Pour piloter la tortue de xcas dès le CP

R. De Graeve

Université Joseph Fourier, Grenoble

xcas est un logiciel libre de calcul formel contenant une tortue Logo. Il est téléchargeable à partir de

```
http://www-fourier.ujf-grenoble.fr/~parisse/giac_fr.html
```

La tortue de xcas est un équivalent de la tortue Logo mais avec plus de primitives et un langage de programmation proche du C++.

Ce cours d'introduction est destiné à faciliter la prise en main la tortue de xcas par un utilisateur sachant reconnaitre les lettres de l'alphabet (niveau CP) puis connaissant un peu de mathématiques (niveau CE1-CM2), et ayant une pratique minimale de l'outil informatique.

Pour une pratique plus avancée, on se reportera à l'aide en ligne et au manuel `La tortue et la géométrie avec xcas` disponibles en ouvrant le menu `Aide` du logiciel xcas .

Le but de ce qui suit est d'aider le débutant en introduisant quelques unes des commandes les plus courantes. Il est conseillé de lire ce document après avoir lancé xcas (en tapant `xcas enter`) et en demandant le niveau `tortue` ou bien en ouverant un écran de dessin pour piloter la tortue (en tapant en même temps `Alt` et `d` ce que l'on note `Alt+d`), On exécutera les commandes proposées une par une pour en comprendre l'effet et on s'exercera en faisant les exercices proposés.

La table des matières et l'index sont à la fin de ce document.

Chapitre 1

Les commandes

1.1 Pour commencer

1.1.1 Interface

Au démarrage de `xcas`, la première fois, on vous demande votre niveau : si vous répondez `tortue`, un éditeur de programmes et un écran Tortue s'ouvrent sur deux niveaux et, par la suite, les lancements ultérieurs de `xcas` avec `xcas enter` provequera l'ouverture de ces deux niveaux.
Si vous n'avez pas choisi `Tortue` au démarrage de `xcas`, vous pouvez utiliser le menu :
`Configuration->Mode->tortue` puis,
`Configuration->Sauver preferences`
cela aura le même effet.
Si vous n'avez pas choisi `Tortue` au démarrage de `xcas`, vous pouvez aussi ouvrir un écran Tortue avec `Alt+d` et un éditeur de programmes avec `Alt+p`.

1.1.2 Le menu Scolaire->Tortue

Les commandes sont regroupées par thèmes dans le menu `Scolaire->Tortue`. Lorsqu'on sélectionne une commande dans un menu, une aide succinte s'affiche dans la fenêtre blanche des messages en bas à droite (double-cliquer pour afficher le message en entier), et le manuel s'ouvre dans votre navigateur à la bonne page.

1.1.3 Entrer des commandes

L'exécution d'une ligne se fait simplement par la touche `Entrée`. On peut éditer plusieurs commandes à la file avant leur exécution à condition de les séparer par un point-virgule.

Les opérations sur les nombres réels sont notées par les signes habituels `+,-,*,/`

Opérations	
+	addition
-	soustraction
*	mutiplication
/	division

Les parenthèses ont le sens usuel pour spécifier l'ordre des opérations. Les priorités entre opérations sont standards (la multiplication est prioritaire sur l'addition). Dans le doute, il est toujours prudent de mettre des parenthèses pour s'assurer que l'ordre des opérations est celui souhaité.

Les commandes sont numérotées, ainsi que les réponses, mais, si vous avez modifié une ligne de commande, celle-ci garde numéro qu'elle avait. On peut retrouver, en appuyant en même temps sur `Ctrl` et ↑ de votre clavier, les questions précédentes.

1.2 Pour commander la Tortue

1.2.1 Le principe

La tortue est un petit robot muni d'un crayon qui se déplace sur l'écran. L'écran est muni d'un repère : l'origine `O` est en bas et à gauche, l'axe des `x` est horizontal dirigé vers la droite et l'axe des `y` est vertical dirigé vers le haut. L'unité est le pixel qui correspond à un pas de tortue.
La tortue à une position (ses coordonnées dans ce repère) et un cap (sa direction repérée par la valeur en degrés de l'angle qu'elle fait avec `Ox`.
On commande la tortue en la faisant avancer, reculer, changer de direction pour qu'elle fasse des dessins avec son crayon : pour cela on n'utilise pas en général les coordonnées mais des commandes relatives à la tortue.

1.2.2 Les déplacements

Les changements de place

Les changements de place changent la position de la tortue.

- `efface` ou `efface()` : efface l'écran et remet la tortue en position (150,150) avec le cap 0.
- `avance` ou `avance()` : fait avancer la tortue de 10 pas.
  `avance(30)` : fait avancer la tortue de 30 pas.
  `avance(-30)` : fait reculer la tortue de 30 pas.
- `recule` ou `recule()` : fait reculer la tortue de 10 pas.
  `recule(30)` : fait reculer la tortue de 30 pas.
  `recule(-30)` : fait avancer la tortue de 30 pas.
- `saute` ou `saute()` : fait avancer sans tracer, la tortue de 10 pas (un point marque sa position finale).
  `saute(30)` : fait avancer, sans tracer, la tortue de 30 pas (un point marque la position finale).

`saute(-30)` : fait reculer, sans tracer, la tortue de 30 pas (un point marque la position finale).

- `pas_de_cote` ou `pas_de_cote()` : fait, sans tracer, 10 pas de coté sur la gauche sans tracer (un point marque la position finale).
  `pas_de_cote(30)` : fait, sans tracer, 30 pas de coté sur la gauche (un point marque la position finale).
  `pas_de_cote(-30)` : fait, sans tracer, 30 pas de coté sur la droite (un point marque la position finale).

Les changements d'orientation

Les changements d'orientation changent le cap de la tortue.
Les angles sont mesurés en degrés.

- `tourne_gauche` ou `tourne_gauche()` : la tortue tourne sur place, à gauche de 90 degrés.
  `tourne_gauche(30)` : la tortue tourne sur place, à gauche de 30 degrés.
  `tourne_gauche(-30)` : la tortue tourne sur place, à droite de 30 degrés.
- `tourne_droite` ou `tourne_droite()` : la tortue tourne sur place, à droite de 90 degrés.
  `tourne_droite(30)` : la tortue tourne sur place, à droite de 30 degrés.
  `tourne_droite(-30)` : la tortue tourne sur place, à gauche de 30 degrés.

Déplacements	
`efface`	pour effacer
`avance`	pour avancer
`recule`	pour reculer
`saute`	pour avancer ou reculer sans tracer
`pas_de_cote`	pour faire des pas de coté sur la gauche ou sur la droite sans tracer
`tourne_gauche`	pour tourner à gauche
`tourne_droite`	pour tourner à droite

1.2.3 La gestion du crayon et de la tortue

La gestion du crayon

On peut déplacer la tortue en laissant, ou non, une trace et changer la couleur du crayon. Il y a 256 couleurs numerotées de 0 à 255. Seules les 8 premières couleurs ont un nom : `0 noir, 1 rouge, 2 vert, 3 jaune, 4 bleu, 5 magenta, 6 cyan, 7 blanc`.

- `leve_crayon` ou `leve_crayon()` : fait lever le crayon de la tortue qui se déplace alors sans laisser sa trace.
- `baisse_crayon` ou `baisse_crayon()` : fait baisser le crayon de la tortue qui se déplace alors en laissant sa trace.

- `crayon` ou `crayon()` : renvoie la couleur du crayon de la tortue.
  `crayon rouge` ou `crayon(rouge)` ou `crayon(1)` : change la couleur du crayon de la tortue selon la couleur spécifiée (ici rouge). `crayon gomme` ou `crayon(gomme)` est identique à `crayon(blanc)`

La gestion de la tortue

- `cache_tortue` ou `cache_tortue()` : cache la tortue.
- `montre_tortue` ou `montre_tortue()` : fait apparaitre la tortue.
- `dessine_tortue` ou `dessine_tortue()` : marque la position de la tortue selon une forme pleine.

Les couleurs et la tortue	
`crayon`	pour connaitre ou changer la couleur du crayon.
`cache_tortue`	pour cacher la tortue
`montre_tortue`	pour voir la tortue
`dessine_tortue`	pour marquer la position de la tortue selon une forme pleine.

1.2.4 Les formes

Le cercle et les arcs de cercle

- `rond` ou `rond()` : dessine un cercle de rayon 10 pas tangent à la tortue et à gauche de la tortue.
- `rond(30)` : dessine un cercle de rayon 30 pas tangent à la tortue et à gauche de la tortue.
- `rond(-30)` : dessine un cercle de rayon 30 pas tangent à la tortue et à droite de la tortue.
- `rond(30,60)` : dessine l'arc de cercle, de rayon 30 pas, tangent à la tortue T, situé à gauche de la tortue et d'angle au centre de mesure 60 degrés à partir de la position T de la tortue. À l'arrivée la tortue est tangente à cet arc.
- `rond(30,-60)` : dessine l'arc de cercle, de rayon 30 pas, tangent à la tortue T, situé à gauche de la tortue et d'angle au centre de mesure (360-60) degrés à partir de la position T de la tortue. À l'arrivée la tortue est tangente à cet arc.
- `rond(-30,60)` : dessine l'arc de cercle, de rayon 30 pas, tangent à la tortue T, situé à droite de la tortue et d'angle au centre de mesure 60 degrés à partir de la position T de la tortue. À l'arrivée la tortue est tangente à cet arc.
- `rond(-30,-60)` : dessine l'arc de cercle, de rayon 30 pas, tangent à la tortue T, situé à droite de la tortue et d'angle au centre de mesure (360-60) degrés à partir de la position T de la tortue. À l'arrivée la tortue est tangente à cet arc.
- `rond(30,60,100)` : dessine un arc de cercle de rayon 30 pas, tangent à la tortue T, situé à gauche de la tortue et allant de A à B, où l'arc(TA) a pour mesure 60 degrés et l'arc(TB) a pour mesure 100 degrés (les mesures sont modulo 360). À l'arrivée, la tortue est tangente à cet arc.
- `rond(-30,60,100)` : dessine un arc de cercle de rayon 30 pas, tangent à la tortue T, situé à droite de la tortue et allant de A à B où l'arc(TA) a pour mesure

60 degrés et l'arc(TB) a pour mesure 100 degrés (les mesures sont modulo 360). À l'arrivée la tortue est tangente à cet arc.

Les formes pleines

- `triangle_plein` ou `triangle_plein()` : dessine un triangle équilatéral plein de côtés 10, à gauche de la tortue.
  `triangle_plein 30` ou `triangle_plein(30)` : dessine un triangle équilatéral plein de côtés 30, à gauche de la tortue.
  `triangle_plein(30,40)` : dessine un triangle rectangle plein de côtés 30 et 40, à gauche de la tortue.
  `triangle_plein(30,30,40)` : dessine un triangle isocèle plein de côtés 30, d'angle au sommet 40 degrés, à gauche de la tortue.
  `triangle_plein(30,40,50)` : dessine un triangle plein de côtés 30 et 40, délimitant un angle de 50 degrés, à gauche de la tortue.
  `triangle_plein(30,40,-50)` : dessine un triangle plein de côtés 30 et 40, délimitant un angle de 50 degrés, à droite de la tortue.

- `rectangle_plein` ou `rectangle_plein()` : dessine un carré plein de côtés 10, à gauche de la tortue.
  `rectangle_plein 30` ou `rectangle_plein(30)` : dessine un carré plein de côté 30, à gauche de la tortue.
  `rectangle_plein(10,30)` : dessine un rectangle plein de côtés 10 et 30, à gauche de la tortue.
  `rectangle_plein(10,30,-90)` ou `rectangle_plein(10,-30)` : dessine un rectangle plein de côtés 10 et 30, à droite de la tortue.
  `rectangle_plein(30,30,50)` : dessine un losange plein de côtés 30, délimitant un angle de 50 degrés, à gauche de la tortue.
  `rectangle_plein(30,40,50)` : dessine un parallélogramme plein de côtés 30 et 40, délimitant un angle de 50 degrés, à gauche de la tortue.
  `rectangle_plein(30,40,-50)` : dessine un parallélogramme plein de côtés 30 et 40, délimitant un angle de 50 degrés, à droite de la tortue.

- `disque` ou `disque()` : dessine un disque plein de rayon 10, à gauche de la tortue et tangent à la tortue. À l'arrivée la tortue est tangente à ce disque.
  `disque 30` ou `disque(30)` : dessine un disque plein de rayon 30, à gauche de la tortue et tangent à la tortue. À l'arrivée la tortue est tangente à ce disque.
  `disque -30` ou `disque(-30)` : dessine un disque plein de rayon 30, à droite de la tortue et tangent à la tortue. À l'arrivée la tortue est tangente à ce disque.
  `disque(30,60)` : dessine un secteur angulaire plein de rayon 30, d'angle au centre 60 degrés, à gauche de la tortue et tangent à la tortue. À l'arrivée la tortue est tangente à l'extrémité de l'arc délimitant ce secteur.
  `disque(-30,60)` : dessine un secteur angulaire plein de rayon 30, d'angle au centre 60 degrés, à gauche de la tortue et tangent à la tortue. À l'arrivée la tortue est tangente à l'extrémité de l'arc délimitant ce secteur.

`disque(30,60,80)` : dessine un secteur angulaire plein de rayon 30, d'angle au centre 20 degrés allant de A à B avec arc(TA)=60 degrés et arc(TB)=80 degrés, à gauche de la tortue et tangent à la tortue. À l'arrivée la tortue est tangente à l'extrémité de l'arc délimitant ce secteur.
`disque(-30,60 80)` : dessine un secteur angulaire plein de rayon 30, d'angle au centre 20 degrés allant de A à B avec arc(TA)=60 degrés et arc(TB)=80 degrés, à droite de la tortue et tangent à la tortue. À l'arrivée la tortue est tangente à l'extrémité de l'arc délimitant ce secteur.

- `disque_centre` ou `disque_centre()` : dessine un disque plein de rayon 10, et de centre la tortue. À l'arrivée la tortue est revenue à sa position de départ.
  `disque_centre 30` ou `disque_centre(30)` : dessine un disque plein de rayon 30, et de centre la tortue. À l'arrivée la tortue est revenue à sa position de départ.
  `disque_centre -30` ou `disque(-30)` : dessine un disque plein de rayon 30, et de centre la tortue. À l'arrivée la tortue est revenue à sa position de départ.
  `disque_centre (30,60)` : dessine un secteur angulaire plein de rayon 30, d'angle au centre 60 degrés, et de centre la tortue. À l'arrivée la tortue a tourné à gauche de 60 degrés.
  `disque_centre (-30,60)` : dessine un secteur angulaire plein de rayon 30, d'angle au centre 60 degrés, et de centre la tortue. À l'arrivée la tortue a tourné à droite de 60 degrés.
  `disque_centre (30,60,80)` : dessine un secteur angulaire plein de rayon 30, d'angle au centre 20 degrés allant de A à B avec arc(TA)=60 degrés et arc(TB)=80 degrés, à gauche de la tortue. À l'arrivée la tortue a tourné à gauche de 80 degrés.
  `disque_centre (-30,60 80)` : dessine un secteur angulaire plein de rayon 30, d'angle au centre 20 degrés allant de A à B avec arc(TA)=60 degrés et arc(TB)=80 degrés, à droite de la tortue. À l'arrivée la tortue a tourné à droite de 80 degrés.

- `polygone_rempli(-n)` : remplit le polygone définit auparavant si il a nécessité `n` instructions, par exemple :
  `repete(4,avance,tourne_droite);polygone_rempli(-8);`

Les formes	
`rond`	dessine un cercle ou un arc de cercle.
`triangle_plein`	dessine un triangle plein.
`rectangle_plein`	dessine un carré, rectangle, losange ou parallélogramme plein.
`disque`	dessine un disque ou un secteur angulaire tangent à la tortue.
`disque_centre`	dessine un disque ou un secteur angulaire de centre la tortue.
`polygone_rempli`	remplit le polygône dessiné juste avant.

1.2.5 Les coordonnées de la Tortue

L'écran de la tortue est muni d'un repère dont l'origine est le point en bas et à gauche, l'axe des *x* est horizontal et dirigé vers la droite, l'axe des *y* est vertical et dirigé vers le haut, l'unité est le pixel.
Les angles sont mesurés en degrés.
La tortue à une position (ses coordonnées dans ce repère) et un cap (sa direction repérée par la valeur en degrés de l'angle qu'elle fait avec `Ox`.
On peut lire, en haut à droite et en noir sur fond jaune, les coordonnées et le cap de la tortue.

