GUYON Arthur
RICHEFEU Mattéo
SANNA Florian

KAISER

Rapport de projet

GIT: https://github.com/iggdrael/ProjetS4

Sommaire

I. Introduction

Kaiser est un jeu de plateau se jouant au tour par tour sur une carte matricielle. L'adversaire de l'utilisateur est une "IA" gérée par le programme. Chaque joueur dispose d'une équipe constituée avant le début de la partie. Ses membres sont des héros appartenant à diverses classes ayant chacune des statistiques différentes.

Le joueur devra user de stratégie pour agencer son régiment sur la carte afin d'exploiter au mieux le potentiel de ses pièces. En maintenant des positions clés, il pourra prendre le dessus sur son adversaire et achever ses personnages.
Le but du jeu est de supprimer l'entièreté du bataillon ennemi. Le joueur ne pourra déplacer un héros qu'une seule fois par tour, l'action de celui-ci s'achèvera s'il attaque un opposant.

Afin d'améliorer l'expérience de gameplay, le nombre ainsi que la constitution de l'équipe adverse sont générés aléatoirement à chaque partie; Il en est de même pour le plateau de jeu.

Un éditeur de carte est également accessible au joueur. Il pourra ainsi créer d'autres plateaux de jeu et les sauvegarder afin d'y lancer de nouvelles parties.

Afin de rendre compte du travail réalisé, ce rapport est organisé en quatre thèmes majeurs.

Dans un premier temps, l'organisation du travail entre les différents membres du groupe y sera présentée. Cette partie décrit l'affinement du choix du thème, la réalisation du cahier des charges, les outils utilisés durant le processus de création ainsi que la répartition du travail.

La seconde partie présente ensuite les fonctionnalités du programme ainsi que les choix de conception réalisés par l'équipe.

La troisième section concerne l'implémentation du jeu. Les technologies et les algorithmes développés au cours du projet y seront présentés.

Enfin, une conclusion présentera les résultats du travail réalisé ainsi qu'une synthèse répondant aux objectifs initiaux.

II. Organisation du travail

1. Choix du thème

Le choix du thème du jeu a pris du temps. En effet, les membres du groupe étant très motivés par le projet, chacun voulait être certain de choisir le meilleur sujet possible.
Après la présentation des divers thèmes, nous avions déjà différentes idées de jeux qui nous plaisaient. La première idée était de réaliser un jeu de rôle dans la continuité du bot Discord développé par Arthur et Florian. Le groupe a décidé d'explorer un autre type de jeu à la place.

Après réflexion, trois thèmes ont retenus notre attention, un tactics arena, un roguelike, ou notre idée personnelle qui était de faire un genre de jeu de combat dans le style de Super Smash Bros. Cette dernière idée a vite été abandonnée car la gestion des animations et des différentes attaques des personnages se révélait trop compliquée à implémenter en SDL2.
Nous avons alors fait une liste des avantages et inconvénients pour chaque thème et nous sommes arrivés à la conclusion que Tactics Arena était le choix le plus judicieux.

Le groupe a tout de même décidé de réaliser un jeu original avec des règles lui étant propres. L'hypothèse d'Arthur de se baser sur le jeu Fire Emblem a motivé tout le monde et nous avons commencé la rédaction du cahier des charges.

2. Cahier des charges

La création du cahier des charges n'aura pris qu'une séance au groupe. Chacun avait déjà réfléchi de son côté à ce qu'il imaginait pour le jeu.

L'équipe a alors fixé les objectifs à court et long terme en décidant de ne pas voir trop grand afin d'avoir de la marge entre chaque étape.

Ce document *(cf. annexe 1)* est le fil conducteur de notre jeu. Il comprend les différentes parties à implémenter, les structures de données, les différentes statistiques des personnages etc... Lors de la conception de celui-ci, nous avons pensé à rajouter des fonctionnalitées optionnelles pour aller plus loin.

3. Outils utilisés

Pour nous aider à réaliser ce projet nous avons mis en place différents outils pour nous faciliter les tâches.
L'outil le plus utilisé a été un document google drive, tout le monde pouvait y accéder avec une connexion internet pour voir le cahier des charges et le planning.
De plus nous avons utilisé GitHub pour pouvoir partager les différents dossiers dont nous avions besoin.

