

planet.i

Rapport final

Membres Pierre-Benjamin Monaco
Lucas Elisei
Gaëtan Othenin-Girard
David Truan

Professeur Eric Lefrançois
Assistant Christophe Greppin

HEIG-VD - Semestre d'été 2017

Table des matières

1	Introduction	3
2	Analyse	3
2.1	Règles du jeu	3
2.2	Partage des responsabilités client-serveur	3
3	Conception du projet	6
3.1	Protocole d'échange	6
3.2	Diagrammes de classes	7
4	Implémentation du projet	7
4.1	Technologies utilisées	7
5	Gestion du projet	7
5.1	Rôles	7
5.2	Plan d'itération initial	8
5.3	Suivi du projet	10
5.4	Stratégie de tests	13
5.5	Stratégie d'intégration	13
6	État des lieux	13
6.1	Ce qui fonctionne	13
6.2	Développement futur	14
7	Auto-critique	14
7.1	Solution technique	14
7.2	Gestion du projet	14
7.3	Plan d'itérations	14
7.4	Améliorations	15
8	Conclusion	15

Table des figures

1	Diagramme d'activité	4
2	Modèle de domaine serveur	5
3	Modèle de domaine client	6
4	Schéma de la base de données	6

1 Introduction

Ce projet a été développé dans le cadre du cours de Génie Logiciel lors du 4ème semestre à la HEIG-VD. Il s'agit d'un jeu sous la forme d'une application client-serveur.

Le but principal du projet était de nous familiariser avec la gestion et le suivi de projet en appliquant la méthode UP. Il s'agissait aussi d'entraîner le travail en groupe et de faire face aux problèmes qui peuvent en découler.

2 Analyse

Ce chapitre se concentre sur la présentation de notre application, les responsabilités client/serveur et la phase d'analyse du projet.

2.1 Règles du jeu

Cette section va présenter notre jeu, ses règles et ses fonctionnalités.

2.1.1 Description

L'application se présente sous la forme de plusieurs clients et d'un serveur. Chaque client se connecte au serveur, renseigne un pseudonyme et arrive dans le lobby du jeu.

Dans le lobby, le client a une vision sur toutes les parties actuellement créées et peut en rajouter dans la limite offerte par le serveur. Il peut joindre une partie en cours. Lorsque il joint une partie, il se retrouve dans un univers en 2D représentant l'espace et a une visibilité sur tous les corps présents dans celui-ci. Le client contrôle donc une planète. Il peut déplacer celle-ci grâce à sa souris. Chaque planète génère une force gravitationnelle qui attire les corps célestes environnants vers elle.

Lorsqu'il y a collision entre deux corps, il y a plusieurs cas de figure :

- Le corps le plus imposant mange l'autre et grossit (ce qui amplifie sa masse et donc son attraction).
- Les deux corps ont une masse trop proche, le plus gros mange le plus petit et finalement explose en fragments.

En plus des planètes, des bonus se baladent dans l'univers. Ceux-ci octroient des pouvoirs sur une courte durée. La liste des bonus est disponible plus bas.

2.1.2 Contrôles

Le but de notre jeu étant de proposer une mini-simulation de gravité, en plus d'être amusant, nous sommes partis du fait que même les contrôles devaient suivre le thème de l'espace/gravité. Pour déplacer sa planète, il faut donc cliquer sur la fenêtre, ce qui va générer une force de gravité proportionnelle à la taille de notre planète et qui va donc interagir avec cette dernière.

2.1.3 Victoire/défaite

Il n'y a pas de victoire à proprement parler, ni de fin de partie. Les joueurs ont un score qui correspond à la masse de leur planète et leur partie se termine lorsqu'ils se font manger par un autre joueur.

2.2 Partage des responsabilités client-serveur

Une des contraintes de ce projet étant de faire une application de type client-serveur, il nous a fallu réfléchir aux différentes responsabilités que l'une ou l'autre des parties devaient avoir.

Nous sommes partis du principe que les calculs devaient se faire du côté serveur et que les clients n'auraient qu'à mettre à jour leur affichage avec les données envoyées par le serveur.

