# 

**Mini projet GEN 2018**

**Snake Multi-joueurs**

Tables des matières

[Description Générale: 4](#_Toc512590506)

[Déroulement d’une partie: 4](#_Toc512590507)

[Contrôles: 5](#_Toc512590508)

[Détails Supplémentaires: 5](#_Toc512590509)

[Création de maps (à voir): 6](#_Toc512590510)

[Limites / Contraintes: 6](#_Toc512590511)

[Possibles futures implémentations (si le temps le permet): 6](#_Toc512590512)

[Principales interfaces utilisateurs: 7](#_Toc512590513)

[Diagramme de cas d’utilisation général: 8](#_Toc512590514)

[Cas d’utilisation : 8](#_Toc512590515)

[Rôles (utilisateurs) 9](#_Toc512590516)

[Rôles (de l’équipe) : 9](#_Toc512590517)

[Ébauche modèle du domaine 10](#_Toc512590518)

[Modèle conceptuel de données : 10](#_Toc512590519)

[Responsabilités client/serveur 11](#_Toc512590520)

[Protocoles 12](#_Toc512590521)

[Backlog : 14](#_Toc512590522)

[Sprints : 15](#_Toc512590523)

[Sprint N°1 15](#_Toc512590524)

[Sprint N°2 16](#_Toc512590525)

[Sprint N°3 16](#_Toc512590526)

[Sprint N°4 17](#_Toc512590527)

[Sprint N°5 17](#_Toc512590528)

[Sprint N°6 18](#_Toc512590529)

[Sprint N°7 18](#_Toc512590530)

## Description Générale:

Le but de ce projet est de créer un jeu se basant sur le principe du célèbre jeu du *Snake*. Contrairement au modèle original, cette version du jeu est multi-joueurs (2 à 4 joueurs) et fonctionne sur un modèle client-serveur.

Chacun des joueurs incarne un serpent et a pour but d’être le dernier survivant. Le monde dans lequel les serpents se déplacent s’appelle une *map*. Les *maps* ne possèdent pas de bordures. Un serpent dépassant les limites de la map, réapparaîtra de l’autre côté et pourra continuer son chemin. Les maps peuvent toutefois posséder un ou plusieurs obstacles: des murs.

Un serpent meurt s’il touche un obstacle, un autre serpent, ou sa propre queue. En cas de choc frontal avec un autre serpent, le plus long des deux survivra. Si les deux serpents sont de longueur identique, il y a égalité.

## Déroulement d’une partie:

Création d’une partie:

Lorsqu’un joueur décide de créer une partie, un salon d’attente (*lobby*) est créé. Ce joueur peut y inviter d’autres joueurs, jusqu’à être un total de 4 joueurs au maximum. Dès qu’au moins 2 joueurs sont présent dans le salon, ils peuvent indiquer qu’ils sont prêt à jouer. Si tous les joueurs du salon sont prêts, la partie débute après un décompte de 5 secondes. Le salon d’attente disparaît pour laisser la place à la map avec tous les serpents.

Déroulement de la partie:

Une partie se déroule en plusieurs manches.

Au début de chaque manche, tous les joueurs possèdent un serpent de taille minimale. Chaque serpent débutera la partie sur des coins différents de la *map*. Afin de piéger plus facilement les autres serpents, le joueur peut manger des pommes qui apparaissent de manière aléatoire sur la surface de la *map*, et ainsi faire grandir le serpent. En effet, avec un serpent de grande taille, il sera plus facile d’encercler ses adversaires.

Pour rendre le jeu plus dynamique, des bonus ou malus, tels que modification de vitesse, modification de taille ou encore invincibilité, apparaissent également aléatoirement au travers de la map et doivent être mangés par les serpents de la même manière que les pommes.

Fin de la manche:

Une manche se termine lorsqu’il ne reste plus qu’un serpent en vie. Les joueurs reçoivent alors des points en fonction du moment de leur mort: 1er mort (0 point), 2ème mort (1 point), 3ème mort (3 points), dernier survivant (5 points). Cet attribution de points reste sujet à modification en fonction des tests futurs. Une égalité peut avoir lieu en cas de choc frontal des deux dernier joueur en vie, pour autant que les deux serpents sont de même longueur. Autrement, le serpent le plus grand l’emporte.

Fin de la partie:

À la fin de la partie, les scores de toutes les manches sont additionnés afin de désigner un grand vainqueur. Les joueurs se retrouvent alors dans un nouveau salon pour commencer une nouvelle partie.

## Contrôles:

La direction du serpent est contrôlée par les touches fléchées du clavier: gauche et droite.

## Détails Supplémentaires:

Les scores et classement de la partie en cours sont affichés sur un des côtés de la fenêtre.

Afin de différencier un joueur de ses adversaires, chaque serpent se verra attribuer une couleur. La couleur du serpent correspond à la couleur du pseudonyme (dans la section scores et classement de la partie en cours) du joueur qui le contrôle.

La longueur des serpents au lancement de la partie sera déterminée après avoir effectué des tests sur une version jouable du jeu. Quant à leur taille maximale, elle n’est pas limitée.

