I. Présentation de Processing.

Processing permet la programmation en langage Java, à l'aide d'instruction simple, on peut très rapidement réaliser des objets graphiques et leurs donner vie. Un programme sous Processing s'appelle un « Sketch ».

Les règles de base dans l'écriture d'un programme sont les suivantes :

- Il faut absolument indenter les programmes. L'indentation consiste en **l'ajout de tabulations** ou d'espaces dans un fichier, pour une meilleure lecture et compréhension du code.
- Il faut impérativement ajouter un**« ; »** à la fin de chaque instruction !!!
- Il faut **ajouter des lignes de commentaires à vos programmes**, ces lignes sont destinées à faire comprendre les parties du programme, elles ne sont pas exécutées lors du programme.

Deux méthodes pour insérer des commentaires dans un programme source :	
Méthode n°1 : Commentaire simple	Méthode n°2 : Zone de commentaire
// Cette ligne est en commentaire	**/* ce paragraphe est en commentaire*/**

: Lancer le logiciel en cliquant sur l'icône Processing sur le bureau puis lire le document : « guide d'utilisation de Processing ».

II. Objectif du travail.

Nous allons réaliser un premier programme en langage JAVA à l'aide du logiciel Processing de façon à dessiner puis animer un petit lapin en écrivant des lignes de codes !!!!

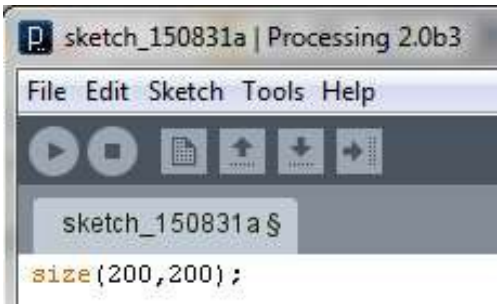
III. Définition de l'espace de dessin sous Processing.

III. 1. Programmation de la fenêtre graphique

La fenêtre de visualisation ou sera placé le ou les objets graphiques affichera les réalisations que vous aurez programmé sous Processing.

Commentaire	Instruction	Paramètres de l'instruction
Cette fenêtre est redimensionnable à l'aide de l'instruction suivante :	`size(l,h);`	`h : hauteur en pixel` `l : largeur de la fenêtre en pixel`

Par exemple, dans la fenêtre d'édition du logiciel	Puis cliquez sur le bouton « **run** », votre fenêtre de

Processing, saisissez la commande suivante : 	visualisation se créé : 

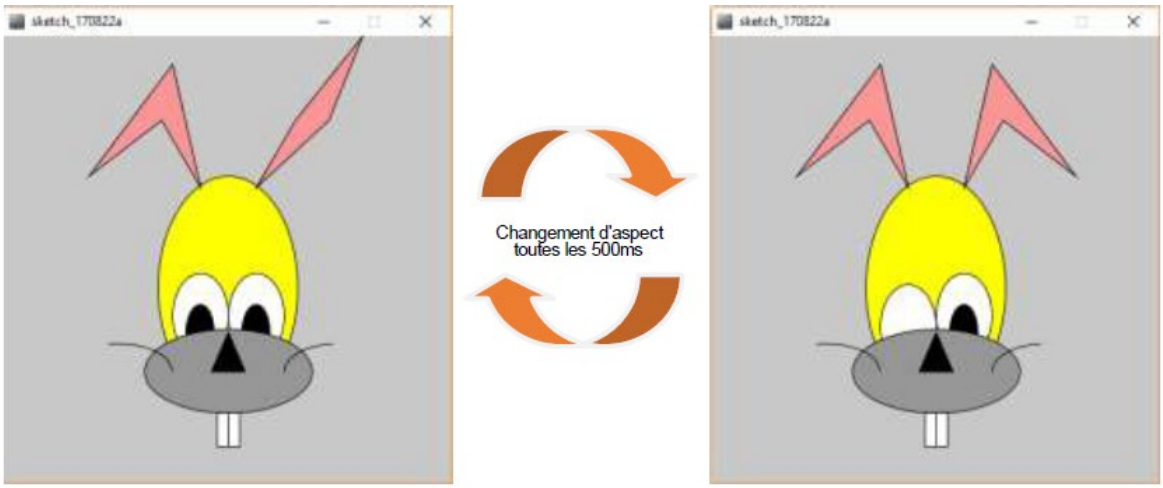
III. 2. Exercice :

Créez par programmation sur votre machine une fenêtre graphique de 400x400 pixels, exécuter le programme sous Processing.

