Initiation à Processing

Florent Grélard
florent.grelard@univ-lyon2.fr

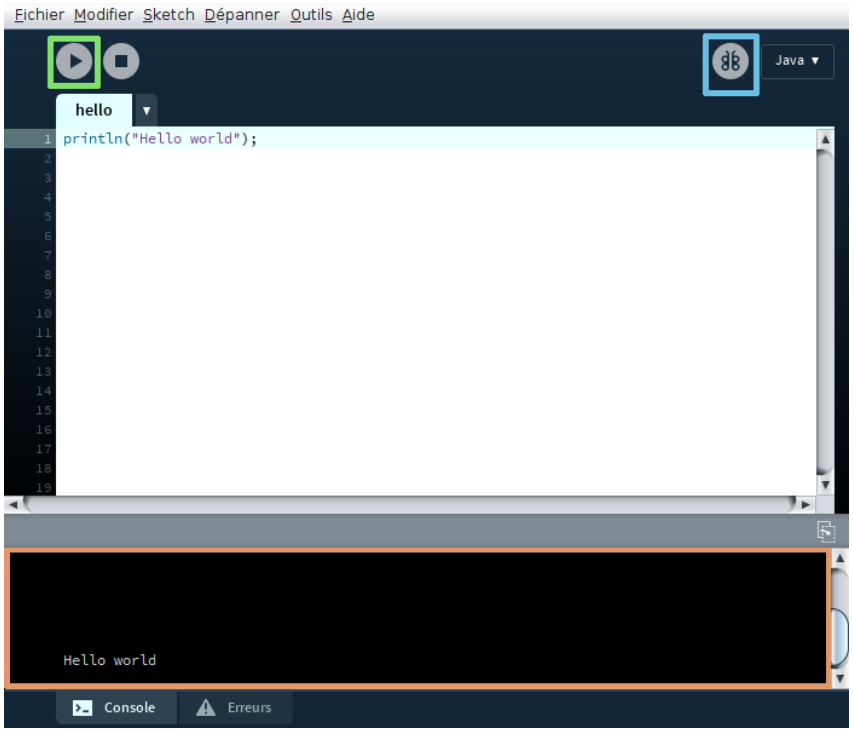
Master 2 Informatique | Conception et Intégration Multimédia, 2018-2019

Sommaire

Présentation

- Bibliothèque et framework **Java** offrant des fonctionnalités dans le contexte des arts visuels.
- Utilisé par des artistes, designers, chercheurs...
- Libre, **portable** (Linux, MacOSX, Windows)
- De nombreux modules disponibles (ex : *OpenCV*)
- Une documentation, de nombreux exemples, des tutoriels : `https://processing.org/`

Interface



Play : exécution du programme

Debug : débuggage du programme, affichage des valeurs de variables

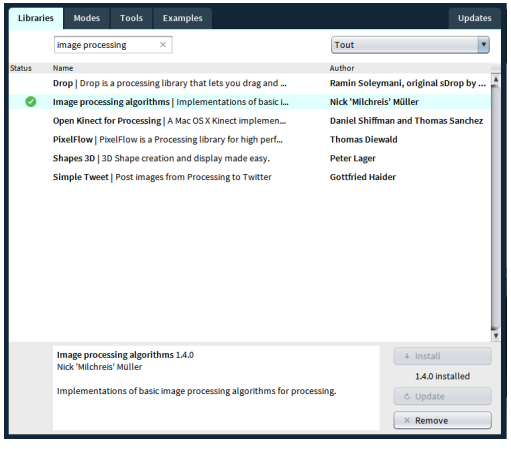
Console : affichage de la sortie standard et des erreurs

Configuration conseillée

Complétion du code : Fichier > Préférences... > Complétion du code avec Ctrl + Espace

Ajout de modules : Sketch > Importer une librairie... > Ajouter une librairie :

- Image processing algorithms : algorithmes de base de traitement d'images
- OpenCV for Processing : traitement et analyse d'images avancés
- Video : manipulation de vidéos, gestion de la caméra.



Sketchs

Sketch : programme + dessin qu'il génère

Enregistrement : un dossier contenant le fichier de code (extension `.pde`) portant tous deux le nom du sketch

Pensez à créer un nouveau sketch à chaque nouveau dessin !

