Cet article est rédigé par des élèves. Il peut comporter des oublis et imperfections,

autant que possible signalés par nos relecteurs dans les notes d'édition.

Le Perchiste

Année 2025 – 2026

Giovanni Prunes-Uruen, Guillaume Ledoux, Lenny Wiss et Oleksandr Khmelnytskyi

élèves de seconde

Établissement(s) : Lycée Marguerite De Navarre à Bourges

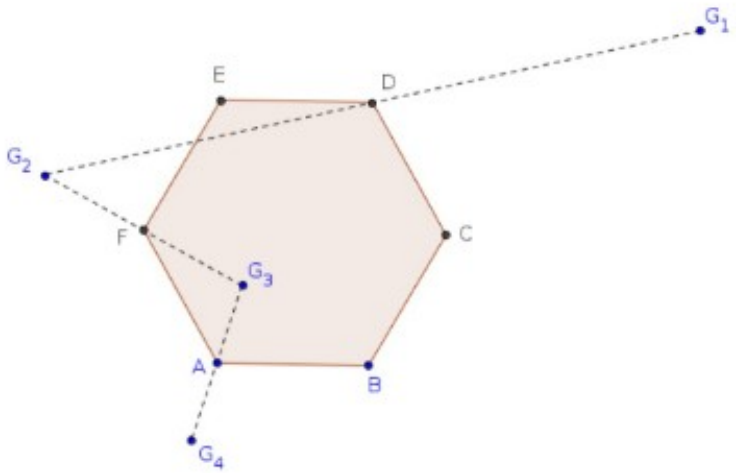
Enseignant·e(s) : Nathalie Herminier, Guillaume Pelletier

Chercheur·Chercheuse(s) : Xavier Bultel, Benjamin Nguyen

1.Introduction

1.1.Présentation du sujet

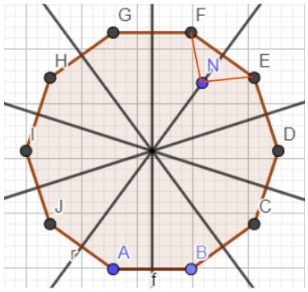
Un sauteur à la perche effectue un parcours dans le plan en commençant au point G_1



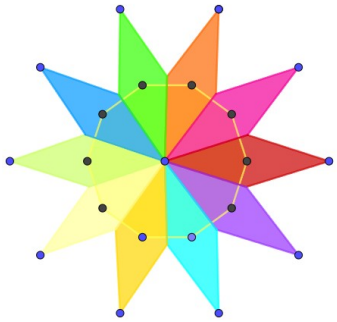
Pour cela, il doit utiliser un des butoirs disposés sur les sommets d'un polygone **régulier** (ici, dans notre exemple ABCDEF). Il doit toujours planter sa perche sur le sommet situé le plus proche de sa position (Ici D pour G_1).

Son point d'arrivée est le symétrique de la position précédente (ici G_1) par la symétrie centrale dont le centre est le sommet du polygone le plus proche (ici D). Puis il recommence ainsi de suite jusqu'au moment où le point d'arrivée est un point du plan déjà visité. (sinon, le procédé est répété indéfiniment)

Par convention, le perchiste s'arrête également quand il arrive sur un point situé sur l'une des médiatrices des segments reliant les sommets (autrement dit il est à *équidistance* de plusieurs sommets).



1.2. Nos résultats



Au cours de nos simulations, nous avons constaté que le perchiste finissait toujours par boucler entre deux points (mais nous ne sommes pas en mesure de le démontrer).(1)

Lors de nos travaux, nous n'avons pas pu démontrer que le parcours du perchiste se termine systématiquement. Par contre, nous avons déterminé la zone d'arrivée du perchiste à partir de son point de départ car nous connaissons la zone dans laquelle le perchiste est plus près d'un certain sommet et la zone dans laquelle le perchiste va tourner en boucle autour de ce sommet s'il y rentre.

Nous avons également pu étudier des configurations très variées de polygones réguliers et déterminer pour chaque point du plan sa zone d'arrivée par une représentation sous forme de pavage.

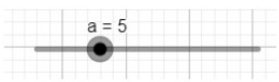
2. Simulation à l'aide de Geogebra

Afin d'éviter les erreurs ainsi que les pertes de temps, nous avons utilisé l'application Geogebra pour optimiser nos recherches.

Nous avons donc créé un polygone régulier pour pouvoir travailler dessus mais cela n'était pas suffisant pour vérifier nos idées, donc nous avons créé un polygone régulier modifiable à l'aide d'un curseur (c'est-à-dire qu'on peut changer le nombre de sommets du polygone immédiatement avec le curseur).

