UNIVERSITÉ PARIS SUD | IUT D'ORSAY
université PARIS-SACLAY

Réalisation technique

DEVELOPPEMENT DU JEU « LA BAGARRE »

Quentin Le Guellanff | Anthime Louvet | Emmanuelle Zenou | Miguel de Oliveira
PROFESSEUR REFERENT : MME KELLER | IUT ORSAY – DUT INFORMATIQUE AS

Table des matières

Introduction

Contexte global

Dans le cadre du projet tutoré S1, il nous a été demandé de reproduire un jeu vidéo existant en 3 mois. Le jeu vidéo à réaliser doit être simple, c'est-à-dire reposer sur un gameplay simple, être en deux dimensions etc… Le projet doit être réalisé par des groupes de 4 ou 5, il nous est imposé d'utiliser le langage C++ et d'utiliser la bibliothèque SFML pour programmer le jeu. Nous devons également utiliser les notions de gestion de projet acquises dans le cours correspondant, l'utilisation de Git est aussi imposée.

C'est dans ce cadre que le groupe G4 s'est constitué, il est composé de Miguel de Oliveira, Anthime Louvet, Emmanuelle Zenou et Quentin Le Guellanff. Après plusieurs échanges nous avons décidé de réaliser un jeu de combat 2D de type versus fighting à l'image de Street Fighter. Nous cherchons avant tout à créer un jeu stimulant, récréatif et très interactif : le jeu de combat s'est révélé être le jeu le plus à même de répondre à ce besoin dans l'univers 2D. Si la plupart des jeux d'arcades ne relèvent aujourd'hui plus que du passe-temps, le jeu de combat versus fighting est le seul à n'avoir cessé de se renouveler jusqu'à aujourd'hui (Mortal Kombat 10, DragonBall Fighter Z, Injustice) sans jamais changer la recette. Ce jeu de combat permet de plaire aux nostalgiques comme au nouveaux joueurs de cette génération : adrénaline et divertissement garantis. Nous soulignons aussi l'aspect multijoueur du jeu qui est aussi une grande valeur ajoutée à nos yeux. Ce jeu offrira sans nul doute un lien, une connexion entre deux utilisateurs au détour d'une partie endiablée. Pour toutes ces raisons, il nous est cher de produire un jeu de combat versus fighting qui saura être dans l'ère du temps.

Le chef de groupe désigné est Emmanuelle Zenou, elle sera en charge de rédiger les comptes-rendus hebdomadaires statuant sur l'avancement du projet. Chaque jeudi après-midi, le groupe organise une réunion pour avoir un visuel sur l'avancement global, et permettre de distribuer des nouvelles directives ou les affiner.

Le projet se découpe en différentes étapes :

1. Prise en main des outils
2. Rédaction du cahier des charges
3. Prototypage du jeu
4. Réalisation

Objectif du jeu

L'objectif principal du projet est de développer un jeu 2D se basant sur le jeu Street Fighter, le jeu se jouera en multijoueur local (2 joueurs) sur ordinateur, via l'utilisation de manettes ou du clavier. Il permettra de faire s'opposer deux joueurs via l'utilisation de personnages sélectionnables. On veut créer un jeu fun, réactif et accessible (jouable facilement). Le jeu a donc pour objectif final que les joueurs s'amusent, souhaitent rejouer et ainsi imposer notre jeu dans les après-midis et soirées entre amis. Le jeu a pour nom : « **La Bagarre** ».

Cahier des Charges

Ce cahier des charges a pour but de définir et présenter un projet de développement de jeu vidéo. Ce document permettra de lister les différents acteurs du projet, de fixer les impératifs du développement et son organisation.

A. Présentation du Projet

Nom du projet : La Bagarre

Genre : jeu de Combat, 1 contre 1

Plateforme : PC Windows, Linux

Jouabilité : Manette ou Clavier

Acteurs du projet : Les développeurs : Anthime Louvet, Miguel de Oliveira, Quentin Le Guellanff, Emmanuelle Zenou.

