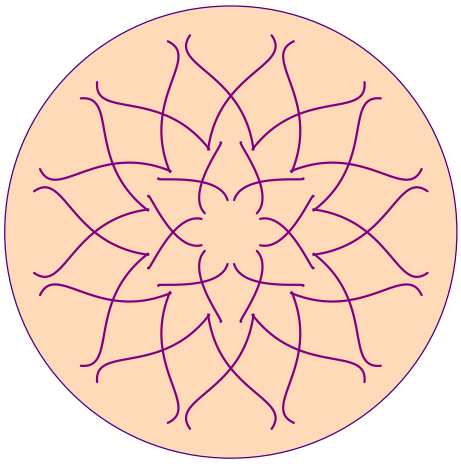
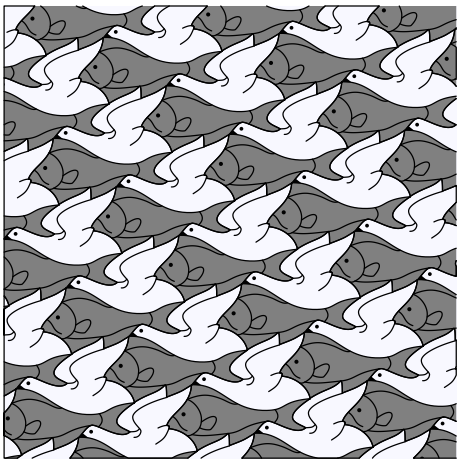
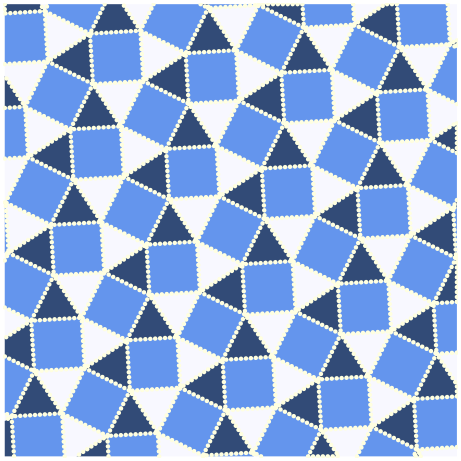
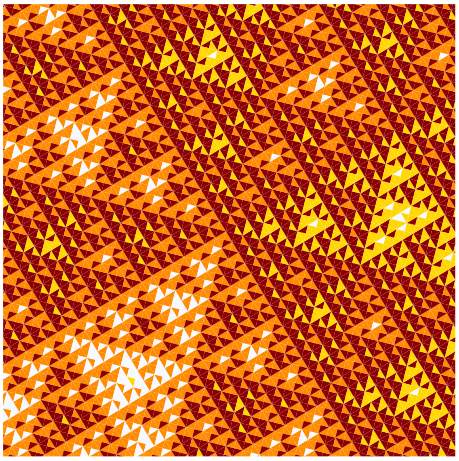
Cristallo.mac









Alphonse Capriani

3 octobre 2010

Table des matières

Liste des tableaux

Avant-propos

`Cristallo.mac` ?

Le fichier `Cristallo.mac` est un fichier de macros permettant de réaliser facilement des frises, des rosaces ou des pavages avec *TEXgraph*. La version présentée ici, nommée 10.10, est une refonte totale de la version créée en 2007. Les différences entre ces deux versions résident principalement sur la manière de créer de tels graphiques ; dans la précédente version, les frises ou pavages étaient créés par le biais de boutons placés dans la colonne de gauche de l'interface graphique de *TEXgraph* alors que l'on est obligé de taper un peu de code avec la version actuelle. Parmi les évolutions depuis la dernière version, on notera aussi un changement au niveau de la manière de créer les frises, les rosaces et les pavages périodiques (voir le paragraphe suivant) et « quelques » nouveautés parmi les pavages apériodiques. Malgré ces différences, les deux versions du fichier `Cristallo.mac` permettent la création des même types de graphiques :

- ⋆ des frises périodiques,
- ⋆ des rosaces,
- ⋆ des pavages périodiques,
- ⋆ des pavages par des polygones réguliers,
- ⋆ des pavages apériodiques.

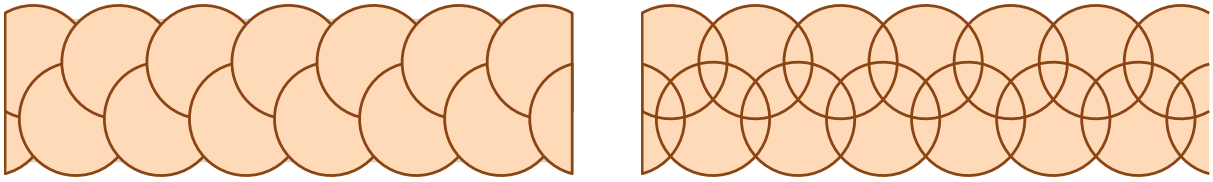
Chacun de ces types de graphiques feront l'objet d'un chapitre de ce document.

Les nouveautés...

Comme on vient de le voir, la version 10.10 de `Cristallo.mac` se différencie nettement de l'ancienne version par son utilisation. L'ancienne version permettait la création de frises, rosaces ou pavages par « clics-boutons ». Ainsi, le choix du type de frise ou pavage, les attributs des motifs ou autres objets du graphiques, ... étaient modifiés par le biais de boîtes de dialogue qui s'ouvraient quand l'utilisateur cliquait sur le bouton voulu. Dans cette nouvelle version, les choses sont radicalement différentes. Plus aucun bouton n'est mis à disposition : l'utilisateur doit lui même créer le code de son graphique. Pour cela, quelques macros graphiques sont mises à disposition pour dessiner les frises, rosaces ou pavages évoqués précédemment. Ces macros sont très simple d'utilisation : elles comportent deux ou trois arguments au maximum. Quant à la modification des attributs des différents objets constituant un tel graphique, ils sont modifiables par le biais d'un certain nombres de variables globales aux noms intuitifs qui seront énumérées au fil des chapitres de ce manuel.

La deuxième innovation consiste en une réécriture complète des macros de construction des frises, rosaces et pavages périodique afin de régler un problème majeur de l'ancienne version. En effet, avec la précédente version, si le motif de base d'une frise (ou d'une rosace ou d'un pavage périodique) était une ligne polygonale avec un remplissage de type `full`, alors on pouvait observer des superpositions indésirables des motifs provoquant la perte des propriétés de base de la frise voulue (resp. de la rosace ou du pavage). Cet inconvénient est illustré sur la figure 1a. Les macros du nouveau fichier sont conçues pour contourner ce problème. Si le remplissage du motif est de type `full`, alors celles-ci vont procéder en deux temps : d'abord, on dessine tous les motifs sans leur bord (c'est à dire avec `LineStyle` valant `noline`, seul le remplissage est dessiné), ensuite on ajoute les bords en redessinant les motifs sans remplissage (avec `FillStyle` valant `none`). La figure 1b nous montre alors ce que l'on obtient avec cette procédure. On note que cette fois-ci, les propriétés de la frise sont bien conservées (on a notamment

bien invariance de la frise par symétrie glissée ce qui n'est pas le cas de la figure 1a). Malheureusement, cette nouvelle technique à des inconvénient par rapport à la précédente. En particulier, la première admettait comme motif de base n'importe quel dessin mélangeant éventuellement lignes, courbes ou points. Dans la version 10.10, on devra se contenter de lignes polygonales.



(a) Ancienne version de `Cristallo.mac` (b) Version 10.10 de `Cristallo.mac`

FIGURE 1 – Chevauchement des motifs d'une frise (cas d'une frise de type f1g)

Enfin, l'autre grande nouveauté du fichier réside dans l'éventail de pavages apériodiques maintenant disponibles. La version précédente permettait la création de trois types de pavages apériodiques : le pavage de Penrose avec fléchettes et cerfs-volant, avec losanges fins et losanges épais et le pavage de Penrose avec triangles d'or [1]. Désormais, le nombre de pavages apériodiques que l'on peut créer avec `Cristallo.mac` est multiplié par 30 ! Au total, plus de 90 pavages différents facilement réalisables sont à disposition dans cette mise à jour.

A propos de ce document...

Ce document est un « manuel » permettant la bonne utilisation du fichier `Cristallo.mac` et de ses macros. Toutes les fonctions disponibles dans le fichier sont décrites avec leur syntaxe et toutes les variables globales permettant de modifier l'apparence d'une frise, d'une rosace ou d'un pavage sont énumérées avec leur valeur par défaut et leur utilité. Chaque chapitre correspond à un type de graphique réalisable avec `Cristallo.mac` :

Chap. 1 : Frises périodiques

Chap. 2 : Rosaces

Chap. 3 : Pavages périodiques du plan euclidien

Chap. 4 : Pavages du plan euclidien par des polygones réguliers

Chap. 5 : Pavages apériodiques

Ces chapitres sont indépendants. Ainsi, il n'est pas nécessaire de lire l'intégralité du document pour dessiner des pavages apériodiques. Bien évidemment, comme beaucoup de variables globales servent pour plusieurs types de graphiques, le lecteur observera probablement de nombreuses redondances. Chacun des chapitres se termine par une série d'exemples permettant à l'utilisateur de se familiariser avec la syntaxe des différentes macros mises en jeu. Une dernière section composée d'un tableau récapitulatif de toutes la variables globales intervenant dans la création de ces graphiques évitera à l'utilisateur de devoir reparcourir le chapitre pour trouver la variable globale dont il a besoin.

Remerciements

Je tiens enfin à remercier Patrick Fradin pour nous avoir offert ce merveilleux logiciel qu'est *TEXgraph* ainsi que pour l'aide qu'il a pu m'apporter durant la mise à jour du fichier `Cristallo.mac`. La rapidité et la qualité de ses réponses sur le forum [2] dédié au logiciel m'ont beaucoup aidé pour venir à bout de ce projet. Je remercie aussi d'avance tous les utilisateurs qui voudront bien me faire parvenir leur remarques, suggestions ou critiques au sujet de `Cristallo.mac`.

1. Dans la version actuelle, ce pavage correspondra à celui nommé Tuebingen triangle
2. Le forum de *TEXgraph* est disponible à l'adresse http://texgraph.tuxfamily.org/forum/

Chapitre 1

Frises périodiques

1.1 Frises périodiques

Le fichier `Cristallo.mac` permet de créer rapidement et simplement des frises périodiques. Les frises périodiques sont analogues aux pavages périodiques (cf. chapitre 3) mais sont engendrées par un seul vecteur de translation. Ainsi, étant donné un vecteur $\vec{u}$ du plan, une frise périodique généré par $\vec{u}$ est la répétition d'un motif, ayant éventuellement subit diverses transformations, par translations de vecteur $k\vec{u}$ où k parcourt l'ensemble $\mathbb{Z}$ des entiers relatifs.
Il existe sept types de frises différents qui se différencient par le contenu de leur groupe de symétrie. Le tableau 1.1 énumère les différents types de frises et donne le contenu de leur groupe de symétrie. Le point Ω est un point de référence permettant de positionner convenablement les centres et axes de symétrie et le vecteur $\vec{n}$ est un vecteur orthogonal à la droite $(\Omega, \vec{u})$.

Type	**Éléments du groupe de symétrie**
f1	Translations de vecteurs $k\vec{u}$
f2	Symétries de centres $\Omega + \frac{k}{2}\vec{u}$ Translations de vecteurs $k\vec{u}$
f1m	Réflexion d'axe $(\Omega, \vec{u})$ Translations de vecteurs $k\vec{u}$
fm1	Réflexions d'axes $\left(\Omega + \frac{k}{2}\vec{u}, \vec{n}\right)$ Translations de vecteurs $k\vec{u}$
f1g	Glissement d'axe $(\Omega, \vec{u})$ et de vecteur $\frac{1}{2}\vec{u}$ Translations de vecteurs $k\vec{u}$
f2m	Réflexion d'axe $(\Omega, \vec{u})$ Réflexions d'axes $\left(\Omega + \frac{k}{2}\vec{u}, \vec{n}\right)$ Symétries de centres $\Omega + \frac{k}{2}\vec{u}$ Translations de vecteurs $k\vec{u}$
fm2	Glissement d'axe $(\Omega, \vec{u})$ et de vecteur $\frac{1}{2}\vec{u}$ Réflexions d'axes $\left(\Omega + \frac{k}{2}\vec{u}, \vec{n}\right)$ Symétries de centres $\Omega + \frac{2k+1}{4}\vec{u}$ Translations de vecteurs $k\vec{u}$

TABLE 1.1 – Groupes de symétries des différents types de frises

1.2 Création de frises – Macro `DrawFrise`

Les sept types de frises que l'on vient de voir peuvent facilement être créer grâce aux macros et variables globales du fichier `Cristallo.mac`. La fonction permettant le dessin d'une frise périodique se nomme `DrawFrise` et possède la syntaxe suivante :

```
DrawFrise(<Type>, [Options])
```

Le premier paramètre qui est obligatoire est une chaine de caractères définissant le type de frise que l'on souhaite créer. Les valeurs possibles de cet argument sont : `"f1"`, `"f2"`, `"f1m"`, `"fm1"`, `"f1g"`, `"f2m"` et `"fm2"`.
Le second argument, optionnel, est une liste d'affectations des variables globales du fichier `Cristallo.mac`.

Ces affectations permettront de modifier l'apparence de la frises que l'on souhaite réaliser. Ces variables globales seront énumérées et décrites dans la section suivante. Il est important de noter que ces affectations peuvent se faire hors de la fonction. Lorsque beaucoup de variables globales sont à modifier, il est peut être plus judicieux de réaliser ces affectations avant d'appeler `DrawFrise` pour obtenir un code plus clair.

1.3 Variables globales

Nous allons dans cette section énumérer l'ensemble des variables globale du fichier `Cristallo.mac` ayant une incidence sur l'apparence d'une frise périodique. Ces variables permettent pour la plupart de modifier les attributs de lignes, de remplissage ou de points. D'autres permettent de décider de l'affichage de certains objets (bords, axes, centres de rotations, ...)

1.3.1 Point de référence et vecteur générateur

On a vu dans la section 1.1 que les frises était une répétition par translation de vecteur $k\vec{u}$, $k \in \mathbb{Z}$, d'un même motif. Ainsi, l'utilisateur doit définir le vecteur générateur $\vec{u}$ de la frise qu'il veut réaliser. Pour cela, la variable globale `Vecteur1` contient l'affixe de ce vecteur générateur $\vec{u}$. Par défaut, `Vecteur1` vaut 1 ce qui permet d'obtenir une frise horizontale.
Suivant le type de frise, l'utilisateur sera également amené à définir l'affixe d'un point de référence que l'on a noté Ω dans le tableau 1.1. Ce point de référence permet de positionner les centres et les axes de symétrie de certaines frises. La modification de ce point de référence se fait par le biais de la variable globale `PtRef` qui, par défaut, vaut 0.

L'utilisateur de `Cristallo.mac` peut être intéressé par l'affichage de ce vecteur générateur ainsi que du point de référence. Pour cela, il existe quelques variables globales supplémentaire permettant de gérer ces affichages. Les booléens `AfficherVecteur1` et `AfficherPtref` permettent de décider de l'affichage de ces deux objets : une valeur de 0 (valeur par défaut) signifiera que l'objet en question ne devra pas être affiché alors que la valeur de 1 imposera un tel affichage.
Si ces affichages sont désirés, il est possible de modifier les attributs de ces objets via quelques variables supplémentaires énumérées dans le tableau 1.2.

`Vecteur1`	`1`	Vecteur générateur de la frise périodique
`PtRef`	`0`	Point de référence de la frise périodique
`AfficherVecteur1`	`0`	Booléen permettant l'affichage du vecteur générateur
`VecteurLineStyle1`	`solid`	Style de trait du vecteur générateur
`VecteurWidth1`	`thicklines`	Épaisseur de trait du vecteur générateur
`VecteurColor1`	`dimgray`	Couleur de trait du vecteur générateur
`VecteurStrokeOpacity1`	`1`	Opacité du trait du vecteur générateur
`AfficherPtRef`	`0`	Booléen permettant l'affichage du point de référence
`PtRefDotStyle`	`dot`	Style du point de référence
`PtRefDotSize`	`2+2*i`	Taille du point de référence
`PtRefDotScale`	`1+i`	Facteurs d'échelle du points de référence
`PtRefDotAngle`	`0`	Angle de rotation du point de référence
`PtRefColor`	`black`	Couleur du point de référence
`PtRefFillColor`	`white`	Couleur de remplissage du point de référence

TABLE 1.2 – Variables globales relatives au vecteur générateur et au point de référence de la frise

1.3.2 Motif de base

On a vu qu'une frise était la répétition d'un motif de base qui subissait diverses transformations du plan (symétries, glissements, translations, ...). Ainsi, pour créer une frise, on doit d'abord définir notre motif de base. Un motif de base sera une une liste de complexes définissant une ligne polygonale. Cette

ligne polygonale est stockée dans la variable globale `Motif`. Un motif peut être découpé en plusieurs composantes séparées par la constante `jump`. Une fois encore, les attributs de ce motifs peuvent être modifiés par le biais de quelques variables listées dans le tableau 1.3.

`Motif`	`[...]`	Motif de base de la frise
`MotifFerme`	`0`	Booléen indiquant si le motif est une ligne fermé ou non
`MotifArrondi`	`0`	Arrondi des angles du motif
`MotifLineStyle`	`solid`	Style de trait du motif
`MotifWidth`	`thicklines`	Épaisseur de trait du motif
`MotifColor`	`darkred`	Couleur de trait du motif
`MotifStrokeOpacity`	`1`	Opacité de trait du motif
`MotifFillStyle`	`full`	Style de remplissage du motif
`MotifFillColor`	`lightgray`	Couleur de remplissage du motif
`MotifFillOpacity`	`1`	Opacité du remplissage du motif

TABLE 1.3 – Variables globales relatives aux attributs du motif de base de la frise

Il est important de noter que le motif peut chevaucher son image par une des transformations du groupe de symétrie de la frise en cours de création. On pourrait ainsi penser, si le style de remplissage choisi est `full`, que la superposition cache partiellement le motif précédent dessiné. En fait, la macro `DrawFrise` dessine la frise en deux temps : d'abord, elle dessine les motifs uniquement avec leur remplissage (avec `LineStyle` valant `noline`) puis, dans un second temps elle dessine les lignes du motifs sans remplissage (c'est à dire avec `FillStyle` égal à `none`). Ainsi, si des motifs se chevauchent, aucun d'entre eux ne seront masqués.

On peut également mettre en évidence le motif de base de la frise construite en encadrant celui ci. L'affichage de ce cadre est déterminé par les variables globales énumérées dans le tableau 1.4.

`EncadrerMotif`	`0`	Booléen permettant l'encadrement du motif de base
`CadreLineStyle`	`solid`	Style de trait de l'encadrement du motif
`CadreWidth`	`thicklines`	Épaisseur de trait de l'encadrement du motif
`CadreColor`	`gray`	Couleur de trait de l'encadrement du motif
`CadreStrokeOpacity`	`0.75`	Opacité du trait de l'encadrement du motif
`CadreFillStyle`	`none`	Style de remplissage de l'encadrement du motif
`CadreFillColor`	`lightgray`	Couleur de remplissage de l'encadrement du motif
`CadreFillOpacity`	`0.5`	Opacité du remplissage de l'encadrement du motif

TABLE 1.4 – Variables globales relatives à l'encadrement du motif de base d'une frise

1.3.3 Arrière-plan de la frise

Les caractéristiques de l'arrière-plan de la frise peuvent également être modifiées à l'aide de variables globales. Celles-ci sont présentées dans le tableau 1.5.

`BordEcart`	`0.125`	Écart entre le bord de la frise et les motifs
`BordLineStyle`	`solid`	Style de trait du bord de la frise
`BordWidth`	`thicklines`	Épaisseur de trait du bord de la frise
`BordColor`	`black`	Couleur de trait du bord de la frise
`BordStrokeOpacity`	`1`	Opacité de trait du bord de la frise
`BackgroundFillStyle`	`full`	Style de remplissage de l'arrière-plan de la frise
`BackgroundFillColor`	`lightpink`	Couleur de remplissage de l'arrière-plan de la frise
`BackgroundFillOpacity`	`1`	Opacité du remplissage de l'arrière-plan de la frise

TABLE 1.5 – Variables globales relatives aux attributs de l'arrière-plan d'une frise

Ces variables sont des variables classiques de modifications d'attributs (style, couleur, opacité de ligne ou de remplissage). Les bords de la frise sont deux droite permettant de borner la frise en cours de création. L'arrière-plan de la frise et alors le fond de la frise délimité par ces deux droites.
La variable `BordEcart` possède une action différente. Celle-ci permet de définir l'écart entre le bord de la frise et les motifs la composant. Cet écart est exprimé en pourcentage de la largeur de la frise (par largeur, on entend la largeur minimale de la bande contenant tous les motifs de la frise). La figure 1.1 donne une illustration des valeurs possible de ce paramètre `BordEcart` (d représente la largeur de la frise sans tenir compte des bord et ε désigne la valeur de la variable `BordEcart`). Ainsi, on remarque qu'une valeur nulle de `BordEcart` revient à ne pas définir d'écart entre la frise et son bord. Une valeur positive de `BordEcart` permet d'ajouter un espace vide entre la frise et son bord alors qu'une valeur négative induit un débordement des motifs à l'extérieur de la bande d'arrière-plan.

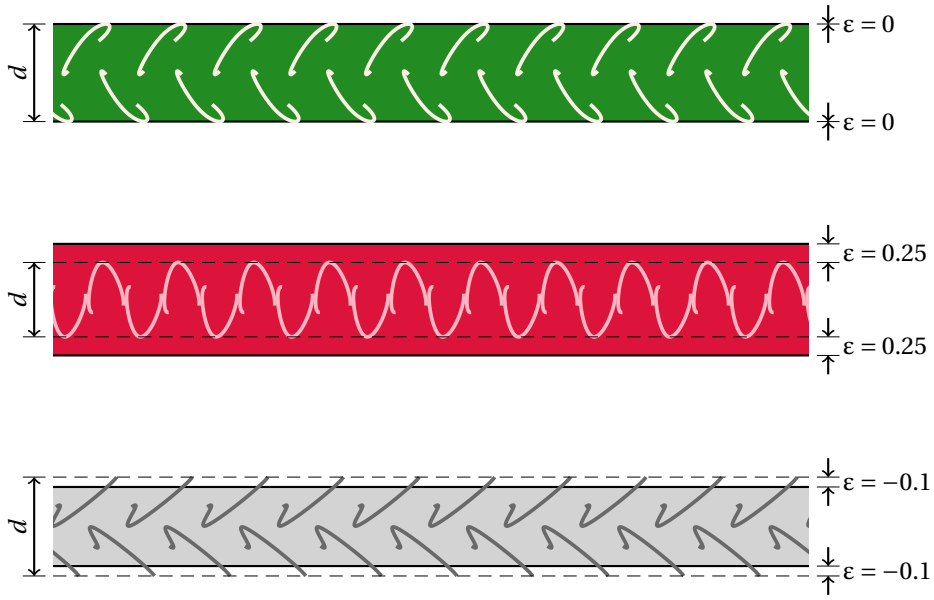


FIGURE 1.1 – Écart entre une frise et son bord – Définition de la variable `BordEcart`

1.3.4 Affichage des axes de symétrie et de glissement et des centres de symétrie

Le fichier `Cristallo.mac` permet pour une frise donnée d'afficher les éléments caractéristiques (axes et centre) des transformations figurant dans son groupe de symétrie. Pour cela, plusieurs variables globales répertoriées dans la table 1.6 sont modifiables afin de permettre cet affichage et d'en modifier l'apparence.

`AfficherAxesSym`	`0`	Booléen permettant l'affichage des axes de symétrie
`AxesSymLineStyle`	`solid`	Style de trait des axes de symétrie
`AxesSymWidth`	`thinlines`	Épaisseur de trait des axes de symétrie
`AxesSymColor`	`crimson`	Couleur de trait des axes de symétrie
`AxesSymStrokeOpacity`	`1`	Opacité du trait des axes de symétrie
`AxesSymEcart`	`0.25`	Écart entre les extrémité des axes et le bord de la frise
`AfficherAxesGliss`	`0`	Booléen permettant l'affichage des axes de glissements
`AxesGlissLineStyle`	`solid`	Style de trait des axes de glissements
`AxesGlissWidth`	`thinlines`	Épaisseur de trait des axes de glissements
`AxesGlissColor`	`dimgray`	Couleur de trait des axes de glissements
`AxesGlissStrokeOpacity`	`1`	Opacité du trait des axes de glissements

AfficherCentres1	0	Booléen permettant l'affichage des centres de symétries
CtrDotStyle1	diamond	Style des centres de symétries
CtrDotSize1	2+2*i	Taille des centres de symétries
CtrDotScale1	1+i	Facteurs d'échelle des centres de symétries
CtrDotAngle1	0	Angle de rotation des centres de symétries
CtrColor1	black	Couleur des centres de symétries
CtrFillColor1	white	Couleur de remplissage des centres de symétries

TABLE 1.6 – Variables globales relatives à l'affichage et à la modification des attributs des axes de symétries et de glissements et des centres de symétries

Là encore, la plupart de ces variables sont des variables de modifications d'attributs classiques. On notera néanmoins la présence de la variable `AxesSymEcart` qui définit un écart entre le frise et les extrémités de axes de symétries orthogonaux à la frise. Cette variable fonctionne de la même manière que la variable `BordEcart` précédemment décrite : il s'agit d'un pourcentage de la largeur de la frise.

