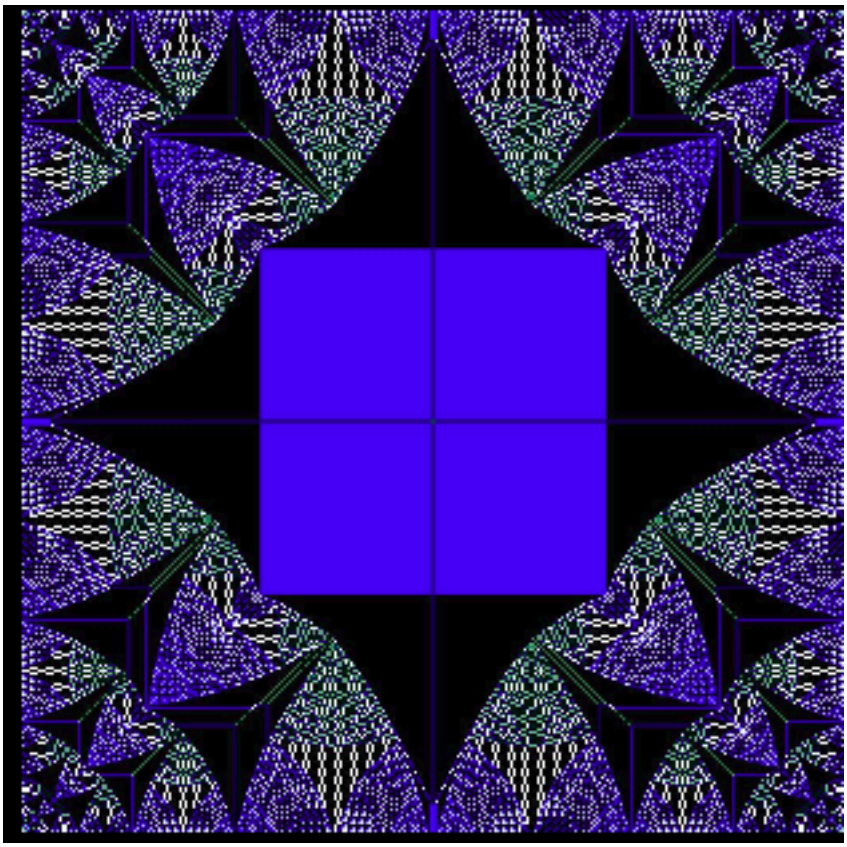
Travail d'étude et de recherche
Sandpile on Tiling



Auteurs :
FERSULA Jérémy
DARRIGO Valentin
ZHANG Lin

Encadrant :
PERROT Kévin

25 mai 2019

Résumé

Le but premier de ce projet consiste à développer une application web qui implémente le modèle de piles de sable Abéliennes sur des parties finis du plan pour des pavages de Penrose, afin d'y étudier *a posteriori* le comportement du modèle.

The main goal of this project consists in developing a web app which implements this models on finite parts of the plane for Penrose tilings, in order to study *a posteriori* the model behavior on it.

Table des matières

1 Introduction

Le modèle des piles de sables Abéliennes est un modèle d'automate cellulaire relativement récent, puisque formalisé en 1987 par Per Bak, Chao Tang and Kurt Wiesenfeld [2]. La règle de ce modèle est simple : sur un graphe non-orienté quelconque, chaque sommet contient un nombre entier fini de grains de sable - et lorsque un sommet contient exactement ou plus de grains que son degré, il donne un grain à chacun de ses voisins et en perd autant. Ce modèle est en particulier applicatble aux pavages du plan. Dans ce cadre, il présente des propriétés géométriques remarquables, par l'apparition sous certaines conditions de formes fractales difficiles à prévoir mais simple à produire expérimentalement. Le projet s'intéresse ainsi au développement d'une application web qui implémente ce modèle sur des pavages potentiellements quelconques, avec pour objectif d'en étudier le comportement sur des pavages de Penrose. Github : https://github.com/hehaOhyeah/JS-Sandpile

2 Contexte théorique

Les piles de sables Abéliennes présentent un certain nombre de propriétés non triviales prouvées par un formalisme mathématique dont l'exposition complète dépasserait le cadre de ce projet. Cependant, pour la bonne compréhension de l'application et des comportements observés, il convient d'énoncer quelques-une de ces propriétées, et d'exposer un formalisme simplifié. Pour en savoir plus, voir [4].

Définition 1 *Une configuration sur un graphe (E, V) est une application* $z : V \to \mathbb{N}$ *qui associe à chaque sommet son nombre de grains de sable.*

Une configuration est dite stable si et seulement si $\forall x \in V$, $z(x) < deg(x)$. Un éboulement (*toppling*) T_v du sommet $v \in V$ avec une dissipation $\alpha \in \mathbb{N}$ est la transformation définit comme suit :

$$z(v) \mapsto z(v) - deg(v) - \alpha \quad \text{si z}(v) \geq \text{deg(v)} + \alpha$$

et pour tout voisin u de v :

$$z(u) \mapsto z(u) + 1$$

Les propositions et théorèmes exposés ici sont valables sous réserve qu'au moins un sommet ait une dissipation > 0.
Plus formellement :

Définition 2 *L'éboulement du sommet v est une application*
$T_v : \mathbb{N}^V \to \mathbb{N}^V$, $z \mapsto z^*$ *et* $\forall v \in V$, $z^*(v) = z(v) - \mathbb{1}_{\mathbb{N}}(v - deg(v) - \alpha_v)(deg(v) + \alpha_v) + \Sigma_{u \sim v} \mathbb{1}_{\mathbb{N}}(u - deg(u) - \alpha_u)$.

