2021

Snake
Rapport de projet

Amstutz Thomas
DIVTEC
25/01/2021

Table des matières

1 Biographie 1
2 But du projet 1
3 Contexte 1
3.1 Snake 1
4 Etapes du projet 1
4.1 Choix du projet 1
4.2 Editeur de code 2
4.3 Création du quadrillage 2
4.4 Création du Painter 2
4.5 Création des points 2
4.6 Création du serpent 2
4.7 Placement de la nourriture 2
4.8 Menu 3
4.8.1 Début de partie 3
4.8.2 Fin de partie 3
4.9 Mode pause 3
4.10 Multijoueur 3
4.11 Paramètres 3
5 Etat du projet 4
5.1 Fonctionnalités implémentées 4
5.1.1 Serpent 4
5.1.2 Pomme 4
5.1.3 Modes de jeu 4
5.1.4 Paramètres 5
5.2 Problèmes connus 5
5.2.1 Angles 5
5.3 Problèmes rencontrés 6
5.3.1 Demi-tour 6
5.3.2 Mode pause 6
5.4 Améliorations possibles 6
5.4.1 Image de fond 6
5.4.2 Pomme dorée 6
5.4.3 Choix de la difficulté 6

5.4.4 Multijoueur en LAN 7
5.4.5 Modification de la vitesse de déplacement 7
5.4.6 Mode « Tron »..... 7
5.4.7 Génération aléatoire de murs..... 7
6 Ce que j’ai appris..... 8
7 Conclusion..... 8
8 Sources 9

1 Biographie

Je m'appelle Thomas Amstutz. J'ai actuellement 18 ans et je suis en troisième année d'apprentissage en informatique à l'Ecole de Métiers Techniques de Porrentruy.

2 But du projet

Dans le cadre de l'atelier 31, nommé « Programmation Orientée Objet », qui fait partit de la troisième année de formation des apprentis informaticiens, nous devons réaliser un petit jeu 2D en C++ ou en Java.

Ce projet permet de mettre en lien la théorie apprise en module et la pratique de la programmation comme on la retrouverait en entreprise.

3 Contexte

3.1 Snake

Snake est un genre de jeu où un serpent doit manger le plus de nourriture afin de remporter un maximum de points.

> *Le concept original vient du jeu Blockade crée par Gremlin en 1976 pour les bornes d'arcade. Plusieurs clones sont sortis dans les années qui suivent. La version de Snake la plus connue est celle que Nokia a inclus dans ses téléphones. - Wikipédia*

4 Etapes du projet

4.1 Choix du projet

Bien avant de coder, j'ai dû choisir le jeu que j'allais réaliser. J'ai hésité un moment entre le Snake et un jeu de plateforme comme Mario.

J'ai finalement choisi de faire un Snake car je pense que je n'aurais pas eu le temps de faire un jeu de plateforme comme je le voudrais et que Snake est un genre de jeu que j'ai toujours bien aimé et qui peut être fait de multiples façons différentes et avec diverses fonctionnalités. L'idée de refaire ce jeu m'est venu du jeu Snake III auquel je jouais sur un vieux téléphone Nokia à clapet.

4.2 Editeur de code

J'ai décidé d'utiliser IntelliJ comme éditeur de code, il est extrêmement polyvalent et il peut être complété à l'aide de nombreuses extensions comme les autres logiciels de la suite JetBrains. Depuis le début de mon apprentissage, j'utilise principalement les logiciels JetBrains pour réaliser les projets de l'école.

4.3 Création du quadrillage

Le plateau de jeu est composé d'un quadrillage. Chaque case de ce quadrillage fait une taille définie par la constante `TILE_SIZE`. Le quadrillage se sépare donc en colonnes et en lignes. Tous les éléments du jeu (hors textes) sont placés selon ce quadrillage.

4.4 Création du Painter

Le `Painter` est la classe qui s'occupe d'afficher des éléments à l'écran, que ce soit des textes ou des images. À chaque exécution de la boucle de jeu, tout s'efface et se reconstruit en commençant par la pomme puis le serpent et en finissant par afficher les textes.

