HEIG-VD

Projet Génie Logiciel

NUKEMAP

Auteurs :Gallay Romain, Koubaa Walid, Muaremi Dejvid, Siu Aurélien

Date : dimanche 20 mai 2018

Table des matières

[Description générale du jeu 3](#_Toc513236855)

[Description de l’environnement de développement 3](#_Toc513236856)

[Fonctionnalités apportées au jeu de base 3](#_Toc513236857)

[Bonus de bombes 3](#_Toc513236858)

[Ennemis originaux 3](#_Toc513236859)

[Apparition des monstres 3](#_Toc513236860)

[Calcul des scores 4](#_Toc513236861)

[Fonctionnement général 5](#_Toc513236862)

[Priorités de la mise en place du jeu 5](#_Toc513236863)

[Fonctionnalités optionnelles (si le temps le permet) 5](#_Toc513236864)

[Interface client-serveur 6](#_Toc513236865)

[Interface de présentation du jeu de la map (accueil) 6](#_Toc513236866)

[Interface du jeu une fois lancé (partie en cours) 6](#_Toc513236867)

[Interface d’affichage de tous les scores et statistiques en fin de partie 7](#_Toc513236868)

[Modèle MVC 8](#_Toc513236869)

[Client 8](#_Toc513236870)

[Serveur 8](#_Toc513236871)

[cas d’utilisation générale 10](#_Toc513236872)

[Diagramme de cas d’utilisation générale 10](#_Toc513236873)

[Player 10](#_Toc513236874)

[Admin 10](#_Toc513236875)

[Échange entre le serveur et le client 11](#_Toc513236876)

[Protocole d’échange Client-Serveur 11](#_Toc513236877)

[Comment se déroulent le lancement du jeu et sa fin ? 13](#_Toc513236878)

[Base de données 13](#_Toc513236879)

[Les différentes responsabilités au sein de la NUKETEAM 13](#_Toc513236880)

[Backlog de produit 14](#_Toc513236881)

[PLAN DES ITERATIONS 16](#_Toc513236882)

[ITERATION 1 16](#_Toc513236883)

[Objectifs 16](#_Toc513236884)

[HISTOIRE 16](#_Toc513236885)

[Bilan du Sprint 16](#_Toc513236886)

[ITERATION 2 18](#_Toc513236887)

[Objectifs 18](#_Toc513236888)

[HISTOIRES 18](#_Toc513236889)

[Bilan du Sprint 18](#_Toc513236890)

[ITERATION 3 19](#_Toc513236891)

[Objectifs 19](#_Toc513236892)

[HISTOIRES 19](#_Toc513236893)

[Bilan du Sprint 19](#_Toc513236894)

[ITERATION 4 20](#_Toc513236895)

[Objectifs 20](#_Toc513236896)

[HISTOIRES 20](#_Toc513236897)

[Bilan du Sprint 20](#_Toc513236898)

[ITERATION 5 21](#_Toc513236899)

[Objectifs 21](#_Toc513236900)

[HISTOIRES 21](#_Toc513236901)

[Bilan du Sprint 21](#_Toc513236902)

[ITERATION 6 22](#_Toc513236903)

[Objectifs 22](#_Toc513236904)

[HISTOIRES 22](#_Toc513236905)

[Bilan du Sprint 22](#_Toc513236906)

[ITERATION 7 23](#_Toc513236907)

[Objectifs 23](#_Toc513236908)

[HISTOIRES 23](#_Toc513236909)

[Bilan du Sprint 23](#_Toc513236910)

[ANNEXES 24](#_Toc513236911)

# Description générale du jeu

**Jeu de type Bomberman**

Le joueur incarne un poseur de bombes, le but étant de faire exploser les adversaires/ennemis pour gagner. Le jeu a connu un grand succès, surtout grâce à son mode multijoueur qui, suivant les machines, permet de jouer jusqu'à une dizaine de personnes en même temps.

Le jeu Bomberman propose généralement un ensemble de bonus permettant d'améliorer les possibilités de son Bomberman. Ces bonus se trouvent la plupart du temps dans les blocs destructibles et apparaissent une fois ceux-ci détruits, ou bien remportés par le joueur lorsqu'il arrive à vaincre un adversaire.

## Description de l’environnement de développement

L’interface graphique sera codée avec JavaFX et utilisera la librairie standard Java ainsi que le framework LibGDX. Ce framework est destiné au développement de jeux, il est gratuit et open source. Nous utiliserons un fichier JSON représentant et spécifiant les cases de la map avec distinction des murs, ennemis et joueurs.

## Fonctionnalités apportées au jeu de base

### Bonus de bombes

Bomb Up : Ce bonus se présente sous la forme d'une bombe. Il augmente le nombre de bombes que peut poser simultanément le Bomberman.

Flamme jaune : Elle augmente la portée de l'explosion des bombes d'une case (portée maximale : environ 10 cases).

N-Bomb (Nuke Bomb) : marquée d'un symbole nucléaire jaune et noir, cette bombe permet au Bomberman de poser une première bombe disposant de la portée maximale d'explosion.

Les bombes posées ultérieurement sont classiques.

### Ennemis originaux

Notre NukeMap met en place des ennemis divers, chacun avec ses spécificités et pouvoirs (simplifiés pour rendre le jeu accessible à tous).

