Projet long de Technologie Objet

Rapport général

Itération n°é

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

cette section est identique au premier rapport

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 intellectuelle et nous n'avons pas consulté le code source pour élaborer notre projet !

2 Principales fonctionnalités

cette section est identique au premier rapport

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 (fonctionnalité non implantée pour le moment).

Un autre menu permet de mettre la partie qui était en cours en pause. L'utilisateur peut alors au choix reprendre, changer quelques paramètres comme la musique 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... Notez que en l'état actuel des choses tous ces paramètres ne sont pas encore modifiables.

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.

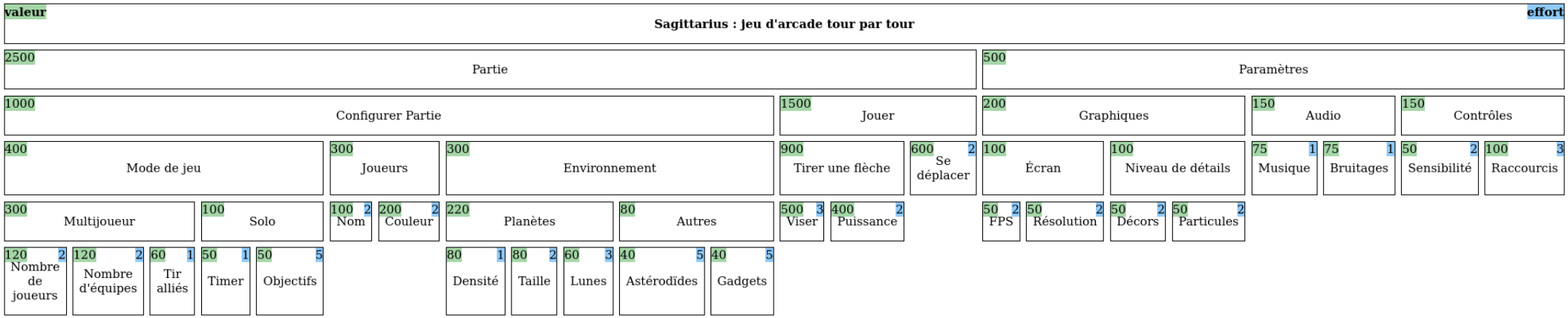


FIGURE 1 – Découpage de la vision du projet jusqu'au Users Stories (voir Méthodes Agiles pour le détail des termes)

Pour la deuxième itération, puisque nous avions posé les bases du projets et que nous avions pu obtenir un rendu preléminaire à l'aspect "débugage", nous nous sommes concentrés sur les aspects esthétiques du projet. Dans le cadre du MVC nous pourrions dire que nous nous sommes attachés à améliorer la vue pendant cette itération. Ainsi nous sommes passés de ce rendu :