Généralités (1/3)

Syntaxe **Java** :

- langage **typé** : `int, float, boolean, String`
- **accolades** encadrant les instructions des structures conditionnelles, des boucles...
- **points-virgules** à la fin de chaque instruction
- possibilité de définir des **classes**

Exemple :

```
1 int somme = 0;
2 for (int i = 0; i < 10; i++) {
3   somme += i;
4 }
5 println("somme=" + somme);
```

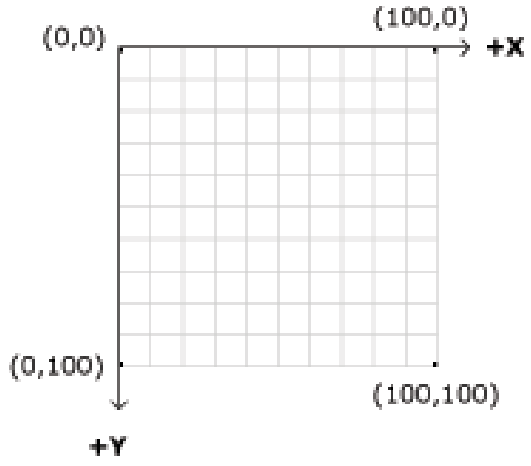
Généralités (2/3)

Structure minimale du code pour un dessin :

```
1 void setup() {
2   //Fonction appelée une seule fois, pour
        initialiser des variables
3 }
4
5 void draw() {
6   //Fonction appelée en boucle, rafraîchit le dessin
7 }
```

Généralités (3/3)

Position dans l'espace : le coin supérieur gauche du dessin correspond à la coordonnée (0, 0)



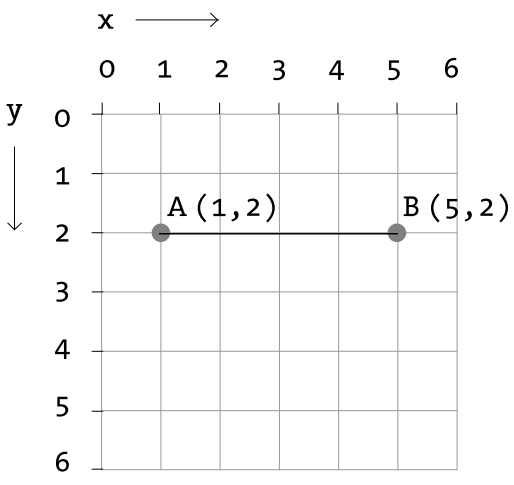
Taille du dessin :

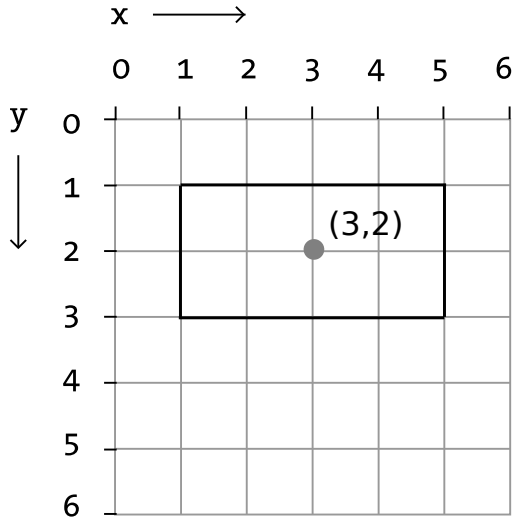
```
1 size(int l, int L); // permet de spécifier la
      largeur l et la longueur L du dessin
2 println(width + "␣" + height); // affiche les
      dimensions spécifiées par size()
```

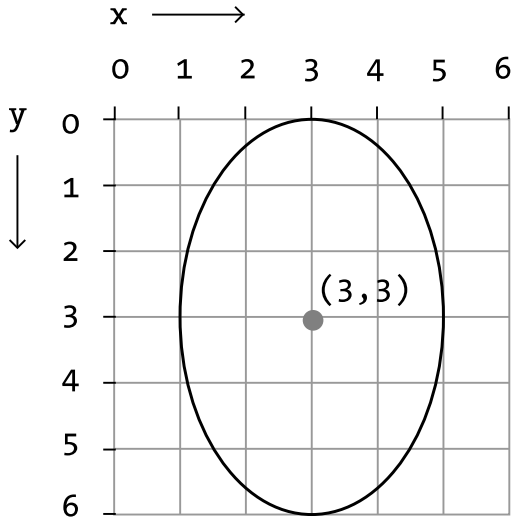
Dessiner des primitives géométriques

Fonctions disponibles :

Ligne: `line(x1, y1, x2, y2);`
Rectangle: `rect(x1, y1, x2, y2);`
Mode rectangle: `rectMode(CENTER);`
Rectangle (2): `rect(x, y, l, L);`
Ellipse: `ellipse(x, y, a, A);`







Couleur et remplissage (1/2)

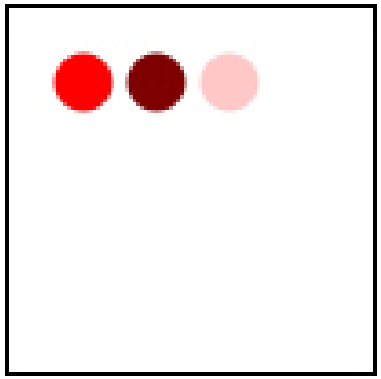
Fonctions disponibles :