Remarque : Les éboulements commutent sur les sommets instables.

Théorème 1 *Il existe une énumération* $(v_1, v_2, ..., v_N) \in V^N$ *unique à la permutation près telle que la transformation* $\Pi_{i=1}^{N} T_{v_i}$ *aboutisse à une configuration stable. Cette transformation est appelée stabilisation.*

On note $\star$ l'opération agissant sur deux piles de sable, consistant à ajouter les grains de chaque tuiles et à stabiliser la configuration obtenue.

Proposition 1 *L'ensemble des configurations stables munit de l'opération* $\star$ *forme un monoïde commutatif.*

Cette proposition est tout à fait intuitive, lorsque l'on stabilise la somme de deux configurations stables, on obtient une configuration stable, et l'élément identité est la configuration uniformément égale à zero.

Définition 3 *Une configuration est dite récurrente si il est possible de l'atteindre à partir de n'importe quelle configuration initiale, par ajout de grains et stabilisation successifs.*

Le nombre de configuration récurrente pour un graphe donné est connu [3] :

Théorème 2 *Le nombre de configuration récurrente est exactement égal au déterminant de la matrice de l'opérateur éffondrement, c'est à dire le laplacien du graphe (Δ) auquel on ajoute la matrice diagonale des dissipations.*

Exemple : pour une grille carrée 2x2 :

$$\Delta = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & 0 & -1 \\ -1 & 0 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix} \text{ D (dissipation)} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

$$\text{Nombre de configuration stable} = 4^4 = 256$$
$$\text{Nombre de configurations récurrentes} = \text{Det}(\Delta + \text{D}) = 192$$

Proposition 2 *L'ensemble des configurations récurrentes munit de l'opération $\star$ forme un groupe commutatif (abélien).*

Cette proposition constitue la clef de voûte du modèle. La configuration identité de ce groupe pour un pavage présente des formes fractales difficiles à déterminer *a priori*. Cependant, il est facile d'étudier expérimentalement ces configurations.

3 Implémentation

3.1 Motivations

Avant de commencer notre application, nous avons testé des logiciels existants afin de nous familiariser avec le modèle et afin de sefaire une idée des fonctionnalités à implémenter.

— **Logiciels existants**
Nous avons trouvé des exemples d' applications de piles de sables :
The Abelian Sandpile Model [1], Sandpiles [2].

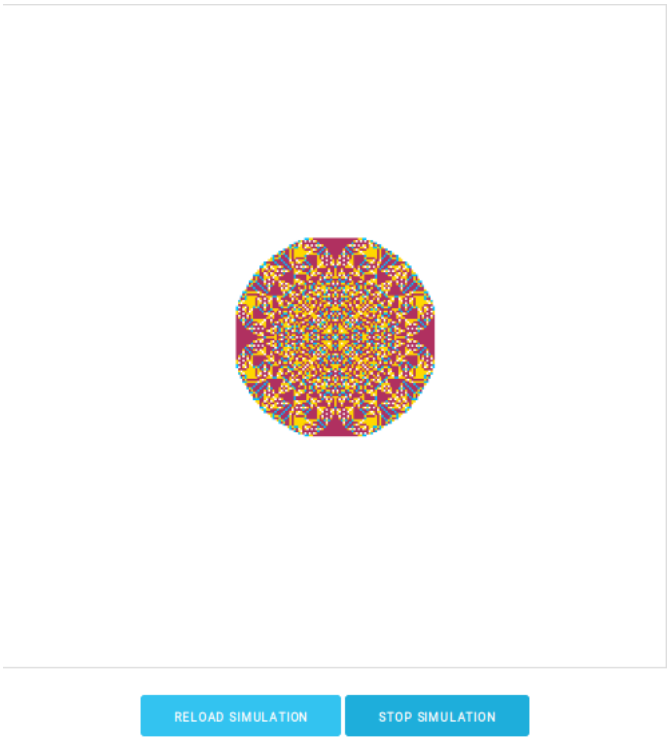


Figure 1: The Abelian Sandpile Model web application

La premiere application sur **la figure 1**, est une application web, qui simule des piles de sables.

Elle présente des fonctionnalités telles que :
— Redémarrage de simulation, Arrêt de la simulation.
Les **qualités** de cette application sont :
— Application web en JavaScript.
— Opensource.

1. http ://www.natureincode.com/code/various/sandpile.html

— Pas besoin d'installation.

Les **défauts** de cette applications sont :

— Manque d'interaction avec les utilisateurs.

— Les choix de nombres de grains, la vitesse d'éboulement, etc...