FIGURE 2 – Premier version du jeu, aspect debug

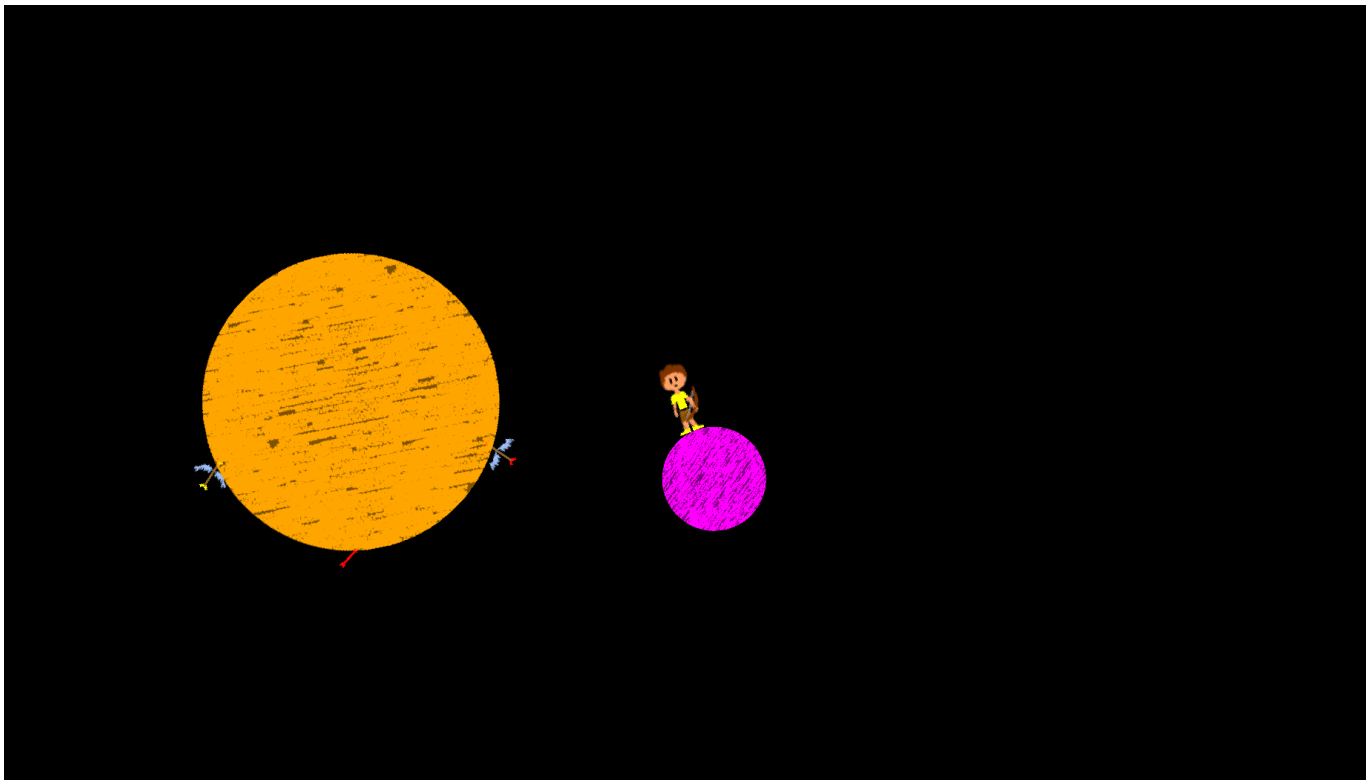


FIGURE 3 – Deuxième bersion, aspect debug

au rendu ci-dessus.