1.3.5 Ajustement de la fenêtre graphique

`Cristallo.mac` dispose de deux autres variables globales permettant d'ajuster la fenêtre graphique à la frise en cours de création. La première se nomme `AjusterFenetre` et est un booléen pouvant prendre les valeurs de 0 ou 1 : si elle vaut 0, les dimensions de la fenêtre graphique ne seront pas gérées par la macro `DrawFrise`. Cette opération devra donc se faire « manuellement ». Si on attribue la valeur 1 à `AjusterFenetre`, alors `DrawFrise` s'occupe elle même de la taille de la fenêtre. Cette taille dépendra de la variable globale `FenetreEcart` qui définit un écart entre les bords de la fenêtre graphique et la frise en cours de création. Par défaut cette variable vaut 0.05.

AjusterFenetre	0	Booléen permettant l'ajustement de la fenêtre graphique à la frise
FenetreEcart	0.05	Écart entre les bords de la fenêtre et la frise

TABLE 1.7 – Variables globales relatives à l'ajustement de la fenêtre graphique à la frise

1.3.6 Ordre d'affichage des différents éléments d'une frise

On a vu que les frises que l'on pouvait créer avec le fichier `Cristallo.mac` étaient constituées d'une multitude d'objets (motif, vecteur générateur, point de référence, axes de symétrie, ...) dont on pouvait modifier les attributs grâce à un certain nombre de variables globales. On remarque ainsi que si par exemple on décide d'afficher les axes d'une frise, ceux-ci peuvent très bien se superposer au motif. Ainsi, suivant les attributs de remplissage et l'ordre d'affichage des motifs et des axes, les axes peuvent être cachés ou bien se superposer aux motifs. Il peut donc être intéressant d'avoir un plus grand contrôle sur l'ordre d'affichage des différents éléments d'une frise. Pour cela, `Cristallo.mac` dispose de la variable globale `FriseOrdreAffichage` permettant de contrôler l'ordre dans lequel vont être afficher les différents éléments d'une frise. `FriseOrdreAffichage` est une liste donnant cet ordre. Pour cela, on utilisera les variables globales suivantes représentant chacun des différents objets affichables :

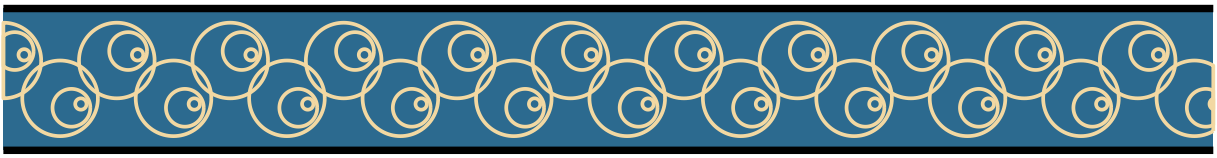
- ⋆ `axesgliss` : axes de glissements,
- ⋆ `axessym` : axes de symétries,
- ⋆ `background` : arrière-plan de la frise,
- ⋆ `bord` : bord de la frise,
- ⋆ `cadre` : encadrement du motif de base,
- ⋆ `centres1` : centres de symétrie,
- ⋆ `motif` : motifs,
- ⋆ `ptref` : point de référence de la frise,
- ⋆ `vecteur1` : vecteur générateur de la frise.

Par défaut, la variable `FriseOrdreAffichage` vaut :

```
[background, bord, cadre, motif, axessym, axesgliss, centres1, vecteur1, ptref]
```

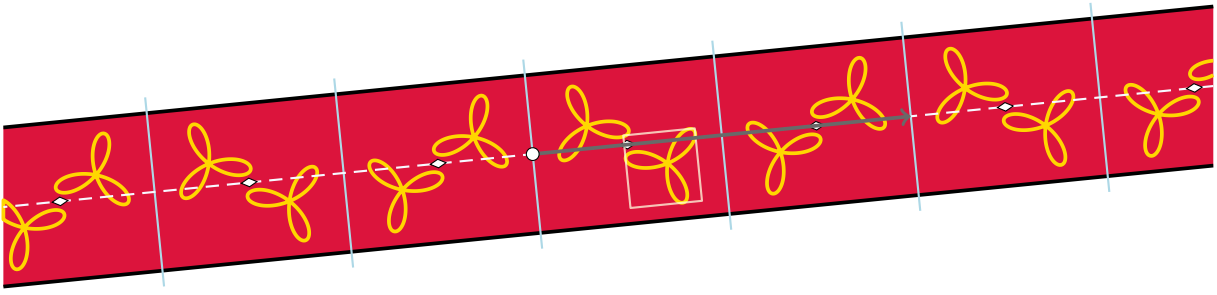
Ainsi, par défaut, la macro `DrawFrise` commencera par afficher l'arrière-plan de la frise, puis, elle dessinera le bord, ensuite elle affichera le cadre de mise en valeur du motif de base, etc...

1.4 Exemples



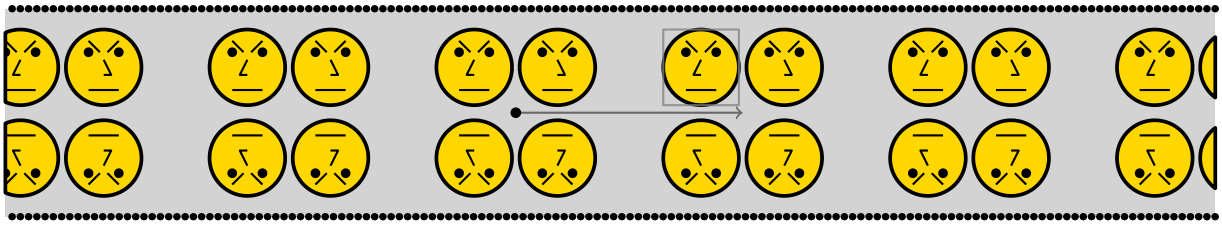
Code

```
[
Fenetre(-16-i, 16+i, 0.5*(1+i)),
tMin:=0, tMax:=pi, NbPoints:=100,
Motif:=[Seq(2+0.75*i+exp(k*i*pi/48), k, 0, 95), jump,
        Seq(2.3+i+0.5*exp(k*i*pi/48), k, 0, 95), jump,
        Seq(2.55+0.9*i+0.15*exp(k*i*pi/48), k, 0, 95), jump],
MotifFerme:=1,
DrawFrise("f1g",
        [
        Vecteur1:=3, PtRef:=0.5+0.25*i,
        BordWidth:=28, BordColor:=black, BackgroundFillColor:=HexaColor("2C6A8F"),
        MotifFillStyle:=none, MotifWidth:=Thicklines, MotifColor:=HexaColor("F2D8A2"),
        ])
]
```



Code

```
[
Fenetre(-4-0.7*i, 4+1.4*i, 2+2*i),
tMin:=0, tMax:=pi, NbPoints:=100,
$L:=Get(Courbe(0.7*sin(3*t)*cos(t)+i*0.7*sin(3*t)*sin(t), t)),
Motif:=simil(L, 0.5+0.5*i, 0.4, pi/8),
DrawFrise("fm2",
        [
        Vecteur1:=2.5+0.25*i, PtRef:=-0.5+0.3*i,
        BordWidth:=Thicklines, BordColor:=black, BackgroundFillColor:=crimson,
        MotifFillStyle:=none, MotifWidth:=Thicklines, MotifColor:=gold,
        AfficherAxesSym:=1, AxesSymWidth:=thicklines, AxesSymColor:=lightblue,
        AfficherAxesGliss:=1, AxesGlissLineStyle:=dashed,
        AxesGlissWidth:=thicklines, AxesGlissColor:=ghostwhite,
        AfficherCentres1:=1, CtrDotAngle1:=180*Arg(Vecteur1)/pi+90,
        AfficherPtRef:=1, PtRefDotStyle:=dotcircle, PtRefDotSize:=2.5+2*i,
        AfficherVecteur1:=1, VecteurWidth1:=Thicklines,
        EncadrerMotif:=1, CadreColor:=lightyellow
        ])
]
```



Code

```
[
{ Fenêtre graphique }
Fenetre(-8-i, 8+1.8*i, 1+i),
{ Options communes aux trois parties }
PtRef:=-1.25+0.4*i, Vecteur1:=3,
AfficherAxesSym:=0, AfficherAxesGliss:=0, AfficherCentres1:=0,
{ Première partie de la frise }
DrawFrise("f2m",
        [
        Motif:=Seq(1.2+i+exp(i*$x)/2, x, 0, 2*pi, pi/48), MotifFerme:=1,
        BackgroundFillStyle:=full, BackgroundFillColor:=lightgray,
        BordLineStyle:=dotted, BordWidth:=mm,
        MotifFillStyle:=full, MotifFillColor:=gold,
        MotifLineStyle:=solid, MotifWidth:=Thicklines, MotifColor:=black,
        AfficherPtRef:=1,
        AfficherVecteur1:=1,
        EncadrerMotif:=1,
        ]),
{ Seconde partie de la frise }
DrawFrise("f2m",
        [
        Motif:=[Seq(1+1.2*i+0.05*exp(i*x), x, 0, 2*pi, pi/24), jump,
                Seq(1.4+1.2*i+0.05*exp(i*x), x, 0, 2*pi, pi/24), jump],
        MotifFerme:=0,
        MotifWidth:=thicklines, MotifFillColor:=black,
        BackgroundFillStyle:=none, BordLineStyle:=noline,
        AfficherPtRef:=0,
        AfficherVecteur1:=0,
        EncadrerMotif:=0
        ]),
```

Code (Suite)

```
{ Troisième partie de la frise }
DrawFrise("f2m",
        [
        Motif:=[1.2+1.1*i, 1.1+0.9*i, 1.2+0.9*i, jump,
                1+1.35*i, 1.15+1.2*i, jump,
                1.25+1.2*i, 1.4+1.35*i, jump,
                1+0.7*i, 1.4+0.7*i, jump],
        MotifFillStyle:=none,
        ])
]
```

1.5 Récapitulatif des variables globales utiles pour la création de frises

Booléens	AfficherAxesGliss	0	Affichage des axes de glissements
	AfficherAxesSym	0	Affichage des axes de symétries
	AfficherCentres1	0	Affichage des centres de symétrie
	AfficherPtRef	0	Affichage du point de référence de la frise
	AfficherVecteur1	0	Affichage du vecteur générateur de la frise
	AjusterFenetre	0	Ajustement de la fenêtre graphique
	EncadrerMotif	0	Encadrement du motif de base
	MotifFerme	0	Motif fermé ou non
Lignes	AxesGlissLineStyle	solid	Style de trait des axes de glissements
	AxesSymLineStyle	solid	Style de trait des axes de symétries
	BordLineStyle	solid	Style de trait du bord de la frise
	CadreLineStyle	solid	Style de trait de l'encadrement du motif
	MotifLineStyle	solid	Style de trait du motif de base
	VecteurLineStyle1	solid	Style de trait du vecteur générateur
	AxesGlissWidth	thinlines	Épaisseur de trait des axes de glissements
	AxesSymWidth	thinlines	Épaisseur de trait des axes de symétries
	BordWidth	thicklines	Épaisseur de trait du bord de la frise
	CadreWidth	thicklines	Épaisseur de trait de l'encadrement du motif
	MotifWidth	thicklines	Épaisseur de trait du motif de base
	VecteurWidth1	thicklines	Épaisseur de trait du vecteur générateur
	AxesGlissColor	dimgray	Couleur des axes de glissements
	AxesSymColor	crimson	Couleur des axes de symétries
	BordColor	black	Couleur de trait du bord de la frise
	CadreColor	gray	Couleur de trait de l'encadrement du motif
	MotifColor	darkred	Couleur de trait du motif de base
	VecteurColor1	dimgray	Couleur du vecteur générateur
	AxesGlissStrokeOpacity	1	Opacité du trait des axes de glissements
	AxesSymStrokeOpacity	1	Opacité du trait des axes de symétries
	BordStrokeOpacity	1	Opacité du trait des bords de la frise
	CadreStrokeOpacity	0.75	Opacité du trait de l'encadrement du motif
	MotifStrokeOpacity	1	Opacité du trait du motif de base
	VecteurStrokeOpacity1	1	Opacité du trait du vecteur générateur

Remplissages	`BackgroundFillStyle`	`full`	Style de remplissage de l'arrière-plan
	`MotifFillStyle`	`full`	Style de remplissage du motif de base
	`CadreFillStyle`	`none`	Style de remplissage de l'encadrement
	`BackgroundFillColor`	`lightpink`	Couleur de remplissage de l'arrière-plan
	`CadreFillColor`	`lightgray`	Couleur de remplissage de l'encadrement
	`MotifFillColor`	`lightgray`	Couleur de remplissage du motif de base
	`BackgroundFillOpacity`	`1`	Opacité du remplissage de l'arrière-plan
	`CadreFillOpacity`	`0.5`	Opacité du remplissage de l'encadrement
	`MotifFillOpacity`	`1`	Opacité du remplissage du motif de base
Points	`CtrDotStyle1`	`diamond`	Style de point des centres de symétrie
	`PtRefDotStyle`	`dot`	Style de point du point de référence
	`CtrDotSize1`	`2+2*i`	Taille des centres de symétrie
	`PtRefDotSize`	`2+2*i`	Taille du point de référence
	`CtrDotScale1`	`1+i`	Facteurs d'échelle des centres de symétrie
	`PtRefDotScale`	`1+i`	Facteurs d'échelle du point de référence
	`CtrDotAngle1`	`0`	Angle de rotation des centres de symétrie
	`PtRefDotAngle`	`0`	Angle de rotation du point de référence
	`CtrColor1`	`black`	Couleur des centres de symétrie
	`PtRefColor1`	`black`	Couleur du point de référence
	`CtrFillColor1`	`white`	Couleur de remplissage des centres de symétrie
	`PtRefFillColor1`	`white`	Couleur de remplissage du point de référence
Autres variables	`PtRef`	`0`	Point de référence de la frise
	`Vecteur1`	`1`	Vecteur générateur de la frise
	`Motif`	`[...]`	Motif de base de la frise
	`AxesSymEcart`	`0.25`	Écart entre les extrémités des axes et le bord
	`BordEcart`	`0.125`	Écart entre le bord de la frise et les motifs
	`FenetreEcart`	`0.05`	Écart entre la frise et le bord de la fenêtre
	`MotifArrondi`	`0`	Arrondi des angles du motif de base
	`FriseOrdreAffichage`	`[...]`	Ordre d'affichage des éléments de la frise

Chapitre 2

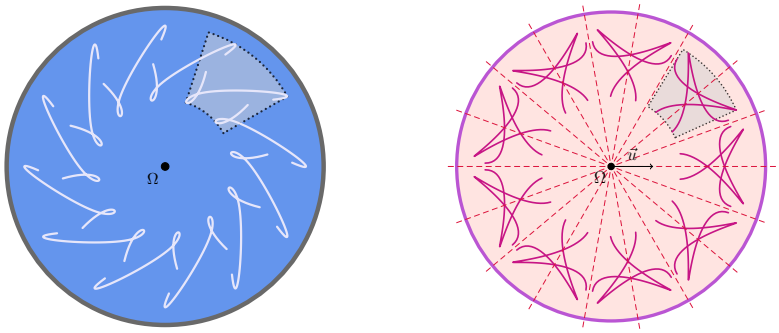
Rosaces

2.1 Rosaces de type rn et rnm

Outre les frises périodiques et autres pavages du plan, le fichier `Cristallo.mac` permet de créer simplement des rosaces compte tenu d'un motif initial donné par l'utilisateur. Il existe deux classes infinies de rosaces : les rosaces de type rn et les rosaces de type rnm, avec $n \in \mathbb{N} \setminus \{0, 1\}$.
Les rosaces de type rn sont simplement construites en appliquant au motif initial n rotations successives par rapport à un centre Ω donné. Les angles de ces rotations sont $\alpha_k = \frac{2k\pi}{n}$, pour $k \in \{1, 2, \ldots, n\}$. Le dessin finalement obtenu sera ainsi invariant par n'importe quel rotation de centre Ω et d'angle $\alpha_k = \frac{2k\pi}{n}$, $k \in \mathbb{Z}$. Le groupe de symétrie de ce type de rosaces sera donc composé des toutes ces rotations. La figure 2.1a est un exemple de rosace de type rn.
Les rosaces de type rnm sont un peu plus complexes que les précédentes. Ces dernières sont construites en deux étapes. Étant donné un axe $(\Omega, \vec{u})$ passant par le centre Ω, on dessine dans un premier temps le symétrique du motif initial par rapport à cet axe ce qui nous permet d'obtenir un nouveau motif de base. Ensuite, comme pour les rosaces de type rn on applique à ce nouveau motif n rotations de centre Ω et d'angles $\alpha_k = \frac{2k\pi}{n}$. Ainsi, le groupe de symétrie de ce type de rosaces est composé non seulement des rotations de centre Ω et d'angle $\alpha_k = \frac{2k\pi}{n}$, $k \in \mathbb{Z}$, mais aussi des symétries d'axe $(\Omega, \vec{u}_k)$ où les vecteurs $\vec{u}_k$ sont définis par :

$$\vec{u}_k = \begin{cases} \rho_{\frac{2k\pi}{n}}(\vec{u}), & \text{si } k \text{ est impair} \\ \rho_{\frac{k\pi}{n}}(\vec{u}), & \text{si } k \text{ est pair} \end{cases}$$

où ρ_θ désigne la rotation vectorielle d'angle θ.



(a) Rosace de type rn ($n = 12$) (b) Rosace de type rnm ($n = 9$)

FIGURE 2.1 – Exemples de rosaces de type rn et rnm

La suite de ce chapitre décrit la manière de produire ces rosaces grâce au fichier `Cristallo.mac`. En particulier, nous verrons en détail la syntaxe de la macro `DrawRosace` permettant de créer de

tels dessins et nous énumèrerons l'ensemble des variables globales utiles pour modifier l'aspect d'une rosace.

2.2 Création de rosaces – Macro `DrawRosace`

Comme on vient de le voir, il existe deux classes infinies de rosaces. Ces classes dépendent d'un entier n déterminant le nombre de rotations que l'on veut effectuer sur le motif de base. La syntaxe de la macros `DrawRosace` permettant de dessiner de tels dessins tient compte de ces remarques. Ainsi, cette fonction possède la syntaxe suivante :

```
DrawRosace(<Type>, <n>, [Options])
```

On note que la syntaxe est très proche de celle de la macro `DrawFrise`. En effet, la seule différence par rapport à cette dernière est la présence d'un second argument obligatoire, à savoir le nombre n de rotations évoqué plus tôt. Cette fois, le paramètre obligatoire `Type` peut prendre deux valeurs : `"rn"` et `"rnm"`.
Le troisième argument optionnel est encore une liste d'affectations de variables globales. La section suivante va nous permettre d'énumérer l'ensemble des variables globales ayant une influence sur l'apparence de la rosace en cours de création.

2.3 Variables globales relatives aux rosaces

Le troisième argument optionnel de la macro `DrawRosace` permet à l'utilisateur de modifier les variables globales du fichier `Cristallo.mac` pour modifier l'apparence de son graphique. Comme pour les frises, il est possible de réaliser ces affectations hors de la macros ce qui peut s'avérer utile pour rendre le code du graphique plus clair. Nous allons passer en revue l'ensemble des variables globales ainsi modifiable ayant des répercussion sur la création d'une rosace. On remarquera qu'un certain nombre d'entres elles peuvent également être utilisées dans le cadre de la création de frises périodiques.

2.3.1 Point de référence et vecteur de base

Pour construire une rosace, l'utilisateur va devoir déterminer un point de référence Ω de son graphique qui sera en fait le centre des rotations à appliquer à notre motif de base. Suivant le type de rosace que l'on souhaite créer, il faudra également définir le vecteur $\vec{u}$ qui permet de diriger un des axes de réflexion (pour les rosaces rnm). Pour cela, le fichier `Cristallo.mac` dispose de deux variables globales nommées `PtRef` et `Vecteur1` qui contiennent les affixes respectives du centre Ω et du vecteur directeur $\vec{u}$. Par défaut, ces deux variables valent respectivement 0 et 1.

Comme pour les frise, on peut être intéressé par l'affichage de ce point de référence et de ce vecteur directeur. Ainsi, `Cristallo.mac` met à disposition un jeu de variables globales permettant de décider de l'affichage de ces objets et d'en modifier les attributs. Ces variables sont listées dans le tableau 2.1 qui suit :

PtRef	0	Point de référence de la rosace (centre Ω)
Vecteur1	1	Vecteur directeur de l'axe de symétrie (vecteur $\vec{u}$)
AfficherPtRef	0	Booléen permettant l'affichage du point de référence
PtRefDotStyle	dot	Style du point de référence
PtRefDotSize	2+2*i	Taille du point de référence
PtRefDotScale	1+i	Facteurs d'échelle du points de référence
PtRefDotAngle	0	Angle de rotation du point de référence
PtRefColor	black	Couleur du point de référence
PtRefFillColor	white	Couleur de remplissage du point de référence
AfficherVecteur1	0	Booléen permettant l'affichage du vecteur $\vec{u}$
VecteurLineStyle1	solid	Style de trait du vecteur $\vec{u}$
VecteurWidth1	thicklines	Épaisseur de trait du vecteur $\vec{u}$
VecteurColor1	dimgray	Couleur de trait du vecteur $\vec{u}$
VecteurStrokeOpacity1	1	Opacité du trait du vecteur $\vec{u}$

TABLE 2.1 – Variables globales relatives à l'affichage du centre de la rosace et du vecteur directeur de l'axe de symétrie

2.3.2 Motif de base

Comme pour les frise, les rosaces se basent sur un motif de base qui va être répété un nombre fini de fois par rotation autour du centre de la rosace. Les variables énumérées au chapitre 1 restent donc valable. Nous les remettons dans ce chapitre afin d'éviter au lecteur des aller-retours entre les deux chapitres.

Motif	[...]	Motif de base de la frise
MotifFerme	0	Booléen indiquant si le motif est une ligne fermé ou non
MotifArrondi	0	Arrondi des angles du motif
MotifLineStyle	solid	Style de trait du motif
MotifWidth	thicklines	Épaisseur de trait du motif
MotifColor	darkred	Couleur de trait du motif
MotifStrokeOpacity	1	Opacité de trait du motif
MotifFillStyle	full	Style de remplissage du motif
MotifFillColor	lightgray	Couleur de remplissage du motif
MotifFillOpacity	1	Opacité du remplissage du motif

TABLE 2.2 – Variables globales relatives aux attributs du motif de base de la rosace

Comme pour les frises périodiques, il est possible de mettre en évidence le motif de base de la frise en encadrant celui-ci. Comme nous somme dans le cadre d'une rosace, l'encadrement du motif ne sera plus un rectangle mais un secteur circulaire centré en Ω. Néanmoins, cela ne change rien à la manière de procéder pour afficher cet encadrement et en modifier les attributs. En effet, les mêmes variables globales que dans la section 1.3.2 sont disponibles pour les rosaces et s'utilisent de la même façon :

EncadrerMotif	0	Booléen permettant l'encadrement du motif de base
CadreLineStyle	solid	Style de trait de l'encadrement du motif
CadreWidth	thicklines	Épaisseur de trait de l'encadrement du motif
CadreColor	gray	Couleur de trait de l'encadrement du motif
CadreStrokeOpacity	0.75	Opacité du trait de l'encadrement du motif
CadreFillStyle	none	Style de remplissage de l'encadrement du motif
CadreFillColor	lightgray	Couleur de remplissage de l'encadrement du motif
CadreFillOpacity	0.5	Opacité du remplissage de l'encadrement du motif

TABLE 2.3 – Variables globales relatives à l'encadrement du motif de base d'une rosace

2.3.3 Arrière-plan de la rosace

L'arrière plan d'une rosace sera délimité par un cercle de centre Ω et dont le rayon va varier selon les valeurs de la variable `BordEcart`. Cette variable permet de définir l'écart entre les motifs de la frise et le bord de celle-ci. Elle agit sur l'apparence d'une rosace de manière similaire que pour les frises (voir la section 1.3.3).
Ainsi, en plus du style de remplissage d'une rosace, `Cristallo.mac` met à disposition des variables globale permettant l'affichage du bord d'une rosace, c'est à dire du cercle la délimitant. Toutes ces variables sont énumérées dans le tableau suivant :

`BordEcart`	`0.125`	Écart entre le bord de la rosace et les motifs
`BordLineStyle`	`solid`	Style de trait du bord de la rosace
`BordWidth`	`thicklines`	Épaisseur de trait du bord de la rosace
`BordColor`	`black`	Couleur de trait du bord de la rosace
`BordStrokeOpacity`	`1`	Opacité de trait du bord de la rosace
`BackgroundFillStyle`	`full`	Style de remplissage de l'arrière-plan de la rosace
`BackgroundFillColor`	`lightpink`	Couleur de remplissage de l'arrière-plan de la rosace
`BackgroundFillOpacity`	`1`	Opacité du remplissage de l'arrière-plan de la rosace

TABLE 2.4 – Variables globales relatives aux attributs de l'arrière-plan d'une rosace

2.3.4 Axes de symétries

Pour les rosaces de type rnm, on a vu que le groupe de symétrie était composé de rotations et de symétries axiales. Il est alors possible d'afficher les axes de symétries d'une rosace de ce type en attribuant la valeur 1 au booléen `AfficherAxesSym`. Là encore, quelques variables globales permettent de modifier les attributs de ces axes. Celles-ci sont regroupées dans le tableau 2.5. La variable `AxesSymEcart` permet de définir la position des extrémités de ces axes de symétrie. Il s'agit d'un nombre représentant un pourcentage du rayon de la frise.

