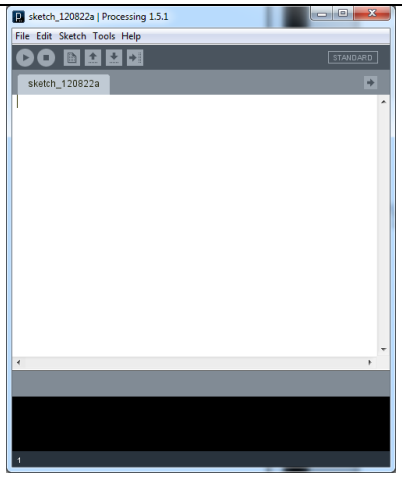
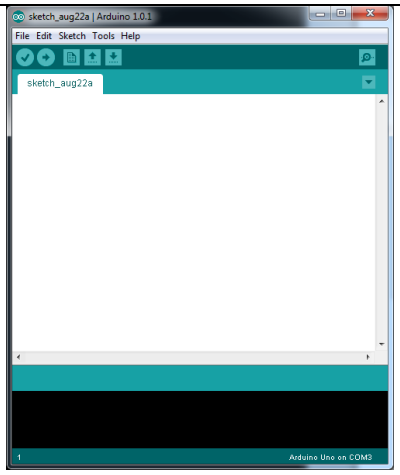
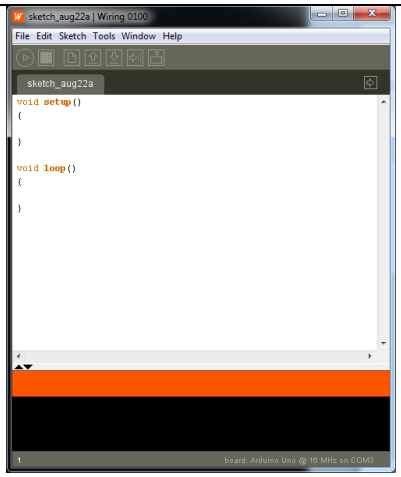
Nom et prénom :

Processing 1.5 – Initiation

Sti2d SIN

Objectif : Apprendre un langage de programmation utilisable sur ordinateur et microcontrôleur. Processing utilise le langage Java proche du langage C utilisé pour programmer les microcontrôleurs.
L'environnement de **Processing** est similaire aux environnements **Arduino** (pour platines Arduino) et **Wiring** (pour microcontrôleurs Atmel AVR, Microchip PIC …)
De plus, ce langage orienté multimédia est particulièrement adapté à la filière STI2D-SIN

		
Processing PC	Arduino Platines Arduino	Wiring µC Atmel AVR, Microchip PIC

processing.pdf - Extrait page 4
« Conçu par des artistes, pour des artistes, Processing est un des principaux environnements de création utilisant le code informatique pour générer des œuvres multimédias sur ordinateur. L'attrait de ce logiciel réside dans sa simplicité d'utilisation et dans la diversité de ses applications : image, son, applications sur Internet et sur téléphones mobiles, conception d'objets électroniques interactifs. »

processing.pdf - Extrait page 7
« Logiciel de création multimédia, Processing permet de dessiner, réaliser des animations en 2 ou 3 dimensions, créer des œuvres sonores et visuelles, concevoir des objets électroniques qui interagissent avec leur environnement. Des dizaines de milliers d'artistes, de designers, d'architectes, de chercheurs et même d'entreprises l'utilisent pour réaliser des projets incroyables dans de multiples domaines :
***publicités, génériques de films, vidéos clips, dessins animés**, Processing permettant de créer des effets visuels originaux ;*
***visualisation de données scientifiques** sous la forme d'images fixes ou animées, facilitant ainsi la représentation d'informations complexes dans de multiples secteurs professionnels (environnement, transports, économie, sciences humaines, etc.) ;*
***production musicale**, Processing permettant non seulement de lire mais aussi de transformer et de créer du son ; »*

Documents ressources :

- En français
  processing.pdf (192 pages) Disque dur ou
  http://fr.flossmanuals.net/_booki/processing/processing.pdf

- tuto_pdf.pdf Le tutoriel du Site du zéro

- En anglais
  Site Processing http://processing.org/learning/

- **XnView** pour modifier les dimensions d'une image

Ce document est à utiliser avec le fichier **processing.pdf.** Les numéros de chapitres (Chx) et les références à une page (Py) concernent le document **processing.pdf**. Nous n'utiliserons qu'une partie du document.

