Projet long de Technologie Objet

Rapport général

Itération n°1

Groupe IJ-1

Fainsin Laurent
Guillemin Johan
Guillotin Damien
Heurtebise Tom
Jourdan Pierre-eliot
Kirupananthan Nadesan

Département Sciences du Numérique
Première année
2020 — 2021

Table des matières

1 Rappel du projet

Sagittarius est un jeu type arcade tour par tour pouvant être joué de 2 à 5 joueurs. Le but d'un joueur est d'éliminer ses adversaires grâce à un arc, des flèches et, la mécanique principale du jeu, la gravité !
Ce projet a pour but de recréer ce jeu existant tout en l'améliorant par divers ajouts/fonctionnalités. Nous n'en revendiquons pas la propriété intelectuelle même si nous n'avons pas consulté le code source pour élaborer notre projet !

2 Principales fonctionnalités

La fonctionnalité principale de l'application est de pouvoir jouer à une ou plusieurs parties (sans remdémarrer le jeu).

Par le biais d'un menu dédié, l'utilisateur a aussi la possibilité de pousser la personnalisation d'une partie tel que les caractéristiques visuelles et physiques. Ceci inclus la sélection par les joueurs de la couleur de leur archer et leur noms. Notons au passage que si deux joueurs choisissent la même couleur, ceux-si sont considérés comme dans la même équipe.

Un autre menu permet de mettre la partie qui était en cours en pause. L'utilisateur peut alors au choix reprendre, sauvegarder la partie ou alors quitter pour revenir au menu principal.

Enfin un dernier menu permet de régler les paramètres généraux du jeu tel que la résolution, le nombre d'image par secondes, le vsync, le niveau de détails...

3 Gestion Agile du projet avec Vision de l'application

Durant les séances dédiées à l'applicaton des méthodes de projet Agile nous avons pu découper notre application en fonctionnalités elle même redécoupées et ainsi de suite jusqu'à obtenir des "Users stories". Sur la page suivante vous pourrez retrouver le découpage que nous avons effectué.

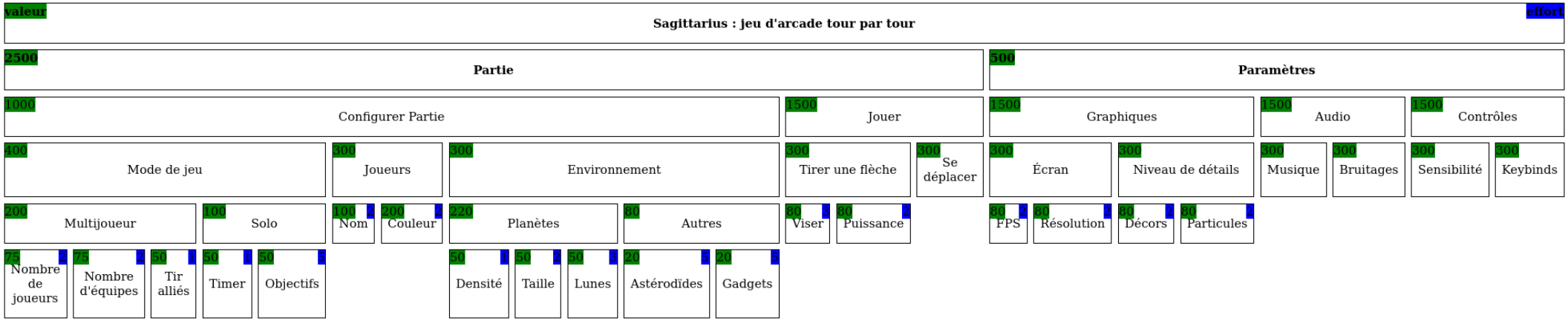


FIGURE 1 – Découpage de la vision du projet jusqu'au Users Stories (voir Méthodes Agiles pour le détail des termes)