`AfficherAxesSym`	`0`	Booléen permettant l'affichage des axes de symétrie
`AxesSymLineStyle`	`solid`	Style de trait des axes de symétrie
`AxesSymWidth`	`thinlines`	Épaisseur de trait des axes de symétrie
`AxesSymColor`	`crimson`	Couleur de trait des axes de symétrie
`AxesSymStrokeOpacity`	`1`	Opacité du trait des axes de symétrie
`AxesSymEcart`	`0.25`	Écart entre les extrémité des axes et le bord de la rosace

TABLE 2.5 – Variables globales relatives à l'affichage et à la modification des attributs des axes de symétries

2.3.5 Ajustement de la fenêtre graphique

Une fois encore, il est possible d'obtenir un ajustement automatique de la fenêtre graphique à la rosace en cours de création. Cet ajustement se fait par le biais de la variable globale `AjusterFenetre` qui vaut 1 si un tel ajustement est souhaité, ou 0 si l'on désire réaliser ces réglages « manuellement ». L'ajustement automatique de la fenêtre prend en compte les valeurs des variables globales `BordEcart` et `AxesSymEcart` (pour les rosaces rnm uniquement). Une autre variable nommé `FenetreEcart` permet de définir un écart entre le bord de la fenêtre graphique et le bord de la rosace. Par défaut cette variable vaut 0,05.

`AjusterFenetre`	`0`	Booléen permettant l'ajustement de la fenêtre graphique à la rosace
`FenetreEcart`	`0.05`	Écart entre les bords de la fenêtre et la rosace

TABLE 2.6 – Variables globales relatives à l'ajustement de la fenêtre graphique à la rosace

2.3.6 Ordre d'affichage des éléments d'une rosace

Comme pour les frises périodiques, l'ordre dans lequel les différents éléments d'une rosace (point de référence, motif, axes, ...) va avoir une influence sur l'apparence globale de la rosace. En effet, en fonction des attributs choisis (notamment avec des remplissages de type `full`), certains objet peuvent être recouverts par d'autre suivant l'ordre dans lequel ils ont été affichés. Pour contrôler cet ordre d'affichage, `Cristallo.mac` dispose de la variable globale `RosaceOrdreAffichage`. Cette fonction contient une liste de valeur permettant de déterminer l'ordre dans lequel ont veut afficher les différents composants de le rosace. Pour plus de clarté, on fera intervenir lors de la modification de cette variables globales les constantes suivantes :

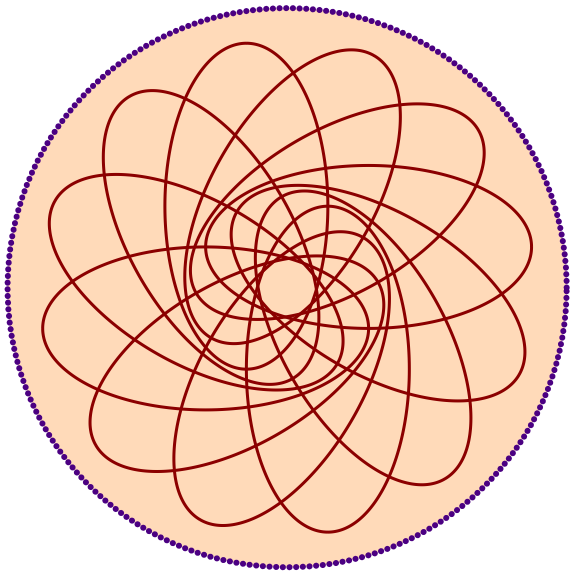
- ⋆ `axessym` : axes de symétrie,
- ⋆ `background` : arrière-plan de la rosace,
- ⋆ `bord` : bord de la rosace,
- ⋆ `cadre` : encadrement du motif de base,
- ⋆ `motif` : motif de base,
- ⋆ `ptref` : point de référence de la rosace,
- ⋆ `vecteur1` : vecteur directeur d'un axe de symétrie.

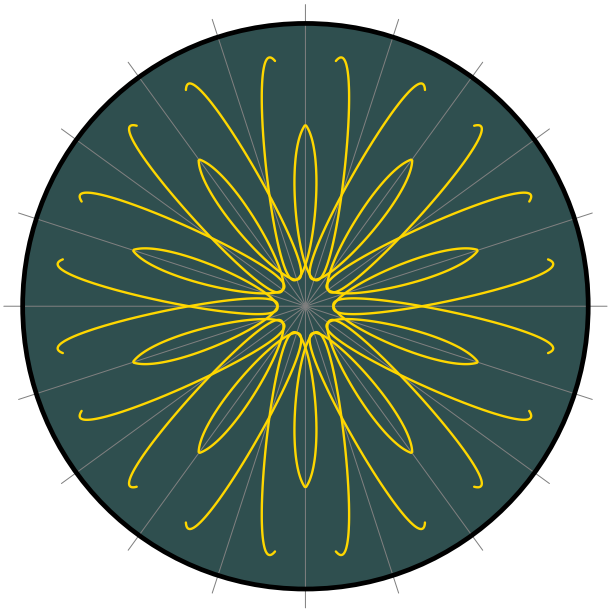
Par défaut, la variable globale `RosaceOrdreAffichage` vaut :

```
[background, bord, cadre, motif, axessym, vecteur1, ptref]
```

Ainsi, `DrawRosace` commencera d'abord par remplir l'arrière-plan de la rosace, ensuite elle dessinera le bord et ainsi de suite...

2.4 Exemples



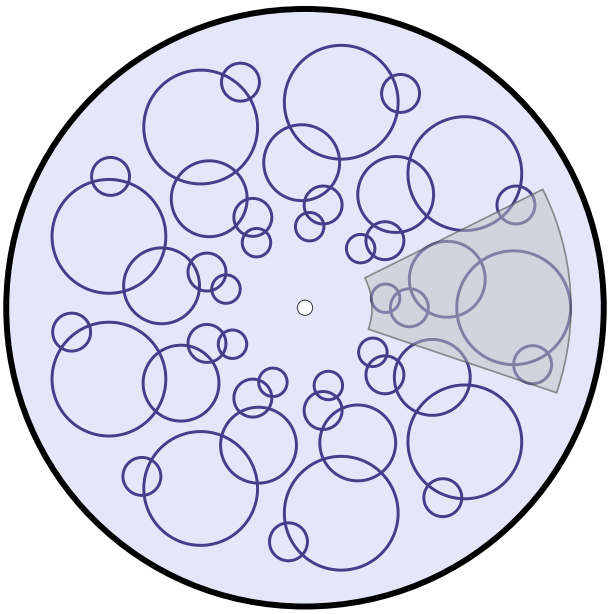


Code (Exemple 1)

```
[
Marges(0, 0, 0, 0),
AjusterFenetre:=1,
Motif:=Seq(1+2*cos(x)+i*(0.5+sin(x)), x, 0, 2*pi, pi/180),
MotifLineStyle:=solid, MotifWidth:=Thicklines, MotifColor:=darkred,
BordLineStyle:=dotted, BordWidth:=mm, BordColor:=indigo,
BackgroundFillStyle:=full, BackgroundFillColor:=peachpuff,
DrawRosace("rn", 12)
]
```

Code (Exemple 2)

```
[
Marges(0, 0, 0, 0),
LineCap:=round, NbPoints:=200,
AjusterFenetre:=1, FenetreEcart:=0,
$L:=[0.52521010418422+0.14113363274373*i,0.70133064896799+0.75031993491575*i,
        0.13199483603239+0.06142165651544*i,0.43978655920364+0.60954159894026*i,
        0.48058554017916+0.16553819854743*i],
Motif:=Get(Spline(L)),
MotifLineStyle:=solid, MotifWidth:=Thicklines, MotifColor:=gold,
BordLineStyle:=solid, BordWidth:=mm, BordColor:=black,
BackgroundFillStyle:=full, BackgroundFillColor:=darkslategray,
AfficherAxesSym:=1, AxesSymEcart:=0.2,
AxesSymLineStyle:=solid, AxesSymWidth:=thinlines, AxesSymColor:=gray,
RosaceOrdreAffichage:=[background, axessym, bord, motif],
DrawRosace("rnm", 10)
]
```



Code

```
[
Marges(0, 0, 0, 0),
NbPoints:=100,
AjusterFenetre:=1, FenetreEcart:=0,
PtRef:=-0.1,
AfficherPtRef:=1, PtRefDotSize:=4+4*i,
Motif:=[Get(Cercle(1.1-0.3*i, 0.1)), jump, Get(Cercle(1, 0.3)), jump,
    Get(Cercle(0.65+0.15*i, 0.2)), jump, Get(Cercle(0.45, 0.1)), jump,
    Get(Cercle(0.325+0.05*i, 0.075)), jump],
MotifLineStyle:=solid, MotifWidth:=Thicklines, MotifColor:=darkslateblue,
EncadrerMotif:=1,
CadreLineStyle:=solid, CadreWidth:=thicklines, CadreColor:=dimgray,
CadreFillStyle:=full, CadreFillColor:=silver,
BordLineStyle:=solid, BordWidth:=mm, BordColor:=black,
BackgroundFillStyle:=full, BackgroundFillColor:=lavender,
RosaceOrdreAffichage:=[background, bord, motif, cadre, ptref],
DrawRosace("rn", 9)
]
```

2.5 Tableau récapitulatif des variables globales

Booléens	AfficherAxesSym	0	Affichage des axes de symétries
	AfficherPtRef	0	Affichage du point de référence de la rosace
	AfficherVecteur1	0	Affichage du vecteur directeur de l'axe
	AjusterFenetre	0	Ajustement de la fenêtre graphique
	EncadrerMotif	0	Encadrement du motif de base
	MotifFerme	0	Motif fermé ou non
Lignes	AxesSymLineStyle	solid	Style de trait des axes de symétries
	BordLineStyle	solid	Style de trait du bord de la rosace
	CadreLineStyle	solid	Style de trait de l'encadrement du motif
	MotifLineStyle	solid	Style de trait du motif de base
	VecteurLineStyle1	solid	Style de trait du vecteur directeur de l'axe
	AxesSymWidth	thinlines	Épaisseur de trait des axes de symétries
	BordWidth	thicklines	Épaisseur de trait du bord de la rosace
	CadreWidth	thicklines	Épaisseur de trait de l'encadrement du motif
	MotifWidth	thicklines	Épaisseur de trait du motif de base
	VecteurWidth1	thicklines	Épaisseur de trait du vecteur directeur de l'axe
	AxesSymColor	crimson	Couleur des axes de symétries
	BordColor	black	Couleur de trait du bord de la rosace
	CadreColor	gray	Couleur de trait de l'encadrement du motif
	MotifColor	darkred	Couleur de trait du motif de base
	VecteurColor1	dimgray	Couleur du vecteur directeur de l'axe
	AxesSymStrokeOpacity	1	Opacité du trait des axes de symétries
	BordStrokeOpacity	1	Opacité du trait des bords de la frise
	CadreStrokeOpacity	0.75	Opacité du trait de l'encadrement du motif
	MotifStrokeOpacity	1	Opacité du trait du motif de base
	VecteurStrokeOpacity1	1	Opacité du trait du vecteur directeur de l'axe
Remplissages	BackgroundFillStyle	full	Style de remplissage de l'arrière-plan
	CadreFillStyle	none	Style de remplissage de l'encadrement
	MotifFillStyle	full	Style de remplissage du motif
	BackgroundFillColor	lightpink	Couleur de remplissage de l'arrière-plan
	CadreFillColor	lightgray	Couleur de remplissage de l'encadrement
	MotifFillColor	lightgray	Couleur de remplissage du motif
	BackgroundFillOpacity	1	Opacité du remplissage de l'arrière-plan
	CadreFillOpacity	0.5	Opacité du remplissage de l'encadrement
	MotifFillOpacity	1	Opacité du remplissage du motif
Points	PtRefDotStyle	dot	Style du point de référence
	PtRefDotSize	2+2*i	Taille du point de référence
	PtRefDotScale	1+i	Facteurs d'échelle du point de référence
	PtRefDotAngle	0	Angle de rotation du point de référence
	PtRefColor	black	Couleur du point de référence
	PtRefFillColor	white	Couleur de remplissage du point de référence
Autres variables	PtRef	0	Point de référence de la rosace
	Vecteur1	1	Vecteur directeur d'un axe de symétrie
	Motif	[...]	Motif de base de la rosace
	AxesSymEcart	0.25	Écart entre les extrémités des axes et le bord
	BordEcart	0.125	Écart entre le bord de la rosace et les motifs
	FenetreEcart	0.05	Écart entre la rosace et le bord de la fenêtre
	MotifArrondi	0	Arrondi des angles du motif de base
	RosaceOrdreAffichage	[...]	Ordre d'affichage des éléments de la rosace

Chapitre 3

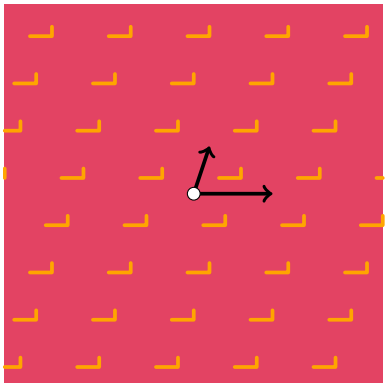
Pavages périodiques du plan euclidien

3.1 Pavages périodiques du plan

Le fichier `Cristallo.mac` permet en plus des frises périodiques et des rosaces de créer de manière similaire des pavages périodiques du plan euclidien. Nous ne donnerons pas dans ce document de détails théoriques concernant les pavages périodiques. Dans la suite, on se contentera d'une définition « intuitive » de la notion de pavage périodique.
Étant donné deux vecteur indépendants $\vec{u}$ et $\vec{v}$ du plan, on appellera pavage périodique la répétition d'un motif par translation de vecteur $\vec{w}_{\lambda,\mu} = \lambda\vec{u} + \mu\vec{v}$ où $(\lambda, \mu) \in \mathbb{Z} \times \mathbb{Z}$. Ce motif peut préalablement avoir subit d'autres transformations (symétries, glissements, rotations, ...). L'ensemble des ces transformations va permettre d'identifier différentes classes de pavages périodiques. Les pavages d'une même classe auront alors systématiquement le même groupe de symétrie.
Il existe au total 17 types de pavages périodiques du plan euclidien. La suite de ce paragraphe consiste en une énumération illustrée de ces classes de pavages. Pour chacun d'entre eux, on a précisé le type de réseau engendré, le groupe de symétrie du pavage ainsi que les différentes transformations en faisant partie.

Notations :
Dans la suite de ce paragraphe, concernant les groupe de symétrie, quelques notations supplémentaires ont été introduite. Elles sont énumérées ci-dessous :

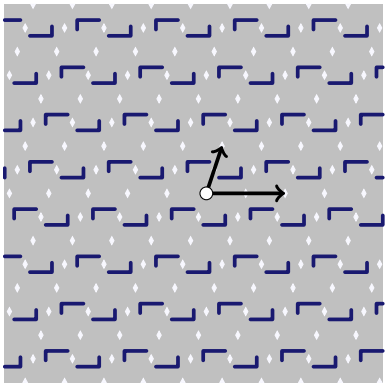
- $\vec{u}$ et $\vec{v}$ désigne les vecteurs générateurs du pavage périodique.
- Ω désigne le point de référence du pavage servant à positionner axes et centres de rotations.
- D est la droite $D(\Omega, \vec{u})$ passant par Ω et dirigée par $\vec{u}$.
- Δ est la droite $D(\Omega, \vec{v})$ passant par Ω et dirigée par $\vec{v}$.
- $\mathcal{D}$ est la droite $D(\Omega, \vec{u} + \vec{v})$ passant par Ω et dirigée par $\vec{u} + \vec{v}$.



p1

Réseau : oblique

Groupe de symétrie : $\langle t_{\vec{u}}, t_{\vec{v}} \rangle$
→ Translations



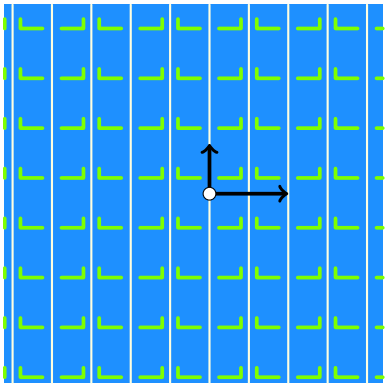
p2

Réseau : oblique

Groupe de symétrie : $\langle t_{\vec{u}}, t_{\vec{v}}, r_{\Omega,\pi} \rangle$

→ Translations

♦ Rotations d'angle π



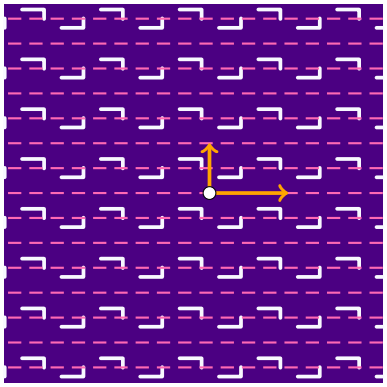
pm

Réseau : rectangulaire

Groupe de symétrie : $\langle t_{\vec{u}}, t_{\vec{v}}, s_{\Delta} \rangle$

→ Translations

— Symétries axiales



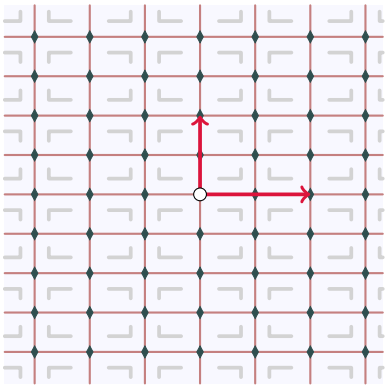
pg

Réseau : rectangulaire

Groupe de symétrie : $\left\langle t_{\vec{u}/2} \circ s_D, t_{\vec{v}} \right\rangle$

→ Translations

--- Symétries glissées



pmm

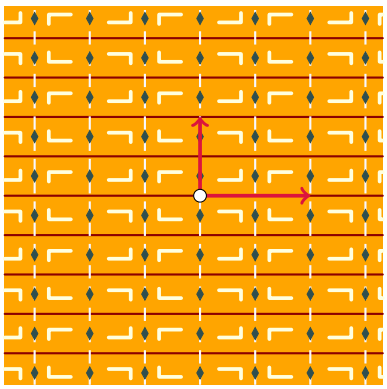
Réseau : rectangulaire

Groupe de symétrie : $\langle t_{\vec{u}}, t_{\vec{v}}, s_D, s_{\Delta} \rangle$

→ Translations

♦ Rotations d'angle π

— Symétries axiales



pmg

Réseau : rectangulaire

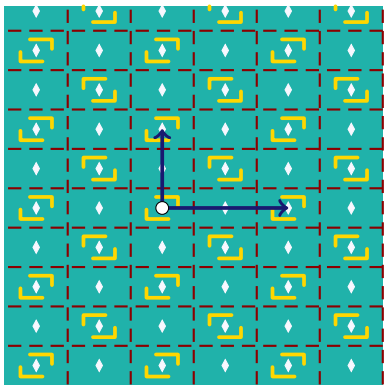
Groupe de symétrie : $\langle t_{\vec{u}}, t_{\vec{v}}, r_{\Omega',\pi}, s_D \rangle$, $\Omega' = \Omega + \vec{v}/4$

→ Translations

♦ Rotations d'angle π

— Symétries axiales

--- Symétries glissées

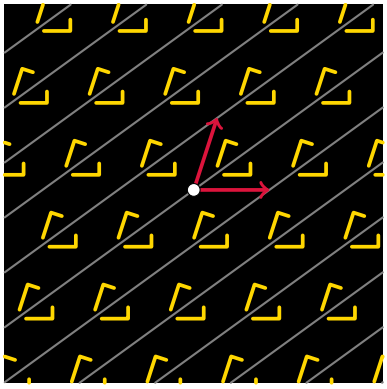


pgg

Réseau : rectangulaire

Groupe de symétrie : $\left\langle t_{\vec{u}}, t_{\vec{v}/2} \circ s_{\Delta'}, r_{\Omega,\pi} \right\rangle$, $\Delta' = t_{\vec{u}/4}(\Delta)$

- → Translations
- ♦ Rotations d'angle π
- --- Symétries glissées

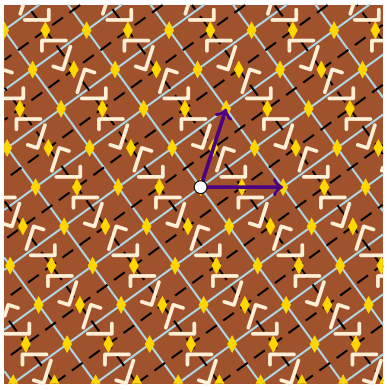


cm

Réseau : rhombique

Groupe de symétrie : $\langle t_{\vec{u}}, t_{\vec{v}}, s_{\mathcal{D}} \rangle$

- → Translations
- — Symétries axiales

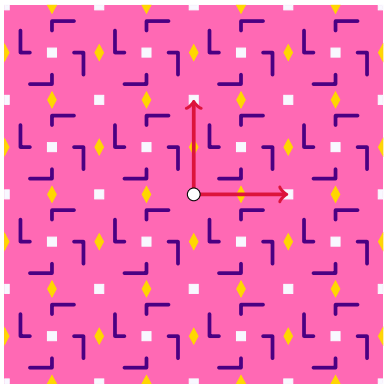


cmm

Réseau : rhombique

Groupe de symétrie : $\langle t_{\vec{u}}, t_{\vec{v}}, s_{\mathcal{D}}, s_{\mathcal{D}'} \rangle$, $\mathcal{D}' = D(\Omega, \vec{u} - \vec{v})$

- → Translations
- ♦ Rotations d'angle π
- — Symétries axiales
- --- Symétries glissées

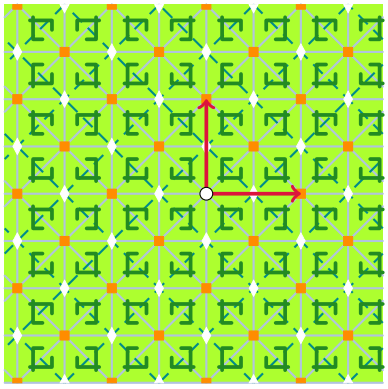


p4

Réseau : carré

Groupe de symétrie : $\left\langle t_{\vec{u}}, t_{\vec{v}}, r_{\Omega,\pi/2} \right\rangle$

- → Translations
- ♦ Rotations d'angle π
- ■ Rotations d'angle $\frac{\pi}{2}$

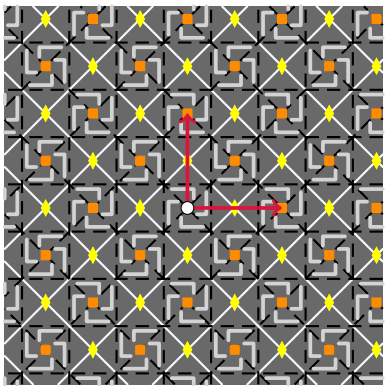


p4m

Réseau : carré

Groupe de symétrie : $\left\langle t_{\vec{u}}, t_{\vec{v}}, r_{\Omega,\pi/2}, s_D \right\rangle$

- → Translations
- ♦ Rotations d'angle π
- ■ Rotations d'angle $\frac{\pi}{2}$
- — Symétries axiales
- --- Symétries glissées

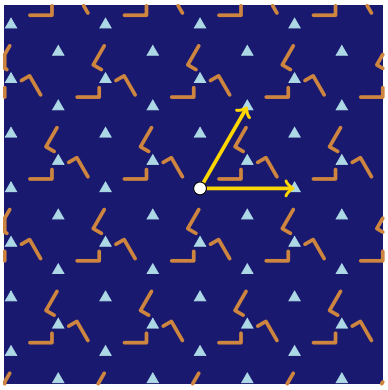


p4g

Réseau : carré

Groupe de symétrie : $\left\langle t_{\vec{u}}, t_{\vec{v}}, r_{\Omega',\pi/2}, s_{\mathcal{D}} \right\rangle$, $\Omega' = \Omega + \vec{u}/2$

- $\rightarrow$ Translations
- ♦ Rotations d'angle π
- ■ Rotations d'angle $\frac{\pi}{2}$
- — Symétries axiales
- --- Symétries glissées

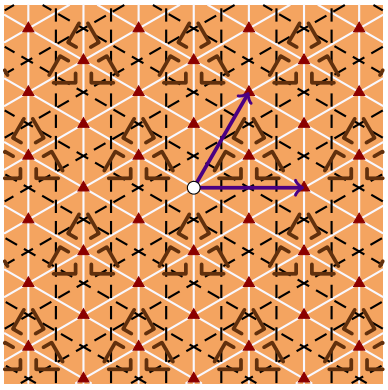


p3

Réseau : hexagonal

Groupe de symétrie : $\left\langle t_{\vec{u}}, t_{\vec{v}}, r_{\Omega,2\pi/3} \right\rangle$

- $\rightarrow$ Translations
- ▲ Rotations d'angle $\frac{2\pi}{3}$

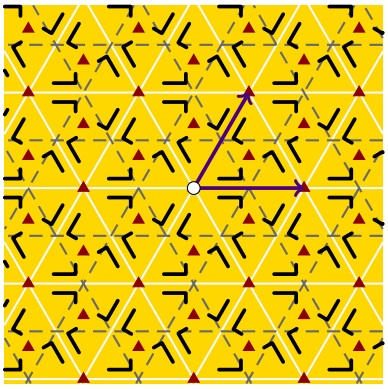


p3m1

Réseau : hexagonal

Groupe de symétrie : $\left\langle t_{\vec{u}}, t_{\vec{v}}, r_{\Omega,2\pi/3}, s_{\mathcal{D}} \right\rangle$

- $\rightarrow$ Translations
- ▲ Rotations d'angle $\frac{2\pi}{3}$
- — Symétries axiales
- --- Symétries glissées

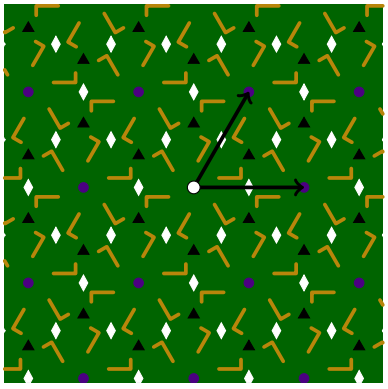


p31m

Réseau : hexagonal

Groupe de symétrie : $\left\langle t_{\vec{u}}, t_{\vec{v}}, r_{\Omega,2\pi/3}, s_{D} \right\rangle$

- $\rightarrow$ Translations
- ▲ Rotations d'angle $\frac{2\pi}{3}$
- — Symétries axiales
- --- Symétries glissées

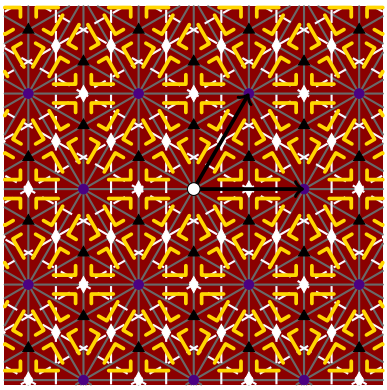


p6

Réseau : hexagonal

Groupe de symétrie : $\left\langle t_{\vec{u}}, t_{\vec{v}}, r_{\Omega,\pi/3} \right\rangle$

- $\rightarrow$ Translations
- ♦ Rotations d'angle π
- ▲ Rotations d'angle $\frac{2\pi}{3}$
- • Rotations d'angle $\frac{\pi}{3}$



p6m

Réseau : hexagonal

Groupe de symétrie : $\left\langle t_{\vec{u}}, t_{\vec{v}}, r_{\Omega,\pi/3}, s_D \right\rangle$

- $\rightarrow$ Translations
- ♦ Rotations d'angle π
- ▲ Rotations d'angle $\frac{2\pi}{3}$
- • Rotations d'angle $\frac{\pi}{3}$
- — Symétries axiales
- --- Symétries glissées

3.2 Création de pavages périodiques – Macro `DrawPavPeriodique`

Nous venons de voir qu'il existait 17 types différents de pavages du plan euclidien. Le fichier `Cristallo.mac` permet donc, à l'aide d'une simple macro, de créer ces pavages compte tenu d'un motif initial que l'utilisateur devra préalablement définir. La macro permettant de réaliser ces pavages se nomme `DrawPavPeriodique` et possède une syntaxe similaire aux macros `DrawFrise` et `DrawRosace` que l'on a décrit dans les chapitre précédents :

```
DrawPavPeriodique(<Type>, [Options])
```

Le premier paramètre obligatoire est une chaîne de caractère permettant d'identifier le pavage à créer. Le tableau 3.1 donne l'ensemble des valeurs possible que peut prendre ce paramètre (première colonne) ainsi que quelques caractéristiques des pavages correspondants.