1. PRESENTATION Chapitres 1 à 4

2 DESSINER Chapitres 5 à 8 et DESSINER PLUS Chapitres 9 à 12)

3 PROGRAMMER Chapitres 13 à 19

1.1 INTRODUCTION (Ch1 P4)

Lisez l'introduction pages 4 et 5

1.2 EXEMPLES D'UTILISATION (Ch2 P7)

Pour information, lisez les exemples d'utilisation pages 7 et 8.

1.3 L'INSTALLATION DE PROCESSING (Ch3 P15)

Site de téléchargement http://processing.org/download/

Il suffit de décompresser les fichiers puis de cliquer sur le fichier exécutable « processing.exe »

1.4 LES BASES DE PROCESSING (Ch4 P19)

Lisez le chapitre 4 et répondez aux questions

Dans Processing, comment appelle-t-on un programme :

Sur quel langage est basé Processing :

Ou sera affiché le sketch que vous allez créer (fenêtre, zone) :

L'INTERFACE (P19)

File > Preferences

Quelle information est indiquée sous Sketchbook location :

BASES DU LANGAGE (P21)

Pour un logiciel, que signifie « Etre sensible à la casse » :

Processing est-il sensible à la casse :

Comment appelle-t-on les fonctionnalités prédéfinies :

A quoi sert la fenêtre console (Zone 4) :

Opérations arithmétiques (P22)

`// programme 1`
`int i;`
`i=5;`
`println(i);`
`i=i+1;`
`println(i);`
`i++;`
`println(i);`
	`// programme 2`
	`int x;`
	`x=50;`
	`println(x);`
	`x=x+10;`
	`println(x);`
	`x+=10;`
	`println(x);`

Exécutez le programme 1 ci-dessus pour vérifier l'équivalence de « `i=i+1` » et « `i++` »

Exécutez le programme 2 ci-dessus pour vérifier l'équivalence de
« `x=x+10` » et « `x+=10` »

2 DESSINER Chapitres 5 à 8 et DESSINER PLUS Chapitres 9 à 12)

2.1 L'ESPACE DE DESSIN (Ch5 P25)

Lisez le chapitre 5 **L'ESPACE DE DESSIN** et exécutez les exemples

Y a-t-il une différence entre les deux instructions ci-dessous

size ();	size (100,100);

A quel coin de l'espace à 2 dimensions (2D) correspondent les valeurs x=0 et y=0.

2.2 LES FORMES (Ch6 P27)

Lisez le chapitre 6 en exécutant les exemples proposés.
Complétez toutes les cases vides du tableau ci-dessous. Utilisez les exemples du document.

Forme	**Instruction**	**Exemple**	**Commentaire** (Facultatif)
LE POINT	point(x,y)	point(50, 50);	Sans commentaire
LA LIGNE	line(xA,yA,xB,yB)	line(15, 90, 95, 10);	xA, yA : Point A xB, yB : Point B
LE RECTANGLE			Sans commentaire
L'ELLIPSE	ellipse(xC,yC,dX,dY);		
LE TRIANGLE			
L'ARC			
LE QUADRILATÈRE			
COURBE			Voir document
COURBE BÉZIER			Voir document
COURBE LISSÉE	Voir document	Voir document	Voir document
FORMES LIBRES	Voir document	Voir document	Voir document
CONTOURS			
REMPLISSAGE			
PRIMITIVES 3D	Voir document	Voir document	Voir document

Pour information (Niveau post-bac)
Les courbes de Bézier sont des courbes de degré 3. Elles sont donc déterminées par quatre points de contrôle.

2.3 LES COULEURS (Ch7 P32)

Expliquez brièvement la méthode utilisée pour obtenir une couleur ?

Pour en savoir plus sur le modèle RVB et les couleurs secondaires cyan, magenta et jaune :
http://www.profil-couleur.com/ec/106-modele-rvb.php
http://fr.wikipedia.org/wiki/Couleur_secondaire

Complétez le tableau ci-dessous – RVB, valeur de 0 à 255 – Vérifiez les valeurs avec l'instruction background

Couleur	R (Rouge)	V (Vert)	B (Bleu)
Rouge	255	0	0
Vert			
Bleu			
Jaune			
	255	0	255
	0	255	255
	0	0	0
	64	64	64
	127	127	127
	191	191	191
	255	255	255

Complétez toutes les cases vides du tableau ci-dessous au fur et à mesure que vous répondez aux questions. Utilisez les exemples du document.