Il nous fallait aussi un outil pour pouvoir communiquer en vocal, pouvoir partager son écran pour de l'entraide. Nous avons utilisé Discord qui est une application idéale pour répondre à ces besoins.
Ces outils nous ont permis de travailler convenablement en groupe tout en coordonnant nos actions respectives.

Le groupe a aussi décidé de mettre en place un planning afin de rythmer l'avancée du projet (*cf. annexe 2)*. Celui-ci comprend les dates des cours encadrés de Projet associées à de grandes lignes directrices pour chaque tâche. Cela permet de laisser aux membres de l'équipe le temps de s'organiser comme il le souhaite entre deux séances. Le planning a fini par être abandonné au profit de nouvelles solutions d'équipe présentées plus bas.

4. Répartition des tâches

La répartition des tâches se faisait donc à chaque début de séance en même temps qu'un point commun sur l'avancement du projet. Dès que chaque membre du groupe était conscient de ce qu'il devait réaliser, il s'occupait de son travail, tout en restant joignable par les autres membres du groupe pour aider ses collègues, que ce soit sur un bug, une idée floue sur la manière de réaliser son code, ou même une modification sur un fichier déjà utilisé par un autre.

En ce qui concerne la partie plus technique, tout le monde a dû apprendre à se servir de la SDL2. Ce qui a permis à ce que chacun travaille sur la gestion de l'interface ainsi que sur les algorithmes.

Arthur a beaucoup travaillé sur le jeu et ce qui concerne les déplacements des personnages, la gestion de collisions, l'aide aux déplacements, l'aspect visuel global du jeu, les combats, la gestion de l'"IA"... Il a aussi réalisé l'éditeur de cartes, le pack de textures et la seconde interface. Florian a réalisé la sélection de l'équipe par l'utilisateur et a implémenté certaines fonctions afin de favoriser l'usage de la SDL2.

Il a aussi développé le menu principal du jeu ainsi que la navigation entre les menus du jeu tout en permettant la responsivité du programme. Pour finir Mattéo a travaillé sur les paramètres (options) du jeu et a édité une bande son en studio pour l'ambiance générale du jeu. Il a aussi commencé à développer la partie réseau du jeu qui a été abandonnée plus tard. De plus, beaucoup d'implémentations du jeu ont été pensées et codées en groupe.

III. Conception

Lors du lancement du programme, l'utilisateur arrive dans le menu. Le groupe avait réalisé un background personnalisé nous avons choisi de mettre un plateau de jeu en fond afin de renforcer l'immersion.

Ce menu contient trois boutons redirigeant chacun à une fonctionnalité différente du programme.

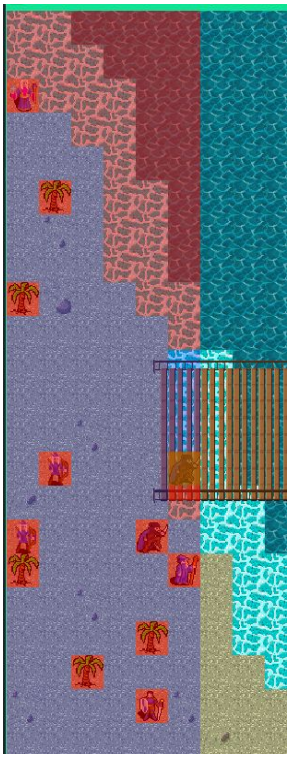
1. Partie Jeu

Le bouton Jouer lance une partie et va alors charger un plateau de jeu aléatoire et générer un chiffre entre 5 et 10 qui sera le nombre de héros de chaque joueur.
C'est lors de la création de l'équipe que le joueur va pouvoir mettre en place une stratégie pour battre l'adversaire.

Il va avoir le choix entre cinq types de personnages avec un gameplay et des statistiques différentes. Ces statistiques sont renseignées dans le cahier des charges bien qu'elles aient été quelque peu revues depuis afin d'équilibrer le jeu. Par exemple le guerrier aura plus de point de vie que l'archer mais ce dernier aura plus de portée et de capacité de déplacement, il pourra ainsi toucher ses ennemis de plus loin.

