Projet Logiciel Transversal

Paul VIELLET – Lucas EHLINGER

Table des matières

1 Objectif

1.1 Présentation générale

Figure 1 - Exemple d'un écran de jeu

Le but du projet est de réaliser un jeu basé sur Armello (figure 1), un jeu vidéo inspiré des jeux de plateau. Bien sûr il sera modifié (vue 2D sans déplacement de caméra, règles simplifiées).
Note : Toutes les règles et mécaniques qui seront énoncées dans la suite sont celles du jeu de base, elles sont donc susceptibles d'être changées ou abandonnées par soucis de simplification.
Légende :

1. Santé du personnage du joueur
2. Portrait du personnage en train de jouer
3. Nombre de points d'actions(1) restant au personnage jouant (joueur seulement)
4. Cartes en possession du joueur
5. Caractéristiques du joueur. De gauche à droite :
    - Combativité(2)
    - Vitalité(3)
    - Présence d'esprit(4)
    - Spiritualité(5)
6. Ressources du joueur. De gauche à droite :
    - Or(6)
    - Magie(7)
    - Prestige(8)
    - Déclin(9)

Glossaire

[1] Points d'actions : détermine le nombre de déplacement par tour.

[2] Combativité : détermine le nombre de dés disponible en combat.

[3] Vitalité : détermine la santé maximale du joueur.

[4] Présence d'esprit : détermine le maximum de cartes dans la main du joueur.

[5] Spiritualité : détermine le maximum de magie du joueur.

[6] Or : sert à jouer des cartes. Une colonie[10] sous le contrôle du joueur rapporte 1 or à l'aube.

[7] Magie : sert à jouer des cartes. La magie est restaurée au crépuscule.

[8] Prestige : à l'aube, le joueur avec le plus de prestige choisi une loi à appliquer. Un joueur perd 1 point de prestige à chaque mort.

[9] Déclin : maladie pouvant affecter les personnages. Le Roi commence à un Déclin de 1 et gagne 1 point de Déclin à chaque crépuscule. Un personnage touché par le Déclin perd 1 point de santé à chaque aube. Entre 1 et 4 points de Déclin, un joueur est considéré comme Infecté[11]. À partir de 5 points, et au-delà, un joueur est considéré comme Corrompu[12].

[10] Colonie : les cases spécifiques villages deviennent des colonies appartenant à un joueur quand ce joueur passe dessus.

[11] Infecté : un joueur infecté perd un 1 point de santé à chaque aube

[12] Corrompu : un joueur corrompu perd 1 point de santé à chaque aube. Une classe de case spécifique (cercle de pierres) tue les joueurs corrompus quand ils passent dessus. Effets supplémentaires en phase de combat.

1.2 Règles du jeu

Généralités

Dans Armello, les joueurs (au nombre de quatre -les places vacantes sont jouées par l'IA-) ont pour but de venir à bout d'un Roi touché par le Déclin (situé au centre du plateau). Le Déclin fait perdre 1 point de santé au Roi chaque jour (un jour est composé de deux tours de table, un diurne et un nocturne) ce qui limite donc le temps de jeu. En plus des joueurs, des Gardes Royaux contrôlés par l'IA sont en jeu, des Fléaux également contrôlés par l'IA peuvent apparaître, les spécificités de ces unités ne seront pas détaillées ici. Lors d'une interaction, un joueur peut perdre tous ses points de santé. Le cas échéant, il revient à sa case départ. Si un joueur perd toute sa santé durant son tour, celui-ci s'achève.

Plateau

Le jeu se déroule sur un plateau rectangulaire composé de cases hexagonales de taille 6x7. Les joueurs commencent aux coins du plateau.

Il existe 8 types de cases différentes :

- Plaine
- Forêt
- Montagne

- Village
- Donjon
- Marais
- Cercle de pierres
- Jardin du palais

Les cases ont des spécificités différentes. Il faut noter que les cases de départ des joueurs sont nécessairement et uniquement mitoyennes de 2 plaines. Il y a seulement 4 jardins du palais en jeu, positionné au centre du plateau. Pour accéder au Roi, un joueur doit se trouver sur une case jardin du palais.

Condition de victoire

Il existe quatre conditions de victoire :

- Tuer le Roi.
- Confronter le Roi en ayant un niveau de Déclin plus élevé que lui.
- Avoir le niveau de prestige le plus élevé quand le Roi meurt à cause du Déclin.
- Confronter le Roi après avoir réuni 4 pierres de d'esprit[1].

Cartes

Figure 1 - Exemple de cartes

Il existe 3 catégories de cartes dans Armello :

- Équipement
- Sort
- Ruse

Les cartes sont séparées en plusieurs types, le type est inscrit dans le coin supérieur droit de la carte, voir figure 2. Au début de chaque nouveau tour (aube ou crépuscule), le joueur pioche des cartes dans une des trois piles jusqu'à avoir rempli sa main.