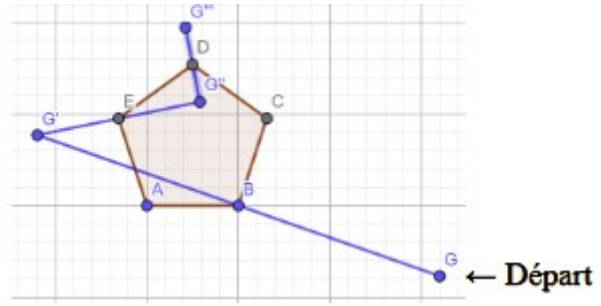






La lettre « a » est définie comme le nombre de sommets de notre figure. Dans le cas de la figure ci-dessus le nombre maximal est 10 sommets, ce qui permet d'obtenir un décagone. On prend également un pas de 1 pour éviter les nombres non entier et faciliter notre travail. Ainsi, nous pouvons réaliser tous les polygones réguliers ayant un nombre de côtés compris entre 3 et 10.

Ensuite on regarde les déplacements du perchiste grâce à des symétries centrales :



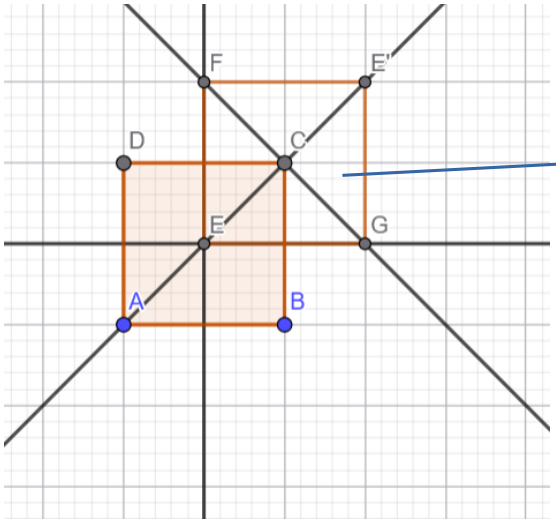
Le perchiste commence au point G. Son trajet ne se termine pas, il tourne indéfiniment sur le point D.

Après de nombreux tests, nous avons ainsi découvert que le perchiste finissait toujours par tourner autour d'un même sommet, indéfiniment.

3. Zone de bouclage

Nous nous sommes donc demandé dans quelles conditions ce cas apparaissait, et nous avons réussi à construire une zone autour de chacun des points du polygone régulier, dans laquelle le perchiste ne peut que tourner indéfiniment.

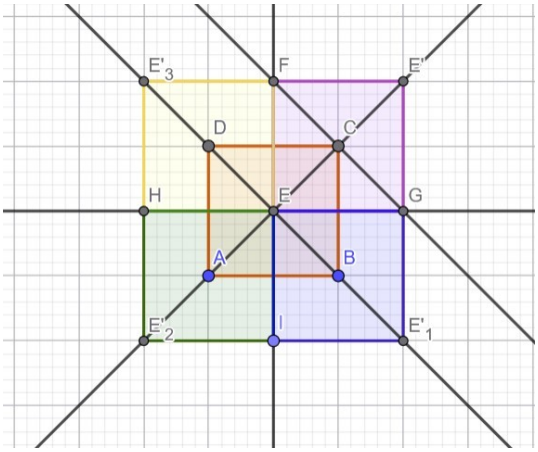
Voici un exemple d'une de ces zones avec le carré, autour du point C :



Pour construire cette zone, nous avons d'abord tracé les médiatrices des côtés du carré, ensuite une droite passant par le point E (centre du carré) et le point C, puis la perpendiculaire à cette droite, et enfin nous avons créé le point E', symétrique du point E par rapport à C.