2.2.1 Diagramme d'activité

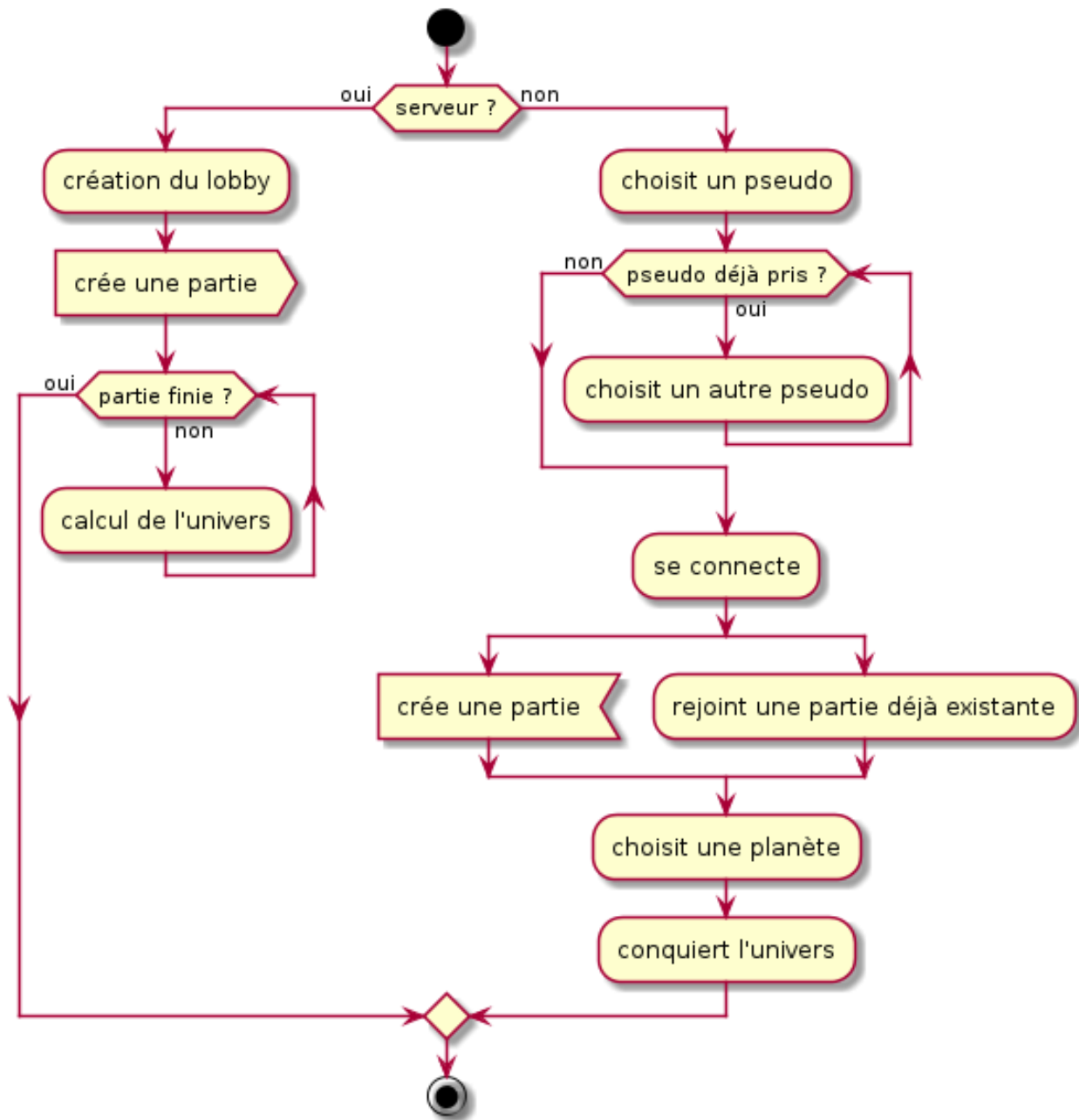


FIG. 1: Diagramme d'activité

2.2.2 Cas d'utilisation

2.2.3 Modèles de domaine

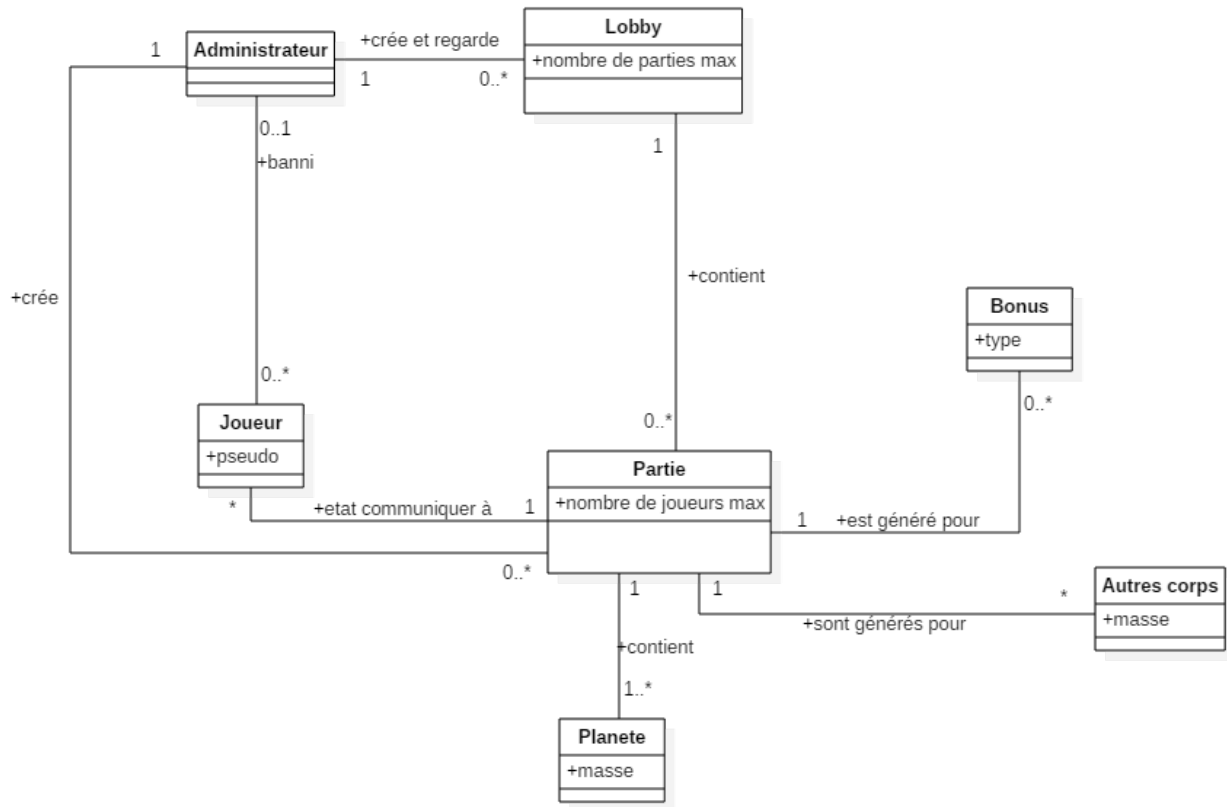


FIG. 2: Modèle de domaine serveur

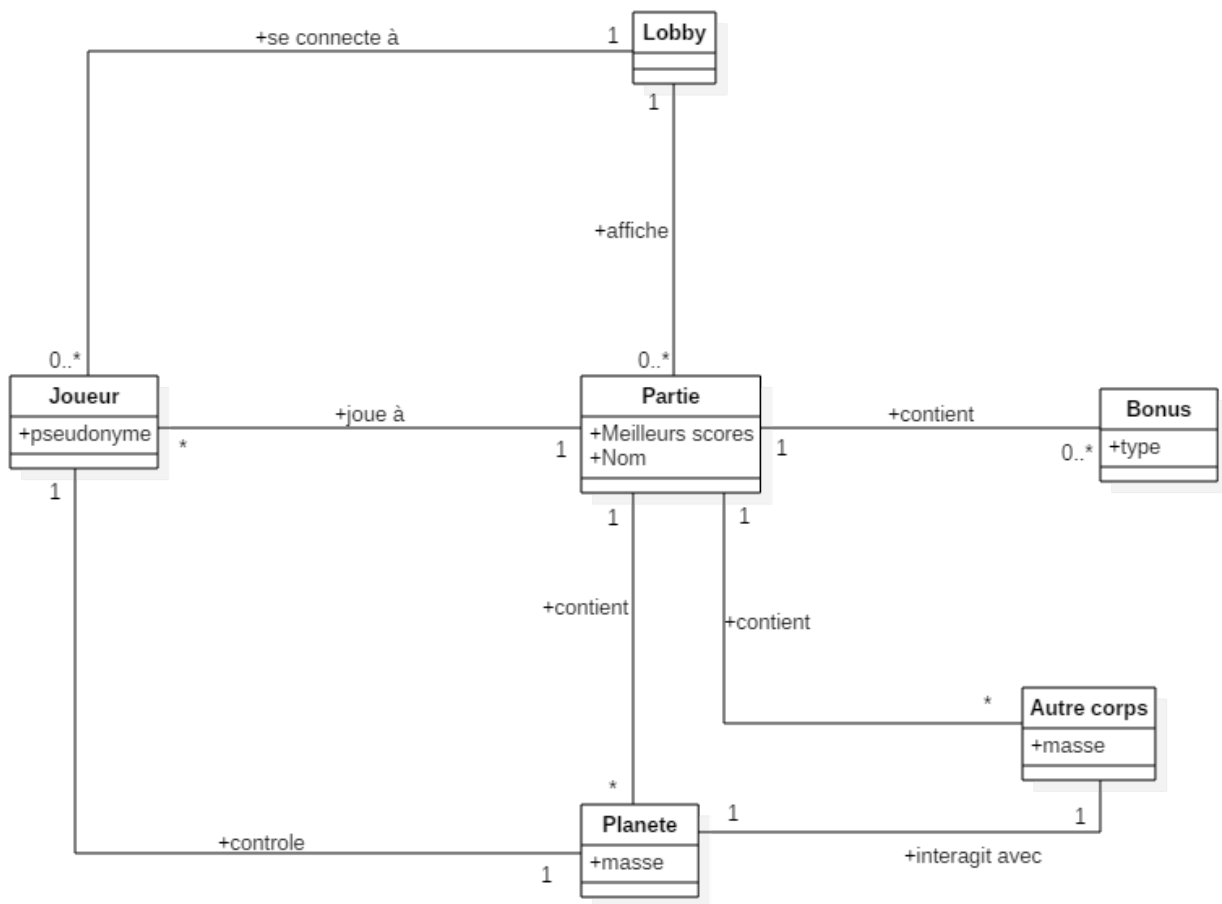


FIG. 3: Modèle de domaine client

2.2.4 Base de données

Notre base de données est assez simpliste car nous sommes partis dans l'optique de faire un jeu arcade où les utilisateurs ne s'enregistrent pas mais s'y connectent juste avec un pseudo avec comme seule contrainte le fait que le pseudo soit déjà pris actuellement sur le serveur. Nous pensons que certaines améliorations sont possibles sur la base de données (scores, nettoyage de la DB mieux géré).


tblUser		
	idUser	integer
	usrName	text
	usrBestScore	integer

FIG. 4: Schéma de la base de données

3 Conception du projet

