

UNIVERSITÉ DE MONTPELLIER  
FACULTÉ DES SCIENCES  
MASTER 2 INFORMATIQUE  
PARCOURS IMAGINE

---

**Bruits à variation spatiale en temps réel**

---

MÉMOIRE DE STAGE — HAI002I

EFFECTUÉ AU LIRMM  
DU 27/01/2025 AU 25/07/2025  
PAR ARTHUR CHATEAUNEUF

ENCADRANT INDUSTRIEL : M. NICOLAS LUTZ  
ENCADRANT UNIVERSITAIRE : M. ERIC BOURREAUI



## **Table des matières**

# 1 Introduction

## 1.1 Contexte du stage

Dans le cadre de mon Master Informatique, j'ai effectué un stage industriel encadré par Nicolas Lutz au Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM). J'ai travaillé au sein du département informatique dans l'équipe ICAR sur le sujet des bruits à variation spatiale en temps réel.

## 1.2 Contexte technique et méthode de travail

Lors de ce stage, j'ai travaillé sur ma propre base de code en utilisant Vulpine, mon propre moteur de jeu et d'applications interactives que j'ai eu l'occasion de développer dans le cadre de mon parcours académique. Je possédais ainsi une forte base technique et une connaissance approfondie de mes outils dès le début de mon travail de recherche. J'ai travaillé en collaboration étroite avec mon maître de stage Nicolas Lutz, qui m'a initié au monde de la recherche et au sujet de mon stage. Nous faisions le point plusieurs fois par mois pour que je lui présente mes avancées et qu'il me propose des pistes d'améliorations.

Durant l'ensemble de mon stage, j'ai eu l'occasion de rencontré de nombreux chercheurs, doctorants et stagiaires. J'ai participé régulièrement à des réunions d'équipes où j'ai eu l'occasion d'assister à la présentation de leur travail. J'ai également présenté mon sujet de stage devant ces mêmes équipes.

## 1.3 Problématique

En informatique graphique, un bruit est un processus stochastique (ou fonction aléatoire) permettant de créer des variations de motifs infinis, le plus souvent en deux ou trois dimensions, utilisé pour la création d'effets visuels complexes comme la simulation de flammes ou d'océans. Cependant, la majorité des bruits utilisés sont dits stationnaires, cela signifie que leurs attributs statistiques (variance, moyenne, histogramme...) n'évoluent pas en fonction de l'espace ou des réalisations.

Il existe des solutions non stationnaires, dont un exemple est visible dans la figure ??, qui ne sont cependant pas utilisables dans un contexte temps-réel. De telles méthodes nécessitent des pré-calculs lourds et la sauvegarde de résultats à l'intérieur d'une texture matricielle. L'un des travaux les plus notables et récents dans cette optique est One Noise to Rule Them All [**One\_Noise**], publié en 2024, qui présente une solution de création de bruit à variation spatiale généré par un réseau de neurone.

Si une méthode permettant d'obtenir des résultats similaires en temps réels était trouvée, cela ouvrirait les portes à l'utilisation de bruits à variation spatiales dans des contextes procéduraux, non bornés, animés ou même interactifs. Une telle avancée améliorerait la qualité et la modularité des outils utilisés dans de nombreux domaines comme le jeu vidéo ou la modélisation 3D.

Une fois familiarisé avec les concepts de technique et de recherche nécessaires, ma mission lors de ce stage a été d'explorer des solutions permettant de combiner des bruits stationnaires afin d'obtenir une sortie non stationnaire. Premièrement en utilisant le pavage et mélange [**HPnoise**] et ensuite en utilisant l'opérateur de mélange Mix-Max [**mixmax**].