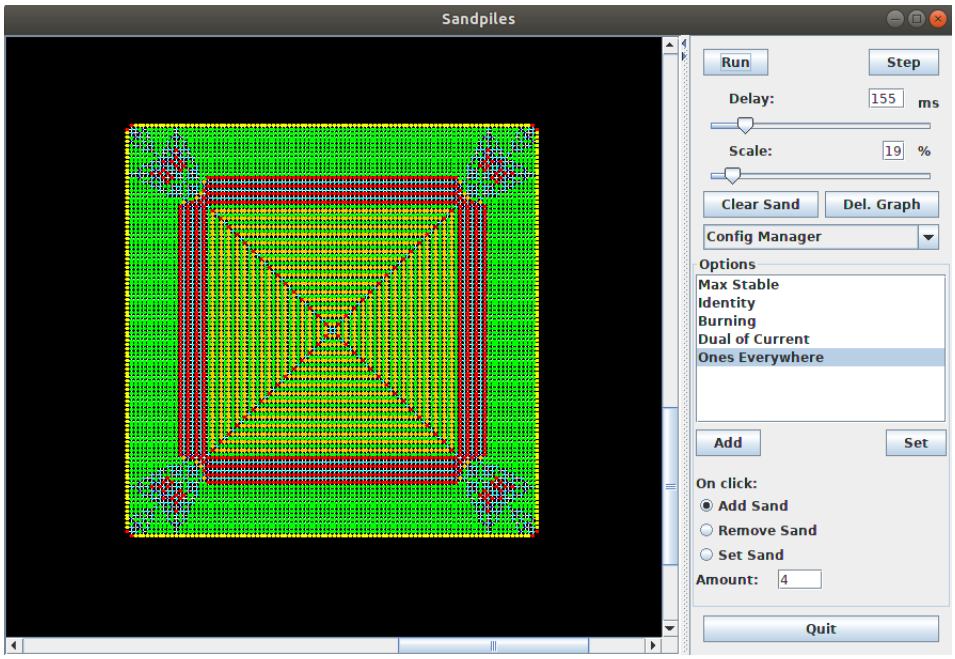


Figure 2: Sandpiles application

La deuxième application sur la **figure 2**, est un logiciel multi-plateforme developpé en java, qui est très complet.

L'application présentes des fonctionnalités telles que :

— Éditeur de configuration flexible pour créer, enregistrer et ajouter des configurations.

— Ajout et soustraction de configurations complexes :
  — Identity, Minimal, Burning, Inverse, Max Stable, Random.

— Plusieurs mode de visualisations :
  — Mode 3D, Nombre de grains, Stabilité, Total des tirs (utile pour visualiser les avalanches, Changement du nombre de grains.

— OpenGL pour un rendu rapide.

— Arrêt de la simulation, faire du drag and drop sur la grille créée.

Les **qualités** de cette application sont :

— Déploiement multi-plateforme.

— Opensource, Ergonomique, Travail de manière évolutive, Facilité

2. http ://people.reed.edu/ davidp/sand/program/program.html

d'installation, Reutilisabilité.

Les **défauts** de cette applications sont :
— Pas d'import/export de configurations.
— L'application est instable selon le système, à cause de IcedTea Java.
— Choix des couleurs inexistant.

Nous nous sommes inspirés de la seconde application, et nous avons amélioré certains points.

— **Fonctionalités attendues**
  — Basiques (Play, Pause, choix du pavage, choix des couleurs, Zoom).
  — Augmenter et baisser la vitesse d'écoulement des grains de sables.
  — Sélection d'une tuile à la souris.
  — Choix parmi quelques pavages implémentés (carré, triangulaire, hexagonal, Penrose).
  — Sauvegarde et importation des configurations en local. Importation des pavages carrés personalisés.
  — Ajout, soustraction de configurations conplexes :
    — Identité, Un de partout, Dual, Max Stable, Aléatoire.

3.2 Choix techniques

Contraintes : L'application doit pouvoir être lancée sur un navigateur web. Elle être relativement rapide.

JavaScript : Langage par défaut afin de réaliser une application complexe coté client.

THREE.js : Package JS utile à la représentation de formes dans l'espace. La camera est déjà implémentée, nous pouvons facilement déplacer la vue, faire des zooms. Il est possible d'envoyer un rayon est de récupérer toutes les faces interceptées par ce rayon, nous utilisons cette méthode pour implémenter le clique (le fait d'ajouter un grain sur une tuile en cliquant par exemple).

JSON : Il permet de stocker des données textuelles, des chaînes de caractères (y compris des images en base64), nombres, tableaux (array), objets, booléens (true, false), la valeur null. Il est très simple à lire, comprendre et utiliser. Le format d'échange de données ouvertes, Il est pris en charge par nombre des langages.

Bootstrap : Il s'adapte à la taille de l'écran, que se soit tablette, ordinateur, smartphone, etc. Fonctionner sur tous les navigateurs. Embarquer un tas de composants prêts à l'usage (labels, badges, boutons, boutons dropdown, icônes, menus, navbar, progressbar, etc.) plutôt soignés.

3.3 Spécifications

L'application contient deux classes à partir desquelles les pavages sont construits. La première classe Tile représente une tuile, la seconde Tiling représente un pavage sous la forme d'un tableau de Tile. On expose ici les attributs principaux, sans en faire l'énumération complète. L'intégralité de la documentation est disponible sur la page github [1].

Tile

— Number id *Id de la tuile*
— Array<Number> neighbours *Ids des voisins*
— Array<Number> points *Points représentants la tuile dans THREE.js*
— Number toppleLimit *Limite d'éboulement de la tuile*

Tiling

— Array<**Tile**> tiles
— THREE.js Mesh mesh *Représentation du pavage*

On construit ainsi différents pavages, de sorte que tous évoluent par l'appel des mêmes méthodes de la classe Tiling. Ces méthodes dont pour la plupart le nom est explicite sont par exemple **iterate()** ou encore **addOne(Number index, Number amount)**.

1. https ://github.com/hehaOhyeah/JS-Sandpile

3.4 Piles de sables implémentées

Square Tilling : Pile de sables la plus classique, est une matrice de taille nm dont les tuiles sont des carrés.

Hexagonal Tilling : Matrice de taille nm dont les tuiles sont des hexagones.

Triangular Tilling : Piles de sables en forme de triangle composée de tuiles triangulaires.

Penrose HK : Pile de sables suivant le pavage de penrose P2 (kite and dart). La construction de ce pavage se fait à partir d'une tuile half-kite.

Penrose HD : Pile de sables suivant le pavage de penrose P2 (kite and dart). La construction de ce pavage se fait à partir d'une tuile half-dart.

