

# PROCESSING : Je débute le code et la programmation.

Un langage de programmation pour les débutants, idéal pour les projets artistiques.

- [Qu'est-ce que Processing ?](#)
- [Comment bien débiter avec Processing ?](#)
- [\(\\*\) Comment créer un petit jeu vidéo avec Processing ?](#)

# Qu'est-ce que Processing ?

**Processing est un langage de programmation** qui a été créé pour faciliter l'apprentissage du code dans les milieux pédagogiques.

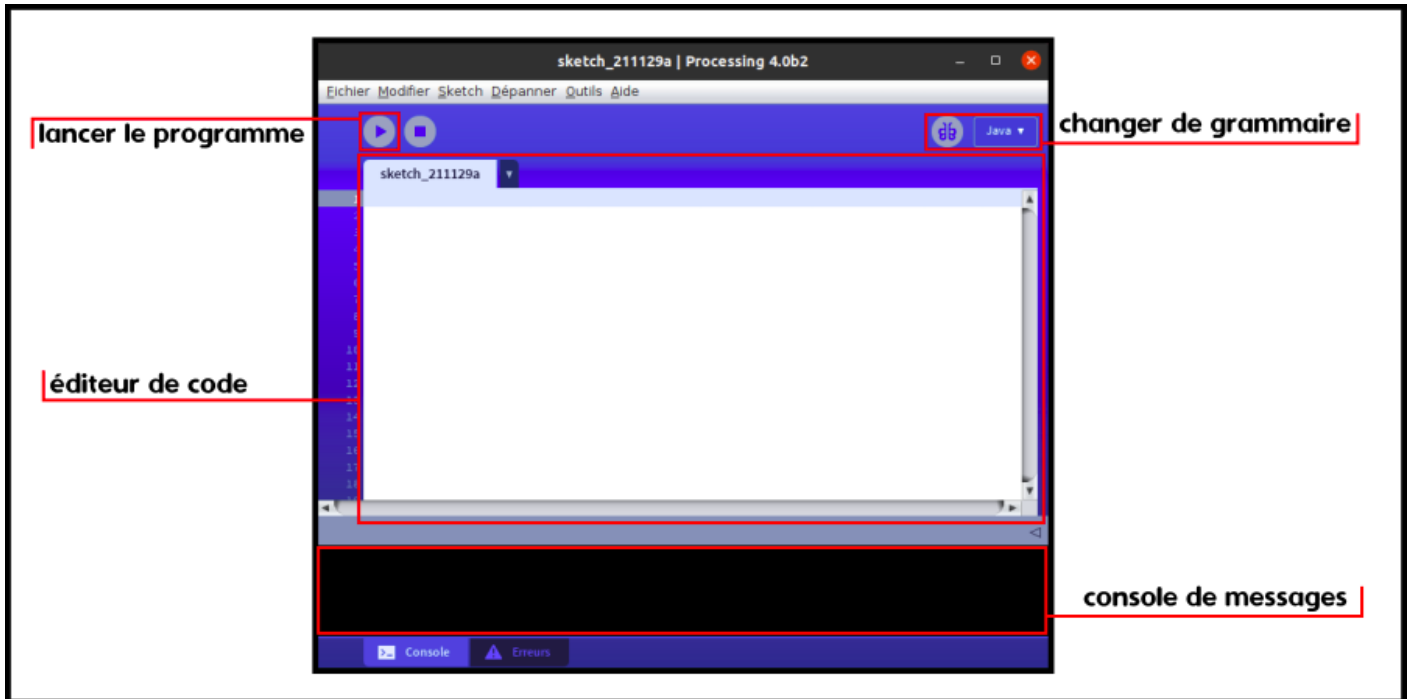
Son intérêt principal réside dans **la facilité de pouvoir créer une fenêtre interactive** sur laquelle on peut dessiner des éléments graphiques.

C'est également **un très petit langage de programmation**, dont on a rapidement fait le tour, bien que les possibilités d'utilisation tendent vers l'infini. Cela permet de **s'initier au code sans se perdre dans des avalanches d'informations**.

**Processing est diffusé sous licence libre, et gratuit, [vous pouvez le téléchargez ici !](#)**

# Comment bien débuter avec Processing ?

Commençons par **jeter un coup d'œil à l'interface !**



Nous pouvons voir plusieurs éléments importants :

- un menu en haut, qui permet d'ouvrir les fichiers, de configurer l'affichage, etc.
- un bouton en forme de flèche qui permet de démarrer le programme
- un bouton, en haut à droite, permettant de changer la grammaire du langage
- un éditeur de texte
- une console pour afficher les erreurs

Pour débuter, je vous conseille de **charger un exemple** ! Ceux-ci sont situés dans le menu *Fichier* > *Exemples*. Je conseille d'ouvrir, dans le dossier *Basics* > *Input*, le fichier nommé *Mouse1D*. Une fois ouvert, lancez le à l'aide du bouton *Exécuter*, en forme de flèche.

Et voilà ! **Prenez le temps de jeter un œil aux exemples** qui vous permettront de vous familiariser avec Processing. Vous **pourrez copier-coller les morceaux de code qui vous intéressent**, et modifier le contenu pour tester des possibilités !

Voyons maintenant les fondamentaux du code. **Deux fonctions spéciales** sont prévues dans Processing. La première est la fonction *setup()*

```
1: void setup() {  
2:   // Votre code ici  
3: }
```

Elle permet d'**indiquer au programme les commandes qu'il doit exécuter au moment où il démarre**. C'est par exemple là que vous indiquerez la taille de la fenêtre à créer.

