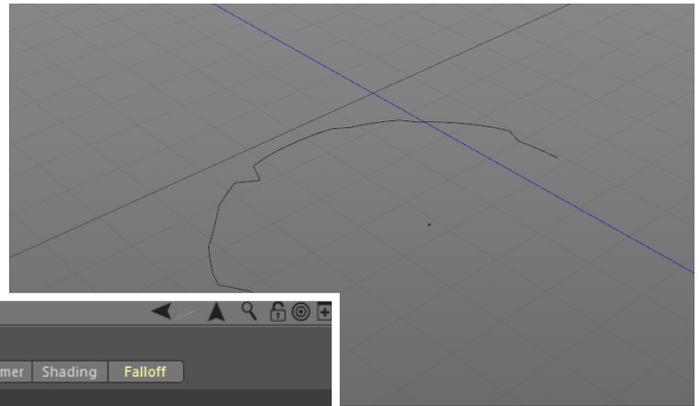
En premier lieu, il s'agit d'animer une courbe selon le bruit. Il n'est pas question ici de « déplacer », mais de « shader », un autre outil du MoGraph de Cinema4D.



Dans les paramètres de ce « shader » est appliqué un bruit qui sera la source du mouvement. Pour ce projet, le « Cell Noise » sera approprié.



Reste à déterminer ce qu'affecte le bruit. Dans le cas de cette courbe, il s'agit de la position des points de cette courbe selon l'axe Z. Pour que la déformation prenne effet, il s'agit dans un second temps d'affecter le « shader » aux points de la courbe, et non pas à l'objet.



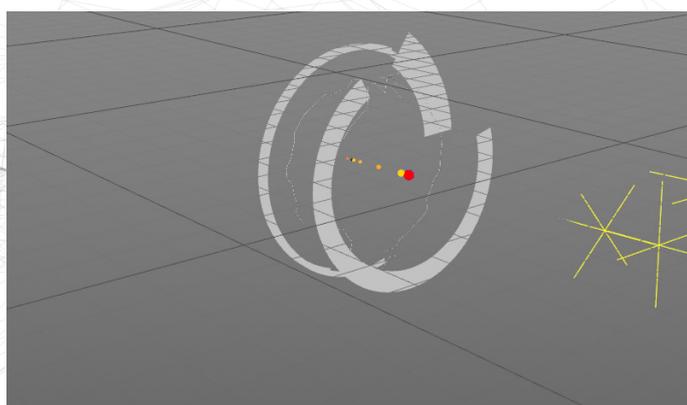
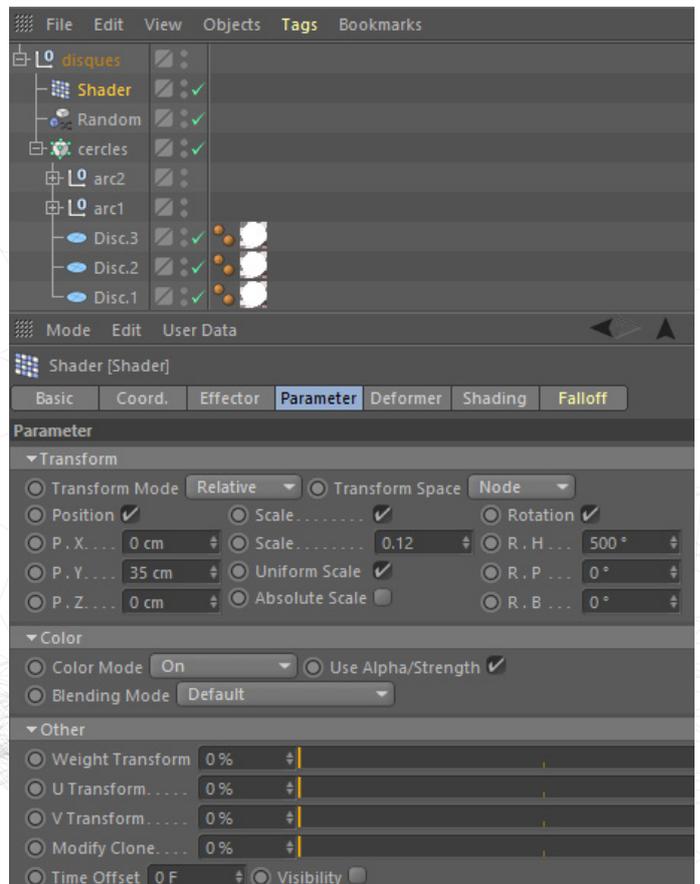
C'est en attribuant enfin une vitesse au bruit que l'animation se crée. Le « Cell Noise » a cette particularité d'avoir une animation très abrupte, ce qui est recherché dans ce projet.

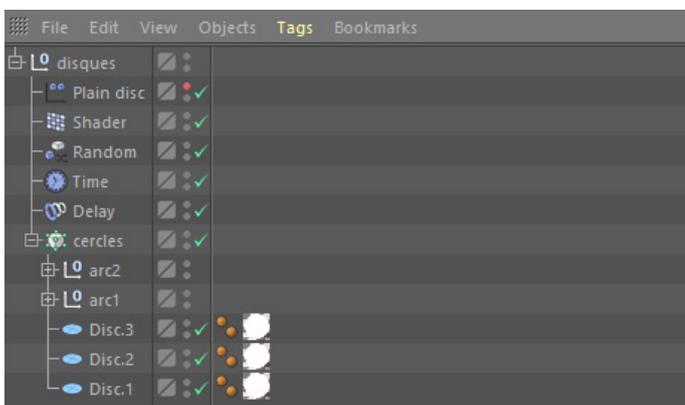
Animation courbe : [https://www.ludwigdejonckheere.com/noise/assets/animation/anim\\_courbe.mp4](https://www.ludwigdejonckheere.com/noise/assets/animation/anim_courbe.mp4)

Vient ensuite l'animation d'arcs de cercle ainsi que de la courbe précédente. À cette fin, ces différents éléments sont entreposés dans un « cloner ». C'est sur ce dernier qu'un nouveau « shader » prend effet.

Il s'agit ici d'utiliser la même méthode que pour la courbe, à la différence que l'animation affecte à la fois la position sur un axe, l'échelle globale, ainsi que la rotation selon un axe.

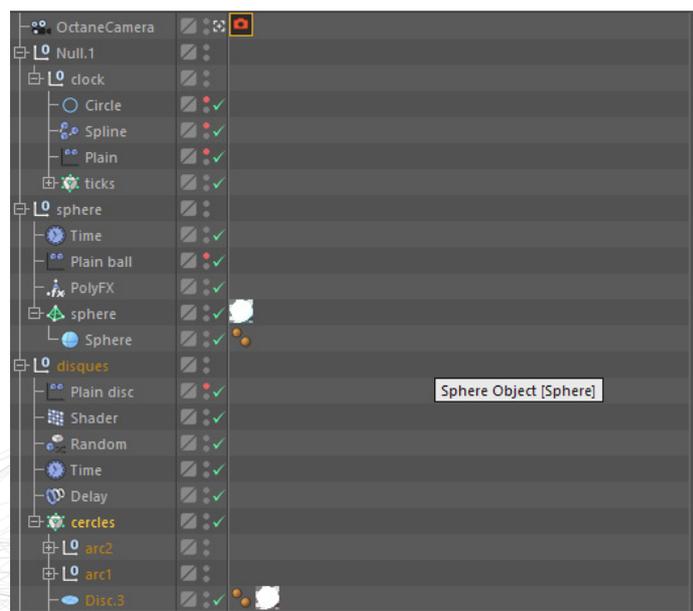
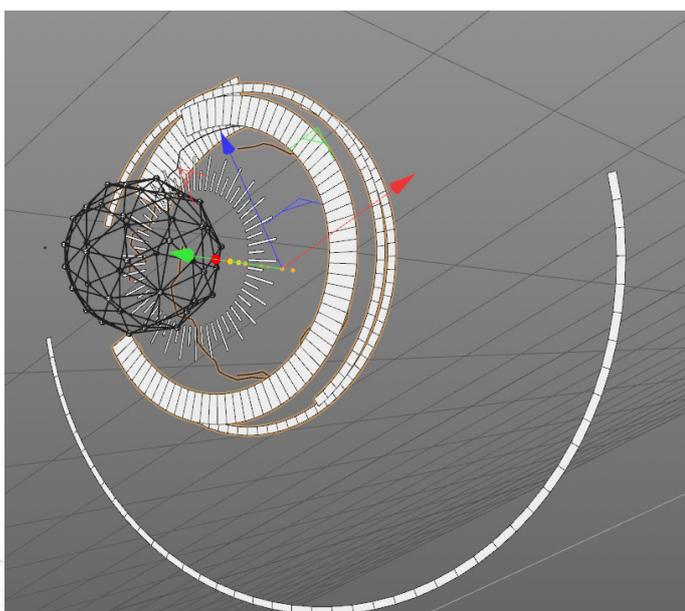
Animation courbe : [https://www.ludwigdejonckheere.com/noise/assets/animation/anim\\_courbe.mp4](https://www.ludwigdejonckheere.com/noise/assets/animation/anim_courbe.mp4)





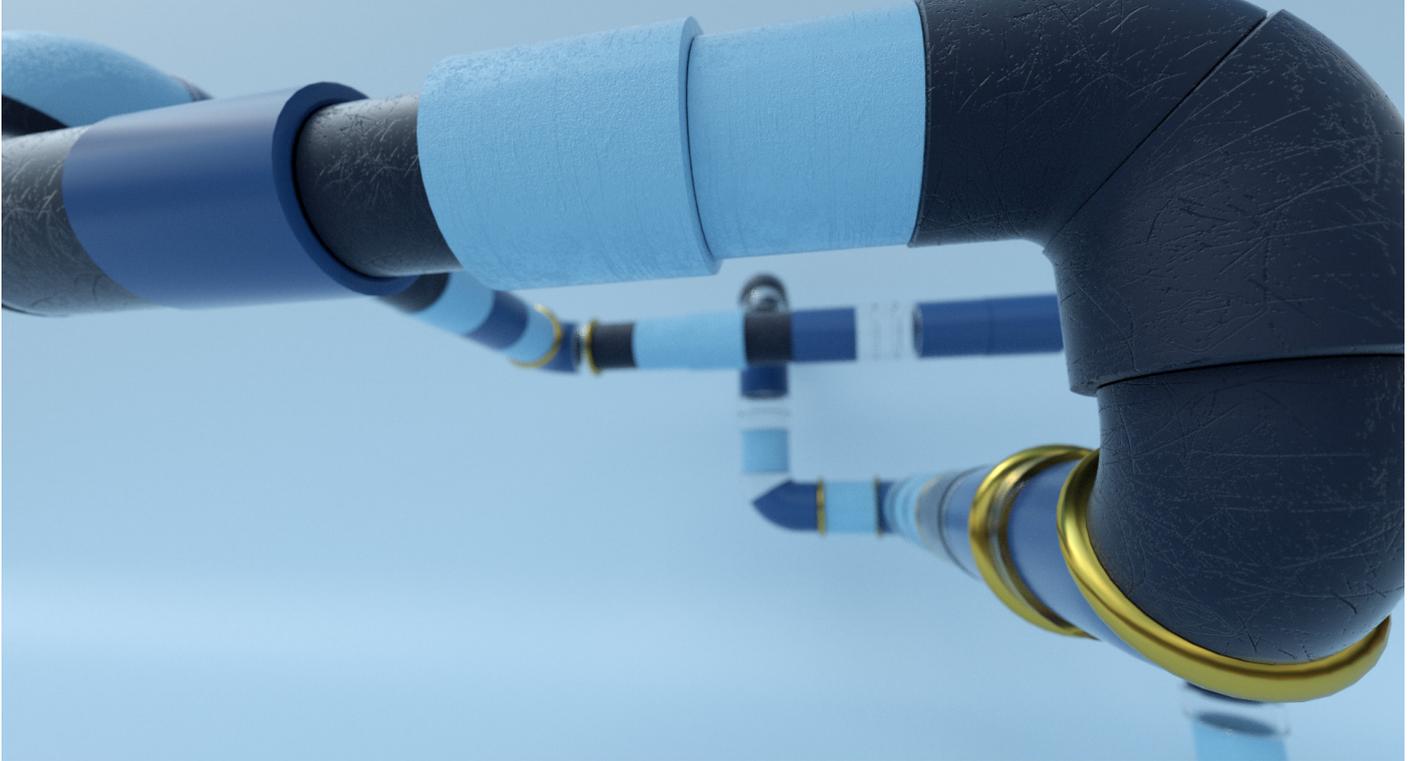
Pour enjoliver le tout, différents outils du « MoGraph » sont appliqués en aval, que ce soit pour rendre l'animation moins brutale lors mouvement, ou encore l'apparition des éléments de manière esthétique, ou enfin de permettre une rotation continue des éléments sans avoir recours aux clés d'animation.

L'animation finale reprend alors les exemples précédents, ainsi que quelques objets 3D ajoutés pour donner du détail supplémentaire à l'animation.



Animation Éléments :  
[https://www.ludwigdejonckheere.com/noise/assets/animation/anim\\_no\\_track.mp4](https://www.ludwigdejonckheere.com/noise/assets/animation/anim_no_track.mp4)

Animation HUD :  
[https://www.ludwigdejonckheere.com/noise/assets/animation/tracking\\_v2.mp4](https://www.ludwigdejonckheere.com/noise/assets/animation/tracking_v2.mp4)

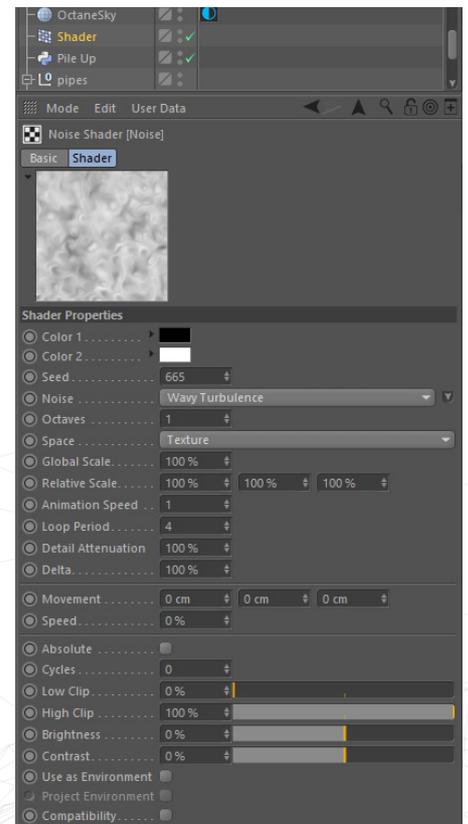
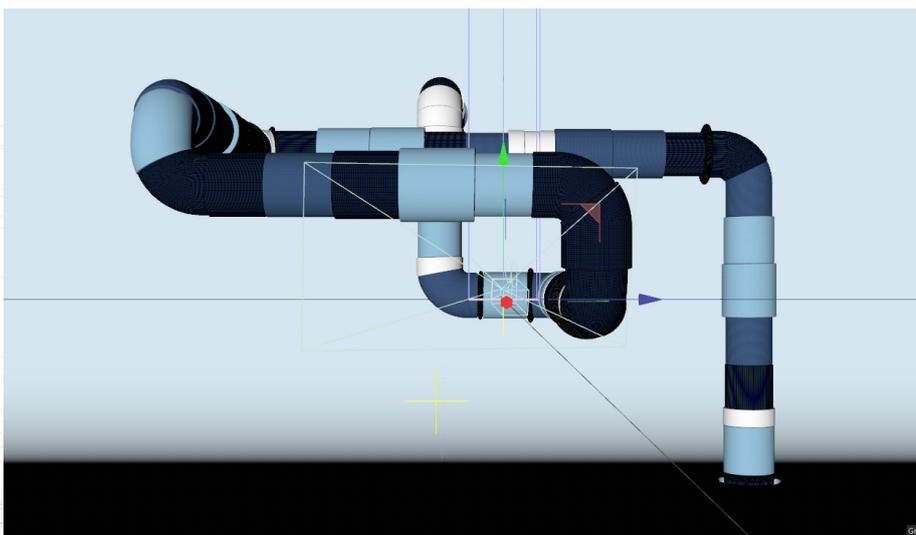


## Tubes

Annexes p.81

Dans cet exemple, la méthode reste la même, à savoir l'utilisation du « shader » sur un « cloner ». À la différence que dans ce projet, il s'agit uniquement d'affecter l'échelle de chaque élément.

Ce projet consiste en un tube dupliqué via le « cloner », dont le « shader » affecte l'échelle, grâce à un bruit, ici le « Wavy Turbulence ». Chaque tube voit alors son échelle varier et donne alors naissance à une animation aléatoire.



Animation Tubes : [https://www.ludwigdejonckheere.com/noise/assets/tubes\\_anim.mp4](https://www.ludwigdejonckheere.com/noise/assets/tubes_anim.mp4)

# SITE WEB

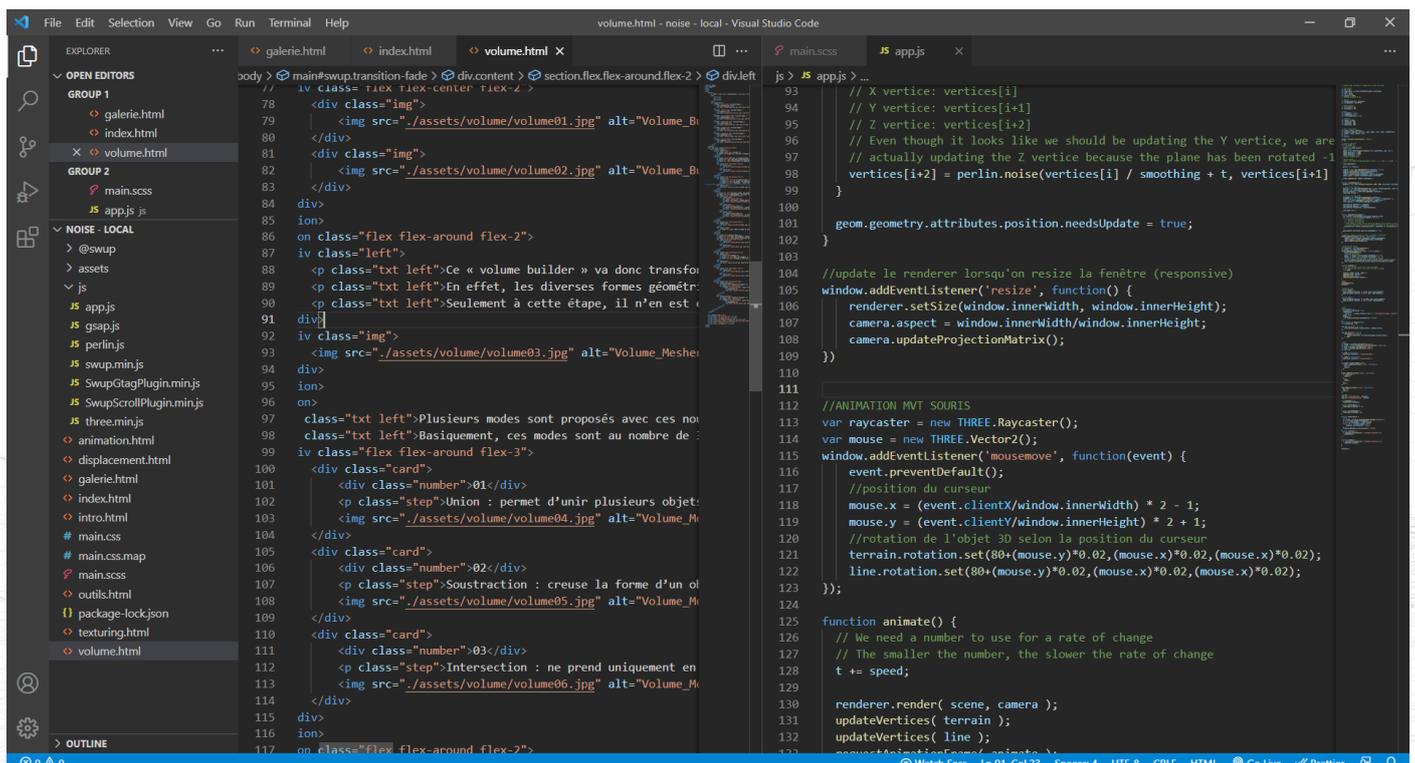
L'objectif final est de créer un site didactique reprenant les différents chapitres ci-dessus. Ce site est entièrement codé manuellement, en HTML, CSS et Javascript.

