Le Sneckers d’Elon Mars

version 1.3





1. Introduction

Elon Mars travaille pour une entreprise concurrente à celle où Mark Sugarberg est employé. Jaloux de la réussite de son rival, il souhaite créer son propre jeu pour dépasser le succès du produit de son ennemi juré. Pour cela, il a engagé une grande équipe de développeurs et vous en faites partie !

Son idée révolutionnaire consiste à créer un jeu dans lequel une forme de serpent appelée « Sneckers » doit se nourrir pour grandir.

Vous allez utiliser Processing pour concrétiser son idée. Processing, c’est un outil permettant d’apprendre à coder par la création graphique. Il est utilisé par des étudiants, artistes, chercheurs et curieux pour apprendre ou créer rapidement des visuels.

Par exemple, la chaîne YouTube [Coding Train](https://www.youtube.com/channel/UCvjgXvBlbQiydffZU7m1_aw), qui fait la découverte de l’algorithmie, se sert souvent de Processing dans ses vidéos. Allez y jeter un œil !

1. Consignes

* Pour ce projet-là, il vous est demandé de choisir comme nom du repository : cc\_Sneckers.. Si vous ne savez pas utiliser git allez lire le tutoriel « Le coffre à jouet du petit git ».
* Vous devez installer Processing en le téléchargeant à ce [lien](https://processing.org/download/).
* Vous pouvez utiliser la [documentation](https://processing.org/reference/) de Processing pour en apprendre plus.
* En cas de question, pensez à demander de l’aide à votre voisin de droite. Puis de gauche. Ou inversement. Puis demandez enfin à un Cobra si vous êtes toujours bloqué(e).

1. L’enclos du Sneckers.

Votre première mission est de créer un environnement pour votre Sneckers !

Sur Processing, setup() et draw() sont des fonctions particulières que vous devez écrire pour avoir du mouvement à l'écran. C'est l'une des rares notions compliquées de Processing.

Commencez par écrire simplement :

void setup()

{

  size(800, 800);

  background(255);

}

void draw()

{

  background(0);

}

Comme son nom l'indique, setup() (« configuration ») sert à préparer le reste du programme. Il ne passera donc qu’une seule fois par cette fonction au tout début.

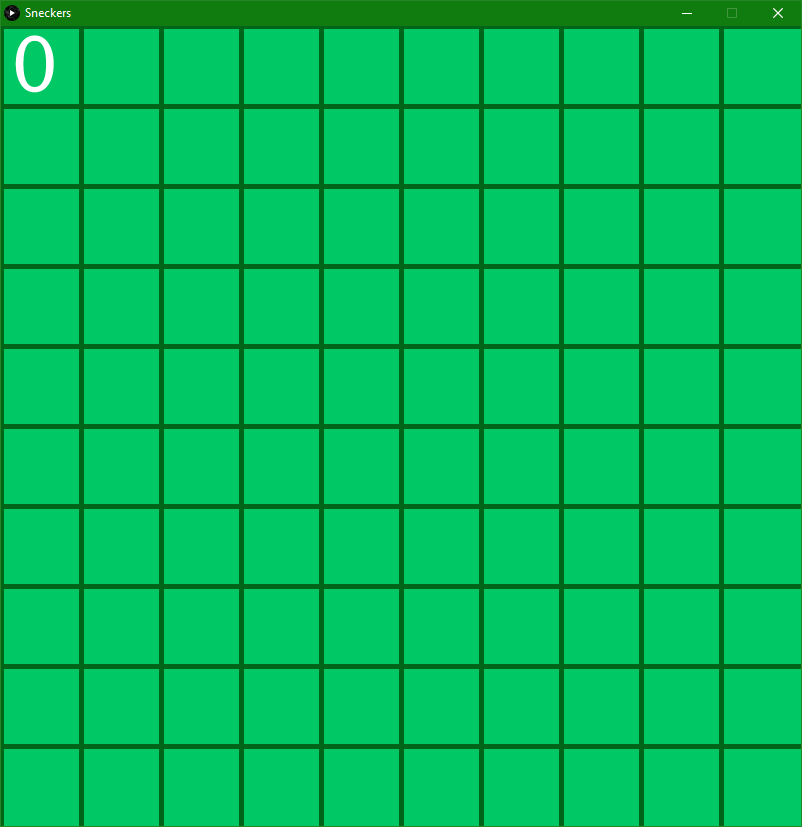
A l'inverse, draw() (« dessin ») est appelé plusieurs fois pour réafficher à chaque fois ce qu'on lui demande. Elle est appelée 60 fois par seconde !