La tutrice : Chantal Keller

Le client : Les utilisateurs du jeu. Notre cible est un public jeune (12-25 ans), plus consommateur de jeux-video en moyenne, mais aussi un public adulte sachant que le premier street fighter est sorti en 1987 et que la licence rassemble encore de nombreux fans aujourd'hui trentenaires.

Chef de groupe/projet : Emmanuelle Zenou

Professeur référent : Mme Keller

Durée du projet : 8 octobre 2019 – 16 janvier 2020 (100 jours)

Ressources :

- Bibliothèque SFML
- Langage C++
- CodeBlocks

- Git
- Moodle (compte-rendu et échange avec le professeur référent)
- Groupe Messenger (groupe de messagerie permettant d'avoir des échanges rapides sur le projet)
- Fichiers png de textures rassemblant les sprites de personnages récupérés sur https://www.spriters-resource.com/
- Fichiers audios récupérés sur internet ou enregistrés par nos soins.
- GIMP (édition des sprites)
- PhotoFiltre 7
- Sublime Text
- PhotoShop
- Suite Office
- Extension DoxyBlocks

Exigences non-fonctionnelles :

- Volumétrie/sécurité : 1 Go maximum (suffisant pour stocker le code source, les ressources graphiques et audios), la sécurité ne sera pas prise en compte.
- Performance : jeu fluide, qui tourne sur la plupart des PC de moins de 10 ans en 60 fps.
- Ergonomie : Gameplay accessible, jeu facile d'utilisation.

Principaux Jalons :

- **9 novembre 2019 :** Rendu du cahier des charges et d'une petite application SFML réagissant à l'appui sur des touches du clavier
- **7 décembre 2019 :** Rendu d'un prototype du jeu et correction du rapport précèdent
- **16 janvier 2020 :** Correction du rapport précèdent, rendu des livrables avec fiches individuelles détaillant l'activité de chacun au sein du groupe
- **Mardi 21 janvier 2020 :** Soutenance du projet

B. Objectifs et fonctionnalités du projet

*Tous les éléments comportant une * ne sont pas obligatoires pour délivrer un jeu fonctionnel et seront donc produits en fonction du temps restant.*

On veut permettre à deux joueurs de pouvoir s'affronter en 1 contre 1 via l'utilisation de personnages sélectionnables. Les deux joueurs peuvent incarner un personnage qui peut être le même ou différent on aura donc au minimum deux personnages pour notre jeu. Les personnages s'opposeront dans une arène de combat sélectionnable. Chaque partie représente une phase de sélection de personnage, d'arène puis de combat.

On découpe donc une partie en deux phases :

- Phase de sélection
- Phase de combat

B1. Phase de sélection

Lors de la phase de sélection on distingue deux affichages différents, la sélection des personnages et la sélection de l'arène de combat.

Lors de la sélection des personnages, chaque joueur pourra sélectionner son personnage et valider son choix via l'utilisation des touches du clavier ou de leur manette. Sur cet écran sera afficher les différents personnages disponibles avec leur nom et une image correspondante. Une fois leur choix validé, apparait l'écran de sélection de l'arène.

Lors de la sélection de l'arène, le joueur 1 pourra choisir une arène dans laquelle les personnages combattront, ici aussi, via l'utilisation des touches du clavier ou de sa manette. Il devra valider son choix pour pouvoir enclencher la phase de combat.

B2. Phase de combat

Possibilité du Joueur

Les joueurs pourront contrôler les différentes actions de leur personnage à l'aide d'appui sur les touches du clavier ou via l'utilisation d'une manette.

Chaque joueur aura un nombre de points de vie. Il démarrera le combat avec un nombre de points de vie totale, ce nombre ne pourra que diminuer jusqu'à atteindre 0. Lorsque le nombre de points de vie d'un joueur atteint 0, il perd le combat et son personnage tombe au sol.