3.1 Protocole d'échange

Le tableau ci-dessous liste les différentes commandes du protocole et leur rôle :

Commande	Description
PLANET_IO_SUCCESS	Retourné par le serveur en cas de réussite.
PLANET_IO_FAILURE	Retourné par le serveur en cas d'erreur.
LOBBY_UPDATED	Envoyé par le serveur lorsque le lobby a été mis à jour.
GAME_UPDATED	Envoyé par le serveur lorsque la partie a été mise à jour.
PLANET_IO_LOBBY_JOINED	Le client au accède au lobby.
PLANET_IO_HELLO	Découverte du serveur.
PLANET_IO_LOGIN	Le client se connecte au lobby.
NB_GAME_MAX_UPDATE	Le client demande le nombre max de parties.
PLANET_IO_CREATE_PLANET	Le client crée sa planète.
PLANET_IO_SET_PLANET	Utilisé pour créer une planète invisible.
PLANET_IO_KILL_PLANET	Utilisé pour détruire une planète.
CMD_CREATE_GAME	Le client crée une partie.
CMD_JOIN_GAME	Le client rejoint une partie.
CMD_DISCONNECT	Le client se déconnecte.
CMD_SEPARATOR	Séparateur <argument>:<commande>
PLANET_IO_LEAVING_GAME	Le client quitte la partie.
END_OF_COMMAND	Signal de fin de commande.

Les échanges multicast se font à l'adresse 239.192.0.2, sur le port 9898. Les échanges unicast se font sur le port 8585.

3.2 Diagrammes de classes

4 Implémentation du projet

4.1 Technologies utilisées

Planet.io a été développé en Java 8 pour le client comme pour le serveur.

Maven a été utilisé pour la gestion de modules, l'exécution des tests unitaires et la compilation.

Notre base de données s'appuie sur SQLite3.

Pour la sérialisation/désérialisation des données, nous avons utilisé Jackson.

Un dépôt Git a été ouvert sur Bitbucket afin que tout le monde puisse travailler de son côté. L'outil Travis a été ajouté au dépôt afin que chaque push ne casse pas la compilation du projet.