IV. Premier programme et animation graphique

IV. 1. Présentation du travail à réaliser.

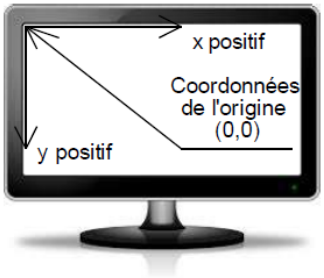
Voici une représentation du lapin dans les deux positions, votre programme devra animer le lapin en permutant les représentations au bout d'un certain temps :



Comme vous pouvez le remarquer, le lapin sera réalisé à partir de formes de bases :

- Ellipse.
- Triangle.
- Quadrilatère.
- Arc de cercle.

Toutes ces formes et bien d'autres sont présentent en tant que fonctions sous Processing, la liste complète ainsi que le descriptif de leur paramétrage est présenté en **annexe 1**.

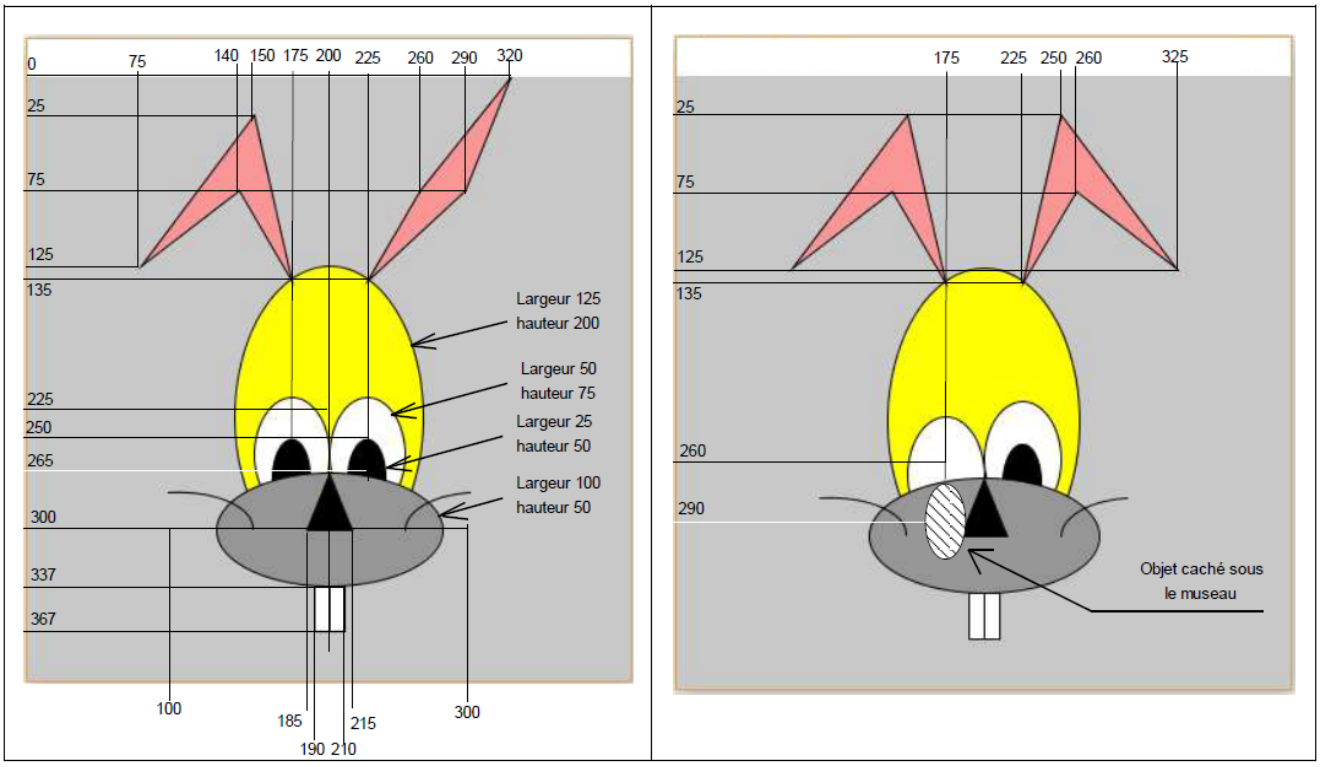
Vous allez travailler en 2 dimensions (2D), pour cela on utilisera deux axes de coordonnées x et y correspondant respectivement à la largeur (axe horizontal) et à la hauteur (axe vertical) d'une situation. Pour dessiner une forme il faudra préciser ses coordonnées sur le plan. Mais attention ici l'origine se situe en haut à gauche de votre écran avec des y positif vers le bas !!!!	