Pour diminuer le nombre de point de vie du joueur adverse, chaque joueur peut infliger des dégâts à l'autre avec les différentes actions de combat de son personnage. Chaque coup porté à l'adversaire entraîne une diminution de point de vie. La diminution dépend du coup que le personnage a porté à l'adversaire.

Lorsqu'un joueur gagne le combat, il gagne la partie et une nouvelle partie peut redémarrer.

*Le joueur peut mettre la partie en pause lors d'un combat via l'utilisation d'une touche.

Un joueur peut donc :

- Contrôler un personnage
- Perdre des points de vie
- Diminuer les points de vie de l'adversaire
- Gagner ou perdre une partie

Possibilité du personnage

Pour combattre, les joueurs s'opposeront donc dans une arène qui sera le décor du combat, et contrôleront leur personnage. Au début du combat les deux personnages seront face à face chacun d'un côté de l'arène.

Les deux personnages reposeront sur un sol statique et ne pourrons pas passer en dessous. Chaque personnage pourra se déplacer au-dessus du sol, pour cela ils auront différentes actions disponibles :

- Avancer
- Reculer
- Faire un dash vers l'avant*
- Faire un dash vers l'arrière*
- Sauter verticalement
- Sauter vers l'avant
- Sauter vers l'arrière
- S'accroupir
- Rester debout

Chaque personnage pourra également donner ou recevoir des attaques, pour cela ils auront différentes actions disponibles :

- Coup de poing lent*
- Coup de poing rapide

- Coup de poing vers le haut*
- Coup de poing accroupi*
- Coup de pied accroupi*
- Coup de pied rapide
- Coup de pied lent*
- Recevoir un coup
- Tomber au sol

Chaque personnage aura un visuel différent (sprite), ce visuel aura une animation pour chaque action que le personnage pourra réaliser.

Les personnages devront se faire face en permanence, ils pourront donc s'éloigner en reculant ou se rapprocher en avançant. Les deux personnages ne peuvent se traverser, ils ne peuvent également pas sortir de l'arène (de la fenêtre d'affichage). Il y a donc une gestion des collisions.

Interface de Combat

La durée maximale d'un combat sera d'une minute, si aucun des joueurs n'arrivent à se départager au bout d'une minute, le joueur avec le plus de point de vie l'emporte. L'affichage des points de vie des joueurs se fera via deux barres de vie en haut de l'écran. Le remplissage des barres de vie dépendra du nombre de points de vie des joueurs. Le compteur de temps sera affiché en haut de l'écran également.

C. Liste des Livrables

- Code source comprenant tous les fichiers .cpp et .h
- Documentation doxygen
- Ressources graphiques utilisées : il s'agit de la totalité des sprites, et textures que nous utilisons afin d'animer les personnages, et faire les images de fond du jeu. Nous utilisons aussi des polices de texte pour afficher le texte.
- Ressources audios utilisées : La musique et les sons des personnages que nous utilisons
- Manuel d'utilisation
- Fichier Exécutable sous Windows et Linux

UNIVERSITÉ PARIS SUD | IUT D'ORSAY

université PARIS-SACLAY

Réalisation technique

DEVELOPPEMENT DU JEU « LA BAGARRE » »

LISTE DE TOUTES LES TÂCHES A REALISER

AVANCEMENT GLOBAL

Notion à maitriser	Commentaires/ Méthodes	durée planifiée en jour	OK/Pas OK ou % de réalisation

0. Formation SFML commune à tous

1. Gestion des déplacements pour un joueur

Creation d'un personnage	découpage des sprites et animation pour chaque personnage créé: Broly, Jotaro, Dhalsim	30	OK
Gestion des commandes	Pouvoir jouer de manière optimale avec des manettes mais aussi avec un clavier ou les deux à la fois	7	OK
Déplacements: avancer, reculer, saut	Si le joueur fait sauter le personnage à gauche, le personnage saute à gauche et tout le long du saut il ne peut pas aller à droite	40	OK
Ne pas sortir de la fenêtre	Si saut contre un côté de la fenêtre le personnage doit retomber verticalement contre la fenêtre.	10	OK
Inversion du sprite et des commandes par rapport à un axe vertical	les deux personnages doivent toujours se faire face. Si un personnage dépasse son adversaire, les deux personnages doivent se retourner.	20	OK

