Dessiner et animer avec p5.js

Thomas Rey

Lycée Marlioz
10 avril 2018

Table des matières

Vous allez réaliser les exercices de ce document en autonomie et recopier le code de chaque programme demandé dans un document LibreOffice. Pensez à le présenter correctement (comme un devoir) avec obligatoirement les éléments suivants :

- vos Nom(s) - Prénom(s) - Classe en entête ;
- date de réalisation de chaque exercice ;
- numérotation des exercices ;

– code mis en forme avec la police `Courier New` ;

Lorsque vous êtes « bloqués », pensez à utiliser internet pour chercher de l'aide avant d'appeler « au secours » le professeur s'il est déjà occupé ! Le site de référence pour p5.js est https://p5js.org/reference/ ; pensez à le consulter !

À la fin de ce TP vous saurez :
– tracer les formes de base (ligne, ellipse, rectangle, ...)
– dessiner ces formes successivement ;
– utiliser les couleurs ;
– lire les informations de la souris
– ...

1 Pour commencer...

Nous allons coder nos programmes en p5.js. Pour cela, deux possibilités :
– utiliser un éditeur en ligne : https://alpha.editor.p5js.org ou http://p5ide.herokuapp.com/editor#
– créer un fichier html et un fichier `script.js` qui contiennent les lignes suivantes...

Le fichier html :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="utf-8">
    <title>p5.js</title>
    <link rel="stylesheet" href="style.css">
    <!-- cette ligne est à décommenter si on souhaite avoir la dernière version
        de p5.js
        <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.4.23/
            p5.min.js"></script>
    -->
    <script src="p5.min.js"></script>
    <script src="script.js"></script>
</head>
<body>
</body>
</html>
```

Le fichier `script.js` qui contiendra ceci au départ :

```
function setup() {
// à compléter ici
}
function draw() {
// à compléter ici
}
```

Enfin, vous mettrez la page en forme grâce au fichier de style `style.css` et vous aurez aussi besoin d'un dossier `images` pour stocker des fichiers (nous verrons ça plus tard).
Finalement, l'ensemble des fichiers dont vous avez besoin pour débuter est disponible sur l'ENT ou ici (à télécharger et dézipper dans le dossier `ICN` de votre clé ou de votre espace personnel).
Pour chaque exercice, vous téléchargerez une nouvelle version de ce projet « vide » et vous renommerez le dossier au nom de l'exercice.
Il ne vous reste plus qu'à ouvrir le fichier `script.js` avec NotePad++ et le fichier `index.html` avec le navigateur.
Vous êtes prêts à travailler !

2 Méthodes setup et draw

De même que Processing est basé sur le langage Java, p5.js est un langage basé sur le javascript (qui n'a pas de « racine » commune immédiate avec Java).
En javascript (et donc aussi en p5.js) on déclare les variables grâce au mot clé `var` suivi du nom de la variable et éventuellement de son initialisation à une valeur. Ces déclarations sont à faire *avant* le « `function setup()` ».
Par exemple un script pourrait commencer par :

```
var nom = "toto";
var age;
function setup() {

}
```

2.1 La méthode setup

La méthode `setup` est la méthode « lancée » au début du script lorsqu'on ouvre la page html (en fait, il peut y avoir une autre méthode qui se lance avant pour charger les images : la méthode `preload` mais nous verrons cela plus tard). C'est dans cette méthode `setup` que nous allons initialiser un certain nombre de paramètres et de variables.

Exercice 1.

1. Copier le code de l'exemple et compléter ainsi :

