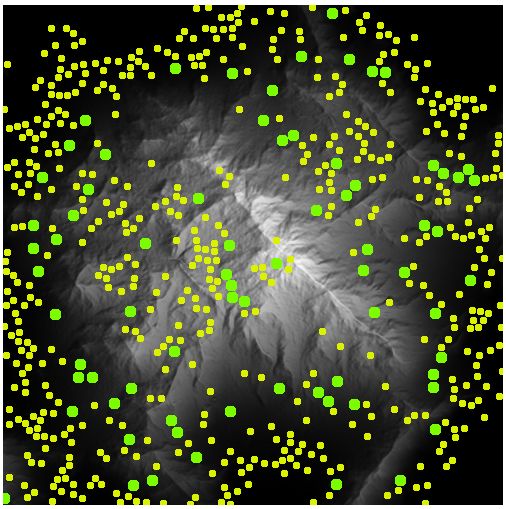
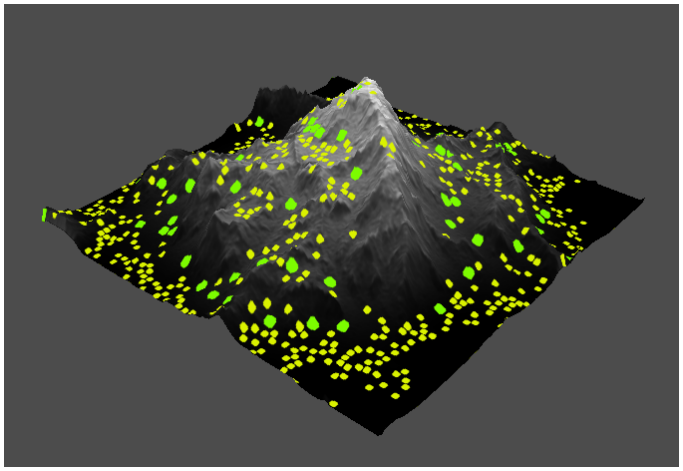
# Rapport du TP3 du module de Modélisation de Mondes Virtuels

Génération de végétation

Antoine BRALET (p2020363) et Guillaume DURET (p2021346)





8 Janvier 2021

# 1 Introduction

Le sujet choisi a été le top down. Ce sujet consiste à distribuer des végétaux en fonction des différentes caractéristiques du terrain telles que l'accessibilité, la pente ou encore l'humidité du terrain. L'idée est donc de générer des végétaux statistiquement en fonction de ces paramètres. Il sera alors réalisé une distribution de disque de poisson à l'aide d'un algorithme de dart-throwing (lancer de fléchettes) initialement introduit dans [Cook, 1986] puis développée jusqu'à obtenir des techniques très avancées comme dans [White et al., 2007]. De plus il sera étudiée la gestion de plusieurs espèces distribué sur un même terrain en considérant un ordre de priorités des espèces. Le lien permettant d'accéder au code est le même que le précédent rapport, à savoir https://github.com/Guillaume0477/Monde_virtuel, le code a été repensé puis amélioré afin de prendre en compte la gestion de végétation en top-down.

# 2 Principe de distribution de végétations

La première étape de la distribution de végétation est de réaliser une distribution de disques de poisson sur le terrain en entier en fonction du rayon de l'espèce choisie. (plusieurs méthodes ont possible (section 4))

La deuxième étape est que pour chaque point obtenu dans la distribution de la première étape, il faut tester statistiquement si l'arbre reste sur la carte en fonction de ces chances de survie. Ces chances de survie dépendant des caractéristiques du terrain. En effet chaque espèce de végétation possède ses propres caractéristiques, dans le cadre du TP deux espèces ont été créées possédant les caractéristiques du tableau ci dessous avec $f_{min}$ et $f_{max}$ des fonctions de $[0,1]$ dans $[0,1]$ définies dans les équations 1 et 2. À l'aide de ce tableau on réalise un champ de densité décrivant la probabilité de survie de chaque espèce en chaque point en prenant le critère de terrain avec la probabilité la plus faible avec de privilégier les contraintes de terrain et si une des caractéristique est impossible l'arbre sera naturellement pas placé.

Dans le cas de plusieurs espèce le principe reste le même mais on considère qu'une espèce est dominante est distribuée sur le terrain, ainsi la deuxième espèce réalise sa distribution sur les espaces libres du terrains.

$$f_{max}(x, inf, 1) = \begin{cases} 0 & \text{si } x \in [0, inf] \\ x & \text{sinon.} \end{cases} \quad (1) \qquad f_{min}(x, 0, sup) = \begin{cases} 1-x & \text{si } x \in [0, sup] \\ 0 & \text{sinon.} \end{cases} \quad (2)$$

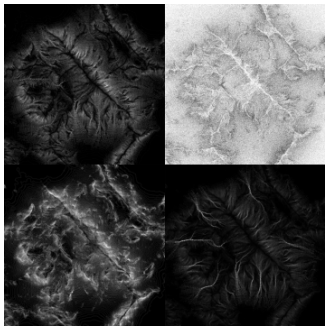
| Param terrain x / Végétations | Humidité | Pente | Stream Power | Accessibilité |
|---|---|---|---|---|
| Sapin | $f_{max}(x, 0.1, 1)$ | $f_{min}(x, 0, 0.2)$ | $f_{min}(x, 0, 0.01)$ | $f_{max}(x, 0.6, 1.0)$ |
| Buisson | $f_{min}(x, 0, 0.2)$ | $f_{min}(x, 0, 0.2)$ | $f_{min}(x, 0, 0.01)$ | $f_{max}(x, 0.6, 1.0)$ |

Table 1: Affichage des caractéristiques de chaque espèce en fonction des caractéristiques de terrain. De plus il est affiché les complexités de l'obtention des caractéristiques du terrain

