

ECOSYSTÈME multi-agents

Atelier École multimédia du 8 au 12 février 2016

LUNDI

Présentation

Processing

Différents projets réalisés à l'aide de Processing

Le Projet : Écosystème multi-agents

Les bases du code

Les variables et leurs types

Méthodes, style d'écriture et commentaires

Forme et apparence

Des boucles et des listes

Mouvement

Position, déplacement, direction

Vecteur

Coordonnées et autres notions mathématiques.

MARDI

Les agents une classe à part !

Les classes ou programmation orientée objet – POO –
Mon premier agent, définition de son comportement !

MERCREDI

Tous agents, tous différents !

Classe et sous-classe.

L'écosystème un système multi-agents.

JEUDI

Des vêtements pour nos agents

Donner des formes à nos agents qui pour l'instant ne sont que informations et positions.

Mise en scène et en lumière de l'écosystème

3D, lumières et système de caméra dans Processing.

VENDREDI

Pilotage de l'application

Introduction de la librairie Controle P5 pour créer une mini-interface de control sur l'application.

PROCESSING

Référence à propos de la librairie java Processing

Processing.org

Nature of code par Daniel Shiffman

Openprocessing.org

Projets réalisés à l'aide de Processing

[Cinematics](#)

[Generative Design](#)

[Puddles](#)

[Romanesco project](#)

Auteurs

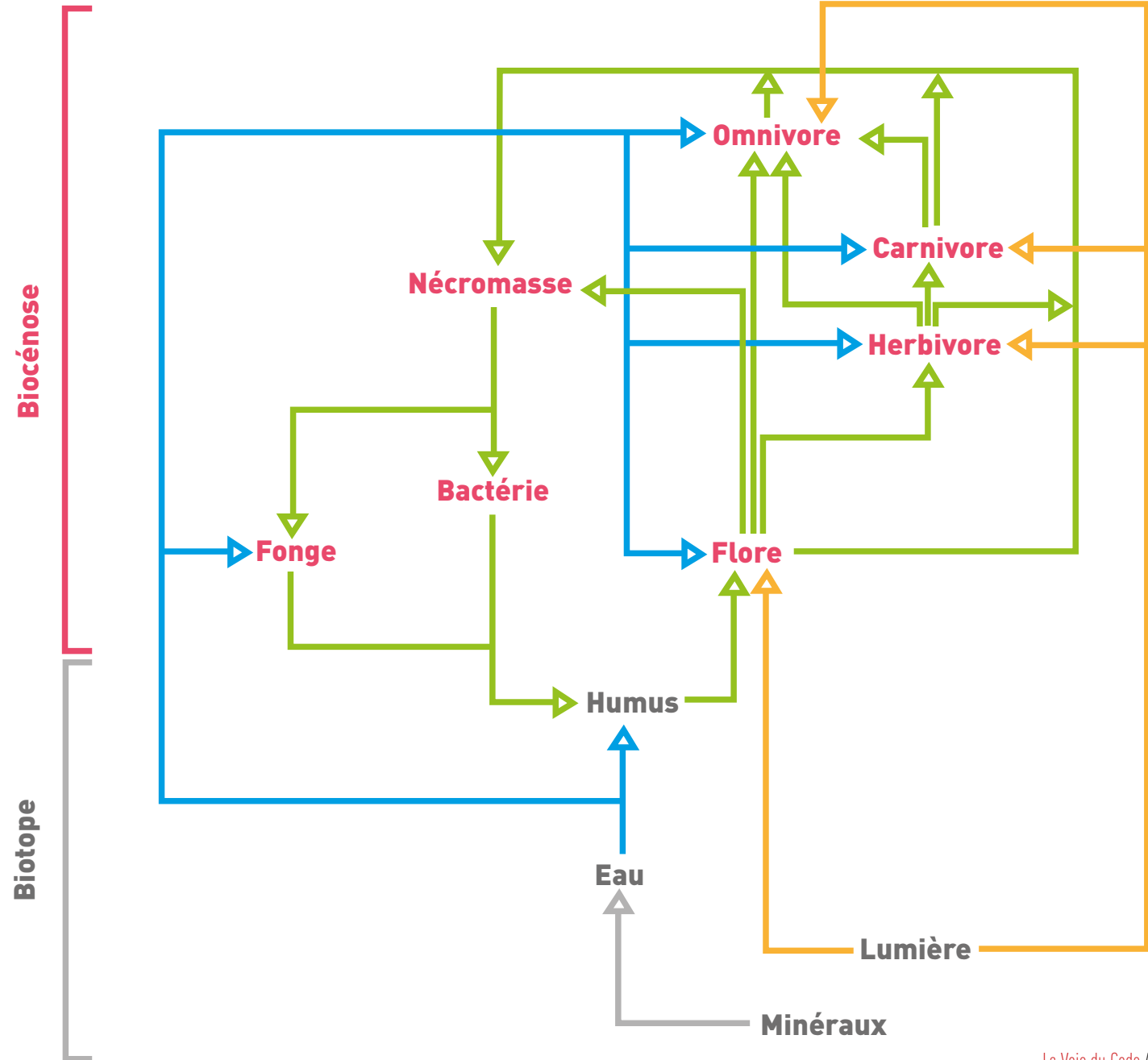
[Andreas Gysin](#)