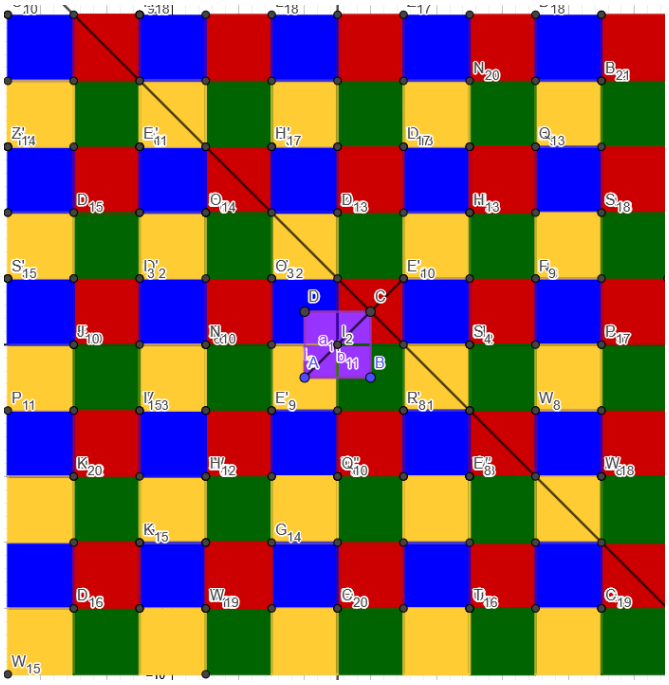
Avec l'aide de toutes ces droites et du point E', nous avons créé les points F et G, intersections des médiatrices et de la perpendiculaire à (EC). Pour finir nous avons créé le carré EGE'F, qui est cette zone dont le perchiste ne peut sortir s'il y rentre. Nous en ferons la démonstration un peu plus loin.

Si on crée une zone semblable pour chacun des points du polygone on obtient ceci :



4. Le pavage et la démonstration

Le pavage est une zone qui peut s'étendre à l'infini. Elle nous permet de prédire avec précision où le perchiste va arriver peu importe d'où il part. Voici un exemple de pavage avec le carré :



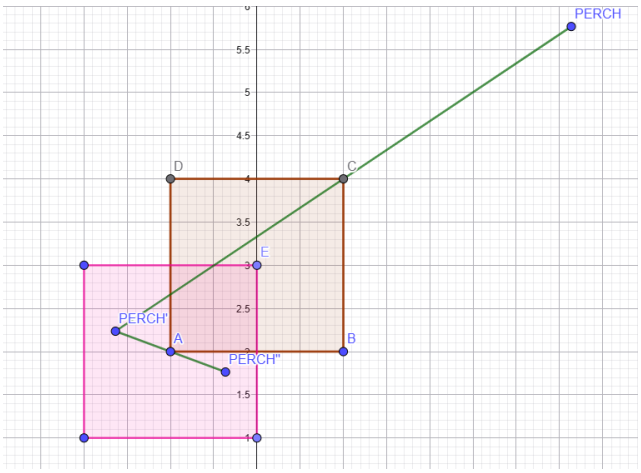
On peut observer ici le polygone régulier originel (le carré violet), et une zone de couleurs différentes pour chacun de ses points. (zone jaune pour le point A, verte pour le point B, rouge pour le point C et bleue pour le point D.) A partir de ces quatre zones, nous avons créé un pavage à la main sur Géogebra en faisant la symétrie centrale de toutes les zones par rapport aux sommets du carré violet. Puis, nous avons recommencé plusieurs fois avec les nouveaux carrés obtenus jusqu'à compléter une grande zone du plan autour du polygone. Cela nous a donné le pavage représenté ci-dessus.

Grâce à ce pavage, nous obtenons une information importante : Où est-ce que le perchiste terminera son parcours (dans la zone du A, du B, du C ou du D) selon son point de départ.

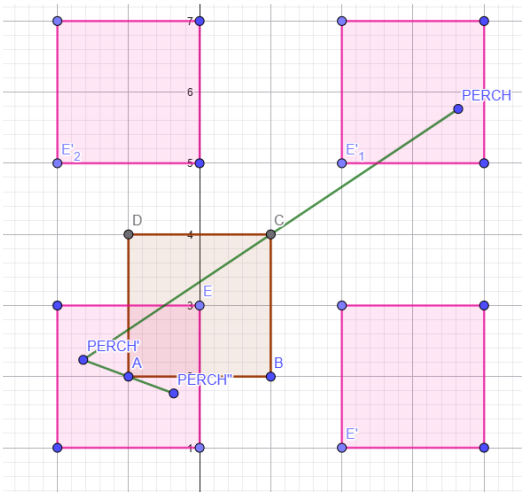
Pour cela, il nous suffit de regarder la couleur de la zone dans laquelle il commence, car elle correspond

à la couleur de la zone dans laquelle il terminera.

Après la création de ce pavage, nous pouvons nous demander pourquoi cela fonctionne ? Pourquoi ces zones fonctionnent-elles ? Et comment le prouver mathématiquement ?



