# Tours de Hanoï et Pavage de Penrose

Guillaume Barbier, Romain Ferrand

3 octobre 2017

**Résumé**

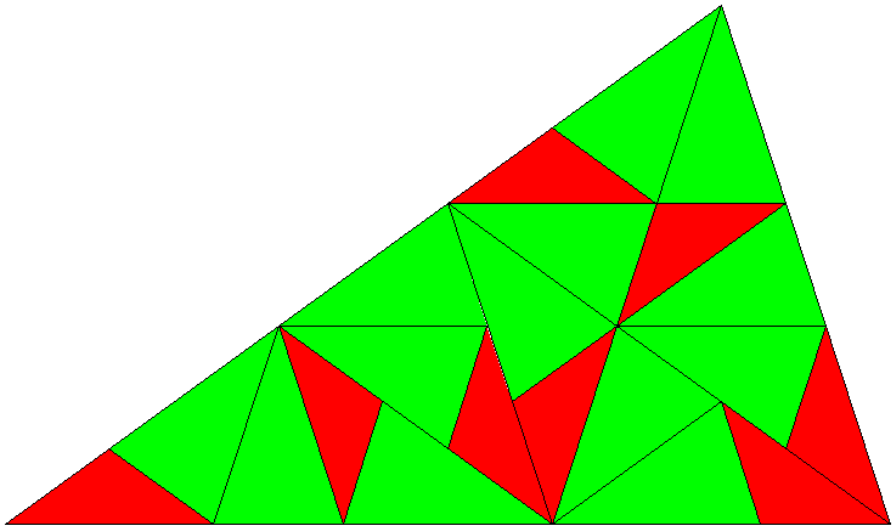

FIGURE 1 – Pavage de Penrose à trois générations

# Introduction

Dans ce document nous allons vous présenter notre étude du problème de Hanoï, ainsi que les différentes implémentations visant à sa résolution. Puis étudierons différents pavages de Penrose.

# 1 Pavage de Penrose

## 1.1 Présentation du problème

Le pavage de Penrose est un pavage du plan non périodique découvert par le mathématicien britannique Roger Penrose dans les années 1970. Notre but sera de découper un triangle d'or (dont les côtés ont un ratio égal au nombre d'or) en des triangles de même forme selon les règles suivantes, et de répéter cette action sur les triangles ainsi formés.

### 1.1.1 Première implémentation

On cherche à découper un triangle d'or un certain nombre de fois. Chaque découpage successif forme une nouvelle « génération »de triangles. Cette succession de génération forme un arbre dans lequel chaque nœud et feuille représente un triangle. Pour afficher la $n$-ième génération, il suffit de dessiner les nœuds de profondeur $n$.

L'algorithme le plus simple se résume ainsi :
— On parcourt l'arbre des triangles (dans un ordre quelconque).
— À chaque feuille, on dessine le triangle associé.

L'intérêt de cette méthode est qu'elle peut être rendue polymorphe. Il n'est pas nécessaire de parler de triangle, on peut parler de polygone, voire même de forme ou d'objet. Pour celà, il faut fournir à l'algorithme :

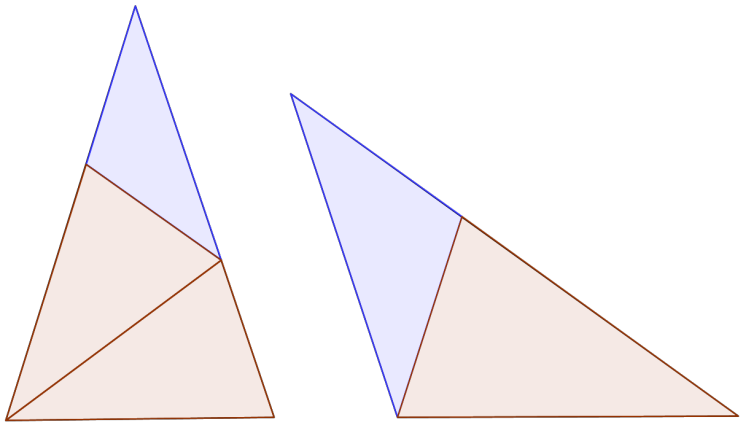

FIGURE 2 – Division des deux types de triangle

— un objet initial à découper qui sert de racine à l'arbre ;
— une fonction pour diviser un objet père en ses objets fils ;
— une fonction pour dessiner un objet (en particulier les feuilles).

On définit donc un type *triangle*, et les fonctions définies ci-dessus. Cependant, je préfère m'attarder sur l'algorithme lui même plutôt que sur la façon dont on affiche et divise les triangles.

## 1.2 Comment améliorer l'algorithme ?

### 1.2.1 Le dessin des contours

Nous dessinons les triangles avec leur contour, cependant le côté d'un triangle est souvent un côté du triangle voisin, ou une section d'un côté du triangle parent.