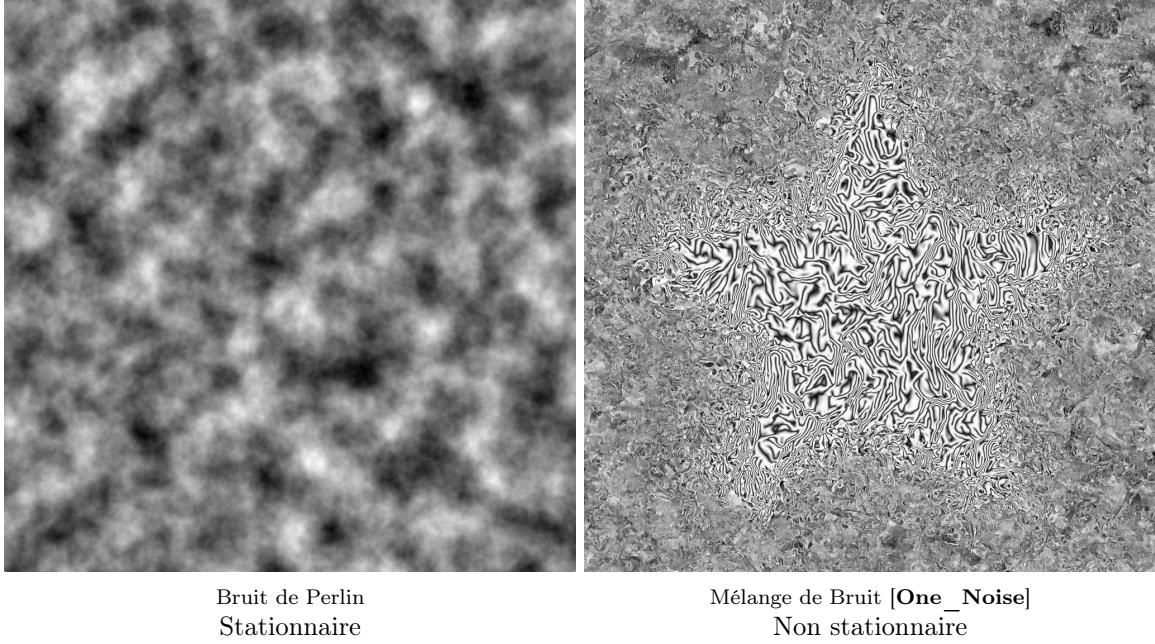


FIGURE 1 – Comparaison visuelle entre un bruit stationnaire et non stationnaire. De claires variations statistiques sont visibles dans les motifs produit par la méthode non stationnaire.

## 2 État de l'art

### 2.1 State of the Art in Procedural Noise Functions

Mon travail a commencé avec la lecture de State of the Art in Procedural Noise Functions [**SOA\_Noise**], un état de l'art avancé sur la conception de bruits procéduraux temps réel. Ce papier fait d'abord état d'un fait dans l'évolution des capacités des processeurs graphique : l'efficacité des calculs évolue à un rythme plus grand que celle de la mémoire. Cela signifie que l'empreinte mémoire des algorithmes est de plus en plus déterminante dans leur rapidité. Cela permet à des méthodes auparavant trop couteuses en calculs de devenir de bonnes alternatives, parfois même préférables.

L'utilisation de bruits procéduraux fait partie intégrante de ces évolutions dans l'informatique graphique. Les bruits sont définis plus formellement comme des processus combinant plusieurs signaux de façon pseudo-aléatoire [**SOA\_Noise**]. Le terme de "bruit" venant, quant à lui, de l'utilisation originelle de ces techniques dans le cinéma et les images de synthèse, qui servaient à recréer de façon rapide des irrégularités dans des signaux visuels tels que des textures.

Ces techniques sont le plus souvent utilisées en pré-calculs. Leurs résultats sont ainsi stockés dans la mémoire sous forme de textures matricielles en deux ou trois dimensions. Cependant, cette méthode garde les limitations techniques de l'utilisation de données pré-calculés. Les textures sont ainsi bornées, possèdent une résolution limitée et augmentent l'empreinte mémoire à chaque échantillonnage. L'alternative est de calculer ces processus pendant leur rendu, pour chaque fragment de chaque image devant être calculé. Ces méthodes sont qualifiées de procédurales, car elles produisent des infinités de motifs non périodiques, pseudo-aléatoires et continus.

L'absence de dépendance sur des données discrétisées diminue grandement l'empreinte mémoire tout en offrant la possibilité de calculer en temps réel des itérations différentes de chaque bruit. Le rendu résultant peut ainsi réagir à n'importe quelle influence extérieure, le rendant animable ou interactif. Les plus grands avantages des processus procéduraux sont leur résolution et leur taille infinies, s'adaptant en temps réel aux besoins du rendu.

Les difficultés du calcul en temps réel sont également abordées. Tous les éléments procéduraux doivent être filtrés analytiquement lors du rendu d'une texture sur une surface. Cela signifie que la moyenne de leur intensité/couleur doit être calculé sur la zone entière que couvre chaque fragment que l'on souhaite rendre. L'un des plus grands défis de la création de bruits procéduraux est de trouver des méthodes permettant de calculer ou d'approximer de façon suffisamment précise ces moyennes. Sur des textures matricielles (composés de pixels stockés en mémoire), cela se fait par l'utilisation de mip-maps, qui sont des versions sous-échantillonnes de la texture originale. Une telle méthode est cependant impossible avec des textures procédurales, car aucune valeur n'est pré-calculée.