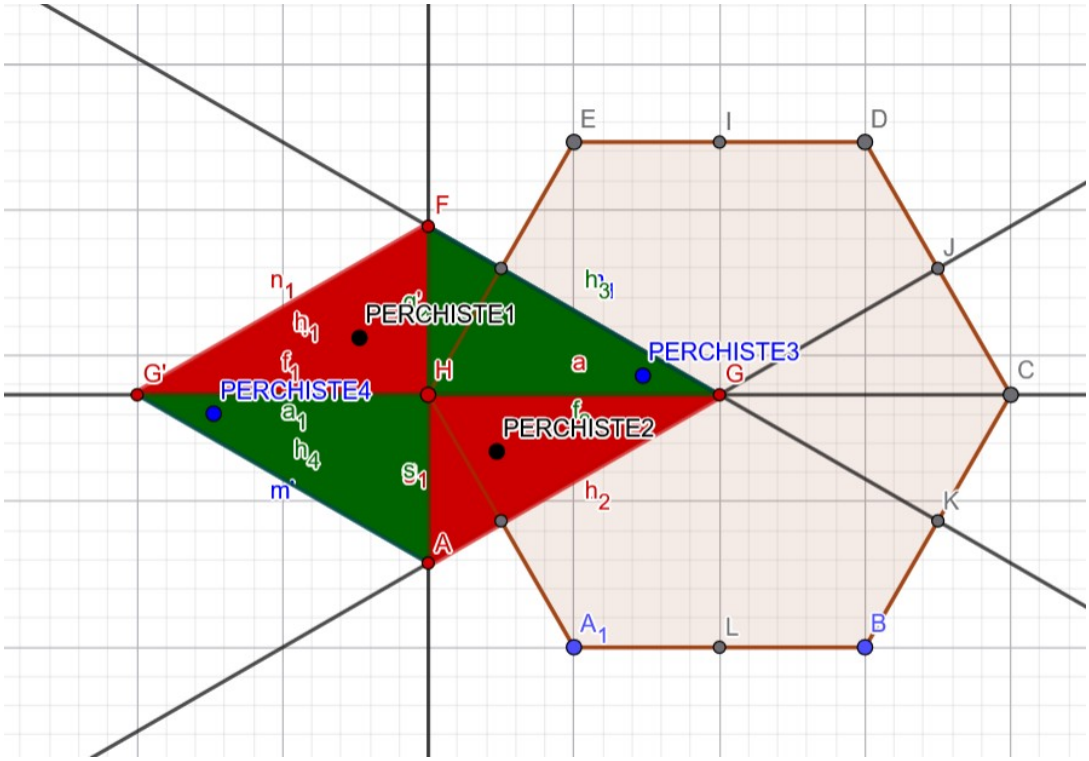
Sur ce parcours du perchiste nous pouvons voir que le perchiste termine dans la zone rose et boucle entre le point PERCH' et PERCH''.



Si on fait les symétriques de la zone rose par rapport aux points du polygone, c'est à dire le pavage de la zone rose, on remarque que le perchiste se trouvait au départ dans une zone rose, donc cela prouve l'efficacité du système de pavage.

<u>1ère partie de la preuve Mathématique :</u>

Comme nous pouvons le voir sur cette représentation d'un hexagone ci-dessous, ce losange vert et rouge est la seule zone dans laquelle le perchiste restera une fois qu'il y pénètre.



[FG] est le symétrique de [AG'] par rapport à H. Donc le triangle FHG'(rouge) est le symétrique du triangle HGA (rouge) par la symétrie centrale de centre H.

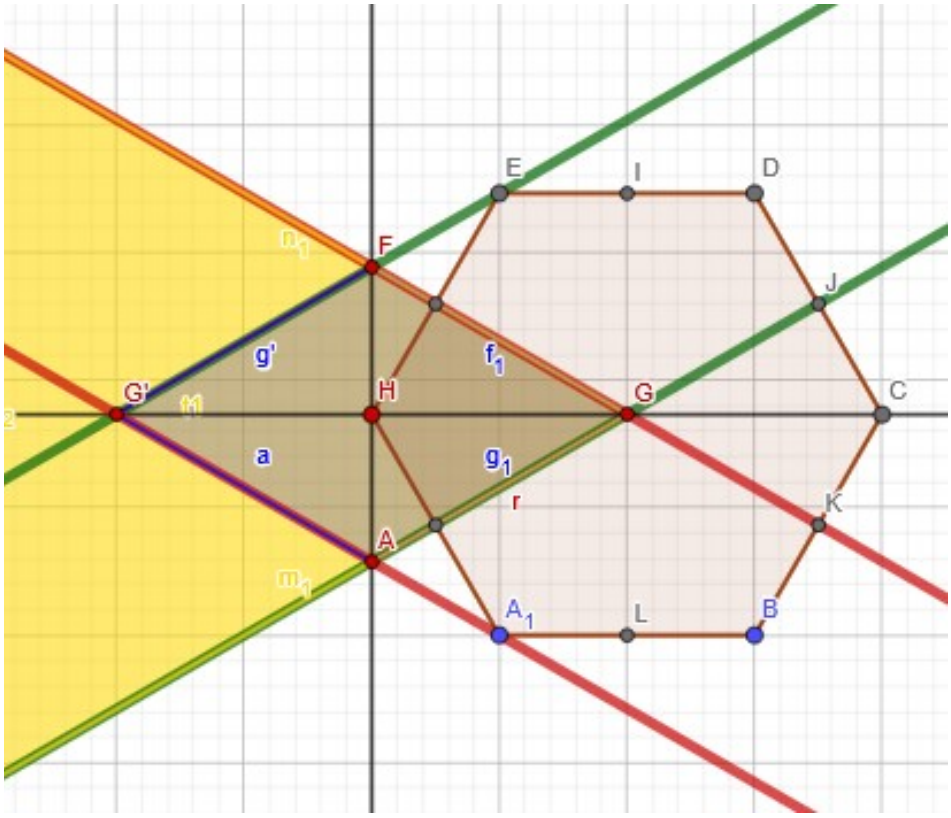
Tout point qui se trouve alors dans le triangle FHG' se retrouvera dans le triangle HGA par symétrie centrale. Si le perchiste se retrouve dans FHG', il se retrouvera dans HGA après un saut, et inversement.