Le groupe a souhaité que le choix des personnages soit suffisamment vaste afin de rajouter de la diversité, il a été choisi de partir sur 5 classes différentes pour commencer (Guerrier, Mage, Archer, Prêtre et voleur) mais il est possible d'ajouter d'autres classes de personnages facilement. Les héros appartiennent aussi à une race (Humain et Elfe pour l'instant), cette fonctionnalité n'a pas d'impact sur le gameplay actuellement.

Le joueur va alors découvrir la carte, à cet instant il va devoir placer tous les membres de son équipe du côté de la carte qui lui est dédié. Pour le début de partie, le groupe a choisi d'implémenter une aide permettant au joueur de voir plus facilement les zones accessibles ou non en les faisant ressortir de différentes couleurs (rouge pour une case non accessible, bleu pour une accessible).



Le joueur peut désormais jouer et laisser place aux différentes stratégies que les personnages et la carte offrent.

Le jeu se déroule au tour par tour. Chaque personnage peut se déplacer sur une case dans la limite de sa capacité de déplacement. Le programme empêche le joueur de jouer plusieurs fois un personnage dans un tour et grise les héros ayant effectué une action.

Lorsque le joueur sélectionne un héros, ses statistiques sont affichées dans l'interface à droite de la fenêtre principale et une aide aux déplacements apparaît. Cette dernière affiche le radius des mouvements autorisés en bleu, les entitées alliées en vert, les entitées ennemies attaquables en violet et les cases interdites en rouge.

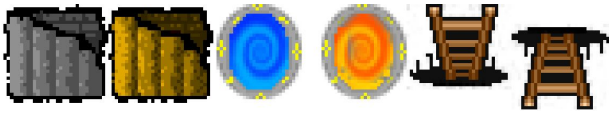
Le combat est une action qui termine le tour du personnage utilisé. Le personnage attaquant attaque en premier et si le héros attaqué est toujours en vie, il peut riposter si sa portée d'attaque est suffisante. Le calcul des dégâts correspond aux dégâts d'attaque de l'assaillant, pouvant être multipliés par 1,25 si un coup critique est porté.
Si l'adversaire esquive il ne reçoit aucun dégât sinon, sa défense réduit les dommages reçus. Deux autres caractéristiques existent, la portée ainsi que la marche, qui correspondent respectivement à la distance, en case, à laquelle l'attaquant peut toucher un adversaire, et la distance qu'un personnage peut parcourir pendant un tour.

Il faut confirmer chaque action avant qu'elle soit effectuée. Un tour se termine quand tous les personnages ont effectué une action ou lorsque le joueur décide d'y mettre fin.

Le joueur peut consulter les statistiques des personnages à n'importe quel moment de la partie en sélectionnant celui qui l'intéresse. Cela actualisera la fenêtre d'interface combat présente sur le côté de l'écran.

Les héros interagissent avec les éléments de décors de la carte. Effectivement, certaines cases sont inaccessibles, comme certains fonds (lave) ou éléments de décor (montagnes). Le joueur va alors pouvoir placer ces personnages derrière celle-ci pour se cacher des ennemis à faible portée et les frapper à distance