Ce papier m'a permis d'obtenir des enseignements importants pour la compréhension de la nature des bruits procéduraux ainsi que de leur pertinence. Il démontre également la complexité des problèmes à résoudre afin de correctement afficher une texture matricielle sur une surface.

## 2.2 One Noise to Rule Them All : Learning a Unified Model of Spatially-Varying Noise Pattern

Par la suite, je me suis intéressé aux avancées en création de processus non stationnaires. La publication la plus pertinente sur cette piste est One Noise to Rule Them All [**One\_Noise**], un modèle de génération de bruits procéduraux non stationnaires.

La méthode proposée dans ce papier utilise un réseau de neurones entraîné sur une série d'échantillons de bruits procéduraux stationnaires. Ce dernier est capable de générer des bruits non stationnaires en mélangeant des patterns de plusieurs bruits. Cette méthode permet ainsi de combiner n'importe quel bruit pré-existant en une infinité de sorties non stationnaires. Les variations spatiales sont apportées grâce à une carte de mélange donnée par l'utilisateur. L'objectif est alors de transformer une entrée non stationnaire simple en une infinité de sorties complexes. Cela permet la création de patterns possédant des attributs uniques variant spatiallement de façon logique et équilibré.

Cet algorithme ne permet cependant pas de générer des entrées en temps réel, car il nécessite l'utilisation d'un réseaux de neurone qui stocke des résultats discrétisés. Le bruit généré est ainsi borné, non continue et ne peut pas réagir aux influences extérieures.

Ce papier montre que le mélange de plusieurs bruits stationnaires est un moyen efficace d'obtenir les résultats que je recherche. J'ai ainsi décidé de m'inspirer de leur approche en explorant les méthodes de mélange de processus procéduraux pouvant être appliquée en temps réels. Cela nécessite une méthode rapide, parallélisable, peu coûteuse en mémoire et analytiquement filtrable.

## 2.3 High-Performance By-Example Noise using a Histogram-Preserving Blending Operator

La première piste d'algorithme de mélange que j'ai étudié était le pavage et mélange, présenté dans le papier High-Performance By-Example Noise using a Histogram-Preserving Blending Operator [**HPnoise**] publié en 2018. Il y est décrit un système de pavage, pouvant être rectangulaire

ou hexagonale, permettant de représenter de façon infinie et non répétitif une texture discrétisée sur une surface 3D.

Chaque fragment que l'on souhaite calculé est interpolé entre trois instances de la texture originelle, chacune ayant une rotation et une translation différente. La sortie obtenue est ainsi majoritairement composé de couleurs interpolées. Afin de réduire le flou et l'introduction de teintes non présentes dans la texture originale, deux équations de mélange sont proposés. La première, applicable en temps réel, préserve la moyenne ainsi que la variance de l'entrée. La seconde, quant à elle, nécessite une grande quantité de pré-calculs et une multitude de défis techniques mais permet de préserver l'histogramme en plus de la variance.

Cette méthode pourrait servir à mélanger des bruits procéduraux et à créer des transitions respectant certaines propriétés statistiques, améliorant la qualité visuelle de mélanges entre deux bruits. La première équation de préservation de la variance permettrait d'appliquer cette amélioration en temps réel.

## 2.4 Mix-Max : A Content-Aware Operator for Real-Time Texture Transitions

L'opérateur Mix-Max [**mixmax**] offre également une piste très prometteuse pour le mélange de bruits. Proposée en 2024, cette méthode a été conçus pour mélanger des textures matricielles représentant des matériaux opaques. Cette méthode permet d'obtenir des transitions nettes et naturelles entre une infinité de textures, tout en étant filtrable analytiquement et utilisable en temps réel. Cependant, cette méthode entraîne l'échantillonage de plusieurs textures pour chaque entrée, augmentant l'empreinte mémoire par rapport à un mélange linéaire simple.

Ces résultats sont obtenus grâce à l'utilisation d'une carte de priorité, créée par l'utilisateur selon ses besoins, qui biaise le mélange en faveur ou contre sa texture attribué. Si la texture de priorité est obtenue à partir du contenu de la texture de couleur, on obtient alors un opérateur de mélange sensible au contenu. Les résultats de cette méthode sont inédits. Si l'on souhaite mélanger, par exemple, une texture de brique et de mousse, il est possible de faire se répandre la mousse en priorité entre les creux des briques. Cet opérateur offre des paramètres tels que le choix des textures d'entrées, de leur priorité, de leur poids de mélange ainsi que la netteté de la transition, le tout en étant analytiquement filtrable. Cependant, il n'est utilisable qu'avec des textures matricielles et nécessite l'utilisation d'au moins deux autres textures de support pour chaque entrée que l'on souhaite mélanger.