La création d’un compte et une authentification seront nécessaires afin de jouer (ou de créer une nouvelle map). Ces comptes servent à différencier chaque joueur et à garder une trace de leurs classements et victoires. Une page avec un classement général sera disponible dans le futur.

Les consommables (pommes, bonus, malus) seront ingérés par les serpents au niveau de leur tête.

Les conséquences exactes de tous les bonus et malus différents ne sont pas encore définies. À noter que ces consommables ne peuvent pas être générés sur un serpent ou sur un obstacle. Ils seront toujours générés dans un espace vide.

Une manière de casser le jeu serait de tourner en rond en ne laissant aucun espace entre la tête et la queue du serpent. Ainsi, aucun obstacles ou adversaires ne pourra se mettre en travers. Pour pallier à ce problème, tous les serpents, dans un intervalle de temps aléatoire, laisseront derrière eux une petite crotte. Cette crotte est considérée comme un obstacle. Aucun joueur ne pourra donc tourner en rond sans se faire prendre à son propre jeu.

Dans le but de rendre les parties plus dynamiques, chaque pomme mangée rapportera 1 point au joueur.

Les administrateurs devront aussi se connecter avant de pouvoir effectuer une action.

## Création de maps (à voir):

Le client offre la possibilité à n’importe quel joueur de créer sa propre *map* grâce à l’outil de création et de la mettre en ligne, disponible aux autres joueurs.

L’outil de création permet uniquement de créer et d’ajouter ses propres obstacles sur une *map* vide. Il se présente sous la forme d’une grille remplissable d’obstacles cases par cases. Pour que chaque partie puisse débuter sans problèmes, l’ajout d'obstacles dans les zones (quelques cases) proches des coins de la *map* est interdit.

Au moment de l’enregistrement de la *map* créée, elle est directement mise sur le serveur et est disponible dans la liste de toutes les *maps* créées par tous les joueurs. Cette liste classe les *maps* en fonction des notes que les joueurs leur ont attribué. Ainsi, une *map* de mauvaise qualité apparaîtra à la fin de la liste et sera, par conséquent, moins visible et moins choisie que les *maps* de bonne qualité. Les joueurs auront la possibilité d’assigner une note à la *map,* allant de 1 à 5, a la fin de leur partie sur celle-ci.

## Limites / Contraintes:

* Le nombre maximum de joueurs dans une même partie est de 4.
* Le serpent peut uniquement se déplacer sur quatre directions (haut, bas, gauche, droite).

## Possibles futures implémentations (si le temps le permet):

Toujours dans le but de rendre les parties le plus dynamique possible, la vitesse de tous les serpents pourrait croître en fonction du temps écoulé de la manche en cours.

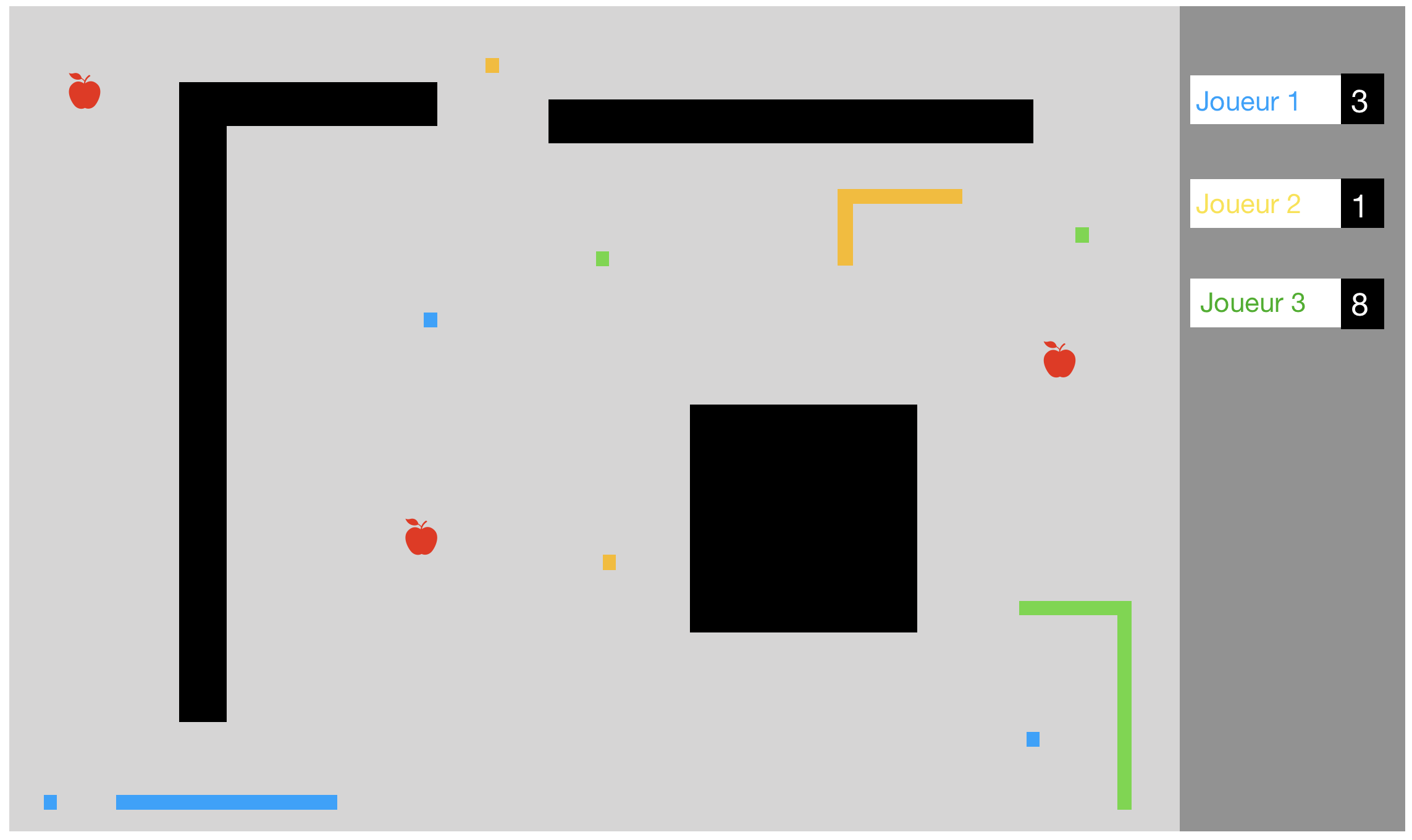
L’ajout de nouvelles variétés de bonus et malus.