```
function setup() {
  // Création d'une fenêtre grise
  createCanvas(400,400);// Dimensions
  background(128);// Couleur de fond
  //On trace un rectangle rempli en rouge
  fill(255,0,0);//remplissage rouge
  rect(100,100,300,100);
  text("un rectangle rouge",100,50);
}
```

2. Ouvrir le fichier `index.html` dans un navigateur (ou cliquer sur le triangle rose dans l'éditeur en ligne).
3. a. Modifier les valeurs `(400,400)` de la ligne 3. Observer les changements en rechargeant la page html.
   b. Même question avec les paramètres `100,100` de la ligne 7.
4. Remplacer la valeur `128` de la ligne 4 par une autre valeur comprise entre 0 et 255. En déduire une explication du rôle de la ligne 4.
5. Remplacer la valeur `128` de la ligne 4 par trois valeurs comprises entre 0 et 255 séparées par des virgules.
6. Remplacer le texte `rectangle rouge` par `carré vert` (ligne 8) et modifier le code pour que le résultat soit cohérent avec le texte.

Exercice 2.
En utilisant la documentation présente ici, écrire un script faisant apparaître des textes dans la fenêtre avec :
- plusieurs tailles (voir `textSize()`) ;
- plusieurs couleurs (voir `fill()`) ;
- plusieurs styles ;
- ...

2.2 La méthode draw

La méthode `draw` est une méthode « lancée » automatiquement un certain nombre de fois par seconde.
C'est dans cette méthode que nous écrirons les instructions de dessin lorsque nous voudrons animer l'écran.

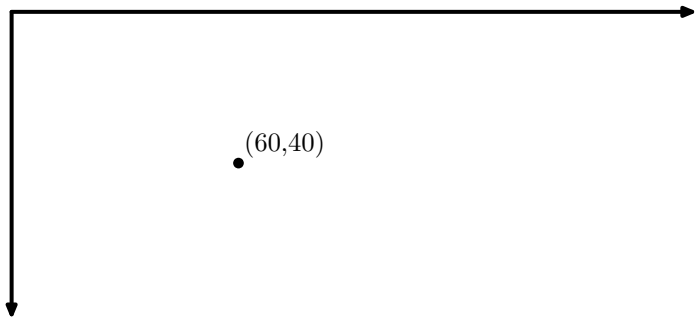
3 Premiers dessins

Avant de voir à quoi sert la méthode `draw()`, nous allons d'abord apprendre à dessiner des formes à l'écran. Dans cette partie, vous écrirez les instructions de dessin dans la méthode `setup()`.

3.1 Repérage

La fenêtre est munie d'un repère orthonormé (comme en maths !) mais orienté différemment du repère habituel (*cf.* ce qu'on a vu en début d'année avec Proccesing) :
- l'origine (de coordonnées `(0,0)`) est située dans le coin supérieur gauche de la fenêtre ;
- le premier axe est horizontal dirigé vers la droite (comme en maths) ;
- le deuxième axe est vertical **mais** dirigé vers le **bas**.

Attention : les coordonnées sont séparées par une virgule (et non pas un point-virgule comme en mathématiques françaises).



Pour une fenêtre définie par `createCanvas(200,100)`, les abscisses vont de 0 à 199 et les ordonnées de 0 à 99.
Enfin, pour en finir avec les généralités, à tout moment du programme, on peut connaître la largeur et la hauteur de la fenêtre à l'aide des mots-clés `width` et `height`. Ceci peut être pratique pour repérer le milieu de l'écran (qui a pour coordonnées `(width/2,height/2)`).

3.2 Formes de base

Les formes de base décrites ici sont les mêmes que celles utilisées en Processing. Petit rappel pour ceux qui ne les auraient pas retenues...

3.2.1 Le point

Pour placer le point de coordonnées $(30; 40)$ à l'écran, on écrit la commande :

```
point(30,40);
```

Attention, et ceci est vrai pour toutes les formes, il faut que les coordonnées du point soient entre 0 et `width-1` pour l'abscisse et entre 0 et `height-1` pour l'ordonnée ; sinon on ne le voit pas !

3.2.2 La ligne droite

On trace un segment [AB] par l'instruction :

```
line(xA,yA,xB,yB);
```

3.2.3 Le rectangle

On trace un rectangle de sommet A, de largeur `larg` et de hauteur `haut` ainsi :

```
rect(xA,yA,larg,haut);
```

Il est aussi possible de modifier le *mode* de définition des rectangles (donner les coordonnées de deux coins opposés ou du centre et des largeurs/hauteurs), si vous en avez besoin, voir ici.

3.2.4 L'ellipse

Une ellipse est une sorte de cercle « aplati ». Pour tracer une ellipse de centre `(50,50)`, de diamètre horizontal 60 et de diamètre vertical 20, on écrit :

```
// ellipse de centre (50,50) de diamètre horiz 60 et vertical 20
ellipse(50,50,60,20);
```

Comme pour le rectangle, il existe plusieurs autres modes décrits ici. Le mode `RADIUS` est sans doute le plus intéressant :