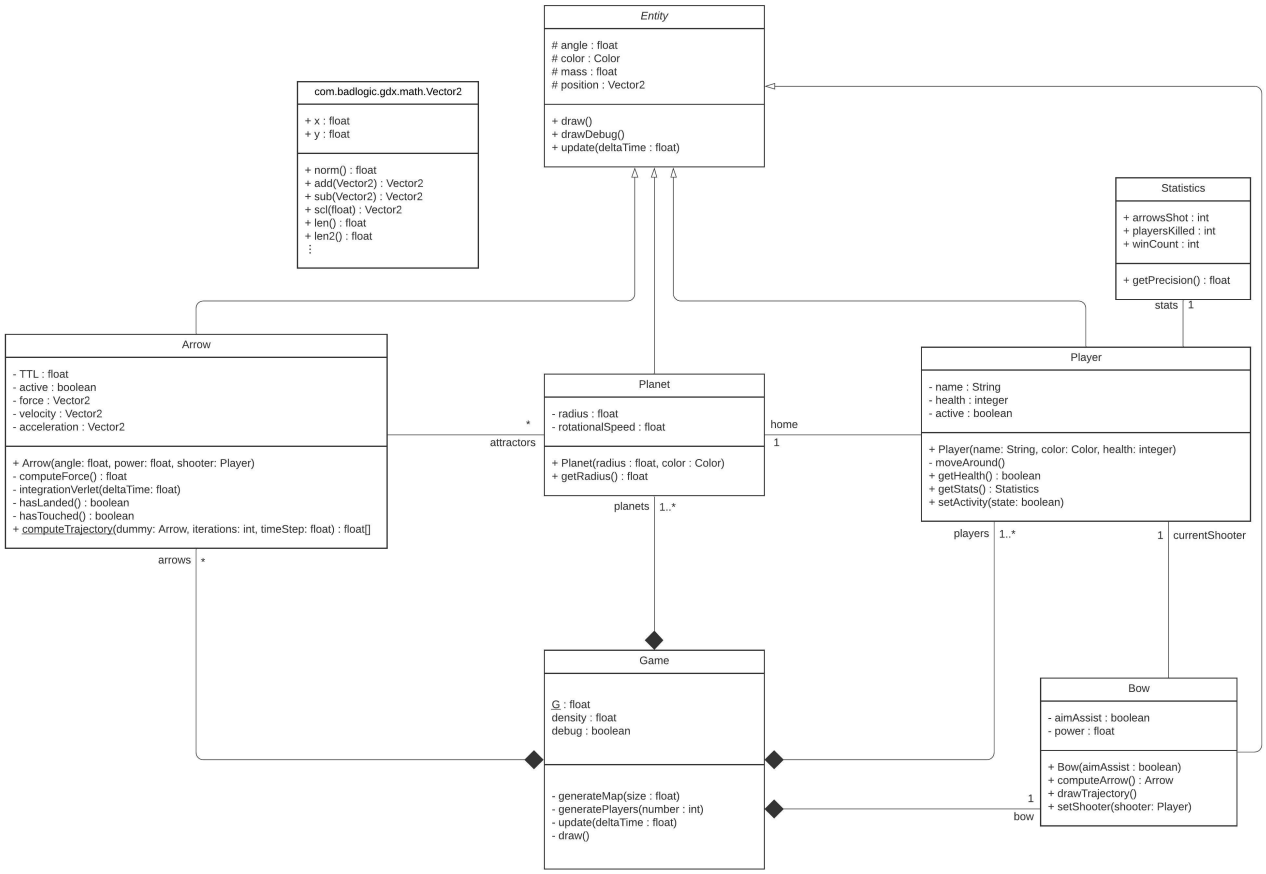
Pour la première itération nous avons du poser les bases du projet en nous attaquant aux Users stories "Viser", "Tirer" et "Se déplacer". Nous avons aussi traiter des aspects lié à la génération de l'environnement du jeu (ce qui en soit ne fait pas partie des users stories). Pour le moment les points d'efforts nous semblent plutôt bien réparti puisque nous sommes parvenus à terminer ces deux premières Users Stories avec un affichage relativement sommaire mais un affichage tout de même et la possibilité de jouer. Par la suite nous allons nous attaquer à améliorer cet affichage et géré au mieux la partie conifguration de partie dans laquelle nous avons diverses users stories.

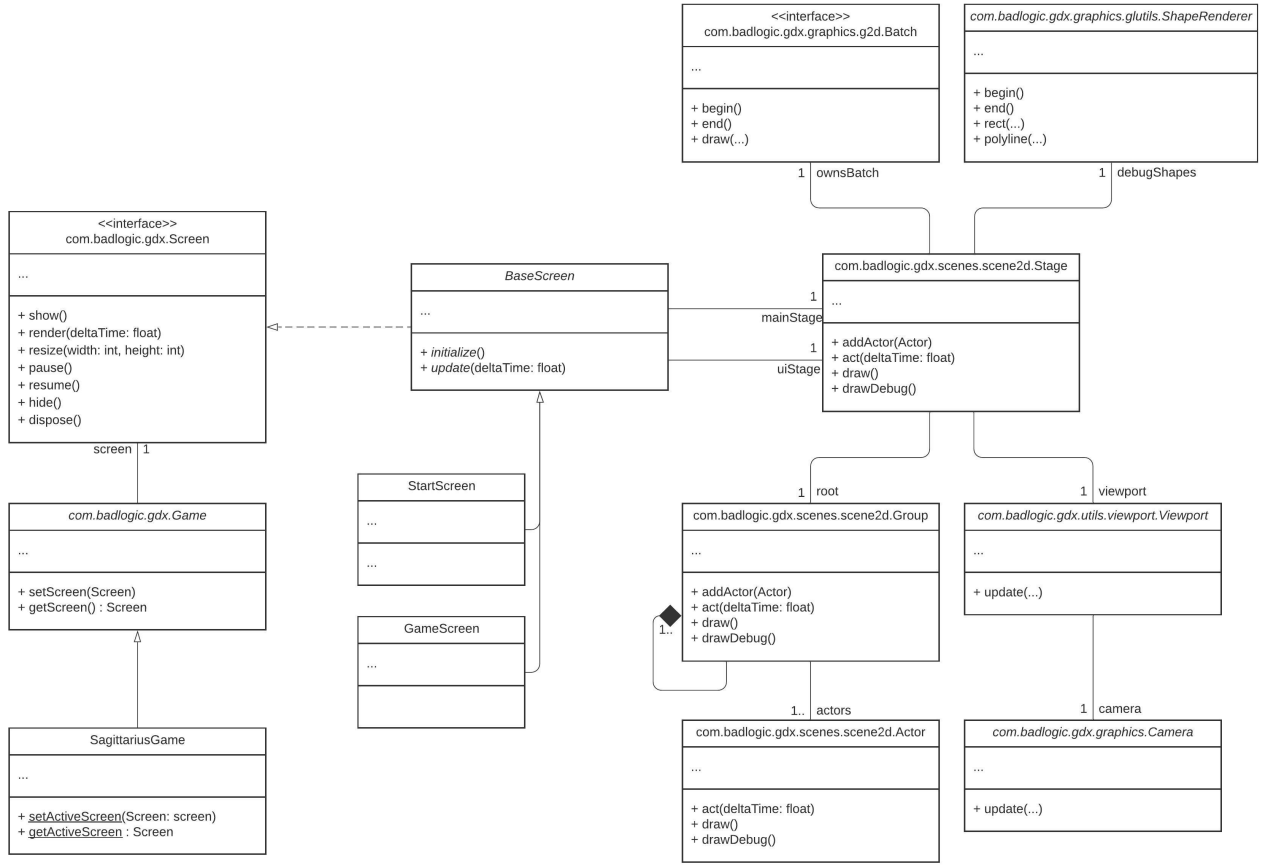
4 Découpage de l'application et application du MVC (Modèle Vue Contrôleur)