Penrose Sun : Pile de sables suivant le pavage de penrose P2 (kite and dart). La construction de ce pavage se fait à partir d'un solei (ensemble de dix tuiles half-kite).

Penrose Star : Pile de sables suivant le pavage de penrose P2 (kite and dart). La construction de ce pavage se fait à partir d'un solei (ensemble de dix tuiles half-dart).

Penrose HK Rhombus : Pile de sables suivant le pavage de penrose P3 (kite and dart). La construction de ce pavage se fait à partir d'une tuile half-kite.

Penrose HD Rhombus : Pile de sables suivant le pavage de penrose P3 (kite and dart). La construction de ce pavage se fait à partir d'une tuile half-dart.

Penrose Sun Rhombus : Pile de sables suivant le pavage de penrose P3 (kite and dart). La construction de ce pavage se fait à partir d'un solei (ensemble de dix tuiles half-kite).

Penrose Star Rhombus : Pile de sables suivant le pavage de penrose P3 (kite and dart). La construction de ce pavage se fait à partir d'un solei (ensemble de dix tuiles half-dart).

Square Moore : Comme la piles de sables carrés mais avec un voisinage de Moore.

4 Penrose

Dans cette partie nous allons parler en profondeurs des pavages, comment ils fonctionnent et comment le générer .Les pavages de Penrose ont été inventé en 1970 par le mathématicien et physicien britannique Roger Penrose.

Il existe 3 types de pavage, ici on s'intéressera à seulement deux d'entre eux.

NB : toutes les images qui suivent ont été généré par une version légèrement modifiée de notre application

4.1 Le pavage kite and dart (P2)

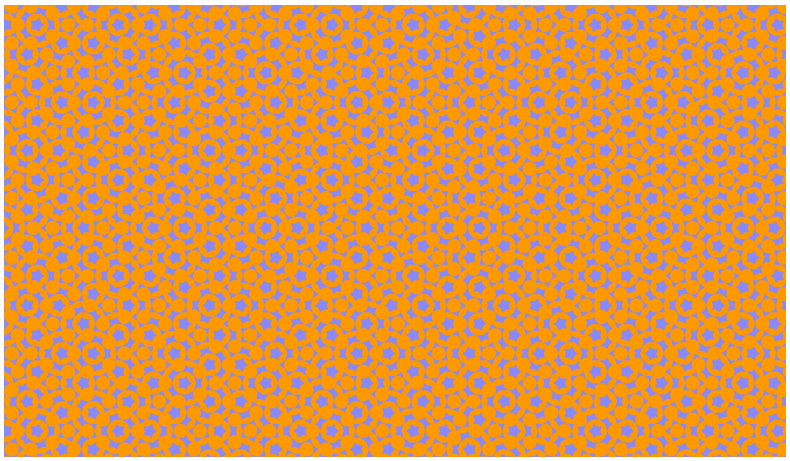


Figure 3: pavage de penrose kite and dart

Il existe plusieurs façons de générer un pavage de penrose, on a choisi d'utiliser la méthode par substitution

Cette méthode consiste à prendre deux sous tuiles : half-kite et half-dart

Figure 4: à gauche la proto tuile half-kite, à droite la proto tuile half-dart

Ces triangles sont aussi connus sous le nom de triangle d'or, le ratio de la longueur du plus long segment du triangle et le plus court est égale au nombre d'or

— la sous tuile half-kite à 1 angle de PI/5 et 2 de 2PI/5
— la sous tuile half-dart à 1 angle de à 3PI/5 et 2 de PI/5

La combinaison de ces sous tuiles forment les deux tuiles du pavage, les tuiles dart et kite :

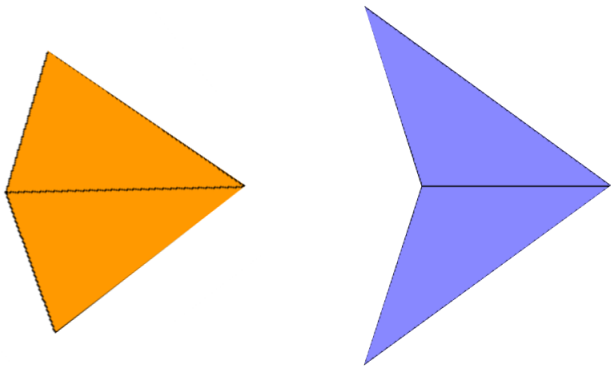


Figure 5: à gauche la tuile kite, à droite la tuile dart

Pour obtenir un pavage de penrose il suffit de partir d'une de ces tuiles et d'y appliquer leurs règles de substitution respectives n fois.

Ces règles sont les suivantes :

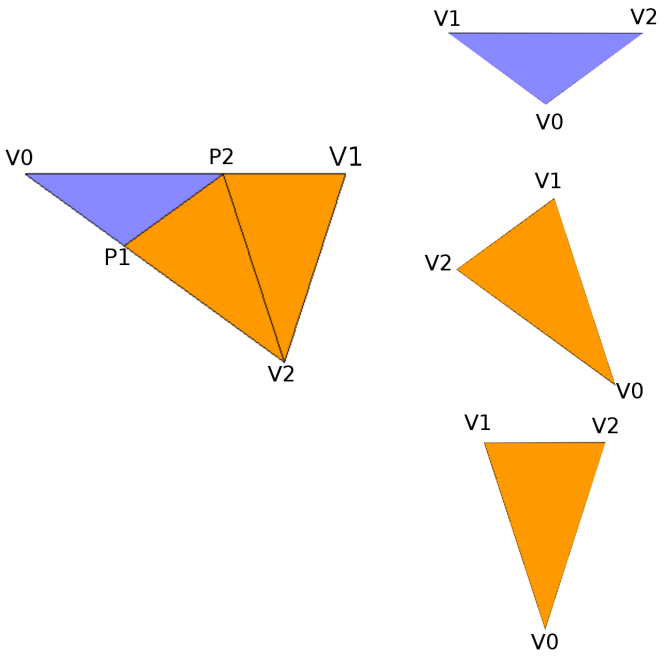


Figure 6: règle de substitution de la sous tuiles half-kite

V0, V1, V2 sont les sommets du triangle leurs dispositions sont importantes, elles définissent dans quel sens le triangle sera découpé. P1 et P2 sont des points intermédiaires définissant les sommets des nouveaux triangles pour la substitution.

