ECOSYSTÈME

initiation Processing via un système 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.

MARDI

Mouvement

Position, déplacement, direction

Vecteur

Coordonnées et autres notions mathématiques.

Les méthodes

Utilisations et création de méthode afin de séparer les informations entre le moteur et le rendu.

Déroulé

MERCREDI

Les événements

Récupérer des données de la souris et du clavier

Dessin vectoriel

Comment récupérer des tracés d'illustrator et comment exporter des fichiers PDF

JEUDI

Costume

Donner un costume à notre forme, faire varier sa taille par une méthode trigonométrique.

SVG

Comment manipuler les fichiers SVG – vectoriels – et récupérer les différents groupes.

VENDREDI

Paramétrage des variables

Introduction de la librairie Controle P5 pour créer une mini-interface de contrôle
afin de modifier le rendu de la scène

À propos de Processing

Références

Processing.org
Nature of code par Daniel Shiffman
Openprocessing.org
OpenClassRoom

Projets réalisés à l'aide de Processing

Cinemetrics
Generative Design
Puddles
Romanesco project

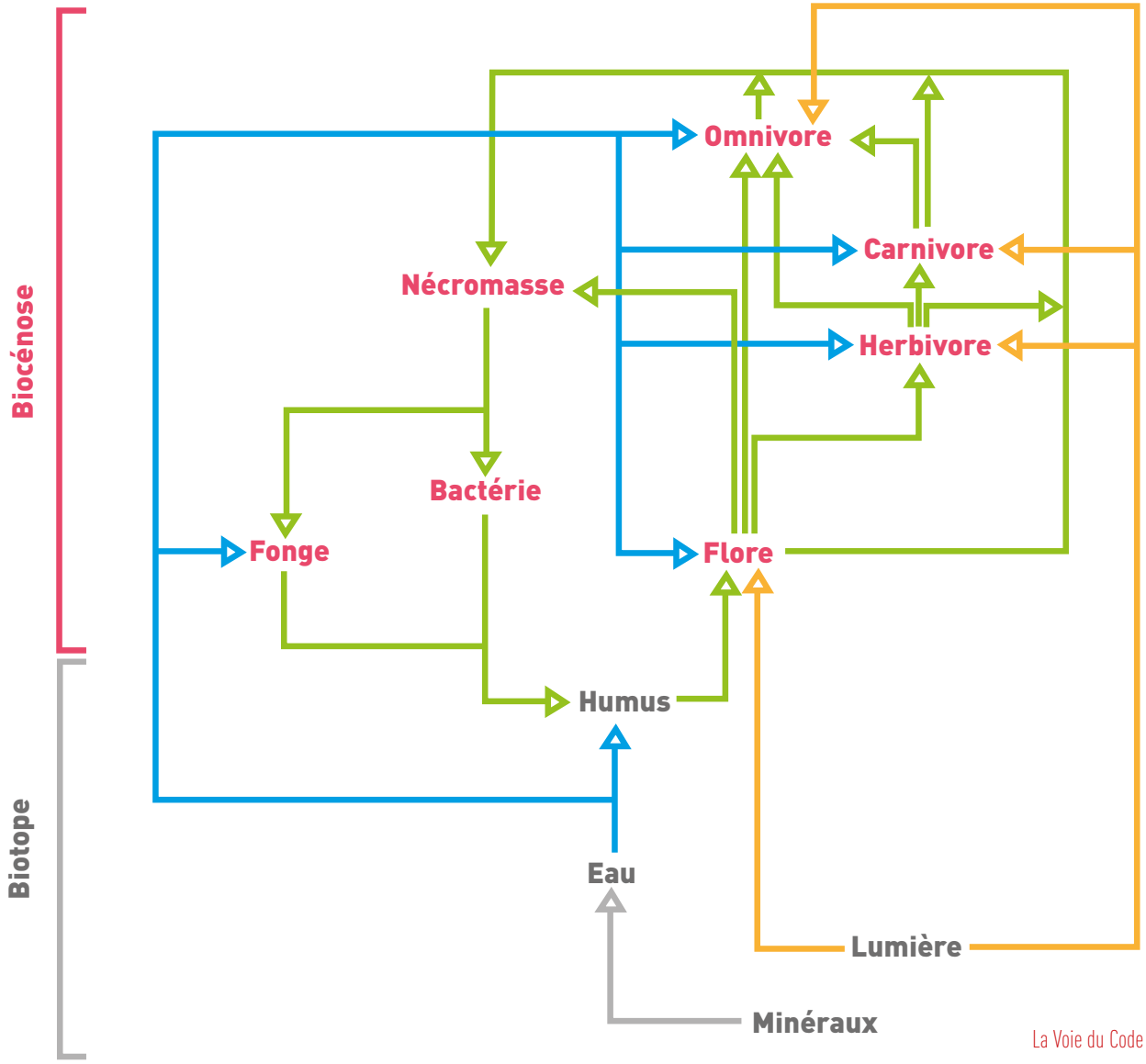
Artistes – Codeurs

Andreas Gysin
Casey Reas
Marius Waltz
Raven Kwok
Stanislas Marçais
V3ga

Le projet

Écosystème

Réalisation d'un système multi-agents sous forme d'écosystème simplifié.
Un agent est un objet de programmation ayant un ou plusieurs comportements lui permettant d'interagir avec son environnement, ainsi qu'avec d'autres agents.
github.com/StanLepunK/Ecosystem



