HEIG-VD

Projet Génie Logiciel

NUKEMAP

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

Date : mercredi 13 juin 2018

Table des matières

[Introduction 3](#_Toc516005789)

[Analyse 3](#_Toc516005790)

[Fonctionnement général de Nukemap 3](#_Toc516005791)

[Objectifs de base 3](#_Toc516005792)

[Fonctionnalités optionnelles (si le temps le permet) 3](#_Toc516005793)

[Utilisation de Nukemap 4](#_Toc516005794)

[Priorités de développement 4](#_Toc516005795)

[Règle du jeu 5](#_Toc516005796)

[Contraintes 5](#_Toc516005797)

[Partage des responsabilité entre le client et le serveur 6](#_Toc516005798)

[Interface client-serveur 6](#_Toc516005799)

[Interface de présentation du jeu de la map (accueil) 6](#_Toc516005800)

[Interface du jeu une fois lancé (partie en cours) 7](#_Toc516005801)

[Interface d’affichage de tous les scores et statistiques en fin de partie 7](#_Toc516005802)

[Diagramme d’activité général 8](#_Toc516005803)

[Cas d’utilisation 9](#_Toc516005804)

[Diagramme général 9](#_Toc516005805)

[Player 9](#_Toc516005806)

[Admin 9](#_Toc516005807)

[Modèle de domaine 10](#_Toc516005808)

[Client 10](#_Toc516005809)

[Serveur 10](#_Toc516005810)

[Données persistante 10](#_Toc516005811)

[Conception du projet 11](#_Toc516005812)

[Protocole d’échange client-serveur 11](#_Toc516005813)

[Diagramme des échanges de données 11](#_Toc516005814)

[Diagramme de classes 12](#_Toc516005815)

[Client 12](#_Toc516005816)

[Serveur 12](#_Toc516005817)

[Implémentation du projet 13](#_Toc516005818)

[Technologies utilisées 13](#_Toc516005819)

[Technologies non traitées en cours 13](#_Toc516005820)

[Problèmes et solutions 13](#_Toc516005821)

[Gestion du projet 14](#_Toc516005822)

[Les différentes responsabilités au sein de la NUKETEAM 14](#_Toc516005823)

[Plan des itérations 15](#_Toc516005824)

[Itération 1 15](#_Toc516005825)

[Itération 2 17](#_Toc516005826)

[Itération 3 19](#_Toc516005827)

[Itération 4 21](#_Toc516005828)

[Itération 5 23](#_Toc516005829)

[Itération 6 25](#_Toc516005830)

[Itération 7 27](#_Toc516005831)

[Extrait du backlog de produit 28](#_Toc516005832)

[Stratégie de tests 30](#_Toc516005833)

[Stratégie d’intégration du code de chaque participant (GIT) 30](#_Toc516005834)

[Etat des lieux 31](#_Toc516005835)

[Fonctionnalité réalisée 31](#_Toc516005836)

[Fonctionnalité à développer 31](#_Toc516005837)

[Planification pour finir le projet 31](#_Toc516005838)

[Auto-critique 32](#_Toc516005839)

[Ce qui aurait pu être amélioré 32](#_Toc516005840)

[Conclusion 33](#_Toc516005841)

[Annexe 34](#_Toc516005842)

# Introduction

Le but du projet Nukemap est de créer un jeu de type Bomberman.

Le joueur incarne un poseur de bombes, le but étant de faire exploser les adversaires/ennemis pour gagner. Le jeu original 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.

# Analyse

## Fonctionnement général de Nukemap

Pour l’implémentation de notre jeu, Nukemap, nous avons défini une liste de priorités des fonctionnalités afin d’obtenir un jeu qui au final sera jouable. L’ordre de leur mise en place a été adapté afin de suivre le modèle agile scrum tout en donnant une version utilisable du jeu à chaque itération à notre client.