L'opérateur Mix-Max, s'il est étendue aux bruits procéduraux, permettrait d'obtenir un mélange de patterns respectant leur contenus et possédaient toutes les propriétés nécessaires à l'utilisation en temps réel. Cela permettrait de simuler des résultats théoriquement proches de One Noise to Rule Them All.

### 3 Mélange de processus stationnaires

#### 3.1 Texture, Bruit et empreinte

Afin de mettre en place des techniques de mélanges de bruits procéduraux, il est nécessaire de définir les aspects théoriques concernés. Premièrement, un processus  $\mathcal{P}$  est une fonction associant un interval de valeurs  $V$  de dimension  $n$  à un espace d'échantillonage  $\mathbf{u}$  de dimension  $m$  :

$$\mathcal{P} : \mathbf{u} \rightarrow V \mid \mathbf{u} \subset \mathbb{R}^m \wedge V \subset \mathbb{R}^n \quad (1)$$

Une texture  $\mathcal{T}$  est un processus dont l'intervalle de valeur est compris entre 0 et 1 pour chaque dimensions, et dont l'espace d'échantillonage est deux-dimensionnel :

$$\mathcal{T} : \mathbf{u} \rightarrow [0, 1]^n \mid \mathbf{u} \subset \mathbb{R}^2 \quad (2)$$

Une texture matricielle  $\mathcal{T}_m$  est une texture possèdant des valeurs composés par l'interpolation d'un interval discret  $\mathbb{D}$ , représentant l'ensemble des combinaisons de pixels stockés dans la texture. Un tel processus possède un espace d'échantillonage borné dans l'intervalle  $[0, 1]$  :

$$\mathcal{T}_m : [0, 1]^2 \rightarrow \text{Interp}(\mathbb{D}) \mid \mathbb{D} \subset [0, 1]^n \quad (3)$$

Une texture procédurale  $\mathcal{T}_p$  est une texture possèdant des valeurs continues et pouvant être échantillonée sur  $\mathbb{R}^2$  :

$$\mathcal{T}_p : \mathbb{R}^2 \rightarrow [0, 1]^n \quad (4)$$

Un bruit  $\mathcal{B}$  est une texture aléatoire associant des intensités scalaires non périodiques [**SOA\_Noise**] à son espace d'échantillonage. Un bruit peut ainsi également être matriciel ou procédural. De plus, il possède un espace de réalisation. Une réalisation  $\omega$  est un tirage du bruit aléatoire.

$$\mathcal{B} : \mathbf{u}, \omega \rightarrow [0, 1] \quad (5)$$

$$\forall t, \exists x \in \mathbf{u} \mid t + x \in \mathbf{u}, \mathcal{B}(x) \neq \mathcal{B}(x + t) \sim \text{non périodicité} \quad (6)$$

L'empreinte moyenne  $\mathcal{T}(\mathbb{P})$  d'un processus  $\mathcal{P}$  sur une zone  $\mathbb{P}$  permet de connaître la moyenne de toutes les valeurs que le processus prend sur cette zone :

$$\mathcal{P}(\mathbb{P}) = \frac{1}{\text{area}(\mathbb{P})} \int_{\mathbb{P}} \mathcal{T}(\mathbf{u}) d\mathbf{u} \quad (7)$$

Toute méthode de filtrage analytique d'une texture doit approximer l'empreinte moyenne pour obtenir un rendu satisfaisant [**SOA\_Noise**].

#### 3.2 Opérateur de mélange

Un opérateur de mélange  $\mathcal{M}$  est un processus représentant le mélange entre deux textures  $\mathcal{T}_1, \mathcal{T}_2$  pondéré par les poids de mélange  $v_1, v_2$ . Le mélange entre deux texture doit tendre vers les valeurs de chaque entrées lorsque son poids devient maximal. Un tel système obéit aux équations ?? et ?? :

$$v_i \in [0, 1], v_1 + v_2 = 1 \quad (8)$$

$$\lim_{v_i \rightarrow 1} \mathcal{M}(\mathbf{u}) = \mathcal{T}_i(\mathbf{u}) \quad (9)$$

L'opacité  $w_i$  d'une texture  $\mathcal{T}_i$  correspond à son importance finale dans le mélange, elle répond à l'équation ?? :

$$\mathcal{M}(\mathbf{u}) = w_1 \mathcal{T}_1(\mathbf{u}) + w_2 \mathcal{T}_2(\mathbf{u}) \quad (10)$$