```
// Mode RADIUS : cercle de centre (100,50) et Rayon=20
ellipseMode(RADIUS);
ellipse(100,50,20,20);
```

D'autres instructions permettant de réaliser des formes diverses sur la page des références de p5.js

Exercice 3.
Réaliser un dessin de votre choix avec au minimum un point, une ligne, un rectangle et une ellipse.

3.3 Couleurs

3.3.1 Principe des couleurs

L'affichage des couleurs à l'écran se fait grâce à un « mélange » de trois sources de lumières : rouge, vert et bleu (ou red, green, blue d'où le nom du codage des couleur : RGB). Pour plus de détails, vous pouvez consulter cet article wikipédia.
Avec 256 nuances de chacune de ces trois couleurs, on obtient : $256 \times 256 \times 256 = 16\ 777\ 216$ nuances de couleurs différentes !
Il existe beaucoup d'outils en ligne pour retrouver la « quantité » de chacune de ces trois couleurs dans une teinte choisie. Par exemple, vous pourrez consulter : `http://htmlcolorcodes.com/fr/`
Il existe d'autre façons de coder une couleur en informatique mais nous nous contenterons du codage RGB.

3.3.2 Remplissages et contours

Toutes les figures tracées jusqu'à présent ont un contour et sont remplies.
Pour modifier la couleur de remplissage :

```
fill(R,G,B,t);
// R, G, B quantités de rouge, vert et bleu entre 0 et 255
// t : degré de transparence (0: transparent, 255:opaque)
fill(N);
// N étant entre 0 et 255 (0=noir, 255=blanc)
fill(#hhhhhh);
```

```
// hhhhhh : code hexadécimal de la couleur entre 0 et ffffff
noFill();// Pas de remplissage
```

La couleur de fond se modifie à l'aide de :

```
background(R,G,B);
```

Attention : modifier le fond après avoir construit des figures les efface !
Pour supprimer le contour, on commence par l'instruction :

```
noStroke();
```

Pour modifier la couleur et l'épaisseur du contour :

```
stroke(R,G,B,t);
strokeWeight(2);
```

Exercice 4.
Même consigne que l'exercice précédent mais avec des couleurs.

Exercice 5.
Réaliser quelques-uns des drapeaux suivants :
- le drapeau français (facile) ;
- le drapeau suisse (ça devrait aller) ;
- le drapeau britannique (plus dur !) ;
- le drapeau américain (50 étoiles !) ;
- un drapeau de votre choix.

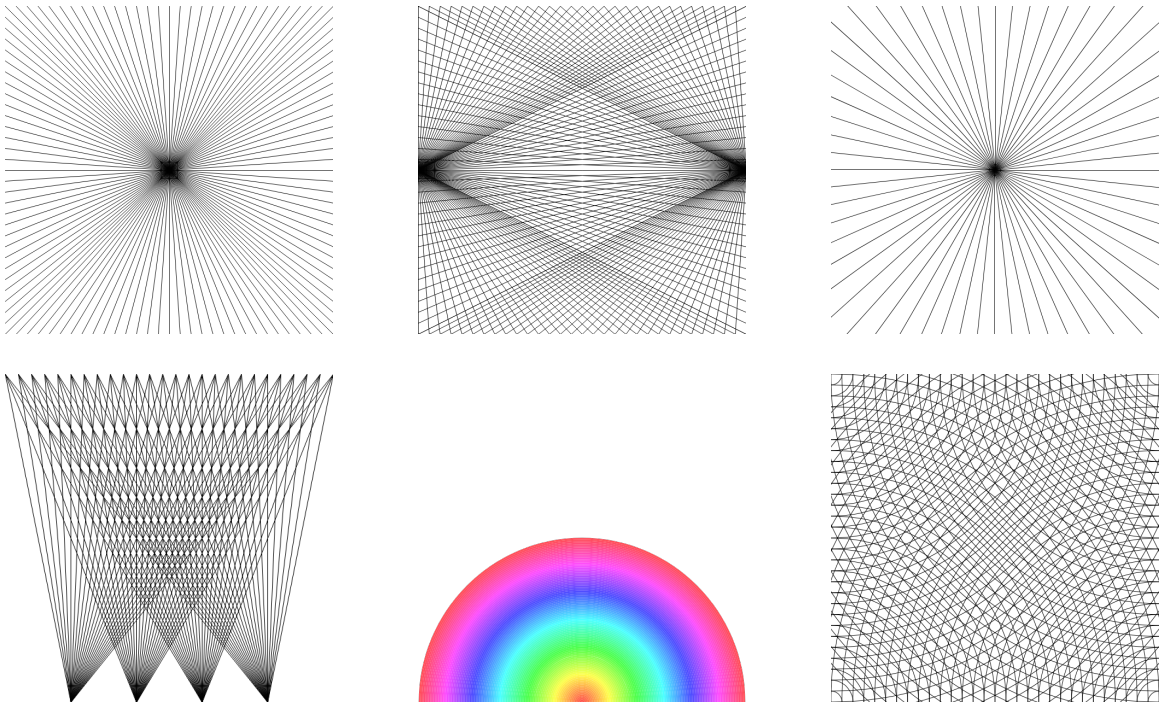
4 Quelques figures imposées pour découvrir les boucles

Exercice 6.
Recopier le code ci-dessous et observer le résultat :