Forme	Méthode (*)	Exemple	Commentaire (Facultatif)
COULEUR DE FOND	background(R, V, B)	background(0, 0, 0);	R, V, B : 0 à 255
COULEUR DE REMPLISSAGE	fill(R, V, B)	fill(255, 204, 51);	
Idem ci-dessus	fill(C);	fill(127);	
COULEUR DU CONTOUR activée	stroke(R,V,B)	stroke(255, 0, 0);	
	noStroke()	noStroke()	Pas de contour

(*)Une méthode est un ensemble d'instructions regroupées dans un même bloc de code et accessibles par un simple mot clé. (Page 170)

Expliquez pourquoi **`fill(255, 204, 51);`** est équivalent à **`fill(#ffcc33);`**

```
noStroke();
fill(255, 204, 51); // orange
rect(25, 25, 50, 50);
fill(255, 255, 255, 127); // blanc semi-transparent
rect(35, 35, 50, 50);
```

Dans le programme ci-dessus on modifie la ligne suivante
`fill(255, 255, 255, 127);`

Qu'obtient-on dans chaque cas (Commentaire à compléter) :

```
fill(255, 255, 255, 0);       //
fill(255, 255, 255, 127);     //
fill(255, 255, 255, 255);     //
fill(255, 255, 255);          //
```

LA COULEUR DU CONTOUR (P33)

Commentez les 3ème et 5ème ligne du programme ci-dessous

```
stroke(255, 0, 0);        // le contour sera rouge
ellipse(20, 20, 40, 40);
stroke(0, 255, 0);        //
ellipse(80, 20, 40, 40);
noStroke();               //
ellipse(20, 80, 40, 40);
```

Peut-on remplacer l'instruction **`noStroke();`** par **`nostroke();`** (avec un s minuscule)

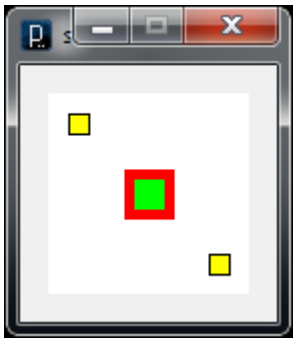
LA PORTÉE DES MODIFICATIONS DE COULEUR (P34)

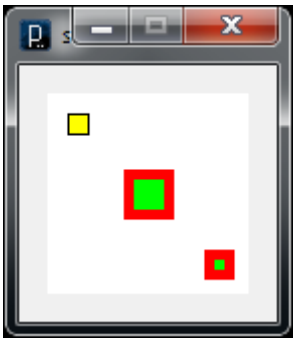
Dans le programme ci-dessous, ajouter un commentaire à toutes les lignes non commentées.

```
size(100, 100);   //
background(255);  //
stroke(#000000);  //
fill(#FFFF00);    //
strokeWeight(1);  // Epaisseur du trait (P48)
rect(10, 10, 10, 10);  //
pushStyle(); // O n ouvre « une parenthèse » de style
stroke(#FF0000);  //
fill(#00FF00);    //
strokeWeight(5);  // Epaisseur du trait (P48)
rect(40, 40, 20, 20);  //
popStyle(); // O n ferme notre parenthèse de style
rect(80, 80, 10, 10);  //
```

Faites ensuite un essai sans les lignes **`pushStyle();`** et **`popStyle();`**
Au lieu de les effacer vous pouvez les mettre en commentaire `// pushStyle();`

Expliquez le rôle des instructions **`pushStyle();`** et **`popStyle();`**





L'ESPACE COLORIMÉTRIQUE (P34)

Lisez le passage concernant la colorimétrie et prenez en compte le complément d'information suivant :
// La teinte sera spécifiée avec un chiffre de 0 à 400. Saturation et Valeur avec un chiffre de 0 à 100
colorM ode(HSB, 400, 100, 100);

« fill(i, 128, 128); » // Petite erreur dans le programme du polycopié puisque « colorMode(HSB, 400, 100, 100); » spécifie des valeur de Saturation et Valeur de 0 à 100
Dans ce cas, « fill(i, 128, 128); » est équivalent à « fill(i, 100, 100); »