Creeper (ennemi du jeu Minecraft) : en remplacement des ennemis de base.

Enderman (ennemi du jeu Minecraft) : ennemi qui pourrait se téléporter (déplacement aléatoire autour du joueur), déplacement symbolisé simplement par un changement de sa position sur la carte dès qu’un des joueurs croise son regard (face à face).

### Apparition des monstres

L’apparition des monstres se fait en début de partie de manière fixe. Une future version du jeu pourrait faire en sorte que ces derniers apparaissent de manière aléatoire au cours de la partie. Ce n’est pas le cas pour le moment.

### Calcul des scores

Le joueur gagne des points pour différentes actions définies dans le tableau ci-dessous, ce tableau de gain de point empêchera un joueur « opportuniste » de se cacher et de « camper » en attendant gentiment dans son coin la fin de la partie, car son total de point sera faible. Au contraire un joueur motivé sera bien plus récompensé en termes de points totaux gagnés. Le joueur motivé gagne plus de points en ayant effectué seulement la première élimination que celui qui sera caché dans son coin.

#### Tableau de gains de points

|  |  |
| --- | --- |
| Action | Point gagné |
| Élimination joueur | +100 |
| Première élimination de joueur | +200 |
| Elimination monstre creeper | +10 |
| Élimination monstre enderman | +20 |
| Destruction de bloc | 0 (mais PowerUp gagné) |
| Dernier survivant | +150 |

# Fonctionnement général

Pour l’implémentation de notre NukeMap, une liste de priorités des fonctionnalités et l’ordre de leur mise en place s’avèrent requis pour un suivi continu du jeu.

## Priorités de la mise en place du jeu

* Création et affichage d'une map
* Possibilité de se déplacer dans la map
* Possibilité de créer de nouveaux joueurs
* Pouvoir poser une bombe qui explose
* Compte à rebours pour chaque bombe
* Explosion avec reconnaissance des murs
* Explosion avec reconnaissance des ennemis/joueurs
* Comptabilisation du score
* Mise en place de l'interface graphique du jeu
* Mise en place de l'interface de début de jeu (choix du nombre de joueurs, choix de la carte)
* Gestion de la musique de fond et les sons d'explosion, Power-ups…

## Fonctionnalités optionnelles (si le temps le permet)

* Gestion du son et animation (Power-UPS/ explosion).
* Mise en place d'une bande-son continue lors du jeu : https://www.youtube.com/watch?v=Z\_Ep03hFfd4
* Lors d'une explosion : lancement en arrière-plan d’un léger son d'explosion
* Gestion du timer : fin de la partie à la fin du temps imparti

# Interface client-serveur

Le jeu sera jouable en multijoueur, offrant donc la possibilité de jouer à plusieurs simultanément sur la même map. Interconnectés, les joueurs pourront donc se défier afin de remporter la partie en ligne. Voici les **quatre principales interfaces** utilisateurs seront proposées.

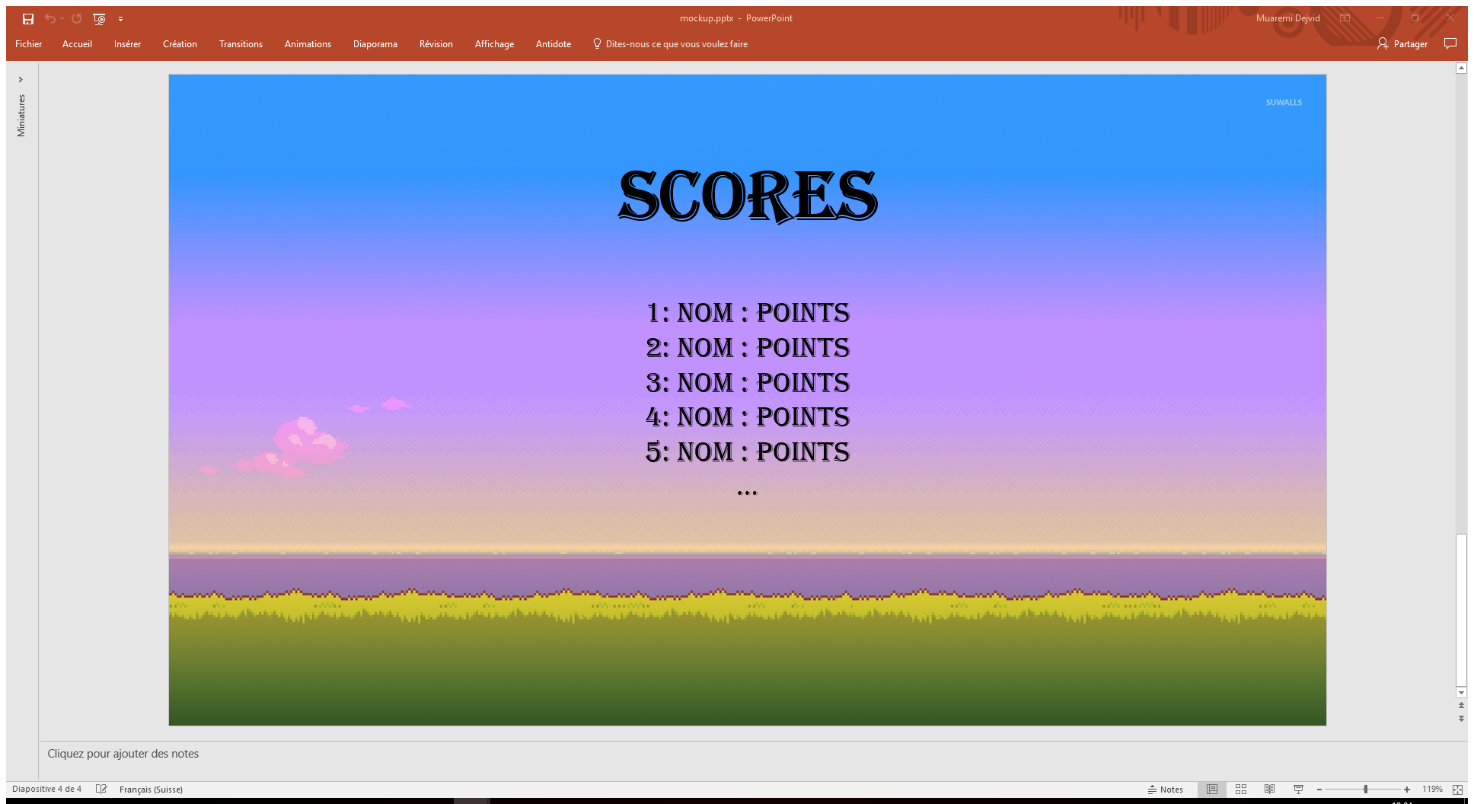
## Interface de présentation du jeu de la map (accueil)