Un mélange est dit équilibré si l'espérance de l'opacité de chaque texture est égal à son poids initial, respectant la propriété de l'équation ?? :

$$\mathbb{E}[w_i] = v_i \quad (11)$$

Le mélange linéaire  $\mathcal{M}^{lin}$  est la méthode la plus simple et répandue de mélanger deux texture. L'opacité est directement égale au poids de mélange initaux :

$$\mathcal{M}^{lin} = v_i \mathcal{T}_1(\mathbf{u}) + v_2 \mathcal{T}_2(\mathbf{u}) \quad (12)$$

### 3.3 Stationnarité

Un bruit, qui est assimilable à une variable aléatoire  $X$ , est dit stationnaire à l'ordre  $n$  lorsque son moment d'ordre  $n$  est constant sur chaque réalisation à n'importe quelle coordonnée  $\mathbf{u}$ . En pratique, les ordres de moment 1 et 2, respectivement la moyenne  $\mathbb{E}$  et la variance  $Var(X)$ , sont suffisantes à prouvés [SOA \_ Noise]. Un bruit stationnaire respecte ainsi l'équation ?? :

$$\begin{aligned} \mathbb{E}[X(\mathbf{u})] &= constante \\ Var[X(\mathbf{u})] &= constante \end{aligned} \quad (13)$$

Inversement, un bruit n'est pas stationnaire si sa moyenne et sa variance sur l'ensemble des réalisations dépend de la coordonnée  $\mathbf{u}$ . Il est ainsi possible de parler de variation spatiale [One \_ Noise] dans les motifs de la texture. Dans le cas d'un mélange de textures, même si ces dernières sont stationnaires, si les poids  $w_1, w_2$  sont non stationnaires, alors le mélange  $\mathcal{M}$  le sera également.

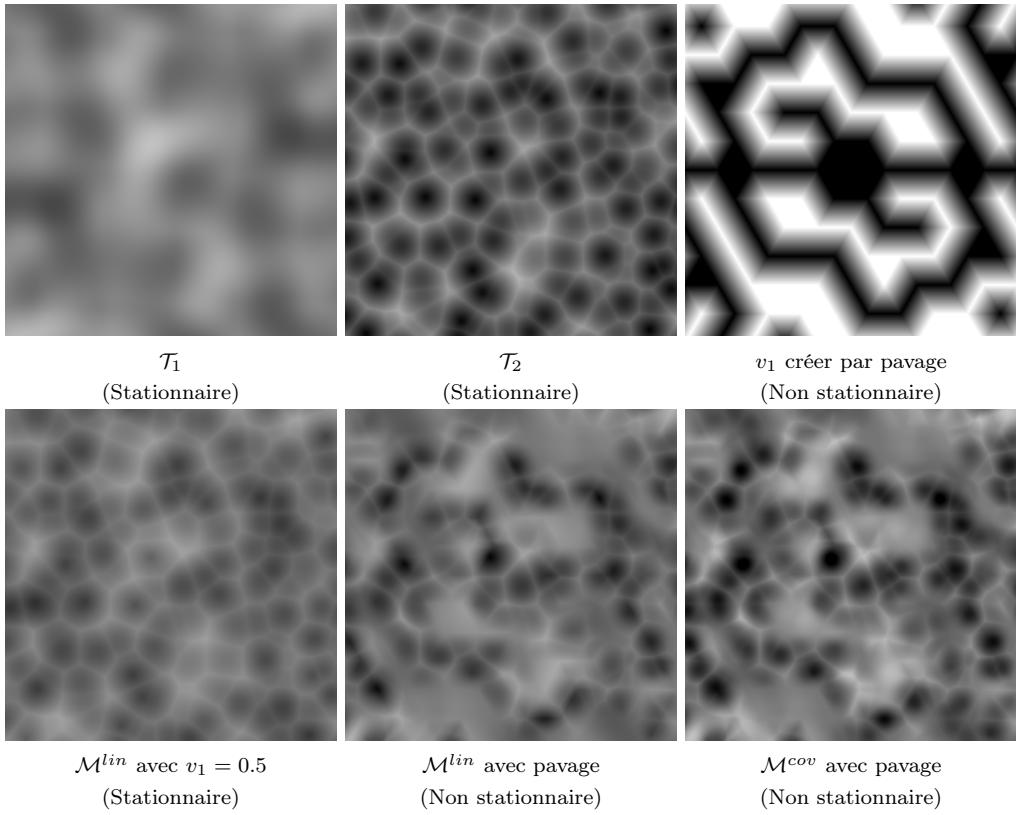


FIGURE 2 – Comparaison entre différentes méthodes de mélanges entre les textures  $\mathcal{T}_1$  et  $\mathcal{T}_2$ . Le pavage apporte des variations spatiales, même avec des entrées stationnaires. Le mélange préservant la variance  $\mathcal{M}^{cov}$  évite que le résultat ne tende vers une gaussienne, rendant le contraste (ou la variance) du résultat similaire à celui des entrées.