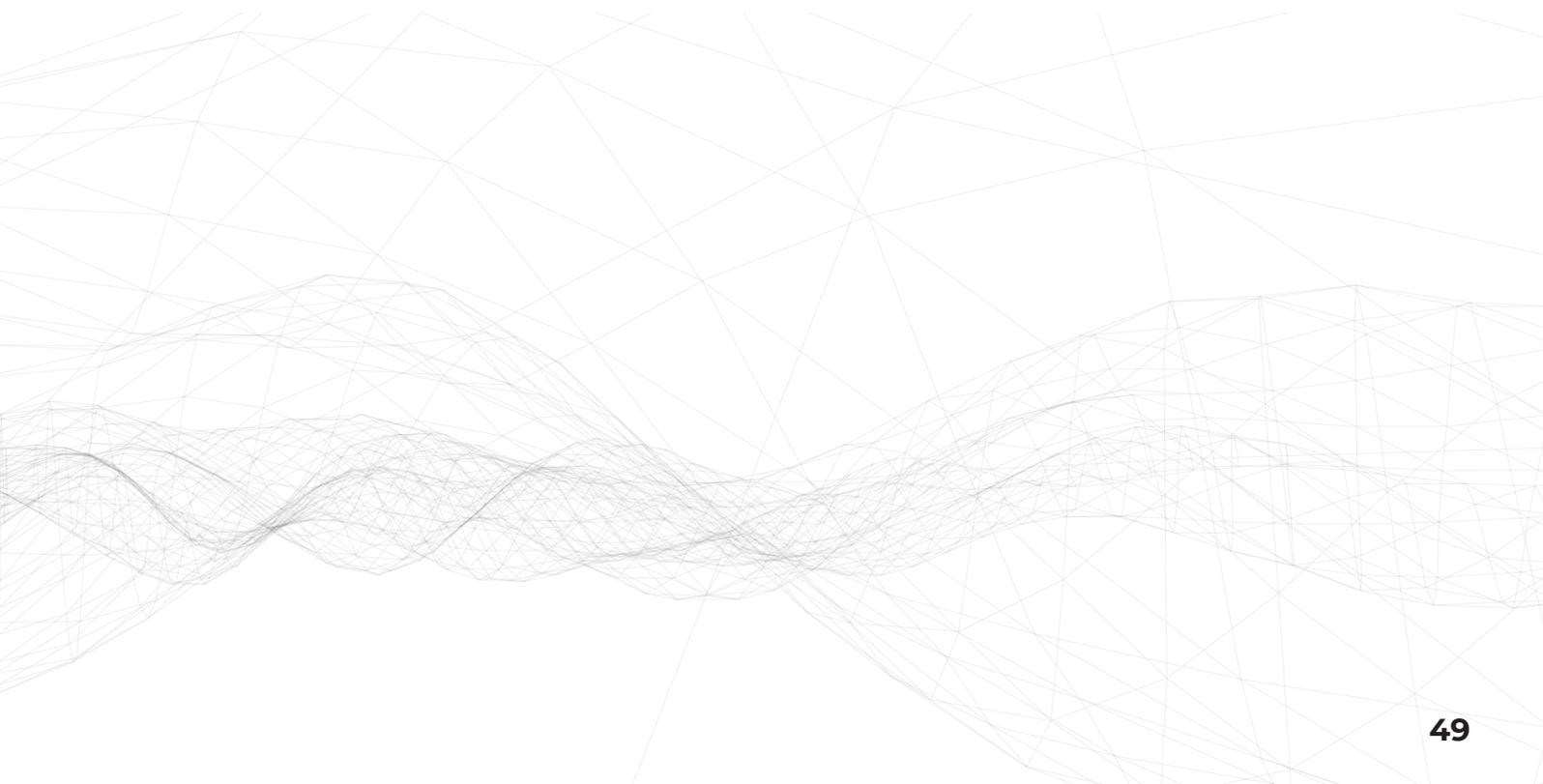
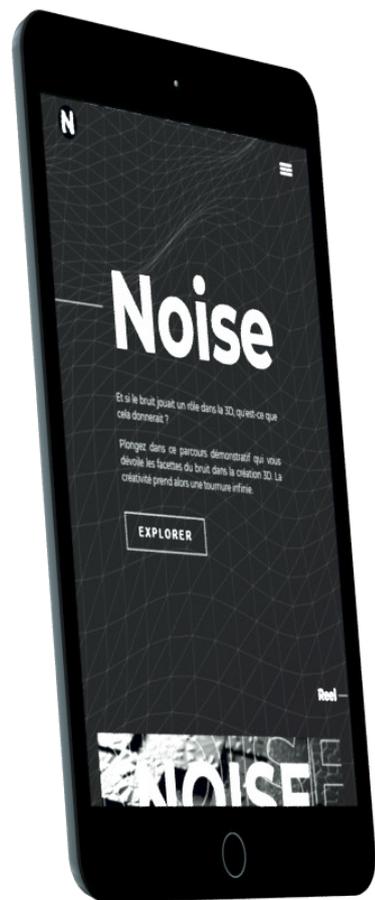
Ce site est néanmoins destiné à un public maîtrisant un minimum Cinema4D et Octane, et qui désire en apprendre davantage sur le bruit et ses usages.

L'outil principal utilisé est Visual Studio Code, qui est complet en termes de code, et présente certaines extensions intéressantes. Parmi elles :

- « **Live Server** » : permet d'actualiser directement le site lors d'une sauvegarde et simule un serveur local.
- « **Live Sass Compiler** » : compile directement le fichier SCSS en CSS lors d'une sauvegarde.
- « **Auto Rename Tag** » : si une balise HTML est modifiée, sa balise fermante change aussi.
- « **Minify** » : lorsque les fichiers CSS et Javascript sont terminés, Minify peut les condenser pour un poids plus léger et une meilleure rapidité de lecture, en supprimant les espaces et les commentaires.

Visual Studio Code est aussi très personnalisable et adopte un code couleur de base qui rend le code très lisible et clair.

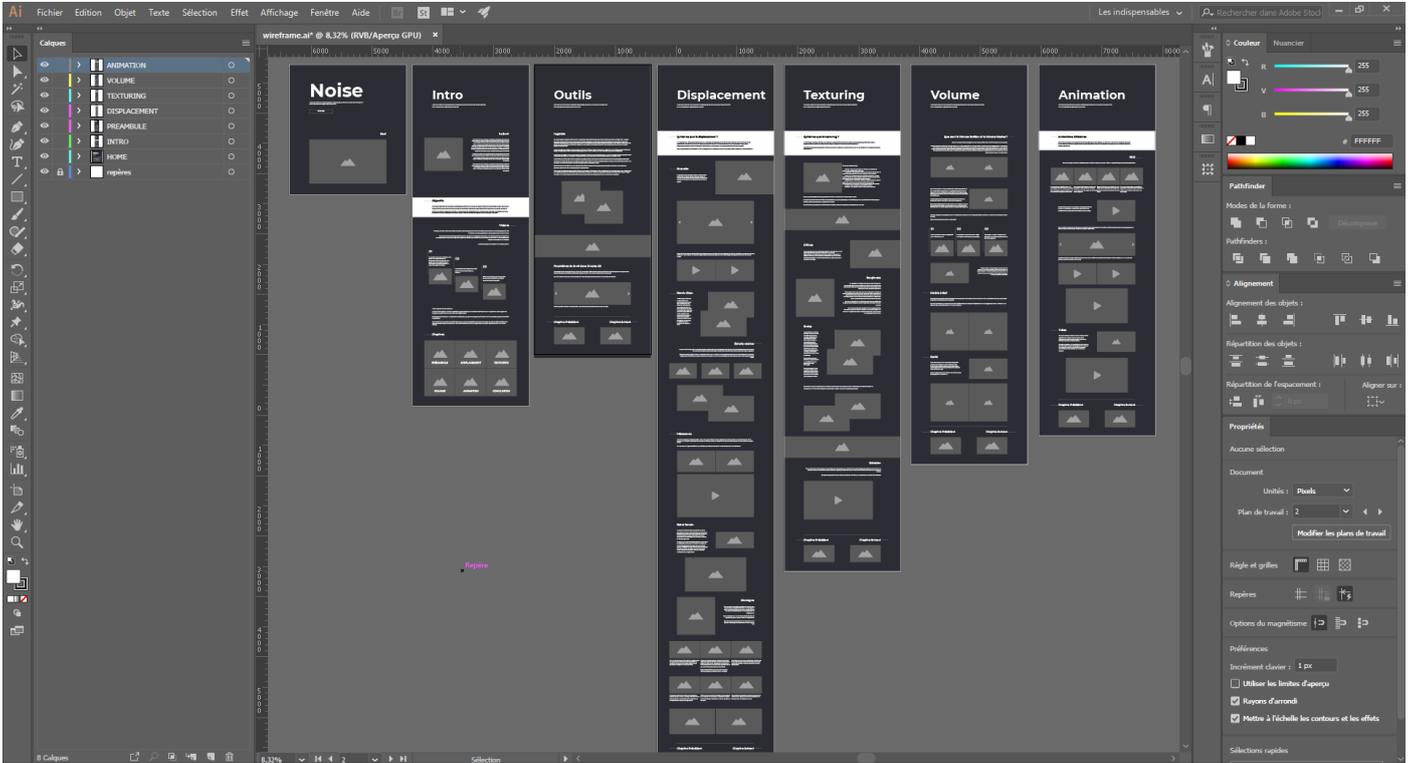




# Wireframes

## Annexes p.82

Avant de se lancer dans le code, il est important de créer un squelette du site. La création des différents wireframes a donc lieu sur Adobe Illustrator. Ces squelettes ne sont pas forcément respectés au pixel près, mais sont plutôt une idée globale de ce à quoi ressemblera le site par la suite.



## Responsive

Il est de coutume de créer des wireframes pour les différentes largeurs d'écran. Cependant, dans le cas présent et étant donné la structure peu complexe du site, les wireframes pour tablettes, mobiles et petites largeurs n'ont pas lieu d'être.

L'adaptation du site est directement réalisée via le code, où il ne s'agit que de changer quelques « flexbox » et largeurs d'éléments, sans complications.

# Noise

Et si le bruit jouait un rôle dans la 3D, qu'est-ce que cela donnerait ?

Plongez dans ce parcours démonstratif qui vous dévoile les facettes du bruit dans la création 3D. La créativité prend alors une tournure infinie.

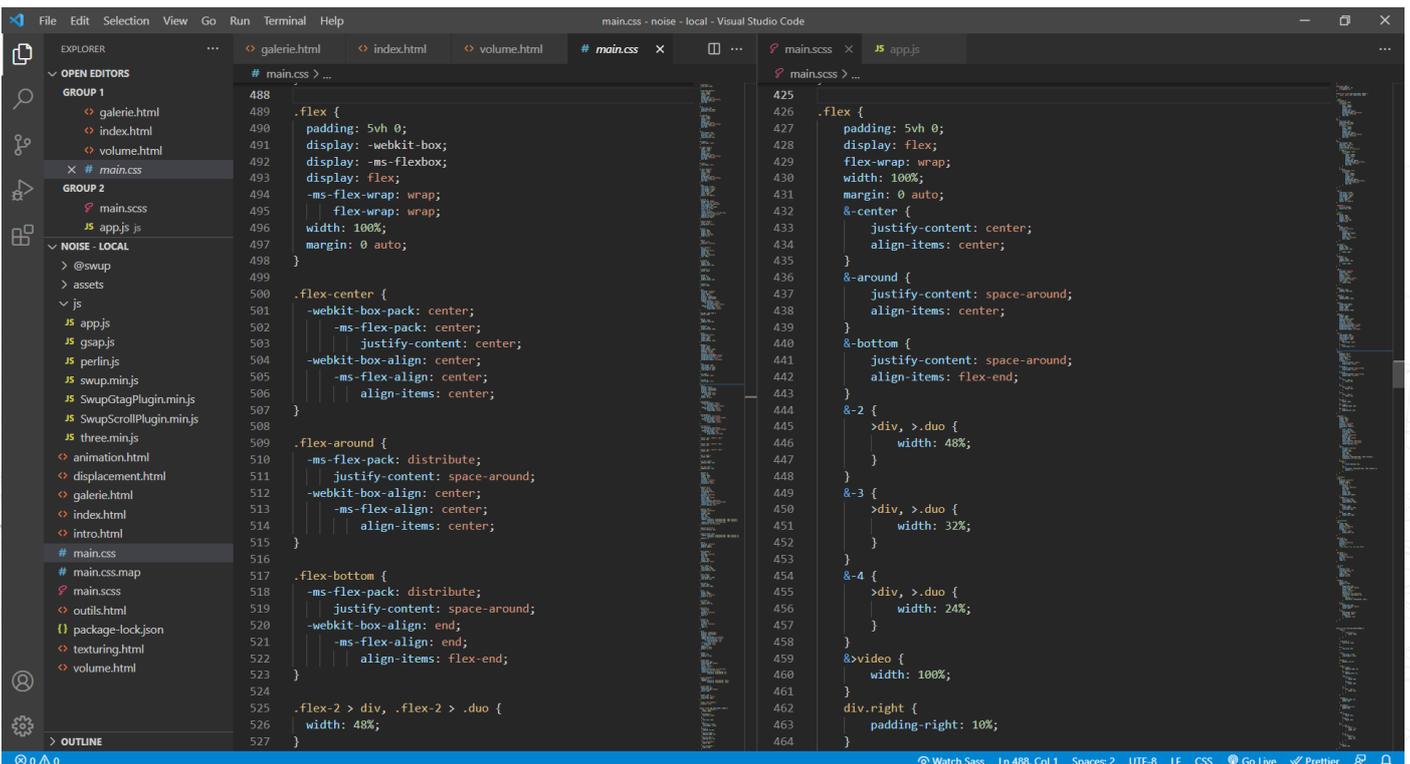
EXPLORER

Reel

## Code

La partie code du site n'a rien de spécial, hormis l'utilisation du SCSS. C'est une forme améliorée du CSS, où il est possible de créer des variables, des « mixins », ou encore de concaténer rapidement des classes ou balises html.

L'extension « Live SASS Compiler » compile alors le code SCSS dans un fichier CSS classique avec la bonne orthographe. Les variables sont transformées et les concaténations sont réalisées directement.



```
main.scss - noise - local - Visual Studio Code
# main.scss > ...
488
489 .flex {
490   padding: 5vh 0;
491   display: -webkit-box;
492   display: -ms-flexbox;
493   display: flex;
494   -ms-flex-wrap: wrap;
495   flex-wrap: wrap;
496   width: 100%;
497   margin: 0 auto;
498 }
499
500 .flex-center {
501   -webkit-box-pack: center;
502   -ms-flex-pack: center;
503   justify-content: center;
504   -webkit-box-align: center;
505   -ms-flex-align: center;
506   align-items: center;
507 }
508
509 .flex-around {
510   -ms-flex-pack: distribute;
511   justify-content: space-around;
512   -webkit-box-align: center;
513   -ms-flex-align: center;
514   align-items: center;
515 }
516
517 .flex-bottom {
518   -ms-flex-pack: distribute;
519   justify-content: space-around;
520   -webkit-box-align: end;
521   -ms-flex-align: end;
522   align-items: flex-end;
523 }
524
525 .flex-2 > div, .flex-2 > .duo {
526   width: 48%;
527 }
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
290
```

Perlin Noise Class  
David Johnson + Follow

HTML

CSS

JS

```

33 }
34
35 * noise(xin, yin) {
36     var n0, n1, n2; // Noise contributions from the three corners
37     // Skew the input space to determine which simplex cell we're in
38     var F2 = 0.5*(Math.sqrt(3.0)-1.0);
39     var s = (xin+ysin)*F2; // Hairy factor for 2D
40     var i = Math.floor(xin+s);
41     var j = Math.floor(yin+s);
42     var G2 = (3.0-Math.sqrt(3.0))/6.0;
43     var t = (i+j)*G2;
44     var X0 = i-t; // Unskew the cell origin back to (x,y) space
45     var Y0 = j-t;
46     var x0 = xin-X0; // The x,y distances from the cell origin
47     var y0 = yin-Y0;
48     // For the 2D case, the simplex shape is an equilateral triangle.
49     // Determine which simplex we are in.
50     var i1, j1; // Offsets to determine (middle) corner of simplex in (i,j) coords
51     if(x0>y0) {i1=1; j1=0;} // lower triangle, XY order: (0,0)->(1,0)->(1,1)
52     else {i1=0; j1=1;} // upper triangle, YX order: (0,0)->(0,1)->(1,1)
53     // A step of (1,0) in (i,j) means a step of (1-c,-c) in (x,y), and
54     // a step of (0,1) in (i,j) means a step of (-c,1-c) in (x,y), where
55     // c = (3-sqrt(3))/6
56     var x1 = x0 - i1 + G2; // Offsets for middle corner in (x,y) unskewed coords
57     var y1 = y0 - j1 + G2;
58     var x2 = x0 - 1.0 + 2.0 * G2; // Offsets for last corner in (x,y) unskewed coords
59     var y2 = y0 - 1.0 + 2.0 * G2;
60     // Work out the hashed gradient indices of the three simplex corners
61     var ii = i & 255;
62     var jj = j & 255;
63     var gi0 = this.perm[ii+this.perm[jj]] % 12;
64     var gi1 = this.perm[ii+i1+this.perm[jj+j1]] % 12;
65     var gi2 = this.perm[ii+1+this.perm[jj+1]] % 12;
66     // Calculate the contribution from the three corners
67     var t0 = 0.5 - x0*x0-y0*y0;
68     if(t0<0) n0 = 0.0;
69     else {
70         t0 *= t0;
71         n0 = t0 * t0 * this.dot(this.grad3[gi0], x0, y0); // (x,y) of grad3 used for 2D gradient

```

Console Assets Comments Shortcuts

## Three.js

Afin d'adapter le site au thème concerné, à savoir le bruit dans la 3D, l'idée est d'avoir une 3D animée aléatoirement en fond, via un bruit. Le contenu du site est donc en avant-plan et n'est aucunement perturbé par l'animation qui a lieu en fond de site. Il faut aussi garder à l'esprit que cette animation ne doit pas être trop présente, au risque de trop capter l'attention de l'utilisateur.