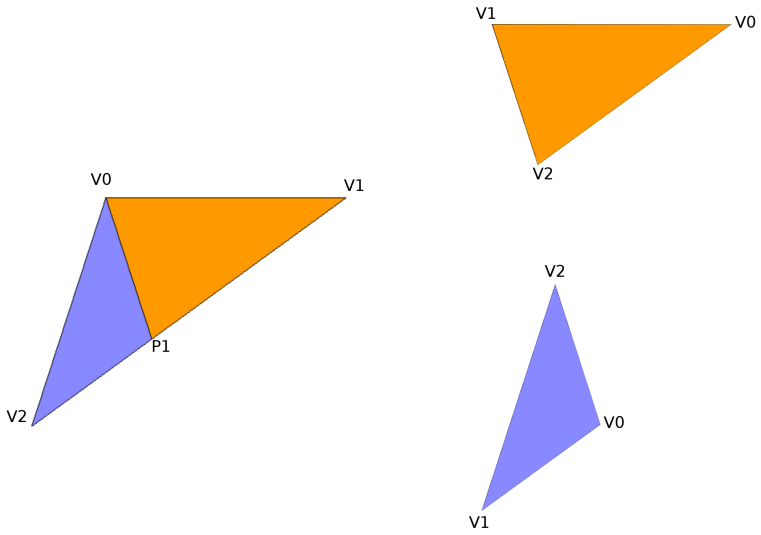


Figure 7: règle de substitution de la sous tuiles half-dart

Dans notre projet il ne suffit pas seulement d'afficher un pavage de penrose il faut aussi savoir quelle tuile est adjacente à quelle tuile. On a trouvé la solution suivante, tous les triangles contienent les références vers leurs arrêtes et chaque arrêtes contiennent les références de leurs triangles incidents. Ainsi on peut reconstituer les tuiles et déterminer les tuiles adjacentes.

4.2 Le pavage Rhombus (P3)

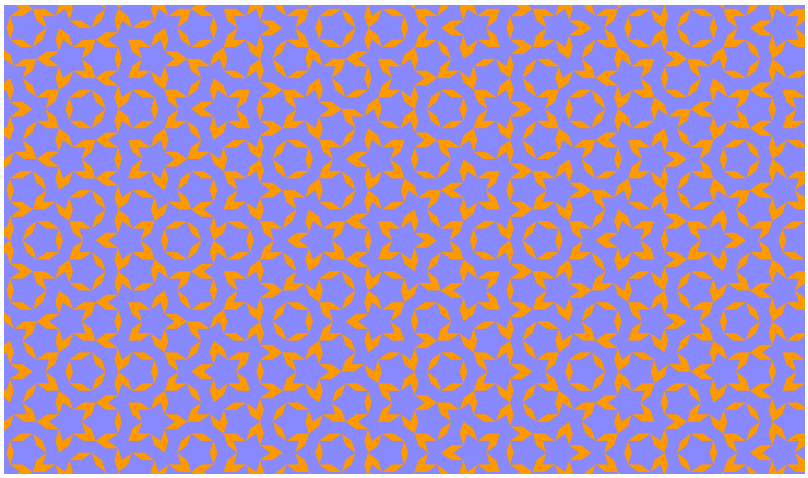


Figure 8: pavage de penrose Rhombus

On peut aussi générer un autre pavage par une méthode de substitution avec les mêmes triangles, cependant les tuiles et les règles de substitutions sont différentes.

Les tuiles composant le pavages sont les losanges "thick" et "thin" :

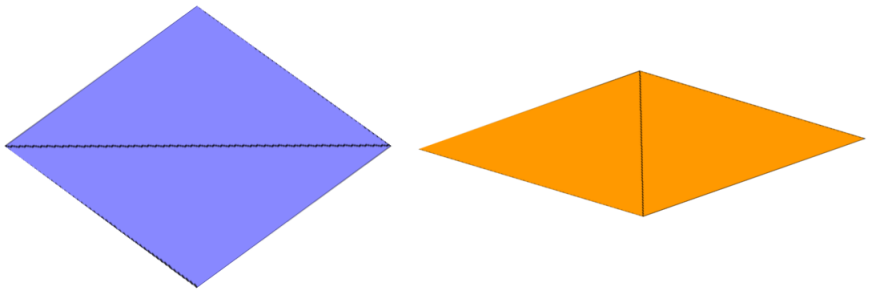


Figure 9: à gauche la tuile thick à droite la tuile thin

Les règles de substitution pour ce pavage sont :

