Informatique et Science du Numérique

Thomas Rey

Classe de Terminale S
6 décembre 2016

Table des matières

TP 1

Processing

Processing est un environnement de programmation inventé en 2001 par et pour des artistes créant des oeuvres numériques ou multimédias. Il est basé sur le langage Java mais son attrait principal est la simplicité d'utilisation. Ce logiciel est libre, gratuit (téléchargeable à cette adresse https://processing.org/), multiplateforme (c'est-à-dire qu'il fonctionne sous Windows, MacOS et Linux) et il permet de créer facilement des applications pour Android.
Nous allons détailler dans ce document quelques fonctionnalités de base qui nous permettront de bien débuter avec ce logiciel, puis quelques instructions élémentaires de Java pour écrire nos premiers programmes. Mais d'abord, pour illustrer les principes de Processing, nous allons étudier un programme...

1.1 Une activité

On a vu la semaine dernière un algorithme de conversion d'un nombre écrit en binaire vers notre écriture décimale. Voici sa traduction en langage Processing :

```
1  // Importer la bibliothèque Java permettant de créer une boîte de dialogue :
2  import javax.swing.*;
3
4  int NBinaire, NDecimal, k;
5  String NBinaireChaine;
6  NDecimal = 0;
7  NBinaireChaine="";
8
9  size(600,200);// Création de la fenêtre
10 background(255);// Fond blanc
11 fill(0);// Écriture en noir
12 textSize(20);// Taille du texte
13
14 text("Conversion d'un binaire en décimal",20,20);
15 NBinaireChaine = (String)JOptionPane.showInputDialog(null,"Saisir un nombre en
       binaire (uniquement des 0 et des 1)", "Dialog", JOptionPane.PLAIN_MESSAGE);
16 k = NBinaireChaine.length();
17 text("Nombre en binaire : " + NBinaireChaine,20,50);
18 NBinaire = parseInt(NBinaireChaine);
19
20 while (NBinaire >0) {
```

```
21    int chiffre = (int)(NBinaire/pow(10,k));
22    println("chiffre = " + chiffre);
23    NDecimal = (int)(NDecimal + chiffre * pow(2,k));
24    NBinaire = NBinaire - chiffre * (int)(pow(10,k));
25    k = k - 1;
26  }
27  text("Nombre en Décimal : "+NDecimal,20,75);
```

1. Que sont les expressions `NBinaire`, `NDecimal`, `k`, ... ?
2. Quelle est la signification des mots clés `int` et `String` qui les précèdent ?
3. Que remarquer sur les fins d'instructions d'un programme Processing ?
4. Quel est le rôle des lignes 14, 17, 27 ?
5. Quel est le rôle du « `20,20` » à la fin de la ligne 14 ?
6. À quoi sert l'instruction `parseInt` de la ligne 18 ?
7. Comment s'appelle la partie du programme comprise entre les lignes 20 et 26 incluses ?

1.2 Démarrer avec Processing

Pour écrire un premier programme simple, par exemple demander le poids et la taille d'un individu et calculer l'IMC (Indice de Masse Corporelle) de cet individu. Nous allons donc créer un *sketch* que nous appelerons `IMC`.
L'algorithme de notre projet est donc celui-ci :

Entrées : m, t et imc sont trois variables numériques;
Sorties : imc contient $\frac{m}{t^2}$;
début
    Demander la masse m ;
    Demander la taille t ;
    Calculer $imc \leftarrow m/t^2$;
Afficher imc

Algorithme 1 : Calcul de l'IMC

Voici comment programmer ce « sketch » avec Processing maintenant :
- lancer le logiciel Processing ;
- dans le menu `File` sélectionner `Préférences` et choisir le dossier où seront enregistrés vos projets : un répertoire `Processing` de votre dossier personnel ;
- dans le menu `File` sélectionner `Save As` et lui donner le nom `imc` ;
- voilà ! Vous pouvez programmer...

Complétez votre programme comme sur le modèle ci-dessous :

```
1  // Importer la bibliothèque Java permettant de créer une boîte de dialogue
2  import javax.swing.*;
3
4  int m, t;// int indique que les variables sont entières
5  float imc;// float indique que la variable est décimale
6  String reponse;
7
```

```
8  size( , );
9  textSize( );
10
11 text("Calcul de votre IMC", , );
12 // Demander la taille :
13 reponse = (String)JOptionPane.showInputDialog(null,"Saisissez votre taille en cm
       ", "Taille", JOptionPane.PLAIN_MESSAGE);
14 t = parseInt(reponse);
15
16 // Demander la masse :
17
18
19 // Calculer l'IMC
20 imc =
21
22 // Afficher le résultat dans la fenêtre :
23 text(
```

Remarques :

- tout ce qui suit un « // » n'est pas lu par Java (et donc par Processing) : il s'agit de commentaires du programme (ils sont essentiels pour la bonne compréhension du lecteur) ;
- par convention, une *variable* commence toujours par une lettre minuscule ;

Pour exécuter le programme, enregistrer le sketch puis cliquer sur l'icône de lecture (le triangle).

1.3 Un peu plus loin...

1.3.1 Quelques notions sur les méthodes

En règle générale, un projet informatique est beaucoup plus long que ces quelques lignes. Il doit donc être organisé en plusieurs parties bien distinctes ; chacune réalisant une opération la plus élémentaire possible.
Dans nos (modestes) projets, nous utiliserons des *classes* (bientôt) pour définir des nouveaux *objets* et nous décomposerons nos programmes en créant des *méthodes*.
Une méthode est une sorte de sous-programme qui effectue une tâche (plus ou moins) élémentaire. Cette tâche peut être de deux types :

- soit elle renvoie un résultat de type `int`, `String`, ... ; on parle alors de *fonction* ;
- soit elle ne renvoie pas de résultat (elle peut par exemple afficher une information à l'écran) ; on dit parfois que c'est un sous-programme.

Exemple : une méthode de type `int` qui renvoie la somme des deux entiers passés en paramètres :

```
1 int addition(int a, int b) {
2   return a+b;
3 }
```

Dans notre programme on pourra écrire `int` somme = addition(2,3); pour que la variable `somme` contienne la valeur 5.

Exemple : méthode qui affiche la somme de deux entiers (et qui utilise la méthode de l'exemple précédent).

```
1 void affiche(int a, int b) {
2   text(addition(a,b), 30,30);
3 }
```

À noter le mot clé `void` qui indique que la méthode ne renvoie pas de résultat (ce n'est pas une « fonction »).
Le nombre de fois où la méthode `draw`() est exécutée par seconde est défini par l'instruction `frameRate`(i) où `i` est un entier.
Attention : une variable qui est définie dans une méthode n'est « connue » et donc utilisable que dans cette méthode.

1.3.2 Quelques méthodes particulières

Il existe en Processing (au moins) deux méthodes particulières :
- la méthode `setup`() qui, si elle existe, est lancée automatiquement au démarrage du programme ;
- la méthode `draw`() qui, si elle existe, est lancée automatiquement un certain nombre de fois par seconde et sert à gérer les affichages.

Ces deux méthodes sont de type `void`.

1.4 Exercices

Exercice 1.1.
Dans chaque cas écrire un programme qui :
- demande le prénom de l'utilisateur et affiche « bonjour » suivi du prénom ;
- demande le prénom, le nom, le numéro de téléphone, ... et affiche une phrase avec ces éléments ;
- demande deux nombres et affiche le résultat des quatre opérations élémentaires ;

Exercice 1.2.
Créer une fonction qui prend en paramètre une question (de type `String`) et retourne un entier qui répond à la question.
Modifier alors le programme `imc` en utilisant cette fonction.

Exercice 1.3.
Écrire un programme qui convertit en binaire un nombre écrit en notation décimale.

TP 2

Variables

Dans le premier TP, nous avons déjà utilisé des *variables*, nous allons préciser ici leur utilisation, les différents types simples, mais aussi nous allons commencer à découvrir ce qui fait la force des langages *orientés objet* comme Java (et donc Processing) : les *classes* qui permettent au programmeur de créer ses propres objets.