## 4 Création de variation spatiale par Pavage et Mélange

### 4.1 Méthode

L'algorithme de pavage et mélange [**HPnoise**] introduit une grille hexagonale de pavage, ainsi qu'un opérateur de mélange permettant de préserver la variance des entrées. Cependant ce système est utilisé sur une seule et même texture matricielle, afin d'en faire une texture procédurale. Mon objectif est de l'appliquer à deux bruits procéduraux différents afin d'obtenir des patterns à variation spatiale.

Pour cela, il est nécessaire d'attribuer à chaque cellule de la grille une texture. L'algorithme original applique une translation et une rotation aléatoire pour chaque cellule. Cette opération n'est pas nécessaire car toutes les textures utilisées sont procédurales. La variation spatiale est ainsi apportée par le choix d'attribution des cellules. Un exemple de notre implémentation est présenté dans la figure ??.

L'opérateur de mélange préservant la variance, montré dans l'équation ??, doit également être

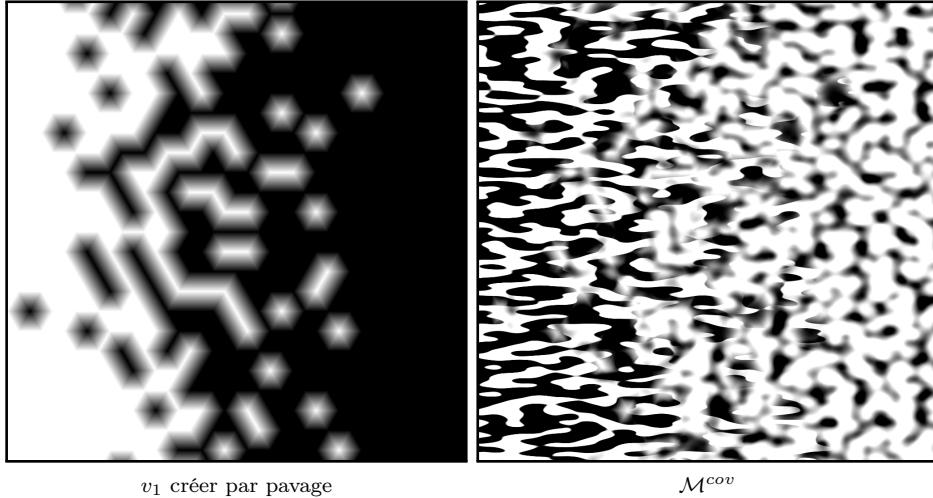


FIGURE 3 – Transition horizontale entre deux textures nettes en utilisant le pavage et mélange préservant la variance présenté dans l'équation ???. La pauvre qualité visuelle du résultat montre la limite de cette méthode qui ne s'adapte pas aux motifs ou à la netteté des entrées.

adapté, car il utilise l'espérance du mélange final, qui est connu dans le cas où chaque entrée provient de la même texture, mais qui est inconnu dans mon cas d'utilisation..

$$\mathcal{M}^{cov} = \frac{v_1 \mathcal{T}_1 + v_2 \mathcal{T}_2 - \mathbb{E}[\mathcal{M}^{cov}]}{\sqrt{v_1^2 v_2^2}} + \mathbb{E}[\mathcal{M}^{cov}] \quad (14)$$

Mon apport sur cette méthode a été de trouver la valeur de l'espérance du mélange qui satisfaisait l'équation ???. Elle est égale à la moyenne pondérée des espérances de chaque entrée :

$$\mathbb{E}[\mathcal{M}^{cov}] = v_1 \mathbb{E}[\mathcal{T}_1] + v_2 \mathbb{E}[\mathcal{T}_2] \quad (15)$$

L'impacte de cette formule est visible sur la figure ??.

## 4.2 Limitations

Bien que le pavage et mélange permette d'obtenir des bruits à variation spatiale en temps réel, la qualité des résultat varie grandement en fonction des entrées, comme le démontre la figure ???. Des textures nettes, par exemple, entraîneront une grande perte de cohésion des patterns. Malgré l'apport de préservation du contraste (ou de la variance) la qualité des transition reste très proche de la méthode linéaire.

Obtenir de meilleur résultats nécessiterait une méthode sensible aux patterns et à l'histogramme des textures que l'on souhaite mélanger. Cela permettrait une bonne adaptabilité à tout les types de bruits.