2. Gestion des attaques et parades pour un joueur

Attaques au sol	Possible que au sol Coup de pied, coup de poing, en l'air	20	OK
Parade	Doit pouvoir être brisée. Est appelée lorsque l'adversaire attaque, elle se déclenche quand le personnage attaque utilise la touche de recul	10	OK
Attaques spéciales	Lorsque la barre d'énergie est pleine, on peut utiliser l'attaque spéciale. Quelques une pour créer la mécanique globale des attaques et donc la possibilité d'en créer d'autres par la suite	10	OK

3. Gestion des collisions entre deux personnages

Les deux personnages ne peuvent pas se superposer	Le dépassement ne peut se faire que par le saut	10	OK
Les attaques, si elles touchent un personnage infligent des dégâts	Les personnages s'ils se touchent entre eux n'infligent aucun dégâts	10	OK
Une barre de vie pour chaque personnage	Elle diminue à chaque attaque	15	OK
Le personnage, s'il est touché par une attaque, est animé en conséquence	+ effet de coup	10	OK

4. Menu

3 choix selectionnables par J1	Jouer, Commandes, Quitter	10	OK
Jouer	Emmène sur un second menu	3	OK
Commandes	Affiche les différentes commandes	10	OK
Quitter	Ferme la fenêtre	1	OK
Musique	Arrêt/Relance de la musique Plusieurs musiques différentes en fonction du mode du jeu et du décor choisi (menu, combat)	7	OK

5. Menu personnages

Trois personnages possibles	Ryu, Greg, Dhalsim Sélectionnables par J1 et J2 Quand les 2 ont choisit :Emmène sur un troisième menu. Les deux joueurs peuvent choisir un même personnage. Le choix est retenu et le joueur combat avec le personnage choisi	50	OK
Le joueur doit pouvoir revenir sur son choix si l'autre joueur n'a pas encore choisi grâce à une commande		5	OK

6. Menu Map

Affichage de toutes les maps (6)		10	OK
Choix de la map par J1	Le choix est retenu et le combat se déroule dans la map choisie et avec la musique correspondante	10	OK

7. Menu Pause

Appui d'une touche met le jeu sur pause	Le jeu s'arrête et le menu pause apparaît. Le menu pause peut être fermé et le jeu peut reprendre où il en était	5	Pas OK
Revenir au menu	Confirmation demandée type "Êtes-vous sur ? (o/n)"	2	Pas OK
Musique	Arrêt/Relance de la musique	2	Pas OK