Toute la partie engine physique a été développée par notre chef de projet. Nous avons utilisé **Swing** pour l'UI mais cela n'était peut-être pas le plus adapté.

5 Gestion du projet

5.1 Rôles

- Pierre-Benjamin Monaco
Chef de projet, responsable des normes et procédures, responsable des tests, développeur.
- Lucas Elisei
Architecte, concepteur en chef, développeur.
- David Truan
Représentant des utilisateurs, développeur.
- Gaëtan Othenin-Girard
Responsable de la configuration, développeur.

Au final, les rôles ont bien été respectés. Sur la fin du projet, tout le monde s'est mis à 100% sur le code.

5.2 Plan d'itération initial

5.2.1 Itération 1

5.2.1.1 Objectifs

Modélisation des schémas UML du client et serveur et prototype :

- Schéma UML du client.
- Schéma UML du serveur.
- Schéma UML des modules externes.
- Création d'un prototype simple de client.

5.2.1.2 Temps escompté

- 2 semaines (12.04 au 27.04).

5.2.1.3 Rôles et effort escompté

- P-B. Monaco : Mise en place de JUnit et implémentation des tests du prototype, 2h.
- Lucas Elisei : Développement du schéma UML du serveur, 2h.
- David Truan : Développement du schéma UML du client, 2h.
- Gaëtan Othenin-Girard : Mise en place du projet Maven, du système de contrôle de version et mise en place du système d'intégration continue, 2h.
- Tout le monde : Développement du prototype (travail collaboratif : bonne technique pour être opérationnel avec git le moment venu).

5.2.2 Itération 2

5.2.2.1 Objectifs

Système de jeu fonctionnel en standalone :

- Création partie.
- Contrôle d'une planète.
- Gestion des événements (collisions, changements d'états).
- Design graphique (simple).

5.2.2.2 Temps escompté

- 1 semaine (27.04 au 04.05).

5.2.2.3 Rôles et effort escompté

- P-B. Monaco : Tests unitaires et revue de code, 4h.
- Lucas Elisei : Gestion des propriétés physiques, 6h.
- David Truan : Design graphique, validation du design auprès d'utilisateurs alpha, 5h.
- Gaëtan Othenin-Girard : Gestion événementielle, 6h.

5.2.3 Itération 3

5.2.3.1 Objectifs

L'objectif de cette itération est l'amélioration de l'application créée lors de la première itération mais en fonctionnant de manière client/serveur. Sur cette version, le serveur contiendra les informations concernant les objets du jeu qu'il enverra au client. Le client va uniquement afficher les objets aux coordonnées reçues.

5.2.3.2 Temps escompté

- 1 semaine (04.05 au 11.05).

5.2.3.3 Rôles et efforts escomptés

- P-B. Monaco : Tests unitaires et revue de code, 2h.
- Lucas Elisei : Conception et implémentation de l'API, 3h.
- David Truan : Interface client générale (différentes zones du GUI), 3h.
- Gaëtan Othenin-Girard : Création et gestion de la connexion au serveur, 3h.

5.2.4 Itération 4

5.2.4.1 Objectifs

Implémentation du concept de joueurs avec la création de comptes stockés dans la base de données et connexion au serveur via l'interface du client.

5.2.4.2 Temps escompté

- 1 semaine (11.05 au 18.05).

5.2.4.3 Rôles et efforts escomptés

- P-B. Monaco : Création de la base de données et ajout de la partie compte utilisateur, 3h.
- Lucas Elisei : Connexion à la base de données et affichage du pseudo sur la planète, 3h.
- David Truan : Analyse et recherche des besoins au niveau sécuritaire et tests unitaires, 2h.
- Responsable de configuration (Gaëtan Othenin-Girard) : Interface de connexion au compte utilisateur sur le client, 4h.

5.2.5 Itération 5

5.2.5.1 Objectifs

Management du serveur :

- Implémentation du système de score.
- Étude des échanges entre client et serveur (adaptation de la taille et du type de données partagées).
- Étude des flux entre les threads du serveur et optimisation.
- Tests unitaires de l'interaction client/serveur.

5.2.5.2 Temps escompté

- 1 semaine (18.05 au 25.05).

5.2.5.3 Rôles et effort escompté

- P-B. Monaco : Implémentation des scores dans l'application, 3h.
- Lucas Elisei : Tests unitaires, test de fonctionnement et revue de code, 4h.
- David Truan : Étude et optimisation des flux entre client et serveur, validation auprès des utilisateurs alpha, 4h.
- Gaëtan Othenin-Girard : Implémentation du système de score dans la base de données, 3h.