2.1 Variables

2.1.1 Définition

Une *variable* est un espace mémoire dans lequel le programme peut stocker de l'information qui peut être numérique, booléenne, alphanumérique, ... On peut « modéliser » une variable comme étant une boîte qui contient *une* information qui peut varier (d'où son nom !) au cours de l'exécution du programme. À noter :

- il existe plusieurs *types* de variables : entière, décimale, alphanumérique, ... ; nous les décrirons plus loin ;
- en Java (et donc en Processing [1] !), une variable doit être *définie* avant d'être utilisée ;
- une fois définie, la variable ne peut pas changer de type (si vous définissez une variable *entière* `maVariable`, elle ne peut pas contenir le décimal 2,5) ;
- concrètement, une variable est désignée par une suite de lettres et de chiffres ainsi que certains caractères (par exemple `_` et `-`) et commence toujours par une lettre minuscule (attention les variables `maVar1` et `mavar1` sont différentes : le nom de variable est sensible à la *casse*).

Pour résumer une variable est constituée de trois informations :

un identificateur : c'est-à-dire son nom ;

un type : c'est-à-dire la description des informations qu'elle contient ;

une valeur : c'est-à-dire son contenu.

1. Processing étant basé sur Java, la plupart des consignes valables pour Java le sont aussi pour Processing. Je ne le répéterai plus à chaque fois...

2.1.2 Les types simples en Java

Nous allons décrire ici les principaux types « primitifs » :

- le type `int` : permet de définir un nombre entier compris entre -2^{31} et $2^{31}-1$ (il est codé sur 32 bits, c'est généralement suffisant [2]) ;
- le type `long` : permet de définir un nombre entier compris entre -2^{63} et $2^{63}-1$ (il est codé sur 64 bits et là c'est carrément énorme [3]) ;
- il existe aussi les types bytes et `short` codés respectivement sur 8 et 16 bits ;
- le type `float` : permet de définir des nombres décimaux (on dit à virgule *flottante*) ;
- les types réels (ou plus précisemment décimaux en maths) sont `float` et `double` (le second étant plus précis) ;
- le type `char` permet de déclarer un caractère par exemple 'a' ou '$' ou ... ; il est codé sur 16 bits ;
- le type `boolean` qui ne peut prendre que deux valeurs : « VRAI » ou « FAUX ».

Exemple : déclaration de variables en Java

```
1    int maVariable, h, monCompteur3;// déclaration "simple"
2    int monAge = 17;// déclaration et initialisation
3    float x;
4    double y;
```

Attention : il est toujours préférable d'initialiser la valeur d'une variable pour éviter des erreurs de compilation.

2.1.3 Affectations et opérations

Quelques exemples :

- affecter la valeur 5 à la variable `a` s'écrit : `a = 5;`
- augmenter la valeur de la variable `x` de 3 s'écrit : `x = x + 3;` ou encore `x += 3;`
- donner à une variable le résultat d'un calcul : `a = 4 + 3 * 5;` et la variable `a` contiendra 19 (priorités opératoires)
- incrémenter la variable `i` (c'est-à-dire l'augmenter de 1) : `i++;`
- tester une égalité `boolean vf = (a == b);` met `true` dans la variable `vf` si les deux variables `a` et `b` ont la même valeur et `false` sinon.

À noter que le type `char` est particulier car on peut effectuer des opérations de comparaison dessus ; par exemple :

```
1 char a = 'a';
2 char b = 'e';
3 println(a < b);
```

Exercice 2.1.
Quel sera l'affichage suite à l'exécution du programme précédent ?
Même question en remplaçant le « `'e'` » de la ligne 2 par un « `'E'` » ?

2. Pour info, $2^{31} = 2\ 147\ 483\ 648$.
3. $2^{63} = 9\ 223\ 372\ 036\ 854\ 775\ 808$

Enfin, il est possible d'effectuer des conversions ou *cast* de types. Par exemple l'expression suivante est incorrecte : `int a = 12.7;` il faut écrire `int a = (int)12.7;` et la variable `a` contiendra la valeur 12 (troncature) et `(int)`-12.7 donne -12 (la conversion des décimaux en `int` n'est donc pas la même chose que la partie entière en maths !)
Attention : l'instruction `float a = 5 / 4;` donne à `a` la valeur 1 (et non pas 1,25) car 5 et 4 sont des entiers, Java calcule donc avec eux comme si le résultat était entier. Il faut écrire (au minimum) : `float a = 5f / 4;`
De manière générale, lorsqu'on initialise un `float`, on lui ajoute un « `f` » à la fin de sa valeur numérique ; par exemple le code suivant ne pose pas de problème :

```
1 float a = 5f;
2 float b = 4f;
3 float q = a/b;
```

Exemple : on peut facilement retrouver le « code » de chaque caractère en castant en `int` un `char`. Que fait le programme suivant :

```
1 char a = 'a';
2 int code = (int)a;
3 code++;
4 a = (char)code;
```

Exercice 2.2.
Retrouver le code du caractère `a`, du `e` et celui du `E` et expliquer le résultat de l'exercice 2.1.

2.1.4 Les chaînes de caractères

Un texte est appelé en informatique *chaîne de caractères*. En Java, une chaîne ou `String` n'est pas un type *simple* comme `int`, `float`, ... La définition d'une chaîne se fait avec le mot clé `String` (on remarquera la majuscule au début...) :

```
1 String maChaine = "Bonjour, ";
2 String monNom = "Bill Gates";
3 println(maChaine + monNom);
```

Quelques opérations sur les chaînes (dans le tableau suivant str, str1, str2 sont des chaînes de caractères) :

Méthode	Description
str.length();	donne la longueur (le nombre de caractères) de la chaîne `str`
str = str1 + str2;	*concatène* les deux chaînes (elles sont mises « bout à bout »)
char c = str.charAt(4);	donne le 5e caractère (celui d'indice 4) de la chaîne str
int i = str.indexOf('c');	donne l'indice de la première apparition du caractère c
str = str1.substring(3,6) str1="Chocolat"; str=str1.substring(3,6)	extrait de la chaîne str1 les caractères d'indices 3 à 5 inclus str contient "col"

Méthode	Description
str1.equals(str2);	donne un booléen qui permet de tester si les deux chaînes ont le même contenu
(str1 == str2)	donne un booléen qui permet de tester si les deux chaînes sont les mêmes
str.toUpperCase();	met toute la chaîne str en majuscules
str.toLowerCase();	met toute la chaîne str en minuscules
str1.compareTo(str2);	compare dans l'ordre lexicographique (voir aussi str1.compareToIgnoreCase(str2);)

Exercice 2.3.
Voici un morceau de programme :

```
1  String str1, str2, str3;
2  str1 = "Voici un exemple formateur";
3  str2 = str1;
4  str3 = new String(str1);
5  println("chaine 1 = " + str1);
6  println("chaine 2 = " + str2);
7  println("chaine 3 = " + str3);
8
9  println("chaine 1 et chaine 2 ont le même contenu ? " + str1.equals(str2));
10 println("chaine 1 et chaine 2 sont le même objet ? " + (str1==str2));
11
12 println("chaine 1 et chaine 3 ont le même contenu ? " + str1.equals(str3));
13 println("chaine 1 et chaine 3 sont le même objet ? " + (str1==str3));
```

Question : où seront les `true`, où seront les `false` dans l'affichage ?

2.2 En mémoire

Dans la mémoire de l'ordinateur chaque donnée est stockée sous la forme d'une suite de 0 et de 1. Nous allons voir dans cette partie comment, à partir de 0 et de 1, stocker des informations aussi diverses que des entiers, des décimaux, du texte, ...
Le bit est une valeur 0 ou 1 (qui correspond à « le courant ne passe pas » ou « le courant passe » dans le circuit). Grâce à un bit, on peut donc compter jusqu'à deux (en commençant à un).
En utilisant le principe de la numération de position, on peut ensuite écrire des nombres plus grands : `0`, `1`, `10`, `11`, `100`, `101`, ... (on dit qu'on compte en base 2).
Un octet (ou byte) est un regroupement de 8 bits. Il permet donc de stocker les nombres de `0` à `11111111` soit 255 (en base 10).
En Java, le type `int` est codé sur 4 octets, le type `long` sur 8 octets et le type `float` sur 4 octets (il existe aussi le type `long` codé sur 8 octets).
Exercice 2.4.
Expliquer pourquoi un `int` est compris entre $-2\,147\,483\,648$ et $2\,147\,483\,647$.

