

Atelier Processing

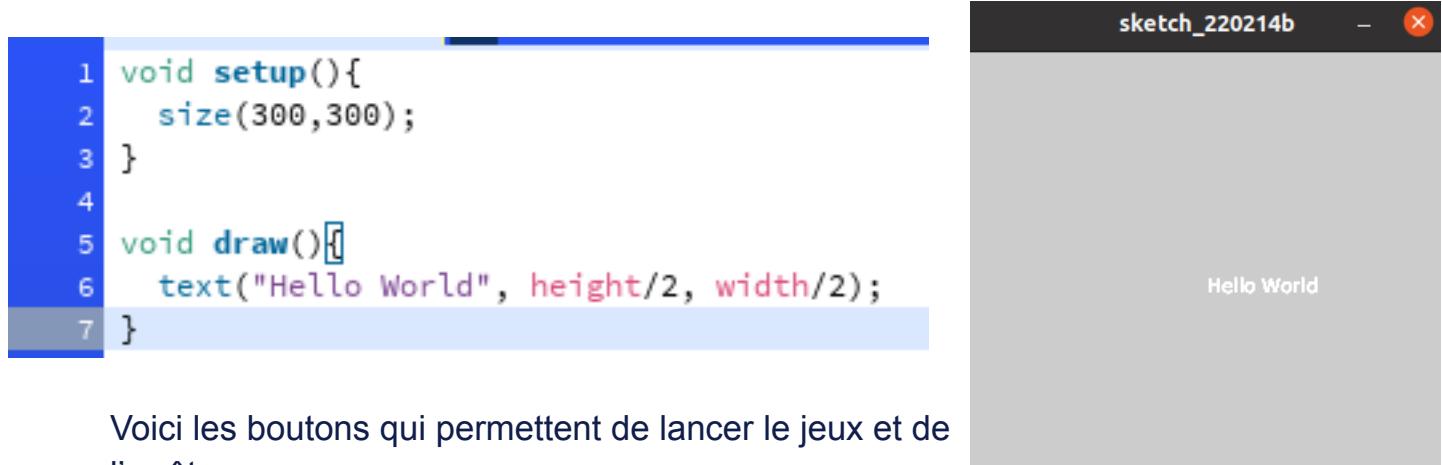
Jeu : Freeshot

Si vous avez un problème sur l'atelier, n'hésitez pas à demander de l'aide à l'étudiant qui s'occupe de la salle.

Voici un visuel du logiciel que nous allons utiliser durant les vingt prochaines minutes. Il s'agit de Processing, un environnement de développement (le langage utilisé est une version simplifiée de Java) qui à la faculté d'avoir un rendu graphique très rapidement et simplement.

La base de tout projet Processing : la fonction “Setup()” et la fonction “draw()”. Setup va permettre de définir des variables au lancement de l'application tel que “size(300,300)” qui permet de définir la fenêtre de l'application avec 300 pixels en largeur et 300 pixels en hauteur.

Et le Draw permet l'affichage d'un élément (60 fois par seconde par défaut). L'application nous affiche donc “Hello world” au coordonnées suivantes → height/2 (prends la hauteur de la fenêtre divisée par deux) et width/2 (prends la largeur de la fenêtre divisée par deux).



The image shows the Processing IDE interface. On the left, there is a code editor with the following Java-like pseudocode:

```
1 void setup(){
2     size(300,300);
3 }
4
5 void draw(){
6     text("Hello World", height/2, width/2);
7 }
```

On the right, there is a preview window titled "sketch_220214b" showing a gray square with the text "Hello World" centered at its midpoint.

Voici les boutons qui permettent de lancer le jeu et de l'arrêter.



Le but du jeu

Le principe est très simple, vous allez apparaître sur une carte avec des zombies et des munitions très limitées. Vous allez devoir survivre et éliminer tous les zombies sans vous approcher de trop ! **Attention une fois à court de munitions, il est pour vous impossible de gagner...**

Voici la structure du code :

Game	Ammo	Bullet	Difficult	Room	Target	Utils	Wall	characterCode	initialization	▼
------	------	--------	-----------	------	--------	-------	------	---------------	----------------	---

Ici vous avez tous les fichiers nécessaires pour le fonctionnement du jeu, seulement les deux premiers : Game et Ammo. Les autres fichiers ne vous seront pas utiles.