Cette fonctionnalité est donc traitée grâce à Three.js, une librairie Javascript qui permet d'intégrer de la 3D dans un navigateur, avec WebGL. L'animation consiste à animer une surface plane moyennement subdivisée, à laquelle un bruit agit sur les points de la géométrie, faisant monter ou descendre ces points.

Dans un premier temps, le bruit est généré depuis un algorithme. Ce dernier se résume en un fichier Javascript dans lequel se trouve une adaptation du code du bruit de Perlin.

L'algorithme du bruit ainsi que le code Javascript l'intégrant dans le navigateur avec Three.js sont des codes repris de David Johnson, utilisateur du site CodePen, une plateforme permettant aux utilisateurs de créer et partager des bouts de codes.

Lorsque le script est intégré au code HTML, l'instance du bruit est ajoutée dans le fichier Javascript principal, et une fonction est appelée pour affecter ce bruit aux sommets de la géométrie.

```
function updateVertices(geom) {  
  var vertices = geom.geometry.attributes.position.array;  
  for ( var i = 0; i <= vertices.length; i += 3 ) {  
    vertices[i+2] = perlin.noise(vertices[i] / smoothing + t, vertices[i+1] / smoothing + t) * amplitude;  
  }  
  
  geom.geometry.attributes.position.needsUpdate = true;  
}
```

```
var perlin = new Perlin();
```

Après avoir intégré le script du bruit, la scène 3D peut être créée. Il s'agit d'une simple surface plane subdivisée. Pour le côté esthétique, il est question d'animer les arêtes de la géométrie, et non pas les polygones. L'objet « PlaneGeometry » qu'offre la bibliothèque Three.js n'est cependant pas adaptée puisqu'il faut ici exploiter les sommets de la surface, ce qui n'est pas possible avec cet objet. Il faut donc choisir l'option du « PlaneBufferGeometry », qui est une représentation du quadrillage, incluant polygones, sommets et arêtes, et qui permet donc d'être exploité par le bruit de Perlin (il s'agit de déplacer les multiples sommets que composent la géométrie via le bruit).

Ses paramètres représentent respectivement la largeur, la longueur, le nombre de subdivisions en largeur et le nombre de subdivisions en longueur. Plus ces derniers nombres sont élevés, plus la géométrie est lisse.

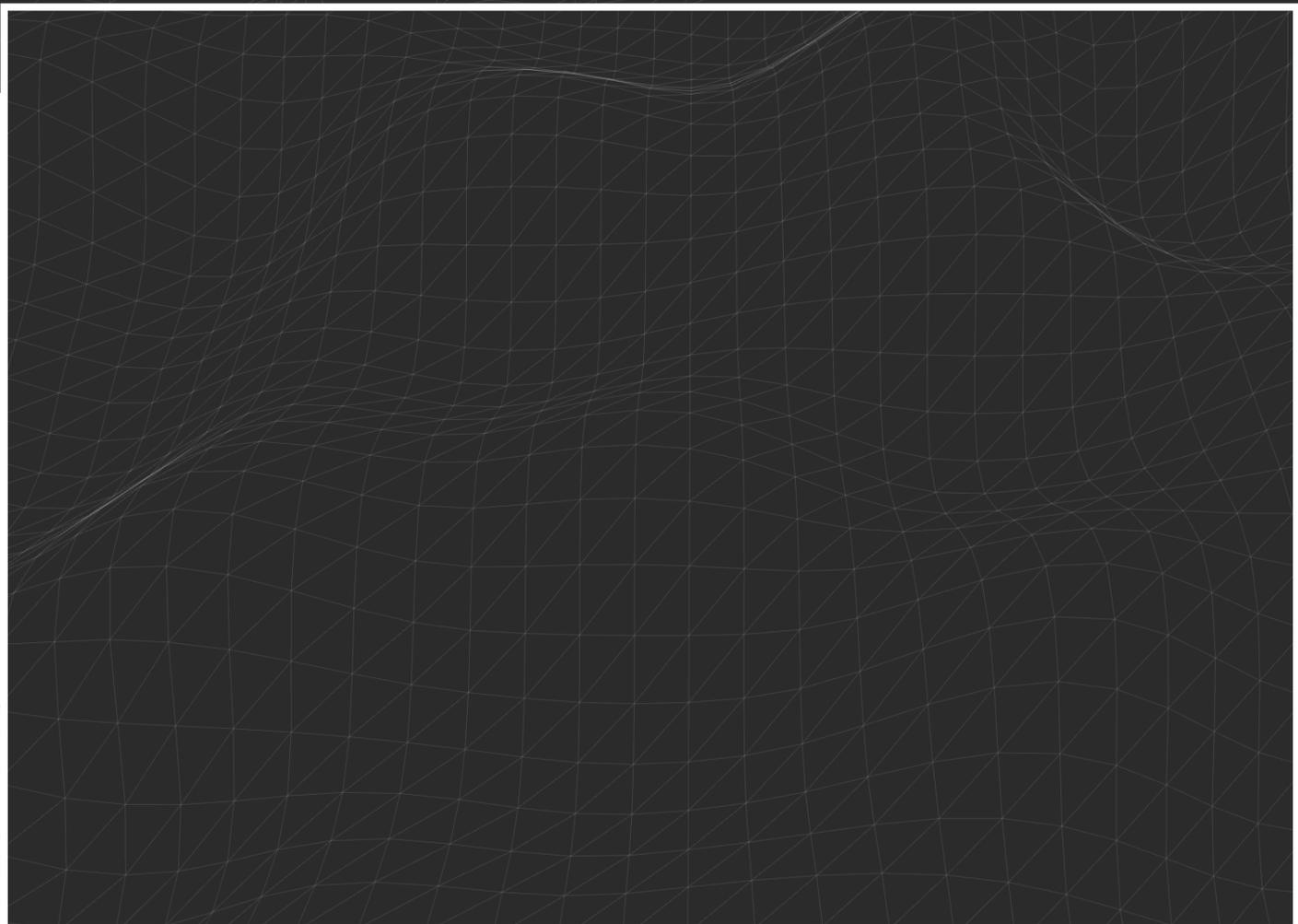
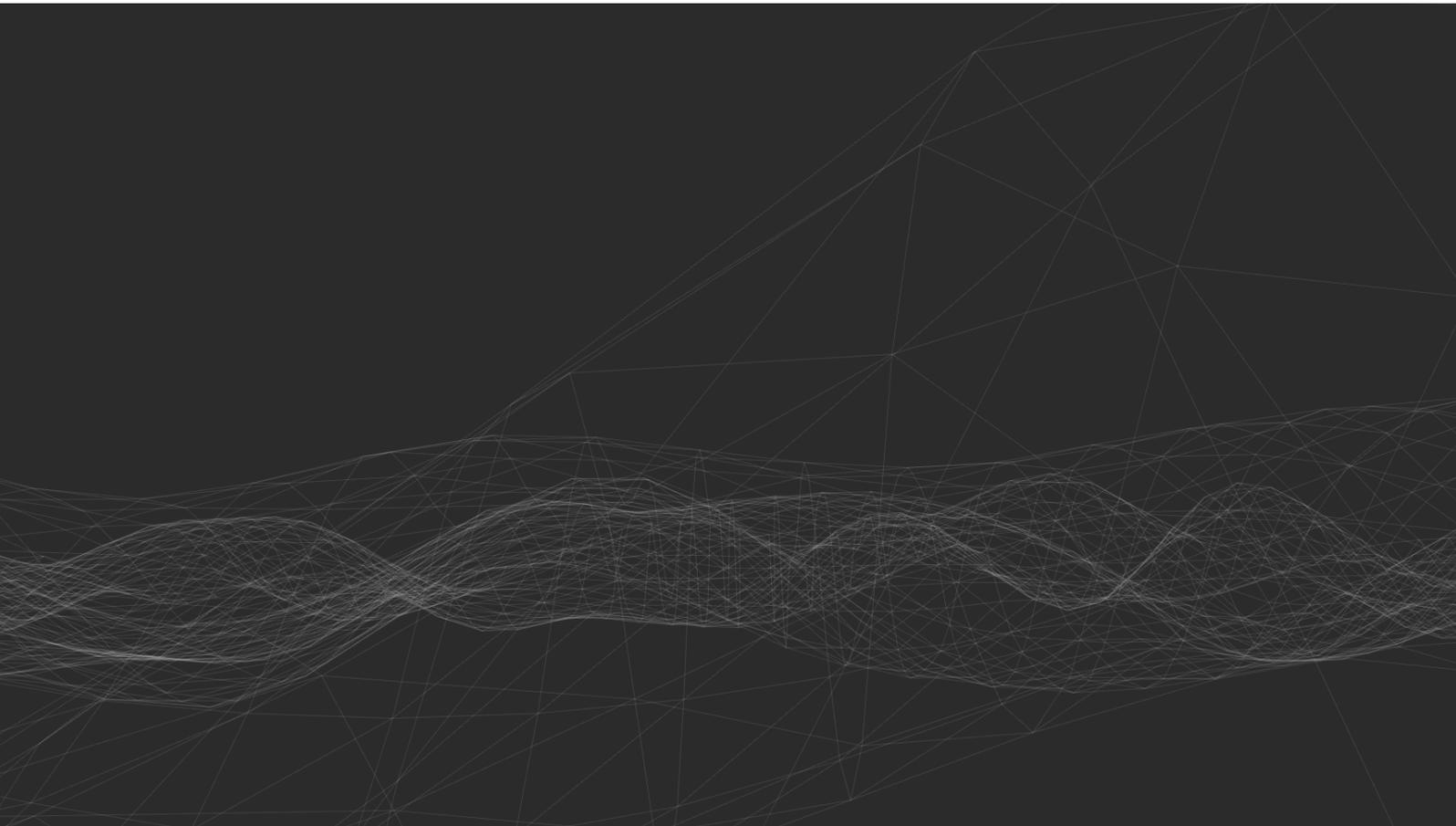
Le wireframe est quant à lui généré depuis la fonction « WireframeGeometry ». C'est une sorte de seconde géométrie créée sur base de la surface plane.

Afin de lui attribuer un matériau, il faut faire appel au « LineBasicMaterial », qui s'adapte uniquement aux wireframes.

Lorsque la géométrie et le matériau sont initialisés, il ne reste qu'à les unifier dans un « lineSegments ».

```
geometry = new THREE.PlaneBufferGeometry( 2000, 2000, divisionsX, divisionsY );
```

Le résultat rend alors une sorte de vague rendue en wireframe, directement dans le navigateur.



## SWUP

La librairie Javascript SWUP permet des transitions de pages web sans rafraîchissement, un peu à la manière de l'AJAX. Cette librairie prend en compte la navigation via l'historique (boutons « précédent » et « suivant » du navigateur). Elle gère aussi la mise en cache et les autres scripts Javascript présents sur le site.

Le fonctionnement est simple ; il suffit de donner un ID spécifique à une balise HTML pour qu'elle soit reconnue par le script SWUP (ici l'ID #swup). Cet ID doit être présent sur les différentes pages HTML dont le contenu doit être modifié.

Dans ce cas-ci il est question de changer le contenu de toute la page, hormis la navigation et le canvas qui contient la 3D en fond. Le contenu HTML de chaque page doit donc logiquement se trouver dans cette balise SWUP, et la navigation ainsi que le canvas ne possède pas cet ID, dans le but de rester à l'écran..

Différentes options sont aussi disponibles, comme le scroll en haut de chaque nouvelle page lors de la transition, la mise en cache ou non, l'animation des transitions lors du clic sur le bouton « précédent » ou « suivant », etc.

```
<!DOCTYPE html>
<html lang="fr" class="swup-enabled">
  <head>...</head>
  <body> == $0
    <header class="fade"> flex
      <a href="/noise/index.html" onclick="up()">...</a>
      <nav>...</nav>
      <div class="menu">...</div> flex
    </header>
    <main id="swup" class="transition-fade" data-swup="0">...</main>
    <div class="three">
      <canvas width="1254" height="980" style="display: block; width: 1254px; height: 980px;
      ">
    </div>
    <script src="./js/three.min.js"></script>
    <script src="./js/swup.min.js"></script>
    <script src="./js/SwupScrollPlugin.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/gsap/3.6.1/gsap.min.js"></script>
    <script src="./js/perlin.js"></script>
    <script src="./js/app.min.js"></script>
  </body>
</html>
```

```

//SLIDER
swup.on('contentReplaced', showSlides);
swup.on('willReplaceContent', unloadCarousel);

var slideIndex = 1;
showSlides(slideIndex);

function plusSlides(n) {
  showSlides(slideIndex += n);
}

function currentSlide(n) {
  showSlides(slideIndex = n);
}

function showSlides(n) {
  var i;
  var slides = document.getElementsByClassName("mySlides");
  if (n > slides.length) {slideIndex = 1}
  if (n < 1) {slideIndex = slides.length}
  for (i = 0; i < slides.length; i++) {
    slides[i].style.display = "none";
  }
  slides[slideIndex-1].style.display = "block";
}

function unloadCarousel() {
  if (document.querySelector('.slideshow-container')) {
    showSlides.unload();
  }
}

showSlides();

```

Certaines pages contiennent des carrousels en Javascript. De ce fait, ces fonctionnalités entrent en collision avec SWUP. En effet, le script d'un carrousel s'exécute lors de l'arrivée sur la page. Cependant, si l'utilisateur quitte cette dernière et y revient plus tard, le carrousel n'est plus fonctionnel.

La solution est de « décharger » le script du carrousel chaque fois que l'utilisateur quitte la page, et le charge à nouveau lors de son retour sur celle-ci.

Le carrousel est donc stocké dans une fonction propre, et une seconde fonction de déchargement du carrousel est initiée. La librairie SWUP prend en charge le reste grâce à des fonctions spécifiques.

## GSAP

GSAP, ou GreenSock Animation Plugin, est une librairie Javascript facilitant des animations complexes d'un site, que ce soit au niveau des propriétés CSS ou, dans cet exemple, un mouvement de caméra de la 3D lors d'un clic sur un lien.

```

//ANIMATION GSAP
function down() {
  gsap.to(camera.position, 3, {y:80, ease: Expo.easeOut});
  gsap.to(camera.rotation, 3, {x:0, ease: Expo.easeOut});
};

function up() {
  gsap.to(camera.position, 3, {y:1000, ease: Expo.easeOut});
  gsap.to(camera.rotation, 3, {x:-1, ease: Expo.easeOut});
};

```

## Javascript

L'intégration du bruit de Perlin ainsi que de la 3D animée sont des codes repris de David Johnson sur CodePen. De plus, les scripts de carrousels sont des adaptations de codes repris sur W3School. Certains éléments de ces carrousels ont été supprimés ou adaptés au site. Hormis ceci, les autres fonctionnalités sont codées à la main de manière personnelle.

Parmi ces fonctionnalités, il y a premièrement la fonction du menu qui apparait et disparaît à droite de l'écran. Après avoir récupéré les éléments HTML nécessaires à la fonction du menu, il s'agit simplement de créer une fonction qui se base sur le modulo. En effet, un compteur est initialisé à 0. Si le modulo 2 de ce compteur vaut 0, le menu apparait. Si son modulo est différent de 0, le menu se rétracte.

```
//MENU
var burger = document.querySelector('nav');
var links = Array.from(document.querySelectorAll('.menu a'));
var menu = document.querySelector('.menu');
var home = document.querySelector('#home');
var nbr = 0;

var hide = function() {
  menu.style.transform = "translateX(100%)";
}
var appear = function() {
  menu.style.transform = "translateX(0%)";
}

//clic sur le hamburger
burger.addEventListener('click', function() {
  if (nbr%2==0) {
    appear();
  }
  else {
    hide();
  }
  nbr++;
})

//clic sur un lien de la nav
for (i=0; i<links.length; i++) {
  links[i].addEventListener('click', function() {
    nbr = 0;
    hide();
  })
}

//clic sur le logo Home
home.addEventListener('click', function() {
  nbr = 0;
  hide();
})
```

Lors du clic sur le hamburger, le menu apparait en premier lieu étant donné la valeur 0 du nombre, suivi de l'incrémement de ce dernier, qui vaut alors 1. Cette fonction est utile si le bouton hamburger est cliqué plusieurs fois de suite. Le compteur augmente donc, avec à le menu qui apparait un nombre sur 2.

Si un lien est ensuite cliqué, le compteur est réinitialisé à 0, et le menu disparaît. Ceci a pour effet d'avoir un compteur neuf lors de l'arrivée de l'utilisateur sur une nouvelle page. Cette fonction est aussi attribuée au logo d'accueil du site. En ce qui concerne le menu, le script comprend une boucle, pour affecter la fonction à chaque lien que composent le menu.



Une animation de la 3D est aussi créée lorsque le curseur bouge. La 3D en fond s'oriente alors en fonction de la position du curseur. Ceci est rendu possible grâce à la fonction de Raycaster de « Three.js », qui demande aussi la fonction « Vector2 ». Cependant, ceci rend impossible la sélection du texte sur le site.

```
//ANIMATION MVT SOURIS
var raycaster = new THREE.Raycaster();
var mouse = new THREE.Vector2();
window.addEventListener('mousemove', function(event) {
  event.preventDefault();
  //position du curseur
  mouse.x = (event.clientX/window.innerWidth) * 2 - 1;
  mouse.y = (event.clientY/window.innerHeight) * 2 + 1;
  //rotation de l'objet 3D selon la position du curseur
  terrain.rotation.set(80+(mouse.y)*0.02, (mouse.x)*0.02, (mouse.x)*0.02);
  line.rotation.set(80+(mouse.y)*0.02, (mouse.x)*0.02, (mouse.x)*0.02);
});
```

Enfin, le dernier script correspond à la « lightbox » présente sur la galerie. Cette fonction permet à l'utilisateur d'agrandir l'image cliquée, et ensuite naviguer sur cette « lightbox » via des boutons. Cette fonctionnalité est en partie reprise du carrousel de base. Il s'agit donc d'ouvrir un carrousel dans une « lightbox ».

