UNIVERSITÉ DE NANTES

Cahier des charges de B engine

Thibaud Courtoison
Dylan Gautier
Mathieu Grondin
Thomas Picard
Hugo Pigeon
Maël Quémard

4 avril 2014

1. Présentation générale du problème

1.1 Projet

1.1.1 Finalités

L'objectif de ce projet est de réaliser un prototype d'un jeu de plateforme en 2D en java se basant sur un moteur de jeu que l'on va faire.

1.1.2 Espérance de retour sur investissement

- Projet Open Source
- Etablissement d'une communauté (github)
- Première expérience dans l'élaboration d'une IHM
- Projet modulaire
- Connaissance sur le développement d'un jeu vidéo et ses contraintes propres

1.2 Contexte

1.2.1 Situation du projet par rapport aux autres projets du groupe

Tous les membres du groupe ont déjà codé le jeu d'Awalé et le jeu du taquin en C en mode console, et nous allons bientôt coder un jeu d'échecs martiens en java. Ces projets nous permettent de connaître l'‘utilisation des matrices ainsi que la décomposition des classes d'un jeu.

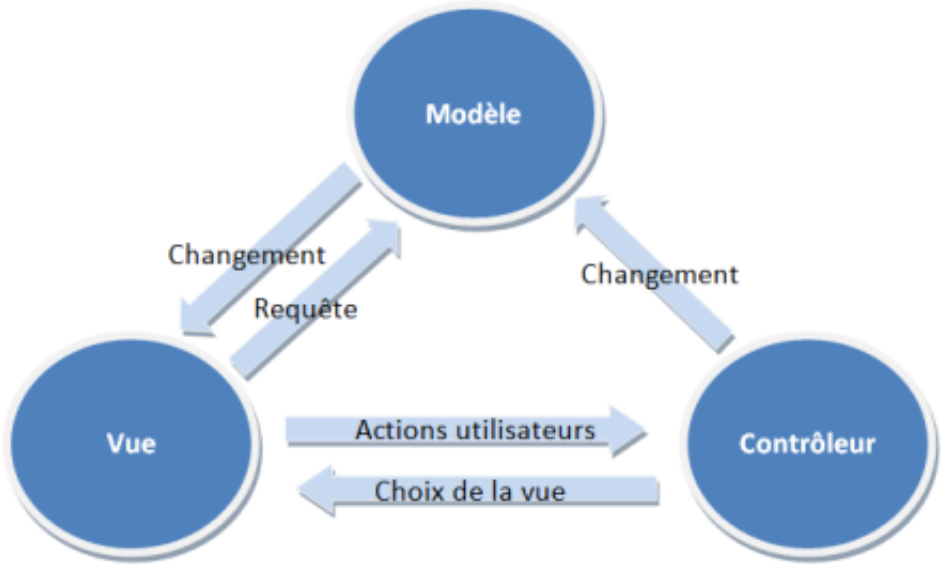
Hugo a déjà créé un jeu "snake" en 2D et le langage utilisé dans ce projet est le C++, ce qui représente une certaine expérience de la création d'un jeu avec un langage orienté objet. Thibaud a déjà programmé un "Guitar Hero like" en 2D en java, ainsi qu'un début de Shoot'm up en scrolling horizontal.

1.2.2 Études déjà effectuées

- **Etude des collisions**
  `http://fr.openclassrooms.com/informatique/cours/theorie-des-collisions`

Il existe plusieurs types de collisions. Le plus simple et le plus rapide dans son exécution est le AABB. Il s'agit de créer un rectangle autour de chaque entité, les collisions seront détectées à l'intérieur de ce rectangle. On peut également choisir de gérer les collisions au pixel près, mais le traitement est plus lourd. Pour optimiser le traitement, il est possible de diviser la carte du niveau en plusieurs parties et seule la partie contenant le joueur sera traitée.

- **Gestion d'une carte ASCII**
  `http://fr.openclassrooms.com/informatique/cours/tile-mapping`
  Ce tutoriel explique les notions de "Tile mapping", de "Scrolling" et de déplacement d'un personnage en suivant la gravité. L'avantage de ce tutoriel est qu'il donne des bouts d'algorithmes permettant de comprendre ces notions à un niveau informatique.
- **MVC**
  `http://www.u-picardie.fr/~furst/docs/3-%20Event_MVC.pdf`
  Le MVC (modèle vue contrôleur) consiste à séparer en trois parties les classes d'un programme utilisant une interface graphique. La partie modèle contient toutes les données, la vue consiste à afficher ces données et à détecter les événements (clavier ou souris par exmple). Le contrôleur reçoit les notifications de la vue pour les transmettre au modèle. Cette architecture permet d'avoir un code très modulaire, ont peut facilement créer plusieurs vues sans modifier les autres parties par exemple.