- `position` ou `position()` : renvoie la liste des coordonnées de la tortue.
  `position(130,20)` : amène la tortue au point de coordonnées (130,20) sans changer son cap.

- `cap` ou `cap()` : renvoie le cap de la tortue.
  `cap 30` ou `cap(30)` : tourne la tortue pour que son cap soit de 30 degrés.

- `vers(130,20)` : tourne la tortue pour qu'elle soit dirigée en direction du point de coordonnées (130,20).

Positionnement	
`position`	pour avoir la position de la tortue ou pour changer sa position.
`cap`	pour avoir le cap de la tortue ou pour changer son cap.
`vers`	pour diriger la tortue selon un point.

1.2.6 Les légendes

- `ecris(Bonjour)` : écris `Bonjour` là où se trouve la tortue et sans modifier la position de la tortue (les guillemets sont inutiles pour écrire un mot).
  `ecris("Bonjour ca va ?",50,30,10)` : écris `Bonjour ca va ?` à partir du point de coordonnées (30,10) avec la fonte 50 et sans modifier la position de la tortue (il faut mettre des guillemets pour écrire une phrase).

- `signe(Jerome)` : écris `Jerome` à partir du point de coordonnées (10,10) avec la fonte 20 et sans modifier la position de la tortue (les guillemets sont inutiles pour écrire un nom).
  `signe("Une maison faite par Jerome")` : écris, à partir du point de coordonnées (10,10), `Une maison faite par Jerome` avec la fonte 20 et sans modifier la position de la tortue (il faut mettre des guillemets pour écrire le nom et le prénom ou pour écrire une phrase).

Les légendes	
`ecris`	pour écrire là où se trouve la tortue.
`signe`	pour signer, en bas à gauche, son dessin.

1.2.7 Les programmes

Le choix

- `si alors fsi:.`
  `si alors sinon fsi:.`

L'itération

- `repete:.`
  `repete(...(repete...)...):.`
- `pour de jusque faire fpour:.`
  `pour de jusque pas faire fpour:.`
- `tantque faire ftantque:.`
- `la r\'ecursivit\'e:.`

La valeur d'une fonction

- `return:.`

Pour faire des entrées

- `lis:.`
- `lis_phrase:.`

Pour gérer les fichiers

- `sauve`: `sauve("toto.tor",tete,bras)`.
- `ramene`: `ramene("toto.tor")`.

La programmation	
`si alors`	fais les instructions si la condition est
`fsi`	réalisée.
`si alors sinon`	fais les instructions1 ou les instructions2
`fsi`	selon la condition.
`repete`	repète les mêmes instructions.
`pour de jusque faire`	fais les instructions en itérant une variable
`fpour`	avec un pas=1.
`pour de jusque pas faire`	fais les instructions en itérant une variable
`fpour`	avec un pas.
`tantque faire`	fais les instructions tant que la condition
`ftantque`	est réalisée.
`return`	pour définir la valeur d'une fonction.
`lis`	lis une expression au clavier.
`lis_phrase`	lis une phrase au clavier.
`sauve`	sauve dans un fichier des procédures.
`ramene`	ramene les procédures qui sont dans un fichier.

1.2.8 Deux exemples de programmes

Les mots d'une phrase

Une chaîne est parenthésée par des guillemets ("). Un caractère est une chaîne ayant un seul élément. Pour avoir les mots d'une phrase on transforme la phrase en une liste de mots :

```
  listemots(ph):={
    local j,n,ch,lm,m;
    lm:=[];
    n:=size(ph);
    j:=0;
    m:="";
    for(j:=0;j<n;j++){
      ch:=ph[j];
      if(ch!=" ") {
m:=m+ch;
      }
      else {
lm:=append(lm,m);m:="";
      }
    }
    lm:=append(lm,m);
    return lm;
  };
```

La conjugaison

Un exemple qui permet de conjuguer les verbes du premier groupe au présent, au futur, à l'imparfait ou au passé simple.
il suffit de taper :
`conjugue1("chanter","present")` ou
`conjugue1("chanter","futur")` ou
`conjugue1("chanter","imparfait")` ou
`conjugue1("chanter","passe_simple")`

```
conjugue1(v,t):={
local debv,fv,s,p,ter,terp,terps,teri,terf,j, vo;
vo:=["a","e","i","o","u","y"];
p:=["je","tu","il","nous","vous","ils"];
terp:=["e","es","e","ons","ez","ent"];
teri:=["ais","ais","ait","ions","iez","aient"];
terf:=["erai","eras","era","erons","erez","eront"];
terps:=["ai","as","a","âmes","âtes","èrent"];
s:=size(v);
debv:=mid(v,0,s-2);
fv:=mid(v,s-2)
si (fv!="er") alors return ("pas du premier groupe") fsi;
si (t=="present") alors ter:=terp fsi;
si (t=="imparfait") alors ter:=teri fsi;
si (t=="futur") alors ter:=terf fsi;
si (t=="passe_simple") alors ter:=terps fsi;
si (member(head(v),vo)) alors print("j'"+debv+ter[j]);
sinon
print(p[0]+" "+debv+ter[0]);
fsi;
pour j de 1 jusque 5 faire
print(p[j]+" "+debv+ter[j]);
fpour;
}
```

1.3 Comment gérer son espace de travail

1.3.1 Pour avoir l'écran dessin tortue

Choisir le menu `Edit`, sous menu `Ajouter` puis `dessin tortue` ou utiliser le raccourci clavier `Alt+d`.
On obtient :

- Au centre, un écran `dessin tortue` et on voit au début de la session, la tortue sous la forme d'un petit triangle, au pixel de coordonnées 150,150 et avec le cap 0. On travaille alors en degrés pour toutes les commandes concernant la tortue.

On remarquera que l'on trouve l'abréviation de certaines commandes sur la barre des boutons de cet écran `dessin tortue`.
– À droite, une ligne d'entrée pour les commandes pilotant la tortue. **Attention** les sorties de la ligne d'entrée de la tortue sont prevues pour afficher une seule ligne.
– À gauche un éditeur de programmes qui garde la trace de toutes les commandes effectuées.

1.3.2 Pour remplir la ligne des commandes

On peut remplir la ligne des commandes de différentes façons :
- on tape la commande en toutes lettres,
- on sélectionne la commande se trouvant dans l'un des items du menu `Scolaire`, sous menu `Tortue`,
- on clique sur l'abréviation de la commande si elle se trouve sur la barre des boutons.
- on tape le début de la commande (par exemple `ba`), puis on appuie sur la touche `tabulation` du clavier (⇆). Cela donne les différentes complétions possibles (par exemple `backquote, baisse_crayon, barycentre....`), et il reste à cliquer sur la commande voulue dans le choix proposé (par exemple `baisse_crayon`).
Il faut savoir qu'en appuyant à la fois sur `Ctrl` et sur la flèche vers le haut du clavier, on remet dans la ligne des commandes, la commande précédente. On peut faire cela éventuellement plusieurs fois pour remonter dans l'historique des commandes et si on appuie à la fois sur `Ctrl` et sur la flèche vers le bas du clavier, on descend dans l'historique des commandes. Il faut savoir aussi que `esc` efface la ligne des commandes.

1.4 Un jeu

Au début du jeu la tortue est en 150,150 avec un cap 0 et une croix verte (une salade) est mise aléatoirement sur l'écran. Les cooronnées de la salade (dans le repère défini par : l'origine est le point 150,150, l'axe des x a pour cap 0 et les unités sur les axes sont de 10 pas de tortue) se marquent en bas et à gauche de l'écran.
Le jeu consiste à deplacer la tortue en un minimum de commandes pour qu'elle mange la salade.
Les commandes disponible sont :
`avance recule tourne_droite tourne_gauche`. Dans le `jeujour` la salade est visible et dans le `jeunuit` seules les coordonn'es de la salade s'affichent.

1.4.1 Le jeu de jour

On tape :
`jeujour()`
`jeujour()`est plus difficile si on remplace 3 par 1 dans la dernière instruction.
Un petit écran s'ouvre dans lequel il y a les commandes disponibles :
`avance recule tourne_droite tourne_gauche`
Il suffit de cliquer sur une commande puis sur `OK` pour déplacer la tortue.
`Perdu` ou `Bravo` se marque au bout d'un certain nombre de coups.

1.4.2 Le jeu de nuit

On ne voit pas la salade mais on a ses coordonnées dans le repère défini par : l'origine est le point 150,150, l'axe des x a pour cap 0 et les unités sur les axes sont de 10 pas de tortue.
On tape :
`jeunuit()`.
`jeunuit()` est plus difficile si on remplace 3 par 1 dans la dernière instruction.
Un petit écran s'ouvre dans lequel il y a les commandes disponibles :
`avance recule tourne_droite tourne_gauche`
Il suffit de cliquer sur une commande puis sur `OK` pour déplacer la tortue.
`Perdu` ou `Bravo` se marque au bout d'un certain nombre de coups.

1.4.3 Les programmes

```
essai(pos,nmax):={
 local n,j,t;
 n:=1;
 for (j:=1;j<=nmax;j++){
   if (choosebox("commande",[avance,recule,tourne_droite,
        tourne_gauche,fin],n)==undef)
     {crayon rouge;ecris "Interrompu!";croix(pos);return j;}
   if (n==5)
     {crayon rouge;ecris "Interrompu!";croix(pos);return j;}
   if (n==1) { avance; }
   if (n==2) { recule; }
   if (n==3) { tourne_droite; }
   if (n==4) { tourne_gauche; }
   if (position==pos){
     crayon rouge;
     ecris " Bravo!";
     croix(pos);
     return j;
   }
 }
 crayon rouge;
 ecris " Perdu!";
 croix(pos);
 return j;
};


croix(pos):={
  local t;
  t:=position;
  crayon vert;
```

```
  leve_crayon;
  position(pos);
  baisse_crayon;
  repete(4,avance,recule,tourne_droite);
  leve_crayon;
  position(t);
  crayon bleu;
  baisse_crayon;
};

jeujour():={
  local hx,hy,p,b;
  efface;
  hx:=hasard(29)
  hx:=hx-15;
  hy:=hasard(25);
  hy:=hy-15;
  p:=[hx*10+150,hy*10+150];
  croix(p);
  signe([hx,hy]);
  crayon bleu;
  si (a==essai(p,3+abs(hx)+abs(hy));
  // mettre +1 au lieu de + 3 pour optimum
};

jeunuit():={
  local hx,hy,p;
  efface;
  hx:=hasard(29)
  hx:=hx-15;
  hy:=hasard(25);
  hy:=hy-15;
  p:=[hx*10+150,hy*10+150];
  crayon vert;
  signe([hx,hy]);
  crayon bleu;
  essai(p,3+abs(hx)+abs(hy));
  // mettre +1 au lieu de + 3 pour optimum
};
```

Chapitre 2

Les activités en CP

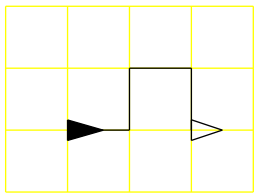
2.1 Généralités

Le but est d'écrire des petites procédures pour les utiliser ensuite pour faire un dessin plus compliqué.
Il faut savoir que les procédures écrites par l'utilisateur ne sont utilisables qu'avec des parenthèses, même pour des procédures sans paramètre, par exemple :
`truc()` si `truc` n'a pas de paramètre, ou
`truc(30)` si `truc` a un paramètre, ou
`truc(30,4)` si `truc` a deux paramètres etc...
Il faut comprendre qu'une procédure n'est pas un dessin figé mais qu'à partir d'une position de la tortue (dite position de départ) la procédure fera effectuer un tracé à la tortue et l'amenera à une autre position (dite position de d'arrivée).
C'est pourquoi, on associera toujours à une procédure :

- le tracé
- la position de départ sera notée avec un triangle plein si celle-ci est différente de la position de d'arrivée,
- la position de d'arrivée sera visible avec le triangle représentant la tortue.

Par exemple voici la procédure `truc` et son dessin (1 carreau = 10 pas) :

```
truc():={
avance;
tourne_gauche;
avance;
tourne_droite;
avance;
tourne_droite;
avance;
tourne_gauche;
}
```

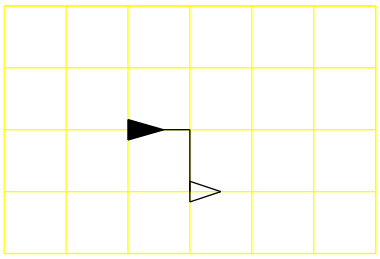


2.2 Une marche et des escaliers

2.2.1 Une marche

La procédure et son dessin :

```
marche():={
avance;
tourne_droite;
avance;
tourne_gauche;
}
```

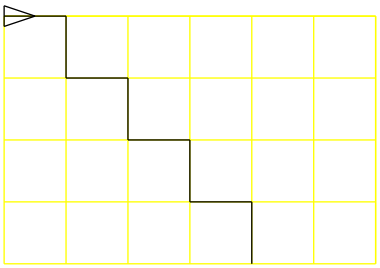


2.2.2 Un escalier

La procédure `escalier4()` dessine un escalier de 4 marches et raméne la tortue à son point de départ :

```
escalier4():={
repete(4,marche());
pas_de_cote(10*4);
saute(-10*4);
}
```

et le dessin `escalier4()`.

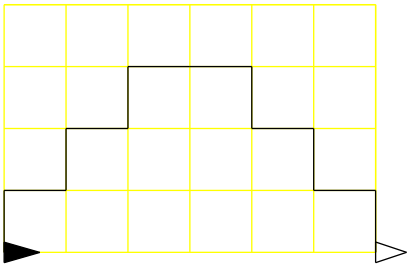


2.2.3 Un éscalier double et sa frise

La procédure `escaliers3()` dessine un escalier double de 3 marches :

```
escaliers3():={
tourne_gauche;
repete(3,marche());
tourne_droite;
repete(3,marche());
}
```

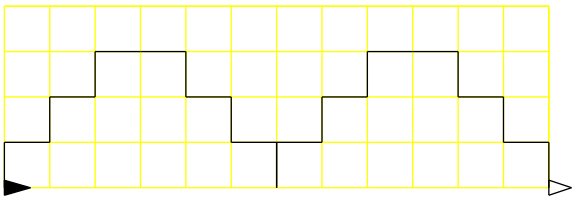
et le dessin `escaliers3()`.



La procédure `frise_escal23()` :

```
frise_escal23():={
repete(2,escaliers3());
}
```

et le dessin correspondant :

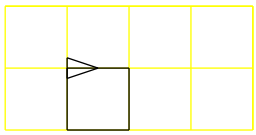


2.3 Des carrés et une croix

`carre` est un mot reservé. On écrit la procédure `carred` (resp `carreg`) qui dessine un carré à droite (resp à gauche de la position de départ de la tortue.