## Interface du jeu une fois lancé (partie en cours)

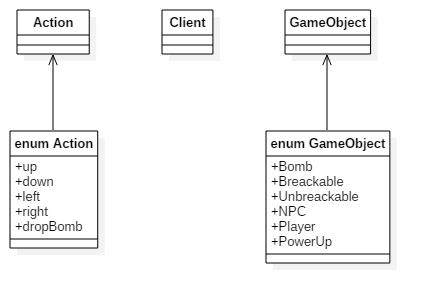


## Interface d’affichage de tous les scores et statistiques en fin de partie

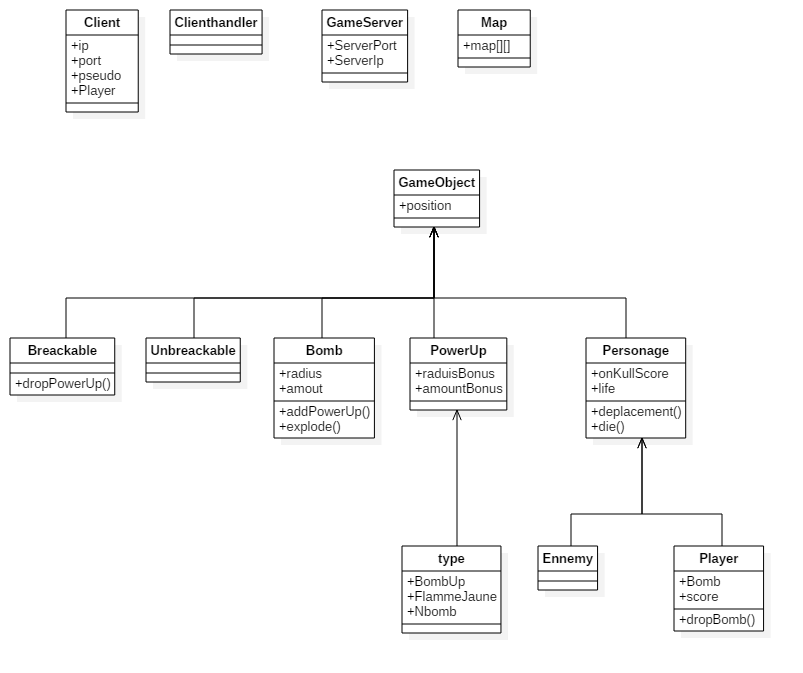


# Modèle MVC

## Client

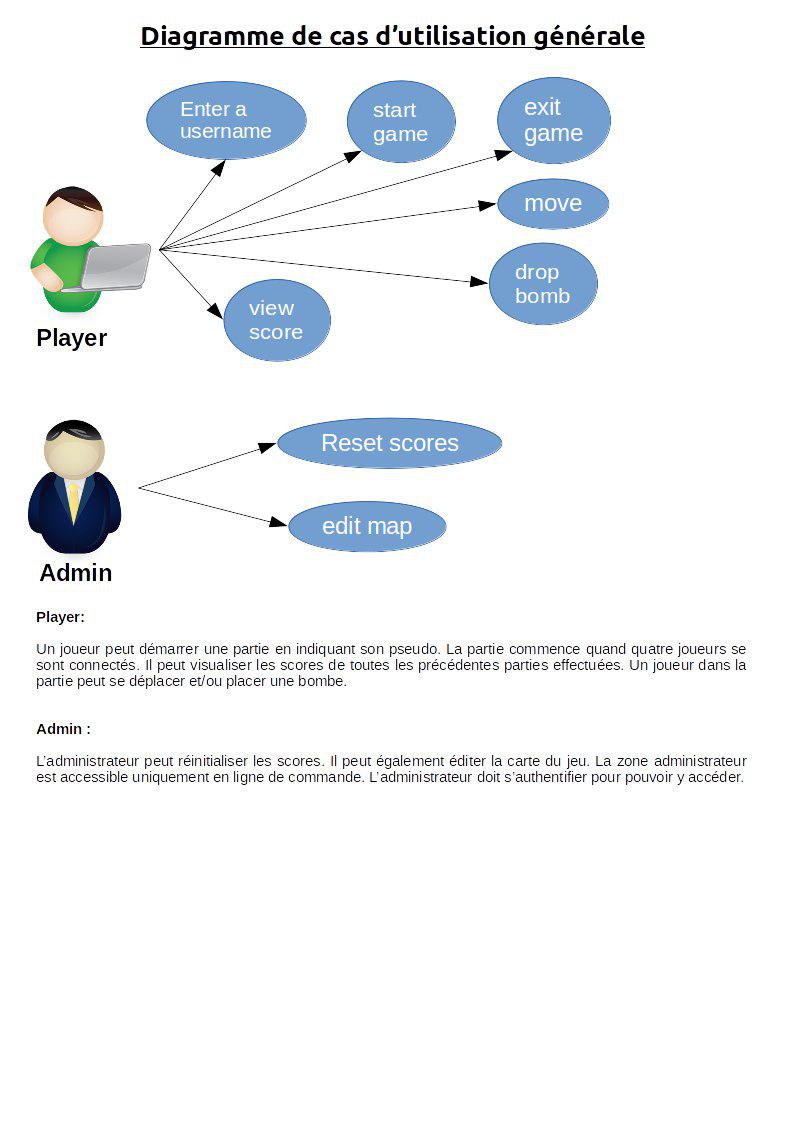


## Serveur



# cas d’utilisation générale

## Diagramme de cas d’utilisation générale



## Player

Un joueur peut démarrer une partie en indiquant son pseudo. La partie commence quand quatre joueurs se sont connectés. Il peut visualiser les scores de toutes les précédentes parties effectuées. Un joueur dans la partie peut se déplacer et/ou placer une bombe :

* Menu : écran d'accueil affiché lors du lancement de l'application, contient trois boutons : start game, view score et exit, ainsi que le champ de texte username.
* Start game : permet de lancer une partie, de base à 2 joueurs.
* View score : affiche un leaderboard contenant le top 10 des scores enregistrés dans la base de données.
* Username : champs de texte permettant au joueur d'entrer un nom d'utilisateur qui sera utilisé pour enregistrer son score.
* Move : le joueur peut se déplacer dans 4 directions différentes : haut, bas, droite ou gauche, il reste limité par les murs et les obstacles.
* Drop bomb : permet de poser une bombe en appuyant sur une touche, lors de l'obtention d'une bombe plus puissante elle sera automatiquement utilisée dans le prochain tour.