Essayez puis commentez les 2 programmes ci-dessous

1er programme

```
noStroke();     //
size(400, 128); //
// La teinte sera spécifiée avec un chiffre de 0 à 400
colorMode(HSB, 400, 100, 100); //
// On fait quatre cent répétitions
for (int i = 0; i < 400; i++) {  // Boucles de 0 à ?
fill(i, 100, 100);  // Teinte 1 à 400 Saturation 100 Valeur 100
rect(i, 0, 1, 128); // 400 rectangles /ligne 0 /largeur 1 /hauteur 128
}
```

2ème pogramme

```
noStroke();     //
size(600, 128); //
// La teinte sera spécifiée avec un chiffre de 0 à 600
colorMode(HSB, 600, 100, 100); //
// On fait six cent répétitions
for (int i = 0; i < 600; i++) {  // Boucles de 0 à ?
fill(i, 50, 100);  //
rect(i, 32, 1, 64); //
}
```

2.4 LE TEXTE (Ch8 P36)

Lisez le chapitre 8 et essayez les programmes. Commentez le programme ci-dessous

```
fill(0);                 // Couleur de remplissage noir R=V=B=0
text("Salut", 10, 20);   // Ecriture en noir colonne 10 ligne 20
fill(255);               //
text("tout", 10, 40);    //
fill(255,0,0);           //
text("le", 10, 60);      //
fill(0,0,255);           //
text("monde", 10, 80);   //
```

Quel est la différence entre les instructions **print** et **text** :

Print :
Text :

EMPILEMENT (P37)

A lire. Pas de difficulté.

2.5 LES IMAGES (Ch9 P39)

IMPORTER UNE IMAGE (P41)

Téléchargez le fichier à partir du lien donné.
Effectuez le travail demandé. Attention, il y a quelques erreurs ou imprécisions.
Faites les modifications nécessaire pour pouvoir utiliser le programme.

Ajoutez des commentaires pour chaque ligne du programme ci-dessous

```
size(500,400);              //
PImage ile;                 //
ile = loadImage("ile.jpg"); //
image(ile,50,10);           //
```

3.1 LES VARIABLES (Ch13 P65)

Complétez le tableau ci-dessous

Mot clé	Type de donnée	Exemple
int	Entier	int x;
float		
boolean		
char		
string		
color		

Une variable peut-être initialisée lors de sa déclaration ou après

Vérifiez que les programmes ci-dessous sont équivalents

`int entier = 3;`
`print(entier);`
	ou	`int entier;`
		`entier = 3;`
		`print(entier);`

`color noir = color(0, 0, 0);`	ou	`color noir;` `noir = color(0, 0, 0);`

Donnez les valeurs minimale et maximale des variable int et float
http://processing.org/reference/ puis Data

Variable	Valeur minimale	Valeur maximale
`int`		
`float`		

3.2 LES CONDITIONS (Ch14 P67)

OPERATEURS DE COMPARAISON

Testez les opérateurs de comparaisons ci-dessous et complétez le tableau

OPERATEUR	FONCTION
==	Egale à
!=	
>	
<	
>=	
<=	

Programme pour tester l'opérateur de comparaison **==**

```
int a = 10;    // Valeur à modifier pour vérifier les 2 cas
if (a == 10) {
  println("a = 10");
} else {
  println("a différent de 10");
}
```

Programme pour tester l'opérateur de comparaison !=

```
int a = 10;    // Valeur à modifier pour vérifier les 2 cas
if (a != 10) {
  println("a différent de 10");
} else {
  println("a = 10");
}
```

Programme pour tester l'opérateur de comparaison >

```
int a = 10;    // Valeur à modifier pour vérifier les 2 cas
if (a > 10) {
  println("a supérieur à 10");
} else {
  println("a inférieur ou égal à 10");
}
```

etc. …

Expliquez la différence entre « **a = 10** » et « **a == 10** »

3.3 LES RÉPÉTITIONS (Ch15 P69)

LA BOUCLE FOR

Lisez les informations sur la boucle for

Expliquez le fonctionnement du programme. (Rappelez le fonctionnement des instructions rec et width)
Pour bien comprendre le fonctionnement du programme, modifiez ensuite la taille de la fenêtre de dessin pour avoir x = 200

```
for (int i = 0; i < width; i = i + 15) {
  rect(i, 0, 10, 10);
}
```

autre écriture

```
int i;
for (i = 0; i < width; i = i + 15) {
  rect(i, 0, 10, 10);
}
```

On peut également utiliser la boucle while (http://processing.org/reference/while.html)

```
int i=0;
while(i<width) {
  rect(i, 0, 10, 10);
  i = i + 15;
}
```

Lisez les passages LES COMLPTEURS et IMBRIQUER DES BOUCLES et essayez de comprendre les programmes en les essayant.