4.5 Création des points

Les éléments du jeu (pommes et serpents) sont composés de points. Chaque point est placé dans une case de la grille. Les points peuvent être déplacés avec la méthode `translate()`. Chaque point à un sprite qui peut être affichée à l'écran depuis la classe `Painter`. Les points ont tous un type : `HEAD`, `TAIL`, `BODY` ou `FOOD`. Les trois premiers sont pour les serpents et le dernier représente la nourriture

4.6 Création du serpent

Le serpent est composé d'une liste de points de serpent (`SnakeDot`), les points à l'extrémité de la liste représentent la tête et la queue du serpent et ont les types `HEAD` et `TAIL`. Le reste du corps est de type `BODY` Chaque point du serpent a sa propre direction et le serpent a une direction générale qui suit la direction de la tête.

4.7 Placement de la nourriture

L'emplacement de la nourriture est généré aléatoirement. Une nourriture ne peut se placer sur la tête du serpent. Quand un serpent passe sur la nourriture, celle-ci disparait et une autre est créé à un autre endroit.

4.8 Menu

Tous les menus du jeu se composent de deux parties, un affichage des textes et des `keyListener` qui détectent les appuis de touches et décident des actions à effectuer.

4.8.1 Début de partie

Le menu principal permet de quitter le jeu, de lancer une partie en solo ou en multijoueur ainsi que d'accéder aux paramètres.

4.8.2 Fin de partie

Quand un serpent meurt, un menu permettant de recommencer une partie, de changer de mode de jeu ou de quitter apparait.

4.9 Mode pause

En appuyant sur **P**, le jeu se bloque et des commandes apparaissent à l'écran. En appuyant sur **P** une nouvelle fois, le jeu reprend. Ce système fonctionne à l'aide d'une variable qui bloque ou débloque le jeu à l'intérieur de la boucle de jeu.

```
timeline = new Timeline(new KeyFrame(Duration.seconds(1), event -> {
      if (!isPaused()) {
            ...
      } else {
            Painter.paintPause();
      }
}));
```

4.10 Multijoueur

Le mode multijoueur permet à deux personnes de jouer. Les serpents sont contrôlés avec les touches **WASD** et ↑←↓→. Ils ont chacun une couleur sélectionnée aléatoirement et les scores contiennent la couleur du serpent afin que les joueurs puissent se différencier.

4.11 Paramètres

L'utilisateur peut modifier des paramètres du jeu dans la section **config** du menu principal (en appuyant sur **C**). De là, tous les paramètres peuvent être modifié et sauvegardés. Tous ces paramètres sont stockés à l'intérieur du fichier **settings.config**. À chaque démarrage de l'application, le programme va chercher le contenu de ce fichier afin de le stockes dans des énumérés qui sont ensuite utilisés dans le reste du programme.

5 Etat du projet

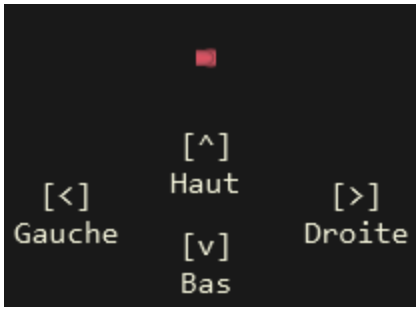
5.1 Fonctionnalités implémentées

5.1.1 Serpent

Les serpents, contrôlés par les joueurs, peuvent manger des pommes, se déplacer et entrer en collision avec eux-mêmes, un mur ou un autre serpent.

- Taille
  Les serpents ont une taille initiale de trois blocs. Après avoir mangé une pomme, ils vont grandir d'un bloc. Un serpent n'a pas de taille maximale mais plus le serpent est grand, plus le jeu est dur.

- Déplacement
  Au début de la partie, le serpent ne se déplace pas. Le joueur doit appuyer sur une des flèches pour qu'il commence à se déplacer. Une fois lancé, le serpent ne s'arrête pas tant qu'il n'est pas mort. Il ne peut se déplacer qu'en haut, à gauche, en bas ou à droite.



Un serpent en début de partie

- Couleur
  À chaque lancement de partie, le serpent prendra une couleur aléatoire : rouge, bleu, vert ou jaune. En multijoueur, les serpents auront forcément une couleur différente l'un de l'autre.

- Collision
  Le serpent peut rentrer en collision avec lui-même, un mur ou un autre serpent ce qui le tue sur le champ et mets fin à la partie en cours.