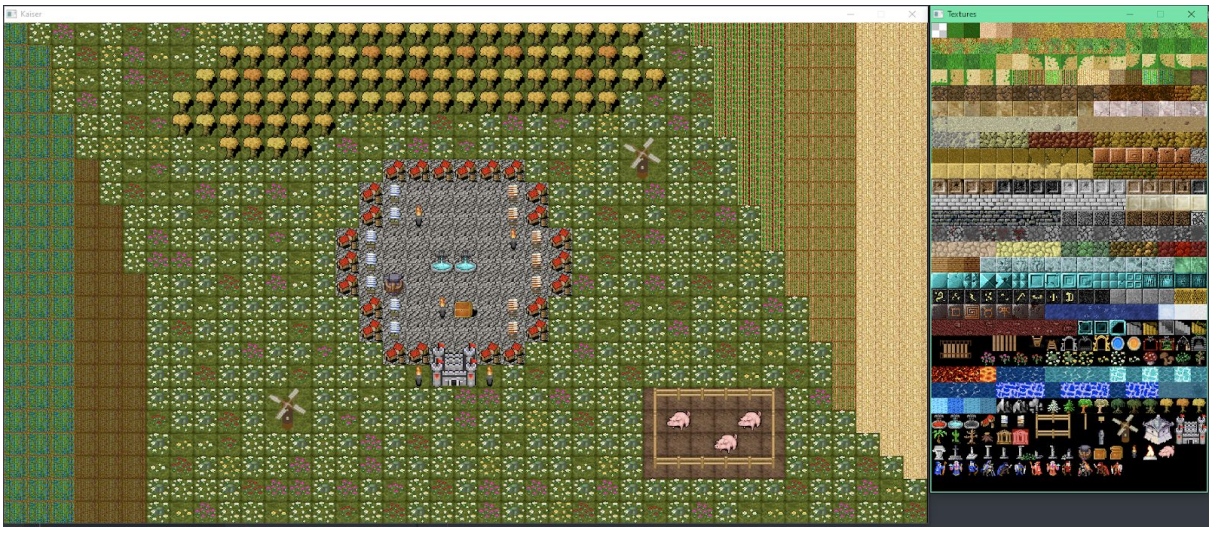
par exemple. Ces différents éléments apporte une variété dans le gameplay et dans la stratégie à adopter sur les différentes cartes existantes.
Il existe aussi des textures avec des interactions spéciales comme les portails.



Le jeu se termine une fois qu'un des joueurs n'a plus aucun personnage en vie et affiche un message de fin indiquant qui remporte la partie. Le joueur est ensuite redirigé sur l'écran d'accueil, il pourra ainsi décider de relancer une partie.

2. Éditeur de carte

Dans le menu, l'utilisateur a aussi accès à l'option l'option "éditeur".
Cette dernière lance l'éditeur matriciel qui va permettre au joueur de créer ses propres cartes.



L'éditeur de map est accessible à l'utilisateur car cela ajoute une dimension créative pour ceux qui le souhaitent. L'utilisateur a ensuite le choix de sauvegarder ses créations afin de pouvoir lancer ses parties sur celles-ci.

Une aide à la création est proposée en affichant une interface avec le pack de texture à droite de la fenêtre principale. La texture actuellement sélectionnée y sera encadrée en rouge pour plus de confort d'utilisation.

L'utilisateur a accès à différents raccourcis comme la possibilité de remplir la matrice avec la texture sélectionnée.

Dans une optique de performance, une case ne peut contenir que 3 textures maximum : un fond (herbe), un élément de décor (arbre) et un élément de type entité. Ce chiffre de 3 peut être modifié directement dans le code pour ajouter plus de textures sans problème.
En ce qui concerne le pack de texture, Arthur l'a totalement créé en récupérant des textures de divers sites ou en en créant certaines lui même. Il les a ensuite classé par catégorie (fond, élement, personnages). Il a utilisé ses compétences sur photoshop pour rendre les textures plus adaptées au jeu et au format matriciel.

3. Paramètres du jeu

Les paramètres du jeu sont la troisième option possible dans le menu. Lorsque le joueur lance le jeu, il peut à partir de là décider entre plusieurs résolutions (1280x720, 1366x768, 1600x900, 1920x1080), toutes étant en 16/9. Pour ce faire, l'utilisateur clique simplement sur le bouton correspondant à la résolution souhaitée, puis est renvoyé sur le menu principal. Tous les boutons, les textes ainsi que le fond ont été réalisés de manière responsive afin que le jeu ne perde pas de sa beauté dès que le joueur souhaite changer de résolution.

Une option son est aussi présente afin d'activer ou non la musique.

IV. Développement du jeu

L'entièreté du jeu est développée en langage C et la librairie graphique utilisée est la SDL2. Pour chacune des parties suivantes, une librairie spéciale lui est attribuée (jeu.h, editeur.h, menus.h), une librairie regroupant des fonctions standards du type itoa ou des primitives de piles (lib.h) ainsi qu'une librairie commune (commun.h) regroupant les structures et données utilisées dans toutes les parties.