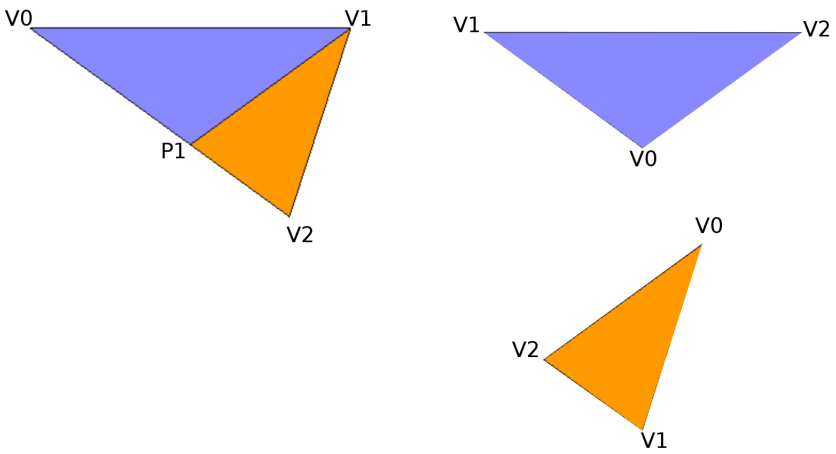


Figure 10: règle de substitution de la sous tuiles half-kite

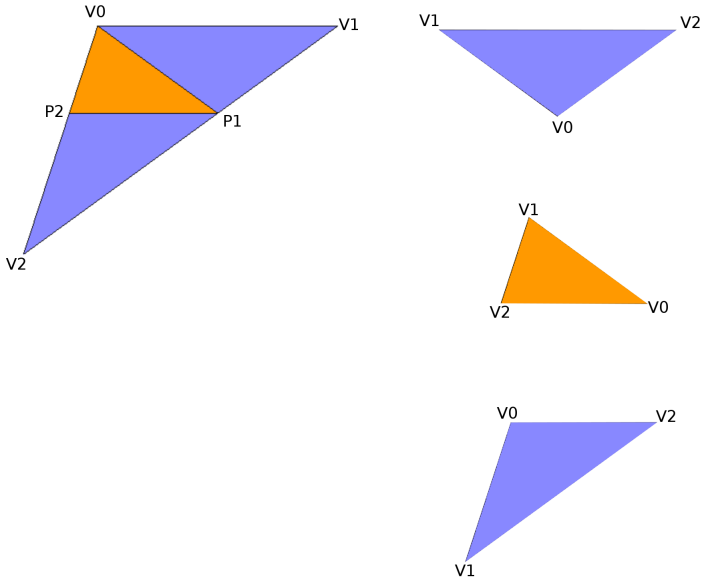


Figure 11: règle de substitution de la sous tuiles half-dart

5 Expérimentations

5.1 Mesures

L'objectif principal du projet rempli, nous avons pu nous pencher sur la génération de données statistiques pour certains éffondrement de piles de sable. Les expérimentations conduites ici sont l'ajout du maximum stable de grains partout, puis un ajout de 5000 grains de façon aléatoire.

