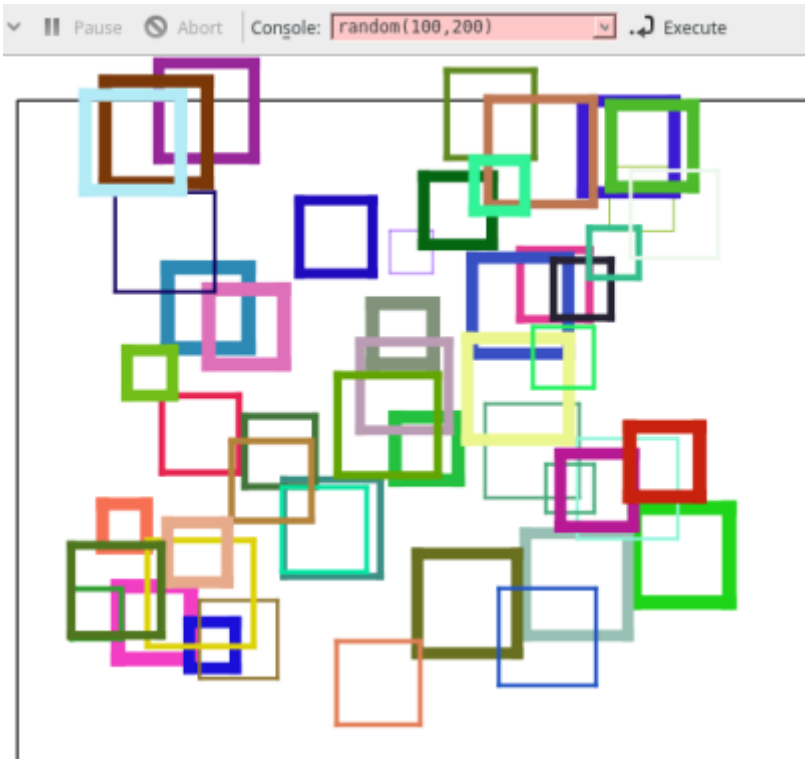
Et le hasard dans tout cela ?

Inducteur

On présente cette image et demande d'analyser ce que l'on y voit.



Pas à pas

Normalement, il en ressort que l'on voit des carrés, que des carrés avec comme changement :

- La couleur
- L'épaisseur du trait
- La position

Impecc ! On sait gérer tout cela. Le seul hic c'est que les chiffres de ces propriétés varient à chaque traçage comme s'il l'on lançait un dé : il va fallait **gérer le hasard**.

Bien que le hasard n'existe pas en informatique, mais ça les élèves ne sont pas obligés de le savoir, il y a quand même une fonction qui va nous rendre service : la fonction **random** .

Cette fonction random fonctionne avec 2 paramètres qui définissent une plage de données : la valeur minimale de cette plage de donnée et la maximale.

Ainsi pour reproduire un dé, on utilisera le code

```
random 1,6
```

Qui fournira un chiffre de 1 à 6 de façon aléatoire “simulée”.

Donc on reprend le plan de construction nécessaire POUR UN CARRE dessiné de façon aléatoire.

```
- Aller à X, Y
(compris dans le canevas, même un peu moins s'il on ne veut pas que ça
dépasse...)

- Prendre un crayon d'épaisseur X

- Choisir une couleur R, V, B (avec R V B qui changent à chaque fois)

- Tracé un carré de coté X
```

Recommencer plein de fois ce protocole et changeant les valeurs. </code>

Ça devient clair, on réalise ce type de plan de construction au tableau.

```
go (mini : 50, maxi: 350),(mini : 50, maxi: 350)
penwidth (mini : 1, maxi:8)
pencolor (mini : 0, maxi:255),(mini : 0, maxi:255),(mini : 0, maxi:255)
# le carré
repeat 4 {
   forward CoteQuiChangeAChaqueCarré
   turnright 90
}
```

Attention, pour le carré, ils auront tendance à coller un random suite au forward mais ce n'est pas la bonne méthode car ainsi on n'aura plus un carré mais 4 segments orthogonaux aux dimensions différentes. Il faut donc définir ce coté AVANT la boucle du carré.