IV. 2. Exercice n°1.

En vous aidant des coordonnées des formes ci-dessous et du squelette de programme ci-après, réaliser une programmation du lapin de gauche.

Attention : L'ordre du code a de l'importance la dernière instruction positionnant toujours la forme au premier plan !!!!

Vous allez devoir saisir le programme ci-dessous, **en complétant les quelques coordonnées manquantes** en vous aidant de la représentation du lapin puis montrer le résultat au professeur.



Remarque 1 : Comme vous pourrez le remarquer l'ordre des instructions à une grande importance !!!

Remarque 2 : Un programme informatique sera exécuté par la machine de façon séquentiel cela signifie que la première instruction exécutée sera « size(400,400) » et la dernière sera « arc(…,…,…,...,...) ».

Remarque 3 :Les deux symboles « { » et « } » encadrant le code source sont obligatoire, ils permettent de définir ce que l'on appelle la portée de la fonction « void setup( ) », (ou l'étendue de la fonction « void setup() ») autrement dit où commencera et où se terminera le code associé à la fonction « setup( ) ».

```
void setup()
{
  size(400,400);
  background(200,200,200);                                  // Permet de mettre le fond de couleur grise
  fill(255,255,0);                                          // couleur de la tête
  ellipse(…….,……,……..,……);                                 // tête
  fill(250,150,150);                                        // couleur des oreilles
  quad(……,…...,……,…...,….,….,….,….);                       // oreille gauche basse
  quad(……,…...,……,…...,….,….,….,….);                       //  oreille
  droite haute fill(255,255,255);                           //
  couleur des yeux
  ellipse(…….,……,……..,……);                                // œil gauche
  ellipse(…….,……,……..,……);                                // œil droit
  fill(0,0,0);                                              // couleur des pupilles
  ellipse(…….,……,……..,……);                                // pupille gauche
  ellipse(…….,……,……..,……);                                // pupille droite
  fill(150,150,150);                                        // couleur du museau
  ellipse(…….,……,……..,……);                                // museau
  fill(0,0,0);                                              // couleur de nez
  triangle(…..,…..,…..,…..,…..,…..) ;                       // nez
  fill(255,255,255);                                        // couleur des dents
  quad(……,……..,……,……,……,……,……,……..);                   //
  dent gauche quad(……,……..,……..,……,……,……,……,……..);//
  dent droite
  noFill() ;                                                // supprime le remplissage
  arc(…...,…...,…...,..….,…...,..….);                      // moustache gauche
  arc(…...,…...,…...,..….,…...,..….);                      // moustache droite
  }
```

IV. 3. Exercice n°2.

Le choix des différentes couleurs du lapin sont complétement arbitraire, après avoir lu les informations de l'annexe 2, proposer une solution permettant de changer les couleurs du lapin ainsi que la couleur de fond, montrer le résultat au professeur.

IV. 4. Exercice n°3 : Créer les nouvelles positions de l'oreille et de l'œil du lapin.

Pour créer une animation, nous devons avoir des dessins dans deux positions différentes, nous avons choisi de modifier la position de l'oreille droite ainsi que de l'œil gauche, nous devons donc disposer de nouvelles instructions dans le programme permettant de positionner l'oreille et l'oeil dans la bonne position.

a) – En vous aidant de la représentation du lapin dans la seconde position, modifier votre programme afin d'afficher simultanément l'oreille droite en position basse et l'oreille droite en position haute à l'écran.

b) – En vous aidant de la représentation du lapin dans la seconde position, modifier votre programme afin d'afficher l'œil gauche dans sa nouvelle position. Vous transformerez en une ligne de commentaire les instructions permettant l'affichage de l'œil dans sa position initiale afin de ne pas perdre le code correspondant.

IV. 5. Exercice n°3 : Animation de l'oreille du lapin.

Nous allons maintenant créer une animation permettant au lapin de s'animer nous commencerons par l'animation de l'oreille qui restera un certain temps en bas puis en haut et ainsi de suite…

Pour cela nous allons devoir modifier le programme légèrement afin de donner vie au lapin.

a) Première étape : Création d'une variable.