Pour stocker les caractères, il a fallu inventer un codage. Le premier codage des caractères fut le code ASCII (American Standard Code for Information Interchange). Ce codage associait à

chaque caractère de l'alphabet latin (et de quelques symboles) une valeur numérique entre 0 et 255 :

code	caractère	code	caractère	code	caractère
34	"	48	0	97	a
35	#	49	1	98	b
36	$	50	2	99	c
37	%	65	A	61	=
38	&	66	B	64	@

Cette table (ici incomplète), est insuffisante pour désigner tous les caractères de tous les alphabets. Depuis, l'*Unicode* a été inventé, il contient près de 110 000 caractères. Il existe en plusieurs déclinaisons : *UTF-8*, *UTF-32*, ...
Une table des principaux caractères de la norme `iso-latin-1` est disponible en annexe A

2.3 Objets

Dans cette partie, nous allons survoler la notion d'objet. Ceci vous permettra de comprendre certains programmes que vous pouvez trouver sur internet. Nous y reviendrons plus en détails plus tard dans le TP 9.
Un objet en Java est une sorte de « super-variable » qui est défini par une *classe*. Cette classe est une sorte de description des éléments communs de tous les objets de cette classe par exemple dans un jeu de paris sur des courses de chevaux, il faudrait créer une *classe* `Joueur` grâce à laquelle nous définirions des *objets* `joueur1`, `joueur2`, ... Ces objets de type `Joueur` auraient tous :

- un nom (mais chacun le sien) ;
- un capital ;
- une mise ;
- un cheval choisi ;
- ...

Ces éléments sont appelés les *variables de la classe* `Joueur`. Mais ils auront aussi des *méthodes* qui permettent de créer des sortes d'actions sur le joueur :

- créditer le capital en cas de victoire ;
- débiter le montant du pari ;
- choisir un cheval ;
- ...

Exemple : un extrait d'une classe `Joueur` :
Cette classe est définie par le mot clé `class`. Elle possède trois variables `nom`, `capital` et `mise`. Une *méthode constructeur* est définie : c'est le code qui est executé lorsqu'un objet de type `Joueur` est *instancié* c'est-à-dire lorsqu'on crée une variable de ce type dans le programme. Et enfin, elle est munie d'une *méthode* (nous reviendrons plus tard sur les méthodes) qui permet d'augmenter la variable `capital`.

```
1 public class Joueur {
2   String nom = new String("Joueur");
3   int capital = 20;// Capital de départ
```

```
4   int mise = 0;
5
6   // Méthode "constructeur" :
7   public Joueur(String nomSaisi, int sous) {
8     // Ci-dessous on "construit" le joueur avec les données de départ :
9     this.nom = nomSaisi;
10    this.capital = sous;
11  }
12
13  // On crédite en cas de victoire
14  public void gagne(int gain) {
15    this.capital = this.capital + gain;
16    this.gain = gain;
17  }
18
19 }// Fin de la classe Joueur
```

Devinette : à quoi peut bien servir le mot clé `this` ?
Dans un programme qui utilise la classe `Joueur` on aurait par exemple :

```
1 // Création du joueur 'joueur1' qui s'appelle Thomas et possède 150000 euros
2 Joueur joueur1 = new Joueur("Thomas", 150000);
3 // On lui crédite 10 euros de plus
4 joueur1.gagne(10);
```

2.4 Exercices divers

Exercice 2.5.
Que contient `c` à la fin de l'exécution du programme suivant ?

```
1 int a = 3, b = 5, c;
2 c = a * (b / a);
```

Exercice 2.6.
Que fait le morceau de programme suivant (`a` et `b` sont des variables de type `int`) :

```
1 a = a + b;
2 b = a - b;
3 a = a - b;
```

On pourra utiliser les tableaux qu'on complétera – d'abord avec des valeurs numériques puis avec des inconnues (au sens mathématiques) x et y – au fur et à mesure des étapes :

a	b

a	b

a	b

Exercice 2.7.
Écrire un programme qui demande le nom et le prénom de l'utilisateur et l'écrit ensuite sous la forme : « NOM Prénom » (en respectant la casse).

Exercice 2.8.
Voici le code d'un programme :

```
1 String mot = "anticonstitutionnellement";
2 int n = 0;
3 n = mot.indexOf('t'); println(n);
4 n = mot.lastIndexOf('t'); println(n);
5 n = mot.indexOf("ti"); println(n);
6 n = mot.lastIndexOf("ti"); println(n);
7 n = mot.indexOf('x'); println(n);
8 n = mot.indexOf('t',5); println(n);
```

Quel sera le résultat de ce programme ?

Exercice 2.9.
Retrouver le texte représenté en ASCII binaire par la suite de bits suivants :
`01000011001001110110010101110011011101000010000001100110011000010110001101101001 0110110001100101`

2.5 Compléments

Uniquement dans la version online (à écrire quand j'aurai le temps !) :
méthode `toString()` à écrire dans la définition d'une classe.

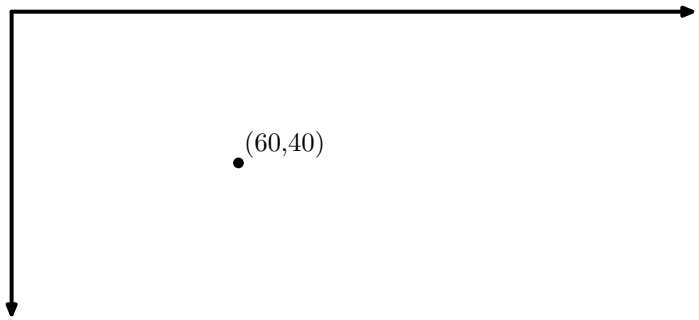
TP 3

Dessiner à l'écran

Dans le TP 1 nous avons vu l'instruction `size`(larg,haut) qui permet de créer une fenêtre graphique de dimensions `larg` et `haut`. Nous avons vu aussi que l'instruction `text`(*"mon texte"*,20,30) permet d'afficher `mon texte` au point de coordonnées $(20;30)$ dans cette fenêtre. L'objet de ce TP est maintenant de dessiner à l'écran des lignes, cercles, disques, polygones, ... et par la suite d'animer ces dessins.

3.1 Repérage sur l'écran

Le système de repérage de la fenêtre graphique utilise le pixel comme unité et le repère a pour origine le coin supérieur gauche de la fenêtre. Le premier axe est horizontal et dirigé vers la *droite*, le second est vertical est dirigé vers le *bas*. À noter que les coordonnées des points et les dimensions utilisées sont des nombres *entiers*. Enfin, ces coordonnées sont séparées par une virgule (et non pas un point-virgule comme en mathématiques françaises).



Pour une fenêtre définie par `size`(200,100), les abscisses vont de 0 à 199 et les ordonnées de 0 à 99.
Une particularité de Processing est qu'il permet aussi de gérer la 3D. Dans ce cas, l'instruction `size`() prend un troisième argument : `P3D`. Pour définir une zone de dessin en 3D on écrit par exemple `size`(600,400,P3D).
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 aura pour coordonnées (`width`/2,`height`/2)).

3.2 Les formes de base

3.2.1 Le point

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

```
1 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 ! (Mais Processing ne vous renverra pas d'erreur).

3.2.2 La ligne droite

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

```
1 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 :

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

Il est aussi possible de configurer Processing pour que le point A soit l'intersection des diagonales :

```
1 rectMode(CENTER);// A ecrire une seule fois
2 rect(xA,yA,larg,haut);
```

Et pour revenir à la situation initiale (A est un sommet du rectangle) :

```
1 rectMode(CORNER);// A ecrire une seule fois
2 rect(xA,yA,larg,haut);
```

Il existe deux autres modes : CORNERS (on donne les coordonnées de 2 sommets opposés) et RADIUS (on donne en paramètres 3 et 4 la moitié de la largeur et de la hauteur).