En premier lieu, il faut donc créer un conteneur en fin de DOM qui sera la visionneuse. Ce conteneur contient une DIV qui réceptionnera par la suite les images, ainsi que les 2 boutons de navigation, et un bouton de fermeture permettant de quitter la « lightbox ».

```
<section class="lightbox fade">
  <div class="close fade">&times;</div>
  <div class="lightbox-slideshow">
    <div class="lightbox-mySlides fade"></div>
    <a class="prev">#10094;</a>
    <a class="next">#10095;</a>
  </div>
</section>
```

L'ensemble de la section est désactivée grâce au « display : none ; », et sera changée en « display : flex ; » lors du clic sur une image de la galerie (« flex » et non pas « block » pour pouvoir centrer le conteneur de l'image directement dans la visionneuse). Après avoir établi ceci et créer le CSS qui est en relation vient le script.

L'idée est que, lors du clic sur une image de la galerie :

- 1) Un clone de cette image est créé
- 2) Ce clone est ajouté dans le conteneur de la visionneuse
- 3) La navigation en haut de l'écran disparaît
- 4) La visionneuse apparaît

Pour que la fonction prenne cours sur chaque image et vidéo de la galerie, cette fonction est initialisée dans une boucle dont le compteur maximum est le nombre d'éléments que compte la galerie.

```
function lightbox() {
  //tableau contenant les images et vidéos de la galerie
  var galerieImg = Array.from(document.querySelectorAll('.column img, .column video'));
  //lightbox
  var lightbox = document.querySelector('.lightbox');
  //conteneur réceptionnant les images dans la lightbox
  var lightboxInner = document.querySelector('.lightbox .lightbox-mySlides');
  //boutons
  var before = document.querySelector('.lightbox .prev');
  var after = document.querySelector('.lightbox .next');
  var close = document.querySelector('.close');

  for (var k=0; k<galerieImg.length; k++) {
    //clic sur une image
    galerieImg[k].addEventListener('click', function() {
      //désactivation de la navigation en haut de la page
      document.querySelector('header').style.display = "none";
      //récupération de l'index de l'image cliquée dans le tableau
      var galerieIndex = galerieImg.indexOf(this);
      //clonage de l'image
      var clone = galerieImg[galerieIndex].cloneNode();
      //ajout du clone dans le conteneur de la lightbox
      lightboxInner.appendChild(clone);

      //attribution de la largeur de l'image dans le conteneur
      clone.style.width = "100%";
      //apparition de la lightbox qui n'est plus en display:none;
      lightbox.style.display = "flex";
```

Vient ensuite les fonctions des boutons de navigation. Lorsque ceux-ci sont cliqué :

- 1) Le clone actuel est supprimé
- 2) L'index du tableau contenant les images et vidéos est changé (-1 pour le bouton précédent, +1 pour le bouton suivant)
- 3) Un nouveau clone est créé sur base du nouvel index
- 4) Ce clone est injecté dans le conteneur de la lightbox

À noter qu'une fonction intermédiaire est initialisée, qui permet de boucler si l'utilisateur arrive en fin de galerie et clique sur le bouton suivant, ou qu'il arrive en début de galerie et qu'il clique sur le bouton précédent.

```
//clic sur le bouton suivant
after.addEventListener('click', function() {
  //suppression du clone actuel
  lightboxInner.removeChild(clone);
  //fonction de boucle du carrousel
  if (galerieIndex == (galerieImg.length)-1) {
    galerieIndex = 0;
  }
  else {
    //incrémentatation de l'index du tableau contenant les images et vidéos
    galerieIndex++;
  }
  //nouveau clone créé sur base du nouvel index
  clone = galerieImg[galerieIndex].cloneNode();
  //ajout de ce nouveau clone dans le conteneur de la lightbox
  lightboxInner.appendChild(clone);

  //nouvel attribut style pour le nouveau clone
  clone.style.width = "100%";
})
//clic sur le bouton précédent
before.addEventListener('click', function() {
  //suppression du clone actuel
  lightboxInner.removeChild(clone);
  //fonction de boucle du carrousel
  if (galerieIndex == 0) {
    galerieIndex = (galerieImg.length)-1;
  }
  else {
    //décrémentatation de l'index du tableau contenant les images et vidéos
    galerieIndex--;
  }
  //nouveua clone créé sur base du nouvel index
  clone = galerieImg[galerieIndex].cloneNode();
  //ajout de ce nouveau clone dans le conteneur de la lightbox
  lightboxInner.appendChild(clone);

  //nouvel attribut style pour le nouveau clone
  clone.style.width = "100%";
})
```

Enfin, une dernière fonction prend lieu permettant de fermer la visionneuse lors du clic sur le bouton de fermeture. Lorsqu'elle est exécutée :

- 1) La navigation en haut de l'écran réapparaît
- 2) La visionneuse disparaît
- 3) Le clone actuel est supprimé

L'entièreté de cette fonction présente aussi sa fonction de « déchargement », pour pouvoir être compatible avec SWUP.

```
//clic sur le bouton de fermeture
close.addEventListener('click', function() {
  //réapparition de la navigation en haut de l'écran
  document.querySelector('header').style.display = "block";
  //disparition de la lightbox
  lightbox.style.display = "none";
  //suppression du clone dans le conteneur de la lightbox
  lightboxInner.removeChild(clone);
})
```





# Noise

Et si le bruit jouait un rôle dans la 3D, qu'est-ce que cela donnerait ?

Plongez dans ce parcours démonstratif qui vous dévoile les facettes du bruit dans la création 3D. La créativité prend alors une tournure infinie.

EXPLORER

Reel

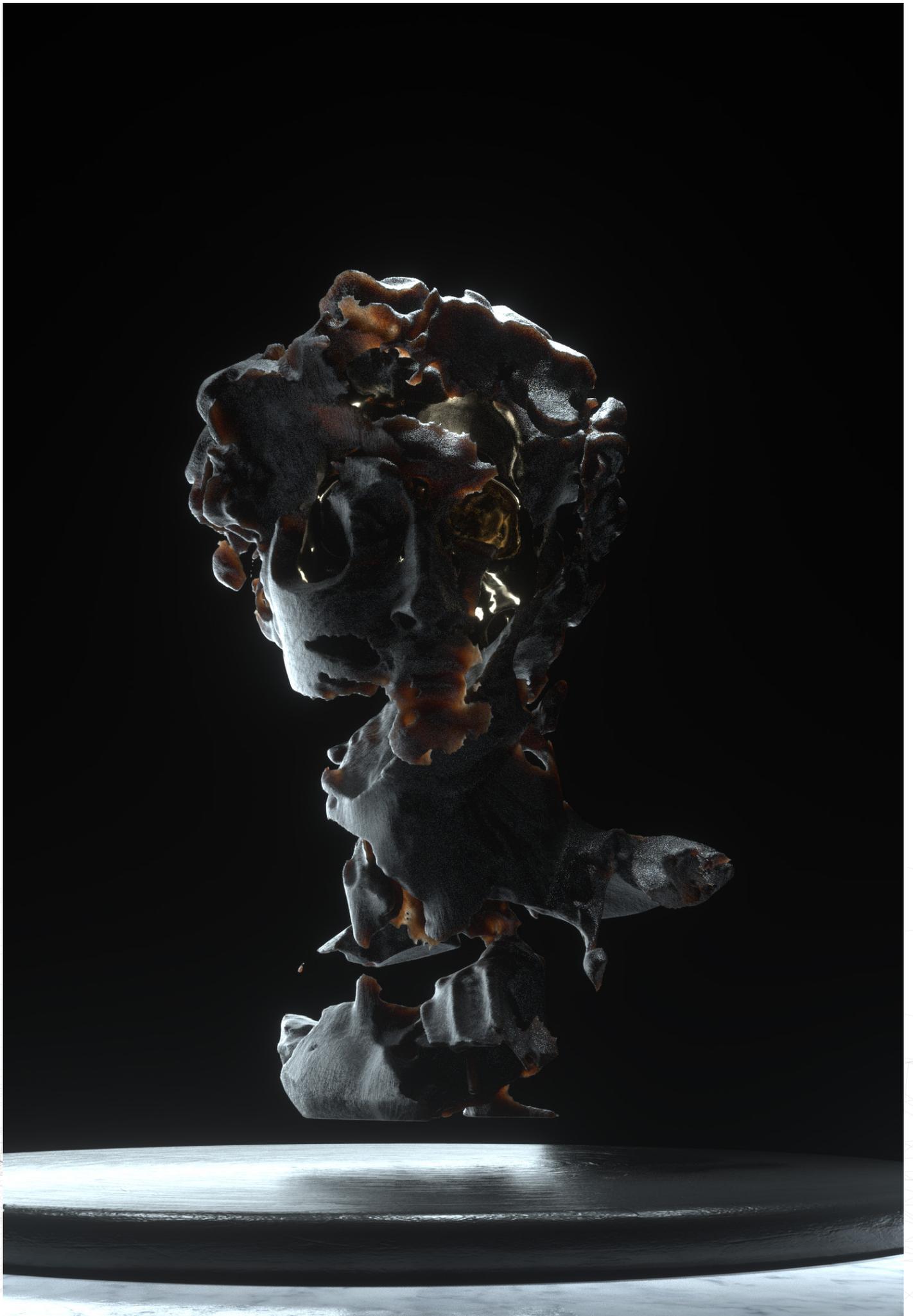
## RÉSULTATS PRATIQUES

Le produit fini consiste donc en un site web didactique répertoriant ce rapport ainsi que ses différentes étapes. Il est disponible via l'adresse suivante :

<https://ludwigdejonckheere.com/noise/index.html>

Une vidéo de type «Showreel» répertoriant tous les projets vus précédemment a été conçue. Cette vidéo est disponible à l'adresse suivante :

<https://www.ludwigdejonckheere.com/noise/assets/noise-reel.mp4>



# CONCLUS

# ION

Le bruit est désormais un outil couramment utilisé et presque indispensable dans le monde de la création 3D, que ce soit dans un logiciel de 3D comme Cinema4D ou dans le web avec Three.js par exemple.

Les possibilités qu'offre ce bruit sont infinies, seules les idées en sont sa limite. Ce travail met brièvement en lumière ses diverses utilisations, parce qu'il est impossible de couvrir complètement le sujet.

Cette innovation de Ken Perlin est révolutionnaire dans le domaine digital, mais aussi dans la création vidéoludique. En effet, nombre de jeux vidéo sont désormais générés de manière procédurale, grâce à des algorithmes se basant sur celui du bruit de Perlin.

Pour conclure, le bruit se présente comme un terrain de jeu, un bac à sable parfait pour toutes les expérimentations possibles. C'est un acteur important dans le monde du motion design, et un nouveau type de jeu qui se présente aux créateurs et motion designers. La recherche de formes ou d'animations générées aléatoirement n'a rien de plus amusante et satisfaisante.

## CONTRAINTES ET DIFFICULTÉS RENCONTRÉES

Les contraintes rencontrées ont surtout été d'ordre techniques. En effet, certaines scènes 3D sont parfois un peu surchargées, et donne lieu à des ralentissements du logiciel. De plus, certains rendus sur Octane ont pris énormément de temps pour une animation de 4 secondes, malgré des paramètres parfois minimums.

Une machine datant de 5 ans fait le travail, mais il est certain qu'un ordinateur plus récent, notamment au niveau du processeur ou de la carte graphique, aurait facilité la tâche.

Une autre contrainte concerne les logiciels utilisés. En effet, certains plugins de Cinema4D auraient été les bienvenus : la dernière version d'Octane, ou encore d'autres plugins tiers comme Forester ou XParticles. Cependant, ces plugins ont un coût un peu onéreux en tant qu'étudiant.

Enfin, certaines difficultés ont été rencontrées, mais plutôt au niveau des connaissances. Certains fonctionnements sont parfois difficiles à comprendre sur Cinema4D, au niveau des éléments « MoGraph », et une maîtrise moyenne du logiciel freine la créativité.

Il en va de même pour le code du site, où la création et la compréhension du code Javascript n'est pas chose aisée, ainsi que la maîtrise de la librairie Three.js.

Ce travail a permis d'approfondir certaines connaissances et de découvrir énormément de nouvelles choses.

# BIBLIOGRAPHIE

## THÉORIE

**Bruit de Perlin In WIKIPEDIA.** *L'encyclopédie libre* [en ligne]. 2021

Disponible sur : <[https://fr.wikipedia.org/wiki/Bruit\\_de\\_Perlin](https://fr.wikipedia.org/wiki/Bruit_de_Perlin)>

[Page consultée le 12 mai 2021]

**Perlin Noise In WIKIPEDIA.** *L'encyclopédie libre* [en ligne]. 2021

Disponible sur : <[https://en.wikipedia.org/wiki/Perlin\\_noise](https://en.wikipedia.org/wiki/Perlin_noise)>

[Page consultée le 12 mai 2021]

**Pseudo-aléatoire In WIKIPEDIA.** *L'encyclopédie libre* [en ligne]. 2019

Disponible sur : <<https://fr.wikipedia.org/wiki/Pseudo-aléatoire>>

[Page consultée le 12 mai 2021]

**Graine aléatoire In WIKIPEDIA.** *L'encyclopédie libre* [en ligne]. 2016

Disponible sur : <[https://fr.wikipedia.org/wiki/Graine\\_aléatoire](https://fr.wikipedia.org/wiki/Graine_aléatoire)>

[Page consultée le 12 mai 2021]

**Bruit de gradient In WIKIPEDIA.** *L'encyclopédie libre* [en ligne]. 2019

Disponible sur : <[https://fr.wikipedia.org/wiki/Bruit\\_de\\_gradient](https://fr.wikipedia.org/wiki/Bruit_de_gradient)>

[Page consultée le 12 mai 2021]

**Gradient In WIKIPEDIA.** *L'encyclopédie libre* [en ligne]. 2021

Disponible sur : <<https://fr.wikipedia.org/wiki/Gradient>>

[Page consultée le 12 mai 2021]

**Path tracing In WIKIPEDIA.** *L'encyclopédie libre* [en ligne]. 2020

Disponible sur : <[https://fr.wikipedia.org/wiki/Path\\_tracing](https://fr.wikipedia.org/wiki/Path_tracing)>

[Page consultée le 12 mai 2021]

**Ray tracing In WIKIPEDIA.** *L'encyclopédie libre* [en ligne]. 2021

Disponible sur : <[https://fr.wikipedia.org/wiki/Ray\\_tracing](https://fr.wikipedia.org/wiki/Ray_tracing)>

[Page consultée le 12 mai 2021]

**Cinema 4D In WIKIPEDIA.** *L'encyclopédie libre* [en ligne]. 2020

Disponible sur : <[https://fr.wikipedia.org/wiki/Cinema\\_4D](https://fr.wikipedia.org/wiki/Cinema_4D)>

[Page consultée le 12 mai 2021]

**Octane (moteur de rendu) In WIKIPEDIA.** *L'encyclopédie libre* [en ligne]. 2019

Disponible sur : <[https://fr.wikipedia.org/wiki/Octane\\_\(moteur\\_de\\_rendu\)](https://fr.wikipedia.org/wiki/Octane_(moteur_de_rendu))>

[Page consultée le 12 mai 2021]

**Displacement mapping In WIKIPEDIA.** *L'encyclopédie libre* [en ligne]. 2019

Disponible sur : <[https://fr.wikipedia.org/wiki/Displacement\\_mapping](https://fr.wikipedia.org/wiki/Displacement_mapping)>

[Page consultée le 12 mai 2021]

**Khan Academy.** *Le bruit de Perlin* [en ligne]. 2021

Disponible sur : <<https://fr.khanacademy.org/computing/computer-programming/programming-natural-simulations/programming-noise/a/perlin-noise>>

[Page consultée le 12 mai 2021]

**MANULE UTILISATEUR D'OCTANE VERSION 2019.1.** *C4D Noise* [en ligne]. 2019

Disponible sur : <<http://www.aoktar.com/octane/OCTANE%20HELP%20MANUAL.html?Noise1.html>>

[Page consultée le 12 mai 2021]

# LIBRAIRIES JAVASCRIPT

**GreenSock.** *Getting Started with GSAP* [en ligne]. 2019

Disponible sur : <<https://greensock.com/get-started>>

[Page consultée le 27 mai 2021]

**SWUP.** *Getting Started* [en ligne]. 2017

Disponible sur : <<https://swup.js.org/getting-started>>

[Page consultée le 27 mai 2021]

**Github.** *Jquery PJAX* [en ligne]. 2017

Disponible sur : <<https://github.com/defunkt/jquery-pjax>>

[Page consultée le 27 mai 2021]

**DOCUMENTATION DE THREE.JS.** *Getting Started* [en ligne]. 2021

Disponible sur : <<https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene>>

[Page consultée le 27 mai 2021]

## GÉNÉRATION DU BRUIT AVEC THREE.JS

**David Johnson In CODEPEN.** *Lunch and Learn : Wave* [en ligne]. 2021

Disponible sur : <<https://codepen.io/struct/pen/pQRBJO>>

[Page consultée le 27 mai 2021]

**David Johnson In CODEPEN.** *Perlin Noise Class* [en ligne]. 2018

Disponible sur : <<https://codepen.io/struct/pen/bQgJmx?editors=0010>>

[Page consultée le 27 mai 2021]

**Victor Vergara In CODEPEN.** *Perlin Noise* [en ligne]. 2018

Disponible sur : <<https://codepen.io/vcomics/pen/djqNrm>>

[Page consultée le 27 mai 2021]



## DISPLACEMENT

### Nuancier

#### Mouvement Cell Noise :

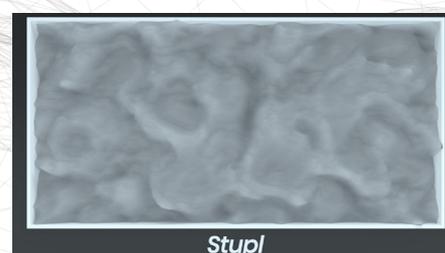
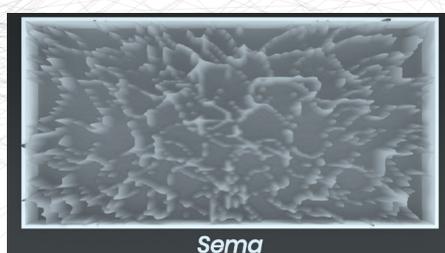
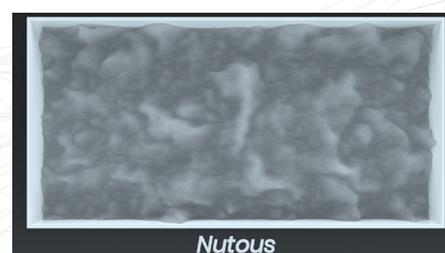
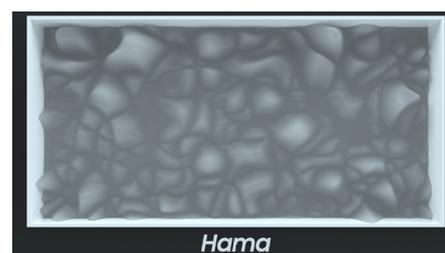
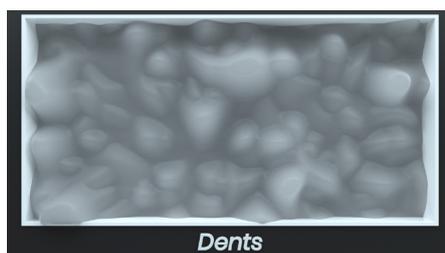
[https://www.ludwigdejonckheere.com/noise/assets/nuancier/cell\\_noise.mp4](https://www.ludwigdejonckheere.com/noise/assets/nuancier/cell_noise.mp4)