* Exit game : quitte la partie pour retourner sur le menu, le joueur qui décide de quitter une partie en cours sera considéré comme perdant.

## Admin

L’administrateur peut réinitialiser les scores. Il peut également éditer la carte du jeu. La zone administrateur est accessible uniquement en ligne de commande. L’administrateur doit s’authentifier pour pouvoir y accéder.

* Reset scores : réinitialise tous les scores.
* Edit map : permet de modifier des cartes. Une carte est représentée par un fichier json qui sera parsé par le programme côté serveur.

# Échange entre le serveur et le client

Partage des responsabilités entre le serveur et le client : Qui fait quoi ?

Le protocole de communication s’effectuera en TCP/UDP et avec des échanges d’informations en JSON contenant les informations suivantes : Position des Joueurs, position des ennemis toujours présents (si plus présent c’est qu’ils sont morts) et positions de blocs casés. Le serveur les « parse » et transmet l’information à tous les joueurs.

Ainsi chaque joueur partage ses coordonnées avec le serveur (bloc avant et après) et la direction ou il va et le serveur décide qui passe) -> du coup déplacement NON case par case, mais pixel par pixel pour permettre une fluidité du jeu et pas de temps de latence très grands (du coup décision ne se fait que si deux jours veulent être sur le même bloc (assez rapide pour le serveur :D )