- **Programmation orientée composants**
  `http://perso.citi.insa-lyon.fr/sfrenot/cours/cours01/POO/cours/Composant.pdf`
  La programmation orientée composants est une technique de programmation plus

abstraite que l'orienté objet. Il s'agit de programmer en faisant en sorte que les fichiers compilés soient réutilisables facilement dans d'autres projets. L'objectif est de disposer de programmes très modulaires qui ne diffèrent que par leur implantation.

1.2.3 Études menées sur des sujets voisins

Nous ne menons aucun projet voisin.

1.2.4 Suites prévues

- Création d'un éditeur de cartes.
- Proposer un export des cartes pour les partager.
- Ajout de niveaux dans le jeu.
- Ajouter une dimension scénaristique au jeu.

1.2.5 Nature des prestations demandées

La prestation demandée est un service, dans le but d'offrir au demandeur un prototype de jeu.

1.2.6 Parties concernées par le déroulement du projet et ses résultats (demandeurs, utilisateurs)

Le demandeur de ce projet est Adrien Bougouin. Les utilisateurs sont les fans de jeux vidéos rétros.

1.2.7 Caractère propre à la diffusion

Ce projet sera distribué sous licence Creative Commons avec attribution (`http://creativecommons.org/licenses/by/2.0/fr/`)

1.3 Énoncé du besoin (finalités du produit pour le futur utilisateur tel que prévu par le demandeur)

L'objectif de ce projet est de réaliser une première version d'un jeu de plates-formes 2D en Java. Dans cette version, un joueur peut se mouvoir sur des plates-formes.

Afin de faciliter la transition de ce prototype (travail demandé) vers une version multi-joueurs et multi-niveaux (travail non demandé, sauf souhait des étudiants), l'aspect modulaire de l'architecture du jeu est très importante. De même, une réflexion sur les patrons de conception à appliquer est attendue.

Dans le but de valider la modularité de leur architecture, les étudiants doivent proposer deux vues (simultanées) et deux gestionnaires de collisions. Le résultat des vues est similaire, mais visuellement différent :

- Affichage 2D classique
- Affichage textuel du déroulement du jeu

Un gestionnaire de collisions, parmi les deux, est sélectionné au lancement du niveau :

- Collisions classiques
- Collisions dans un "monde en caoutchouc"

Pour les gestionnaires de collisions, les étudiants doivent s'appuyer sur des algorithmes existants.

1.4 Environnement du produit recherché

1.4.1 Listes exhaustives des éléments (personnes, équipements, matières...) et contraintes (environnement)

6 étudiants : Thibaud Courtoison, Mathieu Grondin, Thomas Picard, Dylan Gautier, Hugo Pigeon, Maël Quémard.

Le jeu sera disponible sur plusieurs OS.

Le jeu sera développé avec le langage de programmation Java

Nous utiliserons des packs graphiques en licences libres.

1.4.2 Caractéristiques pour chaque élément de l'environnement

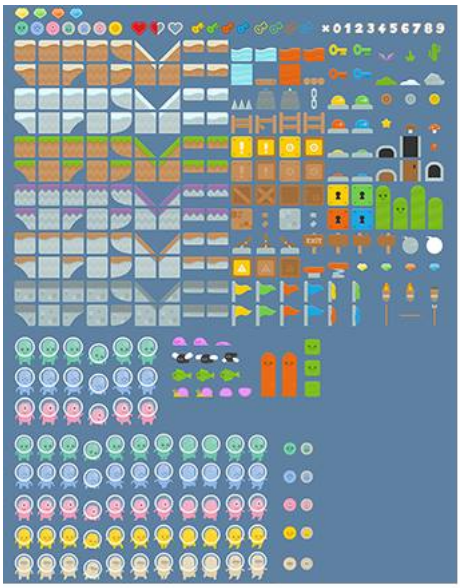
- Responsable de l'éditeur de map - Hugo
- Responsable Vue - Dylan
- Responsable Modèle - Thomas
- Responsable du moteur physique - Mathieu
- Responsable Contrôleur - Maël
- Responsable Communication Interne/Externe - Thibaud

Le jeu sera disponible sur Linux, Windows et MacOS X.