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

cette section est presque identique au premier rapport

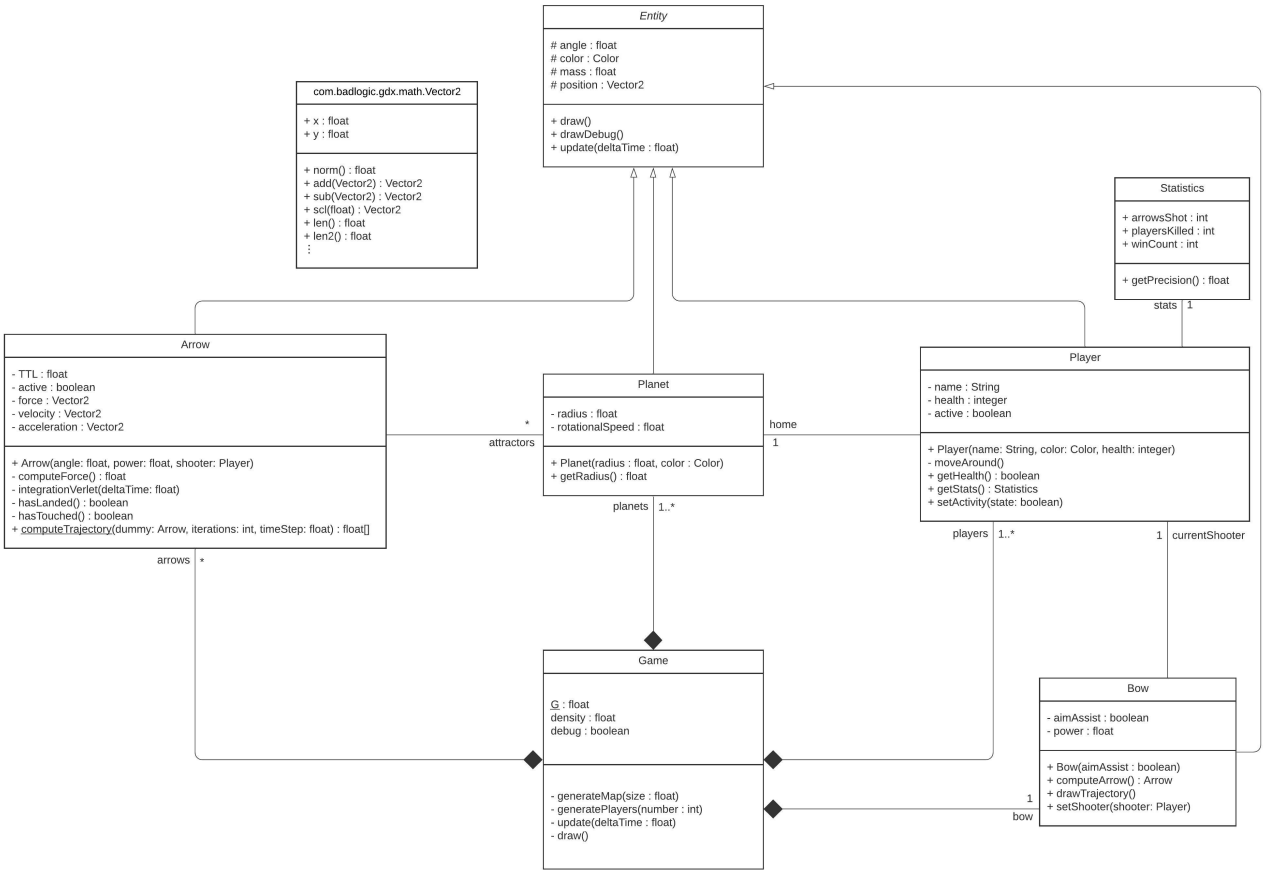
Le jeu est actuellement divisé en deux parties, le modèle physique du jeu, et la partie visuelle du jeu. Nous avons fait ce choix pour rendre le jeu le plus indépendant possible d'une interface graphique. Vous noterez toutefois une très forte dépendance entre le modèle physique et la bibliothèque LIBGDX. En effet, il nous a paru relativement difficile de séparer complètement la partie modèle physique du jeu de libGDX qui apporte des fonctionnalités permettant d'améliorer les performances et qui offre un panel de possibilités plus large que si nous réécrivions toutes ces méthodes.