La seconde s'appelle *draw()*. C'est elle qui contiendra **les instructions pour modifier l'affichage de la fenêtre du programme**. **Vous ne pouvez donner des instructions graphiques que dans cette fonction.**

```
1: void draw() {  
2:   // Votre code ici  
3: }
```

---

Utilisez la fonction *println()* pour afficher des informations dans la console.

```
1: void setup() {  
2:   println( displayWidth, displayHeight );  
3: }
```

---

Enfin, voici [la page de référence](#) pour bien se servir de Processing ! Il s'agit de la documentation, qui vous explique chacune des fonctions préprogrammées dans Processing !

# (\*) Comment créer un petit jeu vidéo avec Processing ?

(\*) : Il manque des informations. Cette page est en cours de construction et n'est donc pas finalisée.

Commençons par **choisir la taille de l'écran** ! 600 x 600 paraît pas mal pour commencer.

```
void setup() {  
  size( 600, 600 );  
}
```

Bien, nous allons maintenant **créer notre joueur**. Ce sera... un gros pixel ! Nous allons lui donner **une position de départ, et référencer sa position**. Pour ce faire, nous utiliserons une variable de type *PVector()*, adaptée aux positions 2D. Puis nous allons **le dessiner**

```
PVector position_initiale;  
PVector position_actuelle;  
  
void setup() {  
  size( 600, 600 );  
  
  position_initiale = new PVector( 20, 20 );  
  position_actuelle = position_initiale;  
}  
  
void draw() {  
  rect( position_actuelle.x - 10,  
    position_actuelle.y - 10,  
    20,  
    20 );  
}
```

Parfait ! Maintenant, nous allons **le faire réagir au clavier** :

```

PVector position_initiale;
PVector position_actuelle;

int vitesse_deplacement = 4;

void setup() {
    size( 600, 600 );

    position_initiale = new PVector( 20, 20 );
    position_actuelle = position_initiale;
}

void draw() {
    rect( position_actuelle.x - 10,
        position_actuelle.y - 10,
        20,
        20 );
}

void keyPressed() {
    if (key == CODED) {
        if (keyCode == UP) {
            position_actuelle.y = position_actuelle.y - vitesse_deplacement;
        };
        if (keyCode == DOWN) {
            position_actuelle.y = position_actuelle.y + vitesse_deplacement;
        };

        if (keyCode == LEFT) {
            position_actuelle.x = position_actuelle.x - vitesse_deplacement;
        };
        if (keyCode == RIGHT) {
            position_actuelle.x = position_actuelle.x + vitesse_deplacement;
        };
    };
}

```

Cela fonctionne... mis-à-part **trois problèmes** !

Premièrement, **les anciennes positions du personnages sont encore affichées à l'écran.**

Cela se règle simplement, en repeignant l'écran entre chaque déplacement, en changeant la fonction *draw()* comme ceci :

```
void draw() {  
    background( 255 );  
    rect( position_actuelle.x - 10,  
        position_actuelle.y - 10,  
        20,  
        20 );  
}
```

Ensuite, le personnage se déplace bizarrement. C'est parce que l'algorithme lié au clavier choisi est le mauvais. C'est dans la fonction *draw()* que l'on peut régler cela :

```
void draw() {  
    background( 255 );  
  
    if (keyPressed == true) {  
        if (key == CODED) {  
            if (keyCode == UP) {  
                position_actuelle.y = position_actuelle.y - vitesse_deplacement;  
            };  
            if (keyCode == DOWN) {  
                position_actuelle.y = position_actuelle.y + vitesse_deplacement;  
            };  
  
            if (keyCode == LEFT) {  
                position_actuelle.x = position_actuelle.x - vitesse_deplacement;  
            };  
            if (keyCode == RIGHT) {  
                position_actuelle.x = position_actuelle.x + vitesse_deplacement;  
            };  
  
        };  
    };  
  
    rect( position_actuelle.x - 10,  
        position_actuelle.y - 10,  
        20,  
        20 );  
}
```

```
}
```

Enfin, il est possible de sortir de l'écran... Voici la solution :

```
void draw() {  
    background( 255 );  
    rect( position_actuelle.x - 10,  
        position_actuelle.y - 10,  
        20,  
        20 );  
}
```

Ce qui donne à ce stade le code suivant :

```
PVector position_initiale;  
PVector position_actuelle;  
  
int vitesse_deplacement = 4;  
  
void setup() {  
    size( 600, 600 );  
  
    position_initiale = new PVector( 20, 20 );  
    position_actuelle = position_initiale;  
}  
  
void draw() {  
    background( 255 );  
  
    if (keyPressed == true) {  
        if (key == CODED) {  
            if (keyCode == UP) {  
                position_actuelle.y = position_actuelle.y - vitesse_deplacement;  
            };  
            if (keyCode == DOWN) {  
                position_actuelle.y = position_actuelle.y + vitesse_deplacement;  
            };  
  
            if (keyCode == LEFT) {  
                position_actuelle.x = position_actuelle.x - vitesse_deplacement;  
            };  
            if (keyCode == RIGHT) {  
                position_actuelle.x = position_actuelle.x + vitesse_deplacement;  
            };  
        }  
    }  
}
```



```

};

if (keyCode == RIGHT) {
    position_actuelle.x = position_actuelle.x + vitesse_deplacement;
};

};

};

if ( position_actuelle.x < 0 ) { position_actuelle.x = position_actuelle.x + 600; };
if ( position_actuelle.x > 600 ) { position_actuelle.x = position_actuelle.x - 600; };
if ( position_actuelle.y < 0 ) { position_actuelle.y = position_actuelle.y + 600; };
if ( position_actuelle.y > 600 ) { position_actuelle.y = position_actuelle.y - 600; };

rect( position_actuelle.x - 10,
position_actuelle.y - 10,
20,
20 );
}

```

[EN CONSTRUCTION]