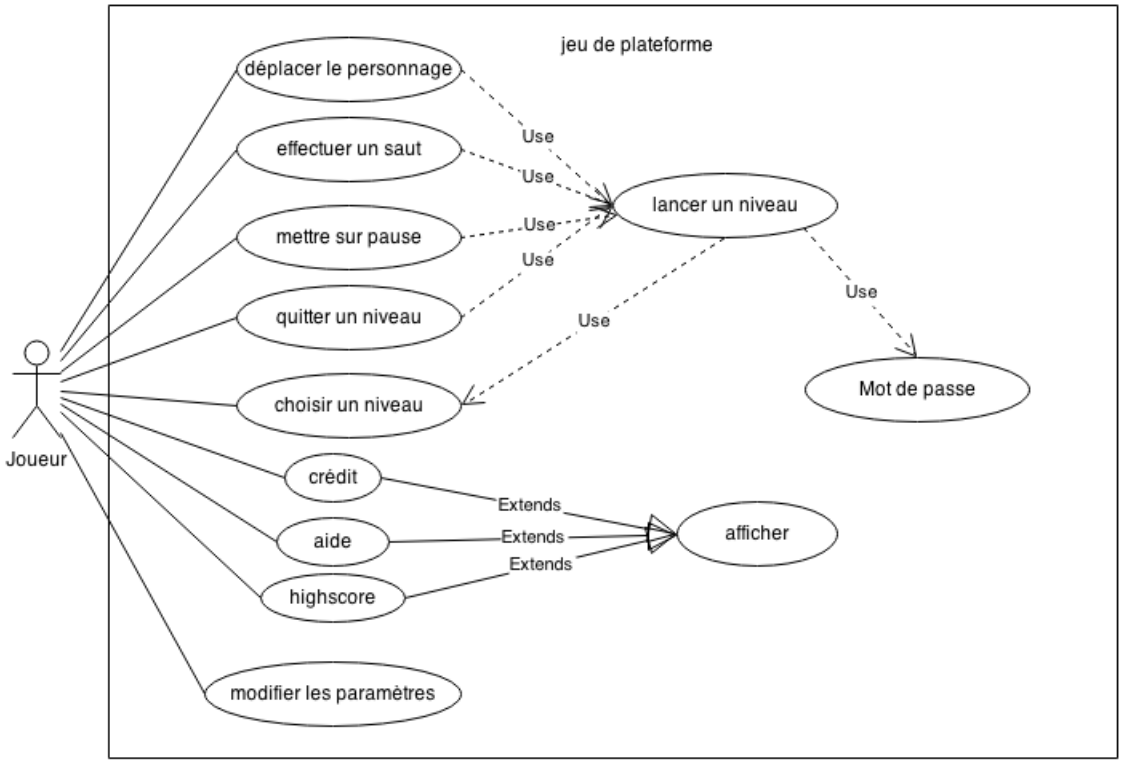
Les packs graphiques suivants seront utilisés pour la conception du jeu :

- UI pack : `http://opengameart.org/content/ui-pack`

– Platformer pack : `http://opengameart.org/content/platformer-art-complete-pack-often-updated`



2. Expression fonctionnelle du besoin



2.1 Fonctions de service et de contrainte

2.1.1 Fonctions de service principales (qui sont la raison d'être du produit)

Les services principaux du prototype de jeu sont basés sur :

- Un moteur de jeu (modèle MVC)
- Modularité et évolution
- 2 méthodes de collisions (normal et en caoutchouc)
- 2 vues (console + graphique)
- Une carte ascii

Exemple de carte ascii :

```
Tilemapping Version 1.0
#tileset
tileset1.bmp
8 1
tile0: vide
tile1: plein
tile2: plein
tile3: plein
tile4: plein
tile5: plein
tile6: plein
tile7: plein
#level
15 13
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 3 4 0 0 0 2 2 2 2 0 0 2 2
0 0 5 6 0 0 0 0 0 0 0 0 0 0 0
0 0 5 6 0 0 0 0 0 0 0 0 0 0 0
0 0 5 6 0 0 0 0 0 0 0 0 0 0 0
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
#fin
```

2.1.2 Fonctions de service complémentaires (qui améliorent, facilitent ou complètent le service rendu)

Parmi les fonctions complémetaires, la conception d'un éditeur de carte ASCII pourrait être un plus à ce projet.