Une fois que ça passe pour un carré, on englobe le tout dans une grosse boucle qui répétera l'opération autant de fois que nécessaire (sur l'image sus-citée, il y en a 50)

Proposition de code

```
reset
spritehide

repeat 50 {

    # Le hasard
    go (random 20,300), (random     20,300)
    $cote = (random 20,50)
```

```
    penwidth (random 1,7)
    pencolor (random 0,255),(random 0,255) ,(random 0,255)

    # Le carré
    repeat 4 {
        forward $cote
        turnright 90
        }

}
```

Attention à bien mettre des parenthèses **(**random 10,100**)** , certes non obligatoires en appel traditionnel, mais sinon ça posera des soucis sur l'appel de **pencolor** (**kturtle** aura du mal à parser les paramètres des différentes fonctions, les **( )** permettront de palier à ce problème...

Finalement, ce n'était pas si compliqué...

Variantes

Idem avec des triangles, des losanges, des rectangles...

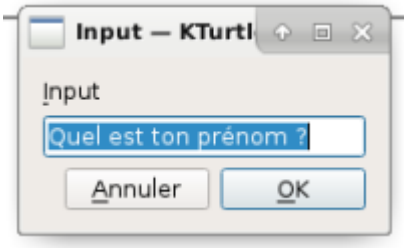
Jouons avec les prénoms

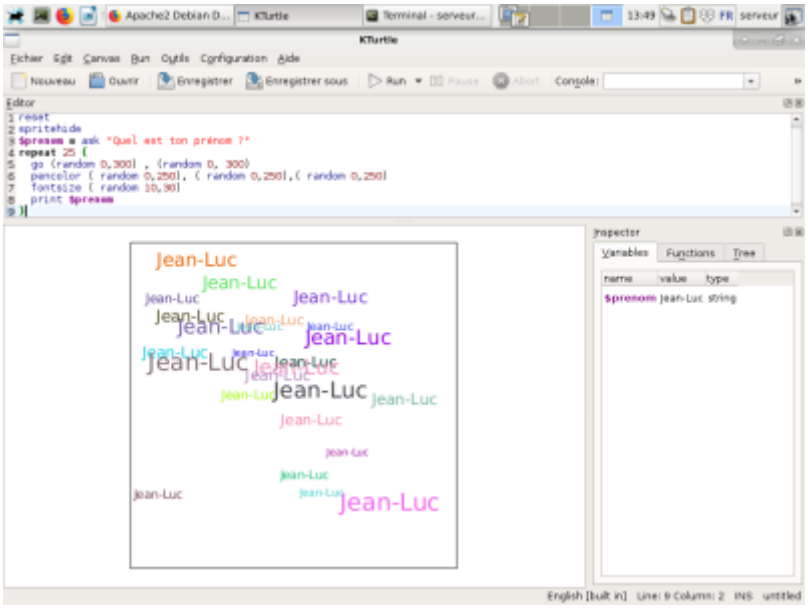
Réinvestissement des séquences précédentes

Ils savent :

- Ecrire un message –> **print** “message”
- Changer la couleur d'écriture → **pencolor** R,G,B
- Changer la taille de la police → **fontsize** X
- Se positionner sur une feuille → **go** X,Y
- Récupérer une entrée d'une pop-up dans une variable → $var = **ask** “Question”
- Faire des boucles (répétitions) → **repeat** {}

On présente le document final souhaité, on mutualise ce que l'on voit, on analyse... et puis c'est à eux de jouer !





Proposition de code

```
reset
spritehide
$prenom = ask "Quel est ton prénom ?"
```

```
repeat 25 {
    go (random 0,300) , (random 0, 300)
    pencolor (random 0,250), (random 0,250),(random 0,250)
    fontsize (random 10,30)
    print $prenom
}
```

Navigation

page précédente	Sommaire	Page suivante
Nombres, géométrie, boucles...	sommaire	La boucle if