```
function setup() {
  createCanvas(600,600);
  for (lgn=0; lgn<600; lgn = lgn+10) {
    line(0, lgn, 600, lgn);
  }
}
```

1. À quoi sert l'instruction `for` ?
2. Modifier ce programme pour qu'il réalise un quadrillage.

Exercice 7.
Réaliser les figures suivantes (un programme par figure) :



Vous pouvez les voir en plus grand à cette adresse http://reymarlioz.free.fr/icnmarlioz/p5dessin/.

5 La méthode draw pour animer vos dessins

Jusqu'à présent, vous avez écrit tout votre code dans la méthode `setup` : le résultat est *statique* à l'écran. La méthode `draw` va être très utile pour créer des animations (par exemple pour déplacer des personnages dans un jeu).

Exercice 8.
Observer le code suivant (à recopier dans une nouveau projet p5.js) :

```
var lgn = 0;
function setup() {
  createCanvas(600,600);
  frameRate(2);
}

function draw() {
  line(0,lgn, 600,lgn);
  if (lgn < 600) {
    lgn = lgn + 10;
  }
}
```

1. À quoi sert l'instruction `frameRate` ? (Modifier la valeur 2 pour tester)

2. Y a-t-il dans ce script une instruction de boucle (`for` ou `while`) ? Comment expliquer que ce script trace plusieurs lignes ?

Exercice 9.
Écrire un script qui trace successivement des lignes diagonales dans la fenêtre.

Exercice 10.
Réaliser une version animée des figures de l'exercice 7. Des exemples sont disponibles sur cette page http://reymarlioz.free.fr/icnmarlioz/p5dessin/ (mais vous n'êtes pas obligés de respecter ces animations).

6 Gestion de la souris et du clavier

p5.js est en permanence à l'écoute des *événements* pouvant survenir (utilisation du micro, du clavier, de la souris, ...). Dès qu'un événement se produit, il exécute une méthode associée à l'événement détecté (si cette méthode au nom imposé existe).
Quelques noms de méthodes sont listées dans le tableau ci-dessous :

Méthode	Événement associé
`mousePressed()`	un bouton de la souris est appuyé
`mouseReleased()`	un bouton de la souris est relâché
`mouseClicked()`	un bouton de la souris est cliqué (appui + relâche)
`mouseDragged()`	la souris est déplacée avec un bouton enfoncé
`keyPressed()`	une touche du clavier est appuyée
`keyReleased()`	une touche du clavier est relâchée

Certains paramètres relatifs au clavier et à la souris sont également stockés en permanence dans des variables :

Variable	Paramètre enregistré
`mouseX`	abscisse du curseur de la souris
`mouseY`	ordonnée du curseur de la souris
`pmouseX`	abscisse de la position précédente du curseur de la souris
`pmouseY`	ordonnée de la position précédente du curseur de la souris
`key`	dernière touche enfoncée
`keyCode`	code de la dernière touche enfoncée

Exercice 11.
Réaliser les deux animations visibles sur cette page http://reymarlioz.free.fr/icnmarlioz/p5dessin/.

Exercice 12.
Réaliser une animation qui « réagit » à l'action du clavier et/ou de la souris.
Un exemple ici (utiliser la souris, les flèches haut et bas du clavier et cliquer sur le éléments de l'image).

7 Importer des images

Lorsqu'on réalise un jeu ou une application quelconque, il peut être utile de créer les graphismes avec une application dédiée et ensuite d'importer les fichiers images à afficher. En p5.js, rien de plus simple :

- placer les fichiers images (au format `.jpg` ou `.png`) dans le dossier `images` du répertoire où est enregistré votre projet ;
- dans la méthode `function preload()` (à écrire au début du programme), on charge chaque image dans des variables (par exemple `img1`, `img2`, ...) à l'aide d'instructions `img1 = loadImage("images/fichier.png")` (en remplaçant `fichier.png` par le nom de votre fichier bien sûr !) ;
- on affiche l'image `img1` à la position `(100,50)` avec l'instruction `image(img1,100,50)` .

Un exemple à voir en ligne réalisé à partir d'une image d'un des gif faits précédemment :

```
var img1;
var x, y;

function preload() {
  img1 = loadImage("images/
     FannyNatacha.png");
}
function setup() {
  createCanvas(600,600);
}
function draw() {
  x = int(random(600));
  y = int(random(600));
  image(img1,x,y);
}
```