2.1.3 Contraintes (limitations à la liberté du concepteur-réalisateur)

Les graphismes, le nom du jeu et le gameplay ont une limitation imposée.
Utilisation de Github (`https://github.com/ErrOrnAmE/S2Game`)

2.2 Critères d'appréciation (en soulignant ceux qui sont déterminants pour l'évaluation des réponses)

- Un déplacement fluide
- Une bonne résolution
- Présence d'items sur la carte (au moins un)
- Système de mot de passe pour chaque niveau
- Un niveau en collision caoutchouc
- Présence d'un menu
- Mode console et mode graphique
- Modularité et évolution
- La distribution de sources commenté, ainsi qu'une documentation est censée permettre la modularité et une évaluation
- Ajout de plusieurs niveaux
- Ajout de textures (HD ou rétro)

2.3 Niveaux des critères d'appréciation et ce qui les caractérise

2.3.1 Niveaux dont l'obtention est imposée

La réalisation des fonctionnalités suivantes est imposée :
- Un déplacement fluide
- Une bonne résolution
- Présence d'items sur la carte (au moins un)
- Système de mot de passe pour chaque niveau
- Un niveau en collision caoutchouc
- Présence d'un menu
- Mode console et mode graphique

Nous n'accorderons cependant pas une grande importance au son.

2.3.2 Niveaux souhaités mais révisables

- Modularité et évolution
- La distribution de sources commenté, ainsi qu'une documentation est censée permettre la modularité et une évaluation

- Ajout de plusieurs niveaux
- Ajout de textures (HD ou rétro)

3. Cadre de réponse

3.1 Pour chaque fonction

3.1.1 Part du prix attribué à chaque fonction

Fonction : temps estimé ⇒ coût ⇒ rémunération
Jeu de base : 3 semaines ⇒ 7500€ ⇒ 1143,60€/personne.
Packs de texture : 1 semaine ⇒ 500€ ⇒ 76,24€/personne.
Niveau(x) “caoutchouc” : 1 semaine ⇒ 500€ ⇒ 76,24€/personne.
Nouvelle(s) carte(s) de jeu : 1 semaine ⇒ 500€ ⇒ 76,24€/personne.
Affichage console : 1 semaine ⇒ 500€ ⇒ 76,24€/personne.
Implanter une nouvelle résolution graphique : 1 semaine ⇒ 500€ ⇒ 76,24€/personne.

3.2 Pour l’ensemble du produit

3.2.1 Prix de la réalisation de la version de base

Le prix de réalisation pour la version de base est de 7500€ toutes taxes comprises.

3.2.2 Options et variantes proposées non retenues au cahier des charges

Les options suivantes n’ont pas été retenu au cahier des charges :
– Possibilité de jouer en résau
– Présence d’une Intelligence Artificielle
– Possibilité de jouer en multijoueur

Nous n’avons pas retenu ces fonctionnalités car leur réalisation nous semble difficile et coûteuse en temps. Nous préfèrons nous concentrer sur les bases du moteur de jeu pour qu’il soit pleinement fonctionnel.

3.2.3 Mesures prises pour respecter les contraintes et leurs conséquences économiques

Le respect des contraintes est évalué par une note de travail.

3.2.4 Outils d'installation, de maintenance... à prévoir

Utilisation de Ant : `http://ant.apache.org/`
Création d'une documentation pour le moteur de jeu
Création d'un manuel d'utilisation pour le jeu

3.2.5 Décomposition en modules, sous-ensembles

Respect des patrons de conception MVC et Observeur/Observable

3.2.6 Prévisions de fiabilité

Aucune conséquence néfaste pour l'utilisateur ainsi que pour le système.

3.2.7 Perspectives d'évolution technologie

- Ajout du multijoueur
- Multilangue
- Mode 18+ (+ de sang, plus de "sous-entendu")
- Jeu en réseau
- Ajout d'une Intelligence Artificielle
- Génération de map aléatoire
- Ajout d'élément décoratif (nuages)
- Ajout de Warpzones