### Objectifs de base

Nous considérons les objectifs suivant comme étant le minimum afin d’avoir un jeu potentiellement utilisable à la fin des 7 sprint du projet GEN :

* 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

### Utilisation de Nukemap

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.

Le jeu se jouera principalement avec les touches fléchées pour déplacer son personnage et se déplacer dans les menus ainsi que de la barre d’espace afin de poser une bombe.

### Priorités de développement

Voir la planification des sprints plus loin dans le document.

### Règle du jeu

Nous avons choisi de reprendre les règles du jeu original, à savoir :

* Le dernier survivant gagne la partie en cours.
* Les scores seront calculés de manière globale pour chaque joueur, somme des points de toutes les parties.
* Si un monstre touche un joueur celui-ci perds.
* Il y a deux types de mur, les cassables et les incassables.
* Pour briser un mur, il faut utiliser la bombe avec la touche indiquée ci-dessus.
* Le nombre de joueur minimum pour lancer une partie est de 4. En dessous, le serveur mettra les joueurs en attente.

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.

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

### Contraintes

Nous avons choisi de personnaliser le jeu en modifiant certains de ses aspects :

* Ajout d’ennemi :
  + Creeper (ennemi du jeu Minecraft) : en remplacement des ennemis de base.
* 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.

## Partage des responsabilité entre le client et le serveur

Le protocole de communication s’effectuera en 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. Le serveur les « parse » et transmet l’information à tous les joueurs.

Ainsi chaque joueur partage ses coordonnées avec le serveur et la direction ou il va et le serveur décide qui passe. Le déplacement se fera en pixel par pixel pour permettre une fluidité du jeu et des temps de latence beaucoup plus courts.

## Interface du jeu

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 proposées.

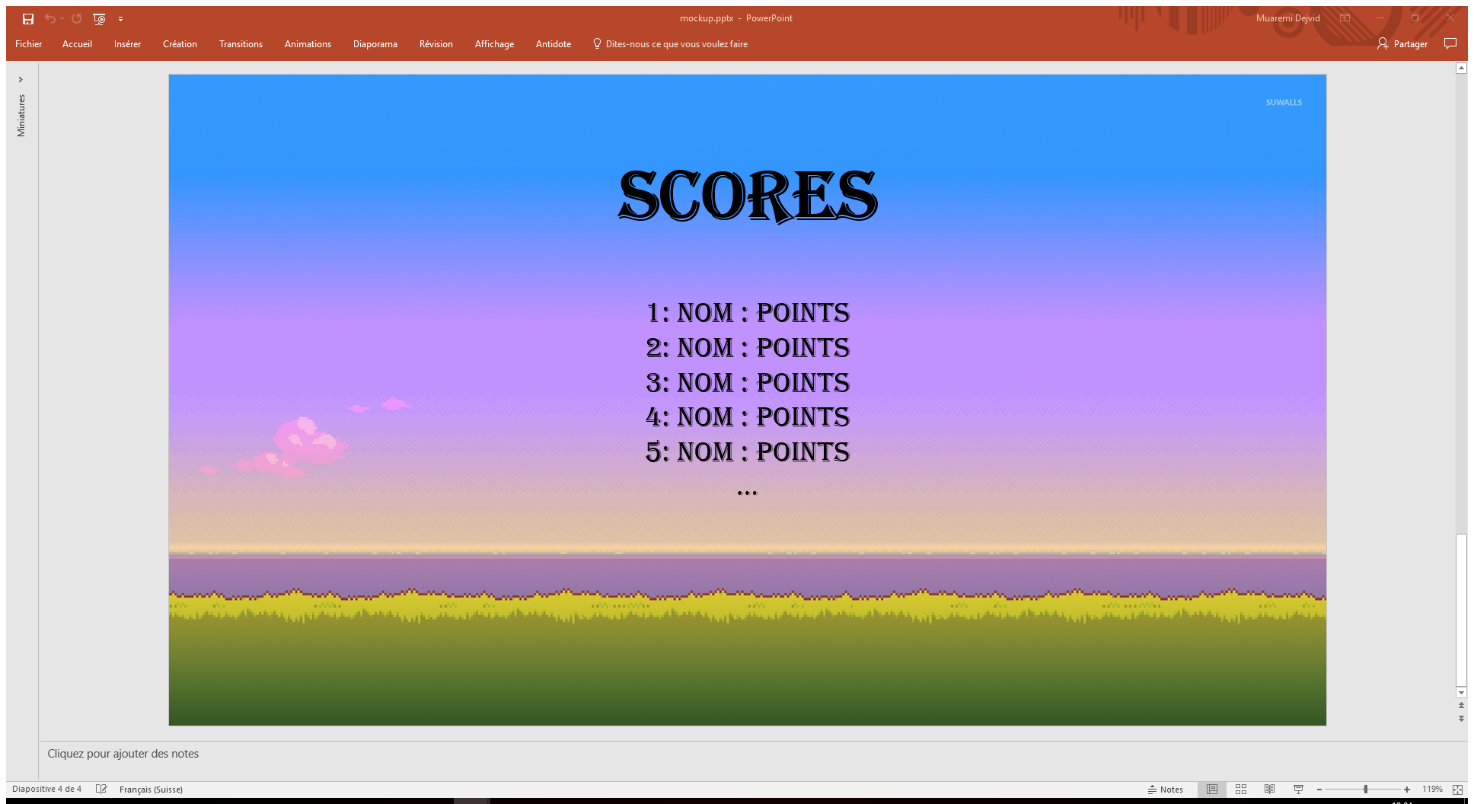
### Interface de présentation du jeu de la map (accueil)