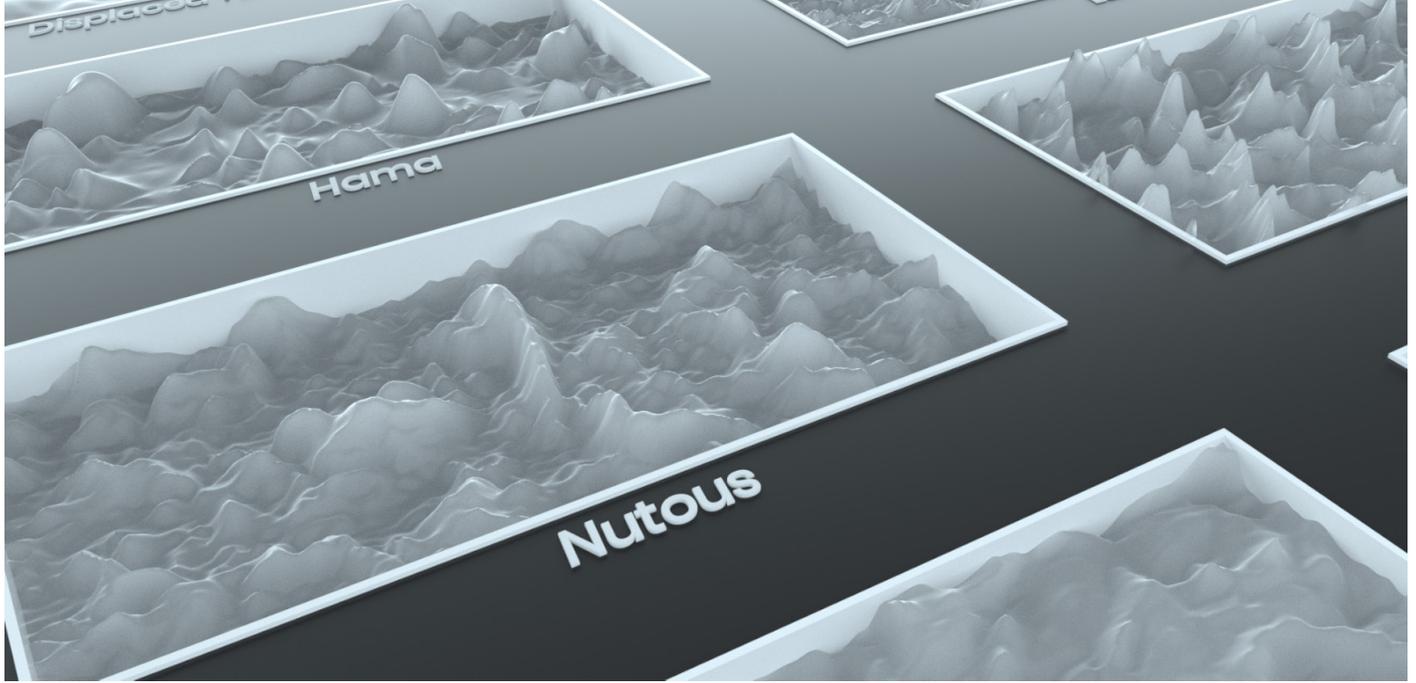
[ludwigdejonckheere.com/noise/assets/nuancier/cell\\_noise.mp4](https://www.ludwigdejonckheere.com/noise/assets/nuancier/cell_noise.mp4)

#### Mouvement Mod Noise :

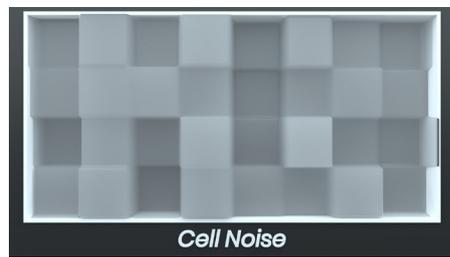
[https://www.ludwigdejonckheere.com/noise/assets/nuancier/mod\\_noise.mp4](https://www.ludwigdejonckheere.com/noise/assets/nuancier/mod_noise.mp4)

[ludwigdejonckheere.com/noise/assets/nuancier/mod\\_noise.mp4](https://www.ludwigdejonckheere.com/noise/assets/nuancier/mod_noise.mp4)





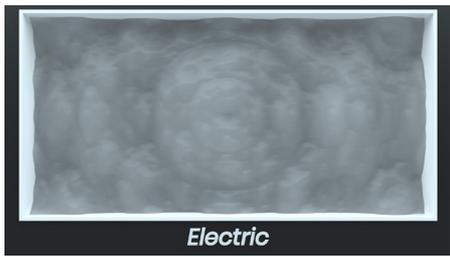
Buya



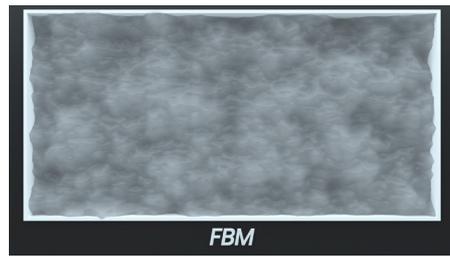
Cell Noise



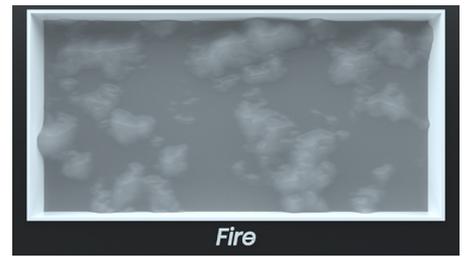
Cranal



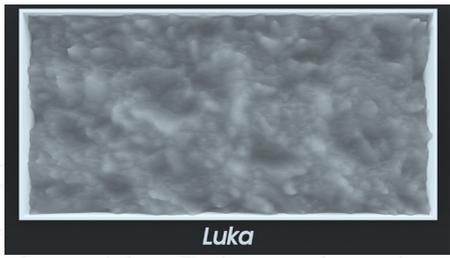
Electric



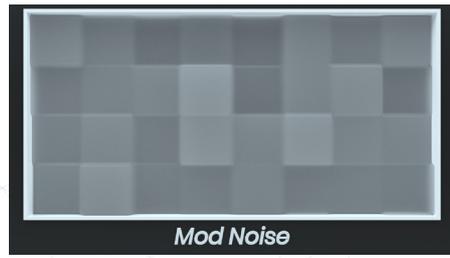
FBM



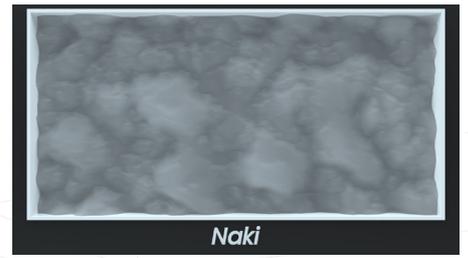
Fire



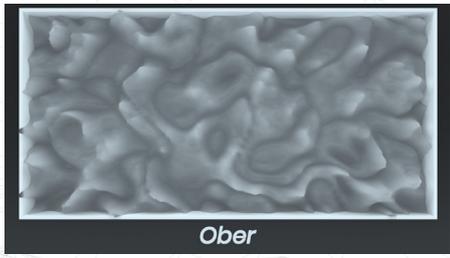
Luka



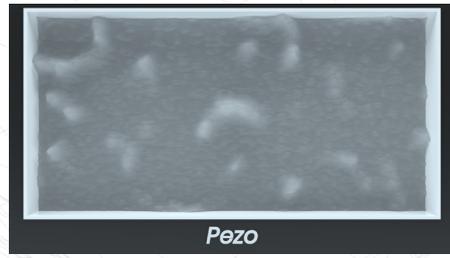
Mod Noise



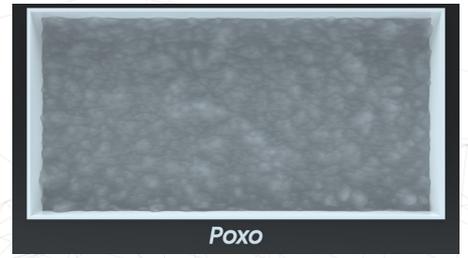
Naki



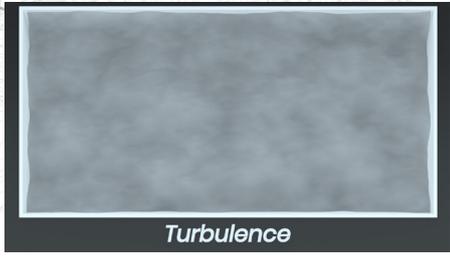
Ober



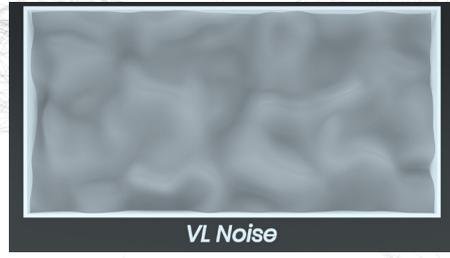
Pezo



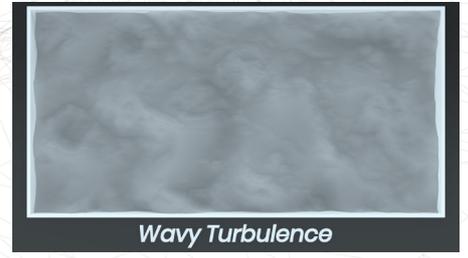
Poxo



Turbulence

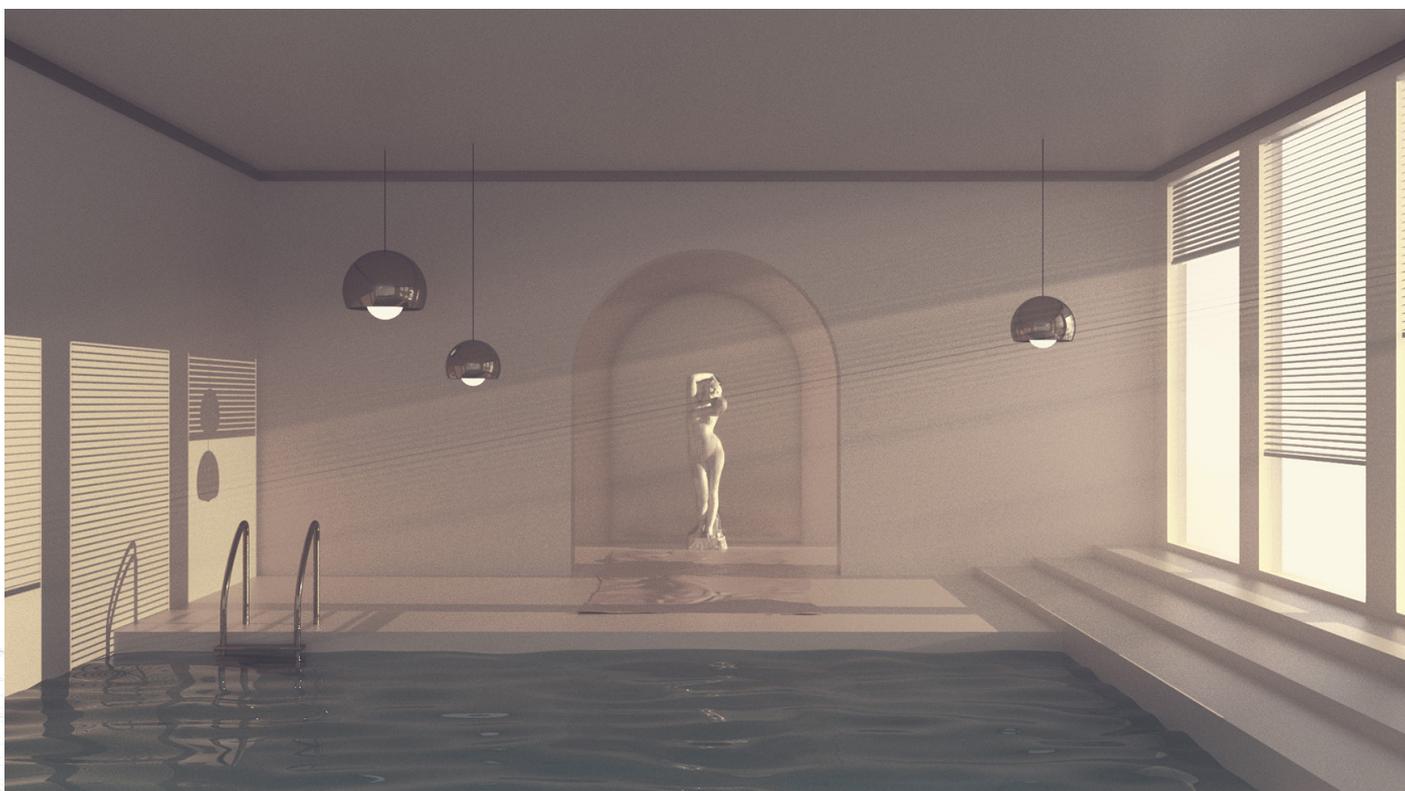
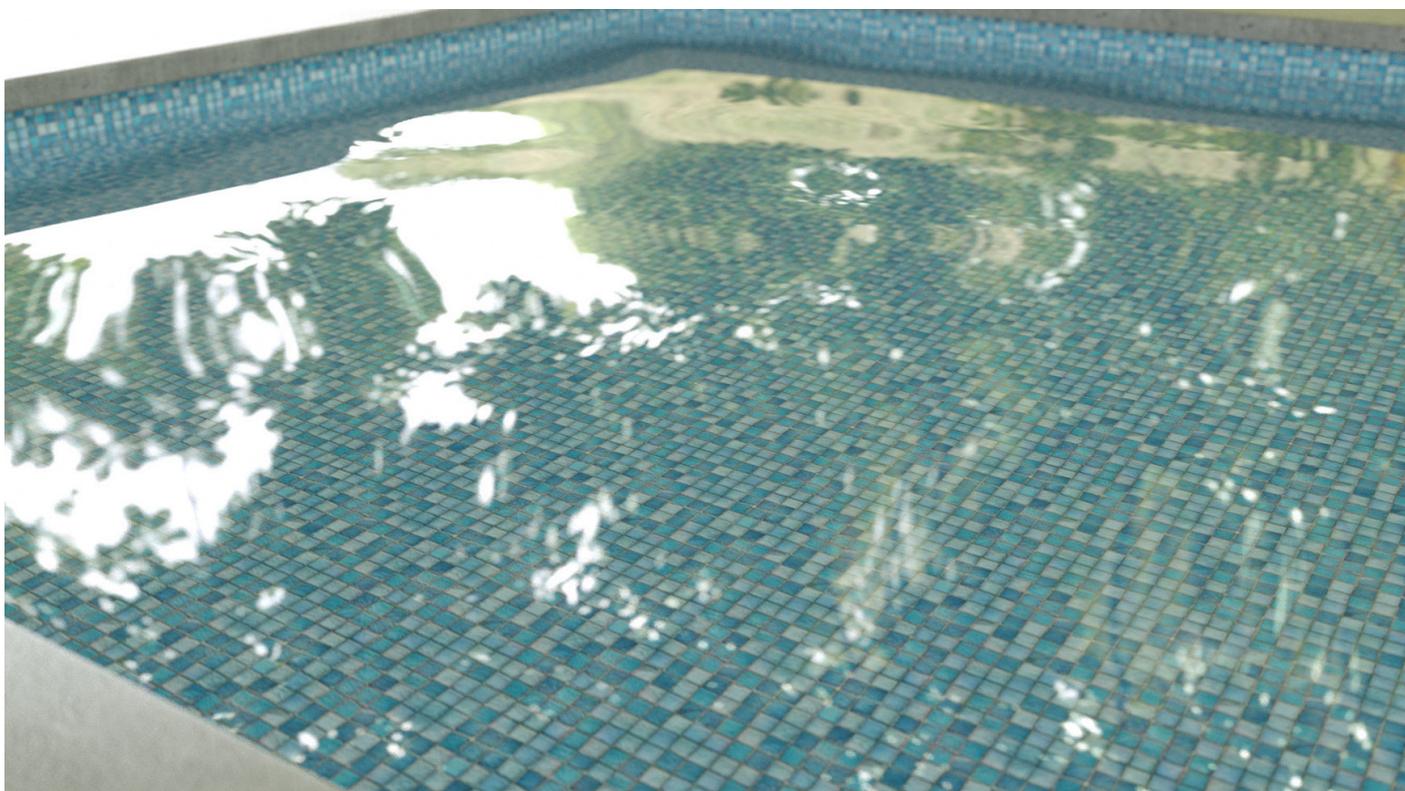