3.2.4 L'ellipse

Comme pour le rectangle, il existe plusieurs modes : CENTER (mode par défaut), CORNER (même principe que pour les rectangles), RADIUS et CORNERS.

```
1 // ellipse de centre (50,50) de diamètre horiz 60 et vertical 20
2 ellipse(50,50,60,20);
3 // Mode RADIUS : cercle de centre (100,50) et Rayon=20
4 ellipseMode(RADIUS);
5 ellipse(100,50,20,20);
6 // Mode CORNER : ellipse enfermée dans rectangle de sommet (50,100)
7 // et de dimension larg=30, haut=30 (le rectangle n'est pas tracé)
8 ellipseMode(CORNER);
9 ellipse(50,100,30,50);
```

```
10 // Mode CORNERS : on donne les 2 sommets opposés du rectangle
11 ellipseMode(CORNERS);
12 ellipse(100,100,150,120);
```

3.2.5 Le triangle et le quadrilatère

Le triangle ABC et le quadrilatère ABCD se tracent ainsi :

```
1 triangle(xA,yA,xB,yB,xC,yC);
2 quad(xA,yA,xB,yB,xC,yC,xD,yD);
```

3.2.6 Arc d'ellipse

Un arc d'ellipse de centre $(50, 50)$, de largeur 80 et de hauteur 80 (pour l'ellipse complète) pour un angle entre $\frac{\pi}{2}$ et π (donc ici un quart de cercle inférieur gauche) se trace ainsi :

```
1 arc(50,50,80,80,PI/2,PI);
```

3.2.7 Remplissages et contours

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

```
1 fill(R,G,B,t);
2 // R, G, B quantités de rouge, vert et bleu entre 0 et 255
3 // t : degré de transparence (0:transparent, 255:opaque)
4 fill(N);
5 // N étant entre 0 et 255 (0=noir, 255=blanc)
6 fill(#hhhhhh);
7 // hhhhhh : code hexadécimal de la couleur entre 0 et ffffff
8 noFill();// Pas de remplissage
```

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

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

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

```
1 noStroke();
```

Pour modifier la couleur et l'épaisseur du contour :

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

3.2.8 Portée des modifications de style

À chaque fois qu'on effectue une modification de style (couleurs, contours, ...) cette modification s'applique à toutes les constructions suivantes. Pour modifier temporairement un style, il faut encadrer les commandes par `pushStyle()` et `popStyle()` :

```
size(100,100);
background(255);
// Style initial :
stroke(0);
fill(127);
strokeWeight(1);
rect(10,10,10,10);

pushStyle(); // On ouvre une parenthèse de style
  stroke(255,0,0);// contour rouge
  fill(0,0,255);// remplissage bleu
  strokeWeight(5);// grosses bordures
  rect(30,10,10,10);
popStyle();// Fin de la parenthèse

rect(10,30,10,10);// carré avec les paramètres de style initiaux
```

3.3 Les transformations

Avec les formes de base, il n'est pas possible de dessiner un rectangle « oblique » par exemple. Dans cette partie nous allons voir comment utiliser des transformations géométriques pour y remédier.
Nous avons vu que dans la configuration initiale, le repère a des axes horizontaux et verticaux et une origine dans le coin supérieur gauche de l'écran. C'est en modifiant ce repère que nous allons réussir à effectuer les transformations géométriques.

3.3.1 Déplacer

La modification de l'origine se fait grâce à la commande `translate` qui prend deux arguments : le décalage horizontal puis le décalage vertical.

Exercice 3.1.

On donne le programme suivant :

```
size(200,200);
fill(0);
rect(50,50,50,50);
// On change d'origine :
translate(50,50);
fill(127);
rect(50,50,50,50);
```

Quelle figure obtient-on ?

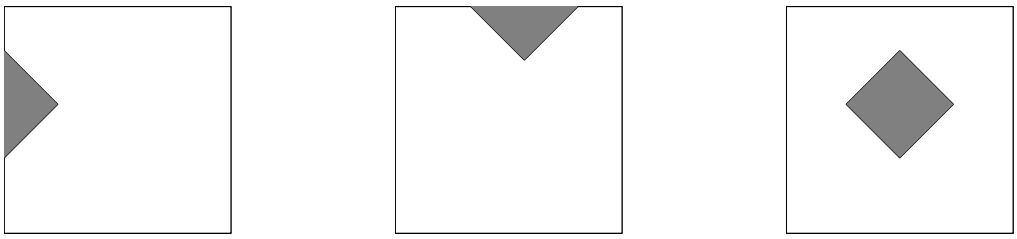
3.3.2 Tourner

De la même façon, il est possible de faire tourner notre repère initial d'un certain angle grâce à la commande `rotate` qui prend un argument : l'angle exprimé en radians. Il existe néanmoins deux commandes permettant de faire les conversions : degrees qui convertit son argument en degrés et radians qui convertit son argument (une mesure en degrés) en radians.
Attention : le sens de rotation est le sens des aiguilles d'un montre (et non pas le sens trigonométrique).

Exercice 3.2.
On donne le programme suivant :

```
1 size(200,200);
2 rotate(PI/4);
3 fill(127);
4 rect(50,50,50,50);
```

Qu'obtient-on à l'écran ?



3.3.3 Mise à l'échelle

La commande permettant d'agrandir ou de réduire une construction est `scale`. Elle prend un ou deux paramètres :

- `scale`(2); agrandira les futures constructions d'un facteur 2 (en maths, on appelle cela une homothétie) ;
- `scale`(0.5,2); divisera par deux en largeur mais multipliera par deux en hauteur (en maths on appelle cela une succession de deux affinités orthogonales).

3.3.4 Provisoire ou définitif ?

De même que pour la gestion des styles (couleurs, ...) il est possible de rendre provisoire un changement de repère, pour cela, on entoure les instructions concernées des instructions `pushMatrix`(); et `popMatrix`();.

```
size(200,200);
//On place l'origine du repère au centre de l'écran :
translate(width/2,height/2);
ellipseMode(RADIUS);
ellipse(0,0,20,20);
// On change provisoirement de repère (décalage vers le haut) :
pushMatrix();
  translate(0,-20);
  rect(-20,0,40,10);
popMatrix();// On revient au repère centré
ellipse(0,60,20,40);
```

3.4 Animations

Le principe d'une animation en Processing est le même que celui d'un dessin animé (un vrai avec un petit carnet dans lequel on fait défiler les pages rapidement). C'est-à-dire qu'on fait un dessin à une position, puis on efface l'écran et on refait le dessin un peu plus loin (ou un peu modifié).
Pour cela, nous allons utiliser une *méthode* dans le programme : la méthode `draw`(). Cette méthode a la particularité de se lancer automatiquement x fois par seconde où x est une valeur qu'on définit grâce à l'instruction `frameRate`(x) (remplacer x par la valeur souhaitée).
Par ailleurs, lorsqu'un programme contient la méthode `draw`(), il est conseillé aussi de créer la méthode `setup`() qui se lance au démarrage du programme pour effectuer par exemple toutes les initialisations.
Enfin, il est possible d'arrêter le rafraîchissement de l'écran grâce à l'instruction `noLoop`(). Dans ce cas, on peut actualiser l'ffichage « manuellement » grâce à l'instruction `redraw`() (valable à partir de la version 3.0 de Processing). Pour relancer le processus de rafraîchissement automatique, on écrira l'instruction `loop`().

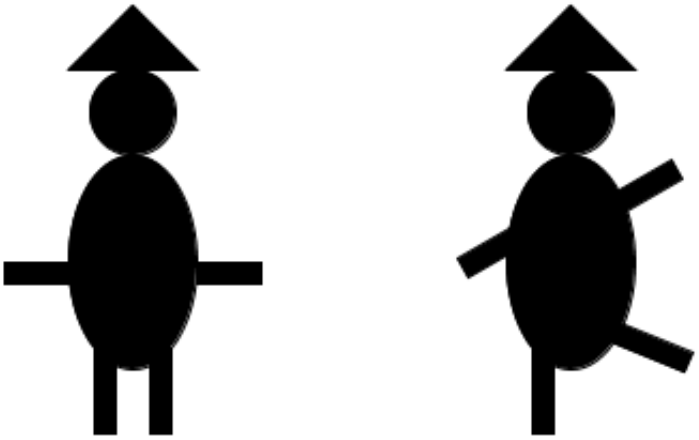
Exemple : une balle qui parcourt l'écran (et puis s'en va...)

```
int x=0;
void setup() {
  size(800,400);// taille fenêtre
  frameRate(50);// Rafraichissement : 50 x / seconde
}
void draw() {
  background(255);// Fond blanc
  fill(0);// Remplissage noir
  ellipseMode(RADIUS);// Mode Radius pour les ellipses
  ellipse(x,200,25,25);
  x++;// Incrémentation de x
}
```

Exercice 3.3.
Écrire le programme qui anime une balle qui tombe (de haut en bas).
Variante : en respectant la loi de gravitation (sans frottement).