Tout notre programme va dépendre de ces fonctions car elles sont utilisées par Processing lui-même !

Il est temps de passer aux choses sérieuses : faites une fonction qui affiche une grille verte de 10 cases par 10 cases et qui prend en paramètre le nombre de cases par ligne/colonne.

Ensuite, il faut créer une variable pour retenir le score et une fonction pour l’afficher en haut à gauche de l’environnement.

Vous devriez obtenir quelque chose comme ça :



Environnement pour le Sneckers avec le score et le quadrillage.

1. La naissance du Sneckers

Il est maintenant temps de créer le Sneckers et de le poser dans son nouvel environnement. On va commencer par écrire une « class » qui vous permettra de créer un Sneckers.

En Processing, une class ressemble aux quelques lignes ci-dessous. Elle se compose de variables, d’un « constructeur » et de fonctions internes.

class Sneckers

{

    int variable; // variable de la class

    // Constructeur

    Sneckers(int valeur)

    {

        // la variable prends la valeur donné au constructeur

        variable = valeur;

    }

    // Fonction de la class

    void do\_something()

    {

        // code

    }

}

Le Sneckers est un « objet », il doit pouvoir être créé par la class grâce au constructeur. Une class est une sorte de moule pour créer des objets. On écrit donc des indications sur le fonctionnement d’un Sneckers.

Le Sneckers a besoin de deux variables :

* D’abord, il doit pouvoir se situer dans le quadrillage du terrain. Vous pouvez imaginer le corps du Sneckers comme une liste de positions par lesquelles il passe. Vous allez utiliser les [ArrayList](https://processing.org/reference/ArrayList.html) de Processing pour le représenter. Commencez par un serpent de 3 carrés.
* Il faut aussi qu’il sache dans quelle direction il va pour ne pas se perdre. Réfléchissez au fonctionnement des directions et créez une variable pour garder la direction en cours.

Créez aussi des fonctions pour donner vie au Sneckers :

* Il faut une fonction pour qu’il soit affiché sur son environnement et au bon endroit ! Commencez par positionner sa tête, soit le premier élément de la liste, au centre de la fenêtre et son corps vers le bas. Soyez créatif sur les formes et les couleurs de votre Sneckers !
* Pour se déplacer, le Sneckers remplace chacun des éléments de son corps par le suivant, sauf le premier qui avance d’une case vers sa direction. Faites une fonction pour ça !
* Lorsque le Sneckers mange, il doit pouvoir agrandir son corps d’un élément. Heureusement, l’ArrayList que vous avez utilisé contient une fonction que vous pouvez utiliser pour ça.
* Enfin, le Sneckers doit repérer s’il a touché un mur ou s’il s’est mordu la queue. A ce moment-là, la partie est perdue. N’oubliez pas que la tête est l’élément le plus important.

Voici les prototypes des fonctions que vous devez créer :

class Sneckers

{

/\*……\*/

Sneckers()

void display()

void move()

void eat()

void lose()

}

Maintenant que le moule à Sneckers est prêt, vous pouvez enfin en générer un et le placer dans son environnement. Avec Processing, on créée un objet de la manière suivante :

Sneckers snek = new Sneckers();

Bon, tout ça c’est bien, mais le serpent ne bouge que dans la direction donnée à la construction, c’est pas terrible… Changez la valeur de la direction en fonction d’un appui de touche. Utilisez la fonction [keyPressed](https://www.processing.org/reference/keyPressed_.html) de Processing, elle fonctionne comme draw() mais n’est appelée que si une touche du clavier est appuyée. Vous trouverez quelle touche est utilisée grâce à la valeur de [key](https://www.processing.org/reference/key.html) 😉

1. Le Sneckers doit se nourrir

Il est temps de nourrir le Sneckers, effectivement même si ce dernier est numérique, il doit quand même se sustenter pour survivre et grandir. Pour ce faire, vous devez créer un nouvel objet : Ourson !

Chaque ourson en guimauve doit juste connaître sa **position**. Pour **construire** un ourson, il faut donc qu’il se donne une **position** aléatoire.