L’outil de création de *map* est une idée audacieuse mais ne fait pas partie des éléments essentiel au jeu et sera, par conséquent, considéré avec une priorité plus faible.

Enfin, le dernier aspect envisageable, si le temps le permet, est la création d’une IA pour pouvoir jouer seul.

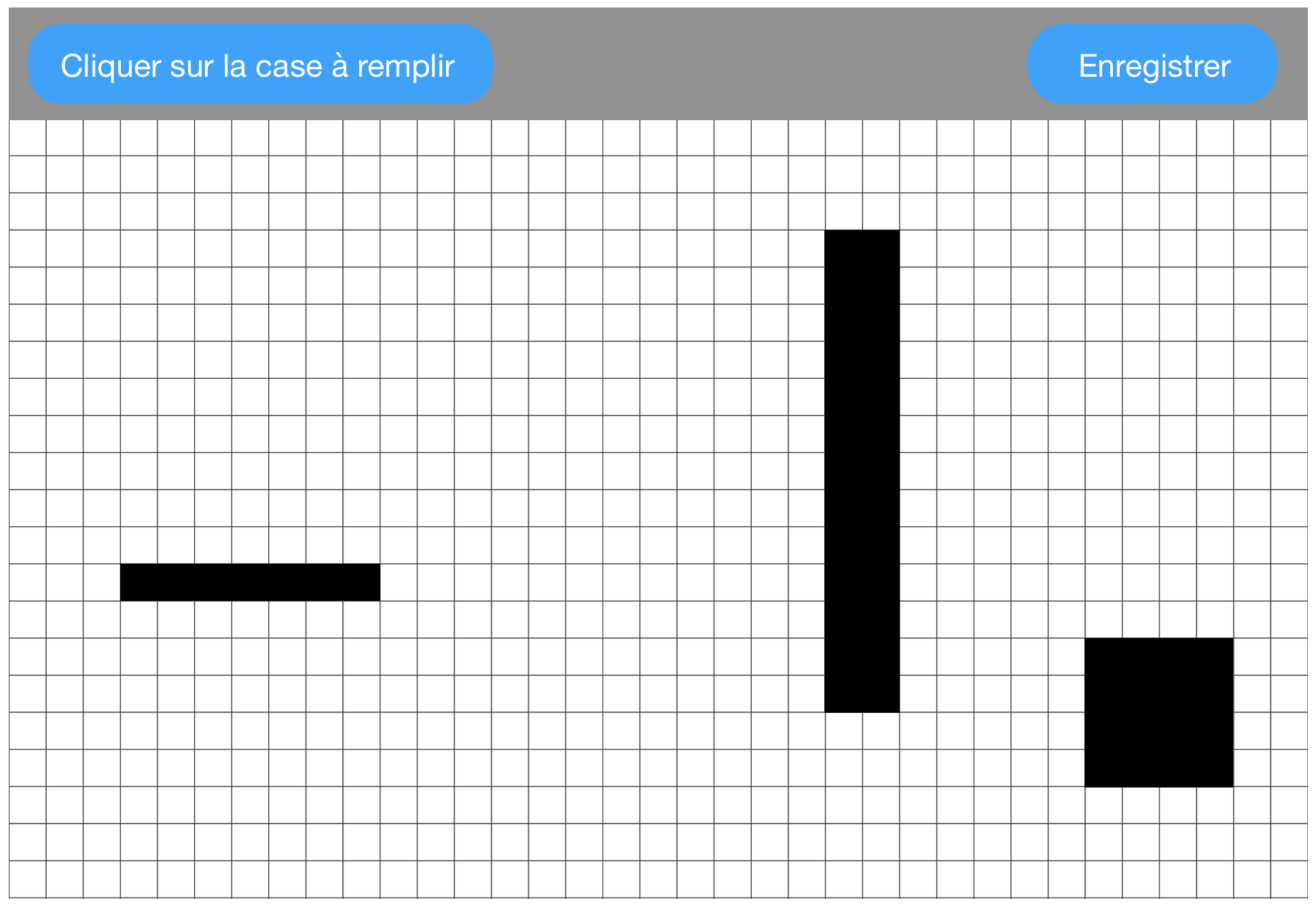
## Principales interfaces utilisateurs:

Interface d’une partie :



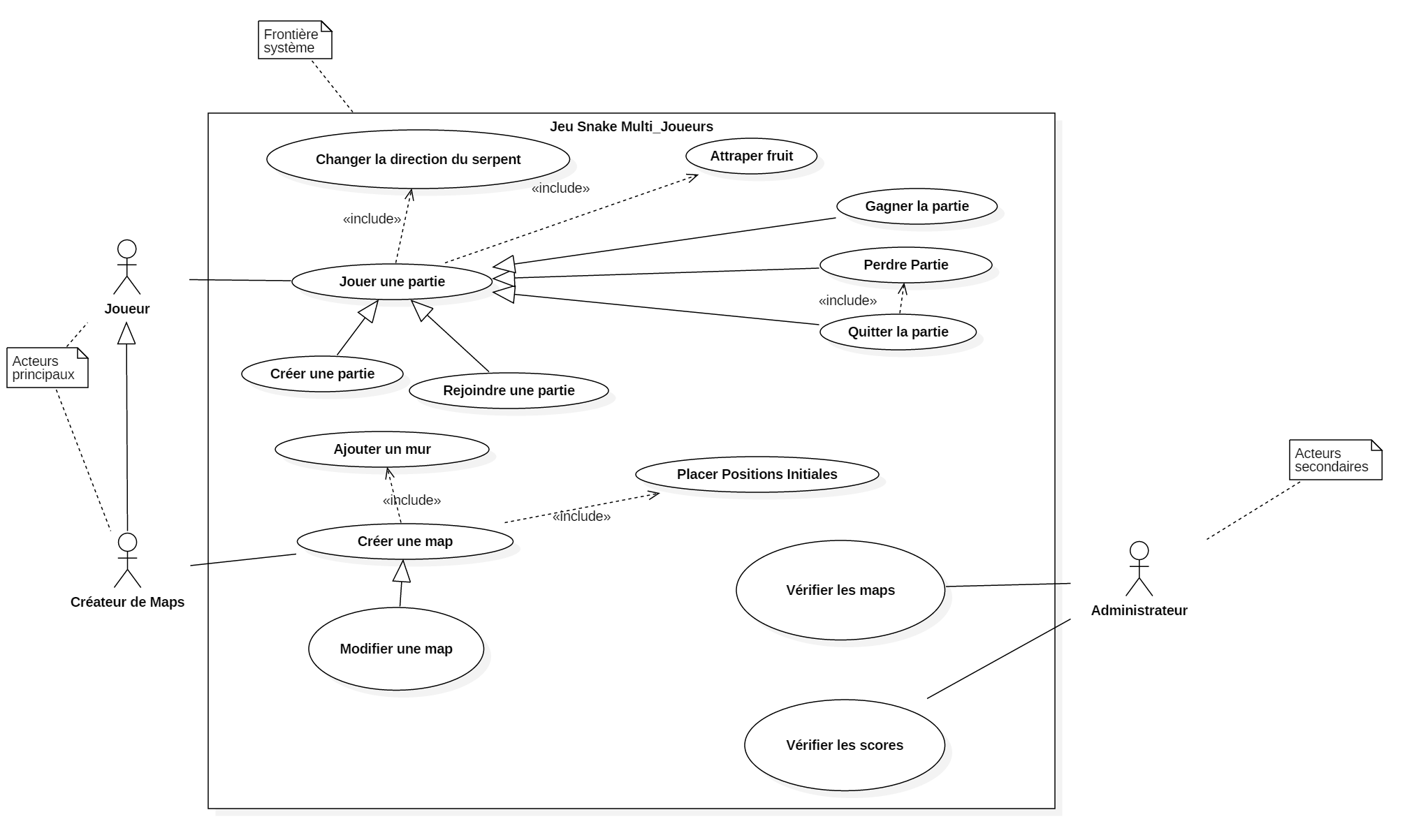
*Une partie, fig.1*

Interface pour la conception de carte :



*Outil de création de maps, fig. 2*

## Diagramme de cas d’utilisation général:

****

## Cas d’utilisation :

- **Jouer Partie** : c’est le cas d’utilisation central pour notre application. Il inclut tous les cas d’utilisation délégués lors du déroulement. Le joueur lance une partie, après s’être authentifié.