3.5 Exercices

Exercice 3.4.
Reproduire les figures ci-dessous (couleurs laissées au choix de chacun) :



Exercice 3.5.
Écrire un programme qui simule les rebonds d'une balle contre un mur puis l'autre (partir de l'exemple de la partie 3.4, mais la balle ne doit plus sortir de l'écran !). On pourra chercher de l'information sur l'instruction `if` que nous verrons ultérieurement.

Exercice 3.6.
Même exercice que le précédent mais avec une balle qui rebondit sur les quatre murs (avec une vitesse initiale « oblique » aléatoire).

Exercice 3.7.
Même exercice que le précédent avec un « trou » dans un mur qui permet à la balle de « s'échapper » et qui affiche « balle perdue » lorsque ceci arrive.

Exercice 3.8.
Expression artistique libre : concours du plus beau dessin en Processing...

TP 4

Instructions de base

4.1 Quelques instructions en Processing

4.1.1 Introduction

Depuis le début de l'année, nos programmes sont constitués d'une suite d'instructions élémentaires qui réalisent chacune une tâche bien définie (un affichage, un changement de couleur, ...). Aujourd'hui, nous allons voir quelques instructions permettant de répéter plusieurs fois une même séquence d'instructions et de tester si une condition est remplie pour réaliser ou non une séquence du programme.

4.1.2 Les boucles

Une boucle « while » :

```
1 while(condition) {
2   instructions;
3 }
```

Une boucle « do ... while » :

```
1 do {
2   instructions;
3 }
4 while(condition);
```

Une boucle « for » :

```
1 for(initialisation; test; incrémentation) {
2   instructions;
3 }
```

Exemple : `for(int i=0;i<10;i++)`

Exercice 4.1.
Expliquer en quelques mots les différences entre une boucle `for` et une boucle `while` ou `do` ... `while`.
Expliquer ensuite la différence entre une boucle `while` et une boucle `do` ... `while`

4.1.3 Instructions conditionnelles

Un test « if » :

```
1 if (condition) {
2   instructions si vrai;
3 }
4 else {
5   instructions si faux;
6 }
```

Le `else` est facultatif et on peut éventuellement le remplacer par un elseif (conditions) { voire plusieurs...

Un « branchement conditionnel » :

```
1 switch(expression entière) {
2   case cas1 : instruction1; break;
3   case cas2 : instruction2; break;
4   ...
5   default : instruction;
6 }
```

Le cas `default` est facultatif.

4.1.4 Nombres aléatoires en Processing

Dans les exercices proposés ensuite, il pourra être utile d'obtenir un nombre aléatoire. Voici les différentes manières d'en obtenir :
`random`(50) renvoie un nombre de type `float` compris entre 0 (inclus) et 50 (exclu).
`random`(10,20) renvoie un nombre de type `float` compris entre 10 (inclus) et 20 (exclu).
(`int`)`random`(1,7) renvoie un entier entre 1 et 6 (inclus).

4.2 Exercices

4.2.1 Avec des nombres...

Exercice 4.2.
Dans chaque cas écrire un programme qui :

1. teste la parité d'un entier saisi par l'utilisateur (`a % b` teste la divisibilité de `a` par `b`) ;
2. écrit la liste des diviseurs d'un entier non nul saisi par l'utilisateur ;
3. demande le nombre n de notes, demande les n notes et calcule la moyenne ;

4. demande des notes et s'arrête à la première note négative puis calcule la moyenne des notes positives ou nulles.

Exercice 4.3.
Écrire l'algorithme (puis le programmer avec Processing) qui permet de calculer les termes de la suite de Syracuse jusqu'à obtenir un terme égal à 1. Afficher alors le nombre de termes calculés.

Exercice 4.4 (Cluedo)**.**
L'exercice consiste à poser des questions à l'utilisateur qui pense à un personnage. Le programme doit deviner à qui l'utilisateur pense grâce aux réponses aux questions posées. L'utilisateur ne peut répondre aux questions posées que par oui ou par non.
Les personnages sont : Mlle Rose, le Professeur Violet, le Colonel Moutarde, le Révérend Olive et Me Leblanc.
Seul le Colonel Moutarde a des moustaches, tous portent des lunettes sauf Mlle Rose, et le Professeur Violet est le seul à avoir un chapeau.
Récupérer le fichier `Cluedo.pde` et compléter le programme de deux façons différentes.

1. D'abord, en laissant le programme poser toutes les questions et ensuite les traiter entre les deux zones de commentaires.
2. Ensuite en modifiant le programme (y compris en dehors des deux zones de commentaires) pour ne poser que trois questions au maximum.

4.2.2 Avec des textes

Exercice 4.5.
Écrire un programme qui demande une chaîne de caractères et la ré-écrit en passant à la ligne après chaque caractère.

Exercice 4.6.
Écrire un programme qui demande une chaîne de caractères et la ré-écrit en transformant les majuscules en minuscules et réciproquement.

Exercice 4.7.
Écrire un programme qui demande un mot de passe contenant au moins une majuscule, une minuscule et un chiffre et teste si c'est bien le cas.

Exercice 4.8.
Écrire un programme qui demande une chaîne de caractères la transforme en majuscules, remplace les lettres accentuées par leurs homologues sans accent et supprime tous les caractères non alphabétiques.

Exercice 4.9.
écrire un programme qui demande une chaîne de caractères et compte :
— le nombre d'espaces ;
— le nombre d'occurences d'une lettre à choisir ;

— le nombre de voyelles.

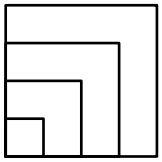
Exercice 4.10.
Cryptographie...

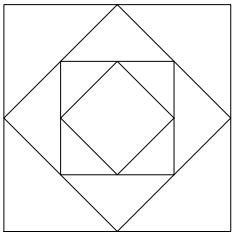
1. Écrire un programme qui permet de crypter/décrypter avec le chiffre de César.
2. Même question avec un codage affine.

4.2.3 Avec des graphiques

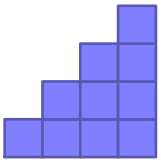
Les exercices suivants sont à réaliser en un minimum de lignes de code (hors lignes de commentaires indispensables...) en utilisant des boucles.

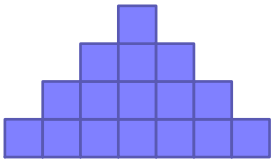
Exercice 4.11.
Réaliser les figures suivantes à n carrés :



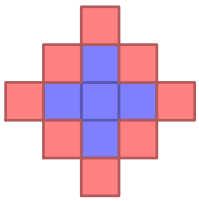


Exercice 4.12.
Réaliser les escaliers suivants à n marches :





Exercice 4.13.
Réaliser la figure suivante où la ligne la plus grande contient $2n+1$ carrés (avec n est un entier naturel) :



TP 5

HTML et CSS

Le langage html est le langage de base de toutes les pages web. HTML est l'acronyme de Hyper Text Markup Language c'est-à-dire langage de marquage hypertexte. Ce langage a déjà connu plusieurs versions (l'actuelle est la version html 5) et dans ce document, nous ne verrons que des bases valables pour toutes les versions. Il a été inventé en 1990 par Tim Berners-Lee lorsqu'il travaillait au CERN. Son objectif était d'afficher des pages d'information ayant les deux propriétés suivantes :

- les pages étaient reliées entre elles par des liens hypertextes (les liens sur lesquel on clique) ;
- les pages devaient être lisibles sur tous les ordinateurs.

5.1 Principes de base

L'objectif d'une page web est qu'elle soit visitée (et lue) par un maximum de personnes, pour cela, elle doit être claire, accessible à tous et (si possible) esthétique. Le premier conseil est de faire dans la sobriété !