5.2.6 Itération 6

5.2.6.1 Objectifs

Management du serveur :

- Implémentation des fonctions de contrôle des joueurs (ban kick, allow, etc. . .).
- Essais d'attaques depuis le client, vérification qu'il n'est pas possible de contrer un bannissement.
- Études des différentes techniques de triches qui pourraient être utilisées (glitches et exploits).

5.2.6.2 Temps escompté

- 1 semaine (25.05 au 01.06).

5.2.6.3 Rôles et effort escompté

- P-B. Monaco : Implémentation des fonctions d'administration, 3h.
- Lucas Elisei : Étude des problèmes/risques de sécurité. Écriture d'un rapport de sécurité, 3h.
- David Truan : Étude des problèmes/risques de sécurité. Écriture d'un rapport de sécurité, 3h.
- Gaëtan Othenin-Girard : Test unitaires, vérification et validation du fonctionnement et revue de code, 2h.

5.2.7 Itération 7

5.2.7.1 Objectifs

Implémentation des modifications administrateurs en pleine partie avec l'ajout de compte "modérateur". Par exemple :

- Ajout de bonus à un endroit.
- Explosion d'une planète sélectionnée.
- Application instantanée d'un bonus sur un joueur.
- Déplacement d'un joueur en drag and drop.
- Mettre la partie en pause.

5.2.7.2 Temps escompté

- 1 semaine (01.06 au 08.06).

5.2.7.3 Rôles et efforts escomptés

- P-B. Monaco : Apparition d'objets aux endroits sélectionnés et tests unitaires, 3h.
- Lucas Elisei : Implémentation du compte spécial "modérateur" sur la base de données et en jeu, 3h.
- David Truan : Interaction directe avec la planète du joueur (déplacement et explosion), 3h.
- Gaëtan Othenin-Girard : Interface de sélection de la liste des joueurs et ajout de bonus sur le joueur sélectionné, 3h.

5.2.8 Itération 8

5.2.8.1 Objectifs

- Finalisation du projet, récupération du retard.

5.2.8.2 Temps escompté

- 1 semaine (08.06 au 15.06).

5.2.8.3 Rôles et efforts escomptés

- P-B. Monaco : Finalisation, revue de code en profondeur, 2h.
- Lucas Elisei : Finalisation, revue de code en profondeur, 2h.
- David Truan : Finalisation, revue de code en profondeur, 2h.
- Gaëtan Othenin-Girard : Finalisation, revue de code en profondeur, 2h.

5.3 Suivi du projet

5.3.1 Itération 1

5.3.1.1 Bilan

Tout bon ! Revoir simplement les cas d'utilisation et compléter les itérations 3 à 8.

5.3.1.2 Problèmes rencontrés

Aucun.

5.3.1.3 Replanification

Non.

5.3.2 Itération 2**5.3.2.1 Bilan**

Revoir la description avec la notion de gestion, infra et UC (fait partiellement ou complètement). Faire de même après en dessous avec puces. Sinon pour tout le reste, c'est ok. Gestion de la planète à compléter. Manque lorsque la souris bouge. Mettre une description plus complète pour expliquer ce que "gérer" veut dire.

5.3.2.2 Problèmes rencontrés

Aucun.

5.3.2.3 Replanification

Non.

5.3.3 Itération 3**5.3.3.1 Bilan**

Faire une séparation entre le scénario principale et échec. Mettre le/les scénarios au niveau de checkbox. Reprendre dans la description les points qui sont traités et dire dans le cas d'un point "orange", qu'est ce qui est visible dans l'ensemble.

5.3.3.2 Problèmes rencontrés

Aucun.

5.3.3.3 Replanification

Non.

5.3.4 Itération 4**5.3.4.1 Bilan**

Revoir la présentation du partage de travail, compléter les itérations jusqu'à la fin. Faire la distinction entre Gestion, fonctionnalité et infrastructure. Créer une BDD pour répondre au cahier des charges.

5.3.4.2 Problèmes rencontrés

Aucun.

5.3.4.3 Replanification

Non.

5.3.5 Itération 5

5.3.5.1 Bilan

Planification à faire pour rattraper le retard.

5.3.5.2 Problèmes rencontrés

Manque de temps pour cette itération.

5.3.5.3 Replanification

Le temps par les autres matières étant trop conséquent, nous avons préféré mettre cette itération de côté afin de pouvoir tenir la cadence. Le retard sera rattrapé les semaines suivantes.