Il existe 6 types de cartes :

- Bouclier
- Épées
- Lune
- Soleil
- Déclin
- Wyld[2]

Combat

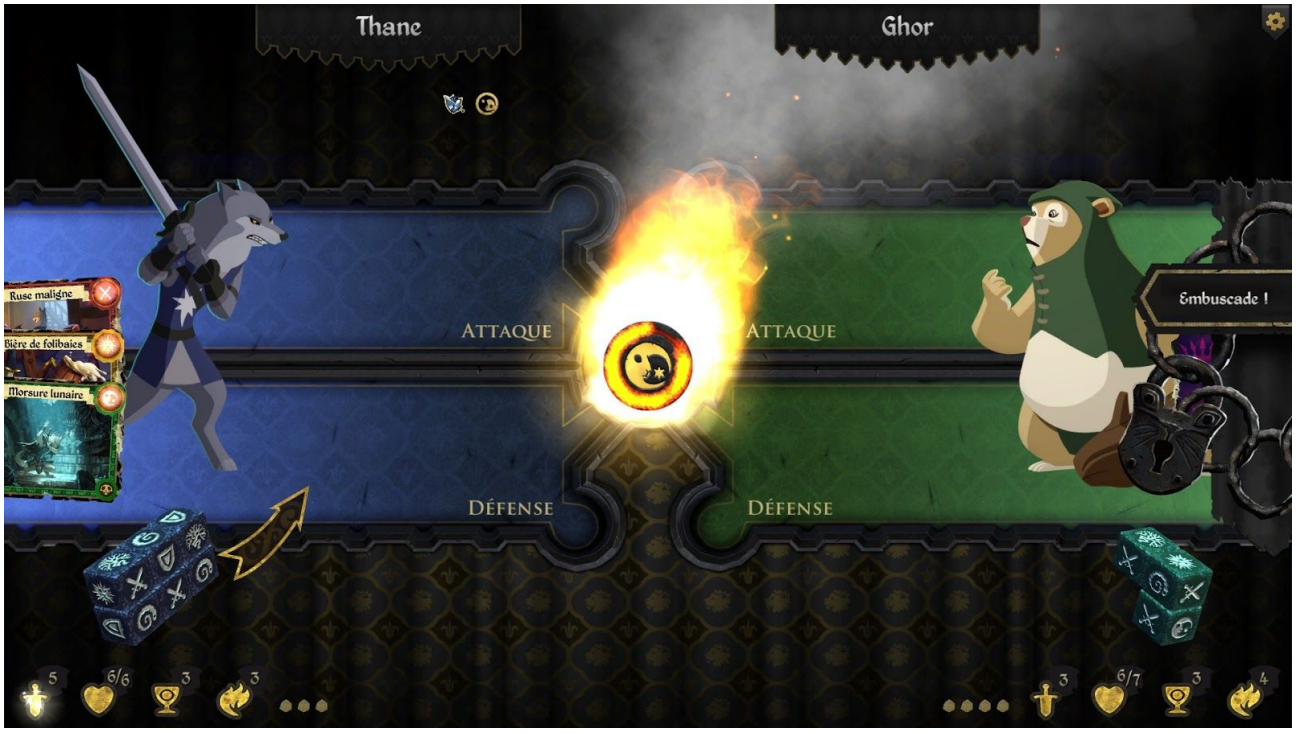

Figure 2 - Début d'un combat

Figure 3 - Résolution d'un combat

Lors du jeu, des combats peuvent se déclencher (joueur contre joueur, joueur contre IA, IA contre IA). Le déroulement est décidé aux dés, voir figure 2, le nombre de dés à lancer est déterminé par la caractéristique de combativité. Les participants peuvent brûler[3] leurs cartes au début du combat afin de fixer le résultat d'un nombre de dés équivalent, en fonction des cartes brûlées.

Dés

Figure 4 - Exemple de dé

Les dés possèdent 6 faces, une pour chaque type de carte, voir figure 4, le détail des effets en combat est précisé :

- Bouclier
  - Le participant gagne une défense[4]
- Épées
  - Le participant gagne une attaque[5]
- Lune
  - La nuit, le participant gagne une attaque
  - Le jour, le dé est défaussé
- Soleil
  - La nuit, le dé est défaussé
  - Le jour, le participant gagne une attaque
- Déclin
  - Participant Corrompu, le participant gagne une explosion[6]
  - Participant Non-Corrompu, le dé est défaussé
- Wyld
  - Participant Corrompu, le dé est défaussé
  - Participant Non-Corrompu, le participant gagne une explosion

Péril

Figure 5 - Exemple de péril

Un péril est une épreuve au cours de laquelle le joueur doit tirer des symboles spécifiques avec ses dés sous peine de subir une pénalité. Comme en combat, le joueur peut brûler ses cartes pour fixer le résultat d'un nombre de dés équivalent, voir figure 5.