Le jeu est actuellement divisé en deux parties, le modèle physique du jeu, et la partie visuelle du jeu. Nous avons fait ce choix pour rendre le jeu le plus indépendant possible d'une interface graphique. Ainsi, bien que nous utilisions la bibliothèque spécialisée dans la création de jeux vidéo 2D libGDX, il est possible de l'interchanger avec une bibliothèque plus traditionnelle telle que Swing. Notons toutefois qu'après discussion entre nous il nous a paru relativement difficile de séparer complètement la partie modèle physique du jeu de libGDX qui apport des fonctionnalités permettant d'améliorer les performances et qui offre un panel de possibilités plus large que si nous réécrivions toutes ces méthodes.
Pour ce qui est du respect strict du MVC nous avons eu beaucoup de difficultés à séparer le moteur du Jeu de sa vue et nous n'y sommes pas parvenu de façon totale. Cependant dans la mesure ou il s'agit d'un jeu dont la vocation n'est pas de changer d'interface graphique souvent cela ne nous semble pas complètement abberrant non plus.
ENfin notons que la partie contrôleur du jeu est celle qui est permise par libGDX, elle ressemble très fortement à celle de Swing mais offre plus de possibilités.

5 Diagrammes de classe

Voici des diagrammes UML pour vous aider lors de la lecture du code.





6 Choix de conception et réalisation

Un choix de conception important que nous avons du réalisé et qui ne respecte pas bien le principe du MVC et le fait que nous ayons inclus des méthodes draw dans la classe entité. Ceci est dû essentiellement à l'utilisation de la librairie dont nous avons déjà fait mention et dont vous pourrez retrouver le principe dans le diagramme UML ci-dessus. Notons toutefois que cette méthode n'impacte pas ce que le modèle physique réalise, le MVC est donc en partie respecté.
Un point qui peut être délicat est la façon dont nous calculons la trajectoire d'une flèche. Nous avons d'abord pensé à générer une carte comportant les champs d'attraction mais cela nous a paru peu efficace en terme d'implémentation, de temps de calcul et de stockage (puisque recalculer cette carte si on y inclue des objets mobiles tels que des lunes est très lourd). Après quelques recherches nous avons trouvé une méthode d'intégration physique qui permet de déterminer la trajectoire de particules qui sont soumises à différentes forces. Nous nous sommes appuyés sur diverses ressources dont celle-ci : https ://www.compphy.com/verlet-algorithm/.

7 Points à améliorer pour la prochaine itération

Lors de la réalisation de la deuxième version sur Swing que nous implémentant à titre de démonstration seulement, nous nous sommes aperçus de plusieurs défauts de conception. Le premier a déjà été évoqué, il s'agit du lien trop fort entre vue et modèle qui est induit par la bibliothèque déjà mentionnée.
Un autre défaut de cinception est la trop forte encapsulation des paramètres de partie. Par exemple la constante G se trouve dans la classe Game alors que le TTL des flèches se trouve dans la classe flèche. Même si cela nous a paru logique au moment de l'élaboration du diagramme UML, il s'avère que modifier ces paramètres depuis un menu relève du casse-tête.

8 Explications supplémentaires

Nous avons préféré utiliser certains outils comme Visual Code et GitLab qui permettent un travail collaboratif plus facile qu'Eclipse et svn. Nous allons continuer à procéder de la sorte.