1. Pour commencer - Usain Bolt

Une fois le jeu lancé et testé (ZQSD pour bouger), vous vous rendrez rapidement compte que le personnage est extrêmement lent... Transformons le ensemble en une machine de guerre ! Pour cela il nous suffit de modifier la valeur de la variable “*SPEED_CHARACTER*”.

```
final int SPEED_CHARACTER = 3;
```

Chaque variable à un type (int, boolean, float etc) : ça peut être un entier, c'est notre cas. C'est pour cela que nous avons affecté “3” à “*SPEED_CHARACTER*”. Essayer de modifier cette valeur pour comprendre la mécanique. *“Final” signifie que cette valeur ne pourra pas être modifiée au cours de la partie.*

2. Dans la continuité - Un peu de décoration ?

Notre soldat est embêtant ! Il aimerait qu'on lui ajoute un tapis et un lustre dans sa vieille demeure. Pour cela nous allons utiliser des formes géométriques afin de dessiner ces éléments.

Pour commencer nous avons besoin de faire un rectangle pour le tapis, vous pouvez le créer avec ceci : rect(x, y, largeur, hauteur);. **Attention la ligne doit être écrite dans la boucle draw(), sinon l'élément ne s'affiche pas !**

```
17 void draw() {  
18  
19   rect(x,y,largeur,hauteur);
```

Fait pareil pour faire le chandelier avec un rond : circle(x,y, diamètre);

Il faut savoir que le code s'exécute "linéairement" c'est-à-dire qu'il s'exécute ligne par ligne. Ainsi l'ordre des lignes dans le code à une importance ! Si vous créez un élément (cercle, rectangle etc) après la création du personnage, alors ce dernier sera en dessous de l'élément. Essayer de jongler avec l'ordre des lignes pour que *le soldat passe au-dessus du tapis et en dessous du lustre !*



3. Les choses sérieuses commencent... - Les munitions

Vous allez devoir coder un système de munition, il affichera le nombre de munition restante sur l'écran ainsi qu'une petite image. Une fois le nombre de munitions à zéro, le soldat ne pourra plus tirer.

Le rendu devrait ressembler à ça :



Pour se faire, nous allons voir ensemble quelques éléments rudimentaires. Nous aurons besoin d'une image et d'une variable, qui sera le nombre total de munitions (voir ci-dessus).

Voici comment implémenter une image en processing en 2 étapes :

- Créer une variable qui permet de gérer l'image :

1 PImage **ammo**;

- Créer une fonction qui va nous permettre d'afficher l'image :

```
/**  
 * Affichage des balles  
 **/  
void showAmmo() {  
    image(ammo, 0, height-50, 50, 50);
```

ammo = loadImage("ammo.png"); → Permet de charger le fichier ammo.png

image(ammo, x, y, largeur, hauteur): → Permet d'afficher une image (ammo dans notre cas) au position x et y avec une largeur et une hauteur donnée en pixel. Vous devrez remplacer x, y, largeur et hauteur par des valeurs.

Maintenant nous devons afficher un nombre de munitions, en simplement 3 étapes :

- Initialiser une variable (un entier comme vu précédemment) qui sera votre nombre de munitions maximum.
- Maintenant affichez cette valeur à côté de l'image sous forme de texte (voir le rendu ci-dessus) à l'aide de ceci :

```
text("10", x,y);
```

Vous pourrez grâce à ceci créer un texte qui prendra comme valeur "10" et vous placerez ce texte en x et y (valeur à changer selon vos envies). Cependant nous voulons que ce soit "30" qui soit afficher à la place de "10", essayé de mettre la variable que vous avez définis un peu plus tôt à la place du 10 (le nombre de munitions maximum).

```
textSize(32);
```

Utilisez textSize(x), si vous voulez changer la taille du texte. X correspond à la taille de la police en pixel.