C'est la même situation pour les triangles FGH(vert) et HAG'(vert). Donc si le perchiste se retrouve dans le losange vert et rouge, **il y restera peu importe le nombre de sauts qu'il effectuera.**

2ème partie de la preuve mathématique :

On considère ci-dessous la zone jaune délimitée par les deux médiatrices (FG) et (GA).

Tous les points qui se trouvent entre les deux médiatrices (zone jaune) sont plus proches du point H que des autres points de l'hexagone. Pourquoi ? Et comment cela nous aide-t-il à créer le losange dont le perchiste ne peut sortir ?



La médiatrice (GA) du côté [A_1H] sépare le plan en deux zones. Un côté où les points sont plus proches de A_1 et une autre zone où ils sont plus proches de H. De même pour la médiatrice (GF) du côté [HE]. Ainsi, les points dans la zone jaune sont plus proches du point H que de tout autre sommet du polygone(2). Ils feront donc leur prochain saut avec le point H.

Le symétrique par H de la médiatrice (GA) est la droite (G'F).

Le symétrique par H de la médiatrice (GF) est la droite (G'A).

Donc si le perchiste se trouve dans la zone jaune, mais pas dans FGAG' (le losange dont les côtés sont les médiatrices et leur symétriques) et qu'il saute, alors il atterrira en dehors de la zone jaune, donc il sera plus proche d'un autre sommet que de H. Il ne pourra donc pas boucler sur le point H.

Conclusion :

Ainsi les zones en forme de losange construites autour des sommets du polygones sont bien les seules zones où l'on peut boucler. Nous les avons réalisés sur un dodécagone et nous avons appelé cette propriété le « théorème du Tournesol ».

5. Le programme PYTHON

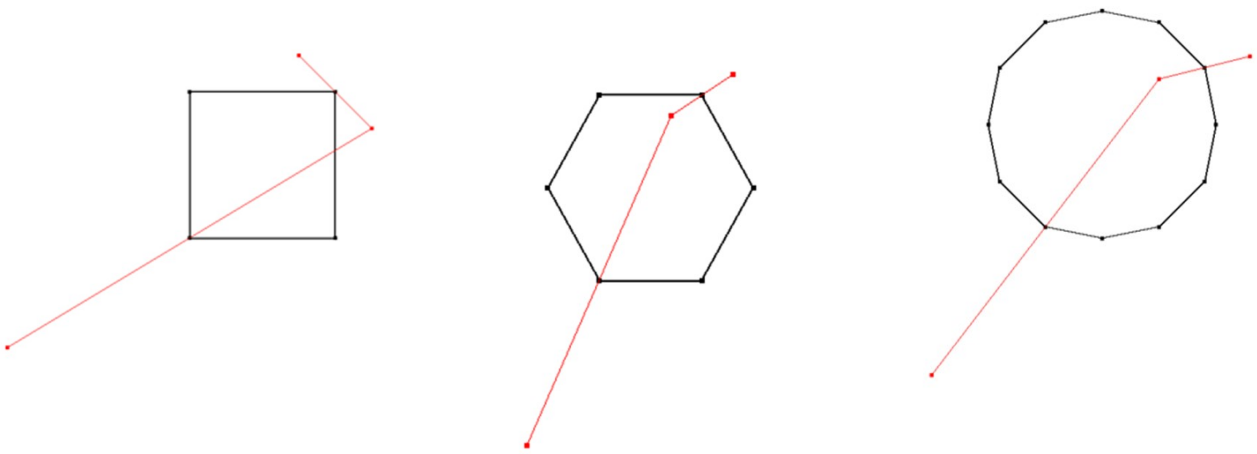
Au cours de notre recherche, nous avons créé plusieurs programmes qui nous ont aidés à vérifier nos idées et à analyser des figures plus complexes. Vous pouvez trouver les programmes en annexe.