## 5 Création de variation spatiale par Mix-Max

### 5.1 Motivations

L'opérateur de mélange Mix-Max permet de créer des transitions (de textures matricielles) paramétrables et s'adaptant au contenu de toutes les entrées [mixmax]. Son principe est de représenter les textures comme des éléments pouvant être soit totalement opaques, soit totalement écrasés lors du mélange. Cette méthode est qualifiée de mélange binaire. Pour savoir quelle texture doit être choisie à chaque endroit, des cartes de priorités  $\mathcal{S}_i$  sont attribuées à chaque entrée. Pour finir, les priorités sont décalées par les poids de mélange, rendant, par exemple, la texture  $T_i$  très présente dans le mélange lorsque  $v_i$  est grand. Les opacités  $w_1$  et  $w_2$  ainsi créées par le Mix-Max respectent l'équation ?? [mixmax] :

$$w_1(\mathbf{u}) = \begin{cases} 1, & \text{Si } \mathcal{S}_1(\mathbf{u}) + v_1(\mathbf{u}) > \mathcal{S}_2(\mathbf{u}) + v_2(\mathbf{u}) \\ 0, & \text{Sinon} \end{cases} \quad (16)$$

$$w_2 = 1 - w_1$$

Dans son état actuel, le Mix-Max possède plusieurs limitations pour mon cas d'utilisation. Premièrement, ce mélange n'est pas équilibré, rendant impossible un contrôle précis de la transition. Deuxièmement, les méthodes de filtrage proposées ne sont compatibles qu'avec des textures matricielles.

### 5.2 Priorité équilibré

Équilibrer le Mix-Max permettrait d'obtenir une probabilité de présence de chaque texture équivalente à son poids initial :

$$\begin{aligned} P(w_1 = 1) &= v_1 \\ P(w_2 = 1) &= v_2 \end{aligned} \quad (17)$$

Cette propriété permettrait à n'importe quel utilisateur de contrôler précisément le niveau de présence de chaque texture.

Pour cela, il faut d'abord un socle commun entre toutes les textures de priorités, car ces dernières peuvent posséder n'importe quel histogramme ou contenu, tant que leur moyenne est proche de 0.5 [mixmax]. Si chaque texture de priorité subissait un aplatissement d'histogramme, elles respecteraient alors l'équation ??, rendant facilement prédictible la probabilité de chaque résultat :

$$P(\mathcal{S}_i > x) = x \quad (18)$$

Avec des priorités à l'histogramme plat, il est possible de trouver un décalage parfait  $\Delta_i$  pour chaque priorité  $\mathcal{S}_i$  qui équilibre le mélange :

$$\Delta_i = \frac{1}{2} \operatorname{sgn} \left( v_i - \frac{1}{2} \right) \left( 1 - \sqrt{1 - 2 \left| v_i - \frac{1}{2} \right|} \right) \quad (19)$$

Le résultat de l'application de cette formule est visible sur la figure ??.

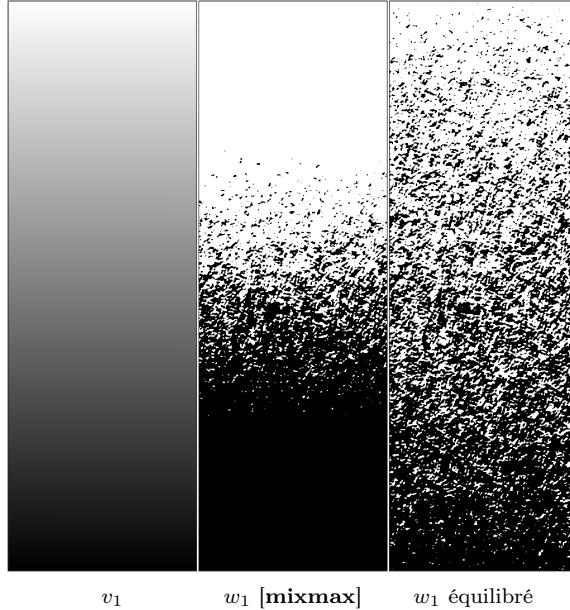


FIGURE 4 – Comparaison de l'équilibrage de la méthode Mix-Max entre deux textures procédurales similaires (Local Random Phase Noise [**LRPN**]) à histogramme gaussien. La première colonne montre le poids original. La seconde colonne montre l'opacité générée par le Mix-Max [**mixmax**] vu à l'équation ???. La dernière colonne montre l'opacité générée par le Mix-Max équilibré utilisant le décallage vu à l'équation ?? ainsi qu'un aplatissement d'histogramme des priorités. L'équilibre du mélange final est bien visible dans la répartition de l'opacité qui suit correctement le poids initial.

