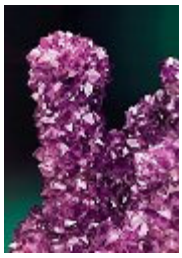
Modélisation mathématique d'une croissance (cristalline)

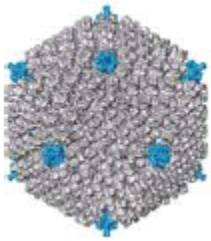
Partie A **Observation**

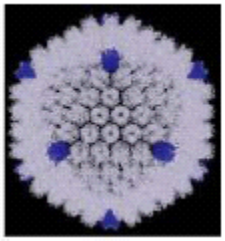
La cristallisation obéit à des lois physiques.



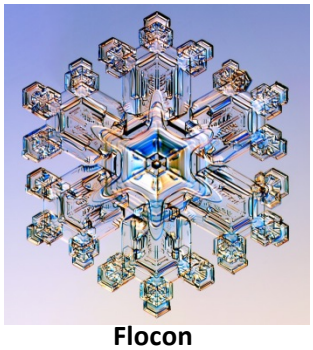
Améthyste

Coquille de Conus





Virus icosaédrique



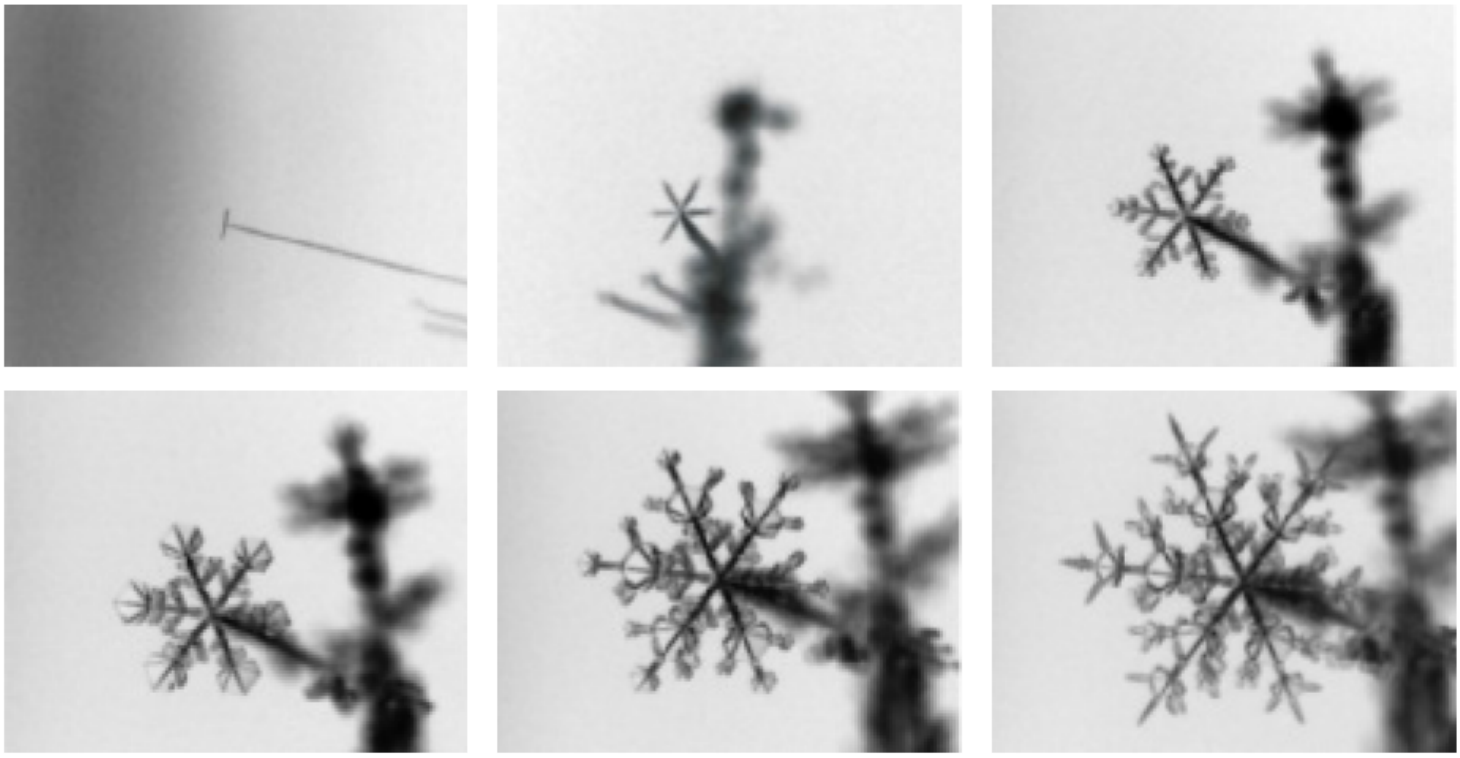
Flocon

Partie B **La formation des cristaux de neige**

Si l'on considère la neige, l'eau donne en gelant une multitude de formes aux motifs complexes. La diversité des formes est si remarquable qu'elle donne presque raison au vieil adage selon lequel il n'existe pas deux cristaux de neige identiques.

