CONDUITE DE PROJET

RAPPORT DE PROJET : TACTICS ARENA

4 avril 2018

Marin Laventure, Aurélien Dad
L2 Informatique, 2017-2018
Université du Maine

Sommaire

Table des figures

1 Introduction

Notre Projet était de développer un jeu du style Tactics Arena en language C, sur une période comprenant 36h de TP. Un Tactics Arena est un jeu de stratégie où deux joueurs s'affrontent au tour par tour en manipulant un ensemble d'unités sur un plateau.
Chaque joueur a en début de partie plusieurs unités différentes à sa disposition pour former une équipe.
Chaque type d'unité dispose de caractéristiques différentes (déplacement, attaque, défense etc..).
Le but du jeu est d'utiliser à bon escient les spécificités de ses unités afin d'éliminer toutes les unités adverses.

2 Organisation du travail

2.1 Répartition des tâches

Liste tâches	Aurélien	Marin	Ossan
Définition règles jeu	40%	40%	20%
Répartition des tâches	40%	40%	20%
Génération et affichage du plateau de jeu	20%	10%	70%
Initialisation des obstacles	80%	10%	10%
Gestion sélection unités	50%	50%	0%
Creation structure jeu	10%	80%	10%
Système de déplacement (algo recherche chemin)	75%	25%	0%
Système d'attaque	30%	70%	0%
Redaction du rapport	40%	60%	0%
Diaporama	30%	70%	0%
Organisation du git + Readme	35%	65%	0%
Makefile	90%	10%	0%

2.2 Outils de travail

Outils de communication : Appel vocal(discord), Mail

Outils de collaboration : Google drive, Github, Share latex

2.3 Méthode de travail

Nous avons essayé d'appliquer les notions du module de communication du premier semestre au tout début du projet, c'est à dire le modèle Le modèle QQOQCCP.

Dans le cadre du projet il était question de fixer les objectifs (fonctionnalités) réalisables dans le temps accordé.

Nous avons fixé des priorités ; jeu fonctionnel dans un terminal, interface graphique minimale et des objectifs secondaires ; mise en réseau, amélioration de l'interface..

Ensuite nous avons essayé de jauger la difficulté de chaque partie du développement et de les répartir selon nos capacités et intérêts.

Dès la première séance, nous avons établi un environnement de travail avec différents outils de collaboration et communication.

Il s'est averé que l'évolution du projet a été perturbé par plusieurs facteurs comme le départ de Raphael à la 5 ème séance et la révision du concept de jeu initial. (*voir difficultés)

3 Conception

3.1 Règles du jeu

1. Constitution de l'équipe :

Le jeu débute par une sélection et un placement sur le plateau successif des unités de chaque joueur.
Les joueurs sélectionnent un crédit limite qui déterminera la constitution de leur équipe :
chaque classe d'unité coûte un certain crédit et la seule restriction pour le joueur est de ne pas former une équipe dépassant le crédit maximum défini au préalable.
La liste des classes de personnages, leur identifiant et leur coût en crédit est affiché à l'écran avant chaque composition d'équipe d'un joueur.(*voir tableau caractéristiques)

2.Déroulement d'un tour :

Le joueur 1 commence à jouer, alternativement le joueur 2.
L'ordre de jeu des unités est définit selon l'ordre de déploiement de celles-ci.

Ordre de jeu :

-Tour 1. Unité N°1 joueur1

-Tour 2. Unité N°1 joueur2

-Tour 3 Unité N°2 joueur1

-Tour 4 Unité N°2 joueur2

...

Pendant un tour, le joueur peut effectuer un déplacement et/ou une attaque contre une unité adverse allié ou adverse.

Une unité ayant attaqué pendant le tour ne peut plus être déplacé durant le même tour.

3. Fin de partie :

Lorsque un des deux joueurs a éliminé toutes les unités adverses, il remporte la partie et celle-ci prend fin.