Comment alors modifier notre algorithme pour dessiner chaque trait une unique fois ?

Il faut changer de point de vue : un trait ne permet pas seulement de dessiner le contour d'un triangle, mais de séparer deux triangles. Ainsi nous allons pouvoir dessiner à chaque fois que l'on divise un triangle (c'est à dire à chaque nœud), pour séparer les triangles nouvellement formés.

Lorsque deux triangles sont côte à côte, ils seront alors séparés par un trait dessiné par le nœud qui est leur plus proche ancêtre commun. Cependant le contour du triangle racine n'est pas pris en compte par cette méthode, il faut donc penser à le dessiner.

L'algorithme devient alors :
— On dessine le contour de l'objet racine.
— On parcours l'arbre dans un ordre quelconque.
— À chaque nœud, on dessine les séparations entre les fils de ce nœud.

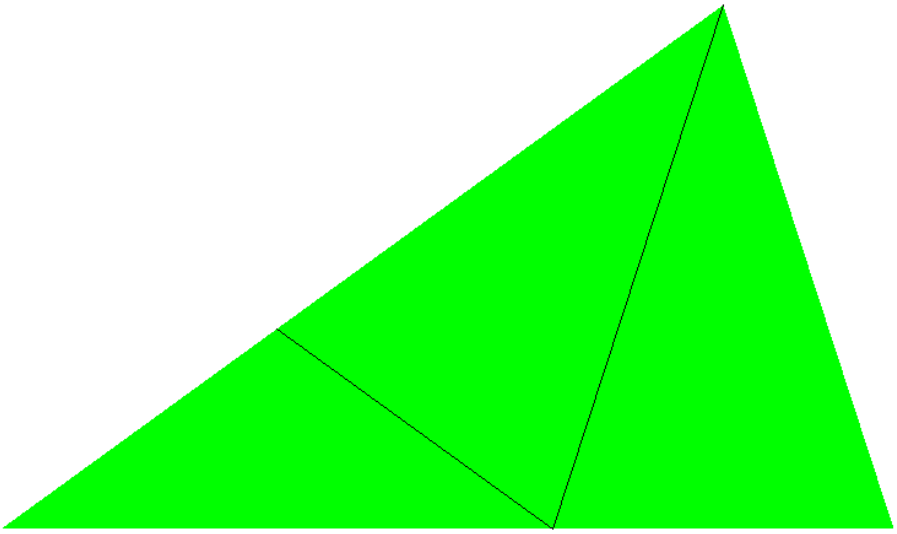

FIGURE 3 – Séparation des triangles fils

— À chaque feuille, on dessine l'objet associé.

L'algorithme a besoin de :
— un objet initial à découper qui sert de racine à l'arbre ;
— une fonction pour diviser un objet père en ses objets fils ;
— une fonction pour dessiner un objet (en particulier les feuilles) ;
— une fonction pour dessiner les séparations entre les fils d'un objet.

### 1.2.2 Le dessin des générations successives

On cherche maintenant à dessiner successivement plusieur générations. Chaque génération correspond à une hauteur particulière dans notre arbre, nous allons donc naturellement utiliser un parcours en largeur de l'arbre. Cependant, nous avons aussi besoin de nous arrêter avant chaque génération pour pouvoir effectuer le dessin.

Pour cela, la fonction récursive de parcours en largeur prendra en paramètre la liste des nœuds d'une certaine hauteur (la génération en cours de traitement) privé des nœuds déjà traités ; la liste des nœuds de la génération suivante, qui est vide au premier appel sur cette génération mais qui se remplie au fil des appels récursifs.

Cela n'est pas tout, il faut aussi prendre en compte le dessin des séparateurs. Dans l'algorithme précédent, nous nous contentions de dessiner les séparations au niveau de chaque nœud lors de l'unique parcours de l'arbre. Ici, lors du dessin de chaque génération, il faut reparcourir les générations précédentes pour dessiner les séparateurs au niveau de chaque nœuds. Comme la division des objets peut être couteuse (dans le cas des triangles, elle ne l'était pas), nous avons décidé de stocker les nœuds déjà parcourus dans une liste.

L'algorithme est schématiquement :

```
let rec parcours_largeur generation generation_suivante n =
  if n = 0 then
    ()
  else if est_vide generation then begin
    (*On a fini de parcourir cette generation *)
    dessiner_separateurs liste_des_noeuds_deja_visites;
    dessiner_generation generation_suivante;
    parcours_largeur generation_suivante vide (n-1)
  end else begin
    let noeud = extraire_noeud generation
    and suite = extraire_suite generation
    in
    let generation_suivante = concatenation (obtenir_fils noeud) generation_suivante
    in
    ajouter_noeud liste_des_noeuds_deja_visites noeud;
    parcours_largeur suite generation_suivante n
  end
;;
```