Frogger

Tony Clavien, Maxime Guillod, Gabriel Luthier & Guillaume Milani

11 avril 2017

Table des matières

Table des figures

Liste des tableaux

1 Fonctionnement général

1.1 Objectifs de base

Le but de ce projet est de réaliser une application client-serveur dans le cadre du cours GEN. Nous avons décidé de proposer un jeu de type « Frogger » en version valaisanne.

Le but du jeu sera de faire descendre une piste de ski à un Valaisan. Il devra éviter les skieurs genevois la traversant la piste à vive allure.

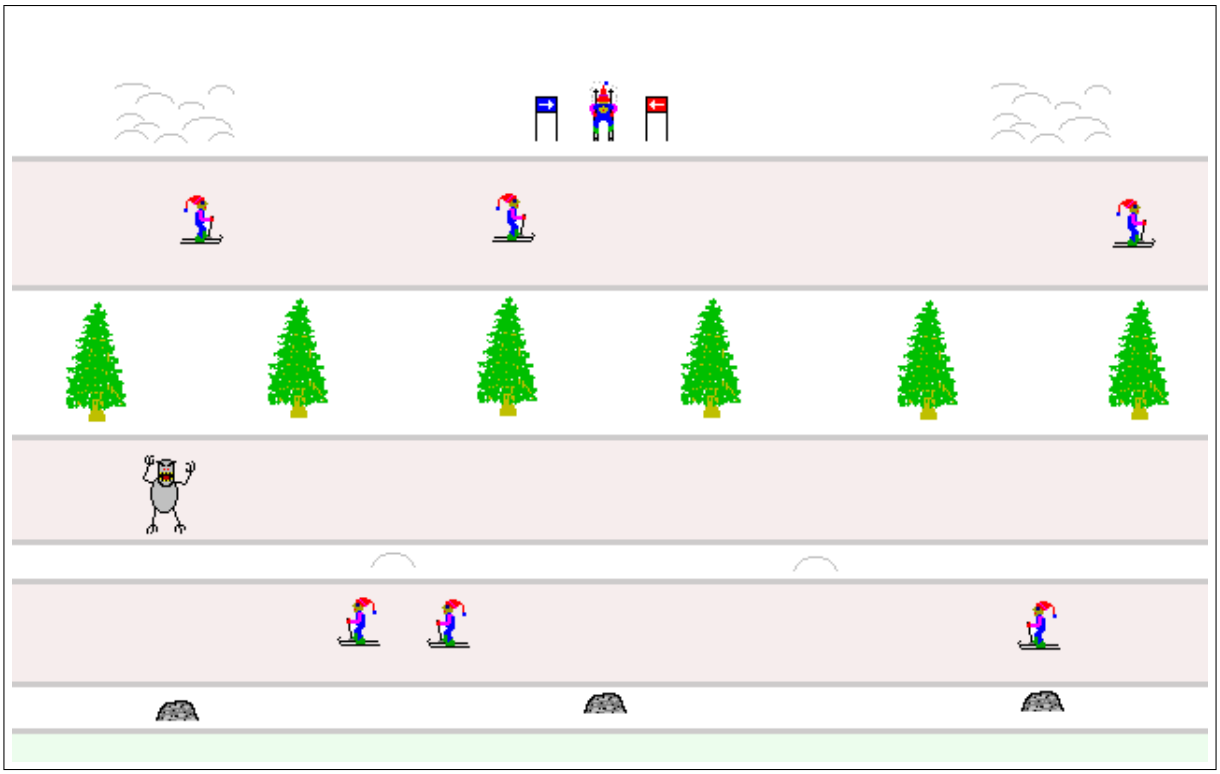

FIGURE 1 – Maquette d'une partie

Les zones rouges sont les zones à disposition du joueur 2 pour placer les obstacles, les sapins sont des obstacles fixes. Le joueur 1 gagne s'il atteint la zone en vert sans heurter d'obstacle.

1.2 Utilisation de l'applicatif

Le joueur 1 contrôle son personnage à l'aide des flèches du clavier (←↓→) et tente d'éviter les obstacles. Le joueur 2 place les obstacles dans les colonnes, ceci fait que l'obstacle se met en mouvement dans la colonne. Une bouteille de Fendant^{TM}se vide au fur et à mesure que le joueur 2 place des obstacles. Il doit ensuite attendre qu'elle se remplisse à nouveau pour placer de nouveaux obstacles.