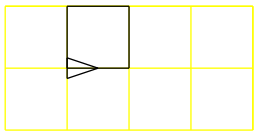
La procédure et le dessin :

```
carred():={
repete(4,avance,tourne_droite);
}
```



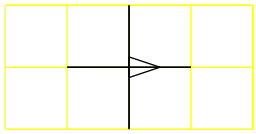
La procédure et le dessin :

```
carreg():={
repete(4,avance,tourne_gauche);
}
```



La procédure et le dessin :

```
croix():={
repete(4,avance,recule,tourne_droite);
}
```

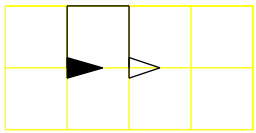


2.4 Un créneau et un chateau

2.4.1 Une porte et un créneau

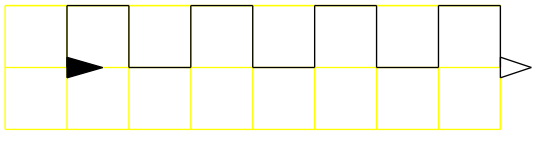
La procédure et le dessin :

```
porte():={
tourne_gauche;
avance;tourne_droite;
avance;tourne_droite;
avance;tourne_gauche;
};
```

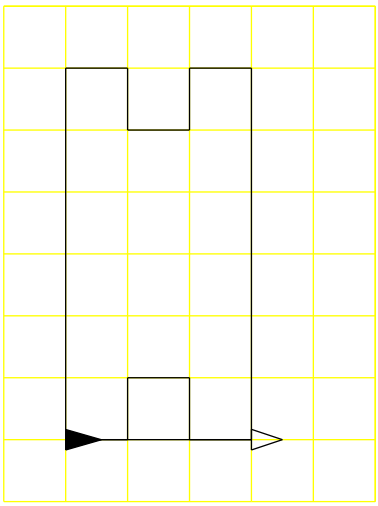


La procédure et le dessin `creneau(3)` :

```
creneau(n):={
repete(n,porte(),avance);
porte();
};
```



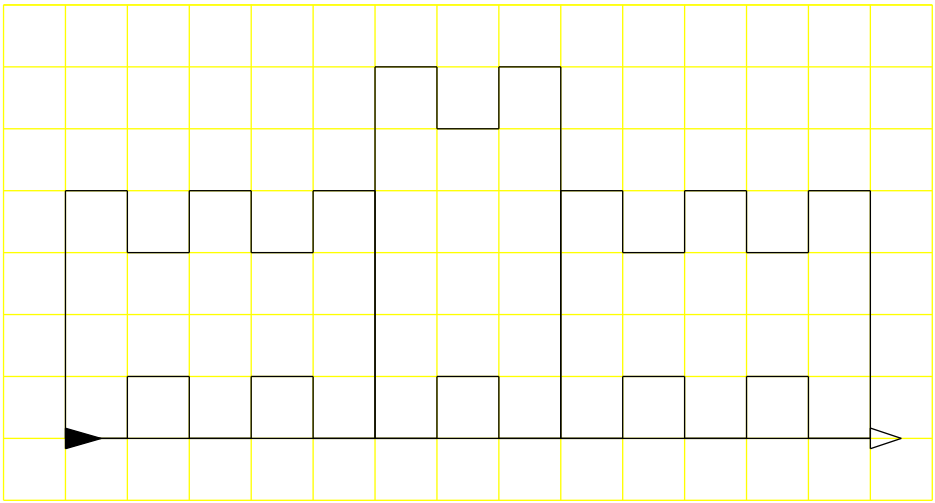
2.4.2 La porte, la tour et le chateau



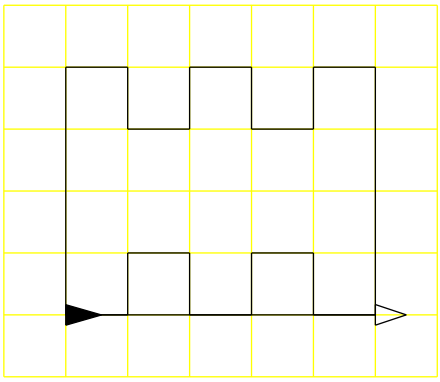
On va utiliser la procédure `porte`.
Le dessin de la tour et sa procédure :

```
tour():={
avance;porte();
avance;tourne_gauche;
avance 60;tourne_gauche;
avance;porte();
avance;tourne_gauche;
avance 60;tourne_gauche;
avance 30;
};
```

Le dessin du chateau :



Pour faire le chateau, on est donc amené à écrire une procédure `aile` et on utilisera la procédure `tour`.
Le dessin `aile` et sa procédure :



```
aile():={
avance;porte();avance;porte();
avance;tourne_gauche;
avance 40;tourne_gauche;
```

```
avance;porte();avance;porte();
avance;tourne_gauche;
avance 40;tourne_gauche;
avance 50;
};
```

Puis on écrit la procédure `chateau`.

```
chateau():={
aile();
tour();
aile();
};
```

2.4.3 Prolongements

Le dessin du chateau et de son reflet dans l'eau s'imaginent facilement !
La procédure :

```
reflet_chateau():={
chateau();
tourne_gauche;
tourne_gauche;
chateau();
};
```

Une frise de 4 chateaux :

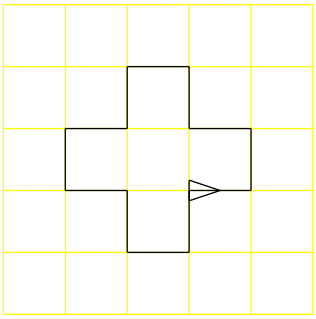
```
frise_chateau():={
repete(4,chateau());
};
```

Ou encore un chateau avec une tour de chaque coté de l'aile :

```
chateau2():={
tour();
aile();
tour();
};
```

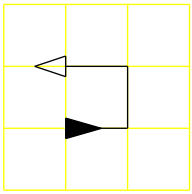
2.5 Une croix de pharmacie

Voici le dessin à réaliser :



On repère le motif qui se répète et on trouve, par exemple, `croisillon`.
Le dessin `croisillon()` et sa procédure :

```
croisillon():={
avance;tourne_gauche;
avance;tourne_gauche;
avance;
}
```



Puis on écrit la procédure `croix_ph` qui dessinera le dessin à réaliser.

```
croix_ph():={
repete(4,croisillon(),tourne_droite);
}
```

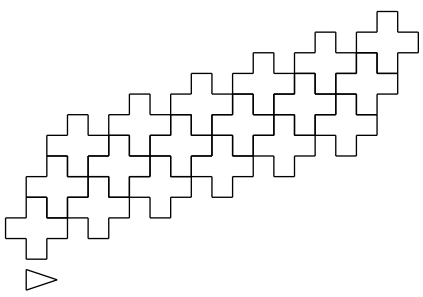
On peut ensuite faire des frises avec `croix_ph` par exemple :

```
frise_croix():={
repete(6,croix_ph(),saute(40));
saute(-240);
}
```

ou encore une frise permettant de faire un pavage avec `croix_ph`, par exemple :

```
frise_pavage_croix():={
repete(6,croix_ph(),saute(30),pas_de_cote 10);
saute(-190);
pas_de_cote -80;
};
```

et on obtiendra un pavage en tapant `frise_pavage_croix()` plusieurs fois par exemple, 3 fois :



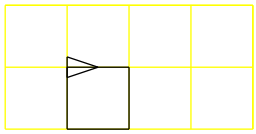
2.6 Un train

On fait tout d'abord un train avec des roues carrées et avec des fenêtres carrées.

2.6.1 Un carré

On écrit la procédure `carred` qui dessine le carré qui va nous servir à faire les roues et les fenêtres du train. On choisit `carred` qui trace un carré à droite de la tortue.
Le dessin `carred()` et sa procédure :

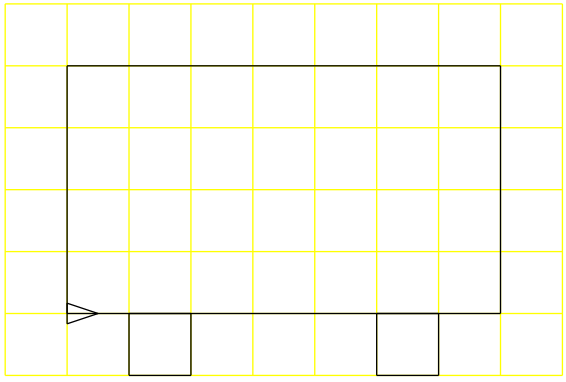
```
carred():={
repete(4,avance,tourne_droite);
}
```



2.6.2 Le wagon de base

On fait une procédure `wagon0` dans laquelle la tortue revient à son point de départ.
Le dessin `wagon0()`
et sa procédure :

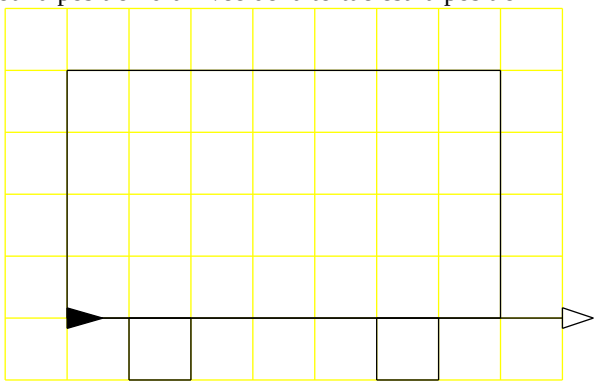
```
wagon0():={
avance;carred();
avance 40; carred();
avance 20;tourne_gauche;
avance 40;tourne_gauche;
avance 70;tourne_gauche;
avance 40;tourne_gauche;
}
```



2.6.3 Un wagon de marchandise

On fait une procédure `wagon1` où la position d'arrivée de la tortue est la position de départ d'un autre wagon.
Le dessin `wagon1()`
et sa procédure :

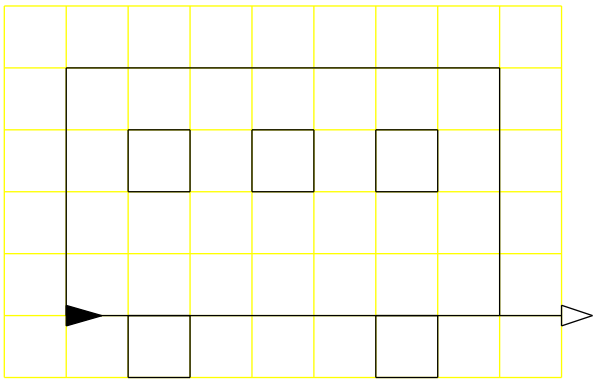
```
wagon1():={
wagon0();
avance 80;
}
```



2.6.4 Un wagon de voyageurs

On fait une procédure `wagon2` (avec des fenêtres dans laquelle la position d'arrivée de la tortue est la position de départ d'un autre wagon.
Le dessin `wagon2()`
et sa procédure :

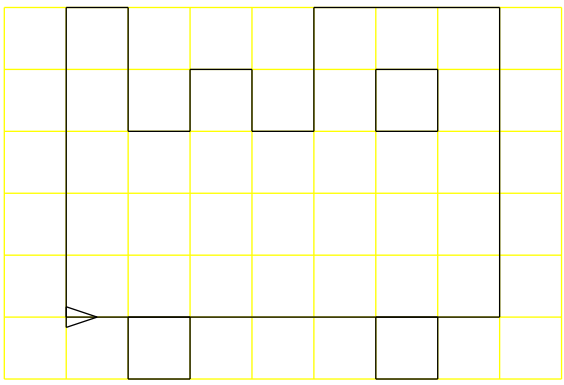
```
wagon2():={
wagon0();
pas_de_cote(30);
saute;
repete(3,carred(),
       saute(20));
pas_de_cote(-30);
avance;
}
```



2.6.5 La locomotive

On fait une procédure `loco` dans laquelle la position d'arrivée de la tortue est la même que sa position de départ.
Le dessin `loco()` et sa procédure :

```
loco():={
avance;carred();
avance 40; carred();
avance 20;tourne_gauche;
avance 50;tourne_gauche;
avance 30;tourne_gauche;
avance 20;tourne_droite;
saute(-20);carred();
saute(20);
avance;tourne_droite;
avance;tourne_gauche;
avance;tourne_gauche;
avance;tourne_droite;
avance;tourne_droite;
avance(20);tourne_gauche;
avance;tourne_gauche;
avance 50;tourne_gauche;
}
```



2.6.6 Le train

On fait une procédure `train` dans laquelle on forme un train à notre guise où la position d'arrivée de la tortue est la même que sa position de départ.
Le dessin :

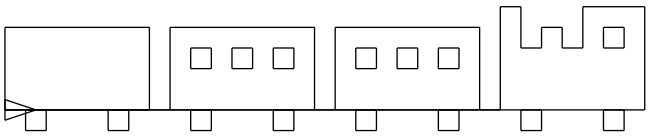
La procédure :

```
train():={
wagon1();
wagon2();
wagon2();
loco();
recule 240;
}
```

Puis on tape :

```
efface;
saute -140;
train();
```

On obtient :



2.6.7 Prolongements

Faire un train avec des roues pleines et rondes. Pour cela il suffit de remplacer `carred()` par `disque(-10)` dans `wagon0` et `loco`. Voici les procédures lorsque l'on remplace aussi les fenêtres par des carrés pleins et que l'on met de la couleur : dans ce cas il faut remplacer `carred()` par :

```
crayon bleu;
carred();polygone_rempli(-8);
crayon noir
```

car `carred()` a 8 instructions,
ou par

```
crayon bleu;
avance,rectangle_plein(-10);avance -10;
crayon noir
```

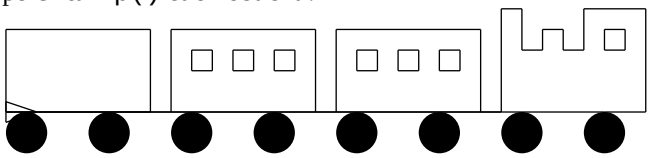
car `rectangle_plein(10)` trace un carré plein situé à gauche de la tortue et `rectangle_plein(-10)` trace le carré plein symétrique du précédent par rapport au point où se trouve la tortue.

```
wagon0p():={
```

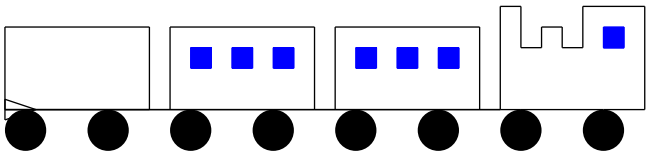
```
avance;disque(-10);
avance 40;disque(-10);
avance 20;tourne_gauche;
avance 40;tourne_gauche;
avance 70;tourne_gauche;
avance 40;tourne_gauche;
};
wagon1p():={
wagon0p();
avance 80;
};
wagon2p():={
wagon0p();
pas_de_cote(30);
saute;
repete(3,crayon bleu,avance,rectangle_plein(-10),recule,
crayon noir,saute(20));
pas_de_cote(-30);
crayon noir;
avance;
};
locop():={
avance;disque(-10);
avance 40; disque(-10);
avance 20;tourne_gauche;
avance 50;tourne_gauche;
avance 30;tourne_gauche;
avance 20;tourne_droite;
saute(-20);
crayon bleu;avance;rectangle_plein(-10);recule;crayon noir;
saute(20);
avance;tourne_droite;
avance;tourne_gauche;
avance;tourne_gauche;
avance;tourne_droite;
avance;tourne_droite;
avance(20);tourne_gauche;
avance;tourne_gauche;
avance 50;tourne_gauche;
};
trainp():={
saute(-120);
wagon1p();
wagon2p();wagon2p();
locop();
recule 160;
```

```
}
```

On tape `trainp()` et on obtient :



ou



2.7 Les couleurs du crayon

Dessiner un grand carré formé de 16*16 carrés de couleur différentes.

```
palette():={
local j,k,n;
efface();saute -140;pas_de_cote 170;
n:=0;
pour j de 0 jusque 15 faire
pour k de 0 jusque 15 faire
crayon(n);
rectangle_plein(20);
avance(20);
n:=n+1;
fpour
saute(-16*20);
pas_de_cote(-20);
fpour
}
```

Dessiner deux grands rectangles formés chacun de 8*16 carrés de couleur différentes.

```
palette2():={
local j,k,n;
efface();saute -140;pas_de_cote 170;
n:=0;
pour j de 0 jusque 15 faire
pour k de 0 jusque 7 faire
crayon(n);
rectangle_plein(20);
avance(20);
```

```
n:=n+1;
fpour
saute(-8*20);
pas_de_cote(-20);
fpour
pas_de_cote(16*20);
saute(10*20);
pour j de 0 jusque 15 faire
pour k de 0 jusque 7 faire
crayon(n);
rectangle_plein(20);
avance(20);
n:=n+1;
fpour
saute(-8*20);
pas_de_cote(-20);
fpour
}
```

2.8 Les carrés et les rectangles pleins

Lorsque la primitive `rectangle_plein` a un paramètre, elle dessine, à gauche de la tortue, un carré plein de côté le paramètre,
Lorsque la primitive `rectangle_plein` a deux paramètres, elle dessine, à gauche de la tortue, un rectangle plein de côtés les paramètres.