En fait, de nombreux aspects de la croissance de ces minuscules chefs-d'œuvre de glace restent très difficiles à expliquer, même qualitativement. Les nuages d'hiver ont beau les livrer à notre observation par milliards, nous commençons tout juste à comprendre pourquoi les cristaux de neige acquièrent des formes aussi caractéristiques et distinctes.

Série d'images prises en laboratoire



Peut-on trouver des règles simples pour reproduire la forme d'un cristal de neige ?

C'est ce que nous expérimentons dans la partie suivante.

Les automates cellulaires

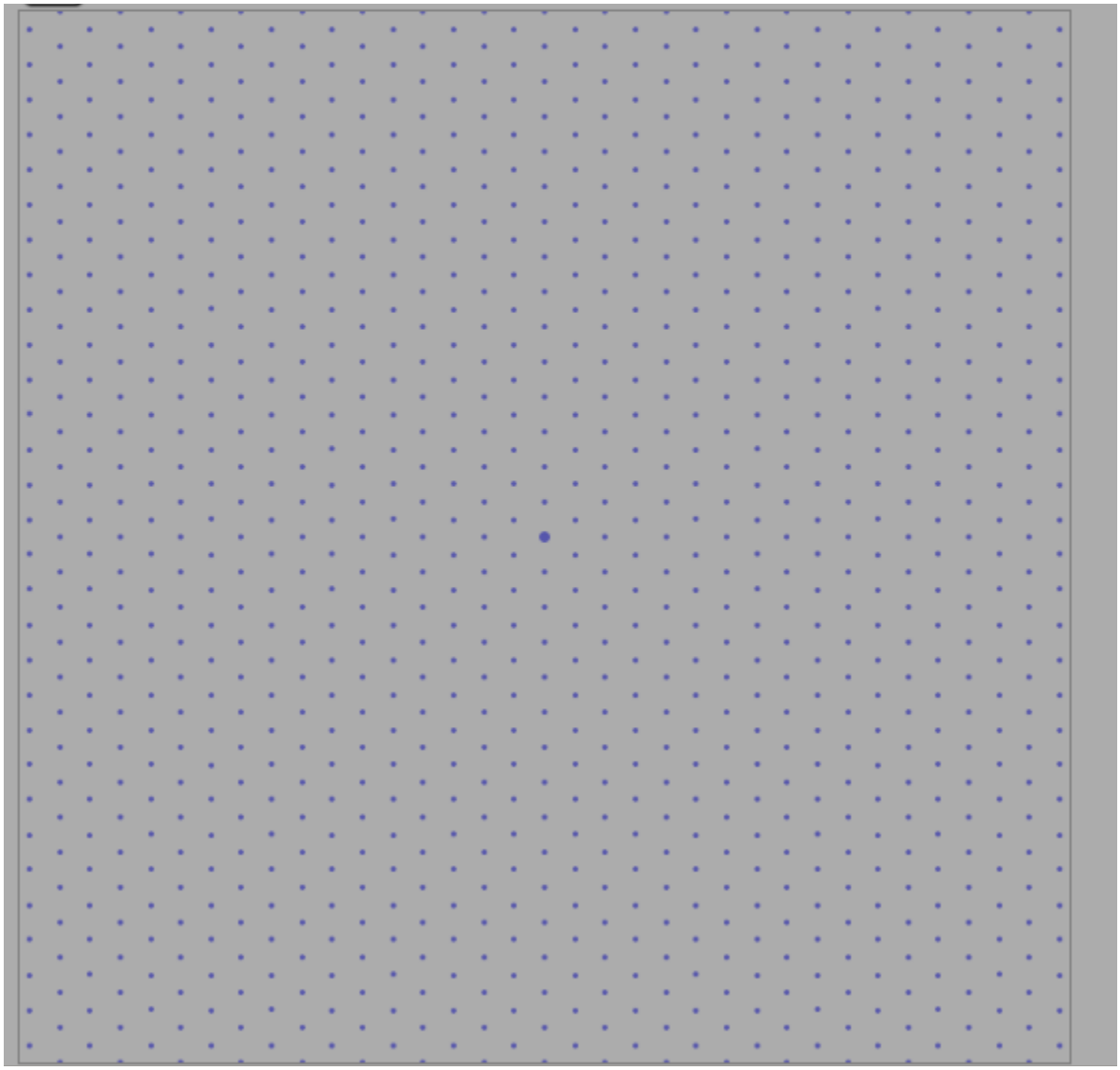
Un automate cellulaire est un ensemble de cases d'un plan (appelées cellules) dont l'état (par exemple : occupé ou vide) dépend de l'état des cases voisines.