- **Changer la direction du serpent** : le joueur peut effectuer deux types de mouvements lors de la partie droite ou gauche (suivant sa position ce le fera aller en haut, en bas, à gauche ou à droite).

- **Attraper fruit** : le but du jeu est d’attraper un maximum de fruits. Lorsque le joueur y parvient, le gestionnaire de l’application met à jour le score du joueur.

- **Gagner la partie** : Le joueur est dirigé vers un écran de scores et les scores sont sauvegardé dans une liste.

- **Quitter la partie** : le joueur peut à tout moment quitter la partie en cours. Il perdra automatiquement.

- **Perdre partie** : lorsqu’un joueur touche un obstacle ou le serpent de l’adversaire, il perd la partie.

- **Créer une partie :** Le joueur peut créer une partie, pour cela il choisit une map et propose sa partie à d’autre joueur.

- **Rejoindre une partie :** Le joueur peut rejoindre une partie proposé par un autre joueur.

- **Créer une Map** : s’il le désire, un joueur peut créer une map de jeu personnalisée. Pour ce faire, il peut placer des murs, ainsi que les positions initiales des participants(il y aura des contraintes, on ne peut pas commencer n’importe où. Un périmètre de position possible sera affiché ). Il doit au préalable s’authentifier auprès du système.Il pourra enregistrer ses maps, qui sera ajouté à une liste possible de map “jouable”.

- **Modifier une map :** Si la map lui appartient un joueur peut modifier la map.

- **Ajouter un mur** : le créateur peut “dessiner” les différents mur de la map.

- **Placer Positions initiales** : le créateur peut déterminer les positions initiales de chaque concurrent.

- **Vérifier les maps :** L’administrateur vérifie parmis la liste des maps, s’il y a des maps inadéquates et les supprimes le cas échéant.

- **Vérifier les scores :** L’administrateur vérifie parmis la liste des scores, si des scores semblent invraisemblable et les supprimes le cas échéant.

## Rôles (utilisateurs)

**Acteurs principaux :**

Joueur : Il pourra jouer une partie ou créer une map mais pas les deux en même temps.

Créateur de maps: est un joueur (en train de créer une map)

**Acteur secondaires:**

Administrateur : Il aura pour rôle de vérifier les maps (pas de dessins inapproprié). Et vérifier le tableau des scores (éviter qu’il y ait des scores trafiqués).

## Rôles (de l’équipe) :

Scrum Master : Mathieu Jee

Product Owner : Guillaume Blanco

Développeurs & Testeurs : Patrick Neto & Lionel Burgbacher

## Ébauche modèle du domaine

modèle pour le serveur

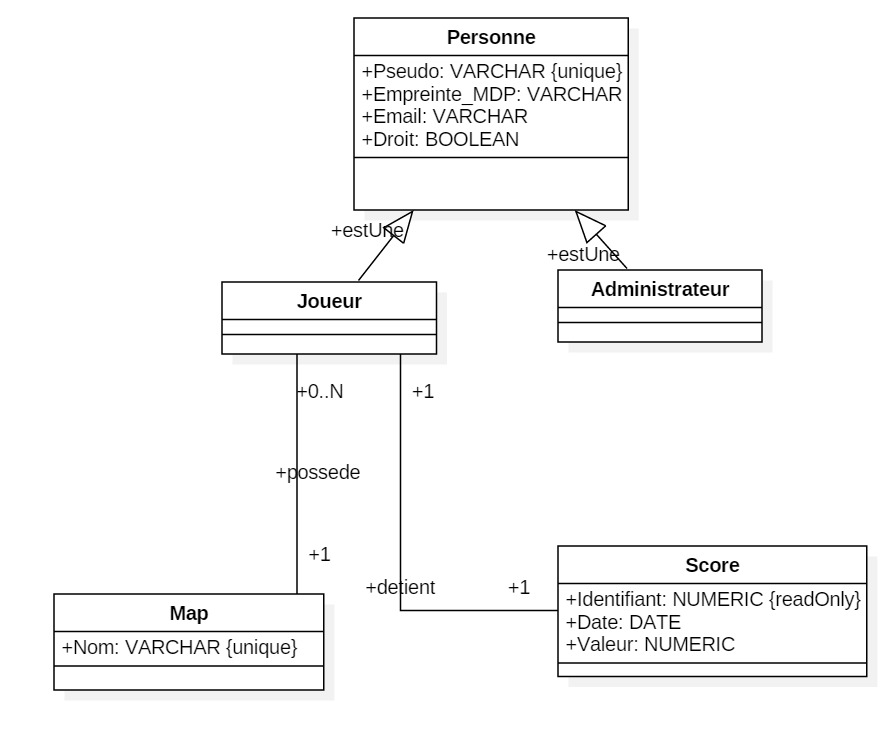
* serpent (joueur)
* position
* map
* consommable

modèle pour le client

* serpent (joueur)
* position
* éditeur de map
* plateau (surface de jeu)
* obstacle

## Modèle conceptuel de données :

Le but de notre base de données est de pouvoir permettre aux utilisateurs de se créer un compte qui gardera en mémoire les différentes *maps* qu’ils ont créé, leur score et certaines informations personnelles ( pseudo, mdp, etc…). Il y aura aussi un booléen qui indiquera si la personne est administrateur ou simple joueur.