Définir une couleur : `color aColor = color(r,g,b,a);`
Arrière-plan: `background(color c);`
Remplissage couleur: `fill(color c);`
Remplissage niveau de gris: `fill(int i);`
Couleur contour: `stroke(color c);`
Pas de contour: `noStroke();`
Pas de remplissage: `noFill();`

Couleur et remplissage (2/2)

Exemple :

```
1  background(255);
2  noStroke();
3
4  fill(255,0,0);
5  ellipse(20,20,16,16);
6
7  fill(127,0,0);
8  ellipse(40,20,16,16);
9
10 fill(255,200,200);
11 ellipse(60,20,16,16);
```



A vous de jouer : ajouter 3 rectangles sous les 3 cercles, avec différentes nuances de vert.

Interaction

Fonctions disponibles :

Détection d'événements: `void mousePressed() {}`

Position de la souris: `mouseX`, `mouseY`

Sommaire

Chargement et sauvegarde d'une image

Classe : PImage.

Fonctions disponibles :

```
Objet image: PImage img;
Chargement: img = loadImage("image.jpg");
Affichage: image(img, x, y);
Sauvegarde: img.save("imagecopy.jpg");
Création image vide: createImage(l, L, RGB);
```

Exemple :

```
1  PImage img;
2  void setup() {
3    img = loadImage("image.jpg");
4  }
5
6  void draw() {
7    image(img, 0, 0);
8  }
```

Stockage d'une image

Image : tableau à deux dimensions

Stocké sous forme de **tableau à une dimension**

How the pixels look:

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

How the pixels are stored:

0	1	2	3	4	5	6	7	8	9	·	·	·		

Modification d'une image (1/2)

Fonctions disponibles :

Lecture de l'image: `img.loadPixels();`
Récupération de l'indice: `color c = img.get(x,y);`
Récupération de l'intensité: `int value = brightness(c);`
Assignation d'une intensité: `img.set(x,y,I);`
Mise à jour de l'image: `img.updatePixels();`

Modification d'une image (2/2)

Exemple : extraction du canal rouge

```
1  PImage img;
2  void setup() {
3    size(200, 200);
4    img = loadImage("couleur.jpg");
5  }
6
7  void draw() {
8    img.loadPixels();
9    for (int i = 0; i < img.width; i++) {
10     for (int j = 0; j < img.height; j++) {
11       color value = img.get(i,j);
12       float r = red(value);
13       color redColor = color(r,0,0);
14       img.set(i,j,redColor);
15     }
16   }
17   img.updatePixels();
18   image(img, 0,0);
19 }
```

OpenCV

Permet d'utiliser des algorithmes de traitement d'images :

- Histogramme, seuillage
- Soustraction de fond
- Reconnaissance de formes : visages...

Utilisation :

Import du module: `import gab.opencv.*;`
Déclaration de l'objet: `OpenCV opencv;`
Instanciation: `opencv = new OpenCV(this, img);`
Utilisation (exemple): `opencv.threshold(120);`

Documentation : `https://github.com/atduskgreg/opencv-processing`

Sommaire

Lecture d'une vidéo (1/2)

Module vidéo :

```
Module: import processing.video.*;
Objet: Movie myMovie;
Instanciation: myMovie = new Movie(this, "movie.mov");
```

Lecture d'une vidéo (2/2)

Exemple :

```
1  import processing.video.*;
2  Movie myMovie;
3
4  void setup() {
5    size(425, 357);
6    myMovie = new Movie(this, "movie.mov");
7    myMovie.loop();
8  }
9
10 void draw() {
11   image(myMovie, 0, 0);
12 }
13
14 //Appel à chaque fois qu'une nouvelle frame est
       disponible à la lecture
15 void movieEvent(Movie m) {
16   m.read();
17 }
```

Modification d'une vidéo

La modification d'une vidéo revient à modifier chaque image la constituant.

Fonction `draw()` :

```
1 void draw() {
2   myMovie.loadPixels();
3   //Faire quelque chose ici
4   myMovie.updatePixels();
5   image(myMovie, 0,0);
6 }
```

Gestion de la caméra (1/2)

Module caméra :

```
Module: import processing.video.*;
Objet: Capture cam;
Instanciation: cam = new Capture(this);
Enregistrement: cam.start(); cam.stop();
```

Gestion de la caméra (2/2)

Exemple :

```
1  import processing.video.*;
2  Capture cam;
3
4  void setup() {
5    size(1300, 768);
6    cam = new Capture(this);
7    cam.start();
8  }
9
10 void draw() {
11   image(cam, 0, 0);
12 }
13
14 //Appel à chaque image dispo
15 void captureEvent(Capture cam){
16   cam.read();
17 }
18
19 void mousePressed() {
20   cam.stop();
21 }
```