Remarque

Si on a écrit par exemple :

```
bord_carred(a) :=repete(4,avance(a),tourne_droite);
```

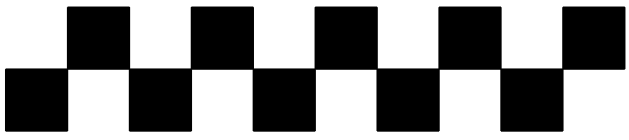
et que l'on veut remplacer le carré tracé à droite de la tortue, par un carré plein, il faut :
soit substituer à `bord_carred(a)` :

```
avance(a);rectangle_plein(-a);recule(a);
```

soit écrire : `bord_carred(a);polygone_rempli(-8)`

2.8.1 Une frise avec des carrés pleins

Le dessin :



On tape par exemple :

```
repete(5,rectangle_plein,saute,pas_de_cote,rectangle_plein,
       saute,pas_de_cote -10);
```

ou encore

```
repete(5,saute,rectangle_plein,tourne_droite 180,rectangle_plein,
       tourne_gauche 180,saute);
```

ou encore

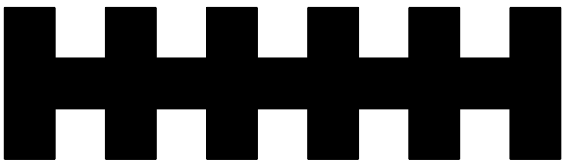
```
repete(5,rectangle_plein,tourne_droite 180,rectangle_plein,
       tourne_droite 180,saute 20);
```

ou encore

```
repete(5,rectangle_plein,rectangle_plein -10,saute 20);
```

2.8.2 Une frise avec des carrés et des rectangles pleins

Le dessin :

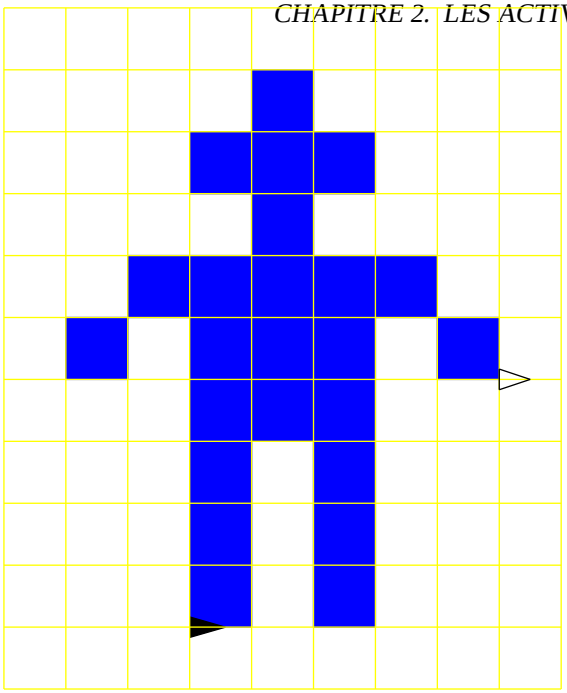


On tape par exemple :

```
repete(5,rectangle_plein(10,30),saute,pas_de_cote 10,
       rectangle_plein,pas_de_cote -10,saute);
```

2.9 Le bonhomme

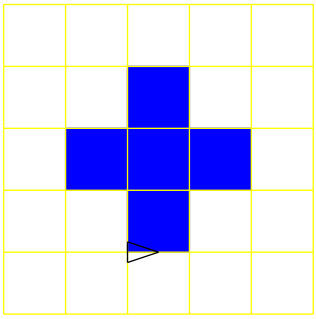
Le dessin :



On décompse ce dessin en : `tete, bras, corps, jambes`.

2.9.1 La tête

Le dessin :

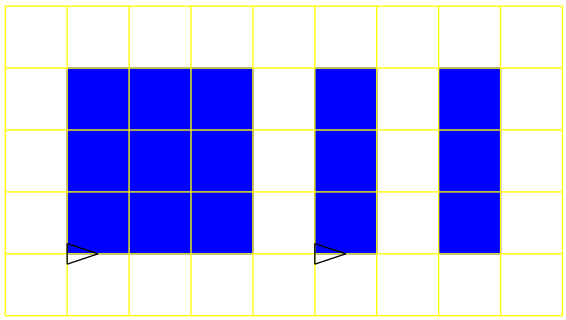


La procédure :

```
tete():={
pas_de_cote;
rectangle_plein;
repete(4,saute,rectangle_plein,tourne_gauche)
pas_de_cote(-10);
};
```

2.9.2 Le corps et les jambes

Les dessins :
`corps()` et `jambes()` :

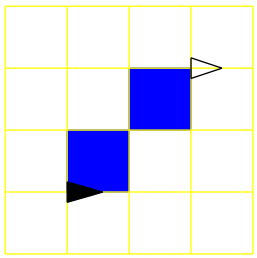


Les procédures :

```
corps():={
rectangle_plein(30);
};

jambes():={
rectangle_plein(10,30);
saute 20;
rectangle_plein(10,30);
saute -20;
};
```

2.9.3 Un bras

Le dessin peut faire l'un ou l'autre bras :



La procédure :

```
bras():={
rectangle_plein;
saute;
pas_de_cote 10;
rectangle_plein;
saute;
pas_de_cote 10;
};
```

2.9.4 Le bonhomme

On commence par faire ses jambes et on termine par son bras gauche :

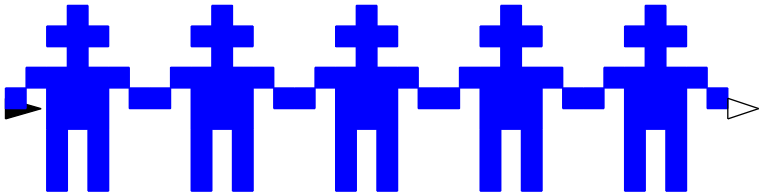
```
bonhomme():={
jambes();
```

```
pas_de_cote 30;
corps();tourne_gauche;
saute;
pas_de_cote 20;
tourne_droite;
bras();
saute;
tete();
saute 20;
tourne_droite;
bras();
tourne_gauche;
};
```

Si on veut faire une ribambelle avec ce bonhomme il faudra placer correctement la tortue avant de taper `bonhomme` : il serait donc plus astucieux de faire le bonhomme en commencant par un bras et de terminer par l'autre bras. Donc on tape :

```
bonhomme_rib():={
bras();
saute;
tete();
pas_de_cote -30;
saute -10;
corps();
pas_de_cote -30;
jambes()
saute 30;
pas_de_cote 60;
tourne_droite;
bras();
tourne_gauche;
}
```

On tape :
`repete(5,bonhomme_rib())`
On obtient :

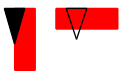


2.9.5 Prolongements

Faire une ribambelle de bonhommes avec un bras levé et un bras baissé et une jambe levée et une jambe baissée etc...

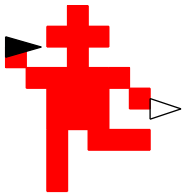
On tape pour avoir une jambe levée et une jambe baissée :

```
jambes_bl():={
rectangle_plein(30,10);
pas_de_cote 20;
rectangle_plein(10,30);
pas_de_cote 10;
};
```



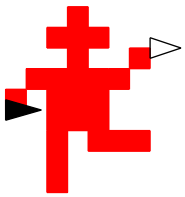
On tape pour avoir un bonhomme avec un bras levé, un bras baissé, une jambe levée et une jambe baissée :

```
bonhomme_lb():={
tourne_droite;
bras();
saute -10;
tourne_gauche;
saute;
tete();
pas_de_cote -30;
saute -10;
corps();
tourne_droite;
jambes_bl()
saute -30;
bras();
tourne_gauche;
};
```



On tape pour avoir un bonhomme avec un bras baissé, un bras levé, une jambe levée et une jambe baissée :

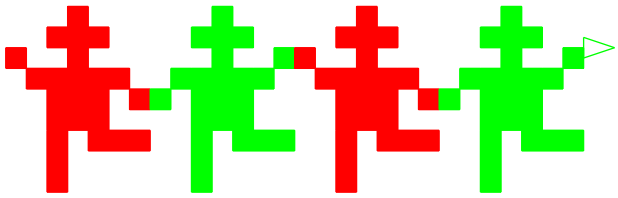
```
bonhomme_bl():={
bras();
saute;
tete();
pas_de_cote -30;
saute -10;
corps();
tourne_droite;
jambes_bl()
saute -20;
tourne_gauche;
bras();
};
```



Puis pour avoir une ribambelle :

```
repete(2,crayon rouge,bonhomme_lb(),crayon vert,bonhomme_bl())
```

On obtient :



2.10 Les triangles pleins

2.10.1 Des frises avec des triangles équilatéraux et pleins

Le dessin :

On tape :

```
repete(6,crayon(hasard(7)),triangle_plein 30,avance 30)
```

Le dessin :

On tape :

```
repete(6,crayon(hasard(7)),triangle_plein 30,avance 15)
```

Le dessin :

On tape :

```
repete(6,
      (repete(2,crayon(hasard(7)),triangle_plein 30,saute -15)),
       saute 60)
```

2.10.2 Une étoile

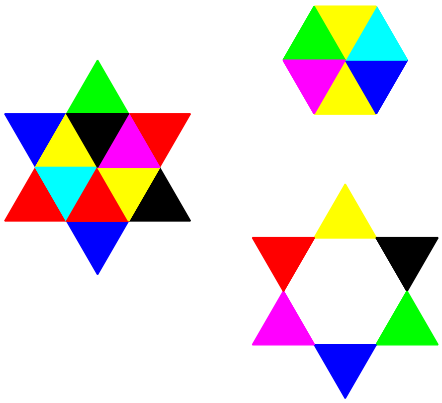
Une étoile faite avec deux triangles équilatéraux.
Le dessin :

On tape :

```
etoile():={
crayon(jaune);
triangle_plein(90);
tourne_gauche 60;
avance 30;
tourne_droite 120;
avance -30;
triangle_plein(90);
avance 30;
tourne_gauche 120;
avance -30;
tourne_droite 60;
}
```

2.10.3 L'étoile et l'hexagone

Les dessins :



On peut faire un hexagone plein et multicolore de différentes façons :
- on part du centre, on tape :

```
repete(6,crayon(hasard(7)),triangle_plein(30),tourne_gauche 60)
```

ou :

```
repete(6,crayon(hasard(7)),triangle_plein(30),tourne_droite 60)
```

- on part du bord, on tape :

```
repete(6,crayon(hasard(7)),triangle_plein(30),avance 30,
        tourne_gauche 60)
```

ou :

```
repete(6,crayon(hasard(7)),tourne_droite 60,triangle_plein(30),
        tourne_gauche 60,avance 30,tourne_droite 60)
```

Pour faire les branches multicolores d'une étoile, on tape

```
repete(6,crayon(hasard(7)),triangle_plein(30),
       avance 30,tourne_droite 60)
```

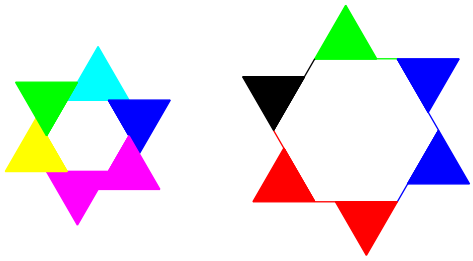
Pour faire une étoile multicolore, on reunit les deux dessins précédent, on tape pour avoir le centre de l' étoile à gauche de la tortue :

```
repete(6,crayon(hasard(7)),triangle_plein(30),
        avance 30,tourne_gauche 60);
tourne_gauche 120;
repete(6,crayon(hasard(7)),triangle_plein(30),
       avance 30,tourne_droite 60);
tourne_droite 120;
```

ou pour avoir le centre de l' étoile à droite de la tortue :

```
repete(6,crayon(hasard(7)),triangle_plein(30),
       avance 30,tourne_droite 60);
tourne_droite 120;
repete(6,crayon(hasard(7)),triangle_plein(30),
        avance 30,tourne_gauche 60);
tourne_gauche 120;
```

D'autres dessins :



On tape :

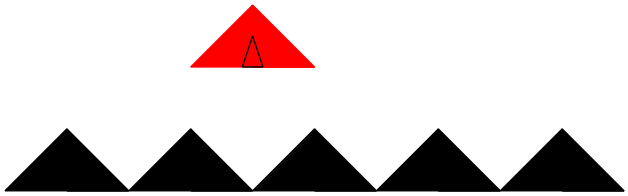
```
repete(6,crayon(hasard(7)),triangle_plein(30),
       avance 20,tourne_droite 60)
```

et

```
repete(6,crayon(hasard(7)),triangle_plein(30),
       avance 40,tourne_droite 60)
```

2.10.4 Une frise avec des triangles pleins

On veut faire une frise avec des triangles rectangles isocéles posés sur leur hypothénuse.
Le dessin du triangle `tri()` et de la frise :

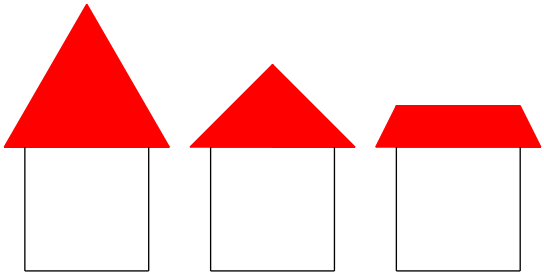


On tape :

```
tri():={
triangle_plein(30,30);
tourne_droite;
triangle_plein(30,30);
tourne_gauche;
}

frise_tri():={
repete(5,tri(),pas_de_cote 30);
};
```

2.10.5 Une maison et son toit avec des triangles pleins

Une maison est un carré de côté 60 surmonté d'un toit.
Le toit est un triangle équilatéral de côt'e 80
ou un triangle rectangle isocéle posé sur son hypoténuse de longueur 80
ou un trapèze isocéle de grande base 80 et de hauteur 20.
Les dessins :



Les procedures pour le toit : La tortue part horizontalement en haut du mur de gauche et arrive horizontalement en haut du mur de droite.

```
toit1():={
```

```
crayon rouge;
recule;
triangle_plein 80;
avance 70;
};

toit2():={
crayon rouge;
avance 30;
triangle_plein(40,40);
tourne_gauche;
triangle_plein(40,40);
tourne_droite;
avance 30;
};

toit3():={
crayon rouge;
rectangle_plein(60,20);
tourne_gauche;
triangle_plein(20,10);
tourne_droite;
avance 60;
triangle_plein(10,20);
};
```

La procédure pour les murs :

```
murs():={
crayon noir;
repete(3,tourne_droite,avance 60);
tourne_droite;
};
Lés proc\’edures pour les diff\’erentes maisons :
\begin{verbatim}
maison1():={
toit1();
murs();
};

maison2():={
toit2();
murs();
};

maison3():={
toit3();
murs();
};
```

On peut aussi écrire une seule procédure dont le paramètre sera le nom du toit :

```
maison_t(t):={
t();
murs();
};
```

Puis essayer :
`maison_t(toit1)`
On peut aussi paramétrer les toits, par exemple modifier `toit1` en prenant comme paramètre `n` la longueur du côté du carré représentant les murs :

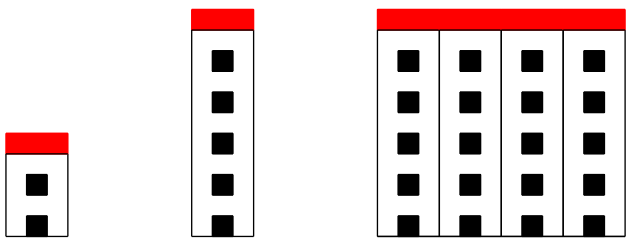
```
toit1(n):={
recule;
triangle_plein(n+20);
avance n+10;
}

maison_toit(t,n):={
t(n);
repete(3,tourne_droite,avance n);
tourne_droite;
};
```

Puis essayer :
`maison_toit(toit1,80)`

2.11 Les paramètres

2.11.1 Une maison et un immeuble

On veut dessiner une maison de deux étages ayant une fenêtre à chaque étages, puis un immeuble de 5 étages et enfin une barre de 4 immeubles de 5 étages.
Les dessins de : `maison()` `immeuble5()` et `barre4_immeuble5()` :



Les procédures :

```
maison():={
avance;
rectangle_plein;
pas_de_cote 20;
rectangle_plein;
pas_de_cote -20;
avance(20);
tourne_gauche;
avance(40);
crayon rouge;
rectangle_plein(10,30);
crayon noir;
tourne_gauche;
avance(30);
tourne_gauche;
avance(40);
tourne_gauche;
}

immeuble5():={
avance;
repete(5,rectangle_plein,pas_de_cote 20);
pas_de_cote -100;
avance(20);
tourne_gauche;
avance(100);
crayon rouge;
rectangle_plein(10,30);
crayon noir;
tourne_gauche;
avance(30);
tourne_gauche;
avance(100);
tourne_gauche;
}
```

Un barre de 4 immeubles :

```
barre4_immeuble5():={
repete(4,immeuble5(),avance(30));
};
```

On veut maintenant faire des immeubles d'un nombre quelconque d'étages. Pour cela on on modifie `immeuble5()` en remplaçant 5 par le paramétre `n`.
On écrit :

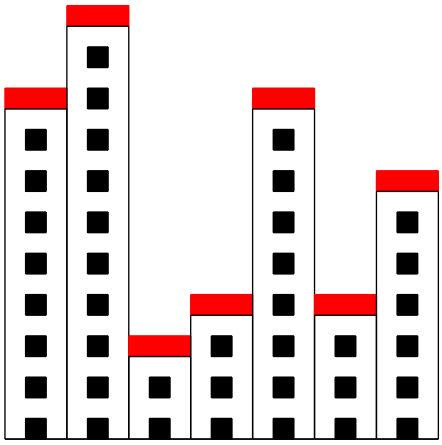
```
immeuble(n):={
```

```
avance;
repete(n,rectangle_plein,pas_de_cote 20);
pas_de_cote -n*20;
avance(20);
tourne_gauche;
avance(n*20);
crayon rouge;
rectangle_plein(10,30);
crayon noir;
tourne_gauche;
avance(30);
tourne_gauche;
avance(n*20);
tourne_gauche;
};
```

Faire une barre de `p` immeubles avec des immeubles de tailles différentes. La valeur de `n` sera choisi aléatoirement entre 2 et 10 : on a alors `n=hasard(9)+2` car `hasard(9)` est un nombre choisi aléatoirement entre 0 et 8.