L'administrateur peut régler les paramètres du jeu (« skin » du jeu, vitesse des obstacles, vitesse de rechargement de la bouteille de Fendant^{TM}). L'administrateur est un rôle hérité de celui de joueur (un administrateur est donc avant tout un joueur). Ce droit est stocké dans la base de données.

1.3 Règles du jeu

Pour le joueur 1 : le but est d'arriver en bas de la piste sans avoir heurté d'obstacle.
Pour le joueur 2 : le but est que le joueur 1 heurte un obstacle.

1.4 Contraintes

Le joueur 1 ne peut pas revenir en arrière, une fois descendu une portion de piste il peut s'arrêter, changer de «colonne» ou de continuer à descendre.

Le joueur 2 peut envoyer des obstacles pour autant que la bouteille de Fendant^{TM}ne soit pas vide.

1.5 Priorités de développement

1. Première version du jeu «standalone» avec un seul joueur qui prend le rôle du joueur 1 (descendre la piste). Les obstacles sont envoyés aléatoirement. (En parallèle, développement du serveur et de l'API).
2. Ajout du deuxième joueur en local (les deux jouent sur la même machine avec des touches différentes du clavier).
3. Les joueurs jouent chacun sur leur machine et se connectent à un serveur distant centralisant la partie. Lorsqu'un joueur se connecte, il voit la liste des parties crées (soit en attente d'un joueur, soit en cours). Il peut soit rejoindre une partie en attente d'un joueur soit créer une nouvelle partie.
4. L'administrateur peut configurer le serveur (paramètres des parties).

1.6 Base de données

La base de données stocke et partage les informations suivantes entre les joueurs et l'administrateur :

- Informations des joueurs (nom d'utilisateur, mot de passe, rôle, points totaux gagnés)
- Informations des parties (joueurs, points remportés par chacun)
- Paramètres de l'application (ce qui peut être configuré par l'administrateur)
- Logs de l'application

2 Partage des responsabilité serveur — client

La figure 2 représente le diagramme d'activité qui illustre la répartition des responsabilités entre le client et le serveur.

3 Rôle des participants

Joueurs : 2 personnes par partie, l'un ayant pour but de réussir à descendre la piste, tandis que l'autre l'en empêcher.

Administrateur : Peut changer les règles du jeu, et s'occuper des données (nettoyer, modifier des joueurs, etc...)

4 Cas d'utilisation

4.1 Acteurs

- Joueurs, dans une partie nommés : *Attaquant* et *Défenseur*
- Administrateur

4.2 Scénario principal (succès)

1. L'un des deux joueurs se connecte au lobby et crée une partie en spécifiant les paramètres qu'il souhaite.
   (a) Difficulté
   (b) Taille de la carte
   (c) Son rôle (*Défsenseur* ou *Attaquant*)
2. Le second joueur rejoint la partie libre du joueur 1 dans le lobby et prends le rôle restant.
3. Lorsque les 2 joueurs sont prêts la partie commence.
   (a) "L'attaquant" tente de traverser la piste en déplaçant son personnage de haut en bas.
   (b) "Le défenseur" tente de l'en empêcher en envoyant des obstacles de gauche à droite.
   (c) La partie se termine si l'attaquant n'a plus de vie ou si il a atteint l'autre côté.
4. À la fin de la partie, soit les rôles s'inversent et on recommence une partie à l'étape 3, soit les joueurs quittent la partie. Cette dernière est donc interrompue et les joueurs retournent au lobby.

4.3 Extensions (ou scénarios alternatifs)

Pour permettre la récupération des parties de manière correct, il faut s'assurer que tous les états et les événements sensibles du système peuvent être récupérés à n'importe quelle étape du scénario.

Lors d'une partie, si l'un des deux joueurs est déconnecté, l'adversaire gagne automatiquement la partie et est renvoyé au lobby.

4.4 Scénario d'administration

1. l'administrateur se connecte à l'interface.
2. Il modifie l'une des options à sa disposition :
   (a) Les paramètres du jeu.
   (b) Les données d'un joueur.
3. puis il peut se déconnecte ou recommencer à l'étape 2.