I En utilisant l'application Flakepad : construction d'un flocon

Le mathématicien Norman Packard, imagina que la croissance d'un flocon pouvait être simulée par des automates cellulaires en utilisant des règles simples.

Packard a proposé l'algorithme suivant : une cellule représentée par un cercle ne devient occupée par une autre cellule (c'est-à-dire un autre cercle) que si elle n'avait qu'une seule cellule voisine occupée.

1. Tester sur la figure ci-dessous.
   a) Commencer par un cercle au centre
   b) À l'étape suivante et, avant de cliquer dessus, choisir les points qui deviendront des cercles en respectant la règle ci-haut mentionnée.
   (Remarque : des points dans la même situation seront automatiquement transformés et plus vous vous éloignez du centre plus il peut y avoir de points à sélectionner autour de chaque cellule)



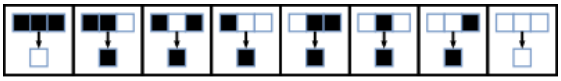
2. Tester cet algorithme en allant sur : http://penflakes.com/flakepad/
3. Imprimer votre production et coller dans votre cahier.

II Fabrication d'un automate cellulaire en appliquant une règle :

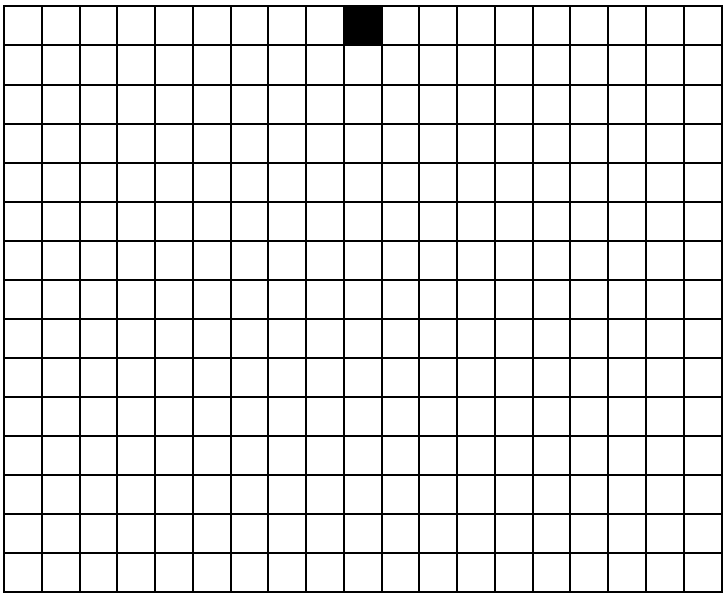
Pour réaliser un automate cellulaire élémentaire, nous allons considérer **des cases disposées en ligne**. Chaque case peut être soit blanche, soit noire. Au démarrage de notre simulation, nous allons supposer qu'une seule case est noire, comme sur le schéma ci-dessous.

Puis nous allons faire évoluer notre système, en définissant des règles qui vont conditionner le changement de la couleur des cases à chaque étape de la simulation. Pour faire au plus simple, nous allons supposer que **la nouvelle couleur d'une case dépend de sa couleur actuelle et de celle de ses voisines**. Voici une règle de ce genre :

Si une case noire est entourée par deux cases également noires, elle devient blanche au tour suivant (C'est-à-dire à la ligne suivante).
Cette règle est appelée règle 126, et voici ce qu'elle donne :



1. Compléter cette grille en coloriant les cases des lignes suivantes selon la règle 126.



2. Vous allez maintenant remplir une grille à l'aide du tableur Excel, en appliquant cette règle 126.
   Dans ce qui suit, on utilisera le fichier automate.xls situé dans le contenu pédagogique MPS.

a) Entrer sous Excel les formules suivantes :

EN A2 : =SI(ET(B1=0;A1=0);0;1)

En B2 : =SI(OU((ET(B1=0;C1=0;A1=0));(ET(B1=1;C1=1;A1=1)));0;1)

b) Sélectionner la cellule B2 et la tirer vers la droite jusqu'à la cellule R2

c) Entrer en S2 : =SI(ET(R1=0;S1=0);0;1)

d) Sélectionner les cellules de A2 jusqu'à S2. Puis tirer la croix noire vers le bas jusqu'à la ligne 15.

e) Sélectionner toutes les cellules de A1 à S15 et Aller dans *Accueil /Style/Mise en forme conditionnelle /règle de mise en surbrillance des cellules/égale à*.

