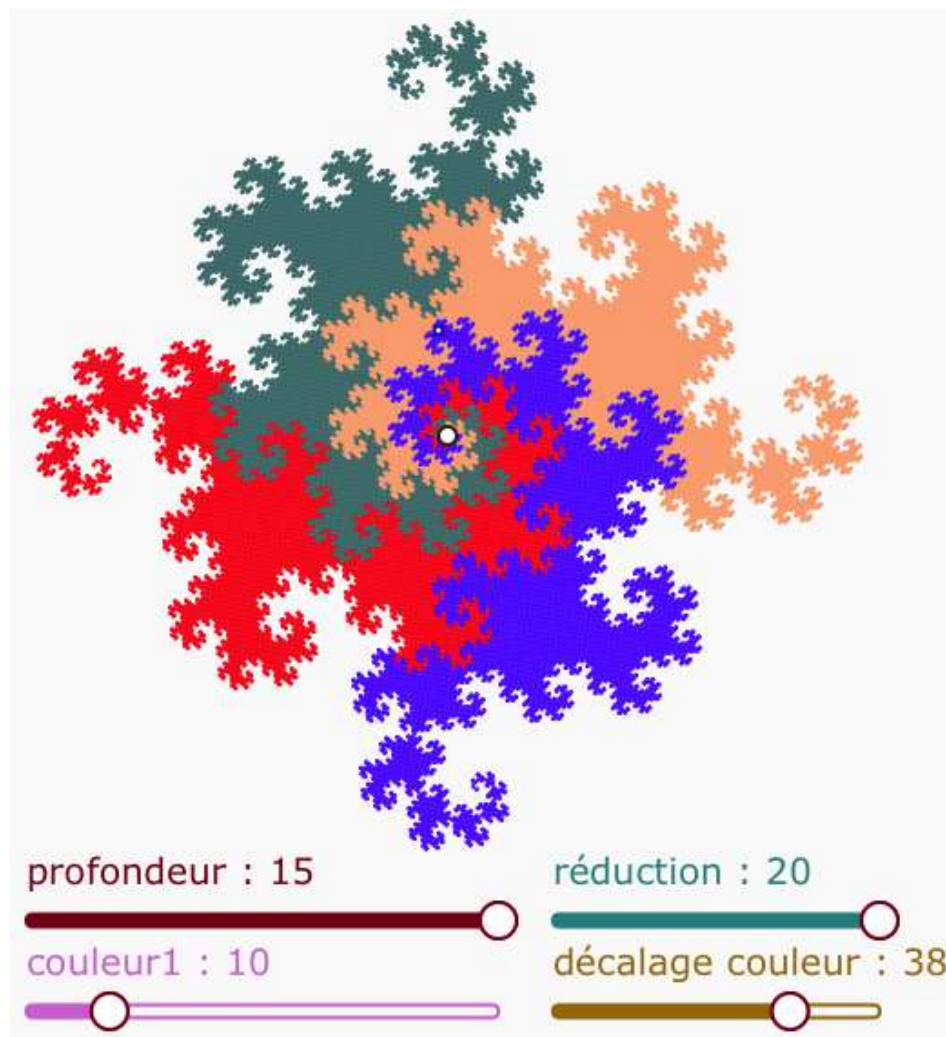
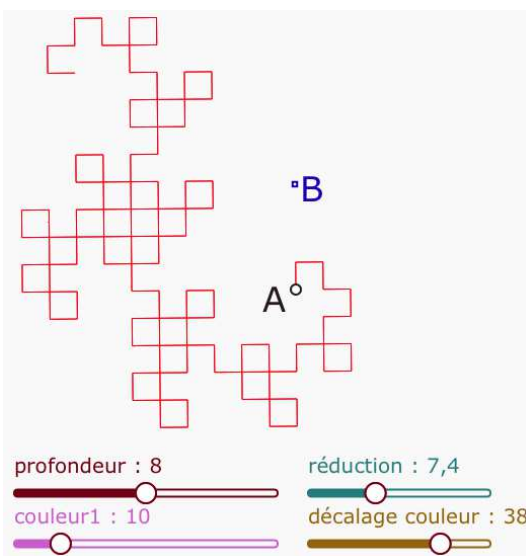