5 Protocole d'échange entre le client et le serveur

5.1 Communication

Nous allons utiliser JSON[1] comme protocole d'échange entre le client et le serveur.
En effet, de part la structure de son contenu, l'échange d'information entre les deux intervenants est très facilement sérialisable/désérialisable, sans compter sur le fait que le contenu est lisible par un individu (contrairement à un contenu binaire).

Commande	**Processus effectué par le serveur**	**Réponse**
{"command":"login", "param":{"user":"toto", "password":"hash(1234)"}}	Vérification du hash de mot de passe	{"token":"as73dhadc&7"} (réussite) ou {"token":null} (échec)

Table 1 – Protocole de communication Client / Serveur

5.2 Données échangées

6 Ébauche du modèle de domaine

7 Base de donnée

L'objectif de la base de données est principalement de stocker les paramètres du jeu et les données utilisateurs. Nous avons fait le choix de ne pas stocker d'informations concernant les parties en cours qui seront chargées uniquement dans la mémoire du serveur.

On trouve donc une entité `User` qui peut être `Administrator`, dont on stocke le login et le nombre de victoires. `Settings` modélise les réglages généraux du jeu, `MapSize` permettra à l'utilisateur qui créé une partie de choisir parmis plusieurs tailles de carte. `DifficultyLevel` modélise les niveaux de difficultés parmi lesquels l'utilisateur créant la partie aura le choix.

1. JSON : JavaScript Object Notation

8 Plan d'itération

1. Création de l'interface de base
2. Ajout du 1er joueur avec les obstacles
3. Ajout du 2ème joueur en local
4. Mise en place des logiques de jeu
   - Conditions de départ
   - Conditions de victoire
   - Barre de recharge pour la pose d'obstacle
5. Ajout de la communication avec le serveur
6. Application administrateur
7. Ajout du Lobby
8. Finalisation

