# Nombres, géométrie, boucles...

## Inducteur : des carrés toujours plus grand

Partir de cette image ou d'une image de ce type et demander dans un premier temps d'analyser oralement sa construction.

Tous sont d'accord pour dire qu'il s'agit d'un carré dont le coté augmente à chaque traçage.

On obtient le carré de base avec ce code. Tous doivent être au moins à ce niveau de compétence depuis le temps...

```
$cote  = ask "Longueur coté ? "

repeat 4 {
    forward $cote
    turnright 90
}
```

C'est à dire, je place un nombre dans une boite nommée $cote, j'ouvre cette boite, je lis le nombre et trace le carré en fonction de ce nombre.

Le reste est tout simple : pour le second carré, je reprend cette boite mais il faudrait que le nombre soit un peu plus grand. C'est tout simple “Il suffit de le faire grandir avant de le remettre dans sa boite après son premier traçage ! ” peut on avoir comme proposition. Tout à fait !

C'est à dire que je remets ma variable dans une boite et que je l'augmente par exemple de 10. Donc

```
 $cote = $cote + 10
```

Et ceci, je le répète autant de fois que je veux tracer de carrés. Je n'ai donc qu'à rajouter une boucle **repeat** pour le nombre de carrés à dessiner. J'obtiens donc un **repeat** qui englobe un autre **repeat**.

```
$cote  = ask "Longueur coté ? "

# Je vais répéter le traçage 5 fois si je veux 5 carrés
repeat 5 {
```

```
    # Je dessine un carré
    repeat 4 {
        forward $cote
        turnright 90
    }

    # Je mets à jour ma boite
    $cote = $cote + 10
}
```

## Une frise de carrés

Demander au élèves de réaliser cette figure avec le moins de lignes de code possible.

La majorité des solutions va être de proposer un code bouclant bien la construction du carré mais pas le déplacement.

Reproduire au besoin la frise au tableau et bien montrant que seules les positions des abscisses varient.

Par exemple, choisissez un pas de 50. Montrer que le carré 1 est en 50, le suivant en 100, le suivant en 150 et que **l'on saute de 50 en 50**.

En suite c'est simple, la fonction **go** attend 2 variables : **le X** et **le Y**.

Donc, on va définir ce **go** avec sa valeur fixe (**le Y**) et une valeur variable : **le X**.

On aura donc, par exemple, un **go 20,100**, un **go 50, 100**, un **go 80, 100**... donc **un pas de X incrémenté de 30.**

Ce qui donne en code

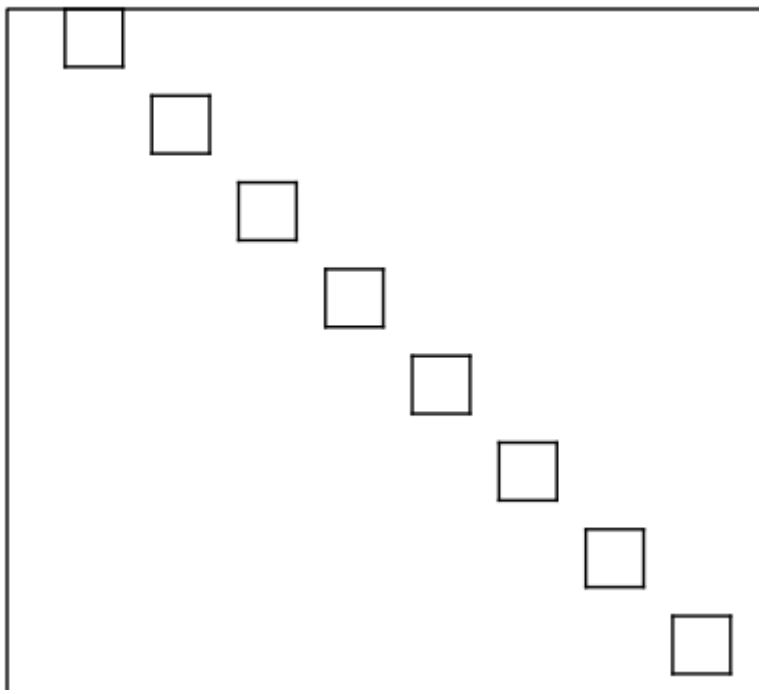
```
$X = 20
repeat 8 {
    go $X, 100
    # Je dessine un carré
    $X = $X + 30
}
```

Et avec le carré :

```
reset
spritehide
```

```
$X = 20


repeat 8 {
    go $X, 100
    # Je dessine un carré
    repeat 4 {
        forward 20
        turnright 90
    }
    $X = $X + 30
}
```