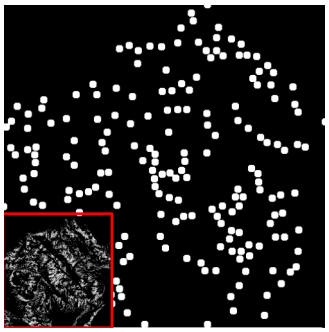
# 3 Résultats

On peut noter aisément à partir des Figures 2 que les résultats obtenus sont parfaitement cohérents avec nos attentes. En effet les densités générées en coin des Figures 2b et 2c correspondent bien aux différentes caractéristiques prises en compte de la Figure 2a et en fonction des poids associés à chacune présentée dans le tableau 1. De plus il est possible de noter que les sapins et buissons générés semblent bien générés aléatoirement à l'aide du Dart Throwing dans les zones où les densités sont respectivement les plus fortes. Notons également que cet algorithme a été implémenté afin de pouvoir générer une cohabitation entre plusieurs végétaux ayant différentes cartes de probabilité, il est donc possible de rassembler les deux cartes de distribution présentées en 2b et 2c afin d'obtenir des résultats similaire à l'image de garde de ce rapport.

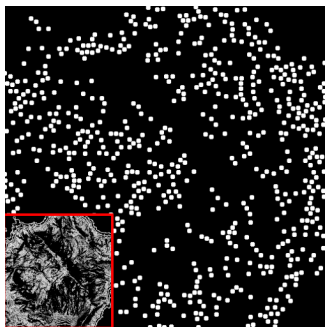
Figure 2: Affichage des caractéristiques de terrains, ainsi que les densité et la distribution associé des deux espèces : sapin et buisson.



(a) Caractéristiques de terrain (Humidité, Accessibilité, Pente, Stream Power)



(b) Distribution de sapins générée à partir de la densité en coin



(c) Densité de buissons générée à partir de la densité en coin

# 4 Amélioration de l'algorithme et complexité

La première idée naïve du Dart Throwing est de tirer un très grand nombre de points aléatoirement sur le terrain et vérifier si le disque du nouveau point ne touche pas le disque d'un ancien point c'est à dire que la distance des deux points est supérieur à la somme des deux rayons. Cependant avec cette méthode il est très difficile de savoir quand arrêter l'algorithme et de savoir si le terrain est rempli ou pas. En pratique il est donc tirer un très grand nombre de point de l'ordre de grandeur de $n^2$ pour être statistiquement sûr que tout le terrain à été parcouru. Afin d'améliorer le temps de calcul du Dart Throwing, l'idée est de séparer le terrain en case de longueur $3r$ (avec $r$ le rayon de l'arbre) et de tirer aléatoirement une position dans chaque case. Ceci se fait cellule par cellule de sorte qu"il suffit de calculer la distance entre la position candidate et celle des arbres des cellules voisines (au maximum 4) pour savoir si la position est adéquate. Si telle n'est pas le cas, on tire de nouveau aléatoirement une position dans la case jusqu'à trouver une position qui soit valable. Cette méthode est détaillée dans [Cohen et al., 2003] en section 3.3. On peut alors noter d'après le tableau 2 que ce processus est nettement meilleur ce qui le conduit aux temps de calcul présentés dans le tableaux 2. En effet la complexité de l'algorithme amélioré est en $\mathcal{O}(NB^2_{grille})$ avec $NB_{grille}$ le nombre de case créé de coté 3*r. Le premier algorithme possède une complexité en $\mathcal{O}(n^2)$ avec $n$ la dimension de l'image auquel il faut prendre en compte que chaque tirage va vérifier la distance à tous les points déjà placés ce qui a une complexité entre 0 au début à $n^2/r_2$ vers la fin résultant en une complexité en $\mathcal{O}(n^4)$. Ce phénomène n'est pas à prendre en compte dans l'algorithme accéléré car au maximum 4 arbres sont testés pour chaque case. Plus précisément en tenant compte que $n = NB_{grille} * 3$ et de la vérification des voisins on obtient une complexité de $\frac{4*n^2}{9*r^2}$

| Terrain / Algorithme | 100x100 | 200x200 | 300x300 | 500x500 | 700x700 | 1000x1000 |
|---|---|---|---|---|---|---|
| Complexité initial | 4/9 | 58/134 | 243/674 | 1788/4860 | 6k/19k | 36k/102k |
| Amélioré | 0.04/0.04 | 0.057/0.075 | 0.056/0.09 | 0.133/0.32 | 0.24/0.54 | 0.53/1.15 |

Table 2: Affichage des temps de calcul en ms pour le lancer de fléchette pour $r = 10$ et $r = 6$ (affiché $t1/t2$) en fonction de la dimension de l'image d'origine.

Il est cependant notable qu'avec cet algorithme efficace la distribution ne suit plus une distribution de Poisson exacte et se retrouve légèrement étalée. Plusieurs algorithmes efficaces procèdent différemment en modifiant moins la distribution et en procédant avec cette fois-ci avec des cases de coté inférieur à $r/sqrt(2)$

et subdivisant les cases qui ont été refusées comme précisé dans [White et al., 2007]. Cependant travaillant sur une distribution d'arbre dont le respect de la distribution parfaite de Poisson n'est pas obligatoire, l'argument retenu pour ce TP a été d'avoir de meilleurs performances tout en conservant un rendu réaliste.

# References


M. F. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. *ACM Transactions on Graphics (TOG)*, 22(3):287–294, 2003.

R. L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics (TOG)*, 5(1):51–72, 1986.

K. B. White, D. Cline, and P. K. Egbert. Poisson disk point sets by hierarchical dart throwing. pages 129–132, 2007.