Il nous reste maintenant une dernière partie, mais pas la plus facile ! Nous allons devoir faire en sorte que lorsqu'on appuie sur le bouton pour tirer, notre nombre de munitions diminue. Ça tombe bien nous avons créé il y a quelques minutes une variable (qui permet donc de varier) du nombre total de balle.

Premièrement il faudra décrémenter cette variable au clic de la souris c'est -à -dire lui enlever 1 (voir l'endroit à modifier ci-dessous).

```
void mouseReleased() {  
    if(mouseButton == LEFT && canShot()){  
        fire();  
        //Modifier la variable  
    }  
}
```

Tester le jeu normalement tout fonctionne bien... À l'exception du fait que vous pouvez avoir un nombre de munitions négatives ! Il faut donc tester avant de tirer si le soldat à encore des balles. Ceci n'est pas très compliqué, cette fonction doit être capable de nous dire si OUI il peut tirer ou si NON il ne peut pas. C'est ce que signifie le mot "boolean". La variable bool est donc de ce type. Elle prend "faux" comme valeur, c'est-à -dire que NON on ne peut pas tirer.

```
boolean canShot(){  
    boolean bool = false;  
    //A faire  
    return bool;  
}
```

Réfléchissons ensemble quelques instants... On peut tirer seulement si nous avons des munitions... Donc il faut que notre variable soit supérieure à zéro. Maintenant il faut ajouter notre réflexion dans le code.

Voici comment faire une condition de ce type

```
boolean canShot(){  
    boolean bool = false;  
    if(la condition){  
        //Dire oui on peut tirer  
    }  
    return bool;  
}
```

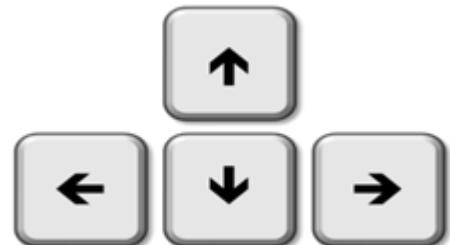
Faites les changements nécessaires dans le code pour que tout fonctionne correctement !

4. Bonus - Touches directionnelle

Actuellement vous pouvez vous déplacer à l'aide des touches ZQSD, vous allez devoir changer le code afin de pouvoir se déplacer avec les flèches directionnelles. Vous pouvez vous aider des documentations ci-dessous.

<https://processing.org/reference/switch.html>

<https://processing.org/reference/keyCode.html>



KeyPressed permet de faire quelque chose à l'appui d'une certaine touche, ZQSD dans notre cas. Et KeyReleased permet de faire quelque chose au relâchement de la touche.

```
void keyPressed() {  
    int speed = (int) SPEED_CHARACTER;  
    switch (key) {  
        case 'z' :  
            soldier.movementCharacterY = -speed;  
            break;  
        case 's' :  
            soldier.movementCharacterY = speed;  
            break;  
        case 'q' :  
            soldier.movementCharacterX = -speed;  
            break;  
        case 'd' :  
            soldier.movementCharacterX = speed;  
            break;  
    }  
}
```

```
void keyReleased() {  
    switch (key) {  
        case 'z' :  
            soldier.movementCharacterY = 0;  
            break;  
        case 's' :  
            soldier.movementCharacterY = 0;  
            break;  
        case 'q' :  
            soldier.movementCharacterX = 0;  
            break;  
        case 'd' :  
            soldier.movementCharacterX = 0;  
            break;  
    }  
}
```

Si vous voyez ce message c'est que vous êtes probablement arrivé à la fin de l'atelier, ou bien que vous êtes curieux ! Dans tous les cas, nous te remercions d'avoir essayé de coder avec nous. Maintenant on peut dire que tu es un jeune développeur ! Et pourquoi pas continuer pour atteindre le rang supérieur ? Si vous avez apprécié cette initiation, c'est que vous avez trouvé votre voie, n'hésitez pas à vous inscrire à l'IUT Informatique pour devenir le développeur de demain.

Auteurs originaux 2022 :

- Bervas Yahn
- Philippe Garnier
- El Mesaoudi Meftah Younes
- Pluviaux Robin