## Tout bouge !

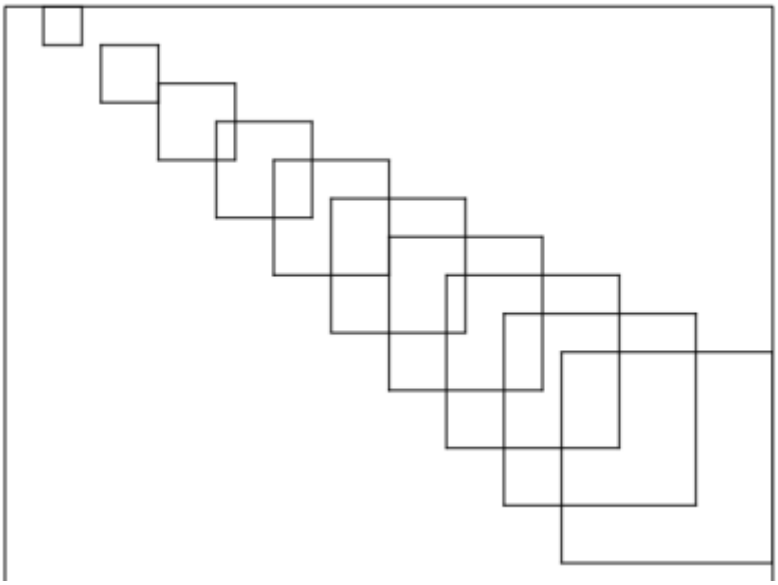


En autonomie, les élèves observent la figure et essayent de l'encoder. Proposition de code

```
reset
spritehide

$X = 20
$Y = 20


repeat 8 {
    go $X, $Y
    # Je dessine un carré
    repeat 4 {
        forward 20
```

```
        turnright 90
    }
    $X = $X + 30
    $Y = $Y + 30
}
```

## Tout bouge et grandit !



Re belote !

Bien montré que les X et les Y varient mais également les valeurs des cotés. Avec ce qui est sus-cité, ça devrait le faire...

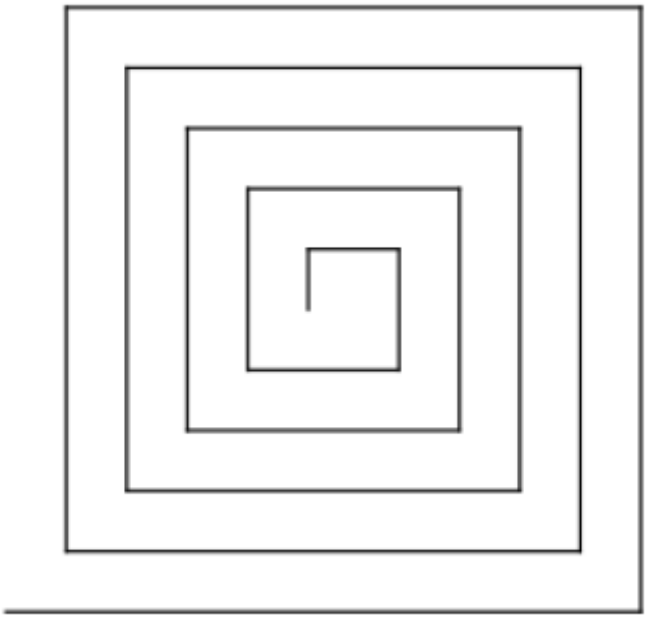
Proposition de code

```
reset
spritehide

$X = 20
$Y = 20
$cote = 20

repeat 10 {
    go $X, $Y
    # Je dessine un carré
    repeat 4 {
        forward $cote
        turnright 90
    }
    $X = $X + 30
    $Y = $Y + 30

$cote = $cote + 10
```

```
}
```

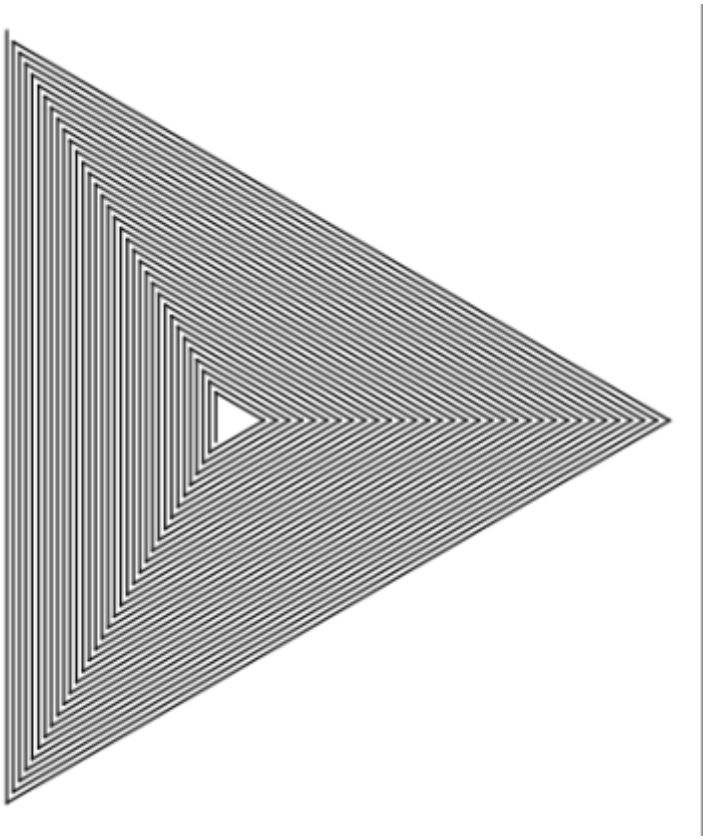
## Les spirales

Code très proche du code précédent, à la différence **que l'on incrémente le coté avant le traçage du dernier coté** et non une fois le traçage fini... Ce qui donne de très belles figures.