3.4 LES LISTES (Ch16 P73)

Ajoutez des commentaires sur le programme ci-dessous

```
int[] numbers = new int[3];  //

numbers[0] = 90;        //
numbers[1] = 150;       // Même principe que ci-dessus
numbers[2] = 30;        // Même principe que ci-dessus

int somme = 0;          //

for (int i = 0; i < 3; i++) {  // 3 boucle avec i =
  somme = somme + numbers[i];  //
}

println(somme);         //
```

Peut-on modifier les mots **numbers** et **somme :**

Peut-on modifier les mots **int**, **new**, **for**, **println** :

Pour vérification, modifiez le programme pour remplacer le mot numbers par le mot nombres

3.5 LES MÉTHODES (Ch17 P81)

Ce chapitre est un peu plus difficile mais très important.

Les méthodes peuvent être comparées aux fonctions, sous-fonctions, procédures ou sous-programmes d'autres langages.

Pour bien comprendre le programme donné en exemple, vous pouvez annuler l'exécution de certaines lignes du programme puis les remettre ensuite.

```
/* début du commentaire
fin du commentaire */
```

Exemple pour ne pas exécuter les 3 lignes ci-dessous.

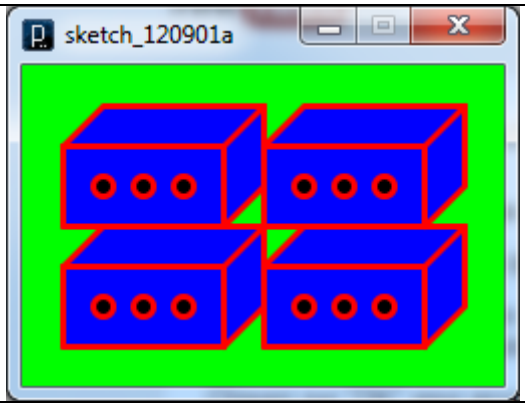
```
/*  ellipse(40, 60, 5, 5);
  ellipse(60, 60, 5, 5);
  ellipse(80, 60, 5, 5); */
```

```
void setup() {
  size(600, 220);          //
  background(153,204,255); //
  smooth();                //

  // l'appel à notre méthode de dessin d'un mouton
  dessinerMouton();   //
  translate(120, 60); //
  dessinerMouton();   //
  translate(120, 60);
  dessinerMouton();   //
  translate(140, -60);
  dessinerMouton();   //
}

// la méthode pour dessiner le mouton
void dessinerMouton() {
  strokeWeight(3);   // Epaisseur ligne P48
  strokeJoin(ROUND); //
  stroke(0);         //
  fill(255);         //
  rect(20, 40, 80, 40); //
  beginShape();      //
  vertex(20, 40);    //
  vertex(40, 20);
  vertex(120, 20);
  vertex(120, 40);
  endShape(CLOSE);    //
  beginShape();       //
  vertex(100, 40);
  vertex(120, 20);
  vertex(120, 60);
  vertex(100, 80);
  endShape(CLOSE);    //
  fill(0);
  ellipse(40, 60, 5, 5); //
  ellipse(60, 60, 5, 5); //
  ellipse(80, 60, 5, 5); //
}
```

Modifiez le programme pour obtenir la figure ci-dessous. La taille des « moutons » n'est pas modifiée.



Taille fenêtre à modifier
Position des »moutons » à modifier
Couleur de fond : Vert
Couleur des lignes : Rouge
Couleur de remplissage : Bleu
Taille des points : Diamètre multiplier par

Lignes à modifier :

3.6 LES OBJETS (Ch18 P88)

Ce chapitre beaucoup plus difficile sera abordé plus tard si nécessaire. Vous pouvez le lire pour une première approche à la POO (Programmation Orientée Objet).

3.7 LES COMMENTAIRES (Ch19 P92)

Chapitre simple. A lire impérativement. Les commentaires sont particulièrement importants dans un programme.

Expliquez à quoi servent les caractères « ***//*** » d'une part, puis « ***/**** et ****/*** » d'autre part

// :

/* et ****/*** :