# Protocole d’échange Client-Serveur

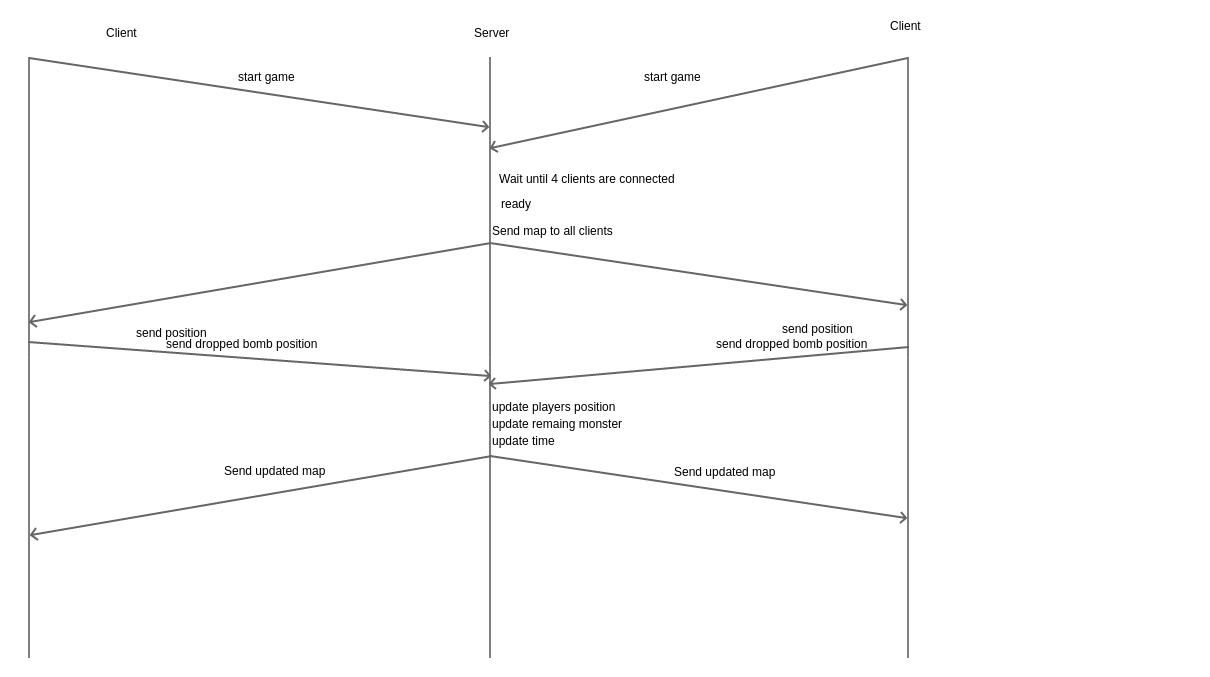
Le serveur attend que 4 clients soient connectés pour démarrer une partie. Dès que c’est le cas, le serveur envoie la carte à chacun des joueurs et la partie peut débuter.

Dès que le joueur s’est déplacé et/ou a déposé une bombe, le joueur envoie ses informations de positions (sa position et celle de la bombe) au serveur.

Le serveur met à jour la carte à chacune de ces requêtes. Cela comprend, la position de tous les joueurs, de tous les ennemis restants et la position des bombes. Il envoie les mises à jour en broadcast après un temps déterminé.

Si un joueur n’a plus de vie, le signal death lui est envoyé. Il peut toujours regarder ce qui se passe dans le jeu, mais ne peut plus interagir avec son personnage (il n’existe plus sur la carte).

Cela se répète indéfiniment jusqu’à ce qu’un seul joueur reste sur la carte.



# Comment se déroulent le lancement du jeu et sa fin ?

Quel programme doit-il être exécuté, comment s’opèrent les connexions, quand est- ce que le jeu commence, quand est-ce qu’il se termine...)

Le programme s’exécute avec un jar fournit, « Nukemap.jar », le jeu est lancé et le joueur attend d’être ajouté à une partie en ligne. Le programme lancera la connexion client au serveur en créant un nouveau client (nouveau joueur) sur le serveur du jeu. La connexion s’opère via TCP en lançant un nouveau thread Client.

Le jeu ne commence qu’une fois que 4 joueurs (strictement et uniquement 4 joueurs) sont en ligne et souhaitent jouer, une fois que les 4 joueurs sont connectés au serveur, le lancement de la partie s’opère. Les joueurs sont demandés à choisir un username (un username différent parmi ceux des joueurs dans la partie sinon redemande un nouveau username), une fois cette vérification faite, le jeu commence réellement et les bombes pleuvent !

Le jeu se termine une fois que le temps imparti d’une partie est écoulé. Une fois la partie terminée et que le score est affiché les joueurs sont à nouveau mis dans l’interface d’attente de joueurs (loading players…), au cas où un des joueurs aurait quitté le jeu après la fin d’une partie (connexion avec le serveur coupée) et relance une nouvelle partie une fois 4 joueurs connectés au serveur.

Dans le cas où un joueur quitte la partie prématurément on arrête la partie et affiche l’interface de score final, et ensuite passe à l’interface d’attente de joueurs (loading players…).

Le score est constamment affiché pendant la partie tandis que le score final est affiché en fin de partie.

On peut redémarrer le jeu en appuyant sur le bouton « Menu », qui va nous remettre dans l’interface de menu avec la possibilité de relancer une nouvelle partie si besoin.

# Base de données

Pas vraiment de base de données a proprement dit.

Les données persistantes sont stockées en JSON ou sous forme de tuiles.

Les cartes sont stockées sous format JSON, ainsi que le nombre de vies, le temps de durée d’une partie.

Les ennemis et le personnage représentant un joueur (Bomberman) sont représentés avec des spreedsheet (tuiles).