****

**Etude préliminaire**

Pour ce projet nous avons décidé de le réaliser en Java. Les clients ainsi que le serveur seront tous les deux codés en Java. Nous utiliserons le protocole UDP car il a été étudié en classe et qu’il est approprié au application en temps réel. L’implémentation de clients et de serveurs en Java ont tous deux été étudié en classe (RES). Nous utiliserons les mécanismes de synchronisation étudiés en cours de GEN pour synchroniser les différents serpent des joueurs. Pour la partie graphique nous utiliserons Java FX. Après quelques recherches nous avons trouvé un outil très utile(Java FX Scene Builder) qui facilite grandement la partie graphique. Comme tous les outils que nous allons utiliser pour la réalisation de notre projet ont déjà été étudié, nous considérons qu’il ne devrait pas avoir de problème quant à sa réalisation.

Pour ce qui est de la base de données, nous avons, là aussi, décidé de se tourner vers une technologie déjà acquise. Nous utiliserons MySQL car nous le connaissons plutôt bien et nous jugeons inutile de se tourner vers des technologie “ultra-rapide” tel que mongoDB car ce projet n’est pas destiné à être utilisé par des millions d’utilisateur et l’accès à la DB sera nécessaire uniquement pour l’authentification des utilisateurs et l’affichage des classements généraux.

## Responsabilités client/serveur

Voici les fonctionnalités et responsabilités gérées par le client et le serveur. Les éléments apparaissant entre parenthèse font parties des possibles implémentations futures. Ils seront développés uniquement si le temps le permet.

**Client:**

* Dessine la *map*
* Envoie la nouvelle position du serpent au serveur
* Met à jour la *map* (pomme/bonus/malus) et la position de tous les serpents
* Met à jour les états des serpents (mort, croissance, vitesse, …)
* (Création de map grâce à l’outil)

**Serveur:**

* Gère plusieurs clients en même temps
* Calcul et conservation des positions des serpents
* Envoi des nouvelles positions des serpents aux clients
* Fait apparaître de temps en temps des bonus/malus
  + Calculs des positions des bonus/malus
* Vérification de la nouvelle position envoyée par le client
  + Calcul des conséquences du déplacement (pomme/bonus/malus/mort)
* Scores calculés et conservés sur serveur
* (Classement général: stocké en DB)
* maps stockées en DB
* Authentification au lancement du jeu avec un nom de compte et un mot de passe
  + Ces credentials sont stockés en DB
* Lancement de la partie
* Fin de la partie

## Protocoles

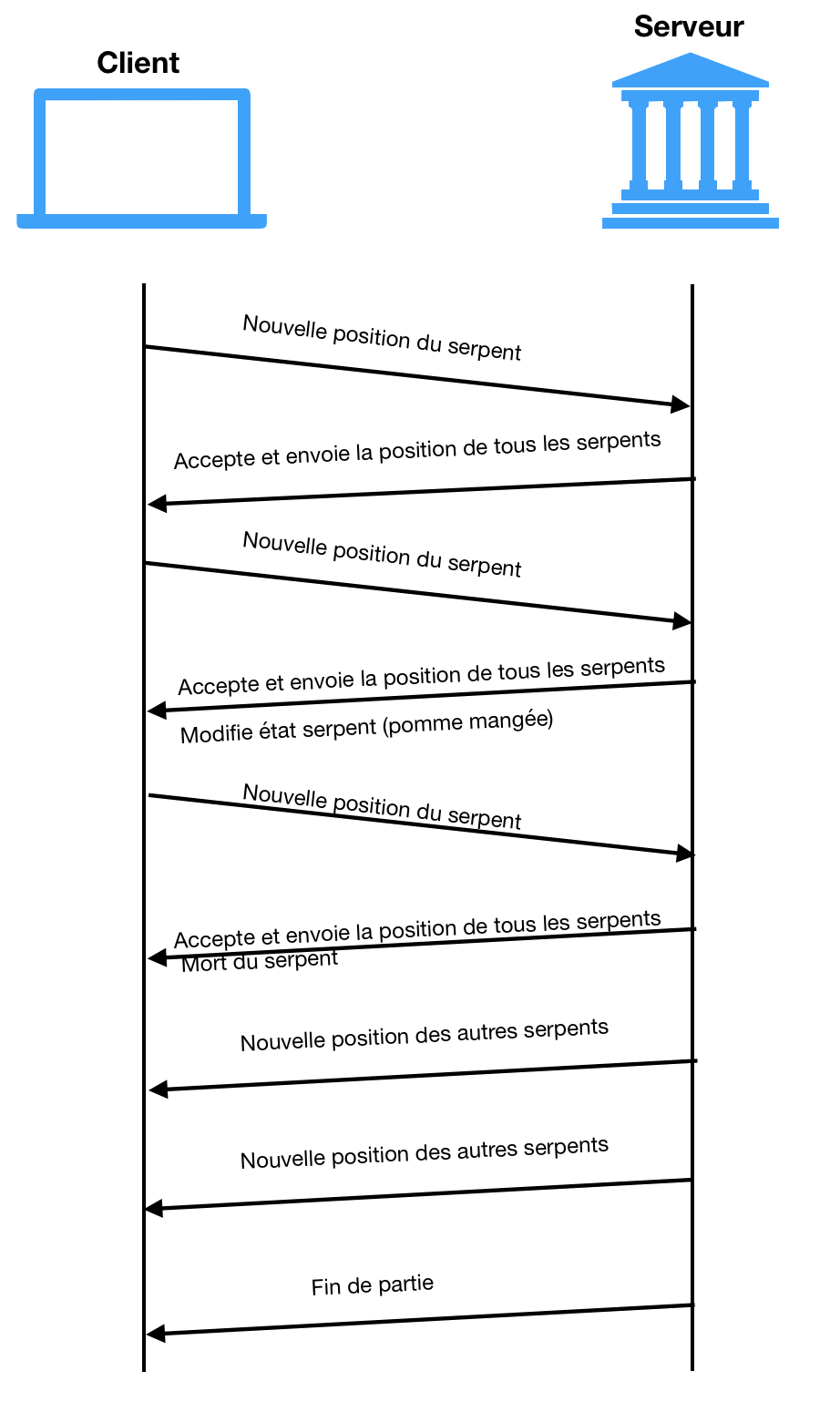
Le serveur envoie le taux de rafraîchissement défini au client. Ce taux mesure le nombre de déplacements possibles du serpent par seconde.