Nous allons devoir utiliser une variable pour réussir l'animation.

Une variable peut être considérée comme une sorte de tiroir dans un classeur, ou il sera possible de déposer et/ou de lire une valeur numérique.

Il existe différents type de variables :

- Des nombres entiers « int »,
- Des nombres à virgule « float »,
- Des caractères « char »,
- Des chaînes de caractères « String »,
- Des valeurs *vrai/faux* « boolean ».

Pour notre programme nous utiliserons le type entier « int ».

Il faudra donc saisir en tout début du programme (avant même le setup()) l'instruction suivant permettant de créer le tiroir et d'y placer la valeur 0 :

Insérer alors l'instruction suivante au bon emplacement :

int etat_oreille=0;

b) Deuxième étape : Création de la fonction draw().

Vous allez devoir ajouter une nouvelle fonction dans votre programme en plus de la fonction « **void setup( )** » il s'agit de la fonction « **void draw( )** ».

Cette fonction permet d'obtenir une exécution en boucle des instructions qui y sont présentes tant que la fenêtre graphique est active.

```
……..                                                    // Compléter ici l'instruction manquante
void setup()
{
  size(400,400);
}
void draw()
{
  size(400,400);
  background(200,200,200);                              // permet de mettre le fond de couleur grise
  fill(255,255,0);                                      // couleur de la tête
  ellipse(…….,……,…….,……);                              // tête
  fill(250,150,150);                                    // couleur des oreilles
  quad(……,…..,……,…..,….,….,….,….);                     // oreille gauche basse
  if (etat_oreille==0) {
                          quad(….,…,…,…,…,…,…,…);      // oreille droite haute
                           etat_oreille=1;              // la prochaine fois l'oreille sera basse
                          }
  Else
{
  quad(…,….,….,…..,…..,…..,….,…..);                    // oreille droite basse
  etat_oreille=0;                                       // la prochaine fois l'oreille sera haute
}
  fill(255,255,255);                                    // couleur des yeux
  ellipse(…….,……,…….,……);                              // œil gauche
  ellipse(…….,……,…….,……);                              // œil droit
  fill(0,0,0);                                          // couleur des pupilles
  ellipse(…….,……,…….,……);                              // pupille gauche
  ellipse(…….,……,…….,……);                              // pupille droite
  fill(150,150,150);                                    // couleur du museau
  ellipse(…….,……,…….,……);                              // museau
  fill(0,0,0);                                          // couleur de nez
  triangle(…..,…..,…..,…..,…..,…..) ;                   // nez
  fill(255,255,255);                                    // couleur des dents
  quad(……,…….,…….,……,……,……,……,…….);
  //    dent    gauche    quad(……,…….,…….,……,……,……,……,…….);
  // dent droite
  noFill() ;                                            // supprime le remplissage
  arc(……,…..,…..,……,…..,…..);                         // moustache gauche
  arc(……,…..,…..,……,…..,…..);                         // moustache droite
  delay(500);  }                                        //créer une pause de 0,5s
```

Comme vous pouvez le lire dans le programme on a ajouté un test conditionnel sur la variable **etat_oreille**, (instruction si-alors-sinon).

Suivant sa valeur 0 ou 1, il sera choisi d'afficher l'oreille vers le haut ou l'oreille vers le bas.

Une modification du contenu de la variable (du tiroir) **etat_oreille**, permettra d'obtenir lors de l'exécution suivante l'affichage de l'oreille dans l'autre position.

Finalement une temporisation de 0,5 seconde a été ajoutée en fin de programme afin de figer l'image durant un court instant, avant la prochaine exécution des instructions.

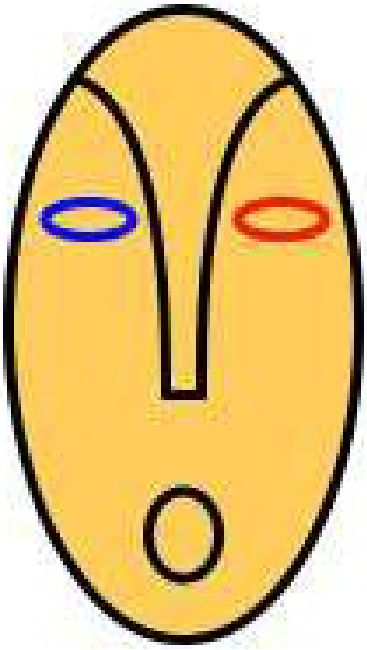
Réaliser cette nouvelle programmation puis montrer le résultat au professeur en vous aidant du squelette du code source ci-dessus.

c) Exercice n°4 : Animation de l'œil du lapin.