5.1.1 Éléments fondamentaux

Concrétement, une page html est un fichier texte dont le suffixe est `.html` ou `.htm` ; pour l'écrire nous allons utiliser un éditeur de texte (Nous utiliserons Notepad++ sur les ordinateurs du lycée).
Une fois le site mis en ligne, il est vivement souhaitable qu'une des pages web se nomme `index.html` ainsi, l'utilisateur qui se connecte à votre site `http://adressedemonsite` tombera directement sur cette page.
Un fichier html doit commencer[1] par `<html>` et se terminer par `</html>`. Un exemple de page web minimale serait :

```
1 <html>
2 Ceci est ma (super) page web
3 </html>
```

1. En fait, pas tout-à-fait, nous verrons dans quelques lignes qu'on va commencer par une autre balise...

Les instructions <html> et </html> sont appelées des *balises* (« entrante » et « fermante » respectivement).
Nous allons détailler quelques autres balises qui ne changeront pas fondamentalement cette page web mais qui ont néanmoins de l'importance pour la suite.
Pour indiquer au navigateur le langage employé il est préférable de commencer le fichier par la balise <!DOCTYPE html>, ensuite, on ouvre la balise <html>. Le fichier sera décomposé en deux : l'entête (balise <head>) et le corps (balise <body>).
Dans l'entête nous fournirons des informations qui ne s'affichent pas directement sur la page :

- la balise <meta charset="utf-8"/> permet d'indiquer l'encodage utilisé dans le fichier (pour que les accents et caractères spéciaux s'affichent correctement) ;
- la balise <title>Mon titre </title> permet de donner un titre qui s'affiche dans la barre supérieure du navigateur.

Finalement, notre page web est devenue :

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="iso-latin1"/>
5      <title>Ma première page web</title>
6    </head>
7    <body>
8      Ceci est ma (super) page web.
9    </body>
10 </html>
```

5.1.2 Premières balises

Comme en Processing, il est indispensable de mettre des commentaires dans le fichier html :

```
1  <!-- Ceci est un commentaire qui ne sera pas interprété par le
       navigateur -->
```

Dans le tableau ci-dessous, quelques balises avec l'effet attendu...

Balise	Effet
texte	met `texte` en caractères gras
texte	met `texte` en caractères italiques
<u>texte</u>	souligne `texte`
^{texte}	met `texte` en exposant
_{texte}	met `texte` en indice
texte	met `texte` dans un nouveau paragraphe
texte	centre `texte` (peut être utilisé avec `left` et `right`)
 	passe à la ligne (pas besoin de « fermante »)
<hr width = 50%/>	trace une ligne horizontale de largeur la moitié de la page (pas besoin de « fermante »)
<code>texte</code>	pour mettre en forme du code informatique

5.1.3 Les listes

Une énumération de plusieurs éléments gagne à être mise en avant grâce à des puces ou des numéros, pour cela il existe les balises <ul> (*unordered* list) et <ol> (*ordered* list) :

```
<p> Conseils importants :
<ul>
  <li> écouter le prof;</li>
  <li> apprendre.</li>
</ul></p>
```

Conseils importants :

- écouter le prof;
- apprendre.

```
<p> Conseils importants :
<ol>
  <li> écouter le prof;</li>
  <li> apprendre.</li>
</ol></p>
```

Conseils importants :

1. écouter le prof;
2. apprendre.

5.1.4 Titres et paragraphes

Nous avons vu que chaque paragraphe doit être encadré par et , le langage html permet aussi de hiérarchiser la page à l'aide de titres :

```
1 <h1>Ceci est gros titre (on dit
      de niveau 1)</h1>
2   <h2>Ceci est un moins gros
      titre (de niveau 2)</h2>
3   <h2>Un deuxième moins gros</
      h2>
4 <h1>Encore un gros</h1>
5   <p>Nous allons détailler ici
      le deuxième gros titre</p>
```

Ceci est gros titre (on dit de niveau 1)

Ceci est un moins gros titre (de niveau 2)

Un deuxième moins gros

Encore un gros

Nous allons détailler ici le deuxième gros titre

La mise en forme des titres et des listes (mais aussi de bien d'autres choses) peut être modifiée grâce à une feuille de style qu'on appelle fichier CSS, nous verrons celà un peu plus loin. À noter que grâce à une feuille de style on peut aussi modifier le comportement de certaines balises comme ou .

5.2 Un peu plus loin

5.2.1 Les liens

Nous avons vu que l'essence même du html est le côté « hypertexte » c'est-à-dire la possibilité de créer des liens (à cliquer) entre plusieurs pages d'un même site ou non. Nous allons détailler

leur utilisation ici.
La balise qui permet de créer un lien est `<a ...>... </a>`, nous allons voir comment compléter les ....
Quelques exemples pour comprendre :

```
<a href="http://reymarlioz.free.fr">Un super site de maths</a> :
   à voir absolument !
<a href="page_processing.html">Mes DM en Processing</a>
<a href="downloads/liste.html">Les téléchargements</a>
```

Le premier lien permet de renvoyer à une adresse « externe » : à noter que seule la partie « Un super site de maths » sera cliquable (pas le « : à voir ... »).
Le deuxième lien ouvre la page `page_processing.html` qui doit être placée dans le même répertoire que la page courante.
Le troisième lien ouvre la page `liste.html` située dans un sous-répertoire `downloads` du répertoire courant.
Pour des liens qui pointe vers votre site il est indispensable de les écrire avec des adresses *relatives* (comme les deuxième et troisième exemples ci-dessus) et non pas *directes* (comme par exemple `adresse.de.monsite/downloads/liste.html`) car sinon, en cas de changement d'hébergeur de site, il vous faudra modifier tous vos liens !

Il peut parfois être utile de créer un lien vers une adresse mail :

```
<a href="mailto:contact@monsite.fr?subject=à propos de ton site"
       title="M'envoyer un courriel"> (pour me contacter)
</a>
```

Un clic sur un tel lien ouvre le logiciel de messagerie par défaut avec un nouveau message dans lequel figure déjà l'adresse du destinataire
Remarque : attention, mettre un tel lien sur un site en ligne est la porte ouverte à la réception de spam (si votre site est beaucoup visité, les robots publicitaires relèveront votre adresse...) ; c'est donc fortement déconseillé (il existe des scripts Javascript pour y remédier).

Si la page de votre projet est longue, il peut être intéressant de réaliser au début une table des matières « cliquable ». Ainsi votre page `projet.html` pourra commencer par :

```
<a href="#description">La description du projet</a>
...
<a href="#code">Le code</a>
<a href="credits">Les participants au projet</a>
...
<h1 id="description"> Description de mon projet</h1>
...
<h1 id="code"> Quelques éléments du programme</h1>
...
<a id="credits">Je remercie ici ...</a>
```

5.2.2 Les couleurs, les polices, ...

Dans les balises qui définissent un titre, un paragraphe, ..., on peut ajouter des attributs divers de couleurs, polices, taille, ... Nous allons en voir quelques uns ici sur l'exemple commenté ci-dessous (il est néanmoins préférable de gérer ceci dans une feuille de style – voir plus loin) :

```
1 <!-- toute la page sera sur fond argenté écriture bleue les liens
      en jaunes et les liens cliqués en marine-->
2 <body bgcolor=silver text=blue link=yellow vlink=navy>
3
4 <font color=red>Ce texte est en rouge</font>
5 <table border bordercolor=blue bgcolor=green> <!-- tableau avec
     des cadres bleu et fond vert -->
6 ...
7 </table>
```

5.2.3 Les images

Pour insérer une image dans une page web, il est recommandé d'utiliser un format compressé (jpeg, gif, png). Nous reviendrons plus tard dans l'année sur les formats d'images...
En récupérant le logo que vous connaissez bien sur le site du lycée et en l'enregistrant sous le nom `lyceemarlioz.jpg` dans un sous répertoire `images` de mon site, je peux désormais écrire :

```
1 <img src="images/lyceemarlioz.jpg"
2      alt="le logo du lycée Marlioz"
3      title="texte de survol"
4      hspace=20 vspace=20
5      align=left/>
```

Le contenu de la balise `alt` sera affiché au cas où le navigateur ne peut pas charger l'image et le contenu de la balise `title` s'affiche lorsque la souris survole l'image. Un espace de 20 pixels est réservé entre l'image et le texte (horizontalement et verticalement) et l'image est alignée à gauche.
Remarque : la balise `<img ... />` ne nécessite pas de balise « fermante ».
Il est possible de redimensionner une image en utilisant les attributs `width` (largeur) et `height` (hauteur). Il est prudent de ne redimensionner qu'une seule de ces deux dimensions (l'autre l'est alors aussi proportionnellement).
Enfin, pour mettre une image en fond de page, on modifie la balise `<body>` ainsi :

```
1 <body background="images/monfond.jpg">
```

5.2.4 Tableaux

Un exemple commenté de tableau à modifier pour comprendre comment ça marche.

