

Projet Snake

🕒 Conseils

- l'objectif est de vous faire tous travailler sur le projet, mais n'hésitez pas à discuter entre vous;
- toutes les notions utiles sont disponibles soit dans le document joint soit dans la documentation¹;
- la plupart des réponses se trouvent dans les questions;
- posez des questions 😊

Contexte

Le jeu **Snake** est un des jeux les plus populaires au monde². L'objectif du jeu est de contrôler un petit serpent. Ce serpent peut se déplacer dans une zone de jeu bien définie et a pour objectif de manger des pommes qui y sont placées aléatoirement. Chaque pomme mangée fait grandir le serpent, qui devient alors plus difficile à contrôler. Le jeu se termine si le serpent quitte la zone de jeu, si sa tête touche une autre partie de son corps ou (dans le meilleur des cas), s'il remplit toute la surface de jeu.

💡 Aide

Dans le squelette de code fourni se trouvent de multiples informations sur les endroits où écrire les morceaux de code. Utilisez ça à votre avantage !

Organisation de la zone de jeu

On définit une **position** dans la zone de jeu par une paire de nombre qui indiquent la distance au coin supérieur gauche : **(0,0)**. La première coordonnée concerne la distance **horizontale** et la seconde la distance **verticale**.

On définit une **unité** comme un nombre de pixels qui vont permettre de diviser la zone de jeux dans une grille. La première colonne (ou ligne) sera **0**, la dernière aura l'indice **max - 1**.

Chaque **élément de jeu** est composé d'un carré de taille **unité**, identifié par la position de son coin supérieur gauche, qui correspondent à sa colonne et sa ligne.

Le serpent commencera **au milieu** de la zone de jeu.

Le serpent est composé de plusieurs carrés de la même couleur. Quand il se déplace, on considère que la tête se déplace dans la direction donnée et que le dernier carré disparaît.

Dans la figure **1**, vous pouvez observer un exemple simplifié d'une grille de **30x20**. Le serpent fait ici **5** carrés de long et la pomme est à la position **(14, 5)**. Bien sûr, vous ne représenterez pas les coordonnées à l'écran, elles sont juste là pour la visualisation.