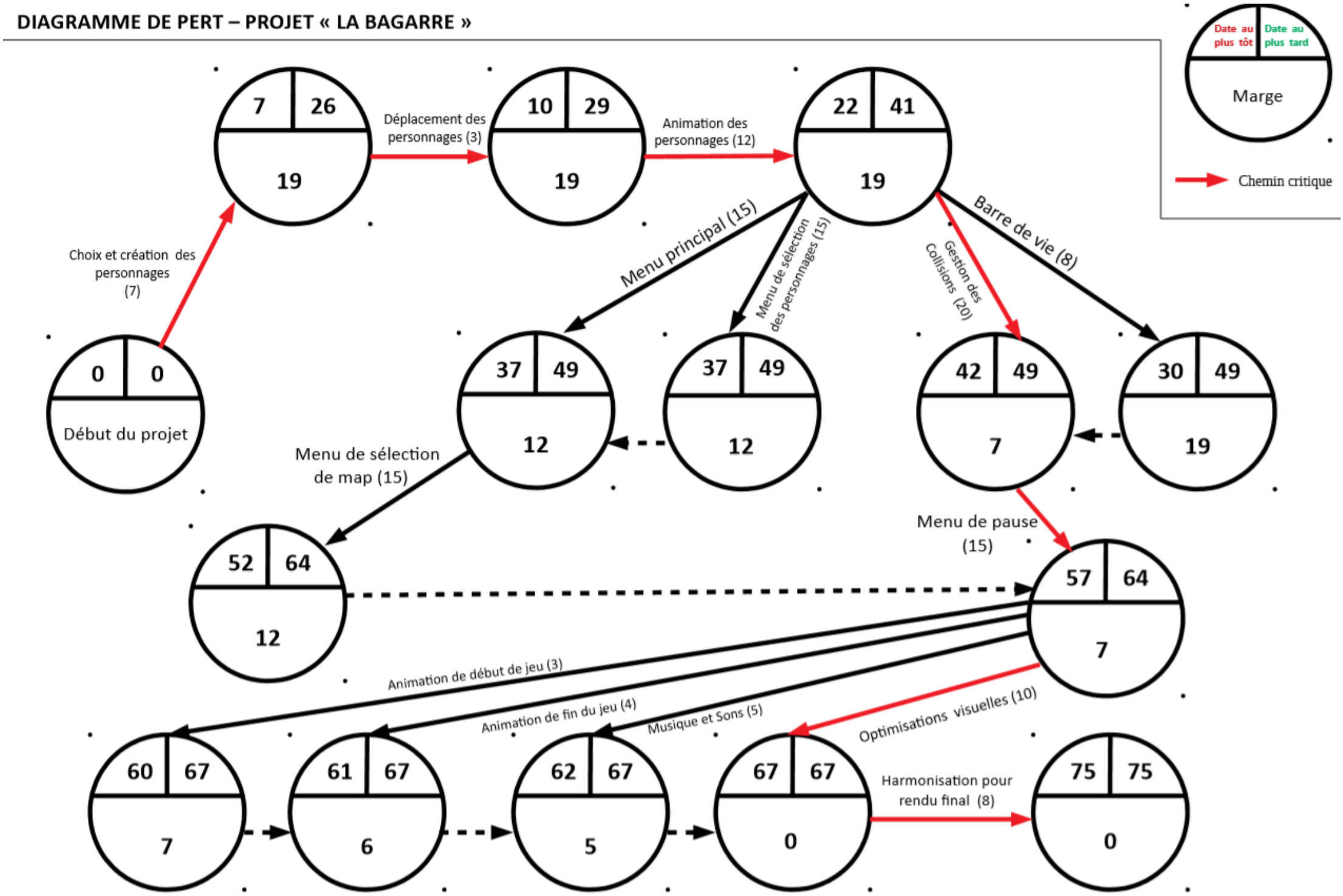
8. Animations

Animations début de jeu	Les personnages ont une animation d'"entrée en scène" « Ready » « Fight ! »	3	OK
Animations fin de jeu	« K.O » Lancée quand barre de vie =0 Revient automatiquement au menu	3	OK
Sprites décor/background	permettre la sélection de l'image de fond	10	OK
Musique de jeu	Boucle Est modifiée en fonction du menu et des différentes maps	7	OK
Son évènement	Sons de sélection dans les menus. Son d'entrée en scène des personnages, de coup, de dégâts, et de victoire.	7	OK

9. Insertion des éléments dans le main

création du main sans les autres composants	avoir un main qui gère la fenêtre d'affichage	1	OK
insertion des différents menu	pouvoir passer d'un menu à l'autre et de retourner jusqu'à atteindre le jeu	7	OK
gestion du combat et des différentes animations	gérer les animations qui se complètent ainsi que leur ordre d'importance	7	OK

DIAGRAMME DE PERT – PROJET « LA BAGARRE »