Le programme utilise aussi SDL_image.h pour la gestion des extensions d'images et SDL_ttf pour la gestion des polices et des textes. L'équipe a fait le choix d'organiser le repository en plusieurs sous dossiers pour une meilleure visibilité et d'utiliser des branches git afin de travailler en coordination tout en gérant les bugs de commit.

Des tests unitaires ont aussi été placés un peu partout dans le code, ces tests nous renvoient des contenus de variable en console afin de pouvoir debugger.
Ces tests sont de la forme :

```
#ifdef DEBUG
printf("+Guerrier : %d, Total : %d\n", guerrier.h, nb_perso_actuel);
#endif
```

1. Partie Jeu

1. Création de l'équipe

La sélection de l'équipe du joueur se fait d'une part, par le nombre de personnages que le joueur doit utiliser, qui est tiré au hasard, par la suite, il a accès à une interface qui lui permettra de choisir entre cinq classes différentes (guerrier, mage, archer, prêtre, voleur) qui ont chacune des caractéristiques définies. Le joueur peut ajouter ou enlever des personnages à sa guise, tant qu'il respecte le nombre de personnages donné, chaque fois qu'il ajoute un personnage, une valeur correspondant au personnage sélectionné est incrémentée, de ce fait, lorsqu'il valide son équipe, les personnages seront déplacés dans un tableau et se verront attribuer leurs caractéristiques respectives.

Pour faciliter cette création d'équipe il y a deux structures qui ont été créées. Une structure personnage qui prend en compte les caractéristiques, la position du personnage sur la carte et une variable pour savoir si le personnage est un bot ou non.

Ensuite il y a une structure joueur qui prend les personnages (donc un type de la structure précédente) et le nombre de personnages présents dans l'équipe. Ce nombre est initialisé aléatoirement en début de partie.

2. Phase de jeu

Pour le jeu nous avons utilisé un algorithme de pathfinding, qui permet de rechercher un chemin, chemin qui sera le plus court possible.
L'algorithme développé est basé sur le même que celui vu en cours (lors du TP labyrinthe), mais il a été modifié pour qu'il soit adapté à nos besoins.
Concrètement le pathfinding est utilisé pour la gestion des déplacements de l'adversaire, pour les rendre plus intelligent, que l'adversaire ait une stratégie, celle d'attaquer l'ennemi le plus proche. Le pathfinding sert aussi pour l'aide aux déplacement mis en place pour indiquer les cases atteignables ou non. Mais aussi pour montrer à l'utilisateur si un personnage adverse est attaquable.

Cet algorithme assez spécifique est donc utile pour l'aide qu'il apporte à l'utilisateur mais aussi à la machine qui va jouer un peu plus intelligemment pour un gameplay plus "difficile".

Pour ce qui est du calcul des dégâts, on génère des nombres aléatoires que l'on compare aux capacité de critique et d'esquive des héros qui sont exprimées en pourcents.

2. Editeur de carte

Afin de gérer la création de cartes, l'équipe a choisi d'implémenter une structure case_t pouvant accueillir jusqu'à 3 textures différentes. Le jeu est basé sur une matrice de case_t.
Les textures ayant été classées par catégorie dans le pack (fond, élément, personnage, atteignables, interdites), lorsqu'une texture est appliquée sur la matrice, sa valeur correspondante dans sa case_t est automatiquement écrasée.

Lorsque le joueur sauvegarde une carte, cela crée un document texte contenant pour chaque case de la matrice, le nombre de textures qu'elle contient ainsi que l'indice de celles-ci dans le pack de textures.
Pour charger une carte on utilise le même procédé, le programme charge la matrice plateau avec les textures renseignées dans le document texte.

3. Paramètres du jeu

Pour ce qui est du choix de la résolution, le programme attend un événement (ici le clic); La fonction définie quatres zone de la taille des rectangles, dans un switch si le clic est détecté dans une des zones alors il change les variables concernant la taille de la fenêtre et utilise un fonction (SDL_SetWindowSize) qui est déjà existante dans SDL2 pour changer la résolution.