# Les différentes responsabilités au sein de la NUKETEAM

* Le scrum master, maitre suprême du projet NUKEMAP est le célèbre Blackhat zedsdead95 (Le truand)
* Aurélien SIU est l’homme à tout faire et product owner (MacGyver)
* Dejvid MUAREMI est le product designer (Le bon)
* Romain GALLAY est le testeur (La brute)
* Eric le François, aka E.L.S est le product owner

# Backlog de produit

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | State | Description | Effort | todoDate | Type | uid |
| Bande son du jeu | Planned | Bande son du jeu (bruitages explosion des bombes, musique de fond) | 1.00 | 18/04/2018 | User story | 13 |
| Création d'un Personnage | Done | Implémentation d'un personnage (points de vie, déplacement, collision, Texture, position) | 8.00 | 18/04/2018 | User story | 8 |
| Démarrage d'une partie | Planned | Ecran de chargement, attente des autres joueurs | 13.00 | 18/04/2018 | User story | 16 |
| Déplacement des ennemis | In progress | Déplacement des ennemis de manière aléatoire dans l'environnement graphique | 8.00 | 18/04/2018 | User story | 3 |
| Déplacement des joueurs | Done | Déplacement des joueurs dans l'environnement graphique (déplacement contrôlé par l'utilisateur) | 5.00 | 18/04/2018 | User story | 2 |
| Gestion des bonus | Planned | Gestion des bonus de bombes | 5.00 | 18/04/2018 | Technical story | 12 |
| Mise en place de l'interface graphique | Done | Avoir une interface graphique permettant d'avoir un premier aperçu du rendu graphique statique (sans déplacement possible des personnages). | 8.00 | 18/04/2018 | User story | 1 |
| Spécialisation d'un Ennemi | Suggested | Spécialisation d'un Ennemi à partir d'un personnage. |  | 18/04/2018 | Technical story | 10 |
| Barre d'infos | Planned | Barre d'infos contenant le score, la vie, et les bonus de bombes. | 2.00 | 18/04/2018 | User story | 7 |
| Gestion de la mort d'un personnage | Planned | Gestion de la mort d'un personnage, notamment les conséquences sur le joueur et sur le jeu en lui-même. | 21.00 | 18/04/2018 | Technical story | 17 |
| Gestion de la victoire | Planned | Gestion de la victoire (écran spécial?) Score | 8.00 | 18/04/2018 | User story | 18 |
| Gestion des contraintes de collisions | In progress | Gestion des collisions entre les différents personnages et les éléments de décors dans le plan | 8.00 | 18/04/2018 | Technical story | 4 |
| Modèle des données persistantes | Done | Définir la structure des fichiers JSON contenant nos données persistantes. | 3.00 | 18/04/2018 | Technical story | 20 |
| Protocole de communication | Done | Définir le protocole de communication client/serveur | 34.00 | 18/04/2018 | Technical story | 11 |
| Gestion des scores | Planned | Méthodes de gestion pour le score | 3.00 | 18/04/2018 | User story | 14 |
| Posage de bombes | In progress | Possibilité de poser une bombe à un emplacement choisi par l'utilisateur. Disparaît après un temps donné | 3.00 | 18/04/2018 | User story | 5 |
| Schema global du projet | Done | Créer un schéma UML de classe représentant le projet côté server et client | 3.00 | 18/04/2018 | Technical story | 19 |
| Gestion explosion bombe | In progress | Gérer l'animation d'explosion d'une bombe | 5.00 | 18/04/2018 | Technical story | 6 |
| Menu de départ | Planned | Menu de départ (Play, Score, login, exit) | 5.00 | 18/04/2018 | User story | 15 |
| Spécialisation d'un personnage | In progress | Spécialisation d'un personnage qui peut être un joueur ou un ennemi (monstre) | 13.00 | 18/04/2018 | Technical story | 9 |

# PLAN DES ITERATIONS

## ITERATION 1

### Objectifs

Fournir une simple interface graphique du jeu avec pour seule interaction possible l'accès aux différentes scènes (menu et carte de jeu). Modéliser le schéma global du projet ainsi que celui des données persistantes.

### HISTOIRE

* Mise en place de l’interface
  + 1 Création des différentes interfaces du jeu
  + 2 Gérer la possibilité de switcher d’une interface à l’autre après un événement
* Schéma global du projet
  + 3 Création du schéma de classe côté serveur
  + 4 Création du schéma de classe côté client
* Modèle des données persistantes
  + 5 Décrire la structure du fichier JSON contenant la carte du jeu
  + 6 Décrire la structure du fichier JSON contenant les scores de chaque utilisateur

### Bilan du Sprint

#### Bilan par le scrum master

Histoire No 1, 19, 20 planifiées : toutes terminées à 100 %.

Vélocité du sprint : 14

Pas de replanification

#### Commentaire général

Globalement chaque tache a été traitée dans les durées estimées et dans les délais impartis. Pas de soucis d’organisation pour ce premier sprint, dénotant ainsi un bon démarrage de projet

#### Autocritique

Ce sprint étant le premier nous avons décidé de prendre le temps d’organiser chacun des sprints suivants et ainsi la charge de travail de celui-ci était relativement faible par rapport aux suivants. Nous aurions pu entamer le second sprint en avance qui quant à lui est plus conséquent.

#### Bilans personnels

|  |  |  |
| --- | --- | --- |
| Membre | Temps prévu | Temps réalisé |
| GALLAY Romain | 6 | 4 |
| KOUBAA Walid | 1 | 1 |
| MUAREMI Dejvid | 6 | 4 |
| SIU Aurélien | 12 | 12 |

#### Commentaire personnel

Gallay Romain : La construction du projet étant à ses débuts, certaines parties du schéma UML risquent de changer selon les besoins futurs.