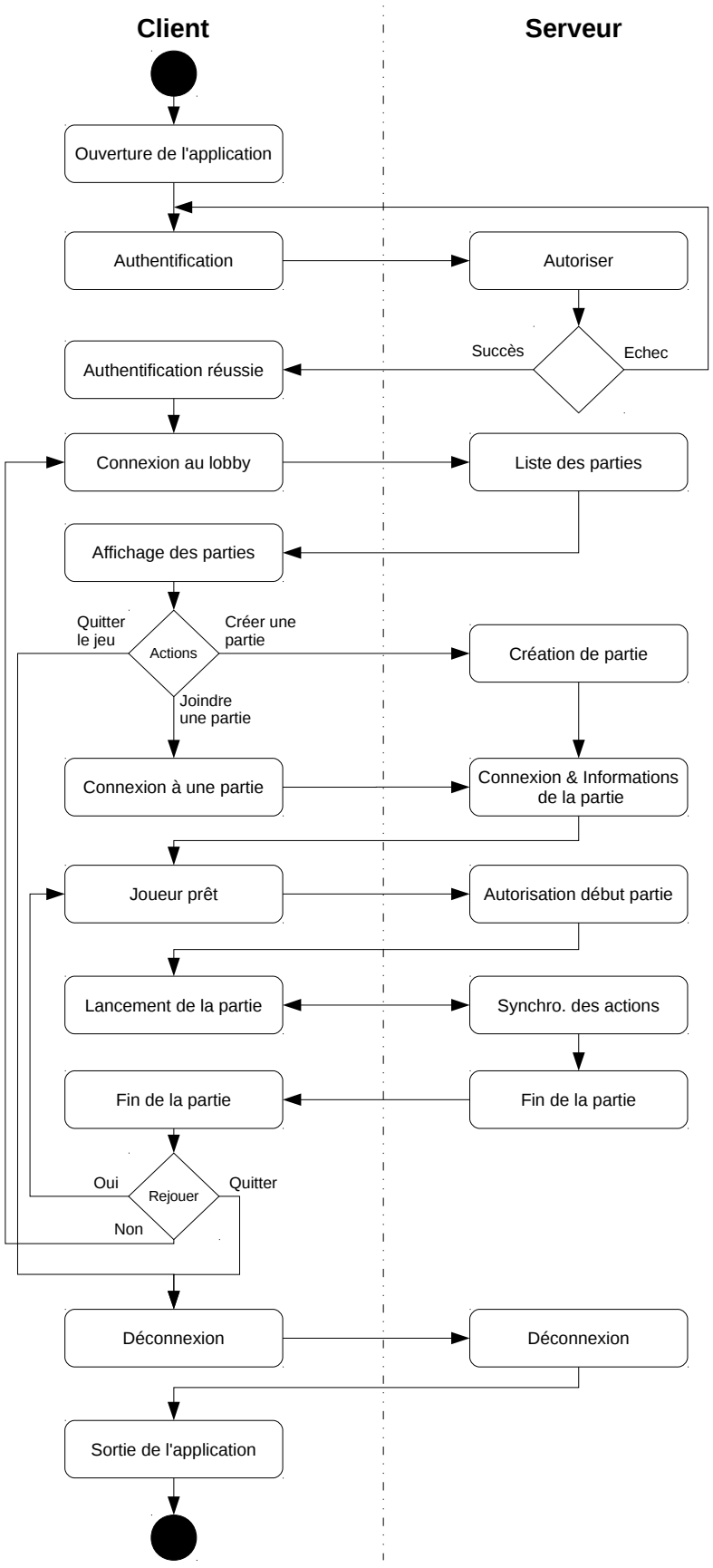

FIGURE 2 – Diagramme d'activité

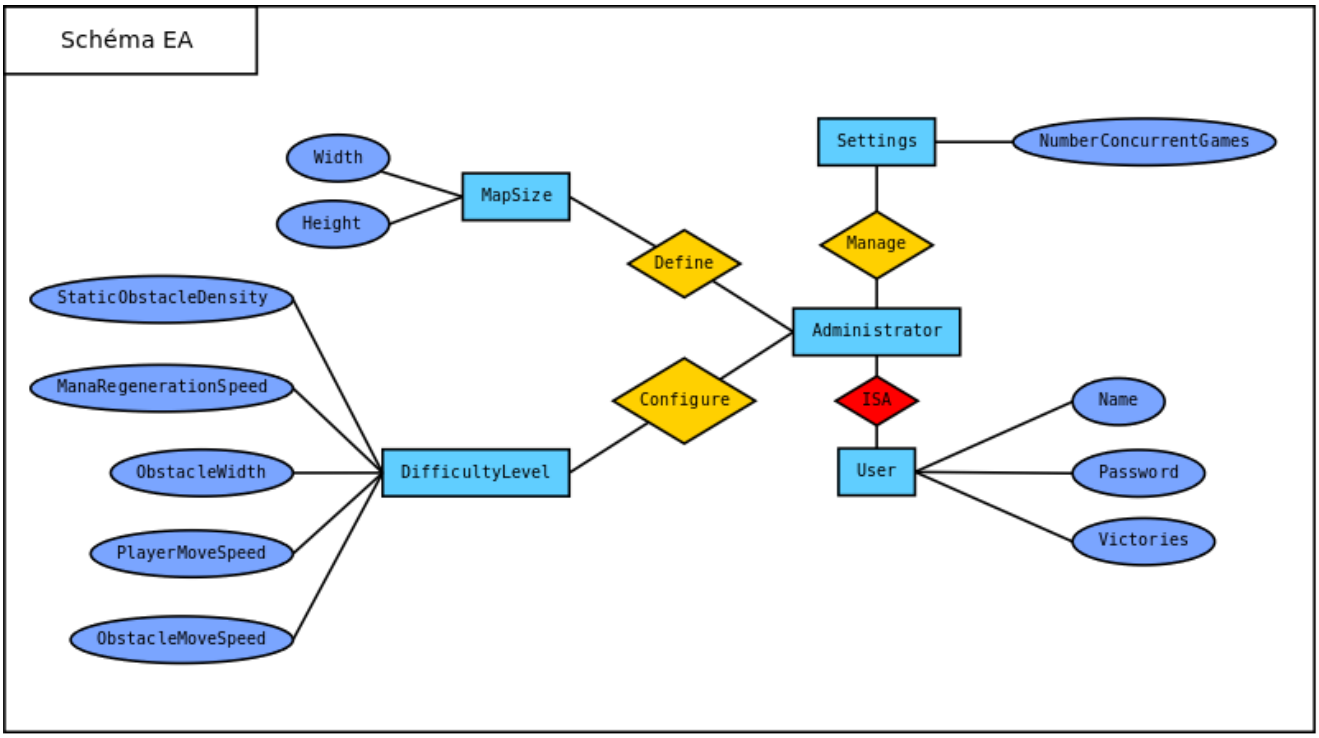

FIGURE 3 – Modèle conceptuel