Le premier programme, à partir des coordonnées du polygone et de la position initiale du perchiste, calcule chacune de ses positions successives.

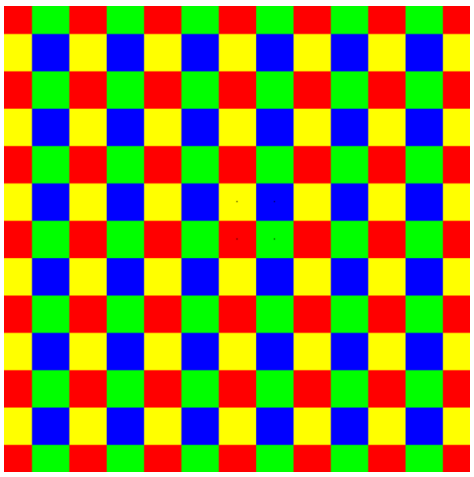
```
Start x: -5
Start y: -3
Nombre points polygone : 4
x: 0
y: 0
x: 4
y: 0
x: 4
y: 4
x: 0
y: 4
[[-5, -3], [5, 3], [3, 5]]
```

```
Start x: -12
Start y: -25
Nombre points polygone : 6
x: 10
y: 0
x: 5
y: 9
x: -5
y: 9
x: -10
y: 0
x: -5
y: -9
x: 5
y: -9
[[-12, -25], [2, 7], [8, 11]]
```

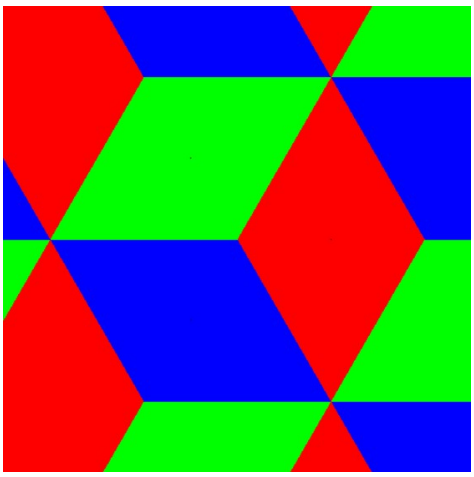
Et voici un autre programme, qui utilise les coordonnées du polygone et la bibliothèque Python PIL, pour dessiner le polygone et visualiser le mouvement du perchiste.



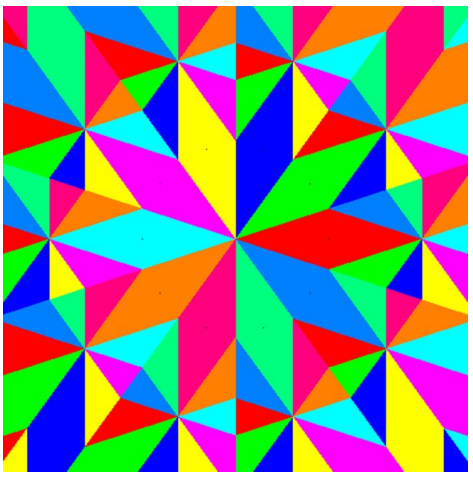
Le dernier programme, également basé sur les coordonnées du polygone et la bibliothèque PIL, construit les zones pour n'importe quel polygone, même irrégulier. Nous attribuons une couleur à chaque sommet, puis pour chaque pixel de l'image de taille choisie, nous lançons le mouvement du perchiste et colorons le pixel initial avec la couleur du sommet qui est le plus proche **à la fin du mouvement, c'est-à-dire lorsque nous bouclons.**