Proposition de code

```
reset
spritehide

$cote = 20

repeat 5 {
    repeat 4 {
    forward $cote
    turnright 90
    $cote = $cote +10
    }
}
```



Proposition de code

```
reset
spritehide

$cote = 20

repeat 25 {
    repeat 4 {
    forward $cote
    turnright 120
    $cote = $cote + 3
    }
}
```

# Et Pythagore dans tout cela ???

Petit exercice sympa, on va demander aux élèves de créer eux même leur(s) table(s) de Pythagore pour les aider dans leur apprentissage de la multiplication.

Écrire une table de multiplication au tableau est demandé de l'analyser. Voir comment on passe d'une ligne à l'autre.

Il en ressort que l'on commence toujours à 1 que l'on multiplie par ce nombre, puis idem avec 2, puis 3, puis 4…

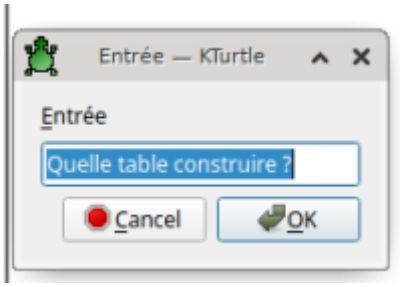
On appellera “**facteur**” terme qui multipliera 1, puis 2, puis 3… ce que l'on nomme table de 4, de 5

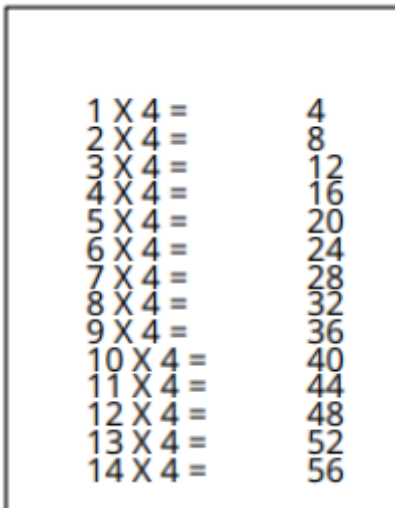
ou de 12.

Exemple 1 X 4 = 4 2 X 4 = 8 ... 12 X 4 = 48

Donc il comprenne qu'on prend 1 X “le facteur”, puis (1 + 1) X “le facteur”, puis (2 +1) X “le facteur”, et ainsi de suite...

Trouver le moyen de coder cette table.



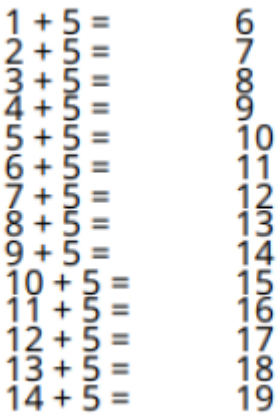

Proposition de code

```
reset
spritehide

$Y = 25
$nombre = 1
$facteur = ask "Quelle table construire ?"


repeat 14 {
    # On se positionne
    go 25, $Y
    # On affiche la formule magique
    print $nombre + " X " + $facteur + " =    " + $nombre * $facteur
    # On met à jour la variable $nombre
    $nombre = $nombre + 1
    # On descend d'une ligne dans l'affichage
    $Y = $Y + 10
```

```
}
```

## Et les autres tables (+,...)

Bah pourquoi pas !




Avec la proposition de code

```
reset
spritehide

$Y = 25
$nombre = 1
$facteur = ask "Quelle table construire ?"


repeat 14 {
    # On se positionne
    go 25, $Y
    # On affiche la formule magique
    print $nombre + " + " + $facteur + " =    " + ($nombre + $facteur)
    # On met à jour la variable $nombre
    $nombre = $nombre + 1
    # On descend d'une ligne dans l'affichage
    $Y = $Y + 10
}
```

Sans limites, que celles de l'imagination ...

## Navigation

| page précédente | Sommaire | Page suivante |
|---|---|---|
| Parlons nombre.... | sommaire | Et le hasard dans tout cela ? |

From:
https://cbiot.fr/dokuwiki/ - **Cyrille BIOT**

Permanent link:
**https://cbiot.fr/dokuwiki/kturtle:kturtle-activites-09?rev=1578420576**

Last update: **2020/01/07 18:09**