`Type`	Réseau	Transformations					
		r_π	$r_{2\pi/3}$	$r_{\pi/2}$	$r_{\pi/3}$	s	g
`p1`	Oblique						
`p2`	Oblique	✓					
`pm`	Rectangulaire					✓	
`pg`	Rectangulaire						✓
`pmm`	Rectangulaire	✓				✓	
`pmg`	Rectangulaire	✓				✓	✓
`pgg`	Rectangulaire	✓					✓
`cm`	Rhombique					✓	
`cmm`	Rhombique	✓				✓	✓
`p4`	Carré	✓		✓			
`p4m`	Carré	✓		✓		✓	✓
`p4g`	Carré	✓		✓		✓	✓
`p3`	Hexagonal		✓				
`p3m1`	Hexagonal		✓			✓	✓
`p31m`	Hexagonal		✓			✓	✓
`p6`	Hexagonal	✓	✓		✓		
`p6m`	Hexagonal	✓	✓		✓	✓	✓

TABLE 3.1 – Valeurs possibles pour le paramètre `Type` et caractéristiques des pavages correspondants

Ce tableau permet donne la nature du réseau de chacun des 17 pavages ainsi que les transformations du plan présentes dans leur groupe de symétrie. Un symbole ✓ dans une colonne r_α signifie que le groupe de symétrie du pavage en question possède des rotation d'angle α. Quant aux colonnes s et g, elles indiquent la présence ou non de symétries axiales ou de glissements dans ces groupes de symétrie.

Comme pour les frises périodiques ou les rosaces, la macro `DrawPavPeriodique` possède un argument optionnel `Options` qui va permettre à l'utilisateur de faire les réglages réglages nécessaires pour obtenir le graphique voulu. Comme pour les frises et les rosaces, ces ajustements vont se faire par l'intermédiaire de variables globales qui seront décrites dans la section suivante. Une fois encore, les modifications de ces variables peuvent se faire à l'extérieur de la fonction ; bien souvent, vu la quantité de variables globales entrant en jeu dans la conception d'un pavage périodique, une telle pratique permettra de produire un code plus lisible qu'en insérant ces affectation comme deuxième argument de la macro `DrawPavPeriodique`.

3.3 Variables globales

La création de pavages périodique à l'aide du fichier `Cristallo.mac` fait intervenir de nombreuses variables globales. Nous allons en donner une liste exhaustive dans cette section en décrivant pour chacune d'entre elle leur utilité ainsi que les valeurs leur étant attribuées par défaut.

3.3.1 Point de référence et vecteur de base

Pour les pavages périodique aussi, il est nécessaire de disposer initialement d'un point de référence et de vecteurs générateurs qui vont nous permettre de définir correctement les centres des réflexions et les axes de réflexions et glissement présent dans les groupes de symétrie de chacun des pavages. Le point de référence d'un pavage est défini par la variable globale `PtRef`. Elle est initialisée à 0.
Contrairement aux frises, deux vecteurs générateurs sont nécessaires pour engendrer un pavage. Pour cela, `Cristallo.mac` possède les variables `Vecteur1` et `Vecteur2` qui contiennent des complexes, affixes de ces vecteurs générateurs. Le tableau suivant récapitule ces variables ainsi que leur valeur par défaut.

Comme pour les frises périodiques, il est possible d'afficher ce point de référence ainsi que les vecteurs générateurs de notre pavage. Pour cela, on attribuera la valeur 1 aux booléens correspondants, c'est à dire `AfficherPtRef`, `AfficherVecteur1` et `AfficherVecteur2`. Les attributs de ces éléments peuvent être modifiés par le biais des variables globales énumérées dans le tableau 3.2.

`PtRef`	`0`	Point de référence du pavage
`Vecteur1`	`1`	Premier vecteur générateur du pavage
`Vecteur2`	`i`	Second vecteur générateur du pavage
`AfficherPtRef`	`0`	Booléen permettant l'affichage du point de référence
`PtRefDotStyle`	`dot`	Style du point de référence
`PtRefDotSize`	`2+2*i`	Taille du point de référence
`PtRefDotScale`	`1+i`	Facteurs d'échelle du points de référence
`PtRefDotAngle`	`0`	Angle de rotation du point de référence
`PtRefColor`	`black`	Couleur du point de référence
`PtRefFillColor`	`white`	Couleur de remplissage du point de référence
`AfficherVecteur1`	`0`	Booléen permettant d'afficher le 1er vecteur générateur
`VecteurLineStyle1`	`solid`	Style de trait du 1er vecteur générateur
`VecteurWidth1`	`thicklines`	Épaisseur de trait du 1er vecteur générateur
`VecteurColor1`	`dimgray`	Couleur de trait du 1er vecteur générateur
`VecteurStrokeOpacity1`	`1`	Opacité du trait du 1er vecteur générateur
`AfficherVecteur2`	`0`	Booléen permettant d'afficher le 2nd vecteur générateur
`VecteurLineStyle2`	`solid`	Style de trait du 2nd vecteur générateur
`VecteurWidth2`	`thicklines`	Épaisseur de trait du 2nd vecteur générateur
`VecteurColor2`	`dimgray`	Couleur de trait du 2nd vecteur générateur
`VecteurStrokeOpacity2`	`1`	Opacité du trait du 2nd vecteur générateur

TABLE 3.2 – Variables globales relatives au point de référence et aux vecteurs générateurs du pavage

Il peut également être intéressant d'afficher sur notre pavage le réseau engendré par nos deux vecteurs de base. Pour cela, il suffit d'affecter au booléen `AfficherReseau` la valeur 1 et de modifier les attributs relatifs à l'affichage de ce réseau. Ces attributs se modifient grâce au variable globales présentes dans le tableau 3.3

`AfficherReseau`	`0`	Booléen permettant l'affichage du réseau du pavage
`ResauLineStyle`	`solid`	Style de ligne du réseau
`ReseauWidth`	`thinlines`	Épaisseur des lignes du réseau
`ReseauColor`	`gold`	Couleur des lignes du réseau
`ReseauStrokeOpacity`	`1`	Opacité des lignes du réseau

TABLE 3.3 – Variables globales relatives à l'affichage du réseau d'un pavage périodique

3.3.2 Motif de base

Comme pour les frises périodiques et les rosaces, les pavages périodiques du plan se basent sur un motif initial qui va subir diverses transformations pour finalement paver le plan. Ce motif n'est autre qu'une liste de complexes stockée dans la variable globale `Motif` qui représente une ligne polygonale. Les attributs de cette ligne polygonale sont modifiables grâce aux variables globales énumérées dans le tableau 3.4.

`Motif`	`[...]`	Motif de base du pavage
`MotifFerme`	`0`	Booléen indiquant si le motif est une ligne fermé ou non
`MotifArrondi`	`0`	Arrondi des angles du motif
`MotifLineStyle`	`solid`	Style de trait du motif
`MotifWidth`	`thicklines`	Épaisseur de trait du motif
`MotifColor`	`darkred`	Couleur de trait du motif
`MotifStrokeOpacity`	`1`	Opacité de trait du motif
`MotifFillStyle`	`full`	Style de remplissage du motif
`MotifFillColor`	`lightgray`	Couleur de remplissage du motif
`MotifFillOpacity`	`1`	Opacité du remplissage du motif

TABLE 3.4 – Variables globales relatives aux attributs du motif de base du pavage

Il est possible de mettre en évidence le motif de base d'un pavage périodique en l'encadrant. Pour cela, il suffit d'attribuer la valeur 1 au booléen `EncadrerMotif`. Les attributs de cet encadrement sont modifiables une fois encore via quelques variables globales qui sont regroupées dans le tableau 3.5.

`EncadrerMotif`	`0`	Booléen permettant l'encadrement du motif de base
`CadreLineStyle`	`solid`	Style de trait de l'encadrement du motif
`CadreWidth`	`thicklines`	Épaisseur de trait de l'encadrement du motif
`CadreColor`	`gray`	Couleur de trait de l'encadrement du motif
`CadreStrokeOpacity`	`0.75`	Opacité du trait de l'encadrement du motif
`CadreFillStyle`	`none`	Style de remplissage de l'encadrement du motif
`CadreFillColor`	`lightgray`	Couleur de remplissage de l'encadrement du motif
`CadreFillOpacity`	`0.5`	Opacité du remplissage de l'encadrement du motif

TABLE 3.5 – Variables globales relatives à l'encadrement du motif de base du pavage

3.3.3 Arrière-plan du pavage

`Cristallo.mac` dispose également d'un jeu de variable globales permettant de modifier l'apparence de l'arrière plan d'un pavage. Ces variables sont les même que celles mentionnées dans les chapitres précédents (à l'exception de quelques unes qui n'ont plus de sens dans le cadre de pavage) et s'utilisent

de la même manière. Le tableau qui suit nous rappelle quelles sont ces variables qui permettent la modification des attributs de l'arrière-plan d'un pavage.

`BackgroundFillStyle`	`full`	Style de remplissage de l'arrière-plan du pavage
`BackgroundFillColor`	`lightpink`	Couleur de remplissage de l'arrière-plan du pavage
`BackgroundFillOpacity`	`1`	Opacité du remplissage de l'arrière-plan du pavage

TABLE 3.6 – Variables globales relatives aux attributs de l'arrière-plan d'un pavage

3.3.4 Centres de rotations et axes de réflexions et de glissements

Jusqu'ici, les variables globales énumérées étaient commune à chaque type de pavage périodique. Elles ne dépendaient pas de la nature des transformations composant leur groupe de symétrie. Dans cette section, nous allons donner une liste des variables globales permettant d'afficher les centres de rotation et les axes de symétries ou glissements figurant dans les groupes de symétrie de ces pavages. Pour connaître les types de transformations présente dans un pavage donné, on se référera au tableau 3.1.

Rotations

Les pavages périodiques du plan euclidien font intervenir quatre types de rotation dont les angles sont respectivement π, $\frac{2\pi}{3}$, $\frac{\pi}{2}$ et $\frac{\pi}{3}$. Les variables globales permettant l'affichage et la modification des attributs des centres de ces rotations sont toutes similaire. Par exemple, les booléens permettant l'affichage de ces centres sont respectivement `AfficherCentres1`, `AfficherCentres2`, `AfficherCentres3` et `AfficherCentres4`. Ces variables ont un nom qui se termine par un nombre qui permet d'identifier le type de rotation : `1` correspond aux rotations d'angle π, `2` correspond aux rotations d'angle $\frac{2\pi}{3}$, etc... Le tableau 3.7 fait l'inventaire de toutes ces variables globales.

`AfficherCentres1`	`0`	Affichage des centres de symétries
`CtrDotStyle1`	`diamond`	Style des centres de symétries
`CtrDotSize1`	`2+2*i`	Taille des centres de symétries
`CtrDotScale1`	`1+i`	Facteurs d'échelle des centres de symétries
`CtrDotAngle1`	`0`	Angle de rotation des centres de symétries
`CtrColor1`	`black`	Couleur des centres de symétries
`CtrFillColor1`	`white`	Couleur de remplissage des centres de symétries
`AfficherCentres2`	`0`	Affichage des centres de rotations d'angle $2\pi/3$
`CtrDotStyle2`	`triangle'`	Style des centres de rotations d'angle $2\pi/3$
`CtrDotSize2`	`2+2*i`	Taille des centres de rotations d'angle $2\pi/3$
`CtrDotScale2`	`1+i`	Facteurs d'échelle des centres de rotations d'angle $2\pi/3$
`CtrDotAngle2`	`0`	Angle de rotation des centres de rotations d'angle $2\pi/3$
`CtrColor2`	`lime`	Couleur des centres de rotations d'angle $2\pi/3$
`CtrFillColor2`	`white`	Couleur de remplissage des centres de rotations d'angle $2\pi/3$
`AfficherCentres3`	`0`	Affichage des centres de rotations d'angle $\pi/2$
`CtrDotStyle3`	`square'`	Style des centres de rotations d'angle $\pi/2$
`CtrDotSize3`	`2+2*i`	Taille des centres de rotations d'angle $\pi/2$
`CtrDotScale3`	`1+i`	Facteurs d'échelle des centres de rotations d'angle $\pi/2$
`CtrDotAngle3`	`0`	Angle de rotation des centres de rotations d'angle $\pi/2$
`CtrColor3`	`sienna`	Couleur des centres de rotations d'angle $\pi/2$
`CtrFillColor3`	`white`	Couleur de remplissage des centres de rotations d'angle $\pi/2$

AfficherCentres4	0	Affichage des centres de rotations d'angle $\pi/3$
CtrDotStyle4	dot	Style des centres de rotations d'angle $\pi/3$
CtrDotSize4	2+2*i	Taille des centres de rotations d'angle $\pi/3$
CtrDotScale4	1+i	Facteurs d'échelle des centres de rotations d'angle $\pi/3$
CtrDotAngle4	0	Angle de rotation des centres de rotations d'angle $\pi/3$
CtrColor4	purple	Couleur des centres de rotations d'angle $\pi/3$
CtrFillColor4	white	Couleur de remplissage des centres de rotations d'angle $\pi/3$

TABLE 3.7 – Variables globales relatives à l'affichage et à la modification des attributs des centres de rotations des pavages périodiques

Réflexions et glissements

On peut également vouloir afficher les axes des symétrise ou des glissements qui interviennent dans la création d'un pavage. Pour cela, on affectera la valeur 1 aux booléens correspondant, à savoir `AfficherAxesSym` et `AfficherAxesGliss` respectivement. Les attributs de ces objets sont modifiables par l'intermédiaire des variables listées dans le tableau 3.8.

AfficherAxesSym	0	Booléen permettant l'affichage des axes de symétrie
AxesSymLineStyle	solid	Style de trait des axes de symétrie
AxesSymWidth	thinlines	Épaisseur de trait des axes de symétrie
AxesSymColor	crimson	Couleur de trait des axes de symétrie
AxesSymStrokeOpacity	1	Opacité du trait des axes de symétrie
AfficherAxesGliss	0	Booléen permettant l'affichage des axes de glissements
AxesGlissLineStyle	solid	Style de trait des axes de glissements
AxesGlissWidth	thinlines	Épaisseur de trait des axes de glissements
AxesGlissColor	dimgray	Couleur de trait des axes de glissements
AxesGlissStrokeOpacity	1	Opacité du trait des axes de glissements

TABLE 3.8 – Variables globales relatives à l'affichage et à la modification des attributs des axes de symétries et de glissements d'un pavage périodique

3.3.5 Ordre d'affichage des éléments d'un pavages périodique

Pour les pavages périodiques les plus compliqués, il peut y avoir de nombreux éléments à afficher. Par exemple, un pavage de type p6m fait intervenir beaucoup de transformations différentes : des symétries, des glissements et trois types de rotations. Ainsi, si l'on veut afficher tous les axes et les centres de rotations, le pavage risque d'être chargé. Les motifs du pavages peuvent ainsi se trouver masqués par ces éléments si `DrawPavPeriodique` les dessine avant de tracer les axes et les centres de rotations. Dans ces conditions, il peut être intéressant de modifier l'ordre d'affichage des différents éléments composant un pavage périodique. Pour cela, `Cristallo.mac` met à disposition la variable globale `PavPeriodiqueOrdreAffichage` qui contient une liste de nombres permettant de déterminer cet ordre d'affichage. Cette variable est analogue aux variables `FriseOrdreAffichage` et `RosaceOrdreAffichage` ; on utilisera donc un jeu de quelques constantes pour déterminer facilement l'ordre dans lequel `DrawPavPeriodique` va dessiner les différents éléments. Voici donc les constantes qui vont entrer en jeu dans la définition de `PavPeriodiqueOrdreAffichage` :

- `axesgliss` : axes des glissements,
- `axessym` : axes des symétries,
- `background` : arrière-plan du pavage,
- `cadre` : encadrement du motif de base,
- `centres1` : centres de symétries,
- `centres2` : centres des rotations d'angle $2\pi/3$,
- `centres3` : centres des rotations d'angle $\pi/2$,
- `centres4` : centres des rotations d'angle $\pi/3$,

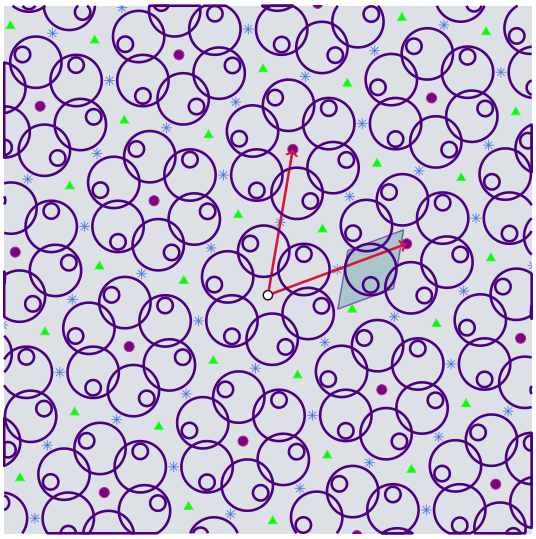
- `motif` : motifs du pavage,
- `ptref` : point de référence du pavage,
- `reseau` : réseau du pavage,
- `vecteur1` : premier vecteur générateur du pavage,
- `vecteur2` : second vecteur générateur du pavage.

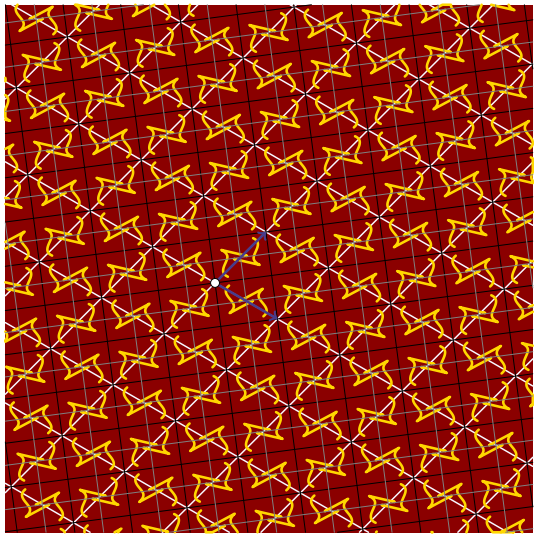
Par défaut, `PavPeriodiqueOrdreAffichage` est définie par la liste suivante [1] :

```
[background, reseau, cadre, motif, axessym, axesgliss,
    centres1, centres2, centres3, centres4, vecteur1, vecteur2, ptref]
```

Le tableau de la section 3.5 donne un récapitulatif complet des variables globales entrant en jeu dans la création d'un pavage périodique du plan euclidien.