f) Imprimer et coller dans votre cahier

3. Algorithme : Flocon de Koch

a) Ouvrir Algobox

b) Écrire le programme ci-dessous sous Algobox

```
VARIABLES
  n EST_DU_TYPE NOMBRE
  i EST_DU_TYPE NOMBRE
  j EST_DU_TYPE NOMBRE
  k EST_DU_TYPE NOMBRE
  a EST_DU_TYPE NOMBRE
  x EST_DU_TYPE LISTE
  y EST_DU_TYPE LISTE
  max EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  AFFICHER "nombre d'itérations entier positif non nul inférieur ou égal à 9:"
  LIRE n
  AFFICHER n
  SI (n>=1 ET n<=9) ALORS
    DEBUT_SI
    max PREND_LA_VALEUR pow(4,n)
    POUR i ALLANT_DE 0 A max
      DEBUT_POUR
      x[i] PREND_LA_VALEUR 0
      y[i] PREND_LA_VALEUR 0
      FIN_POUR
    x[max] PREND_LA_VALEUR 450
    POUR i ALLANT_DE 1 A n
      DEBUT_POUR
      k PREND_LA_VALEUR pow(4,n-i)
      POUR j ALLANT_DE 0 A pow(4,i-1)-1
        DEBUT_POUR
        a PREND_LA_VALEUR 4*j*k
        x[a+k] PREND_LA_VALEUR (2*x[a]+x[a+4*k])/3
        y[a+k] PREND_LA_VALEUR (2*y[a]+y[a+4*k])/3
        x[a+3*k] PREND_LA_VALEUR (x[a]+2*x[a+4*k])/3
        y[a+3*k] PREND_LA_VALEUR (y[a]+2*y[a+4*k])/3
        x[a+2*k] PREND_LA_VALEUR (x[a+k]+x[a+3*k]+sqrt(3)*(y[a+k]-y[a+3*k]))/2
        y[a+2*k] PREND_LA_VALEUR (y[a+k]+y[a+3*k]+sqrt(3)*(x[a+3*k]-x[a+k]))/2
        FIN_POUR
      FIN_POUR
    POUR i ALLANT_DE 0 A max-1
      DEBUT_POUR
      TRACER_SEGMENT (x[i],y[i])->(x[i+1],y[i+1])
      FIN_POUR
    FIN_SI
    SINON
      DEBUT_SINON
      AFFICHER "n doit être plus petit ou égal à 9"
      FIN_SINON
FIN_ALGORITHME
```

4. Algorithme : Triangle de Sierpinski

(on le testera avec n = 10000)

```
VARIABLES
  xa EST_DU_TYPE NOMBRE
  ya EST_DU_TYPE NOMBRE
  xb EST_DU_TYPE NOMBRE
  yb EST_DU_TYPE NOMBRE
  xc EST_DU_TYPE NOMBRE
  yc EST_DU_TYPE NOMBRE
  xm EST_DU_TYPE NOMBRE
  ym EST_DU_TYPE NOMBRE
  n EST_DU_TYPE NOMBRE
  h EST_DU_TYPE NOMBRE
  I EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  xa PREND_LA_VALEUR -10
  ya PREND_LA_VALEUR 0
  xb PREND_LA_VALEUR 10
  yb PREND_LA_VALEUR 0
  xc PREND_LA_VALEUR 0
  yc PREND_LA_VALEUR 10*sqrt(3)
  xm PREND_LA_VALEUR 4
  ym PREND_LA_VALEUR 6
  LIRE n
  TRACER_SEGMENT (xa,ya)->(xb,yb)
  TRACER_POINT (xm,ym)
  TRACER_SEGMENT (xb,yb)->(xc,yc)
  TRACER_SEGMENT (xc,yc)->(xa,ya)
  POUR I ALLANT_DE 1 A n
    DEBUT_POUR
    h PREND_LA_VALEUR ALGOBOX_ALEA_ENT(1,3)
    SI (h==1) ALORS
      DEBUT_SI
      xm PREND_LA_VALEUR (xa+xm)/2
      ym PREND_LA_VALEUR (ya+ym)/2
      FIN_SI
      SINON
        DEBUT_SINON
        SI (h==2) ALORS
          DEBUT_SI
          xm PREND_LA_VALEUR (xb+xm)/2
          ym PREND_LA_VALEUR (yb+ym)/2
          FIN_SI
          SINON
            DEBUT_SINON
            xm PREND_LA_VALEUR (xc+xm)/2
            ym PREND_LA_VALEUR (yc+ym)/2
            FIN_SINON
        FIN_SINON
    TRACER_POINT (xm,ym)
    FIN_POUR
FIN_ALGORITHME
```

Annexes