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



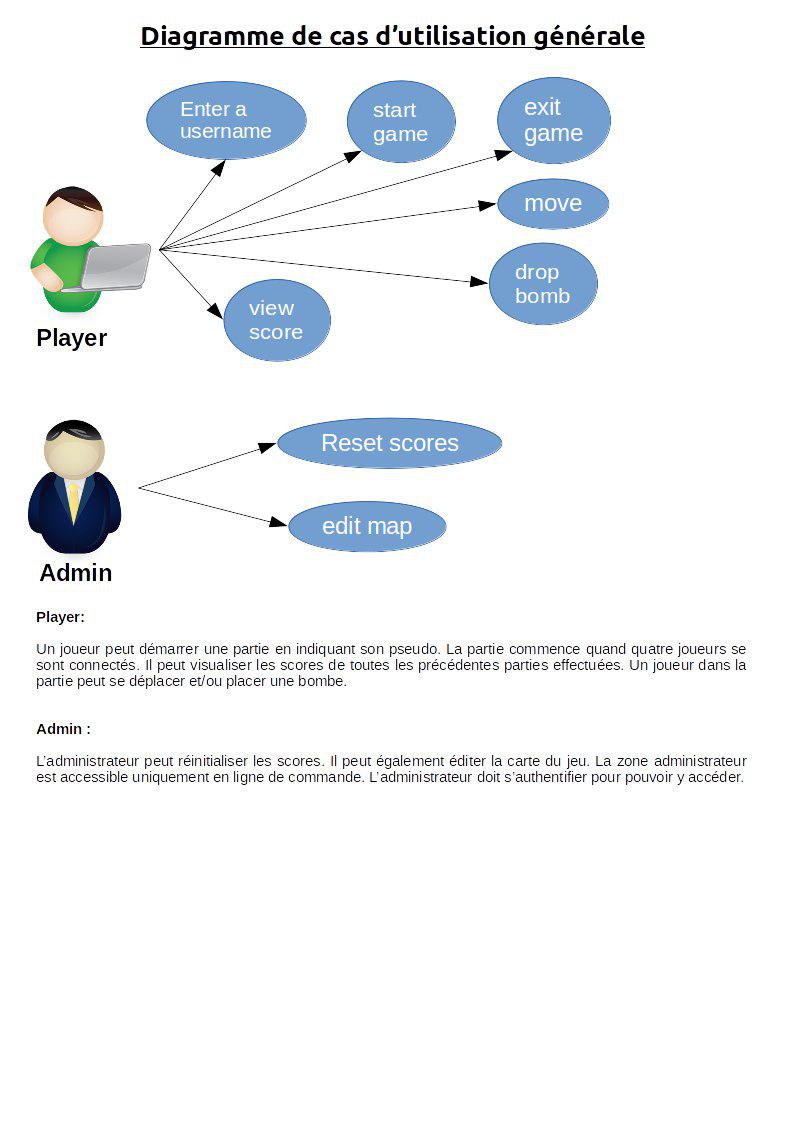
### Interface d’affichage de tous les scores et statistiques en fin de partie