5.3.6 Itération 6

5.3.6.1 Bilan

UC Actions sur une partie à re-planifier. Pour le reste tout est ok. A planifier aussi les points qui sont faits partiellement. Rattrapage du retard pris sur l'itération précédente.

5.3.6.2 Problèmes rencontrés

Aucun.

5.3.6.3 Replanification

Nous avons déplacer toutes l'implémentation des fonctionnalités admin dans l'itération 7 suite à notre retard pris lors de l'itération 5.

5.3.7 Itération 7

5.3.7.1 Bilan

- La création de bonus est presque terminée.
- Reste quelques bugs au niveau de la communication client/serveur pour les calculs mais la séparation est bien mieux définie maintenant.
- Les fonctionnalités admin ont été revues : plus de possibilité de ban. Un admin ne peut pour l'instant que modifier le nombre max de parties dans un lobby et accéder aux parties en tant que spectateur en voyant tous les joueurs plutôt que le top 5.
- La base de donnée est presque implémentée.

5.3.7.2 Problèmes rencontrés

Problème de communication multicast.

5.3.7.3 Replanification

Nous avons encore du édulcorer cette itération comme la 6. La gestion des bonus sera donc finie lors de l'itération 8.

5.3.8 Itération 8

5.3.8.1 Bilan

- Finalisation du projet.
- Correction de bugs majeurs.
- Rédaction du rapport.

5.3.8.2 Problèmes rencontrés

Le fait que nous ayons été mis au courant au dernier moment qu'un rapport était à rédiger, une charge de travail conséquente est venue s'ajouter à celle déjà existante mais la replanification est cette fois-ci impossible.

5.3.8.3 Replanification

Malheureusement non.

5.3.9 Bilan sur Trello

Nous avons trouvé que **Trello** n'était pas l'outil le plus adapté à ce genre de projet. Bien que le projet n'était que sur 8 semaines, le fait d'avoir utiliser une interface web pour un suivi de projet n'était pas ergonomique. Peu pratique pour avoir une vue d'ensemble des itérations, interface graphique de genre jeu-vidéo (peu épurée, commentaires perdus dans un flot d'informations). Nous aurions du avoir un apprentissage de cet outil un peu plus profond par rapport ce qui a été fait.

5.4 Stratégie de tests

Les tests ont été effectués par toute l'équipe, dans la mesure du possible. Ces derniers ont été réalisés dans leur intégralité grâce à la librairie JUnit. Quant aux résultats des tests, tous ont été validés par Travis lors du push sur les différentes branches.

5.5 Stratégie d'intégration

Comme expliqué précédemment, Git a été utilisé pour intégrer le travail de chacun des membres du groupe. Chaque membre avait sa propre branche. Le merge d'une branche sur *master* était fait à chaque fois qu'une nouvelle fonctionnalité était implémentée et fonctionnelle. Travis nous permettait de savoir si le merge allait casser la compilation de la branche principale, au quel cas nous résolvions les conflits avant de merge.

6 État des lieux

6.1 Ce qui fonctionne

- Lobby fonctionnel avec un nombre de parties maximum.
- Nombre de joueurs maximum dans une partie.
- Création de parties.
- Déplacement de la planète grâce à la souris : lorsque le joueur fait un geste de frottement à l'aide de sa souris, cela génère une forte gravité par rapport à la planète du joueur.
- Score se basant sur la masse de la planète du joueur. Mise à jour en temps réel du score des 5 meilleurs joueurs, visible par chaque joueur.
- Bonus offrant des pouvoirs :
 - Lune amassant des fragments pour nous (non fonctionnel mais offre une expérience visuelle sympathique car la lune est quand même là).
 - Atmosphère protectrice (fonctionnel).
- Zoom/De-zoom avec les touches **Q** et **E** et la molette de scroll.
- Déplacement de la caméra avec les flèches directionnelles et WASD.
- Modification de la masse de la planète de contrôle.
- Administration. Au niveau du serveur, un administrateur doit pouvoir effectuer certaines tâches :
 - Gérer le nombre de parties max du lobby.
 - Être spectateur d'une partie en ayant la visibilité sur tous les joueurs.

- Choix d'une texture visuelle de la planète contrôlée.
- Gestion concurrente du maximum de joueurs possibles (si possible, pas de limite).
- Sons des bonus.