Koubaa Walid : La partie d’affichage des différentes interfaces ainsi que les passages d’une interface à l’autre (menu, score, fin) n’a pas causé de grands soucis. Aidé par mon collègue Aurélien, nous avons utilisé nos connaissances acquises pour terminer nos tâches.

Le temps prévu pour ces deux tâches est en accord avec celui prévu sur icescrum par le scrum master.

Muaremi Dejvid : J’ai pu mettre en pratique les notions acquises en SER afin de créer mes divers fichiers json. Il est possible que ces fichiers changent légèrement durant le projet lorsque l’on définira de manière plus concrète leur contenu.

Siu Aurélien : J'ai pu découvrir le framework LibGDX afin de mettre en place l'interface graphique. Pas de problème particulier rencontré lors de ce sprint.

## ITERATION 2

### Objectifs

Établir le protocole de communication entre le client et le serveur. Créer la classe Personnage qui servira de base pour les joueurs et les ennemis.

### HISTOIRES

* Protocole de communication
  + 7 implémentation de la partie serveur
  + 8 implémentation de la sérialisation/désérialisation
  + 9 implémentation de la partie client
  + 10 Tests de clients se connectant au serveur du jeu
* Création d’un personnage
  + 11 création de classe représentant un personnage
  + 12 création de la texture du personnage

### Bilan du Sprint

#### Bilan par le scrum master

Histoire No 8 et 11 planifiées : toutes terminées à 100 %.

Vélocité du sprint : 42

Pas de replanification

#### Commentaire général

Chaque tâche a été terminée dans les délais et de manière satisfaisante. Nous avons également pris un peu d'avance dans la gestion du déplacement des personnages.

#### Autocritique

Nous avons découvert que les deux classes représentant un personnage (côté client et serveur) manquaient de précision concernant notamment la séparation de certains attributs, il a donc fallu revoir leur conception. Nous nous sommes également rendu compte que la durée prévue de certaines tâches a été surestimée, ce qui nous a permis de prendre de l'avance sur le prochain sprint.

#### Bilans personnels

|  |  |  |
| --- | --- | --- |
| Membre | Temps prévu | Temps réalisé |
| GALLAY Romain | 5 | 2 |
| KOUBAA Walid | 5 | 12 |
| MUAREMI Dejvid | 6 | 4 |
| SIU Aurélien | 8 | 5 |

#### Commentaire personnel

Gallay Romain : Le développement du client s'est bien passé grâce à une collaboration étroite avec le développeur s'occupant du server (j'ai nommé Walid).

Koubaa Walid : Nous avons finalement décidé d'utiliser node.js pour implémenter le serveur, ce qui nous a permis d'utiliser certaines librairies bien pratiques. L'utilisation des textures atlas n'a pas été aussi aisée que prévu, mais j'ai fini par réussir à les utiliser de manière satisfaisante. L'implémentation du serveur a pu se faire correctement grâce à une communication efficace avec mon collègue s'occupant de la partie client.

Muaremi Dejvid : Mon sprint s’est déroulé sans trop de problème. Cependant, j’ai dû modifier une partie du personnage afin de gérer la sérialisation.

Siu Aurélien : L'implémentation de la classe Personnage s'est révélé plus consistante que prévu. En effet, j'ai dû définir précisément quelles informations devaient se trouver côté serveur ou client. J'ai tenté d'appliquer une structure d'héritage la plus cohérente possible en utilisant par exemple la super classe Sprite définissant de base les textures avec libgdx.

## ITERATION 3

### Objectifs

Créer les bases des classes représentant un joueur et un ennemi, gérer le déplacement de ceux-ci.

### HISTOIRES

* Spécialisation d’un personnage
  + 15 Spécialisation d'un joueur
  + 16 Spécialisation d'un ennemi (monstre)
  + 17 Définir les textures des ennemis
  + 18 Lier les personnages monstre et joueur au serveur
* Déplacement des joueurs
  + 19 Envoie et réception des événements du clavier
  + 20 Améliorer le client handler du serveur pour gérer les événements reçus
* Déplacement des ennemis
  + 21 Implémentation d'une IA basique
  + 22 Application des mouvements chez les clients

### Bilan du Sprint

#### Bilan par le scrum master

Histoire No 2, 3, 9 planifiées : 2 terminé à 100 %, 3 terminé à 0% et 9 terminé à 75%.

Vélocité du sprint : 5

Replanification des histoires au début du sprint, certaines histoires étaient mal définies au début du sprint.

#### Commentaire général

Ce sprint tombe en pleine semaine de rendu de PRO. Nous avions déjà pris nos dispositions en accomplissant lors du sprint 2 certaines taches du sprint 3 en avance, néanmoins les taches restantes ont dû être reporté au sprint suivant. La moitié des taches ont été néanmoins effectuées.

#### Autocritique

Nous allons devoir supporter une charge de travail supplémentaire lors de l’itération 4, néanmoins celle-ci sera moindre car la structure de tout le programme (fonctionner classes des différents éléments) a déjà été codées, il ne reste qu’a les afficher sur le GUI et échanger ces informations à chaque utilisateur (rajouter des fonctions, récupérant et transmettant ces informations, dans le serveur.