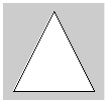
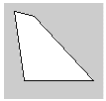
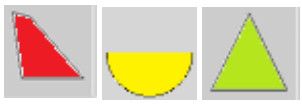
En vous inspirant du programme précédent, proposer une programmation permettant d'obtenir en plus de l'animation de l'oreille, l'animation de l'œil du lapin. Montrer le résultat au professeur.

V. – Exercice pour les plus rapides.

A l'aide de vos connaissances sur la programmation de formes graphique, proposer un programme Java sous Processing permettant d'obtenir le dessin ci-dessous :



Annexe 1 : Instructions (fonctions) permettant de créer des formes précises avec Processing :

Fonction	Paramètres	Intérêt
point(x1, y1) ; 	x1 : Position x du point sur l'écran y1 : Position y du point sur l'écran	Permet d'afficher un point d'un pixel en position(x1, y1) à l'écran
line(x1,y1,x2,y2); 	x1 : Position x1 du 1er point de la ligne y1 : Position y1 du 1er point de la ligne x2 : Position x2 du 2ième point de la ligne y2 : Position y2 du 2ième point de la ligne	Permet d'afficher une ligne entre deux points depuis la position (x1, y1) à la position (x2, y2) à l'écran
rect(x1,y2, L,H) ; 	x1 : Position x du sommet n°1 y1 : Position y du sommet n°1 L : Largeur du rectangle H : Hauteur du rectangle	Permet de créer des rectangles ou des carrés quelconques
ellipse(x1,y1,L,H); 	x1 : Position x du centre de l'ellipse y1 : Position y du centre de l'ellipse L : Largeur du cercle H : Hauteur du cercle	Permet de créer des ellipses ou des cercles suivant les valeurs de L et H
triangle(x1,y1,x2,y2,x3,y3); 	x1 : Position x du sommet n°1 y1 : Position y du sommet n°1 x2 : Position x du sommet n°2 y2 : Position y du sommet n°2 x3 : Position x du sommet n°3 y3 : Position y du sommet n°3	Permet de créer des triangles quelconques
arc(x1, y1, L, H, début, fin) ; 	x1 : Position x du centre de l'arc y1 : Position y du centre de l'arc L : Largeur de l'arc de cercle H : Hauteur de l'arc de cercle début : Angle en radian du début fin : Angle en radian de fin	Permet de créer un arc de cercle entre 2 angles qui seront exprimés en radian
quad(x1,y1,x2,y2,x3,y3,x4,y4); 	x1 : Position x du sommet n°1 y1 : Position y du sommet n°1 x2 : Position x du sommet n°2 y2 : Position y du sommet n°2 x3 : Position x du sommet n°3 y3 : Position y du sommet n°3 x4 : Position x du sommet n°4 y4 : Position y du sommet n°4	Permet de créer des quadrilatères quelconques
fill(R, V, B); 	R, V, B : 3 valeurs comprises entre 0 et 255 permettant de caractériser une teinte de couleur.	Permet remplir les formes avec une couleur précise
noFill( ) ; 	Aucun paramètre, mais instruction à placer avant l'instruction définissant la fonction graphique à dessiner.	Permet de dessiner des formes sans surface de remplissage
noStroke( ); 		Permet de dessiner des formes sans contour

Annexe 2 : Les couleurs des objets et de fond d'écran :

Dessiner une image à l'écran, c'est changer la couleur des pixels. Les pixels sont des petites zones, le plus souvent carrées, qui possèdent une couleur. Chaque couleur se définit par trois canaux qui sont le Rouge, le Vert et le Bleu aussi nommé canaux R V B.

Chacune des trois nuances peut être réglées avec une valeur variant de **0 à 255**, soit un panel de couleurs de : 255 x 255 x 255 = 16581375 couleurs !!!

1. - La couleur de fond :

On peut changer la couleur de fond en appelant la fonction **background(R, V, B);**

Remarque : Utiliser la fonction **background(R, V, B);** dans un programme après y avoir dessiné des formes les effaceraient !!!

2. - La couleur de remplissage d'une forme :

A chaque fois que l'on dessine une forme, on le fait avec la couleur de remplissage qui est choisie à l'aide de la fonction **fill(R, V, B) ;** placée immédiatement avant la fonction dessinant la forme.