VL Noise

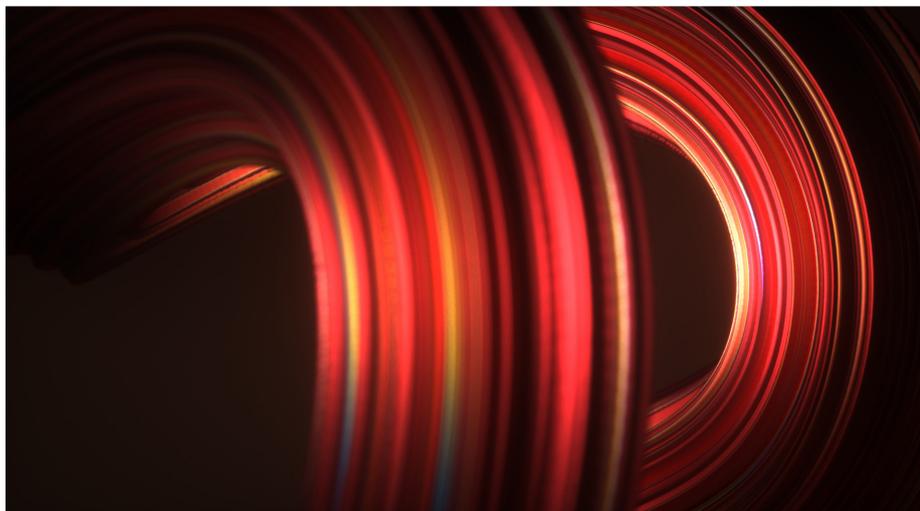
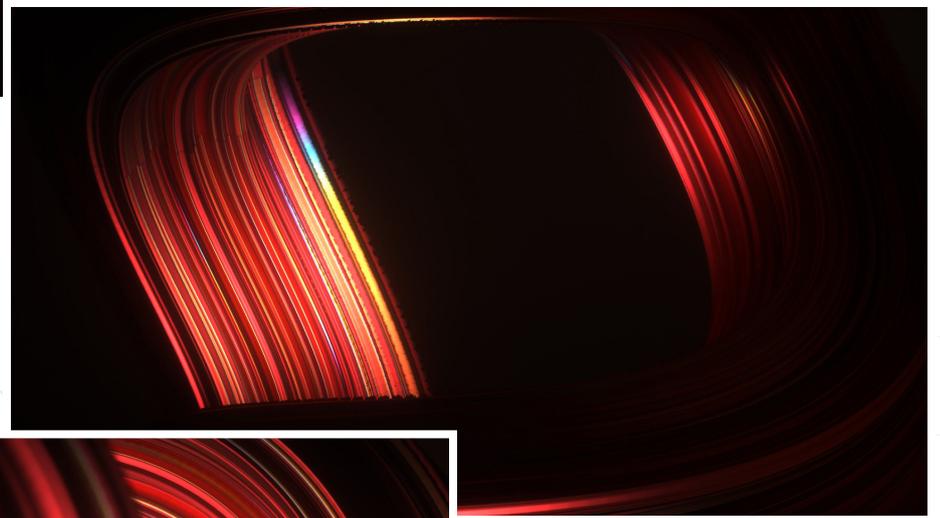
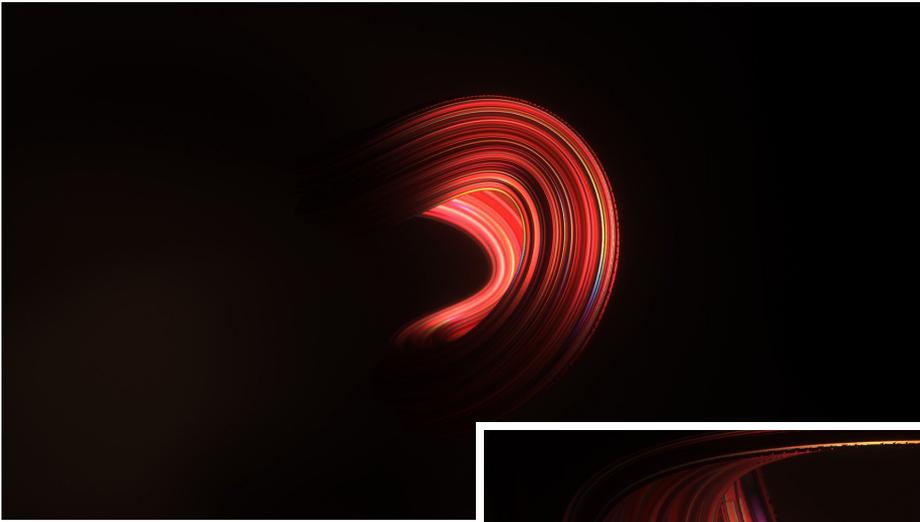
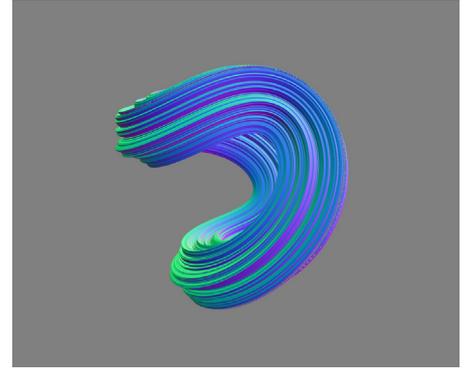
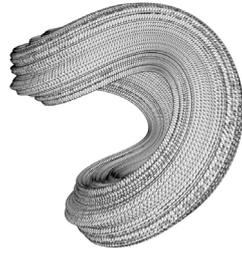
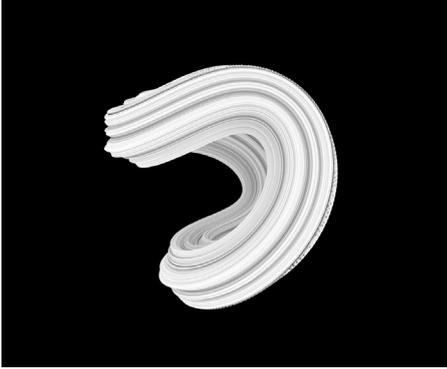


Wavy Turbulence

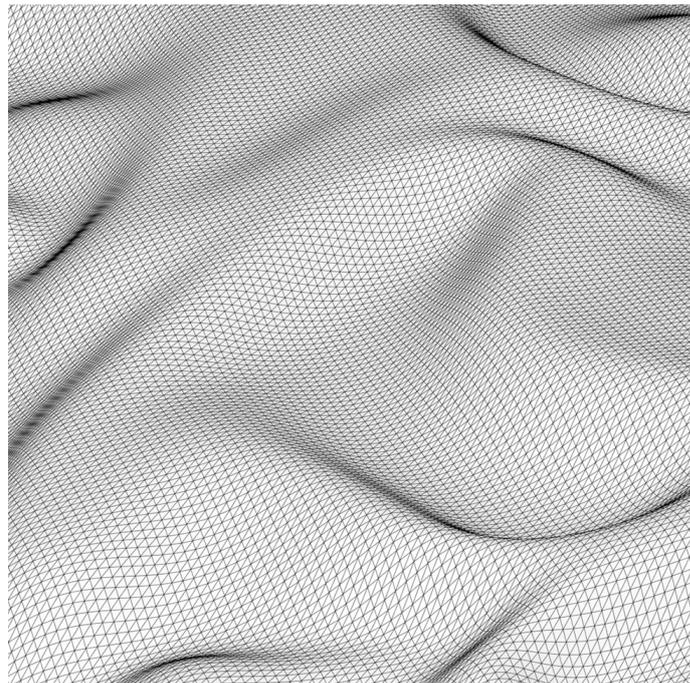
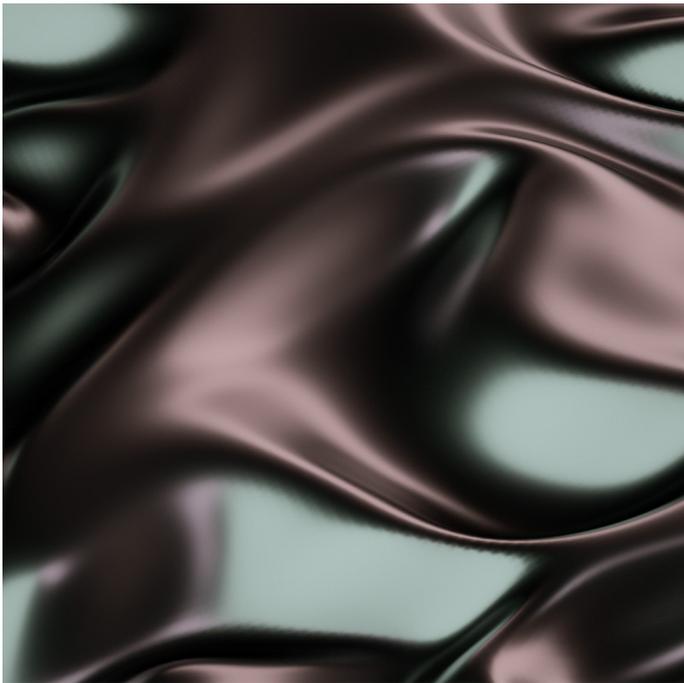
## Rendu d'eau



## Échelle Relative



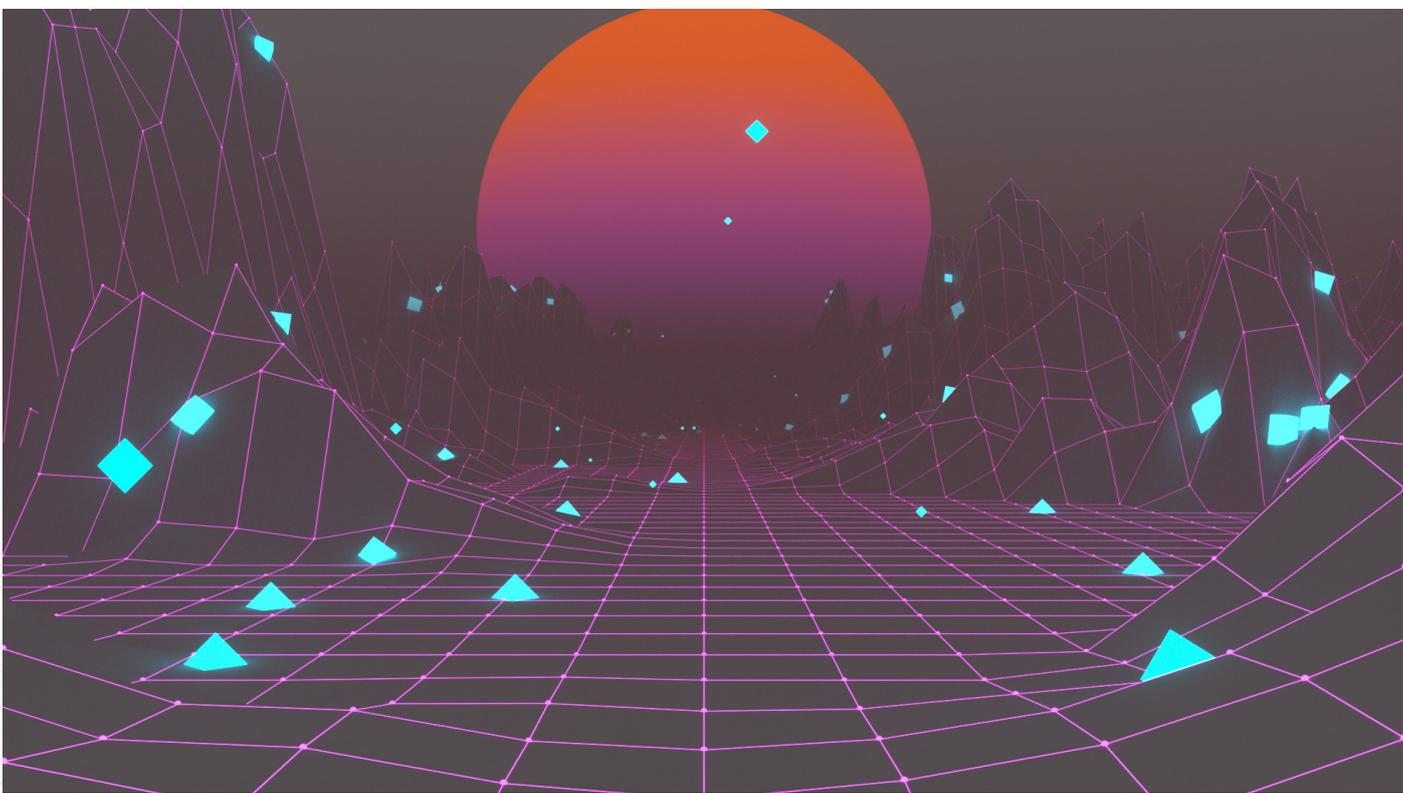
## Rendu d'eau



### Animation Iridescence :

<https://www.ludwigdejonckheere.com/noise/assets/iridescence/iridescent.mp4>

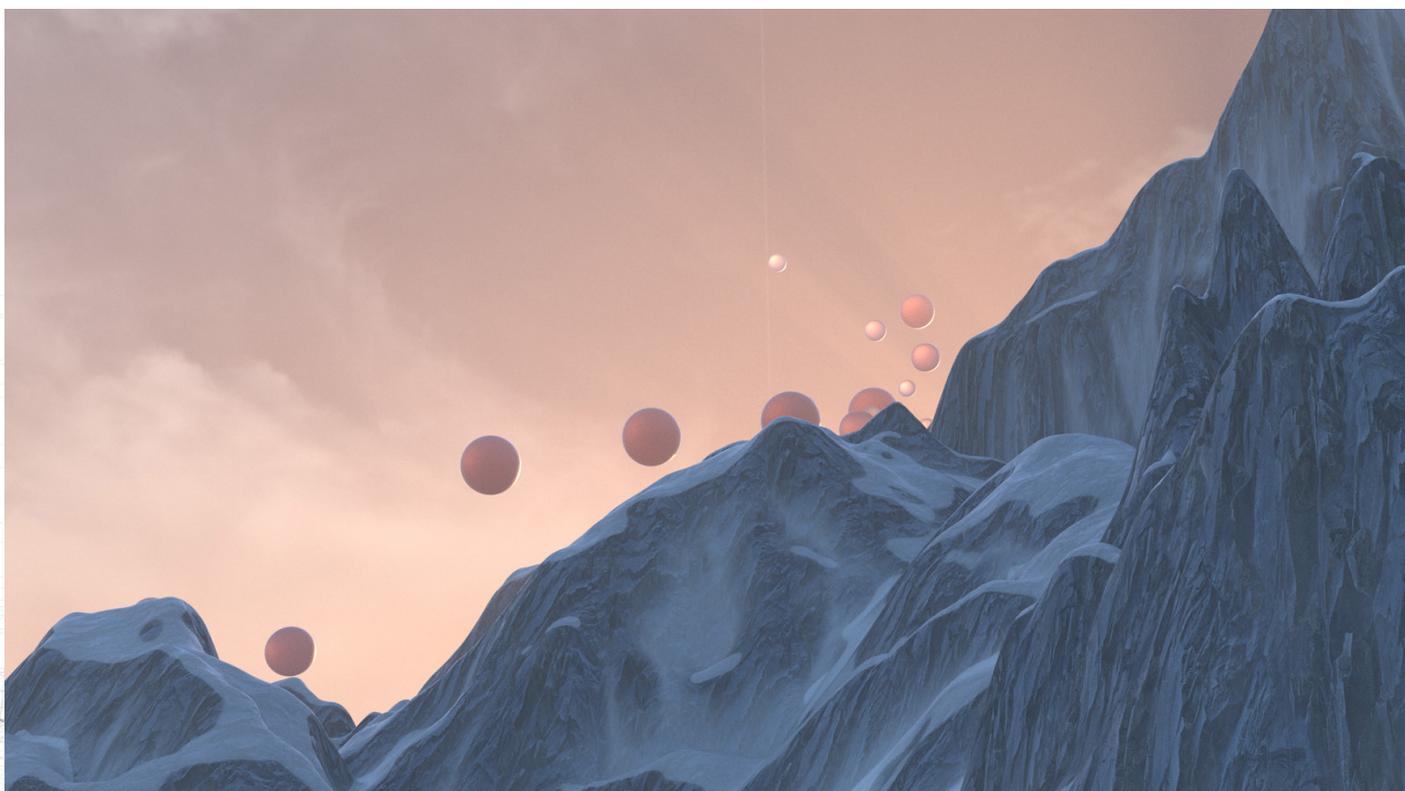
## Retro Terrain



### Animation Retro Terrain :

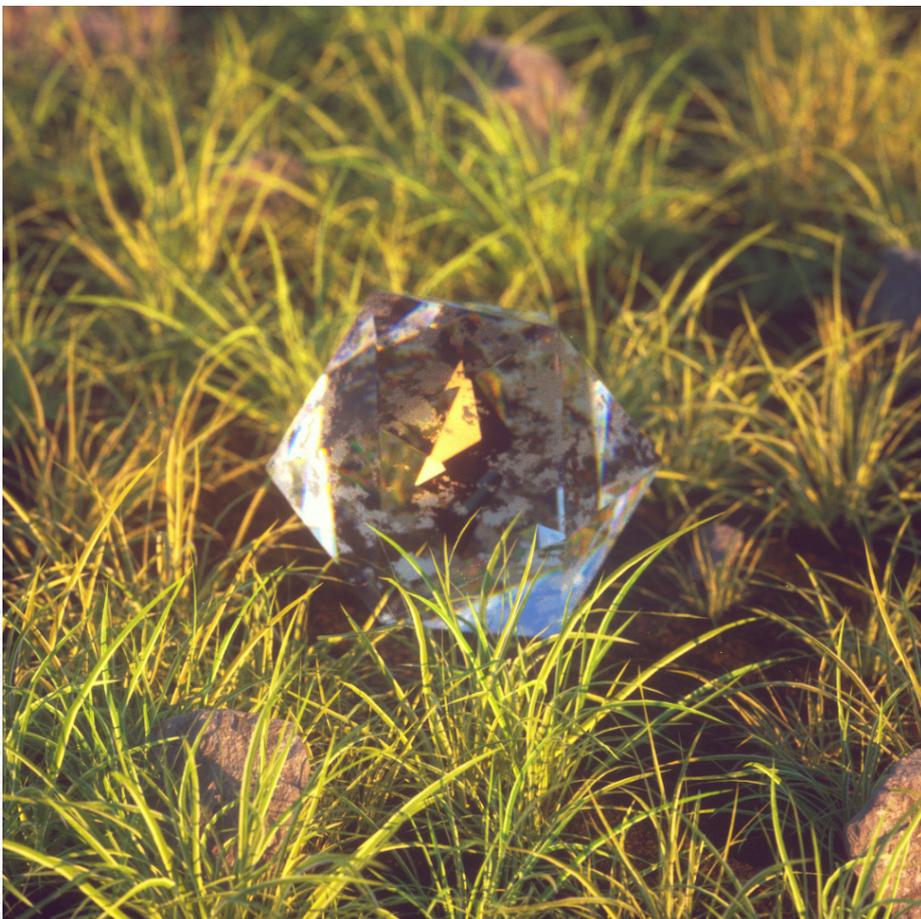
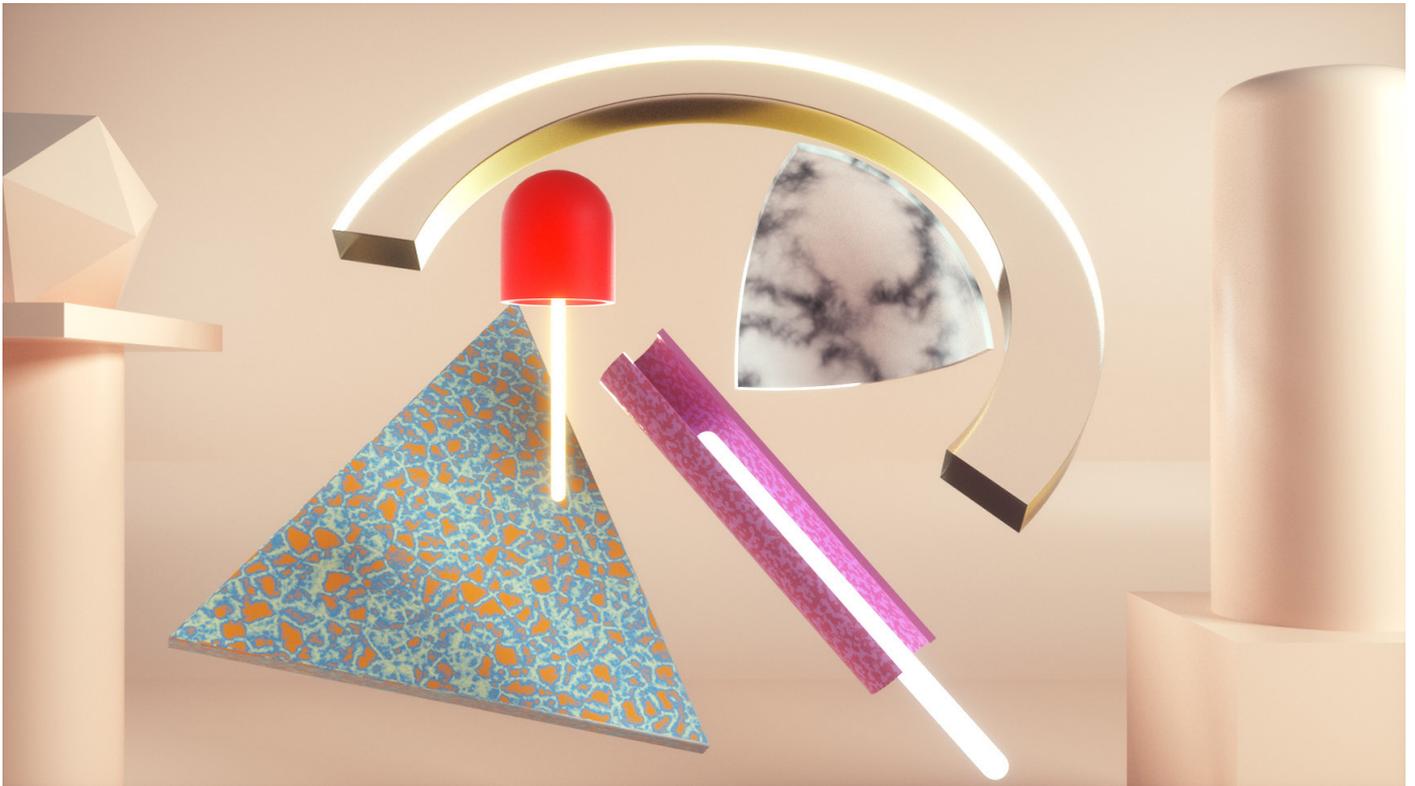
<https://www.ludwigdejonckheere.com/noise/assets/retro.mp4>