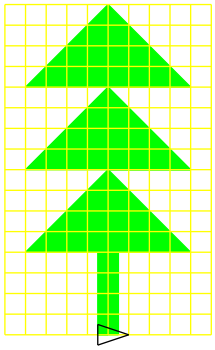
```
barre_immeubles(p):={
repete(p,immeuble(hasard(9)+2),avance 30);
};
```

puis on tape :
`barre_immeubles(7)`
On obtient :



2.11.2 Un sapin

Le dessin :

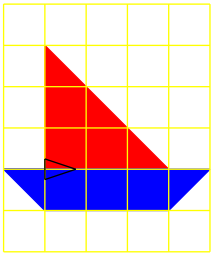


La procédure :

```
sapin():={
rectangle_plein(10,40);
tourne_gauche;
saute 40;
tourne_droite;
avance 5;
repete(3,triangle_plein(40,40),tourne_gauche,
   triangle_plein(40,40),avance 40,tourne_droite);
tourne_gauche;
recule 160;
tourne_droite;
avance -5;
}
```

2.11.3 Un bateau

Le dessin :



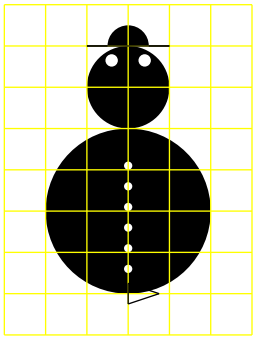
La procédure :

```
bateau():={
crayon rouge;
triangle_plein(60,60);
crayon bleu;
triangle_plein(-20,-20);
tourne_droite;
rectangle_plein(20,60);
pas_de_cote 60;
triangle_plein(20,20);
tourne_gauche;
recule 60;
};
```

2.12 Les disques

2.12.1 Un bonhomme de neige et son balai

Le dessin du bonhomme de neige :



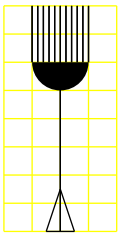
La procédure `neige()` :

```
neige():={
disque(40);
crayon blanc;
repete(6,pas_de_cote 10,disque 2);
crayon noir;
pas_de_cote 20;
disque 20;
pas_de_cote 30;
saute 8;
crayon blanc;
disque 3;
saute -16;
disque 3;
saute 8;
crayon noir;
pas_de_cote 10;
avance 20;
recule 10;
tourne_gauche;
disque(10,180);
tourne_droite;
avance 10;
recule  20;
pas_de_cote 120;
tourne_droite 180;
};
```

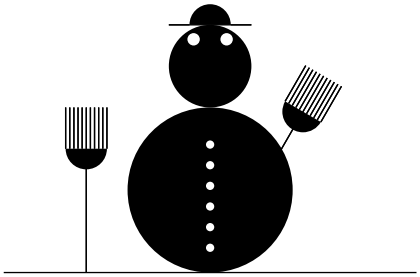
On dessine maintenant un balai.

Le dessin du balai et les procédures `poils()` et `balai()` :

```
poils():={
repete(10,avance,recule,pas_de_cote 2);
avance;
recule;
};
balai():={
avance 40;
pas_de_cote 10;
poils();
tourne droite 180;
disque(10,180);
pas-de_cote -10;
recule 40;
};
```



Le dessin final :

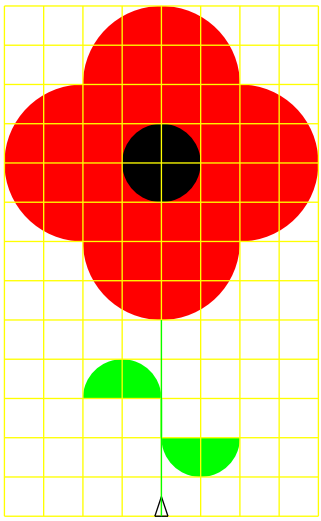


et sa procédure :

```
final():={
efface;
neige();
avance 100;
recule 200;
saute 40;
tourne_gauche ;
balai();
tourne_droite ;
saute 60;
tourne_gauche 60;
avance 30;
balai();
};
```

2.12.2 Une fleur

Voici le dessin d'une fleur fait avec des disques pleins.
On décompose la fleur en :
la procédure `petales` et
la procédure `tige`.



On écrit `petales` (la tortue au départ et à l'arrivée se trouve au centre de la fleur) :

```
petales():={
local n;
n:=crayon;
crayon rouge;
repete(4,disque 20,
       tourne_gauche);
pas_de_cote -10;
crayon noir;
disque;
pas_de_cote;
crayon n;
}
```

On écrit `tige` (la tortue au départ est à une extrémité de la tige et à l'arrivée se trouve à l'autre extrémité au centre de la fleur) :

```
tige():={
local n;
n:=crayon;
crayon vert;
avance 20;
repete(2,avance,disque(10,180),
       pas_de_cote 20);
avance 30;
saute 40
crayon n;
}
```

Puis on écrit la procédure `fleur` :

```
//commencer par : tourne_gauche; saute -90;
fleur():={
tige();
petales();
saute -90;
}
```

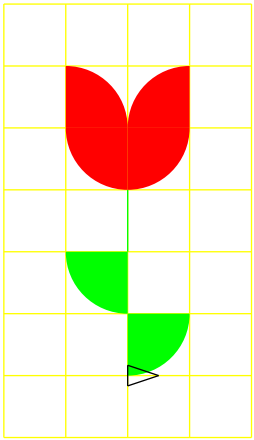
On tape pour obtenir une fleur :

```
tourne_gauche ; saute -90 ;fleur()
```

2.13 Les demi-disques et les quarts de disques

2.13.1 Une tulipe

On veut réaliser le dessin d'une tulipe :

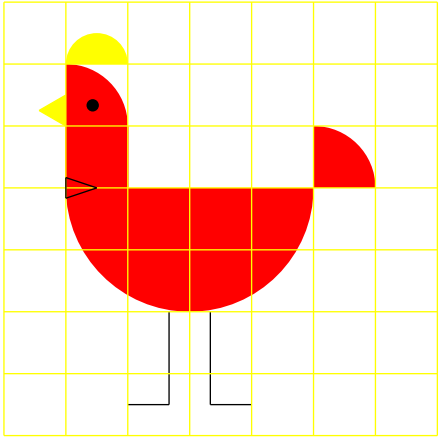


On écrit la procédure `tulipe` :

```
//faire au debut :
// pas_de_cote -90;
tulipe():={
  local n;
  n:=crayon;
  crayon vert;
  disque(20,90);
  pas_de_cote 20;
  tourne_gauche;
  disque(-20,90);
  pas_de_cote -20;
  avance 20;
  crayon rouge;
  tourne_droite;
  disque(20,90);
  pas_de_cote 20;
  disque(-20,90);
  saute -40;
  disque(-20,90);
  pas_de_cote -20;
  disque(20,90);
  tourne_gauche;
  saute -60;
  tourne_droite;
  crayon n;
};
```

2.13.2 Une poule

Voici le dessin :



On écrit par exemple :
`tete_poule` et `corps_poule` :

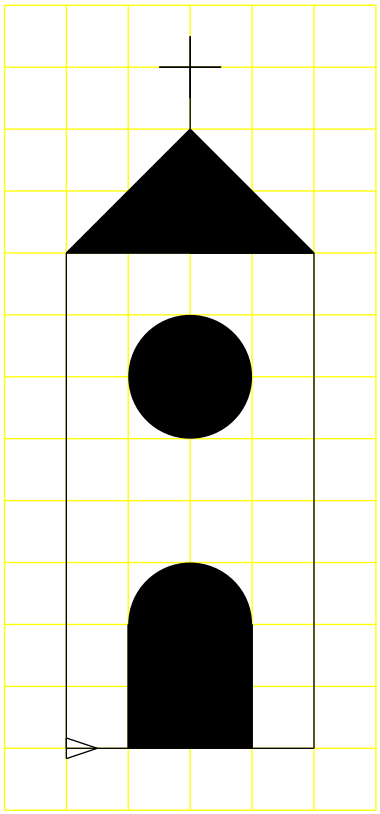
```
tete_poule():={
  local n;
  n:=crayon;
  crayon rouge;
```

```
  rectangle_plein 30;
  pas_de_cote 30;
  tourne_gauche;
  crayon jaune;
  triangle_plein 15;
  pas_de_cote -30;
  crayon rouge;
  disque(30,90);
  saute -30;
  tourne_droite;
  crayon jaune;
  disque(15,180);
  saute 20;pas_de_cote;
  crayon noir;disque(3);
  pas_de_cote -10;
  saute 40;
  crayon n;
};
corps_poule():={
  local n;
  n:=crayon;
  crayon rouge;
  disque(60,180);
  pas_de_cote -30;
  disque(30,90);
  pas_de_cote 135;
  saute 30;
  crayon noir;
  avance 20;
  tourne_droite;
  avance 45;
  pas_de_cote 20;
  recule 45;
  tourne_droite;
  recule 20;
  pas_de_cote 105;
  saute -30;
  crayon n;
};
poule():={
tete_poule();
corps_poule();
};
```

2.13.3 Un clocher

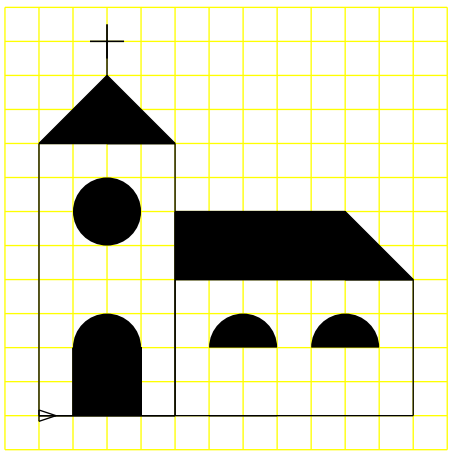
Le dessin et la procédure `clocher()` :

```
clocher():={
avance;
rectangle_plein 20;
tourne_gauche;
avance 20;
disque(-10,180);
avance 20;
tourne_gauche;
avance;
tourne_gauche;
avance 60;
pas_de_cote 10;
disque;
pas_de_cote -10;
avance 20;
tourne_droite;
avance -20;
triangle_plein(20,20);
tourne_gauche;
triangle_plein(20,20);
saute 20;
avance;
repete(4,avance 5,tourne_gauche);
saute -30;
tourne_gauche;
avance 20;
tourne_gauche;
avance 80;
tourne_gauche;
}
```



2.13.4 Une église

Le dessin :



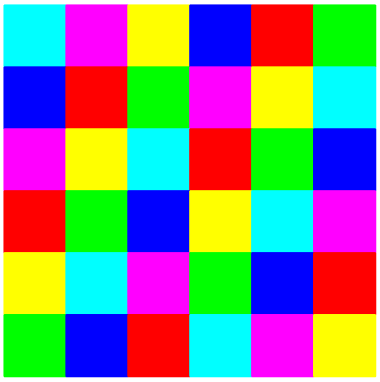
Les procédures `nef()` et `eglise()` :

```
nef():={
avance 30;
tourne_gauche;
saute 20;
disque(10,180);
tourne_gauche;
saute 30;
tourne_gauche;
disque(-10,180);
saute 20;
tourne_gauche;
saute -50;
avance 60;
tourne_gauche;
avance 40;
tourne_droite;
recule 20;
triangle_plein(20,20);
recule 50;
rectangle_plein(50,20);
tourne_gauche;
recule 40;
tourne_droite;
}
```

```
eglise():={
clocher();
avance 40;
nef();
avance -40;
}
```

2.14 Les carrés magiques

Le point de départ de cette activité est un tableau de Richard Paul Lohse vu au musée de Grenoble.
En voici une reproduction avec la tortue (les couleurs ne sont pas respectées !) :



Pour faire une ligne de carrés colorés avec la liste des couleurs comme paramètre, on tape :

```
ligne(coul):={
local j;
pour j de 0 jusque 5 faire
crayon coul[j];
rectangle_plein(30);
saute(30);
fpour
saute -180;
};
```

Les couleurs choisies pour l'impression ont comme code :
`[6,5,3,4,1,2]`
et celles choisies par Lohse ont comme code :
`[180,168,3,136,93,78]`
Le passage d'une ligne à une autre se fait à l'aide de la permutation :
`p :=[3,4,5,1,2,0]`.

```
passage(l,p):={
local n,j,lp;
n:=size(l);
lp:=l;
pour j de 0 jusque n-1 faire
lp[j]:=l[p[j]];
fpour
return lp;
};
```

On tape pour avoir la première ligne et la deuxième ligne :
`ligne([180,168,3,136,93,78])`

```
ligne(passage([180,168,3,136,93,78],p))
```
On écrit pour avoir les 6 lignes :

```
lohse():={
local j,p,coul;
coul:=[180,168,3,136,93,78];
pour j de 0 jusque 5 faire
ligne(coul);
p:=[3,4,5,1,2,0];
coul:=passage(coul,p);
pas_de_cote -30;
fpour;
pas_de_cote 180;
};
```

Plus généralement on tape pour avoir des carrés de différentes dimensions avec un choix de la couleur et le choix d'une permutation.

```
passage(l,p):={
local n,j,lp;
n:=size(l);
lp:=l;
pour j de 0 jusque n-1 faire
lp[j]:=l[p[j]];
fpour
return lp;
};
ligneg(coul):={
local n,j;
n:=size(coul);
pour j de 0 jusque n-1 faire
crayon coul[j];
rectangle_plein(30);
saute(30);
fpour
saute -30*n;
};
lohseg(coul,p):={
local n,j;
n:=size(coul);
si (n!=size(p)) return "erreur";
pour j de 0 jusque n-1 faire
ligne(coul);
coul:=passage(coul,p);
pas_de_cote -30;
fpour;
pas_de_cote 180;
```

```
};
```

Essayez :

```
coul:=[14,13,11,12,9,10,16];
p:=[4,5,6,0,1,2,3];
efface ;pas_de_cote 40;
lohseg(coul,p)
```

On va tout d'abord travailler avec des carrés de côtés 5 : on choisit 5 couleurs et on impose d'avoir dans chaque ligne et dans chaque colonne ces 5 couleurs.