### 5.3 Priorité de bruits procéduraux

L'aplatissement de priorité matricielle n'est qu'une étape de pré-traitement simple à effectuer. Cependant, l'utilisation d'un bruit procédural en texture de priorité nécessite un aplatissement en temps réel de ce dernier.

Une telle opération est réalisable à l'aide de la fonction de répartition  $F_{\mathcal{B}}$  du bruit  $\mathcal{B}$ . Une version uniformément distribué de ce bruit  $\mathcal{U}$  est alors obtenable à l'aide de la formule :

$$\mathcal{U}(\mathbf{u}) = F_{\mathcal{B}}(\mathcal{B}(\mathbf{u})) \quad (20)$$

La fonction de répartition de n'importe quelle texture peut être obtenue en intégrer son histogramme. Dans mon implémentation, une discréttisation de cette fonction sur 16 à 32 points, obtenus en sommant les valeurs de l'histogramme, est suffisante pour obtenir de bons résultats. Cette formule convient à n'importe quel bruit procédural stationnaire. La figure ?? montre les résultats obtenus avec cette méthode.

### 5.4 Micro-priorité et filtrage pour un mélange équilibré

Afin d'utiliser l'opérateur Mix-Max pour créer des textures affichées à l'écran, il est nécessaire d'avoir une méthode de filtrage analytique. Pour cela, le papier original [**mixmax**] a introduit une

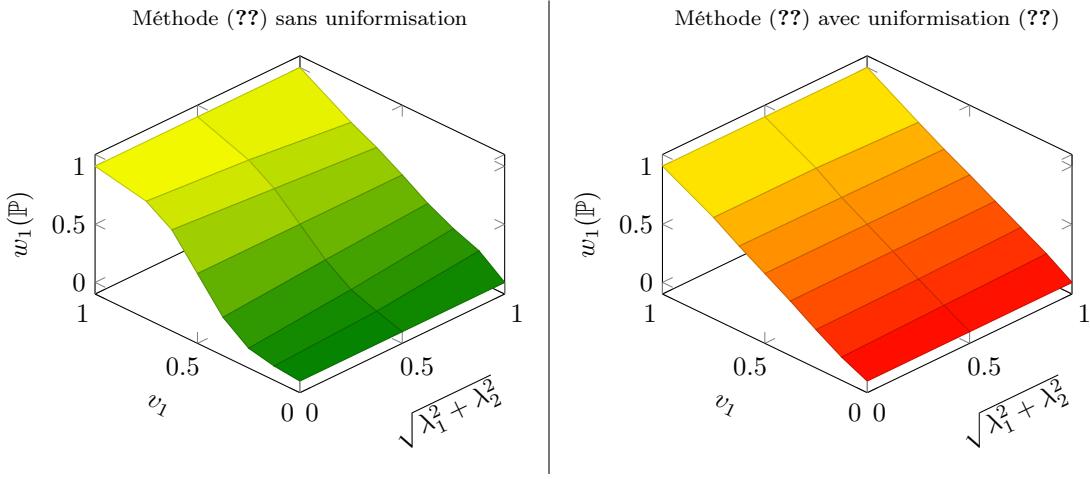


FIGURE 5 – Caption

équation permettant de calculer l’empreinte de l’opacité d’une texture sur une zone  $\mathbb{P}$  :

$$w_1(\mathbb{P}) = 1 - \Phi \left( \frac{\mathcal{S}_2(\mathbb{P}) + v_2(\mathbb{P}) - \mathcal{S}_2(\mathbb{P}) + v_2(\mathbb{P})}{\sqrt{\sigma_1^2 + \sigma_2^2 + \lambda_1^2 + \lambda_2^2}} \right) \quad (21)$$

Où  $\Phi$  est la fonction de répartition de la loi normale standard,  $\sigma_i^2$  la variance de la texture  $\mathcal{S}_i$  sur la zone  $\mathbb{P}$  et  $\lambda_1$  la micro-priorité (ou variance de base) de la texture, utilisée pour représenter les micro-détails de la texture en lissant le mélange.

Cependant, lorsque l’on applique l’aplatissement de l’histogramme de la priorité et le décallage  $\Delta_i$  vu à l’équation ??, le résultat n’est pas équilibré.

## 5.5 Méthode d’estimation de la variance de textures procédurales

## 6 Conclusion et apports

### 6.1 Pavage et mélange

### 6.2 Mix-Max

### 6.3 Perspectives et améliorations