3.4 Exemples





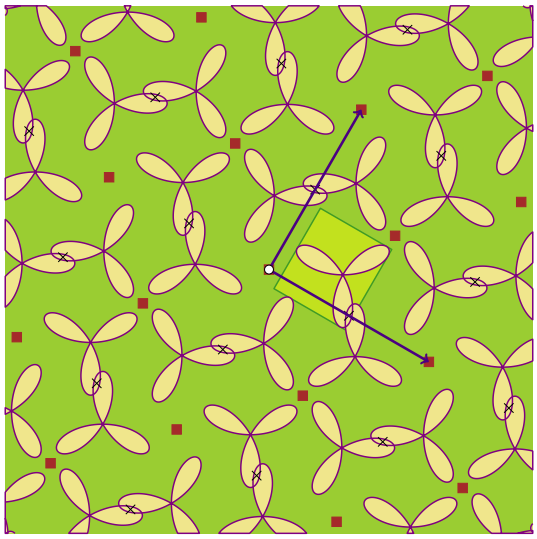
Code (Exemple 1)

```
[
Marges(0, 0, 0, 0),
PtRef:=-0.25*i, Vecteur1:=1.35+0.5*i,
Motif:=[Get(Cercle(1, 0.25)), jump, Get(Cercle(1-0.15*i, 0.075)), jump],
MotifLineStyle:=solid, MotifWidth:=Thicklines, MotifColor:=indigo,
BackgroundFillStyle:=full, BackgroundFillColor:=Light(lightslategray, 0.75),
AfficherPtRef:=1,
AfficherVecteur1:=1,
AfficherVecteur2:=1,
AfficherCentres1:=1, CtrDotStyle1:=asterisk, CtrDotSize1:=3+3*i,
AfficherCentres2:=1, CtrDotSize2:=3+3*i,
AfficherCentres4:=1, CtrDotSize4:=3+3*i,
EncadrerMotif:=1,
CadreLineStyle:=solid, CadreWidth:=thicklines, CadreColor:=darkslateblue,
CadreFillStyle:=full, CadreFillColor:=cadetblue, CadreFillOpacity:=0.4,
PavPeriodiqueOrdreAffichage:=[background, cadre, centres1, centres2,
        centres4, vecteur1, vecteur2, motif, ptref],
DrawPavPeriodique("p6")
]
```

1. Si un élément ne figure pas dans la liste, il ne sera pas affiché, même si le booléen correspondant à son affichage est égal à 1

Code (Exemple 2)

```
[
Marges(0, 0, 0, 0),
LineCap:=round,
PtRef:=-1.5-0.4*i,
Vecteur1:=2*exp(-i*pi/6), Vecteur2:=2*exp(i*pi/4),
AfficherPtRef:=1, AfficherVecteur1:=1, AfficherVecteur2:=1,
VecteurColor1:=darkslateblue, VecteurColor2:=darkslateblue,
$L:=[0.67062370898202+0.35571859637275*i,0.91788840829394+0.49448074982501*i,
        0.37435138830915+0.20470197964459*i,0.43335385224781+0.40269087115302*i,
        0.30348043842241+0.94105794653296*i,0.74630511016584+0.6084428417962*i],
Motif:=Get(Spline(L)),
MotifLineStyle:=solid, MotifWidth:=Thicklines, MotifColor:=gold,
BackgroundFillColor:=darkred,
AfficherReseau:=1,
ReseauLineStyle:=solid, ReseauWidth:=thicklines, ReseauColor:=ghostwhite,
AfficherAxesSym:=1,
AxesSymLineStyle:=solid, AxesSymWidth:=thinlines, AxesSymColor:=black,
AfficherAxesGliss:=1,
AxesGlissLineStyle:=solid, AxesGlissWidth:=thinlines, AxesGlissColor:=gray,
DrawPavPeriodique("cmm")
]
```



Code (Exemple 3)

```
[
Vecteur1:=3.5*exp(-i*pi/6),
AfficherPtRef:=1,
AfficherVecteur1:=1, VecteurColor1:=indigo,
AfficherVecteur2:=1, VecteurColor2:=indigo,
Motif:=Seq(1.4-0.1*i+cos($t)*sin(3*t)+i*sin(t)*sin(3*t), t, 0, pi, pi/180),
MotifWidth:=thicklines, MotifColor:=purple,
MotifFillStyle:=full, MotifFillColor:=khaki,
BackgroundFillColor:=yellowgreen,
AfficherCentres1:=1, AfficherCentres3:=1,
CtrDotStyle1:=times, CtrDotScale1:=2+2*i, CtrColor1:=black,
CtrDotScale3:=2+2*i, CtrColor3:=brown,
EncadrerMotif:=1,
CadreColor:=forestgreen,
CadreFillColor:=yellow, CadreFillOpacity:=0.4,
DrawPavPeriodique("p4")
]
```

3.5 Tableau récapitulatif des variables globales

Booléens	AfficherAxesGliss	0	Affichage des axes de glissements
	AfficherAxesSym	0	Affichage des axes de symétries
	AfficherCentres1	0	Affichage des centres de symétries
	AfficherCentres2	0	Affichage des centres de rotations d'angle $2\pi/3$
	AfficherCentres3	0	Affichage des centres de rotations d'angle $\pi/2$
	AfficherCentres4	0	Affichage des centres de rotations d'angle $\pi/3$
	AfficherPtRef	0	Affichage du point de référence du pavage
	AfficherReseau	0	Affichage du réseau du pavage
	AfficherVecteur1	0	Affichage du 1$^{\text{er}}$ vecteur générateur
	AfficherVecteur2	0	Affichage du 2$^{\text{nd}}$ vecteur générateur
	EncadrerMotif	0	Encadrement du motif de base
	MotifFerme	0	Motif fermé ou non
Lignes	AxesGlissLineStyle	solid	Style de ligne des axes de glissements
	AxesSymLineStyle	solid	Style de ligne des axes de symétries
	CadreLineStyle	solid	Style de ligne de l'encadrement du motif
	MotifLineStyle	solid	Style de ligne des motifs
	ReseauLineStyle	solid	Style de ligne du réseau
	VecteurLineStyle1	solid	Style de ligne du 1$^{\text{er}}$ vecteur générateur
	VecteurLineStyle2	solid	Style de ligne du 2$^{\text{nd}}$ vecteur générateur
	AxesGlissWidth	thinlines	Épaisseur de trait des axes de glissements
	AxesSymWidth	thinlines	Épaisseur de trait des axes de symétries
	CadreWidth	thicklines	Épaisseur de trait de l'encadrement du motif
	MotifWidth	thicklines	Épaisseur de trait des motifs
	ReseauWidth	thinlines	Épaisseur de trait du réseau
	VecteurWidth1	thicklines	Épaisseur de trait du 1$^{\text{er}}$ vecteur générateur
	VecteurWidth2	thicklines	Épaisseur de trait du 2$^{\text{nd}}$ vecteur générateur
	AxesGlissColor	dimgray	Couleur de trait des axes de glissements
	AxesSymColor	crimson	Couleur de trait des axes de symétries
	CadreColor	gray	Couleur de trait de l'encadrement du motif
	MotifColor	darkred	Couleur de trait des motifs
	ReseauColor	gold	Couleur de trait du réseau
	VecteurColor1	dimgray	Couleur de trait du 1$^{\text{er}}$ vecteur générateur
	VecteurColor2	dimgray	Couleur de trait du 2$^{\text{nd}}$ vecteur générateur
	AxesGlissStrokeOpacity	1	Opacité du trait des axes de glissements
	AxesSymStrokeOpacity	1	Opacité du trait des axes de symétries
	CadreStrokeOpacity	0.75	Opacité du trait de l'encadrement du motif
	MotifStrokeOpacity	1	Opacité du trait des motifs
	ReseauStrokeOpacity	1	Opacité du trait du réseau
	VecteurStrokeOpacity1	1	Opacité du trait du 1$^{\text{er}}$ vecteur générateur
	VecteurStrokeOpacity2	1	Opacité du trait du 2$^{\text{nd}}$ vecteur générateur
Remplissages	BackgroundFillStyle	full	Style de remplissage de l'arrière-plan
	CadreFillStyle	none	Style de remplissage de l'encadrement
	MotifFillStyle	full	Style de remplissage du motif
	BackgroundFillColor	lightpink	Couleur de remplissage de l'arrière-plan
	CadreFillColor	lightgray	Couleur de remplissage de l'encadrement
	MotifFillColor	lightgray	Couleur de remplissage du motif
	BackgroundFillOpacity	1	Opacité du remplissage de l'arrière-plan
	CadreFillOpacity	0.5	Opacité du remplissage de l'encadrement
	MotifFillOpacity	1	Opacité du remplissage du motif

Points	CtrDotStyle1	diamond	Style des centres de symétries
	CtrDotStyle2	triangle'	Style des centres de rotations d'angle $2\pi/3$
	CtrDotStyle3	square'	Style des centres de rotations d'angle $\pi/2$
	CtrDotStyle4	dot	Style des centres de rotations d'angle $\pi/3$
	PtRefDotStyle	dot	Style du point de référence
	CtrDotSize1	2+2*i	Taille des centres de symétries
	CtrDotSize2	2+2*i	Taille des centre de rotations d'angle $2\pi/3$
	CtrDotSize3	2+2*i	Taille des centre de rotations d'angle $\pi/2$
	CtrDotSize4	2+2*i	Taille des centre de rotations d'angle $\pi/3$
	PtRefDotSize	2+2*i	Taille du point de référence
	CtrDotScale1	1+i	Facteurs d'échelle des centres de symétries
	CtrDotScale2	1+i	Facteurs d'échelle des centres $(2\pi/3)$
	CtrDotScale3	1+i	Facteurs d'échelle des centres $(\pi/2)$
	CtrDotScale4	1+i	Facteurs d'échelle des centres $(\pi/3)$
	PtRefDotScale	1+i	Facteurs d'échelle du point de référence
	CtrDotAngle1	0	Angle de rotation des centres de symétries
	CtrDotAngle2	0	Angle de rotation des centres $(2\pi/3)$
	CtrDotAngle3	0	Angle de rotation des centres $(\pi/2)$
	CtrDotAngle4	0	Angle de rotation des centres $(\pi/3)$
	PtRefDotAngle	0	Angle de rotation du point de référence
	CtrColor1	black	Couleur des centres de symétries
	CtrColor2	lime	Couleur des centres de rotations d'angle $2\pi/3$
	CtrColor3	sienna	Couleur des centres de rotations d'angle $\pi/2$
	CtrColor4	purple	Couleur des centres de rotations d'angle $\pi/3$
	PtRefColor	black	Couleur du point de référence
	CtrFillColor1	white	Couleur de remplissage des centres de symétries
	CtrFillColor2	white	Couleur de remplissage des centres $(2\pi/3)$
	CtrFillColor3	white	Couleur de remplissage des centres $(\pi/2)$
	CtrFillColor4	white	Couleur de remplissage des centres $(\pi/3)$
	PtRefFillColor	white	Couleur de remplissage du point de référence
Autres variables	PtRef	0	Point de référence du pavage
	Vecteur1	1	Affixe du premier vecteur générateur
	Vecteur2	i	Affixe du second vecteur générateur
	Motif	[...]	Motif de base du pavage
	MotifArrondi	0	Arrondi des angles du motif de base
	PavPeriodiqueOrdreAffichage	[...]	Ordre d'affichage des éléments de la rosace

Chapitre 4

Pavages du plan euclidien par des polygones réguliers

Dans le chapitre précédent, nous avons vue en détail la manière de créer des pavages périodiques avec le fichier `Cristallo.mac` de *TEXgraph*. Dans ce chapitre, nous allons voir comment paver le plan euclidien avec des polygones régulier. Ces pavages sont en fait des cas particulier de pavages périodiques dont les motifs de base sont des ensembles de quelques polygones réguliers. Dans un premier temps, nous allons revenir sur les différents types de pavages par des polygones réguliers qu'il est possible de réaliser avec `Cristallo.mac`. Ensuite, nous aborderons la syntaxe de la macros `DrawPavPolygone` permettant de réaliser de tels pavages. Encore une fois, cette macros va reposer sur un grand nombres de variables globales : nous consacrerons donc un paragraphe énumérant l'ensemble de ces variables tout en mentionnant leur utilité. Enfin, quelques exemples de création de pavage par des polygone seront donnés afin de permettre à l'utilisateur de mieux se familiariser avec la macro `DrawPavPolygone`.

4.1 Pavage du plan par des polygones réguliers

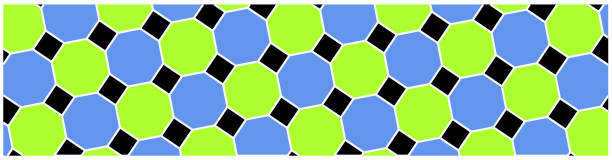
Comme son nom l'indique, un pavage du plan euclidien par des polygone régulier est un pavage périodique constitué uniquement de polygones réguliers. Autrement dit, Un pavage par des polygones régulier est un pavage périodique (voir le chapitre 3) dont le motif de base n'est autre qu'un ensemble de quelques polygones réguliers qui s'agencent correctement lors de la création du pavage. Par « agencement correct » des polygones, on veut dire que les polygones voisins du pavages doivent posséder un côté en commun. Ainsi, chaque sommet d'un polygone doit être confondu avec les sommets d'autres polygones. Le fait que ce type de pavages soit périodique implique aussi que l'agencement des polygones autour d'un sommet doit partout être le même. C'est d'ailleurs cette caractéristiques qui va nous permettre de nommer nos pavages. Il est important de noter que les pavages mentionnés dans ce chapitre sont tous des pavages périodiques. On verra dans le chapitre suivant quelques exemples de pavages du plan par des polygone réguliers qui ne sont pas périodiques. C'est la cas notamment des pavages « Kenyon's non FLC » et « Square-triangle ».

Il existe au total 11 types différents de pavages du plan par des polygones réguliers. Ces pavages ne font intervenir que 5 types de polygones réguliers : des triangles équilatéraux, des carrés, des hexagones, des octogones et des dodécagones. Ces pavages seront nommés suivant les polygones les constituant et leur agencement autour d'un sommet. Ainsi, le pavage $(6, 4, 3, 4)$ désigne un pavage du plan par des hexagones, des carrés et des triangles. Ces polygones sont disposés de la même manière autour de chaque sommet : on trouve d'abord un hexagone régulier, puis un carré, puis un triangle équilatéral et enfin un carré.

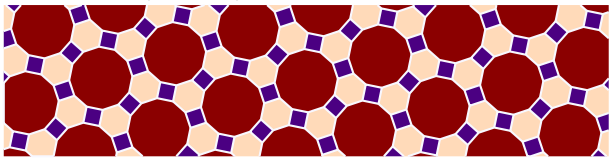
Les dessins suivants sont des représentations de chacun de ces pavages :

Type 1 : $(6,6,6)$

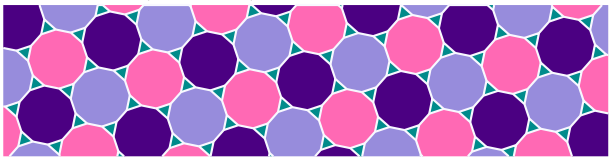
Type 2 : $(8,8,4)$



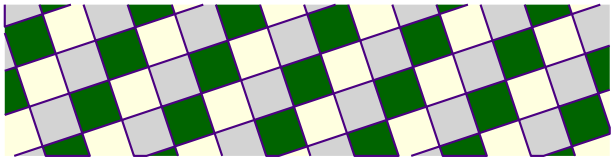
Type 3 : $(12,6,4)$



Type 4 : $(12,12,3)$

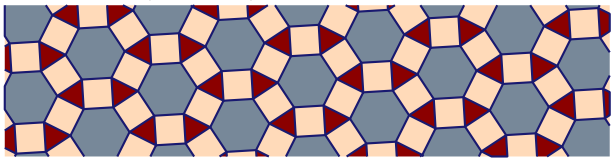


Type 5 : $(4,4,4,4)$



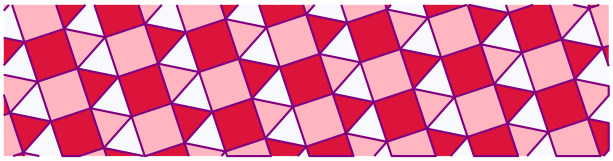
Type 6 : $(6,3,6,3)$

Type 7 : $(6,4,3,4)$



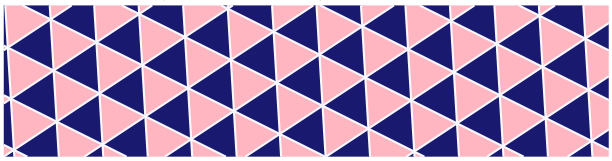
Type 8 : $(4,3,4,3,3)$

Type 9 : $(4,4,3,3,3)$



Type 10 : $(6,3,3,3,3)$

Type 11 : $(3,3,3,3,3,3)$



4.2 Création de pavages du plan par des polygones réguliers – Macro `DrawPavPolygone`

Le fichier `Cristallo.mac` dispose d'une macro permettant de créer rapidement et facilement des pavages par des polygones réguliers. Cette macro nommée `DrawPavPolygone` possède une syntaxe analogue aux macros `DrawFrise` ou `DrawPavPeriodique` et permet ainsi la création de chacun des 11 pavages énuméré dans la section 4.1. Sa syntaxe est la suivante :

```
DrawPavPolygone(<Type>, [Options])
```

Cette syntaxe est malgré tout légèrement différentes de celle des macros `DrawFrise` ou `DrawPavPeriodique`. Le second argument, optionnel, est toujours une liste d'affectations de variables globales permettant de modifier l'apparence du pavage en cours de création. En revanche, le premier paramètre n'est plus une chaîne de caractère mais un entier compris entre 1 et 11 ou une liste d'entiers.

Si le paramètre `type` est un entier, alors celui-ci doit être compris entre 1 et 11 et désigne ainsi un des 11 pavages énumérés plus tôt.
Si `type` est une liste d'entier, alors celle-ci correspond à l'agencement des polygones autour d'un sommet comme on l'a expliqué dans la section précédente. Ainsi, `type=8` ou `type=[4, 3, 4, 3, 3]` désignent le même pavage par des polygones réguliers, à savoir le pavage $(4, 3, 4, 3, 3)$.
Le tableau suivant donne les correspondance entre les deux manières différentes de désigner un pavage dans la macro `DrawPavPolygone`.

Forme 1	*Forme 2*
1	`[6, 6, 6]`
2	`[8, 8, 4]`
3	`[12, 6, 4]`
4	`[12, 12, 3]`
5	`[4, 4, 4, 4]`
6	`[6, 3, 6, 3]`
7	`[6, 4, 3, 4]`
8	`[4, 3, 4, 3, 3]`
9	`[4, 4, 3, 3, 3]`
10	`[6, 3, 3, 3, 3]`
11	`[3, 3, 3, 3, 3, 3]`

TABLE 4.1 – Correspondances entre les deux formes possibles du paramètre `type`

4.3 Variables globales

De la même manière que pour les frises périodiques, les rosaces ou les pavages périodiques du plan, l'apparence d'un pavage par des polygones régulier va se faire par le biais de quelques variables globales. Pour la plupart, celles-ci sont des variables de modification d'attributs de lignes ou de remplissage mais quelques unes d'entres-elles sont de nature différente et permettent plutôt un « ajustement spatial » du dessin que l'on veut créer. Nous allons dans ce paragraphe énumérer toutes ces variables globale et en donner l'utilité ainsi que leur valeur par défaut.

4.3.1 Ajustement de la position des polygones

`Cristallo.mac` utilise deux variables globales pour positionner convenablement les polygones constituant un pavage du plan par des polygones réguliers.

La première est la variable `PtRef` qui permet de définir le centre Ω du premier polygone dans la liste définissant son nom. Ainsi, si l'on veut dessiner le pavage $(6, 4, 3, 4)$, la valeur stockée dans `PtRef` sera l'affixe du centre d'un hexagone.
La variable `Vecteur1` permet quant à elle de positionner un des sommets de ce polygone. Cette variable contient l'affixe d'un vecteur $\vec{u}$ défini de telle sorte que le point $\Omega + \vec{u}$ soit un sommet de ce premier polygone. La figure 4.1 permet d'illustrer la signification de ces variables globales.

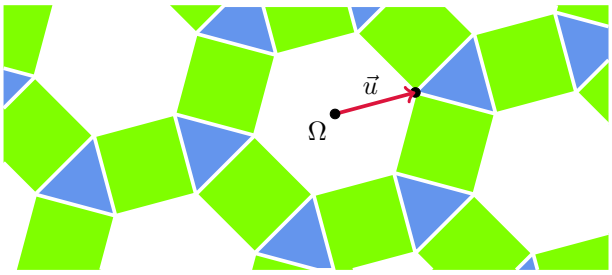


FIGURE 4.1 – Point de référence du pavage et sommet du premier polygone

Ces variables sont regroupées dans le tableau 4.2 qui suit avec leur valeur par défaut.
Il est aussi possible d'afficher sur le pavage en cours le point de référence et le vecteur ayant servi à positionner le premier sommet. Pour cela, on utilisera les booléens `AfficherPtRef` et `AfficherVecteur1` pour décider de leur affichage. Les attributs de ces objets sont alors modifiables à travers quelques variables globales qui sont présentées dans le tableau 4.2.

PtRef	0	Centre du polygone de référence du pavage
Vecteur1	1	Vecteur de définition d'un sommet du polygone de référence
AfficherPtRef	0	Booléen permettant l'affichage du point de référence
PtRefDotStyle	dotcircle	Style du point de référence
PtRefDotSize	2+2*i	Taille du point de référence
PtRefDotScale	1+i	Facteurs d'échelle du point de référence
PtrefDotAngle	0	Angle de rotation du point de référence
PtRefColor	black	Couleur du point de référence
PtRefFillColor	white	Couleur de remplissage du point de référence
AfficherVecteur1	0	Booléen permettant l'affichage du vecteur $\vec{u}$
VecteurLineStyle1	solid	Style de trait du vecteur $\vec{u}$
VecteurWidth1	Thicklines	Épaisseur de trait du vecteur $\vec{u}$
VecteurColor1	crimson	Couleur de trait du vecteur $\vec{u}$
VecteurStrokeOpacity1	1	Opacité du trait du vecteur $\vec{u}$

TABLE 4.2 – Variables globales relatives au point de référence et au vecteur $\vec{u}$

4.3.2 Attributs des polygones

Les attributs des polygones de ces pavages sont une fois encore modifiables via quelques variables globales. Ainsi, on va pouvoir grâce à ces variables modifier l'apparence des traits des lignes polygonales. Pour cela, `Cristallo.mac` met à disposition 4 variables énumérées dans le tableau 4.3.

PolyLineStyle	solid	Style de trait des côtés des polygones
PolyWidth	Thicklines	Épaisseur de trait des côtés des polygones
PolyColor	black	Couleur des côtés des polygones
PolyStrokeOpacity	1	Opacité de trait des côtés des polygones

TABLE 4.3 – Variables globales relatives aux attributs des côtés des polygones

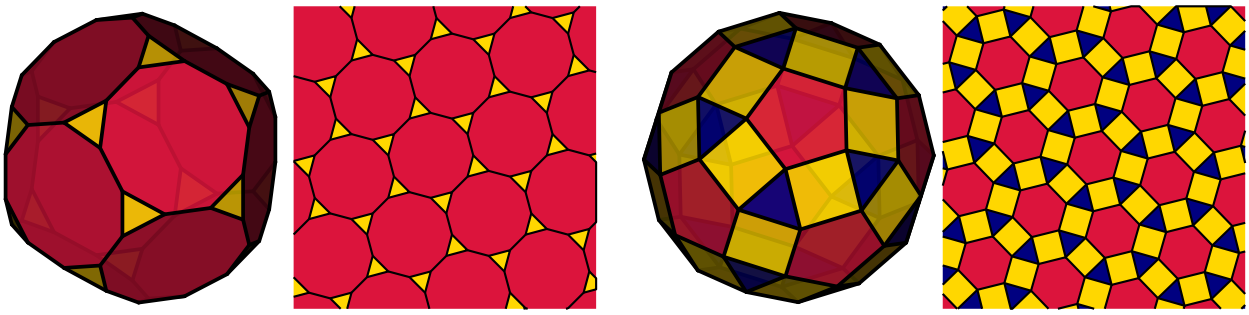
En ce qui concerne le remplissage des polygones, `DrawPavPolygone` utilise le théorème des 4 couleurs. Selon ce dernier, on peut utiliser au maximum 4 couleurs différentes pour remplir les polygones de telle sorte que deux polygones mitoyen ne soient pas de la même couleur. Ainsi, le fichier `Cristallo.mac` dispose d'un jeu de 12 variables permettant de modifier les attributs de remplissage des polygones suivant cette règle. Ces variables sont regroupées dans le tableau suivant.

PolyFillStyle1	full	Style de remplissage du premier type de polygones
PolyFillColor1	crimson	Couleur de remplissage du premier type de polygones
PolyFillOpacity1	1	Opacité de remplissage du premier type de polygones
PolyFillStyle2	full	Style de remplissage du second type de polygones
PolyFillColor2	gold	Couleur de remplissage du second type de polygones
PolyFillOpacity2	1	Opacité de remplissage du second type de polygones
PolyFillStyle3	full	Style de remplissage du troisième type de polygones
PolyFillColor3	navy	Couleur de remplissage du troisième type de polygones
PolyFillOpacity3	1	Opacité de remplissage du troisième type de polygones
PolyFillStyle4	full	Style de remplissage du quatrième type de polygones
PolyFillColor4	limegreen	Couleur de remplissage du quatrième type de polygones
PolyFillOpacity4	1	Opacité de remplissage du quatrième type de polygones

TABLE 4.4 – Variables globales relatives aux attributs de remplissage des polygones

4.3.3 Chiralité

Il est intéressant de faire le parallèle entre les pavages du plan euclidien par des polygones réguliers et les solides d'Archimède (voir le fichier `PolyedresII.mac`). En effet, on remarque par exemple des ressemblance entre certaines configuration de sommets de ces polyèdres et les configurations des pavages précédent. La figure 4.2 donne deux exemples de ces analogies.

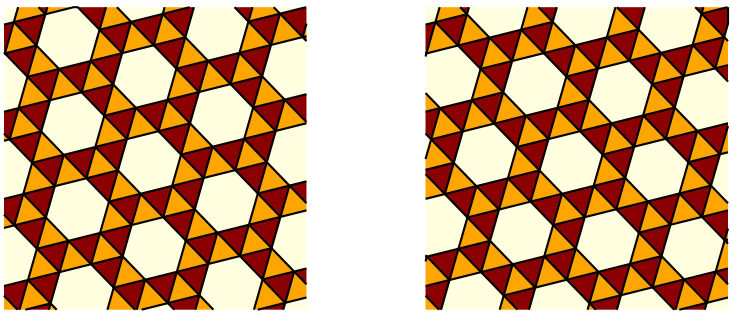


(a) Analogie entre le dodécaèdre tronqué et le pavage $(12, 12, 3)$

(b) Analogie entre le petit rhombicosidodécaèdre et le pavage $(6, 4, 3, 4)$

FIGURE 4.2 – Analogies entre les solides d'Archimède et les pavages du plan par des polygones réguliers

On peut alors lier les pavages $(4, 3, 4, 3, 3)$ et $(6, 3, 3, 3, 3)$ aux cube adouci et dodécaèdre adouci respectivement. Comme ces polyèdres, les pavages correspondant possèdent deux formes chirales (c'est à dire dont le symétrique par rapport à une droite ne peut pas être superposé au pavage initial). Il est ainsi possible de décider de dessiner un de ces deux pavages ou leur énantiomorphe à l'aide du booléen `Chiral` qui vaut 0 par défaut. La figure suivante et une illustration des résultats que l'on obtient avec les deux valeurs possibles de `Chiral`.