## Montagne

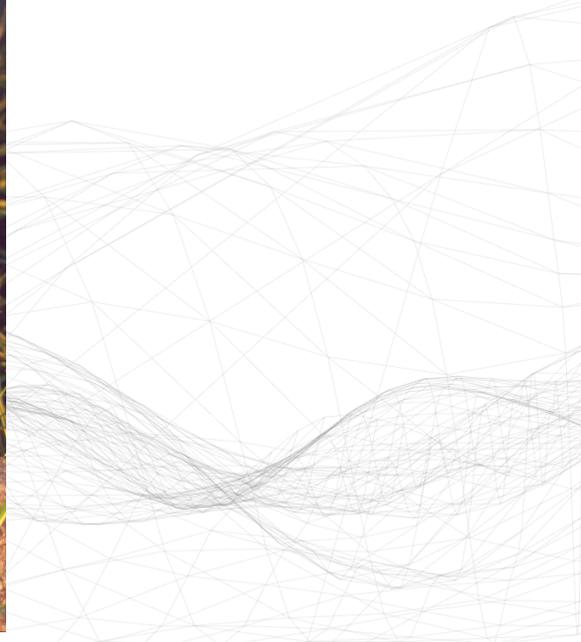


# TEXTURING

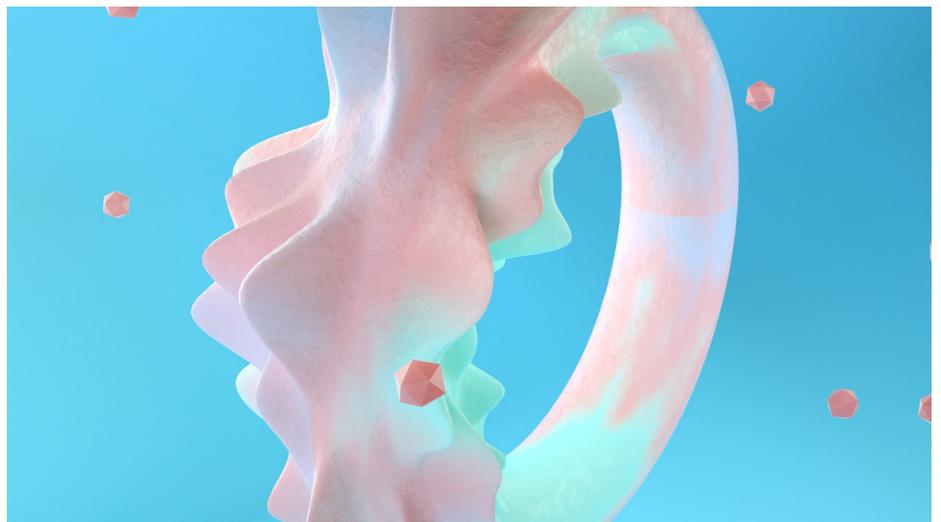
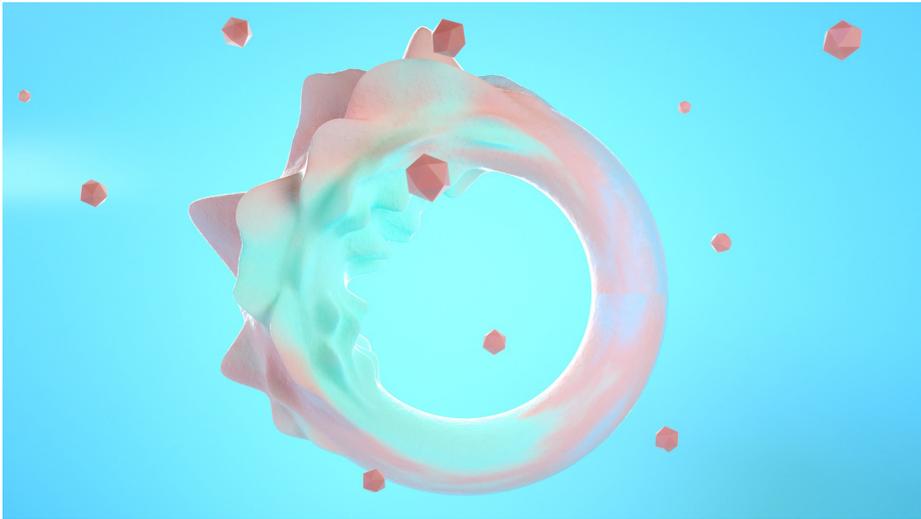
## Diffuse

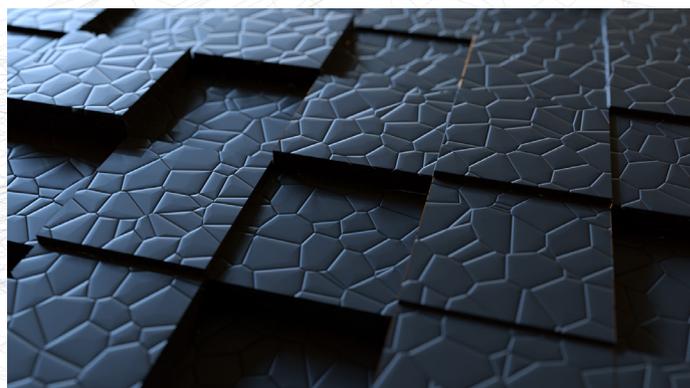
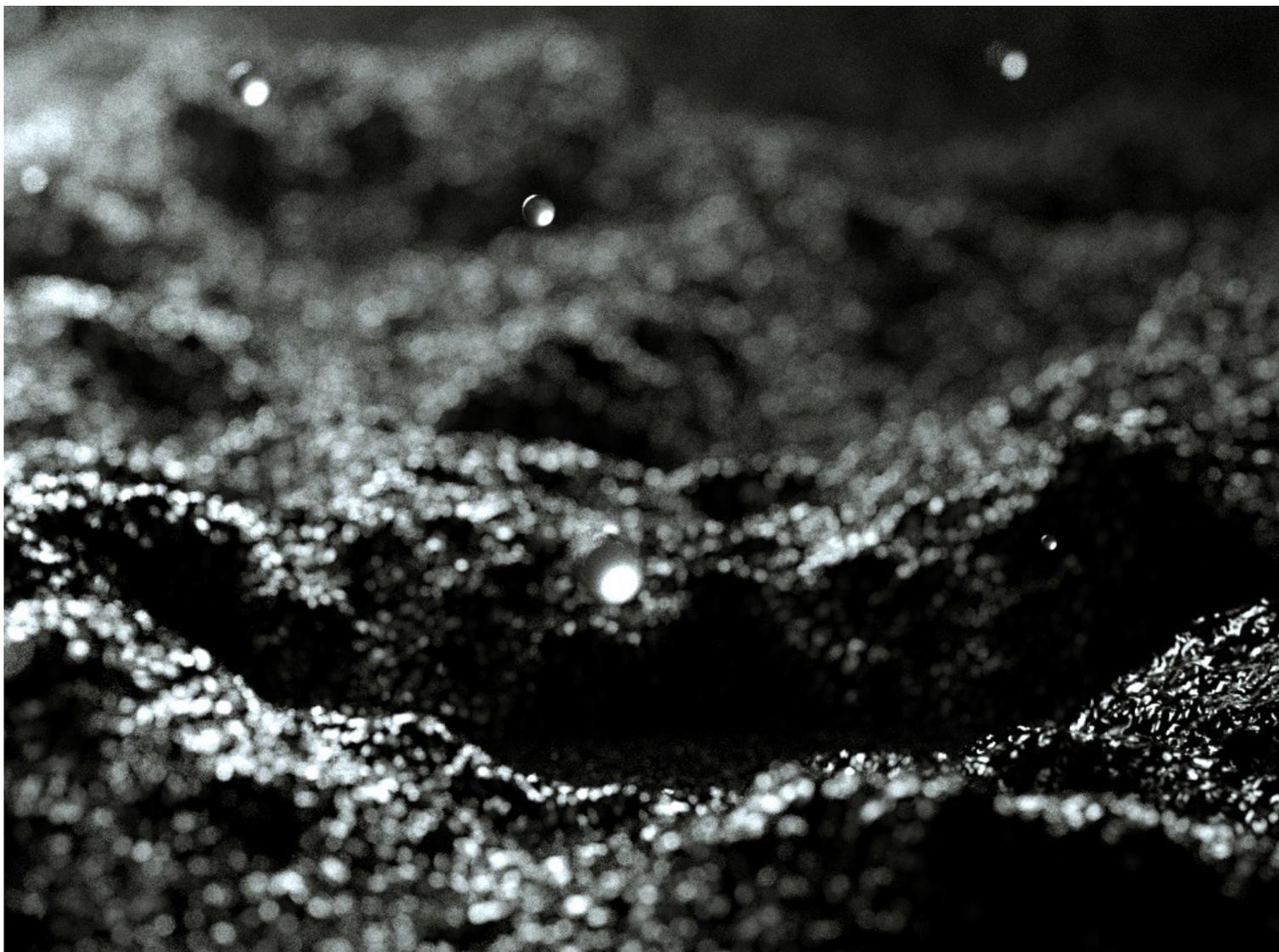


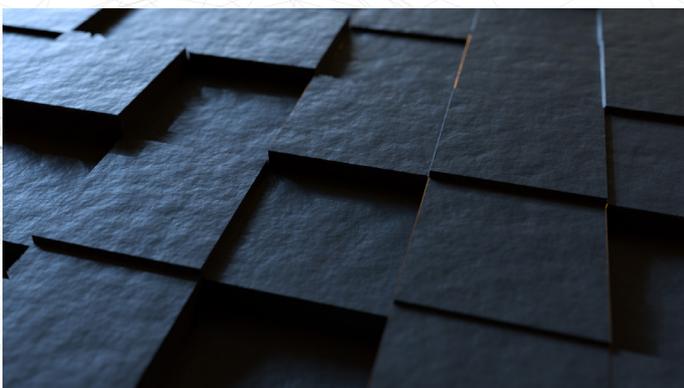
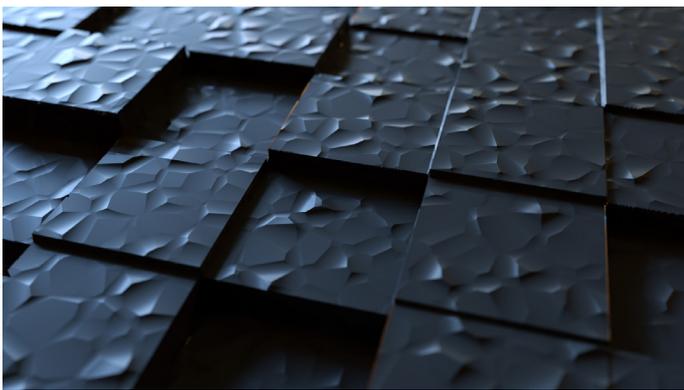
## Roughness



