Découverte de P5.js

P5.js est en quelque sorte une bibliothèque pour javascript permettant de faire des applications graphiques. Pour la découvrir on utilisera le site https://www.openprocessing.org ou https://alpha.editor.p5js.org/ (pour du javascript pur on pourra utiliser codepen.io ou jsfiddle.net). Plus tard on verra comment intégrer nos programmes dans une page écrite en html/css/javascript.

Particularité de Javascript / P5.js

Les grands principes de programmations restent les même que pour Python, sauf que javascript est naturellement asynchrone (il n'attend pas qu'une tache soit finie pour lancer la suivante), les variables doivent être déclarées (par contre, contrairement à Java, processing, C++ etc, on n'a pas à préciser le type de la variable) et que sa structure ne repose pas sur l'indentation mais sur des accolades.

Exemple / comparatif *(programme convertissant un nombre en écriture binaire en écriture décimale)*:

Python

```
def convBinaDeci(bina) :
    binaire = str(bina)
    decimal = 0
    for i in range(len(binaire)) :
        decimal += int(binaire[len(binaire)-i-1])*2**i

    return decimal

nombre ("nombre binaire à convertir")
print("écriture décimale : "+ convBinaDeci(nombre))
```

Javascript

```
function  convBinaDeci(bina) {
binaire =bina.toString ();
var decimal = 0;
for (var i =0; i< binaire.length;i++) {
decimal += parseInt(binaire[binaire.length-i-1])*Math.pow(2,i);
}
return decimal
}
var nombre = prompt("nombre binaire à convertir")
alert("écriture décimale : "+ convBinaDeci(nombre))
```

Remarque : chaque commande se termine par un ; du coup on peut maintenant mettre plusieurs commandes sur la même ligne du moment qu'on pense au symbole de séparation.

Particularité de P5.js

P5.js reprends toutes les particularités de javascript et en ajoute quelques une en plus :
Dans tous les programmes on mettra deux méthodes « void » (qui n'ont pas de retour/return)
void setup() {… } qui sert à indiquer des paramètres initiaux s'il y en a , cette méthode est exécutée une fois et une seule au début de l'exécution du programme, elle contiendra par exemple createCanvas(400, 400); qui permet de créer la fenêtre de 400 pixels par 400 pixels dans laquelle on interagira.
void draw() {…} qui sera lue 30 fois par secondes , et qui permettra de rafraîchir ce qu'il y a sur l'écran.
A ces méthodes on pourra aussi ajouter éventuellement les méthodes void keyPressed() { } et void keyReleased() { } qui s'activerons quand on appuiera sur des touches ou qu'on les relâchera.

Ressources :

Pour faire le tour en solo des immenses possibilités de P5.js
Les vidéos de Daniel Shiffman (en anglais) professeur à l'université NYU qui propose des heures d'applications et d'explications
Sur Github la page de Beranger Recoules : https://github.com/b2renger/Introduction_p5js avec beaucoup d'exemples et des explications détaillées.
Les références officielles de différentes fonctions de P5.js , avec exemples à l'appui : https://p5js.org/reference/

Exercice

Couvrir le canevas d'un damier noir et blanc
Couvrir le canevas d'un damier avec des couleurs aléatoires.
Puis couvrir le canevas d'un damier ou les cases noires sont en dégradé de gris (tester plusieurs interprétations de la consigne)
Couvrir le canevas (que l'on fera carré pour cet exercice) d'un damier interactif :

- Version 1 : la case survolée sera bleue
- Version 2 : plus la case « noire » est loin de la souris plus la case sera foncée
  - créer une fonction qui donnera la distance du point dont on donnera les coordonnées en entrée
  - utiliser la commande map pour associer à la distance une intensité de gris

version 3 : rendre la finesse du quadrillage interactive : plus la souris est à droite plus c'est fin (taille minimale de case = 2 pixels) plus la souris est à gauche plus c'est large (taille maximale de case = taille du canevas)

objet : créer une balle rebondissant verticalement entre les bords haut et bas de votre canevas.

Créer une balle rebondissant dans tous les sens (utiliser la loi de réfraction d'optique pour prévoir la suite de son mouvement après avoir rebondit sur une des parois.

Commandes intéressantes

Nom commande	utilité
point(x,y)	Trace un point de coordonnées x et y
line(xA,yA,xB,yB)	Remplira les figures de la nuance de gris choisie
rect(x1,y1,xdim,ydim)	Trace le segment d'extrémités A et B
ellipse(cx,cy,rx,ry)	Dessinera une ellipse dont le centre aura pour coordonnées (cx,cy) et donc les rayons seront (rx,ry)
curve(0, 300, 10, 60, 90, 60, 200, 100);	Trace une courbe commençant par les 4 points dont les coordonnées ont été indiqués
quad(x1,y1,x2,y2,x3,y3,x4,y4)	Trace un quadrilatère passant par les 4 points.
triangle(x1,y1,x2,y2,x3,y3)	Trace un triangle passant par les 3 points.
background(r,g,b);	Couleur de fond
smooth()	Active le lissage
strokeWeight()	Change l'épaisseur du trait
stroke(r,g,b)	Choisit la couleur des contours (format r,g,b)
fill(r,g,b)	Remplira les figures de la couleur RGB choisie R, g et b sont compris entre 0 et 255
fill(g)	Remplira les figures de la nuance de gris choisie
fill(r,g,b,alpha)	Indique le facteur de transparence (de 0 à 255)
noFill();	Indique que la figure ne sera pas remplie
translate(xt,yt)	Augmentera toutes les coordonnées de (xt,yt)
rotate (angle)	Fait tourner les figures à venir de l'angle choisit L'angle étant en radian on utilisera PI
rotateX(angle), rotateY(angle)…	
scale(coeff)	Change l'échelle, toutes les distances et positions sont multipliées par le coeff, s'il est <1 c'est une réduction, s'il est plus grand que 1 c'est un agrandissement
map(valeur,bme,bMe, bms,bMs)	Prend comme entrée(s) une valeur comprise entre bme et bMe, et donne sa position correspondante entre bms,bMs (Penser Thalès)
image(variable,xpos,ypos); image(variable,xpos,ypos, taillex, tailley);	Affichera l'image au bon endroit , on peut éventuellement ajuster sa taille.