(a) `Chiral = 0`

(b) `Chiral = 1`

FIGURE 4.3 – Pavage $(6, 3, 3, 3, 3)$ et chiralité

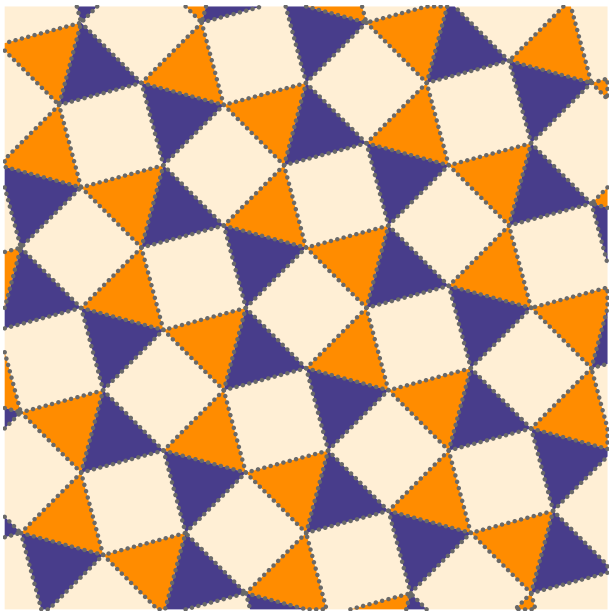
4.3.4 Ordre d'affichage des élément d'un pavage par des polygones réguliers

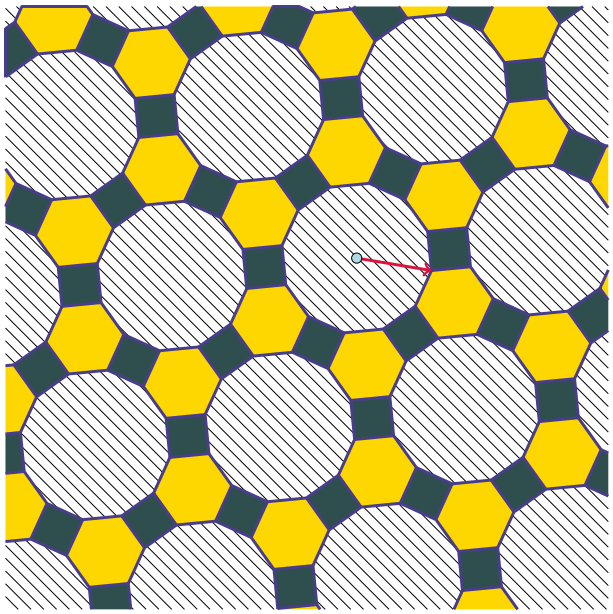
Bien que d'un intérêt plus limité comparé au pavages périodiques, il est possible de modifier l'ordre d'affichage des différents éléments composant un pavage du plan par des polygones régulier (c'est à dire les polygones, le point de référence et le vecteur $\vec{u}$). Pour cela, on utilise la variable globale `PavPolygoneOrdreAffichage` qui contient une liste de constantes permettant de définir cet ordre. Les constantes utilisables dans le cadre de ces pavages sont donc :

- `motif` : affichage des polygones,
- `ptref` : affichage du point de référence,
- `vecteur1` : affichage du vecteur $\vec{u}$.

Par défaut cette variable vaut `[motif, vecteur1, ptref]`.

4.4 Exemples





Code (Exemple 1)

```
[
Marges(0, 0, 0, 0),
Chiral:=1,
PolyLineStyle:=dotted, PolyWidth:=0.75*mm, PolyColor:=dimgray,
PolyFillColor1:=papayawhip,
PolyFillColor2:=darkorange,
PolyFillColor3:=darkslateblue,
PolyFillColor4:=darkred,
DrawPavPolygone(8)
]
```

Code (Exemple 2)

```
[
Marges(0, 0, 0, 0),
DrawPavPolygone([12, 6, 4],
        [
        PtRef:=1+i, Vecteur1:=1.5-0.25*i,
        AfficherPtRef:=1, AfficherVecteur1:=1,
        PtRefDotStyle:=dotcircle, PtRefDotSize:=2+2*i, PtRefFillColor:=lightblue,
        PolyLineStyle:=solid, PolyWidth:=Thicklines, PolyColor:=darkslateblue,
        PolyFillStyle1:=fdiag, PolyFillColor1:=black,
        PolyFillStyle2:=full, PolyFillColor2:=gold,
        PolyFillStyle3:=full, PolyFillColor3:=darkslategray
        ])
]
```

4.5 Tableau récapitulatif des variables globales

Booléens	AfficherPtRef	0	Affichage du point de référence du pavage
	AfficherVecteur1	0	Affichage du vecteur $\vec{u}$
	Chiral	0	Choix d'un pavage ou de son énantiomorphe

Lignes	`PolyLineStyle`	`solid`	Style de trait des côtés des polygones
	`VecteurLineStyle1`	`solid`	Style de trait du vecteur $\vec{u}$
	`PolyWidth`	`Thicklines`	Épaisseur de trait des côtés des polygones
	`VecteurWidth1`	`Thicklines`	Épaisseur de trait du vecteur $\vec{u}$
	`PolyColor`	`black`	Couleur de trait des côtés des polygones
	`VecteurColor1`	`crimson`	Couleur de trait du vecteur $\vec{u}$
	`PolyStrokeOpacity`	`1`	Opacité du trait des côtés des polygones
	`VecteurStrokeOpacity1`	`1`	Opacité de trait du vecteur $\vec{u}$
Remplissages	`PolyFillStyle1`	`full`	Style de remplissage du 1er type de polygones
	`PolyFillStyle2`	`full`	Style de remplissage du 2e type de polygones
	`PolyFillStyle3`	`full`	Style de remplissage du 3e type de polygones
	`PolyFillStyle4`	`full`	Style de remplissage du 4e type de polygones
	`PolyFillColor1`	`gold`	Couleur de remplissage du 1er type de polygones
	`PolyFillColor2`	`crimson`	Couleur de remplissage du 2e type de polygones
	`PolyFillColor3`	`navy`	Couleur de remplissage du 3e type de polygones
	`PolyFillColor4`	`limegreen`	Couleur de remplissage du 4e type de polygones
	`PolyFillOpacity1`	`1`	Opacité du remplissage du 1er type de polygones
	`PolyFillOpacity2`	`1`	Opacité du remplissage du 2e type de polygones
	`PolyFillOpacity3`	`1`	Opacité du remplissage du 3e type de polygones
	`PolyFillOpacity4`	`1`	Opacité du remplissage du 4e type de polygones
Points	`PtRefDotStyle`	`dotcircle`	Style du point de référence
	`PtRefDotSize`	`2+2*i`	Taille du point de référence
	`PtRefDotScale`	`1+i`	Facteurs d'échelle du point de référence
	`PtRefDotAngle`	`0`	Angle de rotation du point de référence
	`PtRefColor`	`black`	Couleur du point de référence
	`PtRefFillColor`	`white`	Couleur de remplissage du point de référence
	`PavPolygoneOrdreAffichage`	`[...]`	Ordre d'affichage des éléments du pavage

Chapitre 5

Pavages apériodiques

5.1 Pavages apériodiques

Le fichier `Cristallo.mac` contient également un grand nombre de macros permettant de dessiner de nombreux pavages apériodiques. Dans ce document, nous appellerons pavages apériodique un pavage du plan qui n'est pas périodique. Néanmoins, la signification du terme « apériodique » peut varier d'un ouvrage à l'autre.
Les pavages réalisables grâce à ce fichier sont tous[1] des pavages dits « par substitution »(ou « substitution tiling » en anglais). Cette terminologie désigne la manière dont sont construits ces pavages. Chacun d'entre eux sont composés de un ou plusieurs types de tuiles. Pour chacun de ces types de tuiles, on définit une règle de substitution. Une règle de substitution permet de décomposer un type de tuiles en un ensemble de plusieurs tuiles de base du pavage. La figure 5.1 donne les règles de substitution des tuiles du pavage Penrose kite-dart.

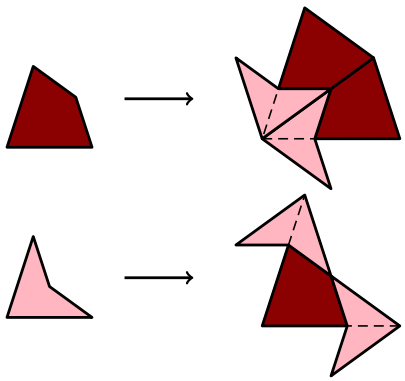


FIGURE 5.1 – Règles de substitution du pavage Penrose kite-dart

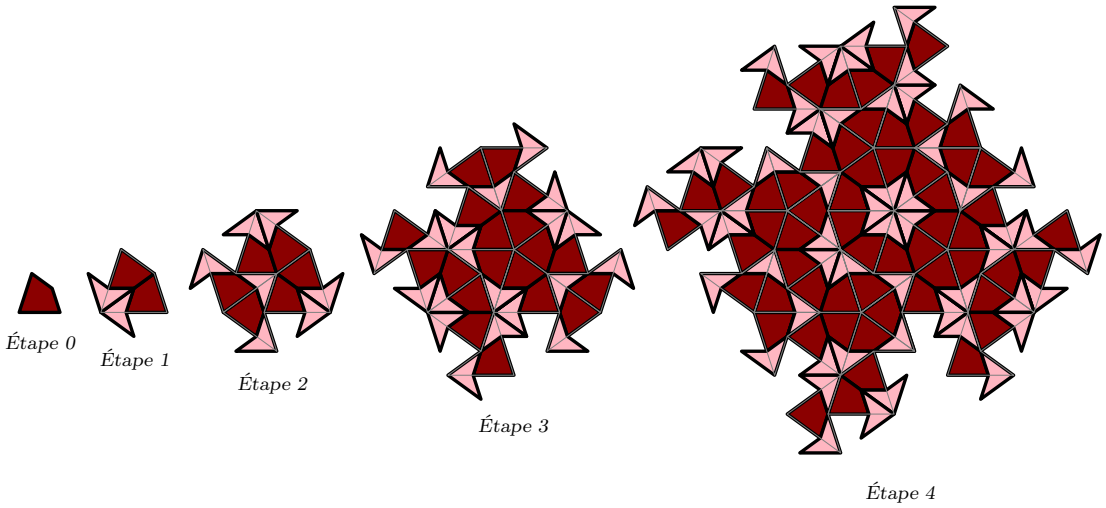


FIGURE 5.2 – Étapes de construction du pavage Penrose kite-dart

1. A l'exception des pavages de Truchet qui feront l'objet de la section 5.5

Une fois nos règles de substitution définies, on construit le pavage correspondant en suivant la démarche suivante :

1. On choisit une tuile servant de base à notre pavage (dans le cas du pavage Penrose kite-dart, une fléchette ou un cerf-volant).
2. On applique sur la tuile de base la règle de substitution correspondante. Cela produit un nouvel ensemble de tuiles de base du pavage.
3. On applique sur toutes les tuiles produites dans l'étape 2 les règles de substitution du pavage ce qui nous permet d'obtenir un nouvel ensemble de tuiles de base.
4. On réitère l'étape précédente jusqu'à obtenir un pavage de la taille souhaitée.

La figure 5.2 illustre ces différentes étapes de construction d'un pavage apériodique dans le cas du pavage Penrose kite-dart.

Enfin, nous terminons cette série de définitions par celle de facteur d'agrandissement (en anglais « inflation factor »). Le facteur d'agrandissement d'un pavage apériodique est un réel strictement supérieur à 1 qui quantifie l'expansion du pavage d'une étape à la suivante. Ainsi, plus ce facteur est grand, plus le pavage s'agrandit lors du passage à l'étape de substitution suivante.

5.2 Les pavages apériodiques du fichier `Cristallo.mac`

Le fichier `Cristallo.mac` met désormais à disposition un jeu de plus de 90 pavages apériodiques, du plus simple au plus compliqué. Le tableau suivant est une énumération de tous les pavages réalisables avec ce fichier. Une illustration de chacun de ces pavages est ensuite donné pour avoir un aperçu de chacun d'entre eux.

TABLE 5.1 – Pavages apériodiques réalisables avec `Cristallo.mac`

N°	Nom	Macro	Tuiles	φ	Fact.
1	Ammann A3	`AmmannA3`	3	1,62	0–10
2	Ammann A4	`AmmannA4`	2	2,41	0–6
3	Ammann–Beenker	`AmmannBeenker`	2	2,41	0–5
4	Ammann–Beenker rhomb triangle	`AmmannBeenkerRhombTriangle`	3	2,41	0–6
5	Ammann chair	`AmmannChair`	2	1,27	0–22
6	Armchair	`Armchair`	2	2,00	0–7
7	Binary	`Binary`	2	1,90	0–8
8	Example of canonical1	`Canonical1`	3	4,24	0–3
9	Example of canonical2	`Canonical2`	3	4,24	0–3
10	Chaim's cubic PV	`ChaimCubicPV`*	3	1,47	0–15
11	Chair	`Chair`	4	2,00	0–7
12	Chair variant	`Chair2`	4	3,00	0–4
13	Coloured golden triangle	`ColouredGoldenTriangle`	10	1,27	0–22
14	Cubic pinwheel	`CubicPinwheel`	6	1,21	0–27
15	Cyclotomic rhombs 7-fold	`CyclotomicRhombs7Fold`	6	2,80	0–5
16	Danzer's non FLC 5	`DanzerNonFLC5`*	5	1,19	0–32
17	Danzer's 7-fold	`Danzer7Fold1`	3	2,80	0–5
18	Danzer's 7-fold variant	`Danzer7Fold2`	3	2,25	0–6
19	Domino variant 1	`Domino1`	2	2,00	0–7
20	Domino variant 2	`Domino2`	2	2,00	0–7
21	Domino variant 3	`Domino3`	2	3,00	0–4
22	Equithirds	`Equithirds`	2	1,73	0–9
23	Fibonacci times Fibonacci	`FibonacciTimesFibonacci`	3	1,62	0–11

N°	Nom	Macro	Tuiles	φ	Fact.
24	Golden pinwheel	`GoldenPinwheel`	2	2,62	0–5
25	Golden triangle	`GoldenTriangle`	2	1,27	0–22
26	Goodman-Strauss 7-fold rhomb	`GoodmanStrauss7FoldRhomb`	3	3,80	0–4
27	Half-hex	`HalfHex`	6	2,00	0–7
28	Harriss's 4-fold rhomb	`Harriss4FoldRhomb`	6	3,41	0–4
29	Harriss's 9-fold rhomb	`Harriss9FoldRhomb`	4	3,88	0–4
30	Imbalanced orientations	`ImbalancedOrientations`	2	3,00	0–4
31	Kenyon's non FLC	`KenyonNonFLC`$^\bullet$	2	3,00	0–5
32	Kenyon $(1, 2, 1)$ polygon	`Kenyon121Polygon`	3	1,60	0–12
33	Kite-domino	`KiteDomino`	2	5,00	0–3
34	Limhex	`Limhex`	1	2,00	0–7
35	Lord	`Lord`	6	1,73	0–7
36	Maloney's 7-fold	`Maloney7Fold`	3	4,05	0–3
37	Octagonal 1225	`Octagonal1225`	2	2,41	0–6
38	Penrose kite-dart	`PenroseKiteDart`	2	1,62	0–9
39	Penrose pentagon boat star	`PenrosePentagonBoatStar`	6	3,60	0–5
40	Penrose rhomb	`PenroseRhomb`	2	1,62	0–8
41	Penrose triangle	`PenroseTriangle`	2	1,62	0–10
42	Pentomino	`Pentomino`	2	2,00	0–7
43	Pinwheel variant 1	`Pinwheel1`	2	2,24	0–6
44	Pinwheel variant 2	`Pinwheel2`	2	3,16	0–4
45	Pinwheel variant 3	`Pinwheel3`	2	3,61	0–4
46	Pinwheel variant 4	`Pinwheel4`	2	8,13	0–2
47	Pinwheel variant 5	`Pinwheel5`	2	8,06	0–2
48	Pinwheel 1-2	`Pinwheel12`	2	1,60	0–11
49	Pinwheel 2-1	`Pinwheel21`	2	2,06	0–7
50	Pinwheel 3-1	`Pinwheel31`	2	2,56	0–6
51	Priebe Frank non PV	`PriebeFrankNonPV`	4	2,30	0–7
52	Psychedelic Penrose variant 1	`PsychedelicPenrose`	4	1,62	0–10
53	Pythagoras 3-1	`Pythagoras31`	3	1,15	0–36
54	Pythagoras 3-2	`Pythagoras32`	3	1,21	0–26
55	Pythagoras 5-2	`Pythagoras52`	5	1,09	0–56
56	Pythia 3-1	`Pythia31`	3	2,32	0–6
57	Quartic pinwheel	`QuarticPinwheel`	4	1,44	0–13
58	Rhomb square oktagon	`RhombSquareOktagon`	3	3,41	0–4
59	Robinson triangle	`RobinsonTriangle`	4	1,62	0–10
60	Rorschach	`Rorschach`	4	2,73	0–5
61	Semi-detached house	`SemiDetachedHouse`	2	2,00	0–7
62	Semi-detached house squared	`SemiDetachedHouseSquared`	12	2,00	0–7
63	Sierpinski square	`SierpinskiSquare`	2	3,00	0–5
64	Sierpinski triangle	`SierpinskiTriangle`	2	2,00	0–7
65	Shield	`Shield`	4	1,93	0–7
66	Socolar	`Socolar`	3	3,73	0–4
67	Sphinx	`Sphinx`	2	2,00	0–7
68	Sphinx 9	`Sphinx9`	2	3,00	0–4
69	Sqrt6 triangles	`Sqrt6Triangles`	3	2,45	0–6
70	Square chair	`SquareChair`	4	2,00	0–7
71	Square triangle	`SquareTriangle`	5	3,73	0–4
72	Squeeze	`Squeeze`*	3	1,32	0–17
73	Squiral	`Squiral`*	2	3,00	0–4

N°	Nom	Macro	Tuiles	φ	Fact.
74	Tangram	`Tangram`	5	1,41	0–15
75	Tetris	`Tetris`	5	2,00	0–7
76	Tipi 3-1	`Tipi31`	3	1,47	0–14
77	Triangle duo variant 1	`TriangleDuo1`	2	1,41	0–14
78	Triangle duo variant 2	`TriangleDuo2`	2	1,41	0–14
79	Triangle duo variant 3	`TriangleDuo3`	2	1,41	0–14
80	Trihex	`Trihex`	2	2,00	0–7
81	Tritriangle	`Tritriangle`	3	1,41	0–15
82	Truchet	`Truchet`	4	—	—
83	Tuebingen triangle	`TuebingenTriangle`	4	1,62	0–10
84	Uberpinwheel	`Uberpinwheel`	4	8,06	0–2
85	Viper	`Viper`	2	3,00	0–4
86	Waltonchair	`Waltonchair`	2	1,41	0–15
87	Watanabe Ito Soma 8-fold	`WatanabeItoSoma8Fold`	2	3,41	0–4
88	Watanabe Ito Soma 12-fold variant 1	`WatanabeItoSoma12Fold1`	3	3,73	0–4
89	Watanabe Ito Soma 12-fold variant 2	`WatanabeItoSoma12Fold2`	3	3,73	0–4
90	Wheel tiling	`WheelTiling`	2	3,73	0–4

5.3 Création de pavages apériodiques

Comme on vient de le mentionné, `Cristallo.mac` permet désormais de créer plus de 90 pavages apériodiques différents. La démarche pour dessiner de tels dessins diffère des types de graphiques détaillés dans les chapitres précédents. En effet, pour créer un pavage apériodique, il faudra passer par trois macros. Une première déterminant la tuile « initiale » du pavage, c'est à dire la tuile servant de point de départ à la succession de substitutions (voir section précédente). La deuxième macro permettra, compte tenu de la tuile initiale, de déterminer toutes les tuiles du pavage après un nombre N d'étapes de substitutions. Enfin, une fois ces tuiles définies, une dernière macro devra être utilisée pour les affichées.
Il existe autant de macros de création de pavage (ou de création de tuile de base) qu'il y a de type de pavages apériodiques définis dans `Cristallo.mac`. Ces macros sont nommées par le nom du pavage [2] et les macros de construction de la tuile initiale ont un nom identique suffixé du mot `Tiles`. L'ensemble des pavages ainsi que des noms de macros correspondant figure dans le tableau 5.1.

5.3.1 Création de la tuile de base

Comme on vient de le voir, les tuiles de bases de chacun des pavages sont accessible par des macro dont le nom est de la forme *`NomPavage`*`Tiles`. Ces macros possède (presque) toute la syntaxe suivante :

```
NomPavageTiles(<Point 1>, <Point 2>, <Type de tuile>)
```

Les deux premiers arguments sont obligatoires : il s'agit de deux points permettant de définir la tuile voulue. Souvent, ces deux arguments désignent deux sommets consécutifs de la tuile, mais parfois (comme pour le pavage Shield) il s'agit du centre de la tuile et d'un de ses sommets. Le troisième argument, lui aussi obligatoire, permet à l'utilisateur de préciser quel type de tuile il désire choisir. Les valeurs de ce paramètre peuvent varier de 1 au nombre de types de tuiles composant le pavage. Ce nombre figure dans la quatrième colonne du tableau 5.1. Ces macros renvoie une liste de complexe

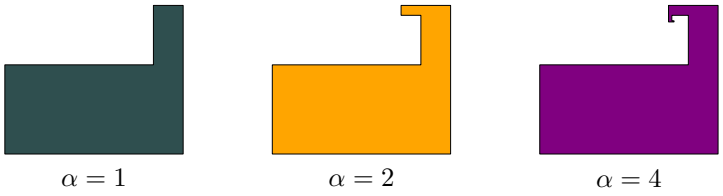
2. Les noms attribués aux pavages sont pour la plupart ceux définis sur le site d'Edmund Harriss et D. Frettlöh, *The Tiling Encyclopedia*, disponible à l'adresse http://tilings.math.uni-bielefeld.de/, site ayant servi de base pour l'élaboration de toutes ces macros.

définissant les contours de la tuile.

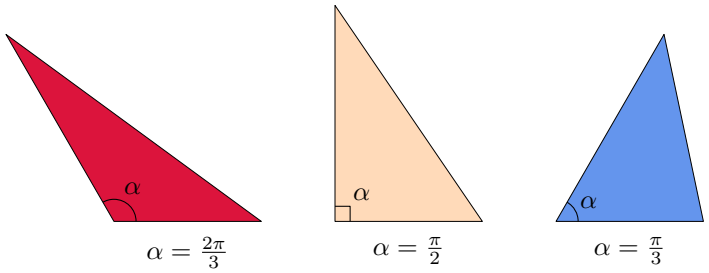
Toutes les macros de construction de tuiles de bases peuvent posséder cette syntaxe. Cependant, quelques unes d'entre elles possèdent un argument optionnel supplémentaire. Ces pavage sont marqué par le symbole $\star$ en exposant dans le tableau suivant.

```
NomPavageTiles(<Point 1>, <Point 2>, <Type de tuile>, [alpha])
```

Pour ces pavages, les tuiles de bases peuvent prendre une infinité de formes différentes, celle ci dépendant d'un paramètre `alpha`. Pour le pavage Squiral, ce paramètre est un entier permettant de définir l'enroulement de la tuile de base. Ce pavage est en fait un exemple montrant qu'un pavage apériodique peut être constitué de tuiles polygonales possédant un nombre infini de côtés. Pour les autres pavages possédant cette syntaxe, le paramètre `alpha` correspond à un angle α permettant de définir le triangle définissant la tuile initiale. La figure 5.3 illustre ce point.



(a) Paramètre `alpha` pour le pavage Squiral



(b) Paramètre `alpha` pour le pavage Chaim's cubic PV

FIGURE 5.3 – Illustration des valeurs possible du paramètre `alpha`

5.3.2 Création des tuiles du pavage

Une fois la tuile de base définie, on va pouvoir appliquer les règle de substitution afin de créer le pavage. Pour cela, on applique la macro de construction du pavage correspondant. Ces macros renvoient une liste de tuiles (c'est à dire de lignes polygonales séparées par des constante `jump`[3]) et possèdent la syntaxe suivante :

```
NomPavage(<Liste de tuiles>, <Facteur>)
```

Le premier paramètre, qui est obligatoire, est une liste de tuiles qui vont subir progressivement les décomposition issue des règles de substitutions. On attribuera souvent à ce paramètre la liste retournée par la macro `NomPavageTiles` mais on peut très bien fournir plus d'une tuile à cette macros comme en témoigne l'exemple 5.6b de la section 5.7.
Le second argument, également obligatoire, permet de déterminer le nombre d'étapes de substitutions

3. En réalité, les tuiles sont séparées par des constantes `jump` modifiées. A chaque type de tuile d'un pavage correspondra un `jump` différent (notés `jump1`, `jump2`, ...). Il s'agit simplement de la constante `jump` à laquelle on à ajouté une partie imaginaire.

que l'on va réaliser. Par exemple, si ce paramètre vaut 2, on va appliquer une première fois les règles de substitutions sur la (ou les) tuile(s) de bases (première étape de substitution), puis on réappliquera sur chacune des tuiles obtenues les règles de substitution du pavage (étape 2). Cette démarche est très bien illustrée sur la figure 5.2. Il est important de noter que selon le type de pavage, *TEXgraph* peut très vite saturer suivant le facteur que l'on à fourni à la macro. Cette saturation est étroitement liée au facteur d'agrandissement φ du pavage. Plus ce facteur est grand, plus la valeurs du paramètre `Facteur` devra être basse. Le tableau 5.1 donne pour chaque pavage son facteur d'agrandissement ainsi qu'un jeu de valeurs conseillées pour le paramètre `Facteur` (colonnes 5 et 6 respectivement). Bien sûr, ce sont bien des valeurs conseillées : l'utilisateur pourra très bien s'essayer à des valeurs dépassant les limites fournies dans le tableau mais le résultat n'est pas garanti !

Notons enfin la présence d'un troisième argument optionnel pour la macro relative au pavage Kenyon's non FLC (marqué par le symbole • dans le tableau 5.1). Ce paramètre est un irrationnel (en fait, un réel. L'irrationnalité du paramètre impose le caractère apériodique du pavage) correspondant à un décalage vertical de la troisième colonne de carré défini dans la règle de substitution (voir la figure 5.4).