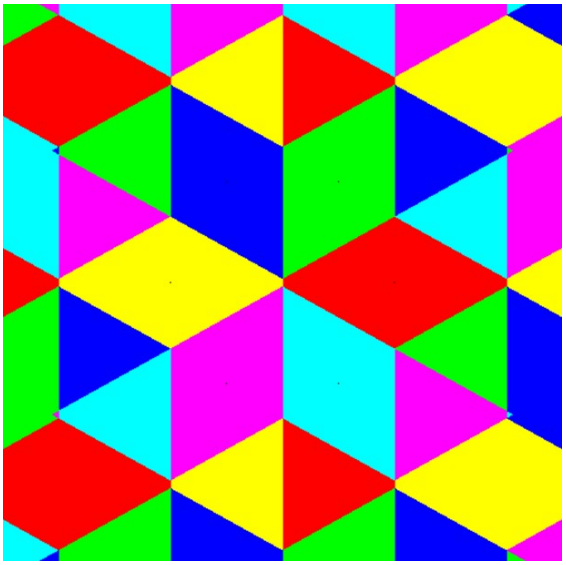
Carré



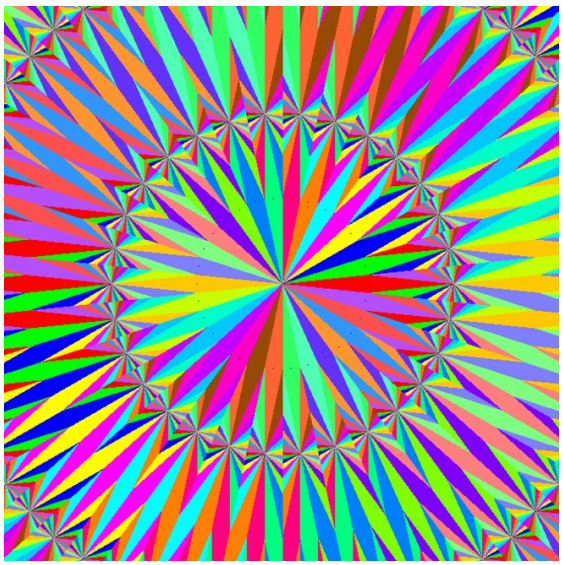
Triangle



Décagone(10 sommets)



Hexagone(6 sommets)