¹. <https://www.pygame.org/docs/>

². au moins à une certaine époque

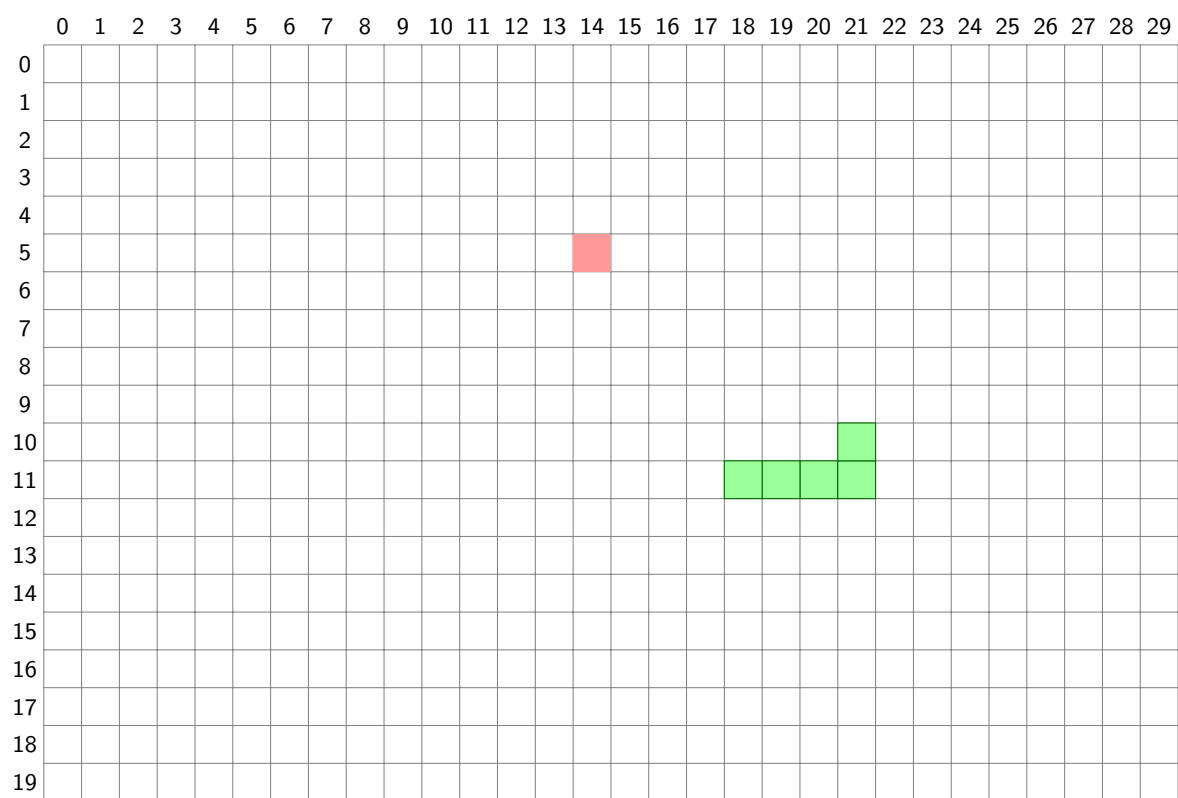


Figure 1 – Exemple de grille

Étape 1 - Définition des variables

Comme dans tout bon projet de code, on va commencer par définir quelques variables qui vont nous permettre de gérer la zone de jeu proprement.

Vous aurez besoin de:

- une taille **unite**, à régler à **10** pour le moment;
- une taille **largeur_zone**, réglée à **60** unités;
- une taille **hauteur_zone**, réglée à **40** unités;
- une couleur **couleur_nourriture**³;
- une couleur **couleur_serpent**³;
- une couleur **couleur_fond**³.

Étape 2 - Préparation de la zone de jeu

Dans cette étape, on se concentre sur la création et la préparation de la zone de jeu.

1. mettez à jour le squelette pour construire une fenêtre de telle sorte à ce qu'elle utilise les variables de largeur et de hauteur de zone que vous avez définies;
2. ajoutez une couleur de fond à la fenêtre;
3. écrivez le corps de la fonction **dessiner_carre**. Cette fonction prend 3 paramètres: **x**, **y** et **couleur**. Elle dessinera un carré de taille **unite** à la position (**x**, **y**).

💡 Aide

N'hésitez pas à tester votre fonction dans votre grille pour vérifier que vous avez bien un carré de la bonne couleur au bon endroit.

Étape 3 - Une pincée d'aléatoire

On veut pouvoir faire apparaître la nourriture à des endroits aléatoires de la grille, sinon le jeu est trop facile. Pour pouvoir faire ça, il faut qu'on soit en mesure de générer des positions aléatoires.

1. Écrivez une fonction **entier_aleatoire** qui prend un paramètre **max** et qui renvoie un entier aléatoire entre **0** et **max** (exclus).
2. Écrivez une fonction **position_aleatoire** qui renvoie une position (donc un couple de coordonnées) qui doit être **valide**, c'est à dire qui est dans la zone de jeu. Vous devriez utiliser la fonction que vous venez de définir et les variables de l'étape 1.

💡 Aide

La structure de données la plus efficace pour représenter ce couple d'entiers est le **tuple**

Étape 4 - Afficher plusieurs carrés

Le serpent va vite être constitué de plusieurs carrés. On va donc devoir définir une fonction **dessine_liste_carres** qui prend en paramètres:

- une liste de positions;
- une couleur.

et qui dessine pour chaque position dans la liste un carré. Vous utiliserez bien sûr la fonction que vous avez définie lors de l'étape 1. Tous les carrés de la liste seront de la même couleur.