Grille carrée, 100 x 100 tuiles

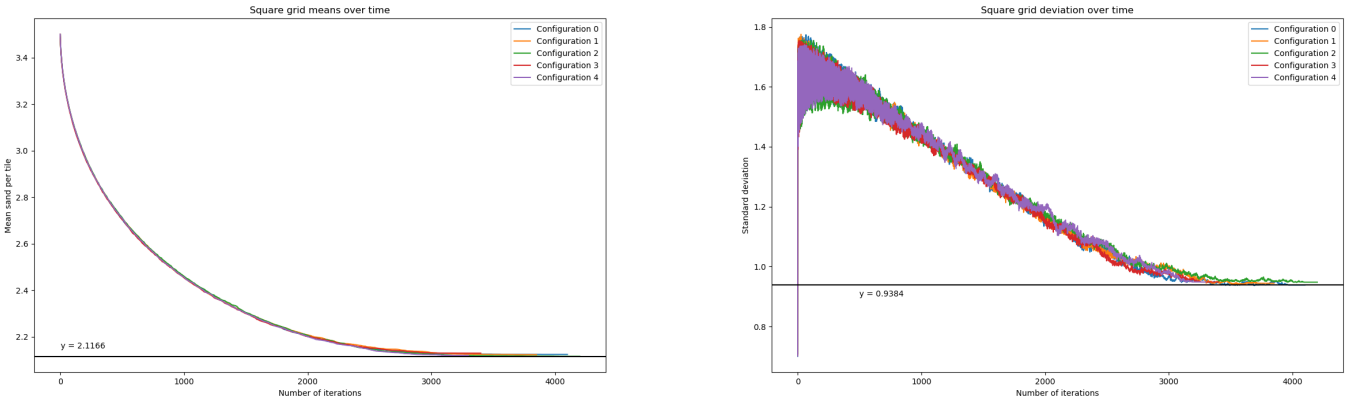


Figure 12: Mesures sur grille carrée

Penrose HK, 11 itérations - 23 047 tuiles

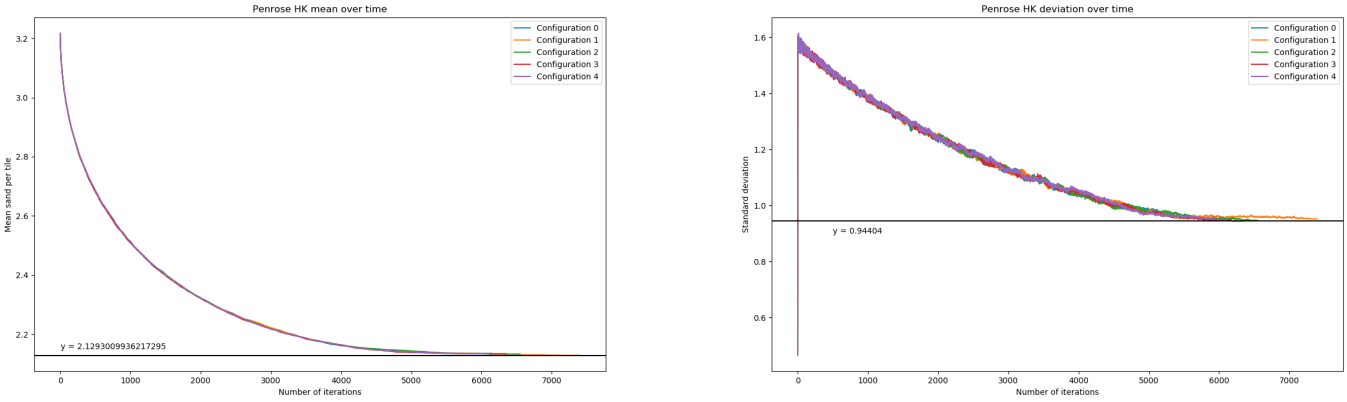


Figure 13: Mesures sur Penrose HK

Rhombus Penrose HK, 11 itérations - 14 212 tuiles

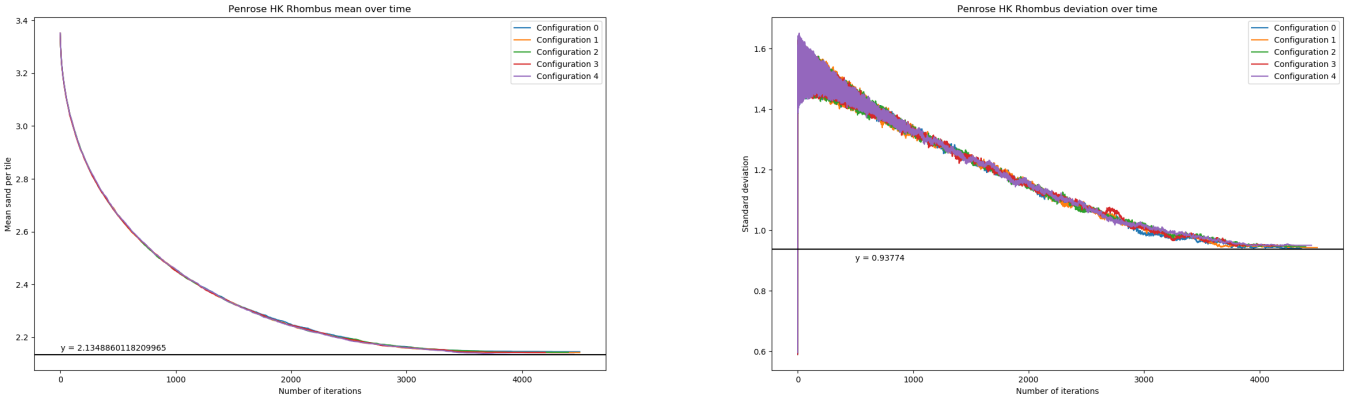


Figure 14: Mesures sur Rhombus Penrose HK

On observe sur ces graphiques que la moyenne et l'écart-type semble converger vers une valeur précise lors de la stabilisation. Il apparaît également que l'écart-type semble suivre une loi exponentielle inverse, mais que la moyenne décroît encore plus rapidement.

Une mesure pertinente sur les piles de sable est le nombre d'avalanches au cours du temps, c'est-à-dire le nombre d'éboulements engendrés par l'ajout d'un seul grain à la configuration. Cette mesure n'a pas pu être faite mais qui constitue un point intéressant pour la continuité du projet.

5.2 Observations

D'après Akiyama, Shigeki, et Katsunobu Imai. "The corona limit of Penrose tilings is a regular decagon." [1], sur les pavages de Penrose kite and dart ainsi que sur les pavages rhombus l'ensemble des tuiles les plus proche d'un centre fixé dessine un décagone régulier. En construisant un pavage de Penrose HK en 11 itérations, et en fixant le maximum stable de sable sur toute les tuiles, il suffit de déposer un grain afin d'observer la forme prise par l'ensemble des tuiles les plus proche de l'endroit où l'on a déposé le grain.

On observe ainsi que notre application corrobore leur résultat.

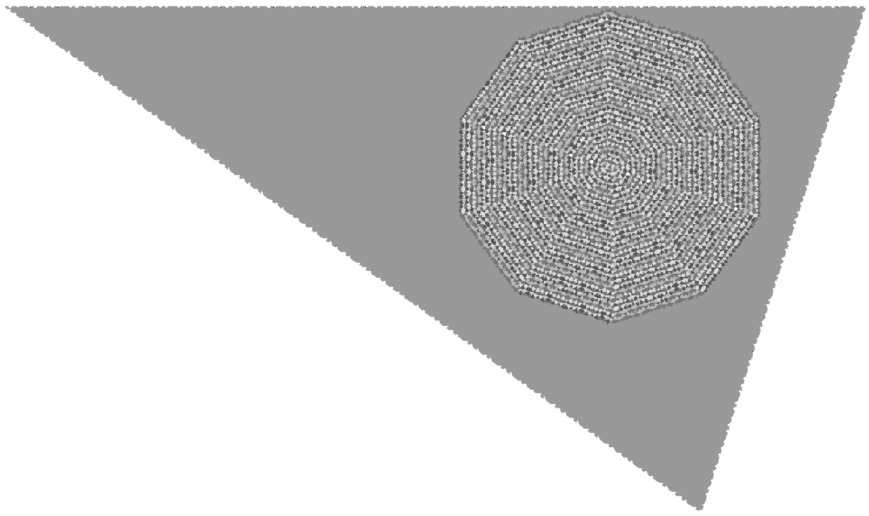


Figure 15: Formation d'un décagone sur pavage de Penrose HK - 11 itérations

6 Vie du projet

6.1 Méthode et progression

La gestion de projet est une partie indispensable au bon déroulement de celui-ci. Les compétences techniques de chacun sont à prendre en compte mais une gestion de projet correcte est aussi capitale pour mener le projet à bien le plus efficacement possible.