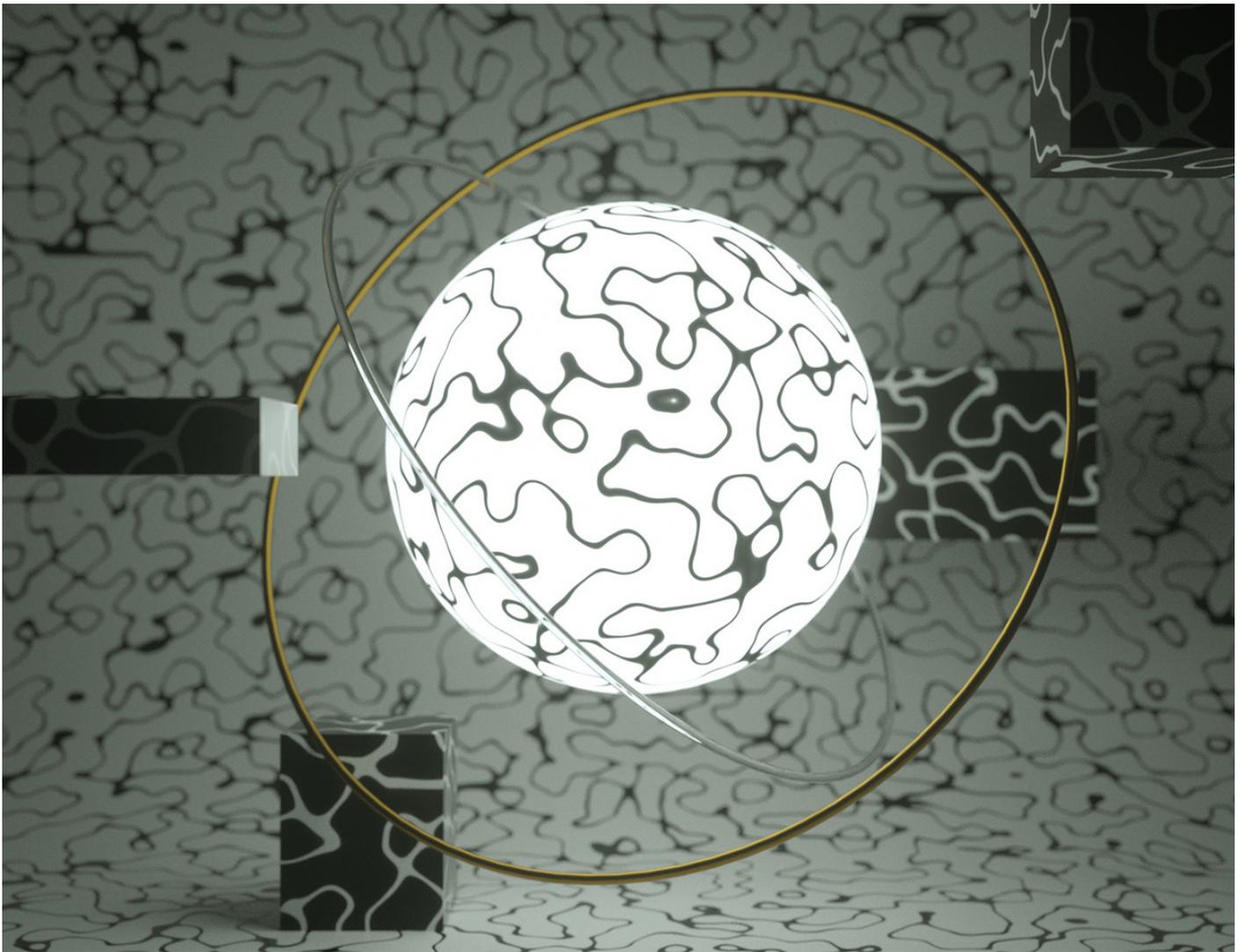
# Bump



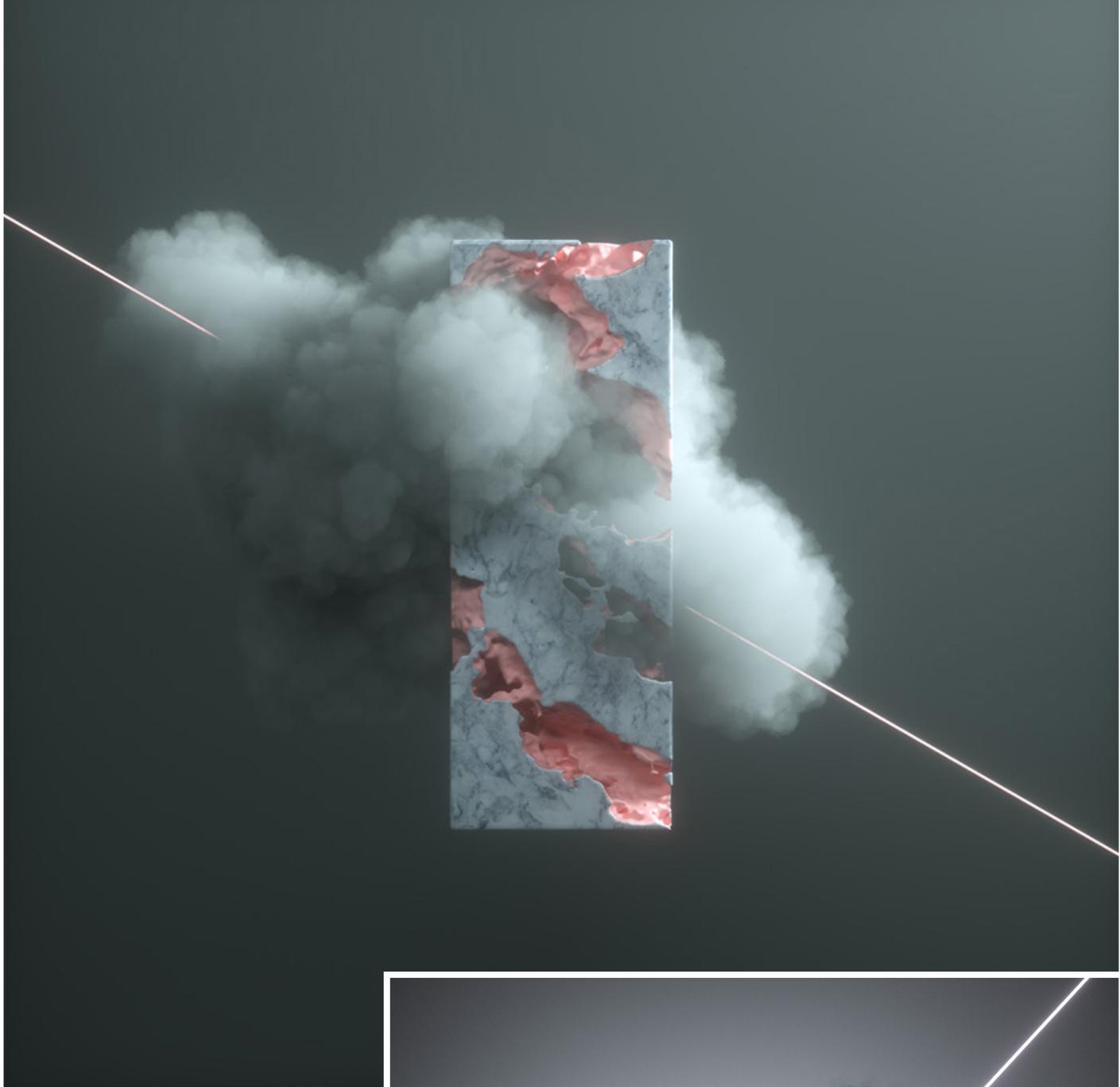




## Emission

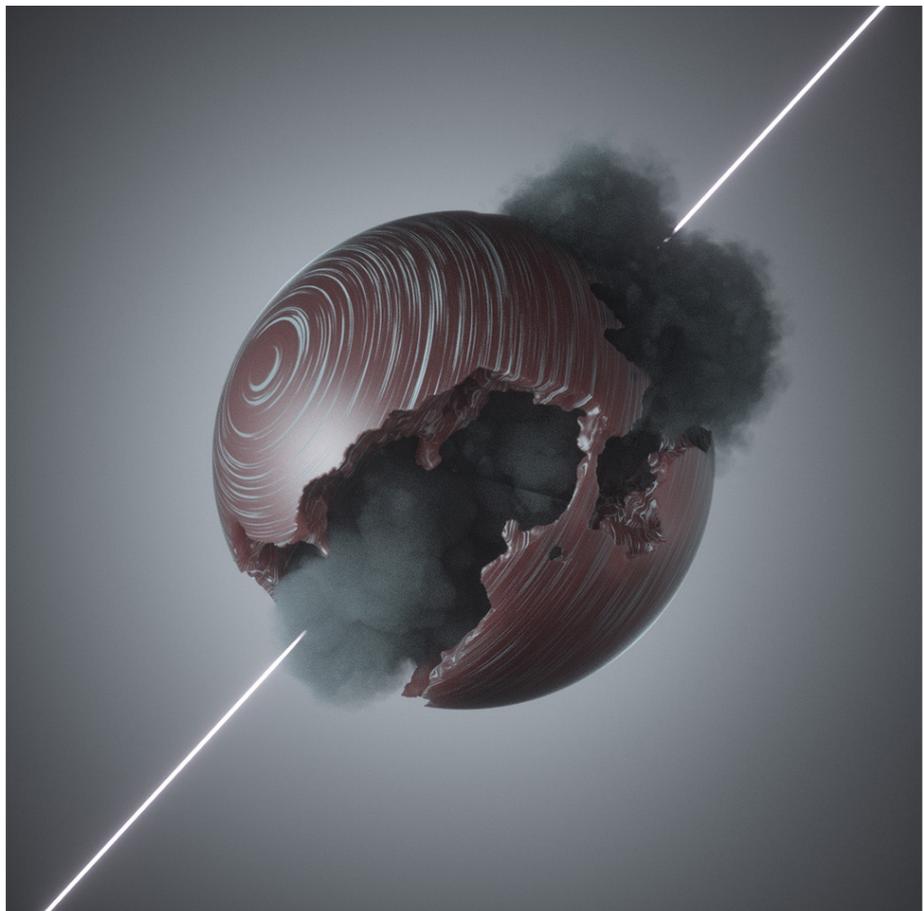


**Animation Emission :**  
<https://www.ludwigdejonckheere.com/noise/assets/emission.mp4>

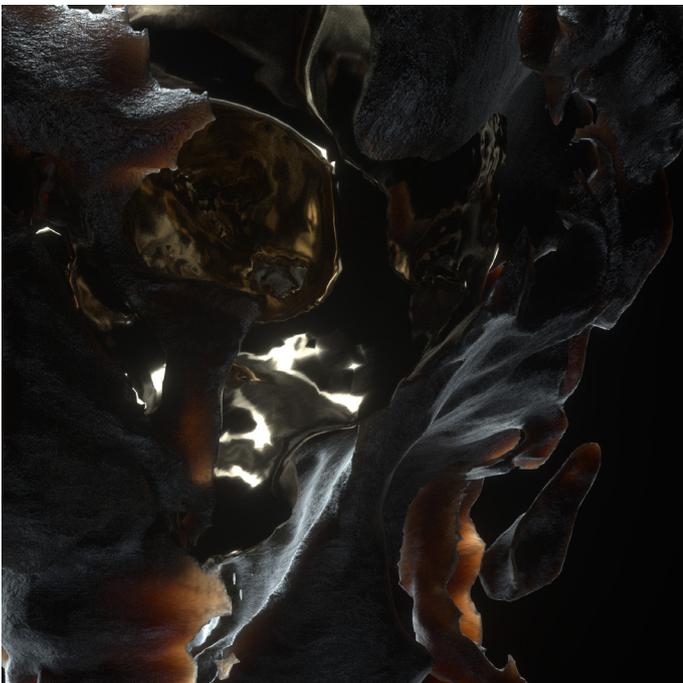


VOLUME

Marble & Ball



David



# ANIMATION

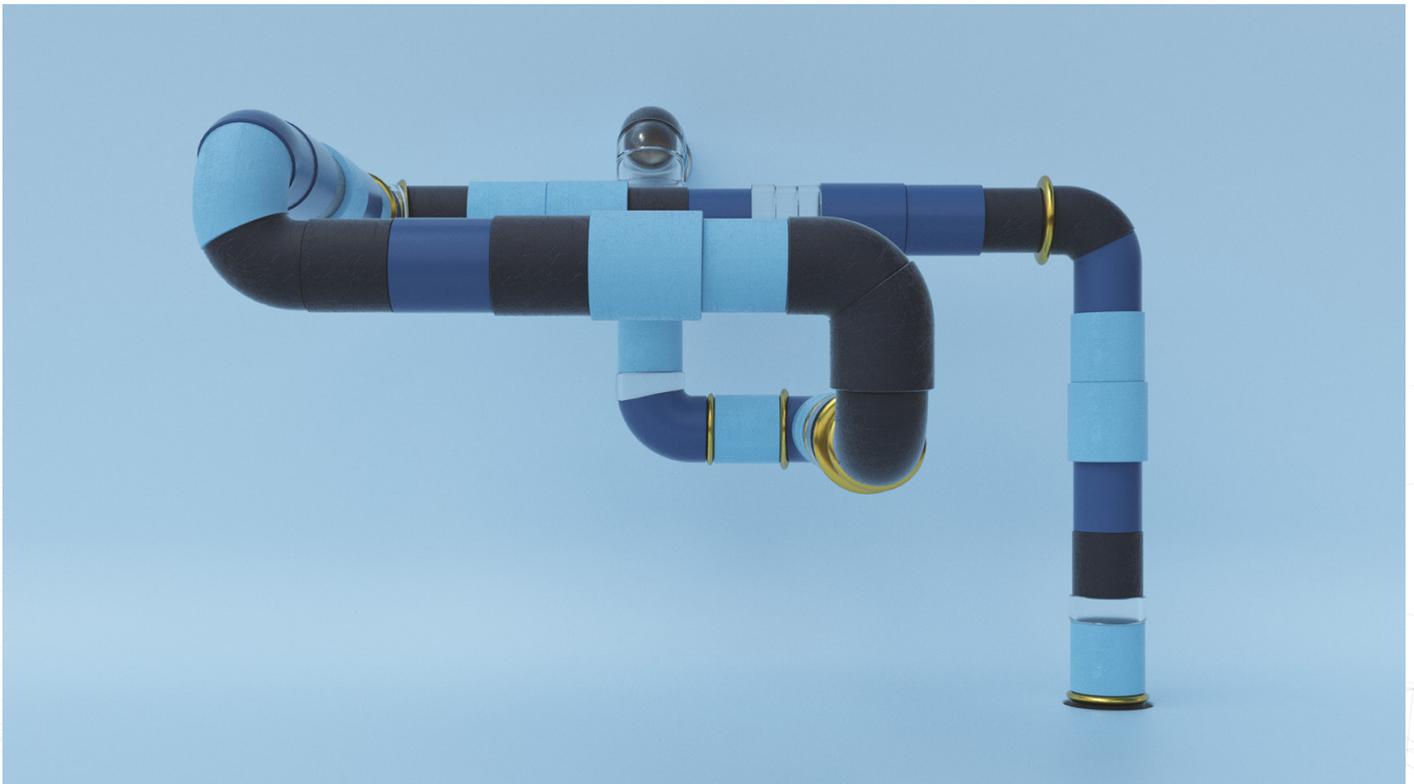
## HUD

### Animation HUD :

[https://www.ludwigdejonckheere.com/noise/assets/animation/tracking\\_v2.mp4](https://www.ludwigdejonckheere.com/noise/assets/animation/tracking_v2.mp4)



## Tubes



### Animation Tubes:

[https://www.ludwigdejonckheere.com/noise/assets/tubes\\_anim.mp4](https://www.ludwigdejonckheere.com/noise/assets/tubes_anim.mp4)

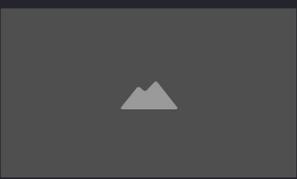
# SITE WEB

## Wireframes

# Noise

Learn about noise and how, connecting algorithms, and then moving into random noise and how it's used in the world of digital art.

EXPLORER



Reel

# Intro

Learn about noise and how, connecting algorithms, and then moving into random noise and how it's used in the world of digital art.

### Le Bruit

Learn about noise and how, connecting algorithms, and then moving into random noise and how it's used in the world of digital art.

### Objectifs

Learn about noise and how, connecting algorithms, and then moving into random noise and how it's used in the world of digital art.

# Theorie

À l'origine, le bruit est une fonction pseudo-aléatoire développée par John von Neumann et ses collègues en 1946. Elle est utilisée pour générer des nombres pseudo-aléatoires.

Il y a trois étapes dans la création d'un bruit :

- 01 La première étape est la génération d'une grille de données, les valeurs de la grille sont des nombres pseudo-aléatoires.
- 02 La deuxième étape de génération est de sélectionner les données de la grille.
- 03 Enfin, la dernière étape est l'interpolation entre les données sélectionnées de la grille.

### Chapitres

PRÉAMBULE	DISPLACEMENT	TEXTURING
VOLUME	ANIMATION	CONCLUSION

# Outils

Learn about noise and how, connecting algorithms, and then moving into random noise and how it's used in the world of digital art.

### Logiciels

Le premier logiciel utilisé est Cinema 4D, un logiciel de modélisation 3D développé par Maxon en 1986. C'est un logiciel de modélisation 3D, utilisé pour créer des objets 3D et les rendre interactifs. Les fonctionnalités de modélisation 3D, la création de scènes et l'animation sont les principales fonctionnalités de Cinema 4D. Les fonctionnalités de modélisation 3D, la création de scènes et l'animation sont les principales fonctionnalités de Cinema 4D.



### Paramètres du bruit dans Cinema 4D

Avec le paramètre de bruit dans Cinema 4D, il est possible de définir un programme de bruit qui génère un bruit. Ce paramètre est très utile pour générer des formes 3D et les rendre interactives.



Chapitre Précédent

Chapitre Suivant

# Displacement

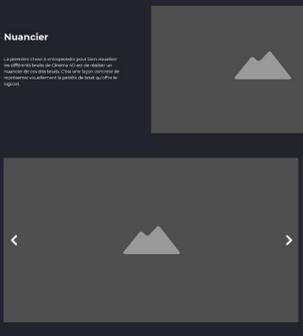
Learn about noise and how, connecting algorithms, and then moving into random noise and how it's used in the world of digital art.

### Qu'est-ce que le déplacement ?

Le déplacement, l'interpolation, l'ajustement et le réglage, est l'action de générer du volume à base d'un bruit 3D. Ce volume est généré en utilisant le bruit 3D et les données de déplacement. Le bruit 3D est utilisé pour générer des formes 3D et les rendre interactives.

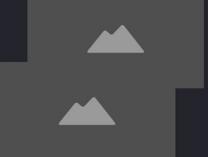
### Nuancier

Le premier nuancier est un diagramme qui permet de visualiser les données de déplacement. Ce diagramme est utilisé pour générer des formes 3D et les rendre interactives.



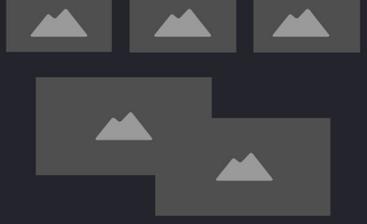
### Rendu d'eau

Le rendu d'eau est un processus qui permet de générer des formes 3D et les rendre interactives. Ce processus est utilisé pour générer des formes 3D et les rendre interactives.



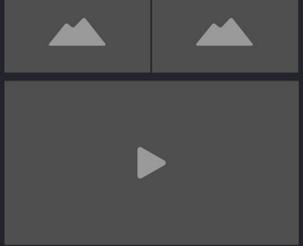
### Échelle relative

Le rendu de bruit est un processus qui permet de générer des formes 3D et les rendre interactives. Ce processus est utilisé pour générer des formes 3D et les rendre interactives.



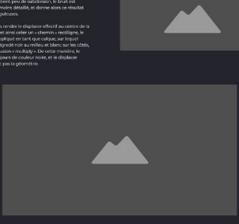
### Iridescence

Le rendu de bruit est un processus qui permet de générer des formes 3D et les rendre interactives. Ce processus est utilisé pour générer des formes 3D et les rendre interactives.



### Retro Terrain

Le rendu de bruit est un processus qui permet de générer des formes 3D et les rendre interactives. Ce processus est utilisé pour générer des formes 3D et les rendre interactives.



Montagne

# Texturing

Learn about how to create, combine and manage textures, and how to use them in your scene.

## Qu'est-ce que le texturing ?

Le texturing est l'application d'une texture à un objet 3D. C'est le processus de donner à un objet 3D l'apparence visuelle que vous souhaitez. Cela peut être une couleur, une texture, une réflexion, etc.



Choisir une texture est la première étape de la création d'un objet 3D. Il est important de choisir une texture qui correspond à l'objet que vous créez.

## Diffuse

Le diffuse est la couleur de base d'un objet 3D. C'est la couleur que vous voyez lorsque l'objet est éclairé.

## Roughness

Le roughness est la mesure de la rugosité d'un objet 3D. C'est la mesure de la façon dont la lumière est réfléchie par l'objet.

## Bump

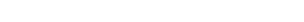
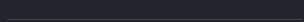
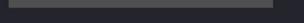
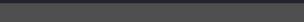
Le bump est une texture qui crée l'illusion de profondeur sur un objet 3D. C'est la mesure de la façon dont la lumière est réfléchie par l'objet.

## Emission

L'émission est la mesure de la luminosité d'un objet 3D. C'est la mesure de la façon dont la lumière est émise par l'objet.

## Chapitre Précédent

## Chapitre Suivant



# Volume

Learn about how to create, combine and manage volumes, and how to use them in your scene.

## Que sont le Volume Builder et le Volume Mesher ?

Le volume builder est un outil qui permet de créer des volumes 3D à partir de primitives. Le volume mesher est un outil qui permet de créer des volumes 3D à partir de maillages.



Le volume builder est un outil qui permet de créer des volumes 3D à partir de primitives. Le volume mesher est un outil qui permet de créer des volumes 3D à partir de maillages.

## 01

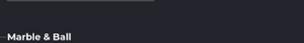
Créer un volume à partir d'une primitive.



Modifier les paramètres du volume.



Appliquer une texture au volume.



Finaliser le volume.



Le volume builder est un outil qui permet de créer des volumes 3D à partir de primitives. Le volume mesher est un outil qui permet de créer des volumes 3D à partir de maillages.

## Marble & Ball

Le marble & ball est un volume qui crée l'illusion de profondeur sur un objet 3D. C'est la mesure de la façon dont la lumière est réfléchie par l'objet.



Le marble & ball est un volume qui crée l'illusion de profondeur sur un objet 3D. C'est la mesure de la façon dont la lumière est réfléchie par l'objet.



Le marble & ball est un volume qui crée l'illusion de profondeur sur un objet 3D. C'est la mesure de la façon dont la lumière est réfléchie par l'objet.



Le marble & ball est un volume qui crée l'illusion de profondeur sur un objet 3D. C'est la mesure de la façon dont la lumière est réfléchie par l'objet.



Le marble & ball est un volume qui crée l'illusion de profondeur sur un objet 3D. C'est la mesure de la façon dont la lumière est réfléchie par l'objet.



Le marble & ball est un volume qui crée l'illusion de profondeur sur un objet 3D. C'est la mesure de la façon dont la lumière est réfléchie par l'objet.



Le marble & ball est un volume qui crée l'illusion de profondeur sur un objet 3D. C'est la mesure de la façon dont la lumière est réfléchie par l'objet.



Le marble & ball est un volume qui crée l'illusion de profondeur sur un objet 3D. C'est la mesure de la façon dont la lumière est réfléchie par l'objet.



Le marble & ball est un volume qui crée l'illusion de profondeur sur un objet 3D. C'est la mesure de la façon dont la lumière est réfléchie par l'objet.



Le marble & ball est un volume qui crée l'illusion de profondeur sur un objet 3D. C'est la mesure de la façon dont la lumière est réfléchie par l'objet.



Le marble & ball est un volume qui crée l'illusion de profondeur sur un objet 3D. C'est la mesure de la façon dont la lumière est réfléchie par l'objet.



Le marble & ball est un volume qui crée l'illusion de profondeur sur un objet 3D. C'est la mesure de la façon dont la lumière est réfléchie par l'objet.



# Animation

Learn about how to create, combine and manage animations, and how to use them in your scene.

## Animations Aléatoires

Les animations aléatoires sont des animations qui se jouent de manière aléatoire. Elles sont utiles pour créer des scènes plus dynamiques et intéressantes.



Les animations aléatoires sont des animations qui se jouent de manière aléatoire. Elles sont utiles pour créer des scènes plus dynamiques et intéressantes.

## HUD

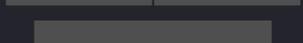
Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



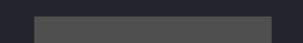
Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



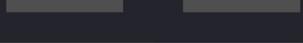
Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Le HUD est un élément de l'interface utilisateur qui permet de visualiser des informations importantes pendant le jeu.



Que signifie « le bruit en infographie » ?

Ce travail met en lumière les différents usages du bruit de Perlin lorsque l'on traite de créations 3D.

Qu'est-ce que le bruit de Perlin ? Quels côtés pratiques présente-t-il concernant le monde de la 3D ?

What does « noise in infographics » means ?

This work sheds lights on different purposes of the Perlin noise when dealing with 3D creations.

What practical sides does it present concerning the world of 3D?