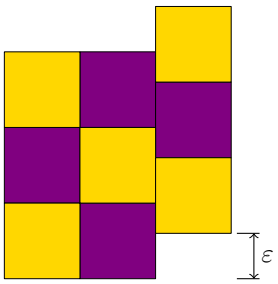


FIGURE 5.4 – Paramètre optionnel ε de la macro `KenyonNonFLC`

5.3.3 Dessin d'un pavage apériodique

Une fois Les tuiles du pavage définie grâce au macros précédemment détaillées, on va utiliser une troisième macro pour les dessiner. Cette macro se nomme `DrawTiles` et possède la syntaxe suivante :

```
DrawTiles(<Liste de tuiles>, [Options])
```

Le premier argument est obligatoire et est une liste de tuiles définies par les macros *`NomPavage`*`Tiles` et/ou *`NomPavage`*. Il doit en fait s'agir d'un groupe de ligne polygonales séparées par une constante `jump`*`k`*, où *`k`* désigne un entier compris entre 1 et 10.
Le deuxième argument à fournir à `DrawTiles` est optionnel et est une liste d'affectations de variables globales permettant de modifier l'apparence du pavage que l'on souhaite dessiner. On pourra alors modifier le remplissage de chaque type de tuile via plusieurs variables de la même manière que pour les pavages par des polygones réguliers. La macro `DrawTiles` pourra distinguer chaque type de tuile grâce aux constante `jump`*`k`* qui permettent justement de les différencier. Le détail des variables globales utiles dans la création de pavages apériodiques figure dans la section suivante.

5.3.4 Autres macros utiles

Le fichier `Cristallo.mac` contient plus de 90 pavages apériodiques avec des noms plus ou moins extravagants ! Il peut donc s'avérer difficile de tous les retenir. Pour contourner ce problème, des macros générales permettent d'accéder à ces pavages sans être obligé de connaître leur nom. Voici leur syntaxe :

```
PavAperiodiqueTiles(<Index/Nom>, <Point 1>, <Point 2>, <Type de tuile>, [alpha])
PavAperiodique(<Index/Nom>, <Liste de tuiles>, <Facteur>, [epsilon])
```

Le premier paramètre peut prendre deux formes : soit il s'agit d'un entier correspondant à un pavage (première colonne du tableau 5.1), soit il s'agit d'une chaîne de caractère contenant le nom de la macro du pavage que l'on souhaite dessiner. Les paramètres suivants ont les mêmes fonctions que celles décrites dans les paragraphes précédents. On notera que le paramètre optionnel `alpha` n'est disponible que pour les pavages Chaim's cubic PV, Danzer's non FLC 5, Squeeze et Squiral, et que le paramètre `epsilon` est utile uniquement pour le pavage Kenyon's non FLC.

Une dernière macro est présente dans `Cristallo.mac` afin de pouvoir utiliser le nom d'un pavage apériodique. Il s'agit de `PavAperiodiqueNames`. Cette dernière possède la syntaxe suivante :

```
PavAperiodiqueNames(<Index/Nom>, <Variable>, [Affichage])
```

Le premier paramètre est un entier ou une chaîne de caractère permettant l'identification du pavage. Cet argument est similaire au premier argument des macros `PavAperiodique` et `PavAperiodiqueTiles`. Le second argument est obligatoire et est un nom de variable (ou plutôt de macro) dans laquelle sera stockée la chaîne de caractère correspondant au nom du pavage. Le dernier argument est optionnel et peut prendre les valeurs 0 ou 1. Il permet de l'ouverture ou non d'une boîte de dialogue contenant le nom du pavage. Si cet argument vaut 0 (valeur par défaut), aucune fenêtre ne s'ouvrira.
On trouvera un exemple d'utilisation de cette macro dans la section 5.7.

5.4 Variables globales relatives aux pavages apériodiques

Comme pour les types de dessins détaillés dans les chapitres précédents, l'apparence d'un pavage apériodique pourra être modifié par le biais de variables globales présentes dans `Cristallo.mac`. Pour ce type de pavage, le nombre de ces variables est beaucoup plus restreint : elles se limitent au remplissage des tuiles ainsi qu'aux attributs de trait du pavage. La modification de ces variables peut se faire soit hors de la macro `DrawTiles`, soit en les insérant dans une listes fournie en deuxième argument de cette macro.

Pour modifier l'apparence des lignes composant un pavage apériodique, on dispose donc de quelques variables répertoriées dans le tableau 5.2.

`TilesLineStyle`	`solid`	Style de trait des côtés des tuiles
`TilesWidth`	`thinlines`	Épaisseur de trait des côtés des tuiles
`TilesColor`	`black`	Couleur des côtés des tuiles
`TilesStrokeOpacity`	`1`	Opacité de trait des côtés des tuiles

TABLE 5.2 – Variables globales relatives aux attributs des côtés des tuiles

Concernant les styles de remplissage des différents types de tuiles, il existe au total 30 variables globales permettant de les modifier. Ces variables sont regroupées dans le tableau suivant avec leur valeur par défaut.

TABLE 5.3 – Variables globales relatives aux attributs de remplissage des tuiles

`TilesFillStyle1`	`full`	Style de remplissage des tuiles de type 1
`TilesFillColor1`	`crimson`	Couleur de remplissage des tuiles de type 1
`TilesFillOpacity1`	`1`	Opacité de remplissage des tuiles de type 1
`TilesFillStyle2`	`full`	Style de remplissage des tuiles de type 2
`TilesFillColor2`	`gold`	Couleur de remplissage des tuiles de type 2
`TilesFillOpacity2`	`1`	Opacité de remplissage des tuiles de type 2
`TilesFillStyle3`	`full`	Style de remplissage des tuiles de type 3
`TilesFillColor3`	`navy`	Couleur de remplissage des tuiles de type 3

TABLE 5.3 – Variables globales relatives aux attributs de remplissage des tuiles

TilesFillOpacity3	1	Opacité de remplissage des tuiles de type 3
TilesFillStyle4	full	Style de remplissage des tuiles de type 4
TilesFillColor4	darkgreen	Couleur de remplissage des tuiles de type 4
TilesFillOpacity4	1	Opacité de remplissage des tuiles de type 4
TilesFillStyle5	full	Style de remplissage des tuiles de type 5
TilesFillColor5	dimgray	Couleur de remplissage des tuiles de type 5
TilesFillOpacity5	1	Opacité de remplissage des tuiles de type 5
TilesFillStyle6	full	Style de remplissage des tuiles de type 6
TilesFillColor6	coral	Couleur de remplissage des tuiles de type 6
TilesFillOpacity6	1	Opacité de remplissage des tuiles de type 6
TilesFillStyle7	full	Style de remplissage des tuiles de type 7
TilesFillColor7	darkgoldenrod	Couleur de remplissage des tuiles de type 7
TilesFillOpacity7	1	Opacité de remplissage des tuiles de type 7
TilesFillStyle8	full	Style de remplissage des tuiles de type 8
TilesFillColor8	cadetblue	Couleur de remplissage des tuiles de type 8
TilesFillOpacity8	1	Opacité de remplissage des tuiles de type 8
TilesFillStyle9	full	Style de remplissage des tuiles de type 9
TilesFillColor9	limegreen	Couleur de remplissage des tuiles de type 9
TilesFillOpacity9	1	Opacité de remplissage des tuiles de type 9
TilesFillStyle10	full	Style de remplissage des tuiles de type 10
TilesFillColor10	lightgray	Couleur de remplissage des tuiles de type 10
TilesFillOpacity10	1	Opacité de remplissage des tuiles de type 10

5.5 Pavages de Truchet

La totalité des pavages précédemment évoqués (sauf un) sont des pavages par substitution (en anglais, substitution tilings) ; ils sont donc tous définis par une ou plusieurs règles de substitution à appliquer sur les différentes tuiles du pavage. Les pavages de Truchet, qui figurent tout de même dans la liste précédente, ne sont pas construits de cette manière : ils sont construits « aléatoirement ».

Un pavage de Truchet est constitué de 4 types de tuiles différentes. La figure 5.5 donne les exemples possibles de ces tuiles présents dans le fichier `Cristallo.mac`. Une fois ces 4 tuiles de base définies, on pourra mettre côte à côte deux tuiles seulement si leurs zones colorées coïncident. Ainsi, pour le premier type de tuile, deux possibilités s'offrent à nous pour définir les tuiles qui lui seront mitoyennes : il s'agira soit de tuiles de type 2 ou de type 3. Pour chacun de ces types de tuiles, on a deux possibilités concernant la nature des tuiles voisines. Le choix de celles-ci se fait aléatoirement.

`Cristallo.mac` permet donc la création de tels pavages. Ainsi, le fichier dispose des macros `TruchetTiles` et `Truchet` comme pour n'importe quel autre pavage apériodique précédemment évoqué. Ces deux macros s'utilisent d'ailleurs exactement de la même manière que les autres. On pourra donc utiliser la macro graphique `DrawTiles` pour dessiner de tels pavages. Les syntaxes de ces deux fonctions sont donc les suivantes :

```
TruchetTiles(<Point 1>, <Point 2>, <Type de tuile>)
Truchet(<Liste de tuiles>, <Facteur>)
```

Une troisième macros relative aux pavages de Truchet est également présente dans le fichier. Il s'agit d'une macro graphique nommée `DrawTruchetTiles` permettant de dessiner des pavages de Truchet en utilisant différents types de tuiles. Cette macros possède la syntaxe suivante :

```
DrawTruchetTiles(<Liste de Tuiles>, <Mode>, [Options])
```

Le premier argument est une liste de tuiles de Truchet. Classiquement, on utilisera une liste renvoyée par la macro `Truchet`. Le second argument permet de définir l'apparence des tuiles du pavage. Ce paramètre `Mode` doit être soit un entier compris entre 1 et 2, soit une liste de 4 entiers compris entre 1 et 2. Dans le premier cas, le pavage sera constitué exclusivement des tuiles correspondant au mode choisi. La figure 5.5 donne un aperçu des différents modes utilisable vis à vis des différents types de tuiles. Si ce second argument est une liste de 4 entiers, alors le premier entier désignera le mode pour les tuiles de type 1, le second définira le mode pour le deuxième type de tuile, etc... Un exemple mélangeant les différents modes dans un même pavage est présenté dans la section 5.7.

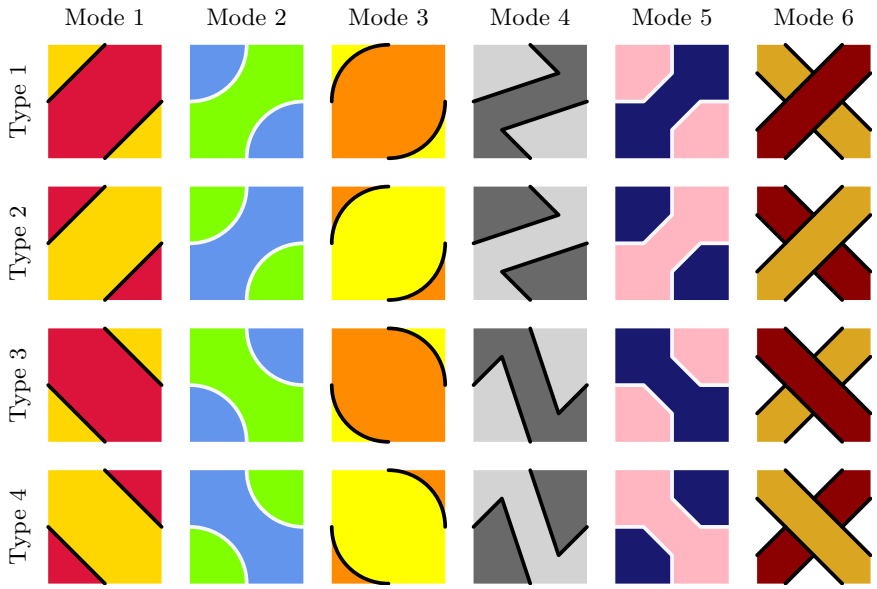
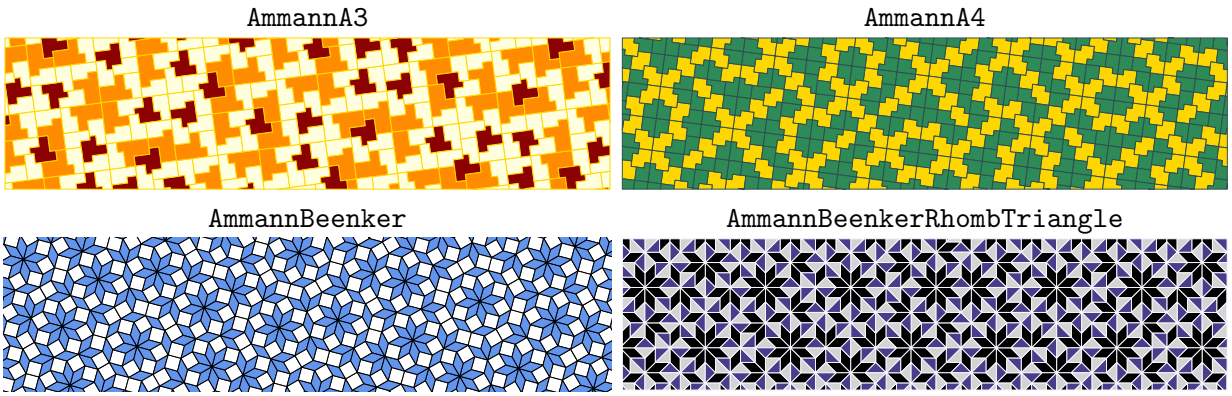


FIGURE 5.5 – Modes et types de tuiles pour les pavages de Truchet

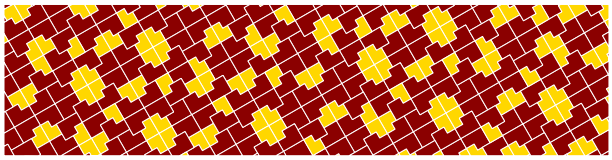
Les différences entre l'utilisation de `DrawTiles` et `DrawTruchetTiles` pour les pavages de Truchet résident alors essentiellement sur deux points :

- La macro `DrawTiles` n'offrent aucun choix concernant le mode de pavage. Seuls le premier mode est utilisable. La macro `DrawTruchetTiles` offre plus d'options à ce niveau : plusieurs modes sont à disposition et on peut même mélanger ceux-ci dans un même pavage.
- Avec la macro `DrawTiles` les tuiles sont considérés comme des carrés bicolores. Ainsi, les cotés des carrés seront dessinés alors qu'avec la deuxième macros ce ne sera pas le cas.

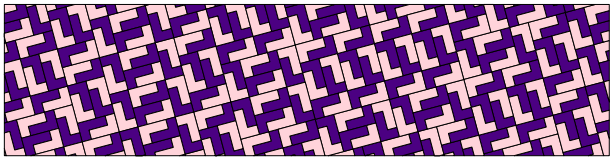
5.6 Aperçu des pavages apériodique de `Cristallo.mac`