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 mesurer le temps écouler avec la variable frameCount placée dans draw()
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) ;
}
```

La Souris

Les cliques la méthode, cette méthode est globale.
Elle ne doit donc pas être utilisée au sein d'une autre méthode !

```
boolean event, click_right, click_left ;
void mousePressed() {
    if(event) event = false ; else event = true ;
    // bouton droit ou gauche
    if(mouseButton == RIGHT)  click_right = true ; else click_right = false ;
    if(mouseButton == LEFT)  click_left = true ; else click_left = false ;
}
```

Il est possible d'utiliser la variable boolean mousePressed directement dans une méthode

```
int colour = 0 ;
void draw() {
  background(colour) ;
  if(mousePressed) colour = 0  ; else colour = 255 ;
}
```

pour la position deux variables :

```
void draw() {
   println(mouseX,mouseY) ;
}
```

Le clavier

Comme pour la méthode mousePressed() Elle ne doit donc pas être utilisée au sein d'une autre méthode.

```
String letter = "" ;
void keyPressed() {
    if ( key == 'a' || key == 'A' ) letter = "A" ;
}
```

Pour la barre espace, il faut ajouter un espace entre ' ' simple !

Comme pour mousePressed Il est possible aussi d'utiliser
la variable boolean keyPressed directement dans une méthode

```
int colour = 0 ;
void draw() {
  background(colour) ;
  if(keyPressed) colour = 0  ; else colour = 255 ;
}
```

Il existe des touches spéciales qui sont unique accesible avec la variable keyCode
UP, DOWN, RIGHT, LEFT, BACKSPACE, DELETE, RETURN, ENTER, TAB, SHIFT, ALT et CMD est en cours !

```
boolean IAM_HIGH ;
void keyPressed() {
   if(key == CODED) {
      if(keyCode == UP) IAM_HIGH = true ; else IAM_HIGH = false ;
   }
}
```

liste

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 ;
}
```

Charger et utiliser un média externe

Avec Processing vous pouvez utiliser des médias externes comme des images, des films,
des polices de caractères... par défault ces médias se trouvent dans le dossier data à la racine de votre sketch.

Image bitmap

```
PImage img ;
img = loadImage("image.jpg") ;
image(img, 0, 0) ;
```

Image vectorielle

```
PShape mon_svg
mon_svg = loadShape("file.svg") ;
shape(my_svg,0,0,100,100) ;
```

Police de caractère

```
PFont font
font = loadFont("police.vlw") ;
textFont(font, 32) ;
text("word", 10, 50) ;
```

Fichier texte

```
String lignes[] = loadStrings("mon_texte.txt") ;
println("il y a " + lignes.length + " lignes") ;
textSize(12) ;
for (int i = 0 ; i < lignes.length ; i++) {
    text(lignes[i], 12 , (i +1) *14) ;
}
```

Charger et utiliser un média externe

Charger et diffuser un film

```
import processing.video.*;
Movie mon_film;

void setup() {
    size(400,400) ;
    mon_film = new Movie(this, "nom_du_fichier.mov");
    mon_film.loop() ;
}

void draw() {
     image(mon_film, mouseX,mouseY) ;
}

// Utiliser pour actualiser à chaque nouvelle image du film
void movieEvent(Movie m) {
    m.read();
}
```

Class

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.

ArrayList : liste dynamique

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) ;
}
```

Plus loin

Pour aller plus loin en géométrie et en typographie

Une superbe librairie a été dévellopée : Geomerative

Voir le site Geomerative

Pour réaliser des interfaces de contrôle

Le librairie Control P5 donne accès à un ensemble d'outils GUI très utiles

Voir ControlP5

Pour aller plus loin que PVector avec Processing

la class Vec, qui différencie les vecteurs 2D et 3D et qui en plus possède des vecteurs 4D qui bénéfient des mêmes méthodes et aussi Vec5 et Vec6 pour stocker d'autres informations.

Vec sur Github

Pour aller plus loin avec vos fichiers SVG

SVG sur Github

Contact
Stanislas Marçais
stanislas@romanesco.xyz