6.2 Développement futur

Pour le développement futur de l'application, nous pensons que les points suivants sont primordiaux :

- Revue de l'architecture du code. Notre code est un peu chaotique dans certaines classes, typiquement au niveau du client. Cela prendrait un temps assez conséquent car nous avons du développer tout le projet sur une base assez fragile. Nous planifions cela à 2 à 3 semaines de travail.
- Amélioration de la communication réseau. Nous utilisons actuellement du multicast pour envoyer à tous nous client l'état de l'univers mais cela pose des problèmes de latence lorsque nous jouons en wi-fi. Nous pensions redévelopper le serveur en *JavaScript* à l'aide de **NodeJS**. Cela prendrait bien entendu un temps conséquent et nous planifierions ça à hauteur de 2 voir 3 semaines pour rendre le réseau totalement stable en devant repartir de zéro avec le serveur. Cela pourrait être fait en parallèle de la restructure de l'architecture du code.
- Ajout de bonus. Dans notre programme nous n'avons pas eu le temps d'implémenter tous les bonus que nous voulions. L'ajout de nouveau bonus/malus (masse x10, zone de ralentissement, ...) ne prendrait pas beaucoup de temps et serait donc réalisable en 1 semaine si cela était à planifier.
- Ajout de fonctionnalités admin. Notre programme ne propose que très peu de fonctionnalités aux admin :
 - Modification du nombre de parties maximum dans le lobby.
 - Devenir spectateur d'une partie et voir la liste de tous les joueurs de cette partie.

Nous avons prévu de permettre aux admins de bannir des joueurs, d'ajouter des bonus manuellement et d'autres fonctions spécifiques à l'administration du jeu. Il faudrait aussi pouvoir s'enregistrer en temps qu'admin d'une façon plus sécurisée (mot de passe). Comme la connexion admin est déjà gérée, l'ajout de ces fonctionnalités ne devrait pas être trop long. 1 à 2 semaines devraient être suffisantes pour la planification.

7 Auto-critique

7.1 Solution technique

Du point de vue technique, nous aurions sûrement dû réfléchir un peu plus longuement sur l'architecture de l'application. Notre solution actuelle fonctionne mais l'implémentation n'est pas vraiment propre. La gestion de la communication réseau est aussi à revoir. Nous avons eu plusieurs problèmes concernant la communication multicast qui n'auraient pas lieu d'être si une autre approche avait été utilisée dès le départ.

7.2 Gestion du projet

La gestion du projet s'est très bien déroulée. Les deadlines ont été respectées dans la quasi-totalité des cas. Le chef de projet avait plus le rôle de leader que de chef ce qui fait qu'aucun sentiment de hiérarchie n'a été ressenti. Chacun a effectué son travail correctement. Nous avons bien su gérer les replanifications sur la fin du projet.

7.3 Plan d'itérations

Notre plan d'itérations initial a dans l'ensemble été respecté malgré quelques réajustements qui ont été nécessaires en cours de route. Le fait d'avoir eu un semestre aussi chargé nous a mis en retard à certains moment mais nous avons su replanifier nos itérations lorsque cela était le cas.

7.4 Améliorations

Si ce projet était à refaire nous changerions certains points :

- Ne pas utiliser **Trello**. Nous avons vraiment trouvé cet outil peut pratique à utiliser, peu ergonomique et pensons que si nous avions un suivi de projet à refaire nous utiliserions une application plus spécifique (pas d'interface web en tout cas).
- S'y prendre plus à l'avance pour certaines itérations ou nous avons coder très vite par manque de temps. Cela à fait que le code est fonctionnel mais largement améliorable.
- Plus vite réfléchir en mode client-serveur que ce que nous avons fait. Nous avions un engine graphique fonctionnel assez tôt mais de l'adapter à une version client-serveur à pris bien plus de temps que prévu.

8 Conclusion

Ce projet, bien que partant d'une bonne idée du fait que l'on doive développer un jeu que l'on puisse choisir avec qui nous travaillons, a été plombé par le fait que le suivit était plus une contrainte qu'autre chose. De plus, la charge de travail qu'une application client-serveur représente et le nombre de laboratoires conséquent dans les autres branches nous a pénalisé pour l'avancement du projet. Comme dit précédemment, nous n'avons pas réussi à appréhender correctement **Trello** et l'avons plus vécu comme un boulet attaché au pied plutôt qu'un moyen de bien suivre notre projet. Au final nous avons réussi à rendre une application fonctionnelle en remplissant presque tous les points du cahier des charges que nous avons fixé au départ.