Objectif : Construire le pavage du plan par 4 courbes du dragon à la tortue



Lien : <https://huit.re/DGPad-fiche-dragon>

Exercice 1 :



Construire la figure ci-contre (1 dragon) à partir des éléments dynamiques suivants :

- * points A, B
- * 1 curseur de profondeur
- * 2 curseurs de couleur
- * 1 curseur (exponentiel) de réduction

La courbe du dragon est une fractale :

<http://www.mathcurve.com/fractals/dragon/dragon.shtml>

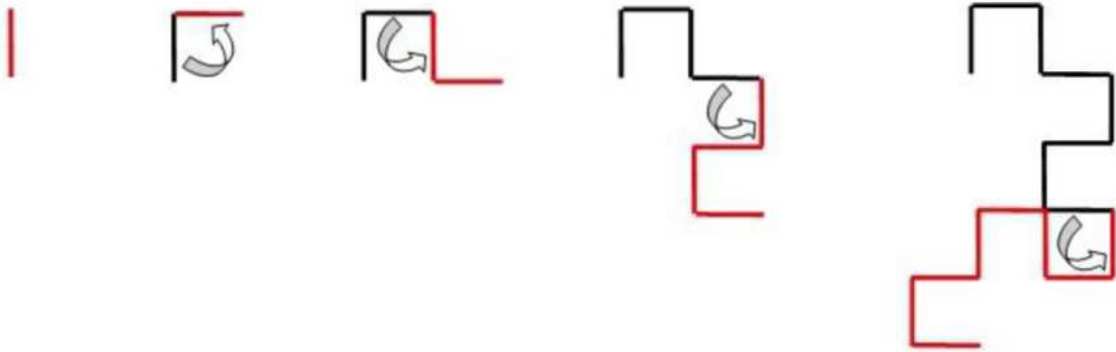
La réduction k s'appliquera sous forme $1, 2^k$.

Le premier dragon a une couleur col1 (couleur 1), le suivant col2 (couleur 2) de plus.

La profondeur est un curseur c entre 0 et 15 qui illustre le dépliement progressif du dragon (on peut le prendre entier pour garder des dragons sans chevauchement).

1. Créer les points libres A et B.

Créer les curseurs c (profondeur), k (réduction), col1 et col2.

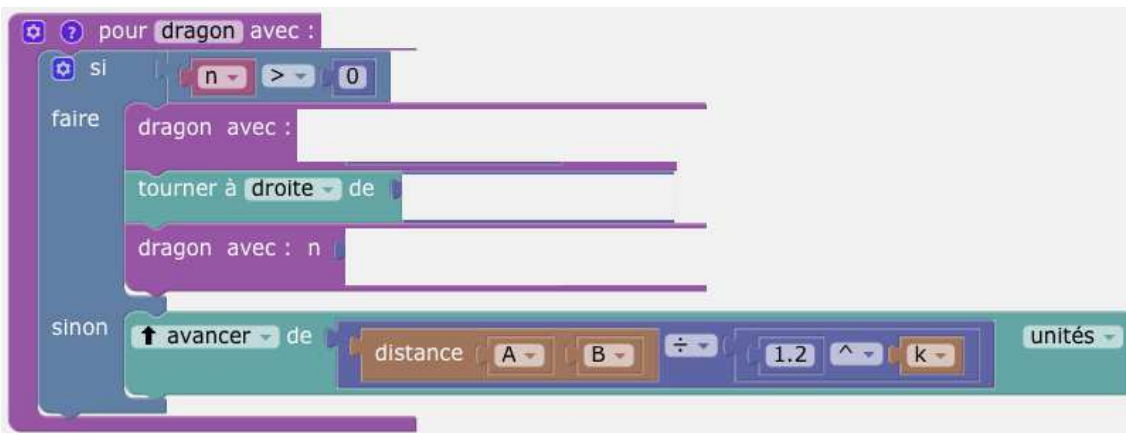


2. On va créer une fonction récursive dragon() pour tracer la courbe.

Sur la partie rouge de la courbe, on parcourt la courbe noire à l'envers, donc si on reprend les instructions de la partie noire (on les reprendra dans le sens inverse) il faut remplacer tourner à gauche par tourner à droite.