#### Bilans personnels

|  |  |  |
| --- | --- | --- |
| Membre | Temps prévu | Temps réalisé |
| GALLAY Romain | 5.5 | 3 |
| KOUBAA Walid | 4 | 2 |
| MUAREMI Dejvid | 2 | 2 |
| SIU Aurélien | 6 | 3.5 |

#### Commentaire personnel

Gallay Romain : Peu de temps pour avancer les tâches prévues dans ce sprint, l'implémentation de l'IA des monstres nécessite également une gestion des collisions que nous mettrons en place lors du prochain sprint.

Koubaa Walid : Il faudra redoubler d’effort lors du sprint 4 pour éviter d’accumuler le retard et terminer chaque tache prévue à temps.

Muaremi Dejvid : C’est le printemps, les journées sont plus longues, tant mieux pour nous car on aura besoin de temps pour finir le projet.

Siu Aurélien : J'ai manqué de temps pour mener à terme mes tâches. J'ai tout de même pu prendre un peu de temps pour préparer le terrain au niveau de la gestion des collisions. Il nous faudra tout de même rattraper ce retard lors du sprint 4.

## ITERATION 4

### Objectifs

Gérer les collisions entre les différents éléments du jeu. Implémenter les bombes dans le jeu et permettre au joueur de les poser.

### HISTOIRES

* Déplacement des ennemis
  + 21 Implémentation d'une IA basique
  + 22 Application des mouvements chez les clients
* Gestion des contraintes de collisions
  + 23 Gestion des bordures des objets cassables
  + 24 Gestion des bordures des objets incassables
  + 26 Les monstres ont conscience des bordures via une hit box
  + 27 Les personnages ont conscience des bordures via une hit box
* Posage de bombes
  + 28 Le joueur peut poser des bombes
  + 29 Le serveur envoie les bombes aux autres joueurs
* Gestion explosion bombe
  + 30 La bombe peut exploser
* Spécialisation d'un personnage
  + 18 Lier les personnages monstre et joueur au serveur

### Bilan du Sprint

#### Bilan par le scrum master

Histoire No X planifiées : X terminées à X %.

Vélocité du sprint : X

Pas de replanification

#### Commentaire général

#### Autocritique

#### Bilans personnels

|  |  |  |
| --- | --- | --- |
| Membre | Temps prévu | Temps réalisé |
| GALLAY Romain | 8 |  |
| KOUBAA Walid | 6 |  |
| MUAREMI Dejvid | 3 |  |
| SIU Aurélien | 7 |  |

#### Commentaire personnel

Gallay Romain :

Koubaa Walid :

Muaremi Dejvid :

Siu Aurélien :

## ITERATION 5

### Objectifs

Implémenter les bonus (power-up). Ajouter la barre d'information à l'écran de jeu. Rendre fonctionnel l'écran de menu et implémenter la gestion des scores (affichage et comptabilisation).

### HISTOIRES

* Gestion des bonus
* Barre d’infos
* Menu de départ
* Gestion des scores

### Bilan du Sprint

#### Bilan par le scrum master

Histoire No X planifiées : X terminées à X %.

Vélocité du sprint : X

Pas de replanification

#### Commentaire général

#### Autocritique

#### Bilans personnels

|  |  |  |
| --- | --- | --- |
| Membre | Temps prévu | Temps réalisé |
| GALLAY Romain |  |  |
| KOUBAA Walid |  |  |
| MUAREMI Dejvid |  |  |
| SIU Aurélien |  |  |

#### Commentaire personnel

Gallay Romain :

Koubaa Walid :

Muaremi Dejvid :

Siu Aurélien :

## ITERATION 6

### Objectifs

Gérer le démarrage d'une partie lorsque 4 joueurs sont connectés. Gérer les conséquences de la mort d'un personnage sur la partie, pour le joueur et pour le client.

### HISTOIRES

* Démarrage d’une partie
* Gestion de la mort d’un personnage

### Bilan du Sprint

#### Bilan par le scrum master

Histoire No X planifiées : X terminées à X %.

Vélocité du sprint : X

Pas de replanification

#### Commentaire général

#### Autocritique

#### Bilans personnels

|  |  |  |
| --- | --- | --- |
| Membre | Temps prévu | Temps réalisé |
| GALLAY Romain |  |  |
| KOUBAA Walid |  |  |
| MUAREMI Dejvid |  |  |
| SIU Aurélien |  |  |

#### Commentaire personnel

Gallay Romain :

Koubaa Walid :

Muaremi Dejvid :

Siu Aurélien :

## ITERATION 7

### Objectifs

Gérer la victoire d'un joueur lors de la fin de la partie. Implémenter la bande-son du jeu (Jihad Trap remix by zedsdead95)

### HISTOIRES

* Gestion de la victoire
* Bande-son du jeu

### Bilan du Sprint

#### Bilan par le scrum master

Histoire No X planifiées : X terminées à X %.

Vélocité du sprint : X

Pas de replanification

#### Commentaire général

#### Autocritique

#### Bilans personnels

|  |  |  |
| --- | --- | --- |
| Membre | Temps prévu | Temps réalisé |
| GALLAY Romain |  |  |
| KOUBAA Walid |  |  |
| MUAREMI Dejvid |  |  |
| SIU Aurélien |  |  |

#### Commentaire personnel

Gallay Romain :

Koubaa Walid :

Muaremi Dejvid :

Siu Aurélien :

# ANNEXES

Backlog de produit complet Stories - NukeMapGEN - All.xls