Personnage : Id	Crédit	Points de vie	Attaque	Défense	Points déplacement*	Portée attaque*	Type*
Witcher : 0	20	10	4	40	3	3/3	3
Mage : 1	15	6	6	20	2	3/3	1
Scoia'tel: 2	10	6	3	20	3	3/2	1
Nains : 3	10	7	4	30	3	2/2	1
Spectre : 4	10	6	2	40	3	2/3	2
Wyvern : 5	20	10	3	40	3	3/3	2
Leshen : 6	15	8	4	40	2	2/2	2
Cyclope : 7	15	10	4	30	2	2/2	2
Doppler : 8	10	5	2	30	4	2/3	3

FIGURE 1: Tableau des caractéristiques des personnages

4. Particularités :

*Points déplacement : Les classes Wyvern et Spectre peuvent se déplacer au travers des obstacles et des autres unités.

*Portée attaque : La portée d'attaque est définie par une certaine portée verticale et horizontale, ici : Horizontale/Verticale.

*Type : Les classes peuvent être de type humanoide, monstre ou hybride, respectivement représentées ici par 1, 2 et 3 Les unités de type 1 infligent plus de dégats à celles de type 2 et réciproquement. Les unités de type 3 infligent plus de dégats à celles de type 1 et 2.

3.2 Cahier des charges

1. Mise en oeuvre du jeu
   (a) Matrice Plateau/Obstacles
      i. Génération de la matrice plateau
      ii. Génération d'obstacles
   (b) Préparation de la partie
      i. Déploiement des unités
      ii. Définition de l'ordre de jeu des unités
   (c) i. Système d'attaque/blocage
      ii. Système de déplacement
2. Sauvegarde/Chargement

   (pas de retour d'ossan)
3. Interface graphique

   (pas de retour d'ossan)

4 Implémentation

4.1 Génération et affichage du plateau de jeu

Le tactics arena se déroule sur un plateau subdivisé en cases. C'est pourquoi l'utilisation d'une matrice pour le représenter nous semblait le plus adapté. La taille du tableau et le contenu des cases étant variable tout au long de la partie, notre tableau à donc été créé de façon dynamique.
Ainsi lorsqu'une fonction nécessite de modifier ses cases, on lui envois un pointeur de pointeur.

Pour générer les obstacles de manière semi-aléatoire, nous avons utilisé la fonction *rand()*, en limitant ses intervalles pour obtenir des obstacles : de nombre proportionnel à la taille de la carte, uniquement présents dans la zone centrale du plateau, placés de manière aléatoire dans cette zone.

```
Combien de crédits souhaitez-vous ?(entre 10 et 100)
50
   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
   0   0   0   0   0  -1  -1  -1   0   0
   0   0   0   0   0   0  -1   0   0   0
   0   0   0   0  -1   0   0   0   0   0
   0   0  -1   0   0   0   0  -1   0   0
   0   0  -1   0   0   0  -1   0   0   0
   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0

 composition de l'équipe du joueur 1 : 50 credits total
```

FIGURE 2: Aperçu dans le terminal[1]

4.2 Préparation et sélection unite

Les unites sont représentées par plusieurs caractéristiques, c'est pourquoi nous avons choisis une structure (nommée *unite_s*) comprenant tous les champs nécessaires afin de pourvoir différencier les unités de même type, classe et joueur. Chaque classe d'unite est ensuite referencée dans un tableau *tab_ref*, qui ne sera plus modifié par la suite.

Pour la sélection, une fois le crédit sélectionné, le joueur 1 commence sa saisie d'unité en entrant également la position souhaitée pour chaque unité. L'unité, si la position est correcte, est enregistrée dans un tableau *tab_j1* comprenant toutes ses unités. Il en est de même pour le joueur2 avec *tab_j2*. Une fois les deux tableaux remplis, un troisième tableau est créé, comprenant les unités des deux joueurs de manière alterné (avec entassement à la fin, chaque joueur n'ayant pas obligatoirement le même nombre d'unités). Ainsi en parcourant ce dernier, on obtient directement notre ordre de jeu.