Créez-leur aussi deux fonctions :

* L’ourson doit pouvoir s’afficher à la bonne **position** dans la grille.
* Evidemment, un ourson en guimauve ça doit pouvoir être mangé ! Pour ça, il faut lui indiquer la position de la bouche qui tente de le croquer et lui répondre si oui ou non elle a attrapé l’ourson.

Votre class ressemblera donc à ceci :

class Ourson

{

/\*……\*/

Ourson()

void display()

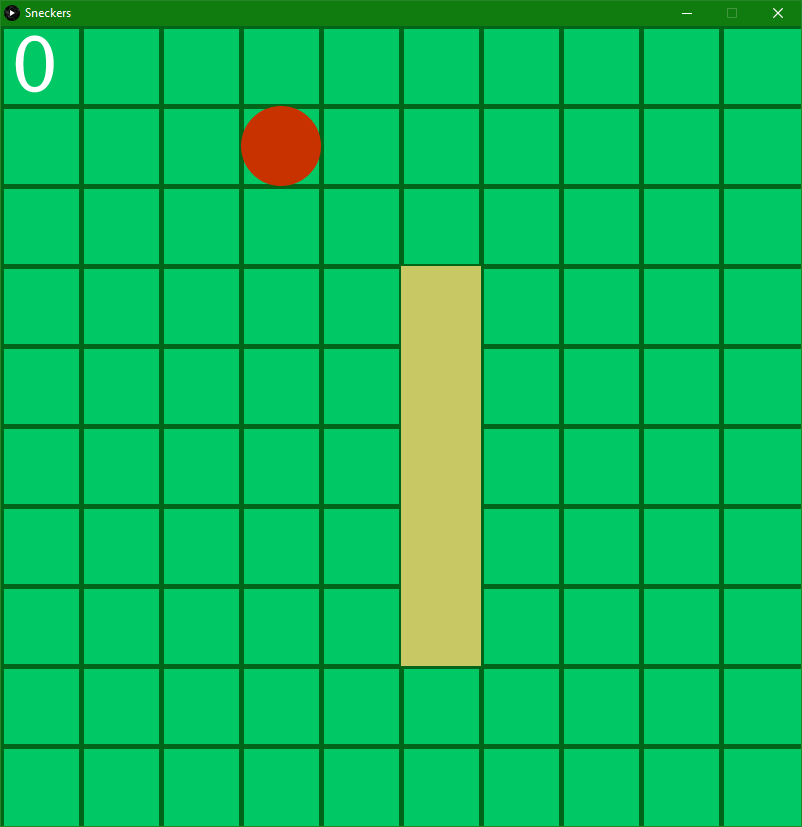
void beEaten()

}

N’oubliez pas de poser un ourson au début du jeu et d’en rajouter un à chaque fois que qu’il est mangé. Si vous pouvez manger l’ourson sans le faire disparaître, c’est qu’il vous manque quelque chose dans votre programme.

Lorsque le serpent mange un ourson, c’est qu’il progresse. Pensez à augmenter son score en conséquence !

Et voilà, vous pouvez maintenant jouer et faire grandir votre Sneckers jusqu’à remplir la fenêtre !



*Exemple de jeu de Sneckers terminé.*

1. Conclusion

Elon Mars est satisfait du travail que vous avez fourni ! Le jeu est bien parti pour devenir populaire dans le monde entier ! Cependant, Elon Mars revient vers vous, il voudrait que le jeu continue de gagner en popularité ! Pour ce faire, il vous demande d’améliorer le jeu pour qu’il soit plus attractif.

Pour ça, il a déjà quelques idées, Elon veux que le jeu ait un « écran titre » au lieu de commencer directement. Il vous donne aussi le logo qu’il voudrait utiliser et que vous pourrez trouver [ici](https://fontmeme.com/snickers-font/).



Exemple de titre

Voilà des idées de bonus que vous pouvez réaliser mais vous pouvez aussi être créatif et en imaginer vous-même :

* Le Sneckers rétrécit s’il ne se nourrit pas assez vite ou s’il se mord la queue.
* Créer un Menu pour afficher le score entre deux parties et faire des pauses.
* Permettre au joueur de personnaliser l’environnement et son Sneckers.
* Plus le Sneckers grandit plus il devient rapide.
* Ajouter de la musique.