Rapport projet PFA

Erwan LEMATTRE Ewen DUFOUR

Avril 2024

1 Le jeu

Le jeu *Arthur la quête de la cuillère* a pour but d'offrir aux joueurs une expérience de jeu agréable avec différents niveaux ayant chacun sa spécificité tout en racontant une histoire. L'histoire est le fil conducteur du jeu : elle permet de donner un sens aux différents niveaux.

Arthur la quête de la cuillère s'inspire en partie de la légende du roi Arthur et de la série *Kaamelott*. Nous avons essayé d'ajouter des références à la série tout en gardant un jeu cohérent pour les non-connaisseurs. On retrouve également une référence à *Star Wars* avec les opening des niveaux inspirés de ceux des films. Enfin, le joueur est accompagné d'une série de musiques médiévales tout au long du jeu qui le plonge d'avantage dans l'aventure du roi arthur.

Le jeu se compose de 4 niveaux et d'un menu. Le **menu** sert uniquement d'accueil du jeu. Le **niveau 1** introduit la problématique de l'histoire (opening) et permet au joueur de prendre en main le jeu avec un niveau plutôt simple. Le **niveau 2** lance réellement la quête du roi. Le joueur ayant une maîtrise des commandes du jeu, il doit maintenant parcourir un niveau plus difficile. On reste dans le même environnement plutôt plaisant. Avec le **niveau 3** les choses se compliquent. L'environnement devient plus hostile et la difficulté augmente. Le sol est maintenant glissant, les blocs sont plus espacés, certains blocs sont invisibles et les ennemis sont plus nombreux. Enfin le **niveau 4** est le niveau final dans lequel le joueur doit faire face au boss final. L'environnement devient lugubre, la musique indique au joueur la bataille finale. Une fois le squelette éliminé la quête est terminée, on retourne au menu.

Il existe 3 ennemis différents dans notre jeu. Les archers (niveaux 2 et 3), les chevaliers (niveaux 2 et 3) et **Alexandre Le Petit** l'ennemi final (niveau 4). Les **archers** ne se déplacent pas, ils tirent des flèches lorsque le joueur leur fait face. Les **chevaliers** suivent le joueur pour l'attaquer avec leurs épées. L'ennemi final peut tirer des boules de feu et mettre des coups d'épée. Ils suit également le joueur.

Le joueur de son côté a différentes capacités qui s'ajoutent à mesure que les niveaux augmentent. Au niveau 1 le joueur n'a aucune capacité particulière il peut seulement se déplacer et mettre des **coups d'épée**. Au niveau 2 le joueur peut utiliser la **téléportation** qui lui permet de se déplacer très rapidement en un coup. Attention la téléportation ne permet cependant pas de passer au travers des objets de l'environnement ou des ennemis. Au niveau 3 s'ajoute les **boules de feu** qui est l'attaque à distance du joueur. Au niveau 4 le joueur a les mêmes capacités qu'au niveau 3. Enfin le joueur peut trouver des **soins** dans les niveaux. Les soins sont représentés par les items poulet.

Nous avons pu mentionner précédemment que chaque niveau possède une musique différente afin de varier l'ambience. Nous avons voulu faire en sorte que plus le joueur avance dans le jeu, plus la musique devient pesante.

Pour terminer, le jeu se joue avec les touches `zqd` (ou directionnelles) pour les déplacements. Les touches utiles aux pouvoirs dépendent des préférences des joueurs. On utilise par défaut `Shift`, `space` et `s`. D'autres touches servent au débogage, nous le verrons dans une section dédiée. Les niveaux ont été conçus de manière à ce que le joueur se serve au moins une fois de chaque pouvoir mis à sa disposition.

2 Organisation du code

L'organisation générale du jeu est proche de celle du code modèle utilisé au début de ce projet. On retrouve dans `src/systems/` l'ensemble des systèmes du jeu. Le système **control** permet d'avoir accès aux entrées du joueur. Il est utilisé par l'entité player et les boutons du jeu. Ce système va appeler une fonction dans l'entité qui va ensuite gérer les actions en fonction des entrées. Le système **real_time** permet de gérer les actions qui doivent se passer sans latence pour le joueur. À chaque frame il appelle une fonction spécifique à chaque entité présente dans le système. Il permet par exemple de gérer les ennemis. Le système **music** s'occupe de la gestion des musiques du jeu. Il contient une variable `current_track` qui permet de sélectionner la track à jouer. Les track sont prédéfinies dans le code elles ne peuvent pas être modifiées depuis l'extérieur. Le système **view** permet de gérer la caméra. Nous reviendrons sur cette partie plus en profondeur dans la suite de ce rapport. Le répertoire `src/` contient trois nouveaux fichiers.

Le fichier `level_loader.ml` permet le chargement des niveaux depuis des fichiers texte. C'est une partie qui a été particulièrement intéressante à implémenter. L'objectif a été de pouvoir créer des niveaux sans avoir besoin de connaître le fonctionnement interne du jeu. Nous pensons être plotôt proche de cet objectif. Les fichiers textes doivent contenir un ensemble de lignes chacune permettant la création d'un élément du jeu. On utilise la notation `id:XxY|WxH|param` avec id l'identifiant de l'entité, XxY la position XY, WxH la largeur et hauteur de l'entité et param les paramètres associés à l'entité. Cette ligne est découpée pour récupérer chacune des parties. Les paramètres entrés sont ajoutés dans une table de hachage. Enfin lors de la création de l'entité on va récupérer un type

3 Organisation du travail

4 Les fonctionnalités

4.1 La caméra (Erwan)

5 Les tests effectués