Glossaire

[1] Pierre de d'esprit : apparaît aléatoirement sur des cercles de pierres, trouvable lors de quêtes ou lors de l'exploration de donjons.

[2] Wyld : élément d'histoire dans le jeu, inutil dans le projet, c'est l'arbre sur les dés et cartes.

[3] Brûler une carte : défausser une carte au début d'un péril ou combat afin de fixer le résultat d'un dé.

[4] Défense : empêche la perte de 1 point de santé lors d'un combat.

[5] Attaque : retire 1 point de santé à l'adversaire lors d'un combat.

[6] Explosion : le dé confère une attaque et un nouveau dé est lancé, disponible seulement en combat.

1.3 Conception Logiciel

Figure 1 - Cases utilisées

Figure 2 - Exemple de sprite de personnage

2 Description et conception des états

L'objectif de cette section est une description très fine des états dans le projet. Plusieurs niveaux de descriptions sont attendus. Le premier doit être général, afin que le lecteur puisse comprendre les éléments et principes en jeux. Le niveau suivant est celui de la conception logiciel. Pour ce faire, on présente à la fois un diagramme des classes, ainsi qu'un commentaire détaillé de ce diagramme. Indiquer l'utilisation de patron de conception sera très appriécé. Notez bien que les règles de changement d'état ne sont pas attendues dans cette section, même s'il n'est pas interdit d'illustrer de temps à autre des états par leur possibles changements.

2.1 Description des états

2.2 Conception logiciel

2.3 Conception logiciel : extension pour le rendu

2.4 Conception logiciel : extension pour le moteur de jeu

2.5 Ressources

*Illustration SEQ Illustration * ARABIC 1: Diagramme des classes d'état*

3 Rendu : Stratégie et Conception

Présentez ici la stratégie générale que vous comptez suivre pour rendre un état. Cela doit tenir compte des problématiques de synchronisation entre les changements d'états et la vitesse d'affichage à l'écran. Puis, lorsque vous serez rendu à la partie client/serveur, expliquez comment vous aller gérer les problèmes liés à la latence. Après cette description, présentez la conception logicielle. Pour celle-ci, il est fortement recommandé de former une première partie indépendante de toute librairie graphique, puis de présenter d'autres parties qui l'implémente pour une librairie particulière. Enfin, toutes les classes de la première partie doivent avoir pour unique dépendance les classes d'état de la section précédente.

3.1 Stratégie de rendu d'un état

3.2 Conception logiciel

3.3 Conception logiciel : extension pour les animations

3.4 Ressources

3.5 Exemple de rendu

*Illustration SEQ Illustration * ARABIC 2: Diagramme de classes pour le rendu*

4 Règles de changement d'états et moteur de jeu

Dans cette section, il faut présenter les événements qui peuvent faire passer d'un état à un autre. Il faut également décrire les aspects lié au temps, comme la chronologie des événements et les aspects de synchronisation. Une fois ceci présenté, on propose une conception logiciel pour pouvoir mettre en œuvre ces règles, autrement dit le moteur de jeu.

4.1 Horloge globale

4.2 Changements extérieurs

4.3 Changements autonomes

4.4 Conception logiciel

4.5 Conception logiciel : extension pour l'IA

4.6 Conception logiciel : extension pour la parallélisation

*Illustration SEQ Illustration * ARABIC 3: Diagrammes des classes pour le moteur de jeu*

5 Intelligence Artificielle

Cette section est dédiée aux stratégies et outils développés pour créer un joueur artificiel. Ce robot doit utiliser les mêmes commandes qu'un joueur humain, ie utiliser les mêmes actions/ordres que ceux produit par le clavier ou la souris. Le robot ne doit pas avoir accès à plus information qu'un joueur humain. Comme pour les autres sections, commencez par présenter la stratégie, puis la conception logicielle.

5.1 Stratégies

5.1.1 Intelligence minimale

5.1.2 Intelligence basée sur des heuristiques

5.1.3 Intelligence basée sur les arbres de recherche

5.2 Conception logiciel

5.3 Conception logiciel : extension pour l'IA composée

5.4 Conception logiciel : extension pour IA avancée

5.5 Conception logiciel : extension pour la parallélisation

6 Modularisation

Cette section se concentre sur la répartition des différents modules du jeu dans différents processus. Deux niveaux doivent être considérés. Le premier est la répartition des modules sur différents threads. Notons bien que ce qui est attendu est un parallélisation maximale des traitements: il faut bien démontrer que l'intersection des processus communs ou bloquant est minimale. Le deuxième niveau est la répartition des modules sur différentes machines, via une interface réseau. Dans tous les cas, motivez vos choix, et indiquez également les latences qui en résulte.

6.1 Organisation des modules

6.1.1 Répartition sur différents threads

6.1.2 Répartition sur différentes machines

6.2 Conception logiciel

6.3 Conception logiciel : extension réseau

6.4 Conception logiciel : client Android

*Illustration SEQ Illustration * ARABIC 4: Diagramme de classes pour la modularisation*