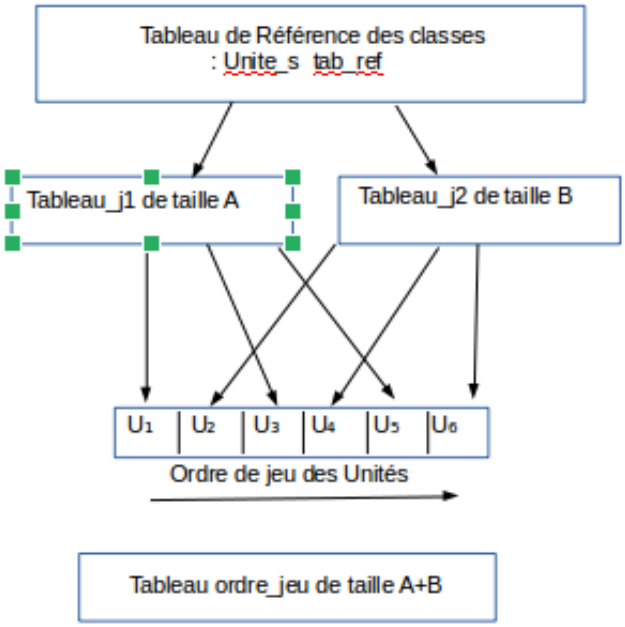

FIGURE 3: Gestion des tableaux au cours de la sélection des unités

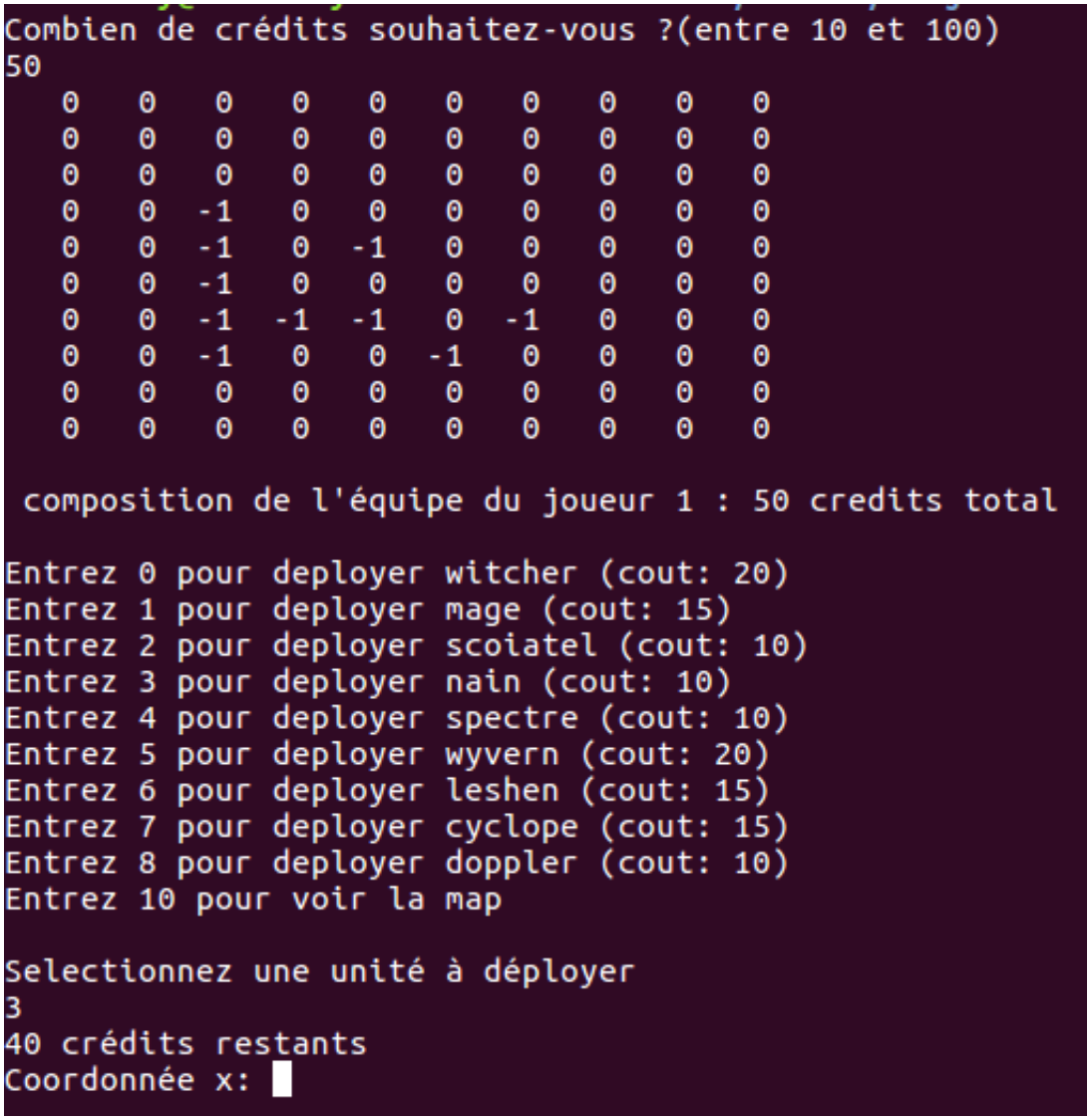

FIGURE 4: Aperçu dans le terminal$_2$

4.3 Système de déplacement

Tout d'abord le joueur doit rentrer des coordonnées *x,y* pour aller à la case choisie. Puis, nous vérifions dans un premier temps que cette case est bien disponible. Dans un second temps nous regardons si le nombres de points de déplacements admet un déplacement de cette longueur. Dans un dernier temps, si l'unité n'est pas de type volant, nous cherchons si un chemin est possible grâce à un algorithme de recherche de chemin (en tenant compte

des contraintes précédentes) n.

Une fois ces vérifications faites, nous pouvons gérer le déplacement en modifiant directement les champs *coord.x* et *coord.y* de l'unité.

4.4 Système d'attaque

Afin de commencer à gérer le système d'attaque, le joueur saisie l'*id* de l'unité qu'il aimerait cibler. Une fois cet *id* récupéré nous pouvons donc accéder aux coordonnés de l'unité ciblé et de ce fait savoir si elle est à portée d'attaque. Ensuite nous déterminons par un *rand* si l'attaque a été bloqué. Dans ce cas, aucune modification n'est apportée, et l'attaque prends fin. Dans le cas contraire, on calcule les dégâts infligés, en prenant compte du type de l'attaquant et de l'unité ciblée. Si une unité n'a plus de point de vie, elle meurt.

Dans une version précédente, le joueur choisissait directement l'id parmis une liste contenant les cibles à portée d'attaque. C'était plus optimisé, cependant beaucoup d'erreurs persistaient.