```
1  <table border="1"><!-- définition
       du tableau -->
2    <tr><!-- nouvelle ligne -->
3      <th><!-- colonne entête -->
4        Matière</th>
5      <th>Coefficient</th>
6    </tr> <!-- fin de ligne -->
7    <tr>
8      <td>Maths</td><!-- colonne
         donnée -->
9      <td> 7 </td>
10   </tr>
11   <tr>
12     <td> ISN</td>
13     <td> 2 </td>
14   </tr>
```

Matière	Coefficient
Maths	7
ISN	2

Résultat : pas terrible ! À vous d'améliorer tout ça...

Remarque : les balises `<td colspan="2">` et `<td rowspan="3">` permettent de fusionner des cellules en colonnes et lignes respectivement.

5.3 Mise en ligne

Une fois votre site créé (c'est-à-dire les fichiers html écrits), il va falloir les mettre en ligne pour que tous les internautes du monde entier puissent en profiter !

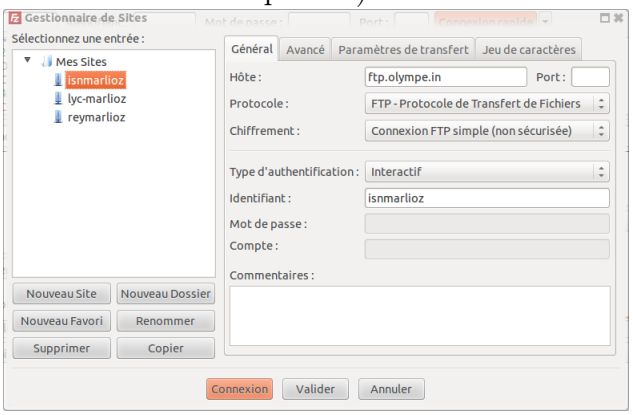
5.3.1 FileZilla

La première chose est de télécharger et installer FileZilla. Une fois lancé, cliquer sur fichier et Gestionnaire de site. Cliquer ensuite sur Nouveau site et compléter ensuite la fenêtre comme ci-contre :

Hôte : `ftp.hebergeur.fr` Port : `21` (à modifier en fonction de votre hébergeur !)

Type d'authentification `Interactif` et compléter l'identifiant par votre celui que vous a fourni votre hébérgeur ; idem pour le mot de passe. Votre mot de passe vous sera demandé une deuxième fois.

Exemple (désuet : cet hébergeur n'existe plus...)



Dans la colonne `Site local`, sélectionner le répertoire où est stocké votre site sur votre ordinateur, puis, faire glisser à la souris les fichiers à envoyer (au minimum `index.html` et les images nécessaires) dans la colonne `Site distant`.

5.3.2 Référencement

Pour que votre site apparaisse en bonne place dans une recherche de moteur de recherche lorsqu'un internaute fait une recherche sur « isn marlioz processing » par exemple, il faut d'abord générer du trafic sur votre page en plaçant des liens partout où vous le pouvez (après la signature de vos mails, sur votre page facebook, j'en mettrai un sur la page du lycée, ...) et pour que les moteurs de recherche vous « indexent » c'est-à-dire qu'ils sachent de quoi parle votre site, nous allons placer des mots-clés.
Dans l'entête (entre les balises `<head>` et `</head>`) nous allons placer des « meta-informations » :

```
<head>
  <meta charset="iso-latin1"/>
  <title>Ma première page web</title>
  <meta name="description" content="contenu des cours d'ISN au
     lycée Marlioz">
  <meta name="keywords" content="rey isn marlioz processing html
     cours">
  <meta name="nameAutor" content="Thomas REY">
  <meta http-equiv="date" content="22.11.2016">
</head>
```

5.4 Compléments

5.4.1 Feuilles de style

Puisqu'il me reste du temps et de la place, quelques rudiments de CSS...
L'ensemble des règles qui régissent l'affichage de vos pages web est appelé *feuille de style*. Il s'agit d'un ou de plusieurs fichiers `.css`. Une feuille de style est constituée de plusieurs *règles de style* toutes construites sur le modèle suivant :

- un sélecteur (c'est-à-dire la balise concernée par cette règle) ;
- une déclaration de style sous la forme d'un bloc d'ensemble `propriété: valeur;`.

```
/* Commentaire : définition du style des titres de niveau 3 */
h3 { font-style: italic;
     font-family: Arial, sans-serif;
   }
```

L'ensemble des règles de style est enregistré dans un fichier `style.css` (ou un autre nom d'ailleurs) et est appelé dans l'entête de chaque page html ainsi :

```
<head>
  <link rel="stylesheet" href="style.css" />
</head>
```

Si vous voulez par exemple écrire en rouge certains paragraphes mais que vous n'êtes pas sûr de la couleur (ou que vous souhaitez pouvoir modifier la couleur de *tous* ces paragraphes ultérieurement), il faut procéder en deux étapes :

— dans vos fichiers html, on définit une *classe* de paragraphe ainsi : `<p class="format1" >.... </p>` (si possible avec un nom plus parlant que « `format1` ») ;
— dans votre feuille de style css, on crée le style `format1` ainsi :

```
1 p.format1 {color: red;}
```

Si vous préférez par la suite que ces paragraphes soient centrés plutôt qu'en rouge vous avez juste à modifier votre feuille de style ainsi :

```
1 p.format1 {text-align: center;}
```

Un exemple :

```
1 /* Définition des titres */
2 h1 { color: red; font-size: 150%; background-color: silver; }
3 h2 { color: blue; font-size: 125%; background-color: silver; }
4 /* les balises ciblées avec un class="titre" seront encadrées */
5 .titre { border: solid; }
6 /* re-définition du strong en bleu sur fond rouge */
7 strong { color: blue; background-color: rouge; }
```

Quelques propriétés CSS sous forme d'exemples :