Chapitre 3

Les activités en CE1

3.1 Illustration d’un poéme

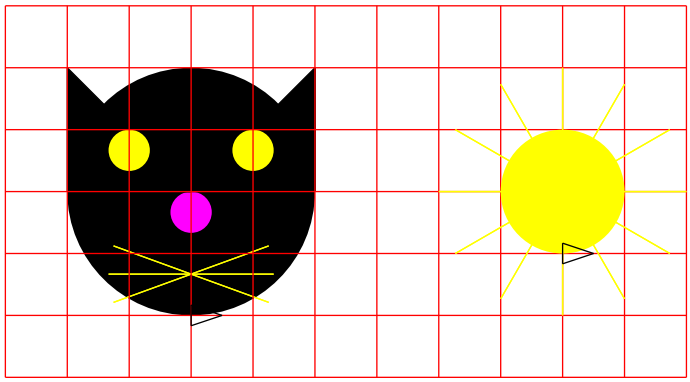
3.1.1 Le poéme

Il s’agit du poéme de Maurice Carême :

```
 Le chat et le soleil

Le chat ouvrit les yeux,
Le soleil y entra.
Le chat ferma les yeux,
Le soleil y resta.

Voilà pourquoi, le soir,
Quand le chat se réveille,
J’aperçois dans le noir
Deux morceaux de soleil.

Maurice Carême
```

3.1.2 Les dessins du chat et du soleil

Voici les dessins où 1 carreau vaut 30 pas de tortue.



3.1.3 La procédure chat

On tape :

```
chat():={
disque(60);
pas_de_cote 20;
crayon jaune;
tourne_gauche 20;
repete(3,avance 40,recule 80,avance 40,tourne_droite 20);
tourne_gauche 40;
pas_de_cote 20;
crayon magenta;
disque;
crayon noir;
pas_de_cote 20;
saute -60;
triangle_plein(60,60);
saute 120;
tourne_gauche;
triangle_plein(60,60);
pas_de_cote 20;
crayon jaune;
saute 20;
disque;
pas_de_cote 60;
disque;
crayon noir;pas_de_cote -20;
saute -80;tourne_droite;
};
```

3.1.4 La procédure soleil

On tape :

```
soleil:={
crayon jaune;
disque 30;
pas_de_cote 30;
repete(12,avance 60,recule 60,tourne_droite 30);
pas_de_cote -30;
};
```

3.1.5 La poésie illustrée

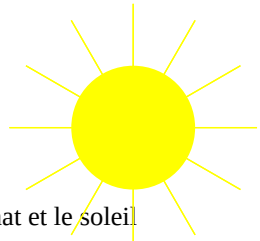
On tape la procédure `poesie` :

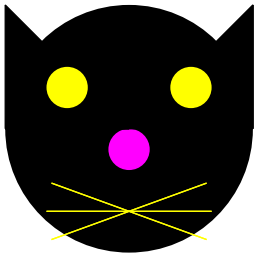
```
poesie(a,b):={
ecris("Le chat et le soleil",20,a+15,b);
ecris("Le chat ouvrit les yeux,",20,10,b-20);
ecris("Le soleil y entra.",20,10,b-40);
ecris("Le chat ferma les yeux,",20,10,b-60);
ecris("Le soleil y resta.",20,10,b-80);
ecris("Voilà pourquoi, le soir,",20,a,b-120);
ecris("Quand le chat se réveille,",20,a,b-140);
ecris("J'aperçois dans le noir",20,a,b-160);
ecris("Deux morceaux de soleil.",20,a,b-180);
ecris("Maurice Carême",20,a+15,b-200);
};
```

La procédure `dessin_poesie` :

```
dessin_poesie():={
poesie(10,210);
pas_de_cote 80;
saute -30;crayon jaune;
soleil();
saute 170;
pas_de_cote -140;
chat();
};
```

On tape `dessin_poesie()` et on obtient :





Le chat et le soleil

Le chat ouvrit les yeux,
Le soleil y entra.
Le chat ferma les yeux,
Le soleil y resta.

Voilà pourquoi, le soir,
Quand le chat se réveille,
J'aperçois dans le noir
Deux morceaux de soleil.

Maurice Carême

3.2 Faire unécran coloré

```
fond(n):={
local c,p;
c:=crayon;
p:=position;
position [190,190];
crayon(n);
repete(4,rectangle_plein(190),tourne_droite);
position p;
crayon c;
};
```

Chapitre 4

Les activités pour les CE2

4.1 Introduction des paramètres

4.1.1 Des escaliers de toutes les tailles

On a déja défini (cf 2.2.1) la procédure `marche()` et la procédure `escalier4()` :

```
marche():={
avance;
tourne_droite;
avance;
tourne_gauche;
};

escalier4():={
repete(4,marche());
pas_de_cote(10*4);
saute(-10*4);
};
```

Si on veut faire un escalier de 5 marches il faudrait écrire la procédure :

```
escalier5():={
repete(5,marche());
pas_de_cote(10*5);
saute(-10*5);
}
```

Mais cela n'est pas très commode car il faut écrire une nouvelle procédure alors qu'il suffit de modifier une valeur. On a la possibilité de remplacer cette valeur par un paramètre : le nom de ce paramètre doit figurer entre des parenthèses apres le nom de la procédure, par exemple `escalier(n)` dont l'exécution se fera en tapant `escalier(5)`. La tortue remplacera alors `n` par `5` dans toutes les instructions définissant la procédure.

Voici la procédure `escalier4()` et sa transformation en `escalier(n)` pour avoir un escalier de `n` marches :

```
escalier4():={
repete(4,marche());
pas_de_cote(10*4);
saute(-10*4);
};
```

```
escalier(n):={
repete(n,marche());
pas_de_cote(10*n);
saute(-10*n);
};
```

4.1.2 Des escaliers avec des marches de toutes les tailles

Voici la procédure `marche()` et sa transformation en `marches(p,q)` pour avoir une marche de largeur `p` et de hauteur `q` :

```
marche():={
avance;
tourne_droite;
avance;
tourne_gauche;
};
```

```
marches(p,q):={
avance p;
tourne_droite;
avance q;
tourne_gauche;
};
```

Voici enfin la transformation de `escalier(n)` en `escalier(n,p,q)` pour avoir un escalier de `n` marches de largeur `p` et de hauteur `q` :

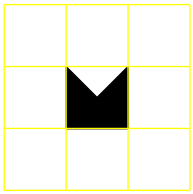
```
escalier(n):={
repete(n,marche());
pas_de_cote(10*n);
saute(-10*n);
};
```

```
escalier(n,p,q):={
repete(n,marche(p,q));
pas_de_cote(10*n);
saute(-10*n);
};
```

4.2 Le bonnet d'âne

4.2.1 Le dessin et sa procédure

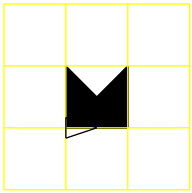
On veut réaliser le dessin suivant :



On peut voir dans ce dessin de différentes façons : un rectangle surmonté de deux triangles rectangles isocèles ou ...ou encore deux triangles rectangles isocèles qui se

superposent. On choisit la dernière déscription et on écrit la procédure `ane()` correspondant au dessin :

```
ane():={
triangle_plein(30,30);
avance 30;
tourne_gauche ;
triangle_plein(30,30);
tourne_droite;
recule 30;
};
```



4.2.2 Des dessins utilisant `ane()`

On a écrit précédemment la procédure `ane()` et on l'utilise pour faire :

La couronne du roi :

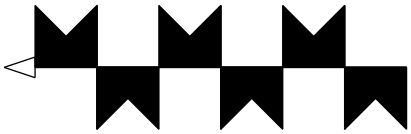
```
repete(5,ane(),saute 30)
```

La couronne de la reine :

```
repete(3,ane(),saute 15,
         ane(),saute 30)
```

Une frise :

```
saute -120;
repete(3,ane(),saute 60);
tourne_droite 180;
repete(3,ane(),saute 60);
```

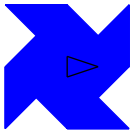


Une autre frise :

```
saute -120;
crayon rouge;
repete(3,ane(),saute 60);
pas_de_cote 30;
tourne_droite 180;
crayon vert;
repete(3,ane(),saute 60);
```

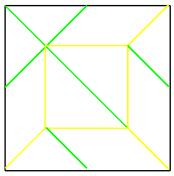
Un tourniquet :

```
repete(4,ane(),tourne_gauche)
```

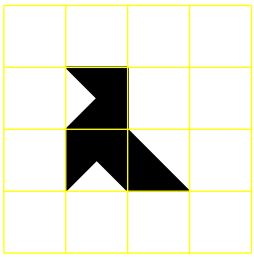


4.3 La cocotte en papier

On fait le pliage correspondant à une cocotte en papier. On prend une feuille de papier carrée et on fait des plis en creux selon les traits jaunes et des plis dans l'autre sens les traits verts du dessin :

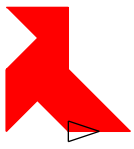


Puis on forme la cocotte (la tête est le triangle situé en haut à gauche) on obtient :



On écrit la procédure `cocot()` qui utilise la procédure `ane()` de la section précédante. On choisit le départ et l'arrivée selon la flèche du dessin :

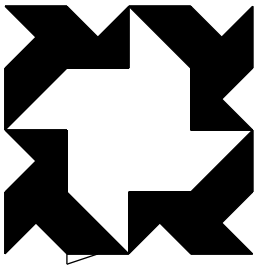
```
cocot():={
triangle_plein(30,30);
tourne_gauche ;
avance 30;
ane();
tourne_gauche ;
ane();
tourne_droite;
recule 30;
tourne_droite;
};
```



4.3.1 Un dessin

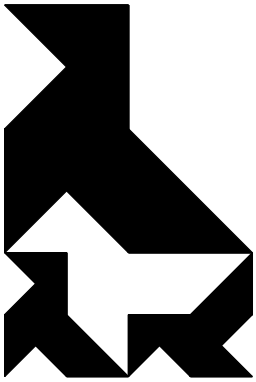
Voici un dessin et sa procédure :

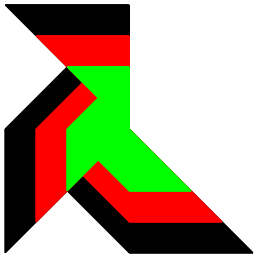
```
repete(4,cocot(),saute 90,
    tourne_gauche 90,saute 30)
```



4.4 Le bonnet d'âne et les cocottes avec un paramètre

On veut réaliser les dessins :





On transforme alors les procédures `ane` et `cocot` pour en faire des procédures paramétrées en rajoutant un `s` aux noms des procédures et le nom du paramètre dans les parenthèses. Puis on remplace 30 par `n` sans oublier d'appeler `anes` avec `n` comme paramètre.
Voici les procédures `ane()` et `anes(n)` :

```
ane():={
triangle_plein(30,30);
avance 30;
tourne_gauche ;
triangle_plein(30,30);
tourne_droite;
recule 30;
};
```

```
anes(n):={
triangle_plein(n,n);
avance n;
tourne_gauche ;
triangle_plein(n,n);
tourne_droite;
recule n;
};
```

Voici les procédures `cocot()` et `cocots(n)` :

```
cocot():={
triangle_plein(30,30);
tourne_gauche ;
avance 30;
ane();
tourne_gauche ;
ane();
tourne_droite;
recule 30;
tourne_droite;
};
```

```
cocots(n):={
triangle_plein(n,n);
tourne_gauche ;
avance n;
anes(n);
tourne_gauche ;
anes(n);
tourne_droite;
recule n;
tourne_droite;
};
```

Puis on écrit les procédures :

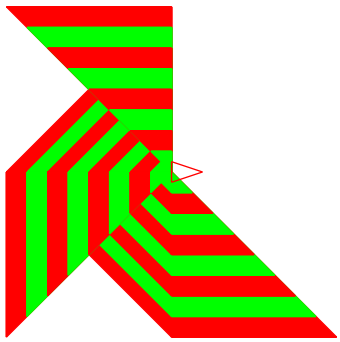
```
cocots_fam():={
cocots(60);
pas_de_cote -60;
saute -30;
cocots(30);
saute 90;
tourne_gauche;
saute 30;
cocots(30);
};
```

```
cocots_emb():={
cocots(60);
pas_de_cote 15;
crayon rouge;
cocots(45);
pas_de_cote 15;
crayon vert;
cocots(30);
};
```

4.4.1 Prolongements

On peut continuer à emboiter les cocottes. Voici la procédure `cocot_zebre(n)` qui est une procédure récursive et le dessin `cocot_zebre(80)` :

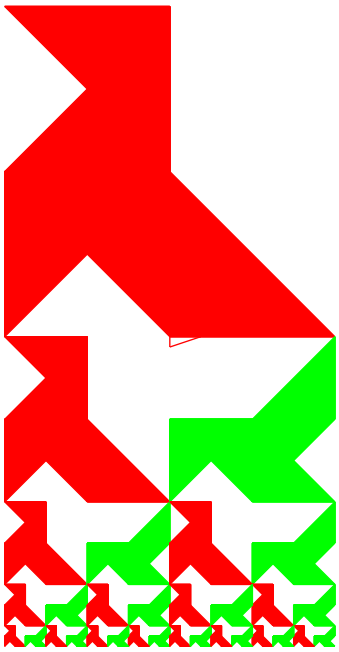
```
cocot_zebre(n):={
si (n>10) alors
crayon rouge;
cocots(n);
pas_de_cote 10;
crayon vert;
cocots(n-10);
pas_de_cote 10;
cocot_zebre(n-20);
sinon
crayon rouge;
cocots(n);
fsi;
};
```



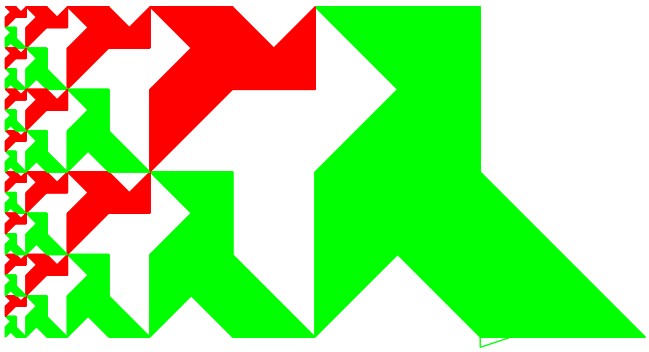
On peut aussi continuer fabriquer la famille des cocottes en faisant des lignes de cocottes de plus en plus petites. On écrit pour cela les procédures `cocots1(n)` et

`cocots2(n)` qui s'appellent l'une l'autre.
Voici les procédures récursives `cocots1(n)` et `cocots2(n)` et les dessins `cocots1(80)` et `cocots2(80)` :

```
cocots1(n):={
crayon rouge;
cocots(n);
si (n>9) alors
pas_de_cote -n;
saute -n/2;
cocots1(n/2);
saute 3*n/2;
tourne_gauche;
saute n/2;
cocots2(n/2);
saute n/2;
tourne_droite;
saute -n;
fsi;
};
```

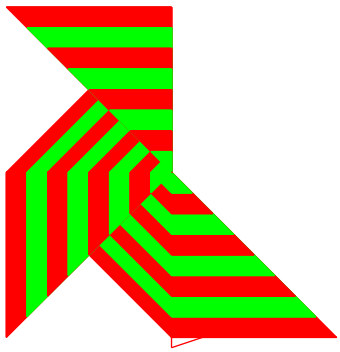


```
cocots2(n):={
crayon vert;
cocots(n);
si (n>9) alors
saute -3*n/2;
cocots2(n/2);
saute -n/2;
tourne_droite;
saute -3*n/2;
cocots1(n/2);
saute 3*n/2;
tourne_gauche;
saute 2*n;
fsi;
};
```



On peut remarquer que les procédures `cocots1(n)` et `cocots2(n)` ramènent la tortue à sa position de départ mais que ce n'est pas le cas pour la procédure `cocot_zebre`.
Voici la procédure `zebre(n)` ci dessous qui raméne la tortue à sa position de départ et le dessin `zebre(80)` :

```
zebre(n):={
si (n>10) alors
crayon rouge;
cocots(n);
pas_de_cote 10;
crayon vert;
cocots(n-10);
pas_de_cote 10;
zebre(n-20);
pas_de_cote -20;
sinon
crayon rouge;
cocots(n);
fsi;
};
```



On peut aussi faire d'autres dessins récursifs en écrivant par exemple une procédure ayant deux appels récursifs.
On va transformer `cocots_fam()` en `cocots_famille(n)` en écrivant une procédure qui aura 2 appels récursifs. Pour cela il faut tout d'abord transformer `cocots_fam()` en `cocots_fami(n)` procédure paramétrée qui dessine la mère et ses deux filles l'une rouge et l'autre verte et fait revenir la tortue à sa position de départ. Puis on remplace le dessin de chaque fille par celui d'une mère et de ses deux filles..., c'est à dire par un appel récursif.
On écrit :

```
cocots_fami(n):={
cocots(n);
pas_de_cote -n;
saute -n/2;
crayon rouge;
cocots(n/2);
saute 3*n/2;
tourne_gauche;
saute n/2;
crayon vert;
cocots(n/2);
saute n/2;
tourne_droite;
saute -n;
};
```

```
cocots_famille(n):={
cocots(n);
si n>9 alors
pas_de_cote -n;
saute -n/2;
crayon rouge;
cocots_famille(n/2);
saute 3*n/2;
tourne_gauche;
saute n/2;
crayon vert;
cocots_famille(n/2);
saute n/2;
tourne_droite;
saute -n;
fsi;
};
```

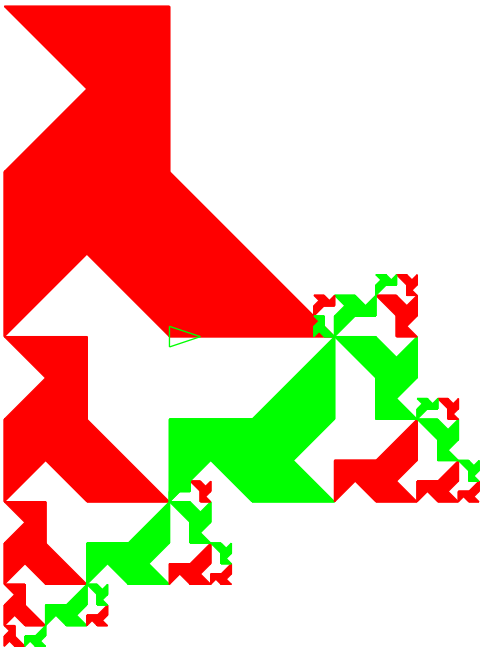
On remarque que l'on a fait en sorte que la tortue revienne à sa position d'arrivée après l'exécution de la procédure `cocots_famille` : cela permet de savoir où se

trouve la tortue après le premier appel récursif.