À chaque *tick* (rafraîchissement), le client calcul la nouvelle position du serpent, en fonction des choix de l’utilisateur et l’envoie au serveur.

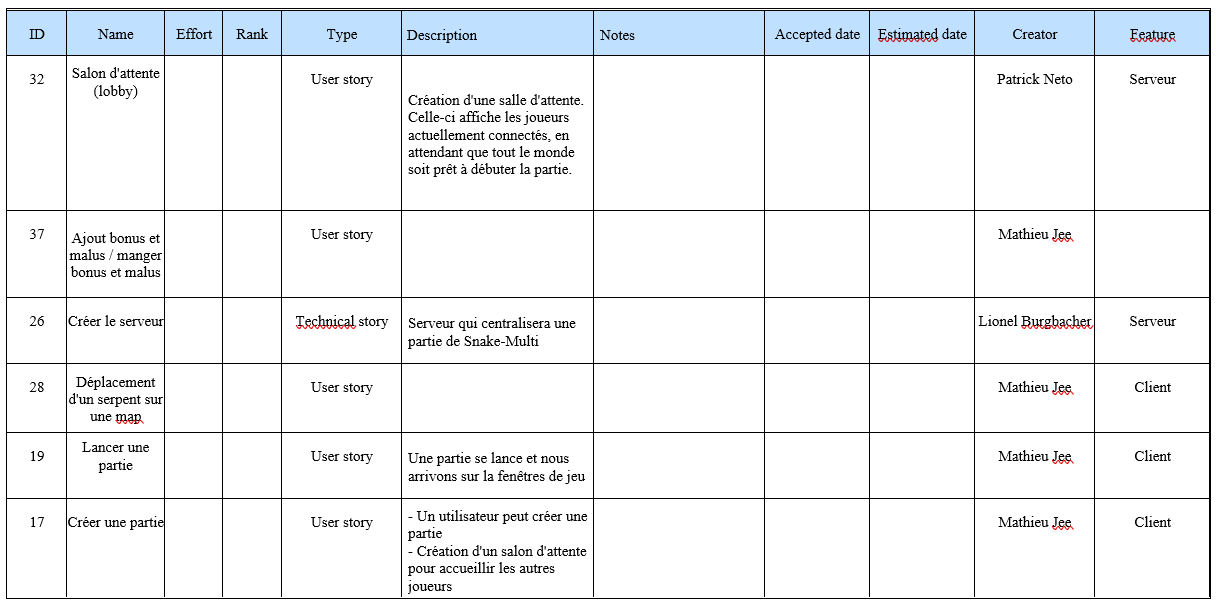
Le serveur s’assure de la pertinence du déplacement avant de le valider. Une fois les déplacements de tous les joueurs considérés comme valides, le serveur notifie tous les clients des nouvelles positions de l’ensemble des serpents de la partie. Les clients de tous les joueurs peuvent donc mettre à jour les différentes positions des serpents. En cas d’un déplacement interdit de la part d’un joueur (problème de communication, triche, modification du code source, etc.), le serveur ignore la nouvelle position fournie par le client concerné et fait comme si le joueur n’avait appuyé sur aucune touche (le serpent continue tout droit). Cet échange entre le client et le serveur aura lieu à chaque *tick*.