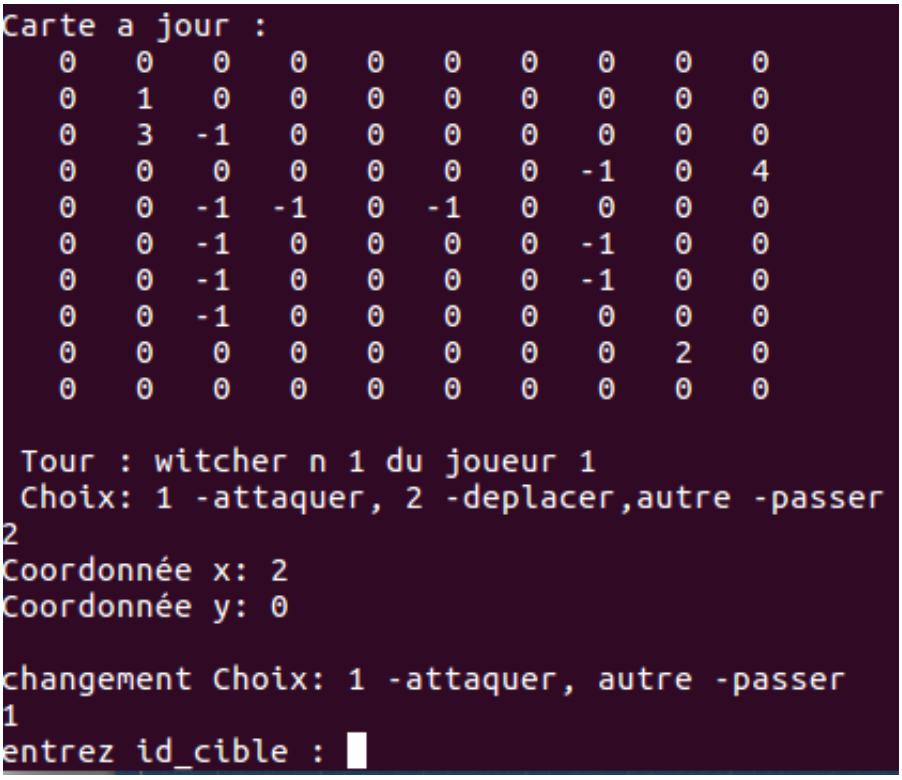

FIGURE 5: Aperçu dans le terminal[3]

4.5 Gestion du main

Pour le main, nous avons essayer de ne pas trop le charger afin qu'il reste clair. Il fait principalement appel aux fonctions créés à l'extérieur, à l'exception de la boucle principale qui gère les tours de jeu des unités.

5 Résultats

5.1 Bilan Objectifs/Résultat Final

L'objectif était le développement d'un Tactics Arena fonctionnel. Cependant certaines fonctionnalités considérées comme obligatoire comme la sauvergarde et le chargement n'ont pas pu être traités.
Aussi, l'interface graphique qui était notre première option à implementer n'a pas abouti. C'est notre principal regret car il permet de rendre le jeu bien plus attractif.
Nous n'avons pas eu le temps d'optimiser l'affichage de fin de partie.
Une IA aurait été intéressante à coder, nous pensons l'ajouter prochainement.

Malgré tout, notre jeu est opérationnel, et tout a fait compréhensible pour une personne ayant pris connaissance des règles présentes dans le readme. Nous allons poursuivre ce projet afin de le finir proprement, avec une SDL fonctionnelle et une IA intégrée.

6 Conclusion

6.1 Difficultés

Nous avons rencontré un certain nombres de difficultés, sur le plan technique mais davantage sur le plan organisationel. En effet la mise en route du projet a été plutôt laborieuse au niveau du développement : après avoir structuré notre approche du sujet initial, nous avons travaillé individuellement sur des parties différentes sans grande concertation. Cela a entraîné des désaccords sur la conception du jeu (de ses règles en particulier) et des incohérences dans le code lorsque nous avons fait le point sur l'avancement pendant la séance 4 et 5. Raphael a quitté le projet à partir de la 5ème séance ce qui a donné lieu à une révision impérative du projet et de ses objectifs.

Comme l'indiquait le pdf de présentation de notre projet, celui-ci était inspiré d'un jeu trop spécifique au niveau des règles et intéressant de par son moteur physique, son interface graphique (total accurate battle simulator). C'est pourquoi nous avons réorienté le projet vers un tactics arena dont les enjeux nous était plus clairs. Le code que nous avions déja produit n'a pas été totalement perdu car il était également adapté au tactics arena : génération de la matrice plateau, des obstacles, contraintes sur le placement des unités..

A partir de là, nous avons réassigné les tâches de chacun et l'avancement du projet a progressé significativement.(*voir répartition des tâches)

6.2 Apports du projet

Marin : Le projet m'a permis de comprendre l'importance d'une bonne organisation et estimation de la difficulté pour palier aux éventuels imprévus. J'ai découvert le markdown que je ne connaissais pas pour la rédaction du readme. Il m'a aussi permis de développer mon autonomie en terme de programmation, comme chercher des alternatives personnelles à certains problèmes.
J'ai aussi mieux compris certaines notions qui me paraissaient parfois abstraites lorsque j'en ai eu la nécessité dans des cas concrets (les pointeurs par exemple).

Aurélien : Tout au long de ce projet, j'ai amélioré mes compétences en C, dont les pointeurs qui me posaient des difficultés.
Pour ma part, ma surprise a été le temps pris par l'organisation, et la rédaction (rapport, commentaire du code, ...) que je sous-estimais.
Enfin, je ne pensais pas rencontrer autant d'imprévus tel que les abandons.