5.1.2 Pomme

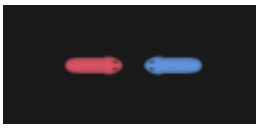
Il y a une seule pomme à la fois sur le plateau. En manger une fait grandir le serpent d'un bloc.

- Placement
  Les pommes sont placées au hasard. Elles peuvent se trouver n'importe où sur le plateau sauf à l'emplacement de la tête du serpent.

5.1.3 Modes de jeu

- Solo
  En solo, un seul joueur doit manger le plus de pommes afin de faire le plus de points possibles sans entrer en collision avec sa queue (ou les murs si cette option est activée).

- Multijoueur
  En multijoueur, deux joueurs s'affrontent afin de faire le maximum de points sans entrer en collision avec les éléments cités plus haut.



Deux serpents lors d'une partie en multijoueur

5.1.4 Paramètres

- Stockage
  Tous les paramètres sont stockés dans le fichier **settings.config**. Les données sont stockées sous ce format : `NOM_DU_PARAMETRE=valeur`.

```
WALLS=false
SNAKE_RAINBOW_SHEDDING=false
LEGACY_SNAKE=false
```

Format du fichier de paramètres

- Modification
  Lors de la sauvegarde depuis le menu de paramètres, le fichier est vidé et réécris avec les nouveaux paramètres.

- Lecture
  Le fichier est lu à chaque lancement de l'application et à chaque fois qu'on quitte le menu de paramètres sans sauvegarder.

5.2 Problèmes connus

5.2.1 Angles

Lorsque que le serpent tourne, son corps créé des angles à 90°. Ces angles ne fonctionnent pas, car ils se placent au mauvais endroit, le choix des angles est le bon mais pas leur emplacement.

Exemple de fonctionnement incorrect

Exemple de fonctionnement correct

5.2.1.1 Résolution du problème

Après y avoir passé plusieurs jours entiers, je pense que le seul moyen de régler ce problème serait de revoir le fonctionnement actuel du déplacement des serpents, car la détection de l'angle à placer fonctionne mais pas l'emplacement où il doit être posé.

5.3 Problèmes rencontrés

5.3.1 Demi-tour

Le serpent pouvait faire un demi-tour sur lui-même si on appuyait les flèches directionnelles assez rapidement.

5.3.1.1 Causes du problème

Quand un joueur appuyait sur plusieurs touches pour tourner, son serpent effectuait toutes les actions à la suite en ignorant tous les tests pour savoir s'il a le droit d'y aller. Cela faisait que le serpent se passait dessus alors qu'il ne pouvait pas se toucher.

5.3.1.2 Résolution

Pour résoudre ce problème, j'ai rajouté une variable booléenne qui indique si un joueur a pressé sur une touche ou s'il n'a rien pressé. À l'appui de la touche, la variable passe sur `true` et on attend que l'action soit affichée. La variable passe en `false` et on peut donner la commande suivante et ainsi de suite.

5.3.2 Mode pause

Quand le jeu était en pause, on pouvait utiliser les touches de redémarrage de la partie et de changement de mode de jeu alors qu'on est censé les utiliser uniquement à la fin de la partie.

5.3.2.1 Causes du problème

Quand le jeu est en pause, une variable passe sur `true`, sauf que cette variable est aussi utilisée pour dire que la partie est terminée, ce qui entraîne ce bug.

5.3.2.2 Résolution

Le résoudre a été assez facile. J'ai séparé cette variable en deux, une indique si le jeu est terminé et l'autre si le jeu est en pause, j'ai aussi rajouté des tests pour qu'on ne puisse pas appuyer sur certaines touches pendant la partie.

5.4 Améliorations possibles

La liste complète des améliorations est disponible sur Github.

5.4.1 Image de fond

Une image de fond permettrait de mieux voir le contenu du plateau de jeu ainsi que de montrer au joueur si les murs du jeu sont activés ou non.

5.4.2 Pomme dorée

Les pommes dorées rapporteraient plus de point qu'une pomme normale. Son inconvénient est qu'elle augmente beaucoup la taille du serpent, ce qui rend la partie plus compliquée

5.4.3 Choix de la difficulté

Le choix de difficulté influerait sur la taille du serpent, le nombre de points gagné à chaque pomme mangée et la vitesse du jeu.