La Méthode Agile :
Afin de mener à bien notre projet, il nous a fallu déterminer les différentes tâches, les découper et les répartir équitablement. Nous avons organisé régulièrement des reunions avec notre encadrant PERROT Kévin, afin de faire le point sur l'avancement du projet, les problèmes rencontrés, les améliorations à faire. À l'issue de cette réunion nous écrivons un "sprint", compte-rendu qui definit les tâches futures.
Organisation type d'un sprint :

- Préparation du sprint
  Cela est fait en présence de l'ensemble de l'équipe, notre première action consiste à se mettre d'accord sur un planning. Durant la réunion de préparation, nous fixons un objectif bien détaillé du **Product Backlog**.
- Planification du sprint
  Les tâches de **Product Backlog** définies, nous estimons la durée, et

les priorités de chaque tâche.

— Le déroulement d'un sprint backlog
Une réunion quotidienne est organisée à distance, soit par ecrit ou par appel vocal. Chacun à son tour explique ce qu'il a accompli la veille, les problèmes éventuels qu'il rencontre, et ce qu'il a prévu de réaliser le jour même. Si quelqu'un a une solution à un problème exposé, il peut apporter son aide pour le résoudre. Cette réunion quotidienne va également permettre d'identifier rapidement tout retard dans la réalisation des tâches.

Versionning :
Nous utilisons Github pour gérer les versions du projet, il permet d'avoir une interface ergonomique, et aussi de travailler à distance. Nous avons référencé chaque modification sur Github.

6.2 Améliorations et perspectives

Bien que l'application remplisse son rôle et implémente les fonctionnalités énoncées dans le chapitre 3, quelques points peuvent encore être améliorés ou bien ajoutés.

Il peut être intéressant d'implémenter les pavages de Penrose avec un voisinage de Moore, c'est à dire en reliant une tuile avec celles qui lui partagent un sommet. Cette fonctionalité est difficile à implémenter, car les tuiles n'ont pas toutes le même nombre de tuiles voisines (dans le cas du voisinage "classique" -voisinage de Von Neumann- toute tuile a exactement 4 voisines). De ce fait il est difficile de déterminer la limite de grains maximum des tuiles se trouvant sur les bords du pavage. En effet le seul moyen de connaître cette valeur c'est de compter les tuiles voisines, or les tuiles sur les bords n'ont pas toutes leurs tuiles voisines qu'elle "devraient avoir".

Dans l'optique d'améliorer les performances, il peut être intéressant de paralléliser le calcul d'une itération. Il est impossible de réaliser une application sur plusieurs threads en JavaScript, mais l'option n'est pas inenvisageable, à l'aide par exemple de Web Workers.

Enfin, puisque l'architecture permet de définir simplement une pile de sable sur un graphe non orienté quelconque, il serait intéressant de permettre l'affichage et la création de graphes de manière intuitive directement dans l'application.

7 Conclusion

Dans la mesure où l'objectif de notre travail était réalisé une application web en JavaScript avec les fonctionnalités indiquées dans le chapitre 3, nous pouvons dire que l'objectif est atteint. L'application est relativement facile à utiliser, Il suffit de mettre les sources en lignes, et de se connecter à la page web. L'utilisteur n'a rien à installer, c'est le navigteur web qui fait tourner l'application. Vis-à-vis de la performance, malgré le fait que JavaScript ne permette qu'un seul thread, il est possible de lancer des simulations sur des grilles carrées 200 x 200 (40 000 tuiles) sans demander trop de ressources.

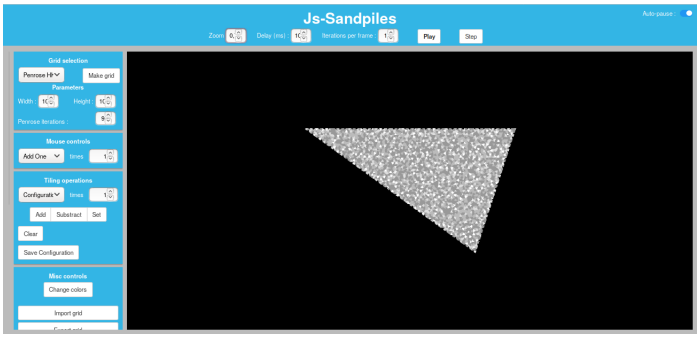


Figure 16: Application Sandpiles

L'application est disponible publiquement sur la page github (https://github.com/hehaOhyeah/JS-Sandpile). Elle est fonctionnelle et peut être utilisée à une fin éducative ou pour effectuer des mesures. Nous vous invitons à lire le manuel d'utilisation pour plus d'informations sur notre page github.

Références

[1] Shigeki Akiyama and Katsunobu Imai. The corona limit of penrose tilings is a regular decagon. In *International Workshop on Cellular Automata and Discrete Complex Systems*, pages 35–48. Springer, 2016.

[2] Per Bak, Chao Tang, and Kurt Wiesenfeld. Self-organized criticality : An explanation of the 1/f noise. *Physical review letters*, 59(4) :381, 1987.

[3] Deepak Dhar and Ramakrishna Ramaswamy. Exactly solved model of self-organized critical phenomena. *Physical Review Letters*, 63(16) :1659, 1989.

[4] Ronald Meester, Frank Redig, and Dmitri Znamenski. The abelian sandpile ; a mathematical introduction. *arXiv preprint cond-mat/0301481*, 2003.