```
1  /* changement de police pour tous les paragraphes */
2  p {font-family: Arial, Verdana, sans-serif;} /* plusieurs choix
       au cas où... */
3  p {font-family: "Courier New", monospace;} /* pas fixe : pour le
       code */
4  /* taille */
5  p {font-size: 150%;}
6  p {font-size: 15px;}
7  /* couleurs */
8  p {color: #0000ff;} /* codage hexa */
9  p {font-weight: bold;}
10 p {font-weight: normal;}
11 p {font-style: italic;}
12 /* texte souligné + barre au dessus + barré + clignotant (ouh c'
       est moche !)*/
13 p {text-decoration: underline overline line-through blink}
14 /* surlignage :*/
15 p {background-color: red;}
16 p {text-align: left;} /* ou right, center, justify
17 /* curseur de souris : */
18 .aide {cursor: help} /* ou default, move, wait, pointer, ... */
```

Il existe encore beaucoup d'autres propriétés, à vous de trouver de la documentation en ligne ou dans la référence bibliographique cité en fin de ce document...

5.4.2 Propriété intectuelle ...

Attention : lorsque vous récupérez des logos, images, textes, animations, ... il faut toujours vérifier que ces images sont libres de droit et consulter la licence sous laquelle ils sont publiés. Par exemple tous les documents du site http://reymarlioz.free.fr sont sous licence Creative Commons NC-BY-SA (utilisations libre sus les trois réserves suivantes : pas d'utilisation commerciale, auteur à citer en cas de réutilisation, et diffusion des modifications sous la même lience).

5.4.3 Bibliographie

Pour ce cours, je me suis largement inspiré du livre suivant : « Premiers pas en CSS 3 & HTML5 »– Francis Daillard – Eyrolles.

5.5 Au travail !

Votre objectif pour cette séance et jusqu'à la fin de l'année est de créer un site web (par bi/trinome) qui respectera les impératifs suivants :

- les pages doivent être écrites par vous (sans l'aide de logiciels de conception de site web type Dreamweaver, GoogleSite, ...) ;
- elles doivent m'être rendues sous forme d'une archive zip (avant les vacances de Noël) que je mettrai en ligne ici : http://isnmarlioz.webou.net ou que vous pouvez mettre en ligne vous même si vous avez un hébérgeur ;
- elles doivent comporter des images, des liens, ...
- elles doivent être les plus lisibles et homogènes possibles (ne pas changer de police, de taille de caractères, de fonds, ... à chaque page !) ;
- elles doivent utiliser une feuille de style `CSS` écrite par vous également ;
- pour le contenu, une page d'accueil, une page de présentation de vous, une page pour les DM en Processing, une page pour votre exposé et prévoir une page pour votre projet (et plus si affinité avec le html !).

Pour vous aider, je vous mettrai sur Pronote un exemple de zip attendu. Ce travail sera noté pour le second trimestre.

TP 6

Tableaux - Lecture de fichier

6.1 À retenir

6.1.1 Déclarer un tableau

Pour déclarer un tableau de 16 entiers de type `int` on peut utiliser le modèle suivant :
`int montableau[] = new int[16];`
Pour définir un tableau à deux dimensions :
`int tabmult[][] = new int[4][3];`

6.1.2 Taille d'un tableau

Pour déterminer la taille d'un tableau unidimensionnel, on utilise :
`int longueur = montableau.length;`
Pour (re)trouver la première longueur d'un tableau bidimensionnel :
`int longueur1 = tabmult.length;`
Et pour l'autre longueur (3 dans l'exemple précédent) :
`int longueur2 = tabmult[0].length;`

6.1.3 Accéder aux éléments

L'élément d'indice `i` du tableau `monTableau` est obtenu ainsi : `monTableau[i]`. On peut s'en servir pour lire une valeur ou pour affecter une valeur au tableau :

```
1 int monTableau = new int[3];
2 monTableau[0] = 0; monTableau[1] = 5; monTableau[2] = 11;
3 println("la valeur d'indice 1 est : " + monTableau[1]);
```

6.1.4 Lecture d'un fichier

Pour remplir un tableau avec des éléments, il peut être utile de lire ces éléments dans un fichier texte obtenu par un autre moyen que la saisie manuelle (capture de données, internet, ...)

Exemple 6.1.
Le code suivant résume ce que vous devez connaître pour réaliser les exercices de ce TP (à charger sur http://reymarlioz.free.fr avec un exemple de fichier map) :

```
1  String[] monTexte; // tableau qui contiendra les lignes du
       fichier texte
2  int niveau = 0;
3  int nbLignesMap = 0;
4  void setup() {
5    size(500,300);
6    fill(0);
7    textSize(20);
8    noLoop();
9    rectMode(CORNER);
10   selectInput("choisir le fichier texte ", "choixFait");
11 }
12
13 void draw() {
14   background(200);
15   text("Niveau " + niveau, 10,20);
16   for(int l=1; l<=nbLignesMap; ++l) {
17     for(int c = 0; c<monTexte[l].length(); ++c) {
18       if (monTexte[l].charAt(c)=='X') {
19         rect(20*c, 20*l+50,20,20);
20       }
21     }
22   }
23   noLoop();
24 }
25
26 void choixFait(File fichierChoisi) {
27   if (fichierChoisi == null) {//pas de fichier
28     println("Erreur : pas de fichier ou illisible");
29   } else {
30     monTexte = loadStrings(fichierChoisi.getAbsoluteFile());
31     niveau = parseInt(monTexte[0]);
32     println(niveau);
33     nbLignesMap = monTexte.length - 1;// à cause de la 1ère ligne
34     loop();// on lance la méthode draw
35   }
36 }
```

6.2 Exercices

Exercice 6.1.

1. Écrire un algorithme qui réalise les opérations suivantes :
   — demander le nom et le prénom de cinq personnes ;
   — puis, pour chaque personne, pose une question du style « Bonjour <Prénom> <NOM>, quelle est ta couleur préférée [1] » ;
   — enfin, écrit à l'écran sous la forme <NOM - prénom - couleur préférée> la liste ainsi obtenue (passage de ligne entre chaque individu mais pas entre le NOM, le prénom et la couleur).
2. Modifier l'algorithme précédent pour qu'il demande au début le nombre de personnes.
3. Modifier l'algorithme de la question 1 pour que la saisie s'arrête dès que l'utilisateur saisit FIN à la place du nom.
4. Même question mais l'utilisateur peut saisir FIN ou fin ou Fin ou FiN ou . . . pour terminer la saisie.
5. Écrire le programme en Processing.

Exercice 6.2.

La suite de Fibonacci est définie ainsi : $\begin{cases} u_0 = 1; u_1 = 1 \\ u_{n+2} = u_n + u_{n+1}, n \in \mathbf{N} \end{cases}$

1. Écrire un algorithme qui demande le nombre de termes à afficher, calcule ces termes, puis les affiche séparés par des « ; ».
2. Programmer cet algorithme en Processing.
3. Modifier le programme précédent pour qu'il affiche le nombre de termes calculés et le plus grand d'entre eux.

Exercice 6.3.

Écrire un algorithme qui calcule dans un tableau à double entrées la table de multiplication pour les entiers de 0 à 12.
Programmer cet algorithme et afficher cette table (attention à respecter l'alignement des colonnes !).

Exercice 6.4.

Écrire un programme qui écrit à l'écran le triangle de Pascal de degré n fixé au départ (où chaque ligne commence et termine par un « 1 », et chaque élément est la somme de l'élément supérieur et de l'élément supérieur gauche) :

n \ k	0	1	2	3	4	5
0	1					
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	4	6	4	1	
5	1	5	10	10	5	1

1. Le rédacteur de cet exercice manque cruellement d'inspiration ce matin. . .

Exercice 6.5.
Écrire un programme qui stocke dans un tableau les n premiers nombres premiers et les affiche.
Aides mathématiques :
- un nombre supérieur à 1 est premier s'il n'est divisible que par 1 et par lui-même ;
- on montre sans difficultés qu'un nombre n est premier s'il n'admet aucun diviseur premier inférieur ou égal à $\sqrt{n}$.

Exercice 6.6.
Dans cet exercice, vous allez utiliser le programme de l'exemple 6.1.

1. Modifier ce programme pour qu'il ignore toute les lignes commençant par le symbole `#`. Modifier le fichier `map.txt` pour tester.
2. Modifier ce programme pour qu'il affiche un rond bleu lorsqu'il rencontre le caractère `'J'` (joueur) et un carré rouge lorsqu'il rencontre le caractère `'E'` (ennemi). Modifier le fichier `map.txt` pour ajouter ces caractères.
3. Même question mais avec des dessins plus élaborés pour le joueur et les ennemis. Vous pouvez aussi ajouter des pastilles bonus, ...
4. Terminer ce jeu de PacMan[2]...

Exercice 6.7.
Dans cet exercice, nous allons étudier un fichier ADN (à charger à l'adresse habituelle pour tester votre programme).

1. Écrire un programme qui lit un fichier texte contenant des séquence de `A`, `T`, `G` et `C` et qui affiche la séquence ADN sous la forme d'une suite de « traits » en utilisant une couleur par base.
2. Modifier ce programme pour afficher à l'écran la position dans la séquence de la première occurence du « mot » : `AGCT`.
3. Modifier le programme précédent pour écrire dans un fichier (même nom que le fichier lu avec l'ajout des caractères `'-M'` à la fin (mais avant le `.txt`)) la séquence en ramplaçant chaque `'T'` par un `'A'` (et réciproquement) et remplacer chaque `'C'` par un `'G'` (et réciproquement).
4. Modifier le programme de la question 2 pour écrire dans un fichier (nommé `genes.txt`) toutes les séquences (une par ligne) situées entre les mots `AGCT` et `TCGA` et en ignorant les autres morceaux du fichier ADN.

2. Belle idée de projet non ?