Mode	Changements
Facile	Pas de modification de vitesse, les pommes font grandir le serpent d'une case.
Moyen	Augmentation de la vitesse toutes les deux pommes, les pommes font grandir le serpent d'une case.
Difficile	Augmentation de la vitesse à chaque pomme, les pommes font grandir serpent de trois cases.

5.4.4 Multijoueur en LAN

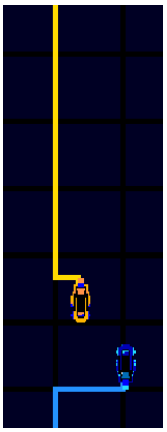
La partie se déroulerait comme en local : deux joueurs s'affrontent et celui qui a mangé le plus de pomme gagne, sauf que chaque joueur jouerait depuis son propre ordinateur.

5.4.5 Modification de la vitesse de déplacement

À chaque fois que le serpent mange une pomme, sa vitesse de déplacement augmente, ce qui rend le jeu plus compliqué.

5.4.6 Mode « Tron »

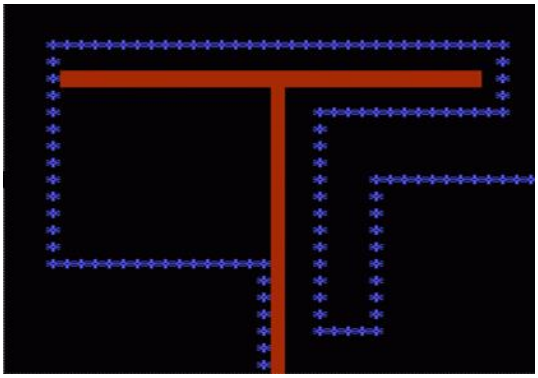
Ce mode reprend le concept de « Tron » : deux joueurs ou plus doivent se bloquer afin d'être le dernier en jeu. Ce mode de jeu n'a pas de pomme mais des boosts de vitesse qui accélèrent le joueur pendant quelques secondes. Les joueurs laissent une trainée solide de couleur derrière eux qui peut bloquer et détruire les autres joueurs ou les bloquer eux-mêmes.



Exemple de jeu Tron

5.4.7 Génération aléatoire de murs

En mode difficile, des murs se génèrent sur le plateau de jeu, empêchant le joueur de passer en ligne droite.



Exemple de Snake avec des murs

6 Ce que j'ai appris

J'ai l'impression d'avoir appris beaucoup de choses grâce à ce projet. J'ai passé la majeure partie du temps disponible à rédiger mon code et je sais maintenant pourquoi on nous demande de tout faire petit à petit. C'est une erreur de débutant à laquelle on ne me reprendra plus.

D'un point de vue plus technique, j'ai découvert comment compiler un projet Java pour en faire un exécutable, comment générer une Javadoc et comment elle se présente. Ce que j'ai le plus apprécié dans ce projet, c'est d'avoir pu mettre à contribution tout ce que nous avions vu en module et d'approfondir les connaissances que j'avais déjà.

7 Conclusion

J'ai bien aimé la liberté que l'on avait pour faire ce projet, que ce soit au niveau du jeu choisi, de son langage ou de son fonctionnement. J'aurais bien aimé pouvoir faire une partie des objectifs du chapitre « Améliorations possibles ». En effet, certains points me semblent assez intéressant à réaliser mais je n'ai malheureusement pas eu le temps de m'y atteler à cause de ma mauvaise gestion du temps et d'un bug récalcitrant de l'autre côté.

8 Sources

- https://rembound.com/articles/creating-a-snake-game-tutorial-with-html5
- https://stackoverflow.com/questions/50337303/how-do-i-change-the-speed-of-an-animationtimer-in-javafx
- https://stackoverflow.com/questions/25468882/change-color-of-background-in-javafx-canvas
- https://stackoverflow.com/questions/29962395/how-to-write-a-keylistener-for-javafx
- http://www.java2s.com/Code/Java/JavaFX/SetScenebackgroundcolorandsize.htm
- https://www.tutorialspoint.com/javafx/javafx_text.htm#:~:text=By%20default%2C%20the%20text%20created,Font%20of%20the%20package%20javafx.