La taille de la fenêtre initiale est définie dans le main, car elle fait partie des variables principales pour rendre les fenêtres responsives, la taille de la seconde fenêtre (interface statistiques et pack de textures) est calculée en fonction de la résolution de la première.

V. Résultats

Les fonctionnalités les plus importantes sont réalisées, telle que la gestion des combats ou encore les déplacements, que ce soit ceux réalisés par le joueur ou l'ordinateur, mais également l'éditeur de carte et une partie des paramètres. La partie correctement terminée des paramètres est celle permettant à l'utilisateur de changer la résolution de la fenêtre de jeu, n'ayant pas implémenté de son dans le jeu, aucune fonction permettant de le rendre muet n'a été ajoutée malgré une bande-son faite par Mattéo. Il manque également toute la partie réseau étant problématique pour une question de portabilité, et ayant des systèmes d'exploitations différents dans l'équipe. Cette partie réseau n'étant pas essentielle au jeu et a donc été abandonnée.

Le planning n'a pas été respecté car nous n'avons pas pris en compte les différents problèmes et difficultés rencontrées lors du développement. C'est une erreur de notre part de ne pas avoir laissé plus de temps pour s'entraider face aux problèmes. Le planning n'était peut être pas assez détaillé, il prenait les grandes lignes du projet. Il aurait peut être fallu utiliser un outil comme Trello pour simplifier l'organisation de notre travail.
La solution trouvée a donc été de faire des appels réguliers pour se tenir au courant de l'avancée de chacun et de s'entraider lors des problèmes.

Si le jeu devait être amélioré, il le serait au niveau du réseau, étant une partie que nous avons abandonné, qui était dans notre planning initial et qui nous tenait à coeur. Le jeu pourrait également avoir une amélioration sur l'IA, qui pourrait faire des mouvements plus réfléchis, comme prendre en compte les classes du joueur et se placer derrière des cases pour casser la ligne de vue des personnages distants, ou encore placer les personnages ayant de plus de vie à l'avant du groupe. Le jeu serait aussi complété par la musique qui a spécialement été créée pour lui.

Ce projet nous a tout d'abord apporté des compétences techniques nouvelles, que ce soit la SDL ou les outils utilisés comme github et doxygen. Le fait de se débrouiller pour chercher la documentation puis ensuite l'adapter au contexte,nous a permis de prendre du niveau dans le développement d'un projet.
Quant à l'organisation, nous avons pu avoir un premier aperçu sur le travail d'équipe qui doit être amélioré. Pour une première expérience nous nous somme rendus compte des différentes difficultés qui peuvent se dresser dans un groupe, comme le travail en coordination car nous ne pouvions pas éditer les fichiers en même temps, donc deux personnes ne pouvaient pas être sur le même problème par exemple.

Nous avons eu grand plaisir lors du développement de ce projet, même face aux difficultés, nous avons trouvé des solutions pour le mener à bien, nous somme fiers du résultat final.

ANNEXES

(TACTICS ARENA) : Kaiser

GUYON Arthur
RICHEFEU Mattéo
SANNA Florian

Cahier des charges

Kaiser sera un jeu de plateau tour par tour où le joueur constitue son équipe et doit anéantir les troupes adverses. Le jeu devra être jouable en réseau à deux joueurs.

Le programme devra proposer un menu constitué de 3 onglets, lançant chacun une fonction.
-Un onglet "Jouer" qui lance la partie
-Un onglet "Editeur" qui lance l'éditeur de map
-Un onglet "Paramètres" qui permet à l'utilisateur de renseigner ses préférences

FONCTIONNALITÉ JOUER

La fonction jouer ouvrira d'abord un menu demandant au joueur de créer son équipe. On aura préalablement tiré aléatoirement le nombre de personnages que peut avoir un joueur.
Les personnages auront des classes avec des statistiques différentes :

- PV
- Attaque
- Défense
- Esquive
- Critique (*1,25 de l'attaque)
- Portée
- Déplacement

Classes avec leurs stats de base : (évoluera en fonction des tests)
(PV, ATK, DEF, ESQ, CRIT, POR, DEP)

- Guerrier (100, 40, 10, 0, 5, 1, 3)
- Mage (65, 60, 3, 5, 8, 2, 4)
- Archer (60, 50, 4, 5, 6, 3, 4)
- Healer (80, 15, 0, 2, 5, 2, 4)
- Voleur (45, 65, 3, 20, 33, 1, 5)
- [OPTIONNEL] Cavalier

Races (stats variables suivant la race) : [OPTIONNEL]

- Humain
- Gobelin
- Elfe
- Nain

Une fois les équipes constituées, on tire aléatoirement une map et on place les personnages à notre guise dans notre zone.