On va donc ajouter un paramètre sens à la fonction dragon() :

- si sens = 1, ce sera la courbe du dragon.
- si sens = -1, ce sera la courbe du dragon dans le sens inverse.



Créer cette fonction et la tester pour n = partie entière de c.

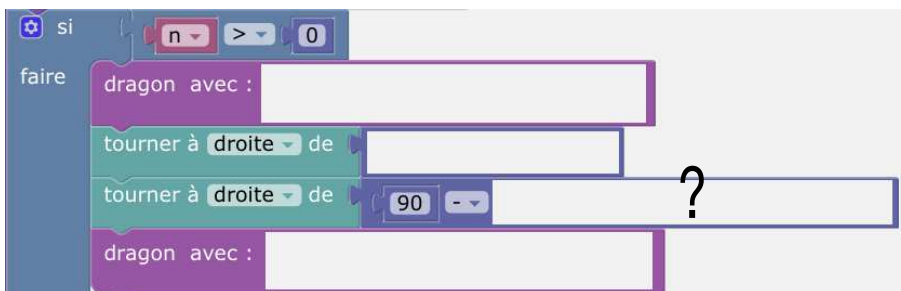


3. On veut maintenant créer l'effet de pivotement, comme si la courbe était générée par pliage.

L'apparition de la courbe rouge sera donc différée : juste avant, on tournera à droite de $90 - e$

Au départ, e vaudra 0 et la courbe coïncidera avec la courbe noire. Ensuite, e augmentera jusqu'à 90.

La formule pour e est $e = \max(0, \min(c * 90 - n * 90, 90))$.



4. ... (voir coups de pouce)

Exercice 2 :

Créer le pavage par les quatre dragons (première figure).

Exercice 1, coup de pouce n°1 :

```
pour dragon avec : n, sens
  si n > 0
    faire
      dragon avec :
      tourner à droite de
      Créer une nouvelle liste vide pivot
      Créer une nouvelle liste vide inter

      tourner à droite de
      dragon avec :
    sinon
      avancer de distance A B / 1.2 k unités

  pivoter vers le point B
  mettre la couleur à col1
  dragon avec : n arrondir par défaut c sens 1
```

The image shows a Scratch script for a dragon curve algorithm. It starts with a 'pour dragon avec : n, sens' loop. Inside, there is an 'si n > 0' conditional block. The 'faire' branch contains a recursive call 'dragon avec :', followed by 'tourner à droite de', 'Créer une nouvelle liste vide pivot', and 'Créer une nouvelle liste vide inter'. The 'sinon' branch contains 'avancer de distance A B / 1.2 k unités'. After the loop, there are three final blocks: 'pivoter vers le point B', 'mettre la couleur à col1', and 'dragon avec : n arrondir par défaut c sens 1'.

Exercice 2, coup de pouce n°1 :

```
pour dragon avec : n, sens
  si n > 0
    faire
      dragon avec :
      tourner à droite de
      Créer une nouvelle liste vide pivot
      Créer une nouvelle liste vide inter
    tourner à droite de
    dragon avec :
  sinon
    avancer de distance (A) (B) / (1.2 ^ k) unités

mettre la couleur à col1
compter avec i de
faire
  pivoter vers le point B
  tourner à droite de
  dragon avec : n arrondir par défaut c sens 1
  lever le stylo
  rejoindre le point A
  poser le stylo
  ajouter col2 à la couleur
```