ECOSYSTÈME multi-agents

Atelier École multimédia du 8 au 12 février 2016

Déroulé

LUNDI

Présentation

Processing
Différents projets réalisés à l'aide de Processing
Le Projet : Écosysteme 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.

Déroulé

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'instants 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

Cinemetrics

Generative Design

Puddles

Romanesco project

Artistes – Codeurs

Andreas Gysin

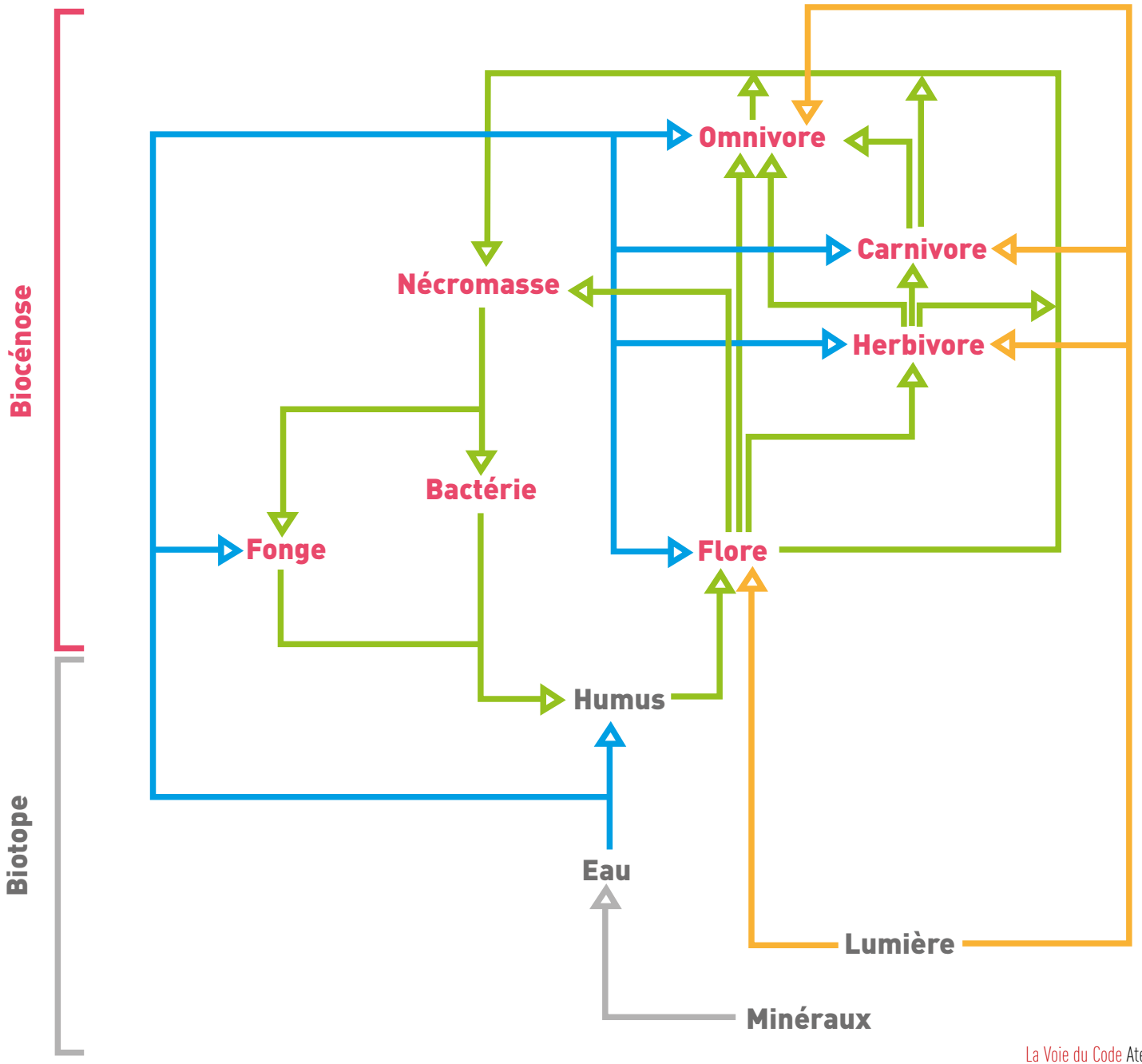
Casey Reas

Marius Waltz

Raven Kwok

V3ga

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



Les bases

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() éxécute une opération mais ne retourne rien. void en anglais signifie vide !
void name(type variable) { } une méthode peut recevoir une ou plusieures 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 éxécute.
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 innaccesible 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.

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...

Information

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

Mon premier sketch

```
void setup() {
   size(400,400) ;
   println(framecount) ;
}
void draw() {
   background(0) ;
   println(framecount) ;
   fill(255) ;
   noStroke() ;
  ellispse(mouseX,mouseY, 50,50) ;
}
```

Les boucles

Des boucles une fonction pour les partisans du moindre effort !

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 ;
   // arg += 1 ;
    if ( arg > 4 ) break ;
}
println(arg) ;
```

Les listes

Des listes une façon simple et efficace de stocker et d'accèder à l'information

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 les initaliser, c'est à dire leur donner une valeur facilement grâce au boucle.

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

Les classes ou programmation orienté objet - POO -

La classe est une partie d'un programme qui peut stocker différentes variables
Elle possède également des méthodes qui lui son propre.
Ces variables et méthodes sont uniquement accesible aux objets issus de cette classe.
Par convention le nom d'une class commence par une majuscule.

```
class C {
    int x ;
    C (int x) {
    this.x = x ;
    }
    void method(int var) {
       x += var ;
    }
}
```

La class est créée de la façon suivante :

```
void setup() {
   int var = 2 ;
   C name = new C (var) ;
}
```

Elle est utilisée au sein d'une autre méthode telle que setup() {} ou draw() {} ou other_method() {}
name.x retourne la valeur de l'argument x de votre classe.
name.method(arg) permet d'utiliser la votre méthode au sein de l'objet que vous aurez créé.

Les ArrayList, les listes dynamiques

Les ArrayList sont des listes dynamiques, où l'on peut ajouter, retirer, remplacer des objet à volonté...
Ces objets doivent être issus d'une classe.
il y a biensûr d'autres fonctions plus avancées.

Déclarer la liste et ajouter des objets à votre ArrayList

```
int var = 1 ;
ArrayList<C> list_name = new ArrayList<C>() ;
void draw() {
    C obj_name  = new C (var) ;
    list_name.add(obj_name j) ;
}
```

Utiliser votre ArrayList. Deux méthodes possibles, la première :

```
for(int i = 0 ; i  <list_name.size() ; i++) {
    C temp_obj = (C ) list_name.get(i) ;
    temp_obj.method(var) ;
}
```

la seconde :

```
for (C obj_name :  list_name ) {
    obj_name .method(var) ;
}
```