FONCTIONNALITÉ GESTION DES DÉPLACEMENTS

Il faudra tester si le personnage a accès à une case en fonction de son type et de sa capacité de déplacement. On cliquera sur le personnage pour le déplacer, nous devrons tester s'il peut se déplacer sur la case sélectionnée.

On devra implémenter une aide au déplacement indiquant au joueur si il peut aller sur certaines cases en fonction de leur type. Les cases possibles seront highlight en bleu opacité 50% et les inatteignables en rouge.

Le jeu aura une interface composée de la map et du jeu en lui même à gauche et des informations du joueur sélectionné et d'une partie annonçant une prévision de l'issue d'un combat entre deux personnages à droite.

Pour attaquer un personnage adverse, le joueur devra cliquer sur son personnage puis sur celui de l'adversaire, on affichera alors une prévision du combat sans les taux d'esquive et de critique pour enlever la partie aléatoire si la capacité de déplacement du personnage est suffisante. Le joueur devra cliquer sur un bouton confirmer pour valider un déplacement ou un combat.

Un déplacement ou une attaque conclut le tour d'un personnage, il devient alors inutilisable jusqu'au tour suivant.

FONCTIONNALITÉ ÉDITEUR DE MAPS

L'éditeur de map devra permettre à l'utilisateur de cliquer sur une texture d'un pack et d'éditer la map avec.

- Créer un pack de texture
- Gérer la transparence
- Classer les textures par type (fond, élément, personnage)
- Permettre à l'utilisateur de remplir la map avec la texture en appuyant sur une touche.
- Permettre à l'utilisateur de sauvegarder et de charger des maps depuis un fichier texte
- Une case de la map ne doit contenir qu'une texture de chaque type.

FONCTIONNALITÉ PARAMÈTRES

L'utilisateur pourra modifier la taille de sa fenêtre en fonction de plusieurs résolutions proposées en 16/9, il pourra aussi régler le volume de la musique..

FONCTIONNALITÉ RÉSEAU

Le jeu devra être jouable en local en communiquant avec une autre machine.
[Optionnel] Intégrer la possibilité de jouer en wifi.

(annexe 1)Document comprenant le fil conducteur du jeu

PLANNING

05/02: Cahier des charges, statistiques classes, début menu jeu, début paramètres

13/02: Finalisation des menus, changement de résolution, makefile

14/02: Éditeur de map, pack de textures

06/03: Éditeur de map, changement de la matrice en matrice de cases

12/03: Début de la création du jeu, création des structures cases, personnages

- Début création jeu, tirages aléatoires, partie "initialisation" des équipes, map etc...
- Finition des paramètres.
- Version graphique de la sélection des équipes des joueurs.

19/03: Gestion des déplacements en fonction du type de cases et de la capacité de déplacement du personnage

- Gestion des déplacements des personnages et aide au déplacement.
- Création d'une interface permettant de choisir un serveur et le numéro de joueur.
- Finition de la version graphique de la sélection des équipes des joueurs.

25/03: Gestion des combats et prévision des combats, gestion taux de critique/esquive

- Tests communication réseau/SDL.
- Finition interface du début de partie(choix serveur).
- Fonction placement des personnages dans les zones (début de partie).

01/04: BETA TEST, validation

-
-
- Encore sur la fonction placement des personnages dans les zones (début de partie).

08/04: Gestion affichage aide au déplacement, affichage des interfaces

24/04: Mise en réseau

29/04: Mise en réseau, TESTS FINAUX, RELEASE

(annexe 2)Planning utilisé par le groupe, chaque couleur correspond à une personne