³. il y a des exemples de couleurs dans le squelette, mais n'hésitez pas à utiliser les vôtres !

Étape 5 - Variables des éléments de jeu

On va redéfinir quelques variables supplémentaires, qui cette fois vont nous aider à stocker des informations importantes sur la partie. Vous allez avoir besoin de:

- une liste **positions_serpent**, qui représente la liste des positions des carrés constituant le serpent. Au départ, vous pouvez n'y mettre qu'un seul élément qui est le milieu de la zone de jeu.
- un booléen **fin_jeu**, qui nous indiquera que la partie est terminée. Bien sûr, cette variable vaut **False** au départ.
- la direction du serpent **direction_serpent**, chaîne vide au départ.
- la position de la pomme **position_pomme**, qui sera initialement choisie en appelant une fonction bien choisie.

Étape 6 - Un tour de jeu

Un tour de jeu est constitué de plusieurs actions, que l'on va décrire maintenant:

1. appeler la fonction **dessine_carre** pour faire apparaître la pomme au bon endroit;
2. appeler la fonction **afficher_message** pour faire apparaître le score en haut à gauche. On définit le score comme étant la longueur du serpent moins 1.
3. appeler la fonction **dessine_liste_carres** pour afficher le serpent.
4. utiliser la variable **fin_jeu** dans la boucle principale.

Étape 7 - Les directions

Notre serpent va pouvoir se déplacer dans plusieurs directions. On va créer une fonction **direction**, qui va prendre deux paramètres: la touche sur laquelle on appuie et la direction du serpent actuelle. On se base ensuite sur la touche pressée pour changer la direction. Si c'est:

- **HAUT**, alors la coordonnée **x** ne change pas et **y** augmente de 1;
- **BAS**, alors on ne touche pas à **x** et on décroît **y** de 1;
- **GAUCHE**, alors **x** décroît de 1 et **y** ne change pas;
- **DROITE**, alors **x** augmente de 1 et **y** ne change pas.

La fonction renverra un tuple d'entiers qui représente les changements pour **x** et **y**. Si une autre touche est pressée, alors on ne change rien bien sûr.

Étape 8 - Réagir à un évènement

Deux évènements principaux peuvent arriver pendant la partie. Il faudra gérer ce qu'il se passe pour chaque.

1. Le joueur appuie sur la bouton de sortie. Dans ce cas, la partie se termine, une variable doit changer.
2. Le joueur appuie sur une touche. Dans ce cas, la direction du serpent peut changer. Utilisez une fonction définie précédemment pour gérer ceci.

💡 Aide

Vous pouvez utiliser **event.type** pour obtenir le type d'un évènement.

Étape 9 - Calcul de la nouvelle position

Écrivez une fonction **nouvelle_position** qui prend en paramètre le serpent et la direction actuelle. La fonction renvoie la nouvelle position de la tête du serpent.

Vous calculerez la nouvelle coordonnée à partir de la direction. Pour faire plus simple, vous pourrez ajouter la nouvelle position à la fin de la liste des positions du serpent, mais gardez en tête que c'est le dernier élément de cette liste qui représentera la tête à chaque fois.

Étape 10 - Respecter la zone

On ne veut pas que notre serpent puisse quitter la zone (ça pourrait même être un motif de fin de partie). Écrivez donc une fonction **dans_zone** qui renvoie **True** si la position est dans la zone de jeu et **False** sinon.

🔗 Aide

Utilisez les variables que vous avez définies à l'étape 1 pour déterminer si on est dans la zone!

Étape 11 - Déplacer le serpent

Il faut maintenant déterminer la nouvelle position de la tête et vérifier deux choses :

1. si la tête du serpent est dans la zone de jeu, alors elle est ajoutée au serpent, sinon, la partie est finie;
2. si la tête du serpent est sur la pomme, alors il faut déterminer une nouvelle position valide pour la nourriture. Sinon, alors on retire de la liste la position de la queue du serpent.