On tape :

```
crayon rouge;
cocots_famille(80);
```

On obtient :



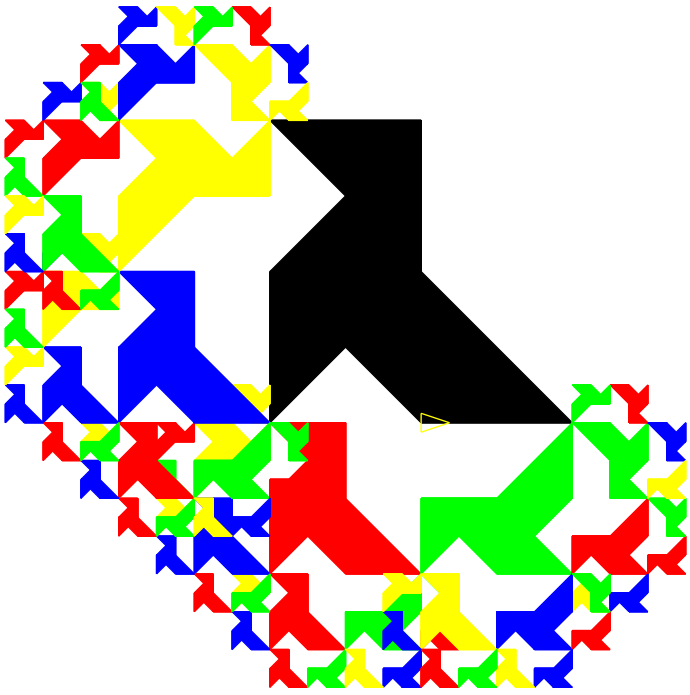
On peut aussi dire que la mère donne naissance à 4 filles qui donnent naissance chacune à 4 filles et écrire une procédure `cocots_family(n)` avec 4 appels récursifs.
Voici `cocots_family(n)` :

```
cocots_family(n):={
cocots(n);
si n>19 alors
pas_de_cote -n;
saute -n/2;
crayon rouge;
cocots_family(n/2);
saute 3*n/2;
tourne_gauche;
saute n/2;
crayon vert;
cocots_family(n/2);
saute n/2;
tourne_droite;
saute -5*n/2;
crayon bleu;
cocots_family(n/2);
saute -n/2;
tourne_droite;
saute -3*n/2;
crayon jaune;
cocots_family(n/2);
saute 3*n/2;
tourne_gauche;
saute 2*n;
fsi;
};
```

On tape :

```
pas_de_cote 40;
saute 80;
cocots_family(80);
```

On obtient :



4.5 Un tapis carré

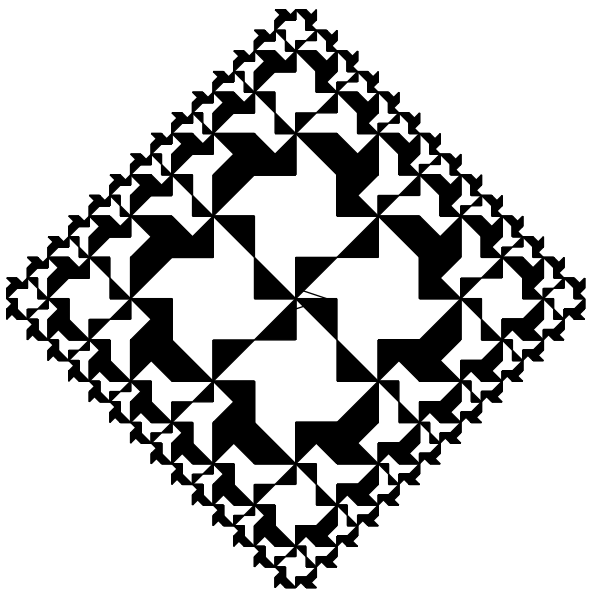
On définit la procédure `coins(p,n)` :

```
coins(p,n):={
repete(p,cocots(n),saute(n),tourne_droite,saute(-n),
         triangle_plein(n,n),pas_de_cote(2*n),
         tourne_droite(180),triangle_plein(n,n),
         saute(-3*n),tourne_droite,saute(-n),
         cocots(n),pas_de_cote(-2*n),saute(2*n))
};
```

Puis, on définit la procédure `coincoin(n)` :

```
coincoin(n):={
saute(-3*n);
pas_de_cote(-2*n);
coins(1,n);
saute -13*n/2;
pas_de_cote 5*n;
coins(3,n/2);
saute -29*n/4;
pas_de_cote 13*n/2;
coins(7,n/4);
pas_de_cote 15*n/2;
saute -n/4
};
```

On tape :
`efface; saute 80; repete(4,coincoin(20),tourne_droite);`
On obtient :



4.6 Les losanges emboités

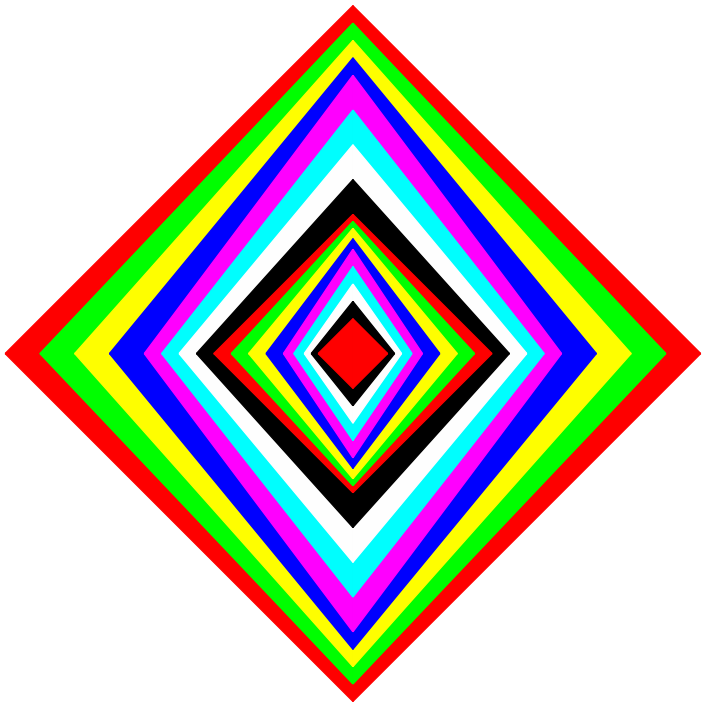
4.6.1 Le losange plein

Dans un losange, les diagonales se coupent en leur milieu et sont orthogonales. Réciproquement si un quadrilatère convexe a ses diagonales qui se coupent en leur milieu et sont orthogonales, ce quadrilatère est un losange. Si de plus les longueurs des diagonales sont égales, le quadrilatère est un carré.
Le losange plein de demi-diagonales de longueur `a` et `b` est formé de 4 triangles rectangles dont les cotés de l'angle droit sont de longueur `a` et `b`.
On utilise `triangle_plein(a,b)` et un paramètre `c` pour la couleur, on tape :

```
losange_plein(a,b,c):={
local cc;
cc:=crayon();
crayon c;
repete(2,triangle_plein(a,b),tourne_gauche,
         triangle_plein(b,a),tourne_gauche);
crayon cc;
}
```

4.6.2 Les losanges emboités

L'activité consiste à observer la figure ci-dessous et à la reproduire.



On tape :

```
losange_plein(100,100,1);
losange_plein(90,95,2);
losange_plein(80,90,3);
losange_plein(70,85,4);
losange_plein(60,80,5);
losange_plein(55,70,6);
losange_plein(50,60,7);
losange_plein(45,50,0);
losange_plein(40,40,1);
losange_plein(35,38,2);
losange_plein(30,36,3);
losange_plein(25,33,4);
losange_plein(20,30,5);
losange_plein(17,25,6);
losange_plein(14,20,7);
losange_plein(12,15,0);
losange_plein(10,10,1);
```

et on obtient bien le dessin voulu !

Chapitre 5

Les activités pour les grands

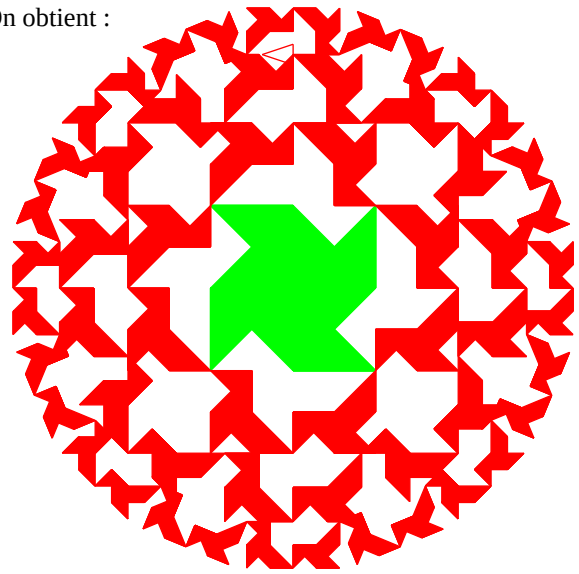
5.1 La ronde des cocottes

```
block(n):={
saute 2*n;
tourne_gauche ;
saute -n;
cocots(n);
saute -n;
tourne_droite;
saute -3*n;
cocots(n);
pas_de_cote 2*n;
saute n;
};
ronde(n,p):={
crayon rouge;
repete(p,block(n),saute 2*n,tourne_gauche 360/p ,saute 2*n);
};
```

On tape :

```
crayon vert;
cocots(40);
pas_de_cote 80;
tourne_droite 180;
cocots(40);
ronde(20,4);
pas_de_cote -40;
ronde(40*tan(pi/8),8);
pas_de_cote -80*tan(pi/8);
ronde(40*(1+tan(pi/8))*tan(pi/16),16);
```

On obtient :



5.2 Les carrés emboités

5.2.1 Les carrés emboités bicolores

On laisse la tortue à l'arrivée prête à faire un autre petit carré.

```
carre_emboites(n):={
cache_tortue;
si (n>4) alors {
repete(4,triangle_plein(n/2,n/2),avance n,tourne_gauche);
avance n/4;
tourne_gauche;
avance n/4;
tourne_droite;
carre_emboites(n/2);
}
};
```

On raméne la tortue à sa position de départ.

```
carre_emboites(n):={
cache_tortue;
si (n>4) alors {
repete(4,triangle_plein(n/2,n/2),avance n,tourne_gauche);
```

```
avance n/4;
tourne_gauche;
avance n/4;
tourne_droite;
carre_emboites(n/2);
tourne_gauche;
avance -n/4;
tourne_droite;
avance -n/4;
}
}
```

Question :
Si l'on poursuit indéfiniment, qu'elle est l'aire de la partie noire si le premier carré est de coté 1 unité ?
Reponse :
Soit A l'aire cherchée.
Les quatres premiers triangles ont comme aire 1/2 unité^2. Pour avoir l'aire totale il faut ajouter à cette aire, l'aire noire des carrés emboites commençant par un premier carré de coté 1/2 unité. Cette aire est donc égale à $A/4$.
On a donc $A = 1/2 + A/4$ soit $A = 2/3$.
Ou encore :
Soient A l'aire cherchée et B l'aire de la partie blanche.
On a donc $A + B = 1$ Le plus grand carré blanc a une aire égale à la moitie du grand carré noir.
On a donc $B = A/2$ Cela implique que $A + A/2 + 1$ soit $A = 2/3$.
Ou encore avec des séries :
$A = 1/2 + 1/8 + 1/32 + ... + 1/2^{2k+1} + ... A = 1/2 * (1 + 1/4 + 1/4^2 + ...1/4^k + ...) = 1/2 * 1/(1 - 1/4) = 2/3$

5.2.2 Les carrés emboités multicolores

On laisse la tortue à l'arrivée prête à faire un autre petit carré.

```
carre_emboite2(n,c):={
cache_tortue;
si (n>4) alors {
crayon c;
rectangle_plein(n);
avance n/2;
tourne_gauche 45;
carre_emboite2(n/sqrt(2),c+1)
}
}
```

On raméne la tortue à sa position de départ.

```
carre_emboite2(n,c):={
cache_tortue;
si (n>4) alors {
crayon c;
rectangle_plein(n);
avance n/2;
tourne_gauche 45;
carre_emboite2(n/sqrt(2),c+1)
tourne_gauche -45;
avance -n/2;
}
}
```

5.3 Le tapis

```
tapis(n):={
si (n>4) alors {
saute n/3;
pas_de_cote n/3;
rectangle_plein n/3;
pas_de_cote -n/3;
saute -n/3;
repete(4,tapis(n/3),avance n/3,tapis(n/3), avance 2*n/3,tourne_gauche);
}
}
```

On tape :

```
leve_crayon;
position(10,10);
baisse_crayon;
tapis(243);
```

Question :
Si l'on poursuit indéfiniment, qu'elle est l'aire de la partie noire si le premier carré est de coté 1 unité ? Reponse :
Soit A l'aire cherchée.
Le premier carré a comme aire 1/9 unité^2. Pour avoir l'aire totale il faut ajouter à cette aire l'aire noire de 8 tapis de côtés 1/3, c'est à dire $8*A/9$.
Donc on a :
$A = 1/9 + 8A/9$.
Cela implique que $9*A = 1+8*A$
soit $A = 1$ ce qui veut dire que l'aire de la partie blanche est constituée d'une infinité de points qui forment un ensemble de mesure nulle. Ou encore avec des séries :
$A = 1/9 + 8*1/81 + 8^2*1/9^3 + ... + 8^{k-1}*1/9^k + ...$
$A = 1/9*(1+8/9+(8/9)^2 + ...(8/9)^k + ...) = 1/9*1/(1-8/9) = 1.$

5.4 Une spirale

```
spire(n):={
repete(2,avance n,tourne_gauche);
};

spirale(n):={
si (n<200) alors {
spire(n);
spirale(n+10);
}
}
```