## Diagramme d’activité général

## Cas d’utilisation

### Diagramme général



### Player

Un joueur peut démarrer une partie, mais celle-ci commence quand quatre joueurs se sont connectés. Le joueur 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.
* 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.
* 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.
* Exit game : quitte la partie pour retourner sur le menu, le joueur qui décide de quitter une partie en cours sera considéré comme perdant.

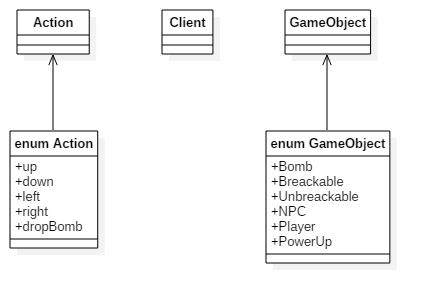
### Admin

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.

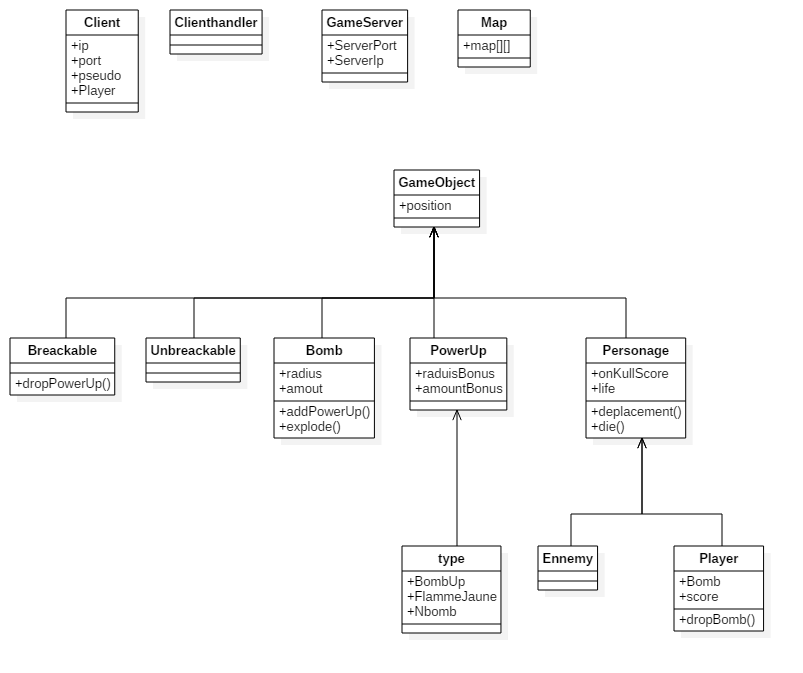
* Reset scores : réinitialise tous les scores.
* Edit map : permet de modifier des cartes. Une carte est représentée par un fichier json qui sera parsé par le programme côté serveur.

## Modèle de domaine

### Client



### Serveur



## Données persistante

Nous avons différents types de données persistantes :

* Les scores sont stockés en JSON sur le serveur
* La carte est stockée en .tmx (dérivé de XML) chez le client
* Les textures sont en .png chez le client.