Le serveur se charge également de calculer les positions des éventuels pommes, bonus ou malus et les envoie aux clients. Ainsi, si le serveur reçoit une position d’un serpent qui superpose celle d’une pomme (par exemple), il enverra la nouvelle position aux autres clients avec le changement d’état (changement de taille du serpent) et la pomme mangée (elle disparaît).

Si le serveur calcule une mort d’un des serpents, le client du joueur concerné continuera à recevoir les mise à jour des éléments de la part du serveur, malgré le fait que son serpent soit mort. Cela permet de suivre la fin de la manche en tant que spectateur.

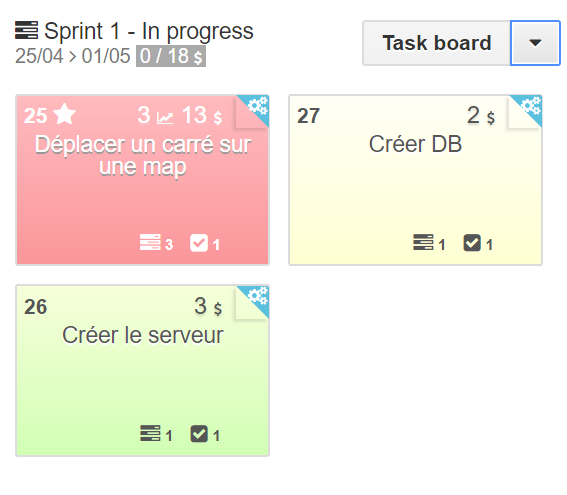


## Backlog :

****

## Sprints :

### Sprint N°1



1 : déplacer un carré sur une map

2 : Créer une base de données

3 : Créer le serveur (de synchronisation des serpents)

Ce sprint est important car il nous permettra de montrer quelque chose au client pour qu’il puisse y voir une avancé et nous permettra aussi de mettre en place une base à notre projet

### Sprint N°2



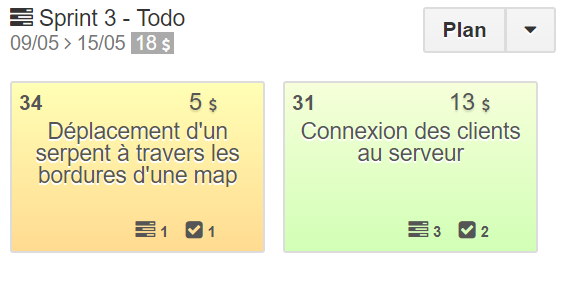
1 : Déplacement d’un serpent sur une map

2 : Authentification

3 : Connexion au jeu

Ce sprint là nous permettra de présenté une avancée significative de notre projet au client. En effet, si le client peut déjà voir qu’un serpent peut se déplacer, il sera confiant pour la suite. Parallèlement les deux autres histoires nous permettent d’avancer dans la réalisation du projet.

### Sprint N°3



1 : Déplacement d’un serpent à travers les bordures d’une map

2 : Connexion des clients au serveur

Sans doute un des sprints qui va nous prendre le plus de temps et un des plus important. Il comprend le calcul des déplacements pour pouvoir traverser les murs ainsi que la partie la plus dur du développement, la synchronisation des serpents avec le serveur.

### Sprint N°4



1 : Calculs des positions

2 : Mort d’un serpent

Dans la continuité du sprint n°3, ce sprint nous permettra d’avancer dans notre jeu et de pouvoir montrer un jeu déjà « fonctionnel » à notre client. En effet, le client pourra déjà jouer (bouger son serpent, rentrer dans un autre serpent et mourir).

### Sprint N°5



1 : Ajout d’une pomme / manger une pomme

2 : Salon d’attente (lobby)

Ce sprint impliquera la modification de la taille du serpent et donc une grande partie graphique qui nécessitera pas mal de jours/hommes. Ensuite la deuxième partie n’aura pas une très grande valeur pour le client mais demandera aussi pas mal de moyen et de synchronisation.

### Sprint N°6



1 : Ajout de bonus et malus / manger bonus et malus

2 : Fin de la partie

3 : Créer une partie

4 : Inviter des joueurs

5 : Lancer une partie

Ce sprint est le plus gros en terme d’investissement de temps et contient tous ce qu’il nous reste à faire pour que le jeu soit complétement opérationnel. Il coïncide avec une période où nous auront plus de temps d’où notre choix de le placer avant le dernier sprint.

### Sprint N°7



1 : Documentation

2 : Présentation

Le sprint final, le jeu devrait être complétement à ce moment-là. Nous ne devrions plus qu’à avoir à finir la documentation et nous préparer à présenter notre projet. Peut-être nous restera-t-il quelques choses à fignoler mais le gros du travail devrait être finit à ce moment-là.

|  |  |  |
| --- | --- | --- |
| Version | Date | Ce qui a été fait |
| 1 | 25.04.2018 | Le rapport intermédiaire a été rendu |
| 2 | 27.04.2018 | Le rapport intermédiaire a été modifié selon des demandes du professeur (ajout de backlog et de sprint). |
|  |  |  |
|  |  |  |