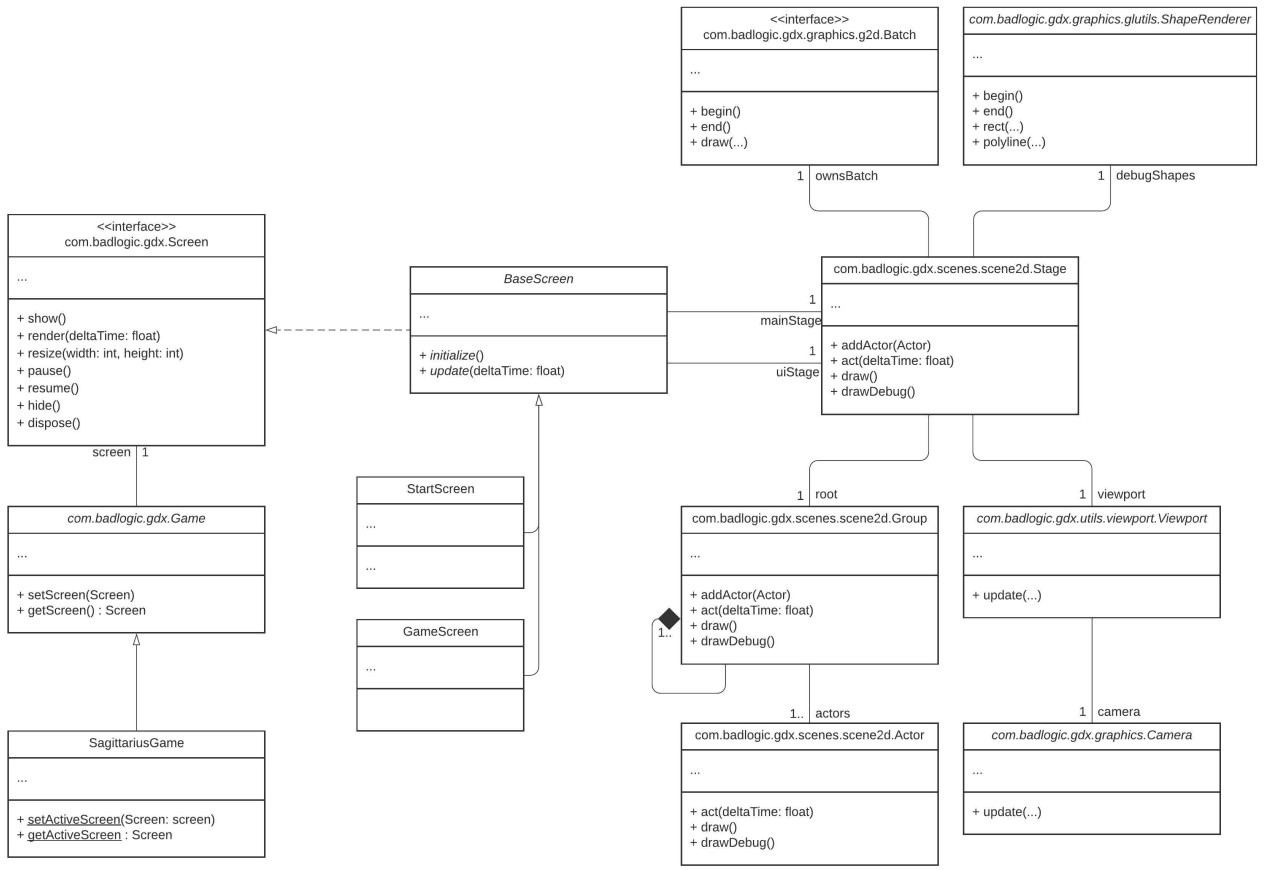
Pour ce qui est du respect strict du MVC nous avons eu beaucoup de difficultés à séparer le moteur du Jeu de sa vue et nous n'y sommes pas parvenu de façon totale. Cependant dans la mesure ou il s'agit d'un jeu dont la vocation n'est pas de changer d'interface graphique souvent cela ne nous semble pas 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 en se rapporchant plus de l'aspect bas niveau (possibilité d'interagir directement avec le processeur par exemple).

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 est le fait que nous ayons inclus de nombreuses méthodes de LIBGDX dans la plupart de nos classes. Nous avons déjà abordé ce point dans le rapport précédent.
Au cours de cette itération nous avons d'avantage développée la vue et nous nous sommes aperçu qu'il est très difficile dans la rendre indépendante du modèle physique. En effet prenons l'exemple du "skin" des personnages. Il parait contreintuitif de ne pas placer associer la tenue d'un joueur au joueur lui-même. C'est pour cette raison que la gestion de l'affichage des tenues se fait dans la clasee Player. Nous aurions très bien pu placer cette gestion dans une classe à part mais le problème restera le même : il est impossible de séparer totalement vue et modèle dans notre cas.
Un autre exemple qui vient illustrer cette idée est l'utilisation des musiques. En effet leur implantation est disséminée un peu dans toutes les classes du projet. Vous en retrouverez dans les classes associées aux menus (dans la partie vue) mais aussi dans certaines classes qui réalisent des actions spécifiques (comme les joueurs). Ce choix a été fait (et n'est pas définitifà afin de ne pas multiplier de façon exponentielle le nombre de classes et le nombre de Listener. De plus concernant les musiques, notez que changer de moteur graphique ne pose pas de souci puisqu'il est possible de les désactiver par un booléen dans la classe SagittariusGame.java.

7 Points à améliorer pour la prochaine itération

Il semble que dans certaines classes des redondances apparaissent. On peut citer l'exemple de la gestion de la musique qui se retrouve un peu partout et avec un code relativement similaire ou encore les deux menus qui gèrent les paramètres dont la ressemblance est asse frappante. A ce stade là il nous faudrait faire un bilan du diagramme UML pour voir ce qui peut être réifier.