# Conception du projet

## 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 permet aux joueurs de faire des actions et la partie peut débuter.

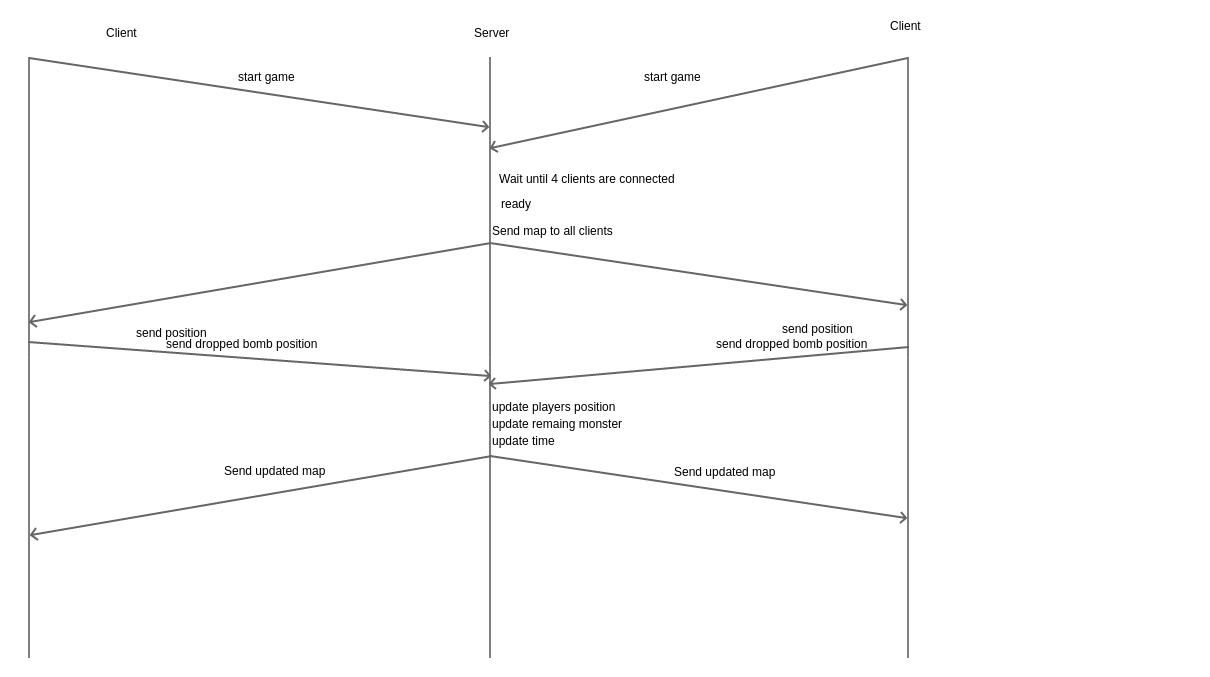
Dès que le joueur s’est déplacé et/ou a déposé une bombe, le joueur envoie ses informations de positions (sa position et celle de la bombe) au serveur.

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, il est déconnecté de la partie et mené sur l’écran des scores.S

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

### Diagramme des échanges de données



## Diagramme de classes

### Client

### Serveur

# Implémentation du projet

## Technologies utilisées

|  |  |
| --- | --- |
| Technologie | But |
| Java | Langage de développement |
| Framework LibGDX | L’interface graphique |
| JSON | Stockage des données persistantes |
| NodeJS | Serveur de jeu |

## Technologies non traitées en cours

## Problèmes et solutions

# Gestion du projet

## 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

## Plan des itérations

### Itération 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.

### Itération 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.

### Itération 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 juste avant la 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.

### Itération 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 3, 4, 5, 6, 9 planifiées : 4 terminées à 75 %, 3, 5, 6, 9 reportée.