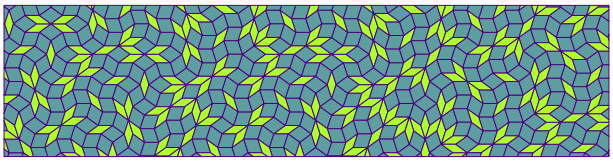
AmmannChair



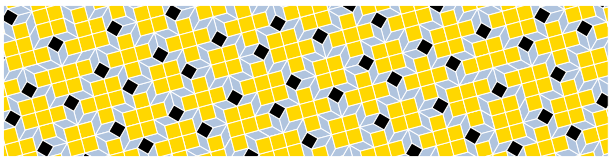
Armchair



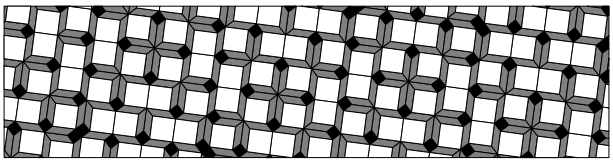
Binary



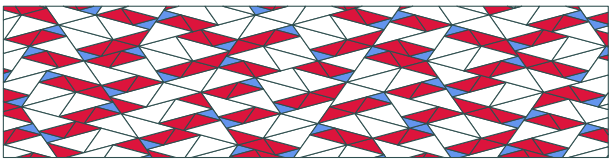
Canonical1



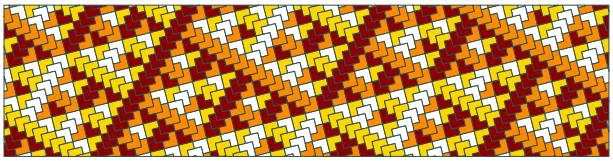
Canonical2



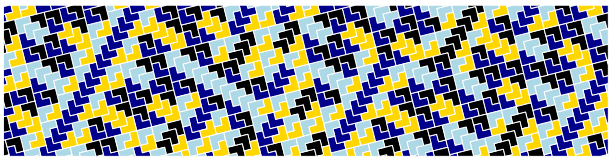
ChaimsCubicPV



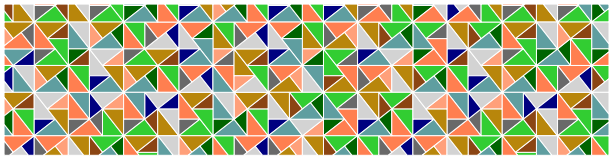
Chair1



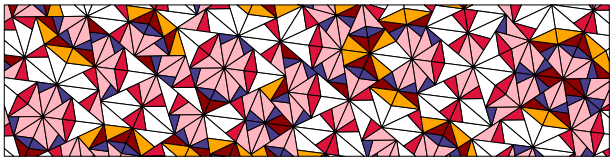
Chair2



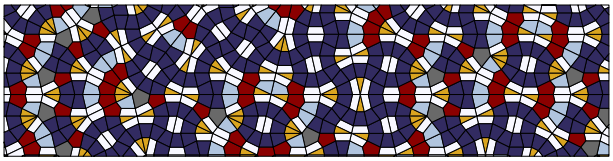
ColouredGoldenTriangle



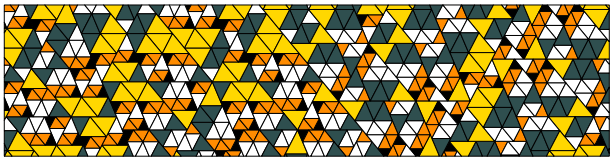
CubicPinwheel



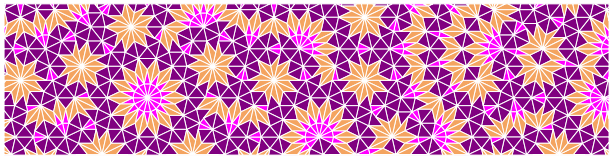
CyclotomicRhombs7Fold



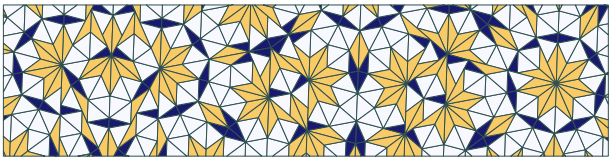
DanzersNonFLC5



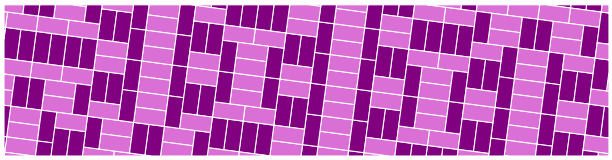
Danzers7Fold1



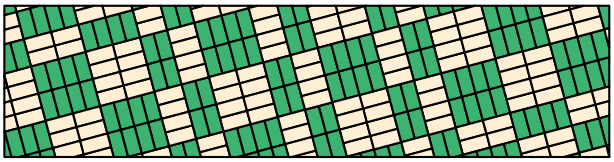
Danzers7Fold2



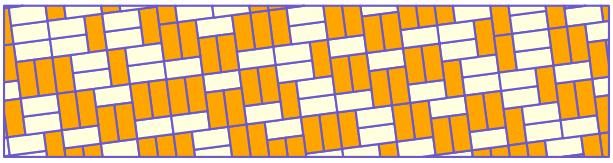
Domino1



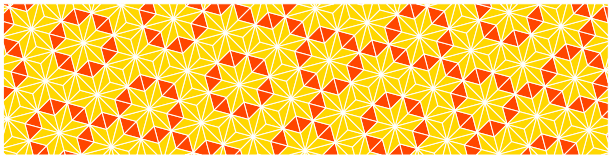
Domino2



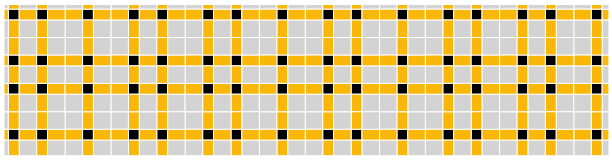
Domino3



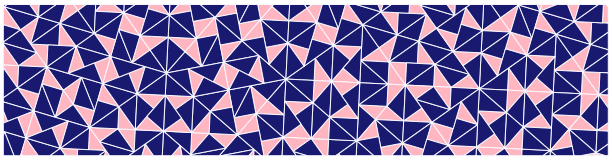
Equithirds



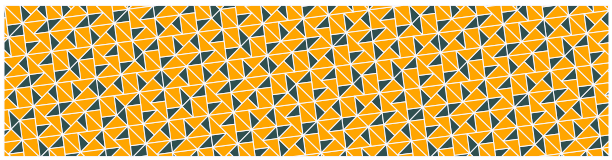
FibonacciTimesFibonacci



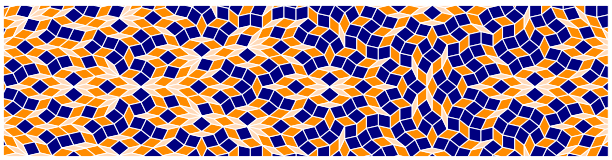
GoldenPinwheel



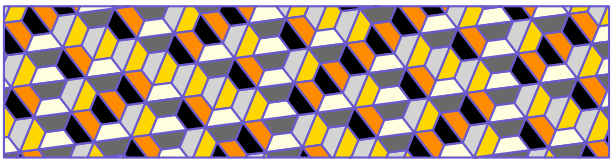
GoldenTriangle



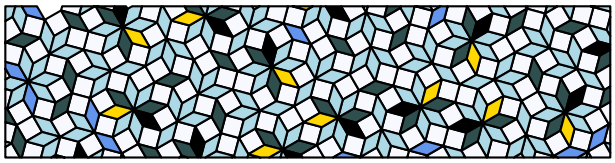
GoodmanStrauss7FoldRhomb



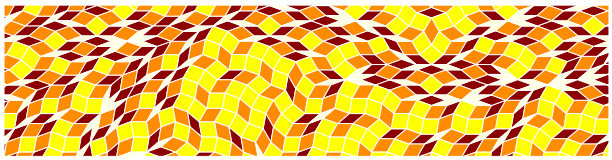
HalfHex



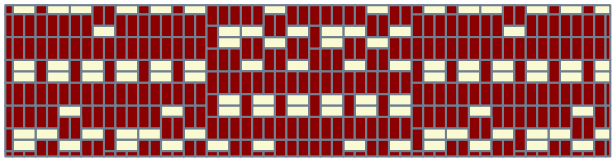
Harriss4FoldRhomb



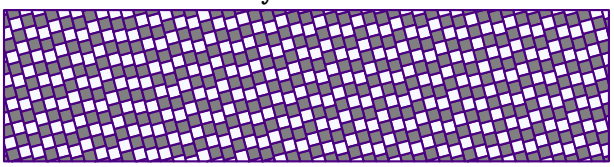
Harriss9FoldRhomb



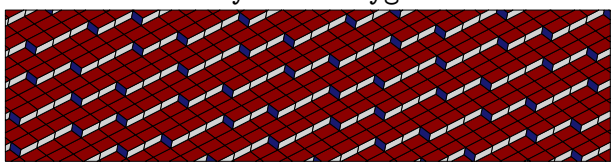
ImbalancedOrientations



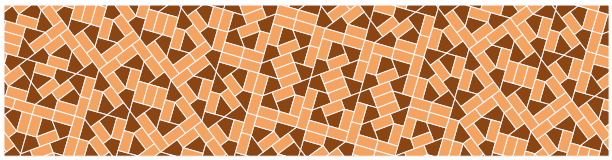
KenyonNonFLC



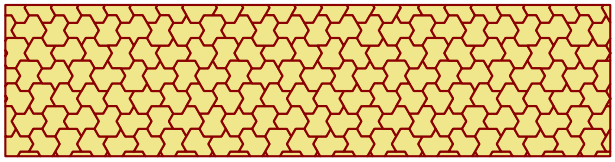
Kenyon121Polygon



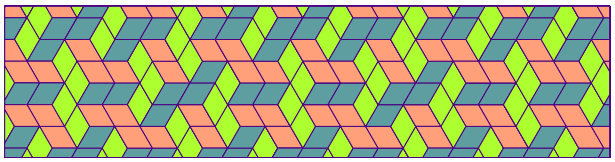
KiteDomino



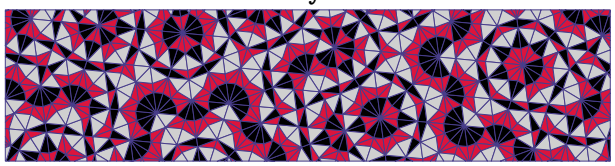
Limhex



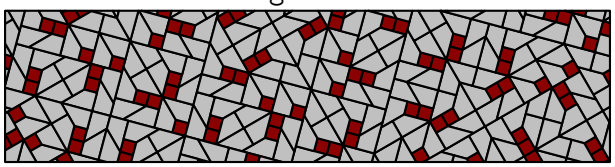
Lord



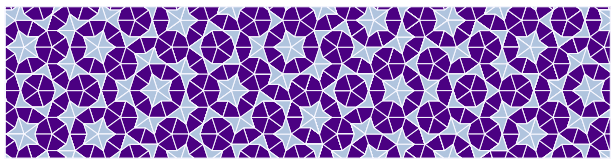
Maloney7Fold



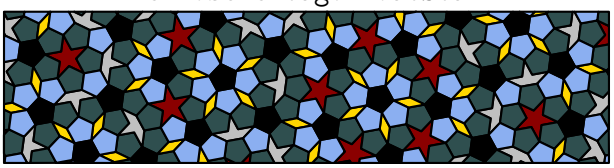
Octagonal1225



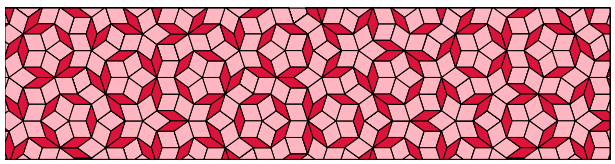
PenroseKiteDart



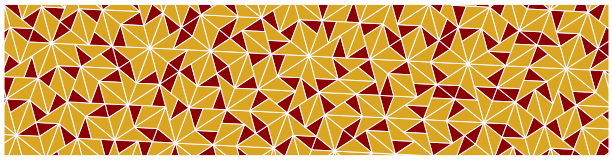
PenrosePentagonBoatStar



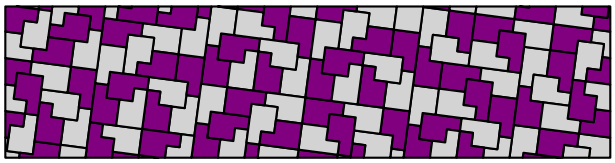
PenroseRhomb



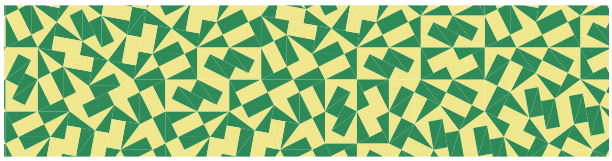
PenroseTriangle



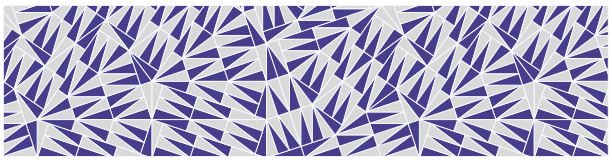
Pentomino



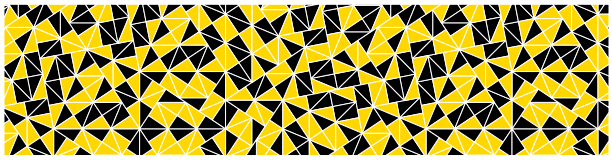
Pinwheel1



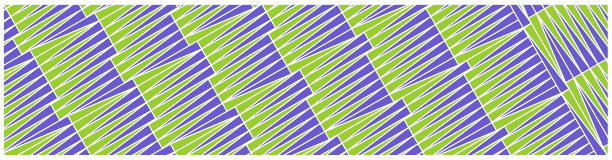
Pinwheel2



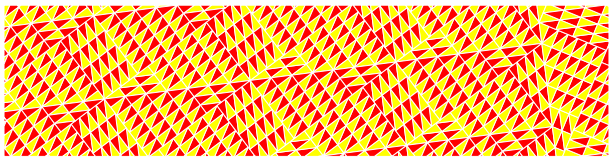
Pinwheel3



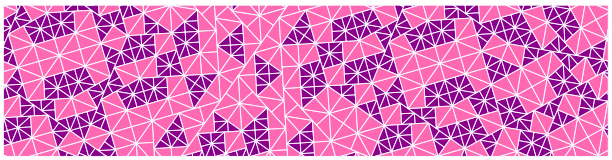
Pinwheel4



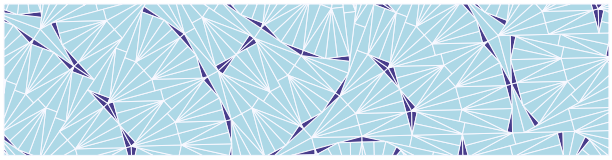
Pinwheel5



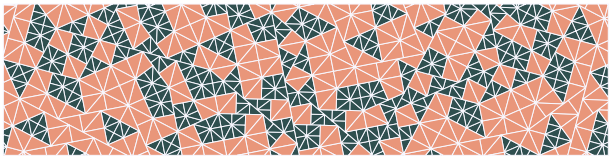
Pinwheel12



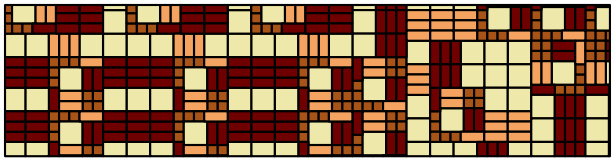
Pinwheel21



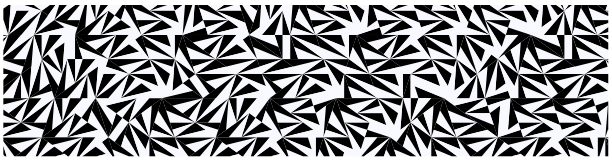
Pinwheel31



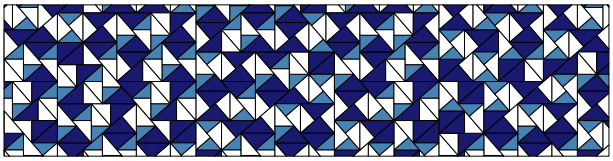
PriebeFrankNonPV



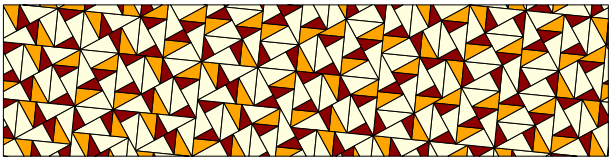
PsychedelicPenrose



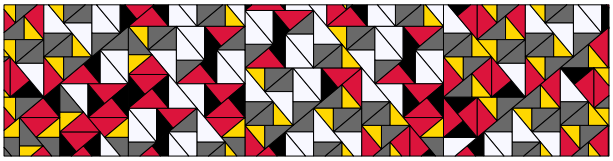
Pythagoras31



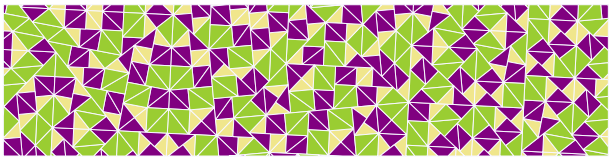
Pythagoras32



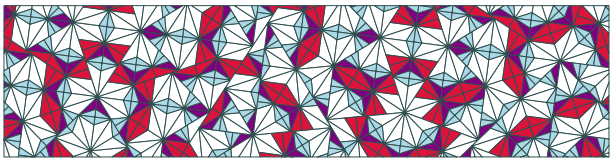
Pythagoras52



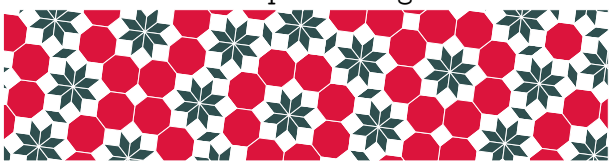
Pythia31



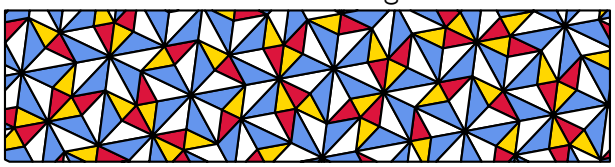
QuarticPinwheel



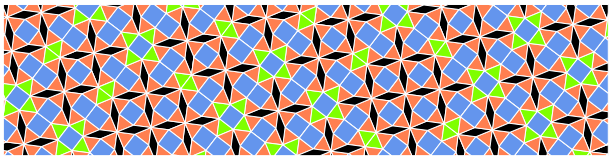
RhombSquareOktagon



RobinsonTriangle

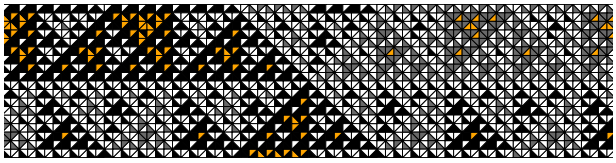


Rorschach

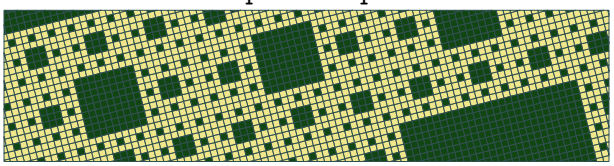


SemiDetachedHouse

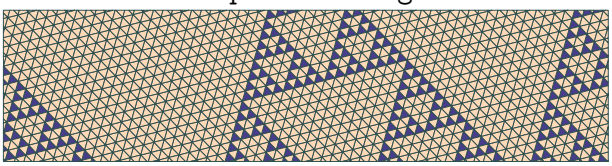
SemiDetachedHouseSquared



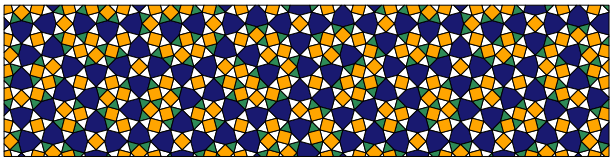
SierpinskiSquare



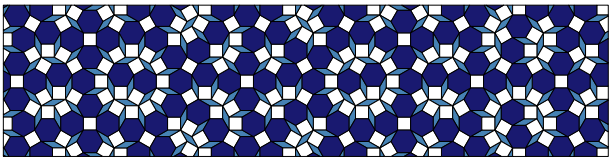
SierpinskiTriangle



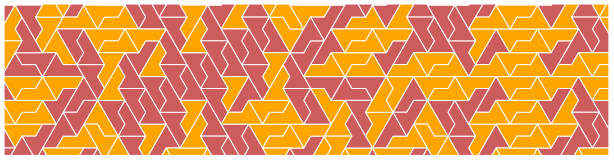
Shield



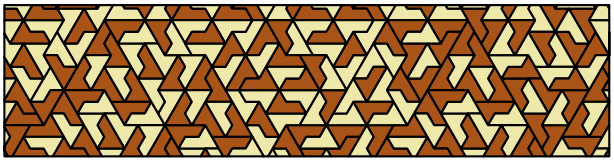
Socolar



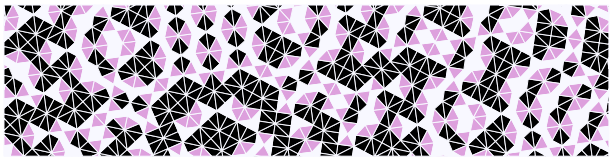
Sphinx



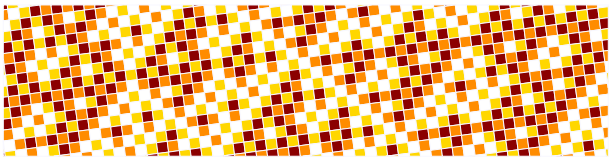
Sphinx9



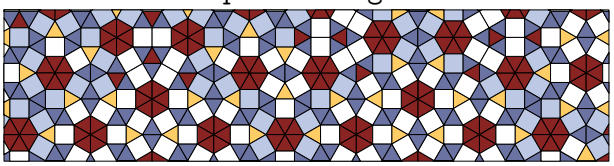
Sqrt6Triangles



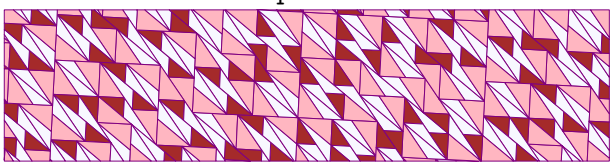
SquareChair



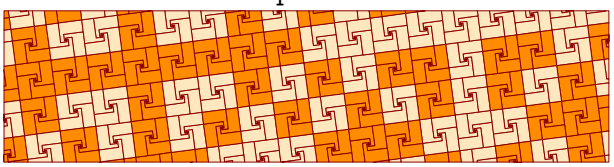
SquareTriangle



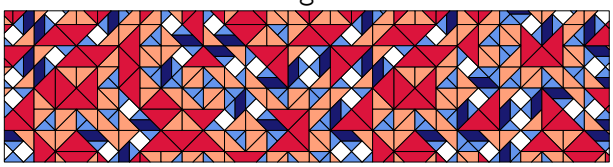
Squeeze



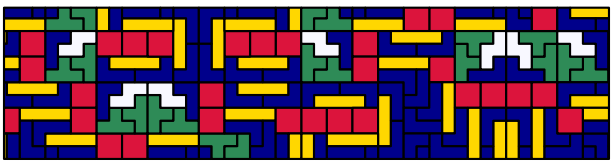
Squiral



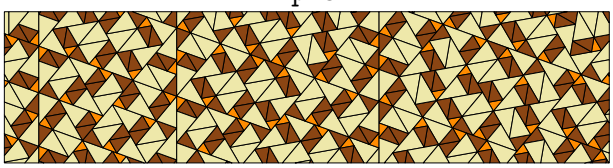
Tangram



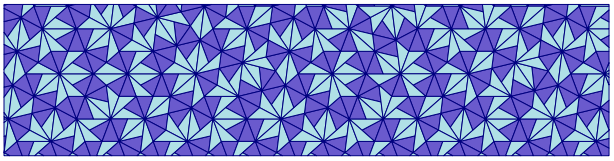
Tetris



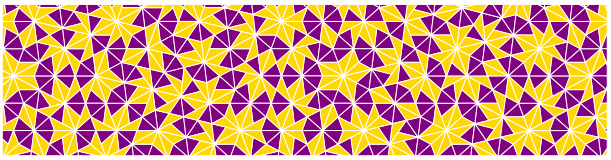
Tipi31



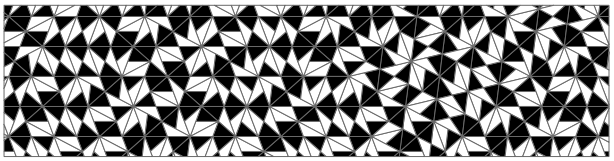
TriangleDuo1



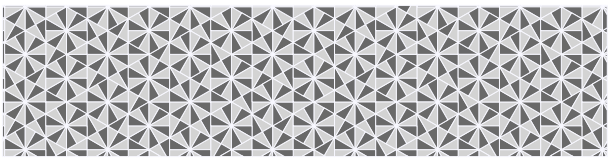
TriangleDuo2



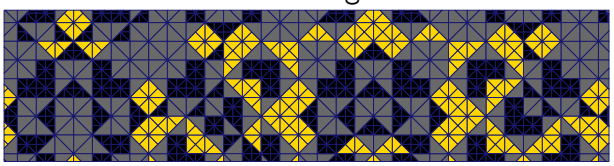
TriangleDuo3



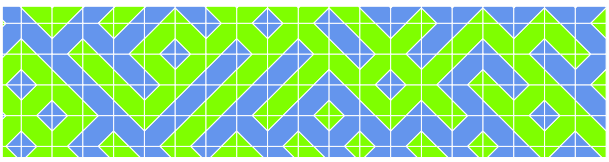
Trihex



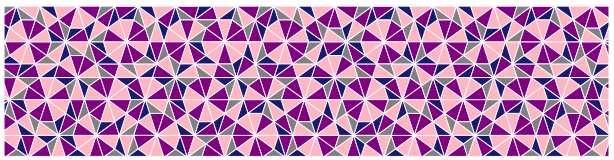
Tritriangle



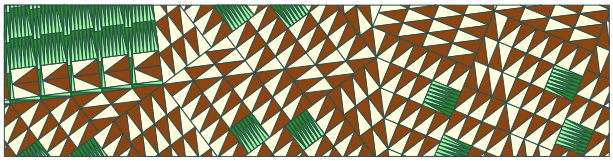
Truchet



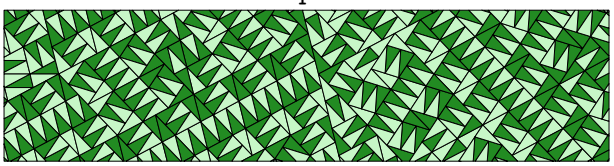
TuebingenTriangle



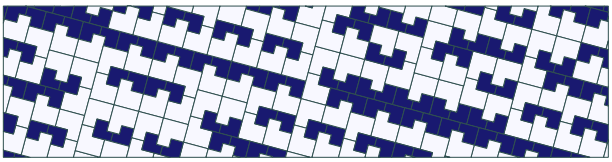
Uberpinwheel



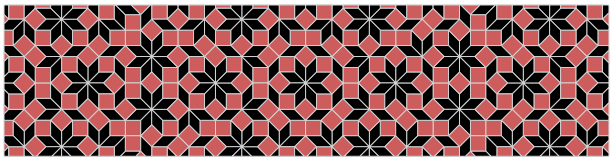
Viper



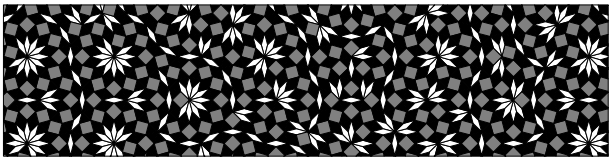
Waltonchair



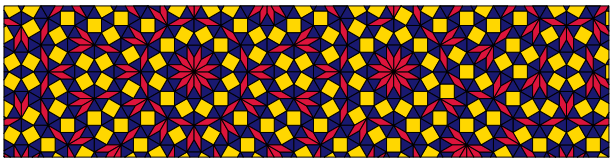
WatanabeItoSoma8Fold



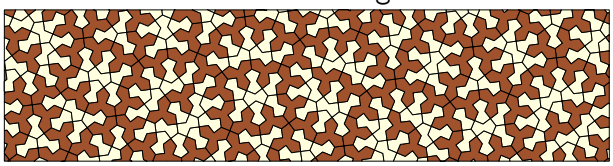
WatanabeItoSoma12Fold1



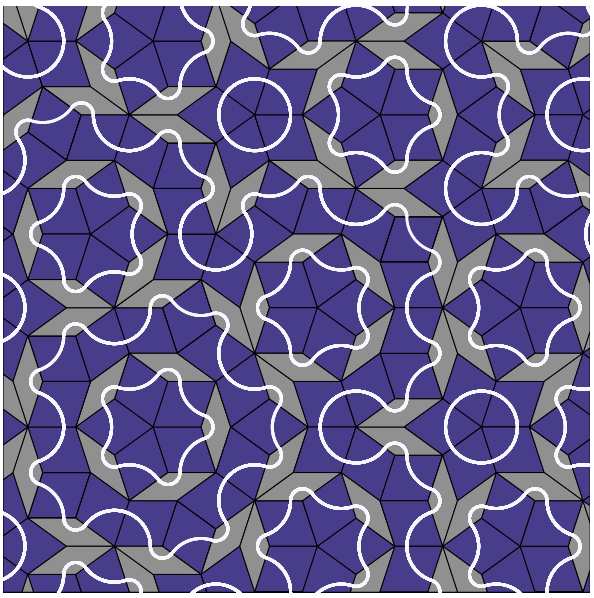
WatanabeItoSoma12Fold2



WheelTiling



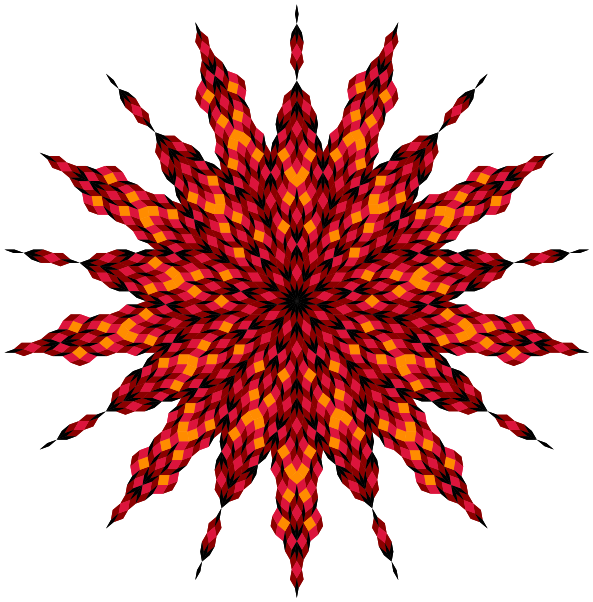
5.7 Exemples



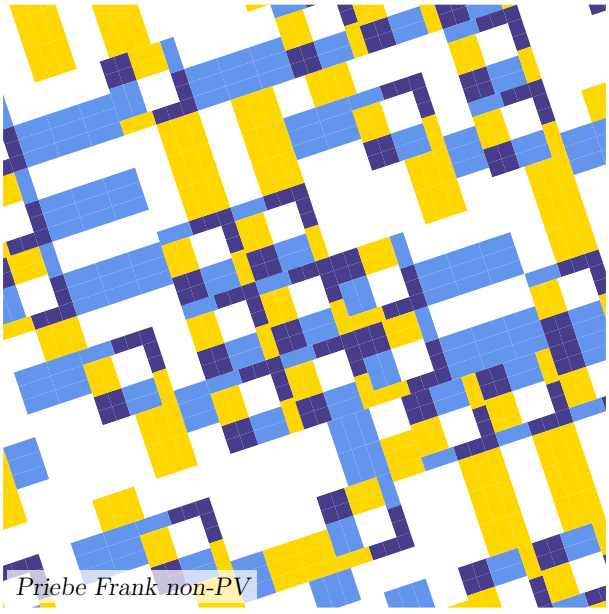
(a) Exemple 1

Code (Exemple 1)

```
[
Fenetre(0.3+0.2*i, 1+0.9*i, 15+15*i),
Marges(0, 0, 0, 0),
TilesFillColor1:=darkslateblue,
TilesFillColor2:=Dark(silver, 0.25),
$P:=PenroseRhombTiles(0, 1, 1),
$Pav:=PenroseRhomb(P, 6),
DrawTiles(Pav),
Width:=Thicklines, Color:=ghostwhite,
$f:=1/4,
for $T in Pav By jump do
  $A:=T[1], $B:=T[2], $C:=T[3], $D:=T[4],
  if sep=jump1 then
    Arc(D, C, B, (1-f)*abs(D-C))
  elif sep=jump2 then
    Arc(A, D, C, f*abs(D-C))
  fi
od
]
```



(b) Exemple 2



(c) Exemple 3

Code (Exemple 2)

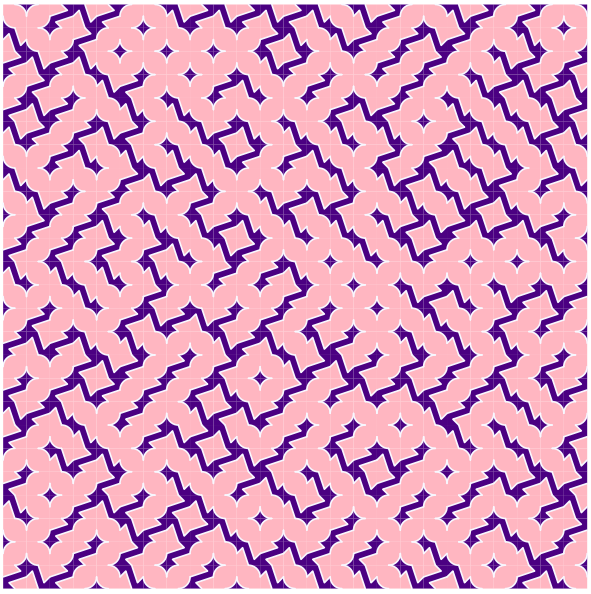
```
[
Fenetre(-2-2*i, 2+2*i, 1+i), size(10),
Marges(0, 0, 0, 0),
TilesFillColor1:=darkred,
TilesFillColor2:=darkorange,
TilesFillColor3:=crimson,
TilesFillColor4:=black,
TilesLineStyle:=noline,
$P:=[Seq(Harriss9FoldRhombTiles(0, exp(2*$k*i*pi/9), 4), k, 1, 18),
      Seq(Harriss9FoldRhombTiles(2*cos(pi/18)*exp(i*(pi/6+2*k*pi/9)),
            exp(2*i*pi*(k+1)/9), 4), k, 1, 18)],
$Pav:=Harriss9FoldRhomb(P, 2),
DrawTiles(Pav)
]
```

Code (Exemple 3)

```
[
Fenetre(0+0.25*i, 0.5+0.75*i, 1+i), size(10),
Marges(0, 0, 0, 0),
TilesFillColor1:=darkslateblue,
TilesFillColor2:=white,
TilesFillColor3:=cornflowerblue,
TilesFillColor4:=gold,
TilesLineStyle:=noline, TilesColor:=white,
$P:=PriebeFrankNonPVTiles(0, 0.75+0.25*i, 1),
$Pav:=PriebeFrankNonPV(P, 5),
DrawTiles(Pav),
PavAperiodiqueNames("PriebeFrankNonPV", "pfnpv", 0),
LineStyle:=noline, FillStyle:=full, FillOpacity:=0.7,
LabelStyle:=bottom+left+framed, LabelSize:=large,
LabelDot(Xmin+i*Ymin, ["\slshape ", @pfnpv], "NE", 0, 0.1)
]
```

Code (Exemple 4)

```
[
Marges(0, 0, 0, 0),
Fenetre(0, 1+i, 10*(1+i)),
$P:=TruchetTiles(0, 1, 1),
$Pav:=Truchet(P, 25),
LineCap:=round,
TilesWidth:=thicklines,
TilesColor:=ghostwhite,
TilesFillColor1:=indigo,
TilesFillColor2:=lightpink,
DrawTruchetTiles(Pav, [4, 3, 4, 3])
]
```



(d) Exemple 4

Index