Avec chaque spire est remplie de couleurs différentes

```
spirec(n,c):={
crayon c;
repete(2,(repete(n,rectangle_plein(10),avance 10)),avance 10,tourne_gauche);
};

spiralec(n,c):={
si (n<200) alors {
spirec(n,c);
spiralec(n+1,c+1);
}
}
```

Chaque spire est remplie avec des carrés de couleurs différentes :

```
ligne(n,c):={
local k;
pour k de 0 jusque n-1 faire
crayon(c+k);rectangle_plein(10);avance 10;
fpour;
};
spirecc(n,c):={
local k;
pour k de 0 jusque 1 faire
ligne(n,c+k*n);avance 10;tourne_gauche;
fpour;
};

spiralecc(n,c):={
si (n<200) alors
spirecc(n,c);
spiralecc(n+1,c+2*n);
fsi;
}
```

et on tape :
`spiralecc(1,0)`
ou encore on écrit :

```
ligne(n,c):={
local k;
pour k de 0 jusque n-1 faire
crayon(c+k);rectangle_plein(10);avance 10;
fpour;
};
spirecc(n,c):={
ligne(n,c);avance 10;tourne_gauche;
ligne(n,c+n);avance 10;tourne_gauche;
};
spiralecc(n,c):={
local j ;
pour  j de 1 jusque 16 faire
spirecc(j,c+j*(j-1));
fpour;
}
```

n est le nombre de carrés de la derniére ligne, c la couleur

```
spiral(n,c):={
local j ,k,l;
pour  j de 1 jusque sqrt(n) faire
pour k de 0 jusque 1 faire
pour l de 0 jusque j-1 faire
crayon(c+j*(j-1+k)+l);rectangle_plein(10);avance 10;
fpour;
avance 10;tourne_gauche;
fpour;
fpour;
}
```

Le carré est blanc si son numéro est un nombre premier et il est rouge sinon. ceil(sqrt(n)) est le nombre de carrés de la derniére ligne, c la couleur On regarde donc les nombres de 1 à floor(sqrt(n))*(ceil((sqrt(n)). Si n =p*p avec p entier alors on va de 1 à n+p. Si p*p<n<(p+1)*(p+1) on va de 1 à p*p+p=p*(p+1)

```
spiralprem(n,c):={
local j ,k,l;
pour  j de 1 jusque sqrt(n) faire
pour k de 0 jusque 1 faire
pour l de 0 jusque j-1 faire
si (is_prime(c+j*(j-1+k)+l)) alors crayon noir; sinon crayon rouge; fsi;
```

```
rectangle_plein(10);avance 10;
fpour;
avance 10;tourne_gauche;
fpour;
fpour;
}
```

On colorie le premier carré en vert.

```
spiralpremv(n,c):={
local j ,k,l;
crayon vert;
rectangle_plein(2);avance 2;
avance 2;tourne_gauche;
crayon noir;rectangle_plein(2);avance 2;
avance 2;tourne_gauche;
pour  j de 2 jusque sqrt(n)+1 faire
pour k de 0 jusque 1 faire
pour l de 0 jusque j-1 faire
si (is_prime(c+j*(j-1+k)+l)) alors crayon noir; sinon crayon rouge; fsi;
rectangle_plein(2);avance 2;
fpour;
avance 2;tourne_gauche;
fpour;
fpour;
}
```

On tape ensuite :

```
efface; saute 80; pas_de_cote 80;
spiralpremv(10000,0);
```

On colorie le premier carré en vert et on ne dessine en noir que les carrés correspondant aux nombres premiers. On fait un cadre rouge pour délimiter notre dessin.

```
spiralpremvc(n,c):={
local j ,k,l;
crayon vert;
rectangle_plein(2);avance 2;
avance 2;tourne_gauche;
crayon noir;rectangle_plein(2);avance 2;
avance 2;tourne_gauche;leve_crayon;
pour  j de 2 jusque sqrt(n) faire
pour k de 0 jusque 1 faire
pour l de 0 jusque j-1 faire
si (is_prime(c+j*(j-1+k)+l)) alors baisse_crayon;crayon noir;
rectangle_plein(2);avance 2;leve_crayon;
sinon avance 2 fsi;
```

```
fpour;
avance 2;tourne_gauche;
fpour;
fpour;
//on trace un rectangle rouge pour faire le cadre
tourne_droite;avance -2;tourne_gauche;
baisse_crayon;crayon rouge;
repete(2,avance 2*sqrt(n)+2,tourne_gauche,avance 2*sqrt(n),tourne_gauche)
}
```

On tape ensuite :

```
efface; saute 80; pas_de_cote 80;
spiralpremv(10000,0);
```

On a alors les nombres premiers de 1 à 10100 : le cadre est un rectangle de 2*101 sur 2*100

5.5 Pour faire un fond quadrillé ou triangulé

5.5.1 Fond quadrillé

```
//a represente la longueur du carreau et k la couleur du maillage
maillage1(a,k):={
local p,c,cc;
p:= position;
c:=cap;
cc:=crayon;
leve_crayon;position([0,0]);cap 0;baisse_crayon;crayon k;
repete(ceil(260/a),avance(ceil(400/a)*a),pas_de_cote(a),
tourne_droite 180, avance(ceil(400/a)*a),pas_de_cote(-a),
tourne_droite 180);
leve_crayon;position([0,0]);cap 0;baisse_crayon;
tourne_gauche;
repete(ceil(200/a),avance(ceil(520/a)*a),pas_de_cote(-a),
tourne_droite 180, avance(ceil(520/a)*a),pas_de_cote(a),
tourne_droite 180);
crayon(cc);
leve_crayon;position(p);
cap(c);baisse_crayon;
};
```

On tape :

```
maillage1(30,22)
```

On obtient un fond quadrillé avec des carrés de couleur gris pâle.

```
maillage1(30,3)
```

On obtient un fond quadrillé avec des carrés de couleur jaune.

Pour les dessins de ce document je vais utiliser la procédure `maillage` ayant comme paramétres : `a,b,k,l,n,co` qui va dessiner avec le crayon `co` (jaune si `co=3`, un maillage de maille `n` dans le rectangle dont le sommet en bas et à gauche a pour coordonnées `a,b`, et qui a `k` carreaux horizontalement et `l` carreaux verticalement.

```
maillage(a,b,k,l,n,co):={
local p,c,cc;
p:= position;
c:=cap;
cc:=crayon;
leve_crayon;position([a,b]);cap 0;baisse_crayon;crayon co;
repete(l+1,avance(n*k),pas_de_cote(n),saute(-n*k));
leve_crayon;position([a,b]);cap 0;baisse_crayon;
tourne_gauche;
repete(k+1,avance(n*l),pas_de_cote(-n),saute(-n*l));
crayon(cc);
leve_crayon;position(p);
cap(c);baisse_crayon;
};
```

5.5.2 Fond triangulé

```
//a represente la longueur du triangle et k la couleur du maillage
//tricot fait un zig-zag a droite ou a gauche selon que s=-1 ou 1
tricot(a,s):={
repete(ceil(400/a),avance a,tourne_gauche s*120,avance a,
tourne_droite s*120)
};
maillage2(a,k):={
local p,c,cc;
p:= position;
c:=cap;
cc:=crayon;
leve_crayon;position([0,0]);cap 0;baisse_crayon;crayon k;
repete(ceil(200/a),avance(ceil(400/a)*a),tourne_gauche 120,
tricot(a,1),avance a,tourne_droite 120,avance(ceil(400/a)*a),
tourne_droite 120,avance -a,tricot(a,-1),tourne_gauche 120);
crayon(cc);
leve_crayon;position(p);
cap(c);baisse_crayon;
}
```

On tape :

```
maillage2(30,22)
```

On obtient un fond triangulé avec des triangles équilatéraux de côtés 30 et de couleur gris pâle de code 22.

5.6 Une illusion d'optique

```
para_plein(a,b,c):={
repete(2,avance a,tourne_gauche c,avance b,tourne_gauche 180-c);
polygone_rempli(-8);
}

colonne(a,b,c,coul):={
local n,j;
n:=size(coul);
pour j de 0 jusque n-1 faire
crayon coul[j];
para_plein(a,b,c);
tourne_gauche c;
saute(b);
tourne_droite c;
fpour
tourne_gauche c;
saute -b*n;
tourne_droite c;
saute a;
};

colonnes(p,n,a,b,c,coul):={
local k,cc;
for (k:=0;k<p;k++) {
cc:=mid(coul,k*n,n);
colonne(a,b,c,cc);
};
saute -p*a;
};

coul(d):={
local k,l;
l:=makelist(0,1,d);
for (k:=0;k<d;k++) {
l[k]:=hasard(256);
};
};
```

Pour les différents dessins qui suivent seules les couleurs sont mises de façons différentes

```
fonce():=((hasard(5))+4)*16+hasard(4);
clair():=((hasard(5))+4)*16+13+hasard(4):
l1:=[clair(),clair(),clair(),fonce(),fonce(),fonce()]
qui donne l1:=[110,142,127,113,129,83] et
l2:=[clair(),clair(),clair(),fonce(),fonce(),fonce()]
qui donnne l2:=141,143,77,115,80,65] et on pose :
lc:=concat (l1,l2)
qui donnne lc:=[110,142,127,113,129,83,141,143,77,115,80,65]
ou
lc:=coul(12);
ou
lc:=[63,161,58,179,81,84,59,18,162,57,166,130];

dessin1():={
colonnes(2,2,20,20,90,mid(lc,4,4));
saute 40;
colonnes(2,6,20,15,45,lc);
tourne_gauche 45;
colonnes(6,2,15,20,45,revlist(lc));
tourne_droite 45;
pas_de_cote(40);
saute -40;
colonnes(2,6,20,15,45,lc);
tourne_gauche 45;
saute(15*6);
tourne_gauche 135;
saute -80;
colonnes(2,2,20,20,90,mid(lc,4,4));
saute 80;tourne_droite 135;
saute -90;tourne_droite 45;
pas_de_cote -40;
};
ou avec 16 couleurs
lc:=makelist(x->16*9+x, 0,15) (16 couleurs)
ou
lc:=makelist(x->16*8+x, 0,15)
dessin1():={
colonnes(2,2,20,20,90,mid(lc,4,4));
saute 40;
colonnes(2,8,20,15,45,lc);
tourne_gauche 45;
colonnes(8,2,15,20,45,revlist(lc));
tourne_droite 45;
pas_de_cote(40);
saute -40;
colonnes(2,8,20,15,45,lc);
```

```
tourne_gauche 45;
saute(15*8);
tourne_gauche 135;
saute -80;
colonnes(2,2,20,20,90,mid(lc,4,4));
saute 80;tourne_droite 135;
saute -15*8;tourne_droite 45;
pas_de_cote -40;
};

ou avec 16 couleurs mais avec une longueur de 8 et une de 6
lc:=makelist(x->16*9+x, 0,15) (16 couleurs)
ou
lc:=makelist(x->16*8+x, 0,15)
dessin1():={
colonnes(2,2,20,20,90,mid(lc,4,4));
saute 40;
colonnes(2,6,20,15,45,lc);
tourne_gauche 45;
colonnes(8,2,15,20,45,revlist(lc));
tourne_droite 45;
pas_de_cote(40);
saute -40;
colonnes(2,8,20,15,45,lc);
tourne_gauche 45;
saute(15*6);
tourne_gauche 135;
saute -80;
colonnes(2,2,20,20,90,mid(lc,4,4));
saute 80;tourne_droite 135;
saute -15*6;tourne_droite 45;
pas_de_cote -40;
};

dessin2():={
colonnes(2,2,20,20,90,mid(lc,4,4));
saute 40;
colonnes(2,6,20,15,45,lc);
pas_de_cote(40);
saute -40;
colonnes(2,6,20,15,45,lc);
tourne_gauche 45;
saute(90);
tourne_gauche 135;
saute -80;
colonnes(2,2,20,20,90,mid(lc,4,4));
```

```
pas_de_cote(40);
saute 40;
tourne_droite;
colonnes(2,6,20,15,135,lc);
tourne_droite 45;saute -90;
tourne_droite 45;saute -40;
};
ou avec 16 couleurs
lc:=makelist(x->16*9+x, 0,15) (16 couleurs)
ou
lc:=makelist(x->16*8+x, 0,15)
dessin2():={
colonnes(2,2,20,20,90,mid(lc,4,4));
saute 40;
colonnes(2,8,20,15,45,lc);
pas_de_cote(40);
saute -40;
colonnes(2,8,20,15,45,lc);
tourne_gauche 45;
saute(15*8);
tourne_gauche 135;
saute -80;
colonnes(2,2,20,20,90,mid(lc,4,4));
pas_de_cote(40);
saute 40;
tourne_droite;
colonnes(2,8,20,15,135,lc);
tourne_droite 45;saute -15*8;
tourne_droite 45;saute -40;
};

dessin3():={
colonnes(2,2,20,20,90,mid(lc,4,4));
saute 40;
colonnes(2,6,20,15,45,lc);
pas_de_cote(40);
saute -40;
colonnes(2,6,20,15,45,lc);
tourne_gauche 45;
saute(90);
tourne_gauche 135;
saute -80;
pas_de_cote(40);
tourne_droite;
colonnes(2,2,20,20,90,mid(lc,4,4));
pas_de_cote(40);
```

```
colonnes(2,6,20,15,135,lc);
tourne_droite 45;saute -90;
tourne_droite 45;saute -40;
};
ou avec 16 couleurs
lc:=makelist(x->16*9+x, 0,15) (16 couleurs)
ou
lc:=makelist(x->16*8+x, 0,15)
dessin3():={
colonnes(2,2,20,20,90,mid(lc,4,4));
saute 40;
colonnes(2,8,20,15,45,lc);
pas_de_cote(40);
saute -40;
colonnes(2,8,20,15,45,lc);
tourne_gauche 45;
saute(15*8);
tourne_gauche 135;
saute -80;
pas_de_cote(40);
tourne_droite;
colonnes(2,2,20,20,90,mid(lc,4,4));
pas_de_cote(40);
colonnes(2,8,20,15,135,lc);
tourne_droite 45;saute -15*8;
tourne_droite 45;saute -40;
};
```

5.7 Un autre effet d'optique

```
truc(a,b):={
crayon(noir);
rectangle_plein(a);
crayon(rouge);
saute 5;
pas_de_cote 5;
tourne_droite 1;
rectangle_plein(a-b);
saute 5;
pas_de_cote 5;
tourne_droite 1;
};
caremb1(a,n):={
for(k:=0;k<n;k++){
truc(a-k*10,10);
```

```
};
};
caremb2(a,n):={
for(k:=0;k<n;k++){
truc(a-k*20,10);
};
};
```

caremb1(100,10) ; ou caremb2(100,5)

5.8 Un sudoku

```
ligne(coul):={
local n,j;
n:=size(coul);
pour j de 0 jusque n-1 faire
crayon coul[j];
rectangle_plein(30);
saute(30);
fpour
saute -30*n;
};
```

Pour l'impression il n'y a que 8 couleurs on fait donc un carré fond noir et rayé rouge pour la couleur 9 que l'on nomme `rect` :

```
rect(n,c):={
crayon 0;
rectangle_plein(n);
crayon irem(c,8);
repete(8,avance n,recule n,pas_de_cote n/8);
avance n;
recule n;
pas_de_cote -n;
};
ligner(coul):={
local n,j,c;
n:=size(coul);
pour j de 0 jusque n-1 faire
c:=coul[j];
si (c!=9) alors crayon coul[j];
rectangle_plein(30);
sinon rect(30,c);
fsi;
saute(30);
fpour
```

```
saute -30*n;
};

sudoku(m):={
local k,p;
p:=size(m);
pour k de 0 jusque p-1 faire
ligner(row(m,k));
pas_de_cote -30;
fpour;
pas_de_cote -30*p;
};
```

On tape par exemple :

```
m:=[[8,3,9,5,2,7,1,4,6],[7,2,5,1,4,6,3,8,9],[4,6,1,8,3,9,7,2,5],
[9,1,6,2,5,8,4,7,3],[3,8,4,9,7,3,6,5,2],[2,5,7,3,6,4,8,9,1],
[5,7,2,6,8,3,9,1,4],[6,9,8,4,1,2,5,3,7],[1,4,3,7,9,5,2,6,8]]
```

puis :

```
efface;pas_de_cote 120;saute-90;sudoku(m)
maillage(60,30,3,3,90,0)
```

On obtient :