Vélocité du sprint : 0

Pas de replanification pour le moment.

##### Commentaire général

Comme anticipé nous n’avons pas eu le temps d’avancer de manière conséquente lors de ce sprint du au rendu final de PRO (.jar + rapport) et une semaine chargée de travaux écrits. Nous avons néanmoins réussi a finir deux histoires qui sont la gestion des collisions ainsi que la spécialisation des personnages.

##### Autocritique

Nous allons devoir rattraper ce retard lors de l’itération 5, ou plusieurs rendez-vous seront à mettre en place afin d’avancer sur les taches choisies par chacun. Le temps passe vite, il s’agit de ne pas laisser toute la charge de travail pour la dernière semaine.

##### Bilans personnels

|  |  |  |
| --- | --- | --- |
| Membre | Temps prévu | Temps réalisé |
| GALLAY Romain | 9 | 4 |
| KOUBAA Walid | 6 | 2 |
| MUAREMI Dejvid | 3 | 1 |
| SIU Aurélien | 7 | 1 |

##### Commentaire personnel

Gallay Romain : Comme prévu, le rendu de PRO a accaparé tout mon temps libre. Dès lors, mon avance sur le mini-projet a été fortement compromise.

Koubaa Walid : Nous avons maintenant le temps de rattraper notre retard, et il sera rattraper. Le groupe se doit d’être productif, et chacun devra faire une part des taches de manière efficace pour être à jour lors des prochains sprints.

Muaremi Dejvid : Ce sprint était pile pendant la semaine de rendu de PRO, par conséquent, je n’ai pas pu beaucoup travailler sur le mini-projet cependant ce retard devra être rattrapé d’ici la fin.

Siu Aurélien : Étant débordé de travail, j'ai juste pu prendre un peu de temps pour avancer la gestion des collisions sans pour autant y mettre un terme.

### Itération 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 contraintes de collisions
  + 26 Les monstres 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
* Spécialisation d'un personnage
  + 18 Lier les personnages monstre et joueur au serveur
* Menu de départ
  + 33 Créer un menu de départ permettant d’accéder aux autres menus

#### Bilan du Sprint

##### Bilan par le scrum master

Histoire No 4, 5, 9, 15 planifiées : 4, 9, et 15 terminées à 100 %, 5 terminée à 50% et repoussée.

Vélocité du sprint : 29

Replanification de la fin du projet. Certaines tâches seront probablement ajoutées à la liste des fonctionnalités optionnelles.

##### Commentaire général

Globalement ce sprint a été efficace. Nous avons rattrapé le retard des sprints précédents, et au vu de certaines histoires des itérations suivantes qui ont déjà été faites en avance (itération 2), nous sommes à jour dans notre suivi des tâches à effectuer. Nous avons aussi fait un refactoring de tout le code afin de rendre le programme conforme au modèle MVC. De plus nous avons effectué un refactoring de toutes les histoires sur ICESCRUM faites et /ou restantes avant la fin du mini projet NUKEMAP.

Nous comptons nous investir beaucoup plus ces deux semaines restantes afin de rendre un jeu fini et conforme au cahier des charges.

##### Autocritique

Nous devons malheureusement finir le projet d'ici deux semaines, ce qui n'augure rien de bon quant à la quantité de sommeil que chaque membre va effectuer les semaines à venir...

##### Bilans personnels

|  |  |  |
| --- | --- | --- |
| Membre | Temps prévu | Temps réalisé |
| GALLAY Romain | 2 | 3 |
| KOUBAA Walid | 3 | 3.5 |
| MUAREMI Dejvid | 2 | 3 |
| SIU Aurélien | 6 | 6.5 |

##### Commentaire personnel

Gallay Romain : Après 2 semaines très chargées par PRO, j'ai enfin pu me concentrer sur GEN. Le refactoring des histoires a été nécessaire et les prochains sprints semblent réalisables malgré les quelques histoires qui sont venues s'y ajouter. J'ai pu avancer le code et terminer mes tâches de ce sprint.