[Casey Reas](#)

[Marius Waltz](#)

[Raven Kwok](#)

Écosystème Réalisation d'un système multi-agents sous forme d'un écosystème simplifié



Type

byte, boolean, char,
int, long,
float, double,
String

Variable

c'est le nom donné à un type afin de lui attribuer une valeur et de la stocker en mémoire.

Méthode

float name() retourne une valeur de type float, chaque type peut retourner une valeur de son type

void name() exécute une opération mais ne retourne rien. void en anglais signifie vide !

void name(type variable) { } une méthode peut recevoir une ou plusieurs variables.

Quand une méthode est utilisée son type n'a pas besoin d'être écrit, ni le type d'argument qu'elle reçoit, par contre la phrase doit impérativement se terminer pas " ; "

methode(arg) ;

Donc d'un côté on écrit la méthode et de l'autre on l'utilise.

Deux méthodes incontournables de Processing

void setup() {} puis **void draw() {}**

La première initialise votre programme, la seconde le fait tourner !

C'est en partie grâce à ces deux méthodes que Java reconnait la librairie Processing.

Notation

Il est préférable que le nom d'une méthode exprime ce qu'elle exécute.

`void nameMethode() {}` écriture chameau !

ou `void name_methode() {}`

Le but étant que ce soit lisible. Pour cela un système de commentaire existe

`//` permet de rendre une ligne non lisible pour le compilateur

`/* ce texte est inaccessible au compilateur */`

`/**`

Cette ligne peut être mise en avant par certain éditeur de texte

`*/`

`/*`

* Ce texte est rendu accessible au système de documentation Java

`*/`

Forme

`Rect(,,), Ellipse(,,), Point(,,), box(arg), sphere(arg)...`

Apparence

`fill(arg)` et `noFill()` définit la couleur de l'intérieure d'une forme

`stroke(arg)` et `noStroke()` et `strokeWeight(arg)` définit la couleur et l'épaisseur du contour d'une forme

`background(arg)` définit la couleur de fond de votre fenêtre.

Information

une méthode bien utile permet de se tenir informer de que fait Processing
la méthode `println(arg,arg,arg) ;`

Des boucles

Les boucles permettent de répéter une action sans se fatiguer, et c'est important !
Pour voir l'évolution de votre variable, la méthode `println()` est idéale.

```
int arg = 0 ;  
for( int i = 0 ; i < 10 ; i = i + 1) {  
    arg = arg + 1 ;  
    println(arg) ;  
}
```

une boucle peut-être arrêtée, ça peut servir !

```
int arg = 0 ;  
for( int i = 0 ; i < 10 ; i = i + 1) {  
    arg = arg + 1 ;  
    if( arg > 4) break ;  
}  
println(arg) ;
```


Des listes

Processing comme beaucoup de langage de programmation permet de créer des listes variables par type d'objet

```
int [] arg = new int[5] ;
```

Cette petite phrase m'a permis de créer cinq variables int de `arg[0]` à `arg[4]`

Nous pouvons maintenant l'initialiser, c'est à dire lui donner une valeur facilement grâce au boucle

```
int [] arg = new int[5] ;  
for(int i = 0 ; i < 5 ; i = i + 1) {  
    int[i] = i ;  
}
```

Utile

`size(arg,arg)` ; se place en premier dans `le setup()`, il définit la taille de votre fenêtre.
On peut récupérer les coordonnées de la souris avec les variables `mouseX` et `mouseY`.
Et aussi le nombre de boucle de la méthode `draw()` avec la variable `frameCount`
`width` et `height` renvoient la taille de votre fenêtre

Et aussi les coordonnées `0,0` commencent en haut à gauche de votre fenêtre...