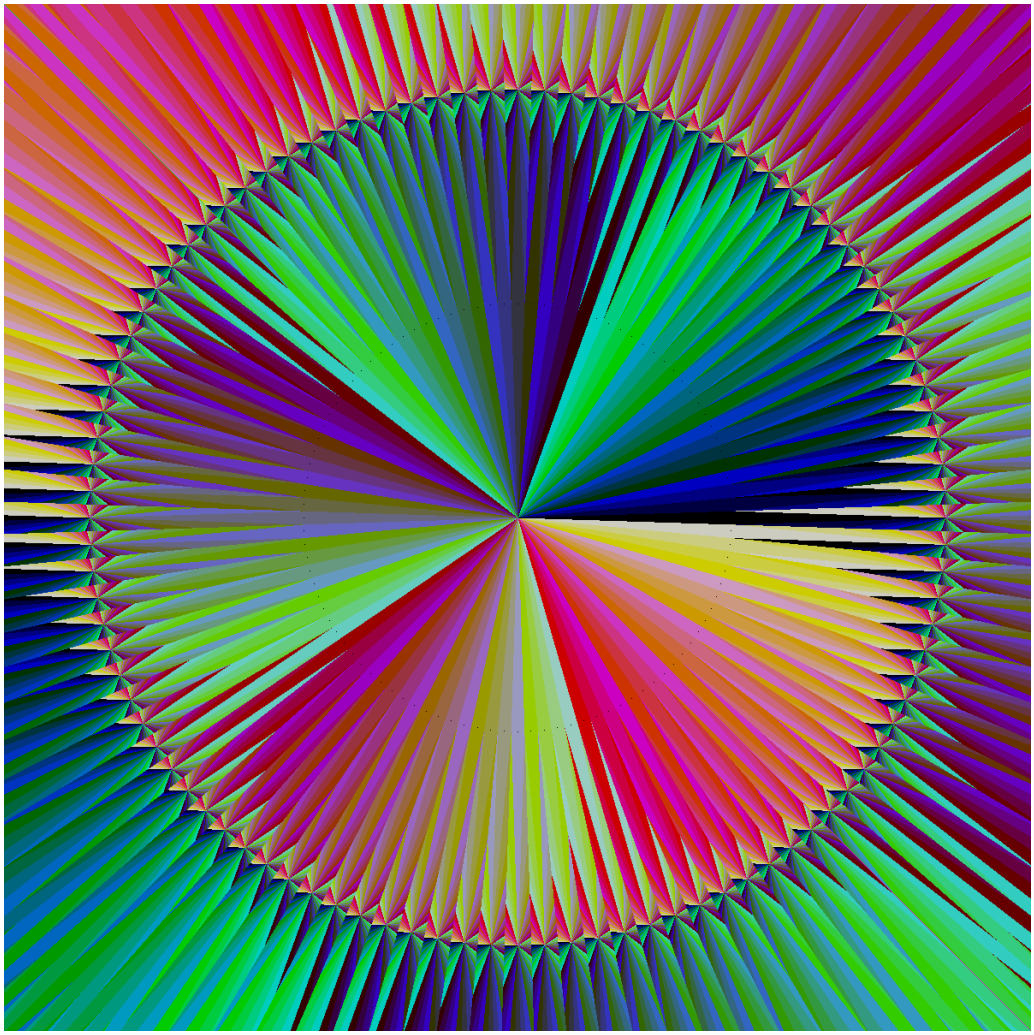


Triacontagone(30 sommets)

Cependant, une question importante se pose : si deux sommets se trouvent à la même distance, quelle couleur doit être attribuée au pixel ?

C'est ici qu'apparaît une des imprécisions du programme. Nous avons essayé d'éviter ce cas en ajoutant une très petite valeur (bruit) aux coordonnées des sommets du polygone. Cependant, cela peut produire des résultats inattendus en raison des erreurs de calcul (avec des nombres à virgule flottante) et de leur accumulation au cours d'un grand nombre d'itérations du mouvement. Ainsi, le résultat peut varier selon les appareils ou ne pas toujours se manifester.(3)

Le plus grand polygone pour lequel nous avons réussi à générer les zones est un hécatontagone (polygone à 100 côtés) avec une image de 1200×1200 pixels. La génération de cette image a pris 12 minutes.

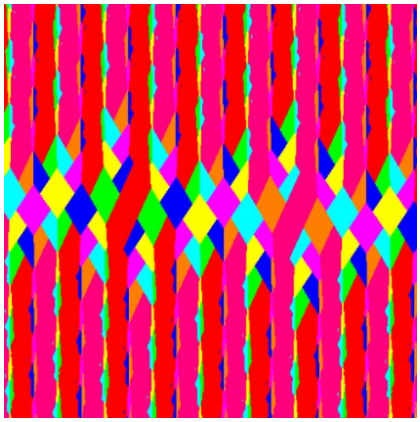


En théorie, le programme n'est presque pas limité en taille de polygone. Les principales contraintes sont le nombre de couleurs disponibles, le choix correct des coordonnées des sommets (qui peuvent être calculées à l'aide de formules mathématiques), le plan discret et une mise à l'échelle correcte de l'image.

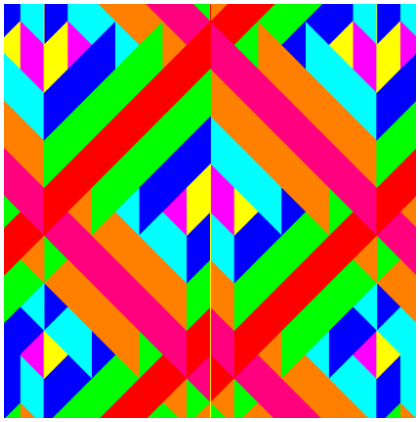
Au fil du temps, nous nous sommes intéressés aux zones des polygones irréguliers et nous avons eu l`idée de placer les sommets du polygone sur différentes fonctions mathématiques.

L`idée est très simple, nous disposons nos sommets régulièrement sur un certain intervalle de la courbe et nous lançons le programme de dessin de zones.

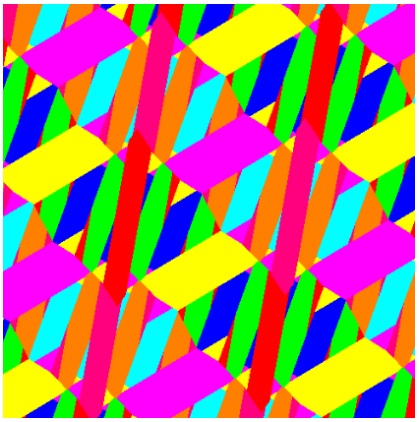
Exemples de polygones irréguliers :(4)



y = sin(x)



y = |x|



y = tan(x)

Notes d’édition

(1) C’est ce que l’on appelle une conjecture. Un énoncé mathématique dont on pense qu’il est vrai mais sans être (encore) capable de le démontrer.

(2) Ces zones sont assez étudiées en mathématiques. Elles portent le nom de « cellules de Voronoï ».

(3) Il aurait été bien de détailler un peu plus cette partie afin d’expliquer clairement comment cette ambiguïté de la définition a été contournée informatiquement.

(4) Un schéma représentant les « polygones » choisis sur ces courbes aiderait à la compréhension.