Koubaa Walid : Nous restons confiants et en tant que scrum master je ferais en sorte que les membres du groupe soient au "taquet".

Muaremi Dejvid : Maintenant que PRO est fini, j’ai pu prendre le temps d’avancer sur le projet GEN. Cette semaine, nous avons replanifié la fin du projet, certaines tâches vont peut-être prendre plus de temps mais je suis confiant quant à la suite de Nukemap.

Siu Aurélien : Dans ce sprint, nous avons rattrapé un peu notre retard. Je pense qu'il sera comblé lors du prochain sprint.

### Itération 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

* Gestion explosion bombe
  + 30 La bombe peut exploser
* Déplacement des ennemis
  + 21 Implémentation d'une IA basique
  + 22 Application des mouvements chez les clients
* Barre d’infos
  + 34 Utilisé pour afficher des informations pendant la partie. Affichage à définir durant le sprint.
* Démarrage d’une partie
  + 36 Attente des joueurs avant de lancer la partie
* Gestion de la mort d’un personnage
  + 37 Définir l'action qui sera réalisée lors de la mort d'un personnage joueur
  + 38 Implémenter l'action en cas de mort d'un joueur.
* Gestion des bonus
  + 35 Apparition des bonus

#### 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 : Après le refactoring de la semaine passée, nous avons du avancer sur un sprint très chargé. L'IA des monstres a pour le moment été implémentée de manière très basique mais ceci pourra être repis dans une prochaine version.

Koubaa Walid :

Muaremi Dejvid :

Siu Aurélien : Cette semaine a encore une fois été bien remplie mais j'ai réussi à terminer mes tâches de manière satisfaisante. Le rendu final approche et je sens que notre rythme de travail nous permettra d'obtenir un produit fini.

### Itération 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 des scores
* Bande-son du jeu
* Gestion de la victoire

#### 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 :

## Extrait du 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 |

## Stratégie de tests

A la fin de chaque itération, nous nous sommes assuré que le code écrit fonctionnait correctement. Nous avons pu diviser les tests en 3 catégories :

1. Les tests des classes d’affichage graphique ont essentiellement été effectués par étude visuelle du résultat à l’écran. Nous avons ainsi défini entre nous si le rendu nous semblait acceptable par rapport aux objectifs fixés. Ceci concerne les classes des package scenes, screens et tools.
2. Les tests des classes de communication client – server ont été faits grâce à l’affichage des logs des messages reçus et envoyés du côté client et serveur. Nous avons également utilisé l’outil Wireshark pour analyser les paquets réseaux. Ceci concerne les classes du package client ainsi que le serveur.
3. La plupart des classes de modèle du package GameObject ont été jugées suffisamment simples pour se passer de tests. Nous avons néanmoins implémenté une classe de test unitaire utilisant JUnit 4.0 et ceci s’est révélé plus compliqué que prévu. En effet, nos classes modèles s’appuient pour des composants du framework libgdx, et celui-ci utilise des éléments OpenGL. Il est donc nécessaire de lancer les tests dans un contexte d’exécution OpenGL. Pour cela nous dû utiliser une classe héritant de BlockJUnit4ClassRunner et réimplémentant certaines de ses méthodes, BlockJUnit4ClassRunner étant la classe standard de test utilisée par JUnit.

## Stratégie d’intégration du code de chaque participant (GIT)

Push first -> win !

# Etat des lieux

## Fonctionnalité réalisée

## Fonctionnalité à développer

### Planification pour finir le projet

# Auto-critique

## Ce qui aurait pu être amélioré

# Conclusion

# Annexe

Backlog de produit complet Stories - NukeMapGEN - All.xls

Manuel d’utilisateur GEN\_Nukemap\_Manuel.docx

Manuel d’installation GEN\_Nukemap\_Manuel.docx