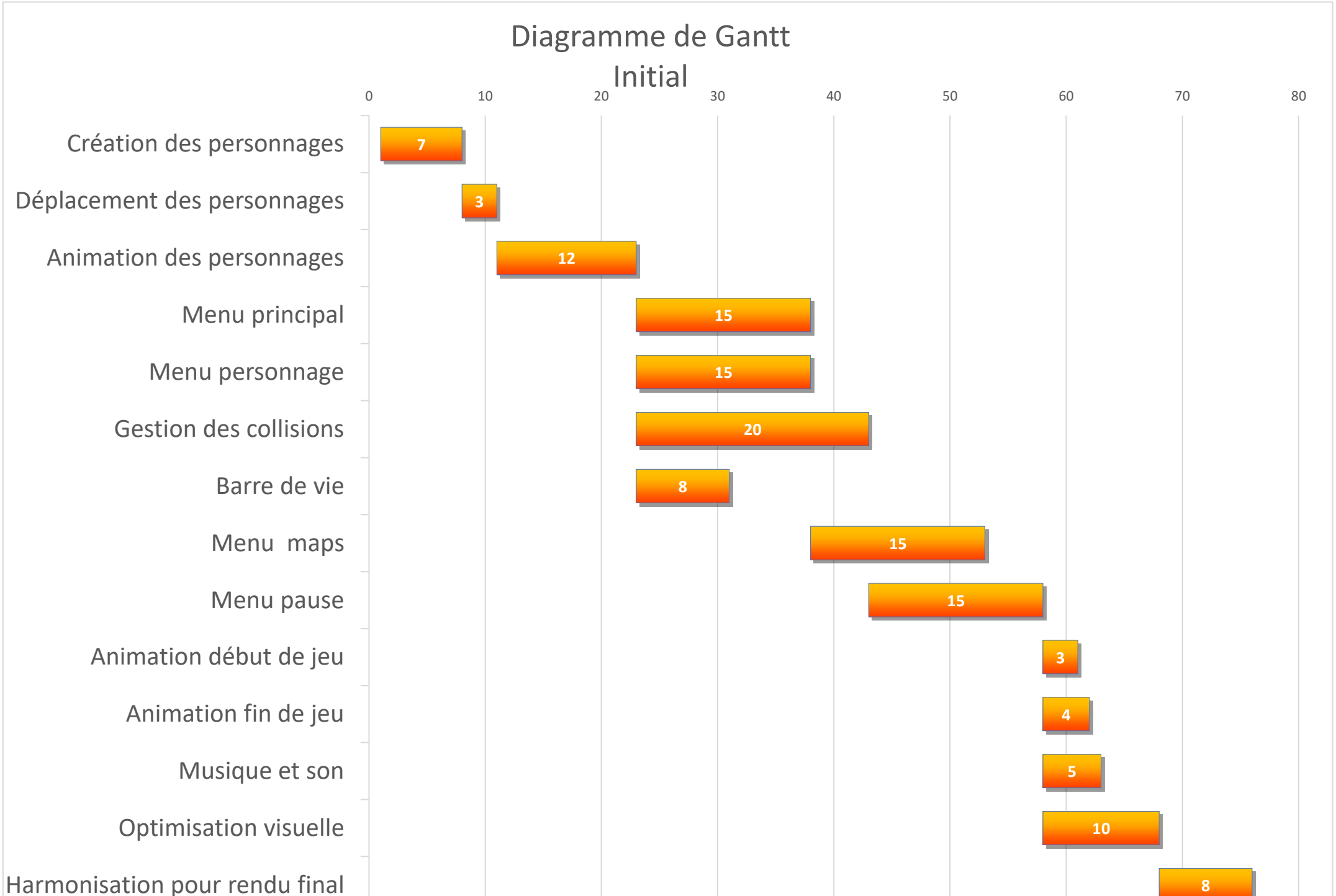


Diagramme de Gantt
Initial
0
10
20
30
40
50
60
70
80
Création des personnages
7
Déplacement des personnages
3
Animation des personnages
12
Menu principal
15
Menu personnage
15
Gestion des collisions
20
Barre de vie
8
Menu maps
15
Menu pause
15
Animation début de jeu
3
Animation fin de jeu
4
Musique et son
5
Optimisation visuelle
10
Harmonisation pour rendu final
8

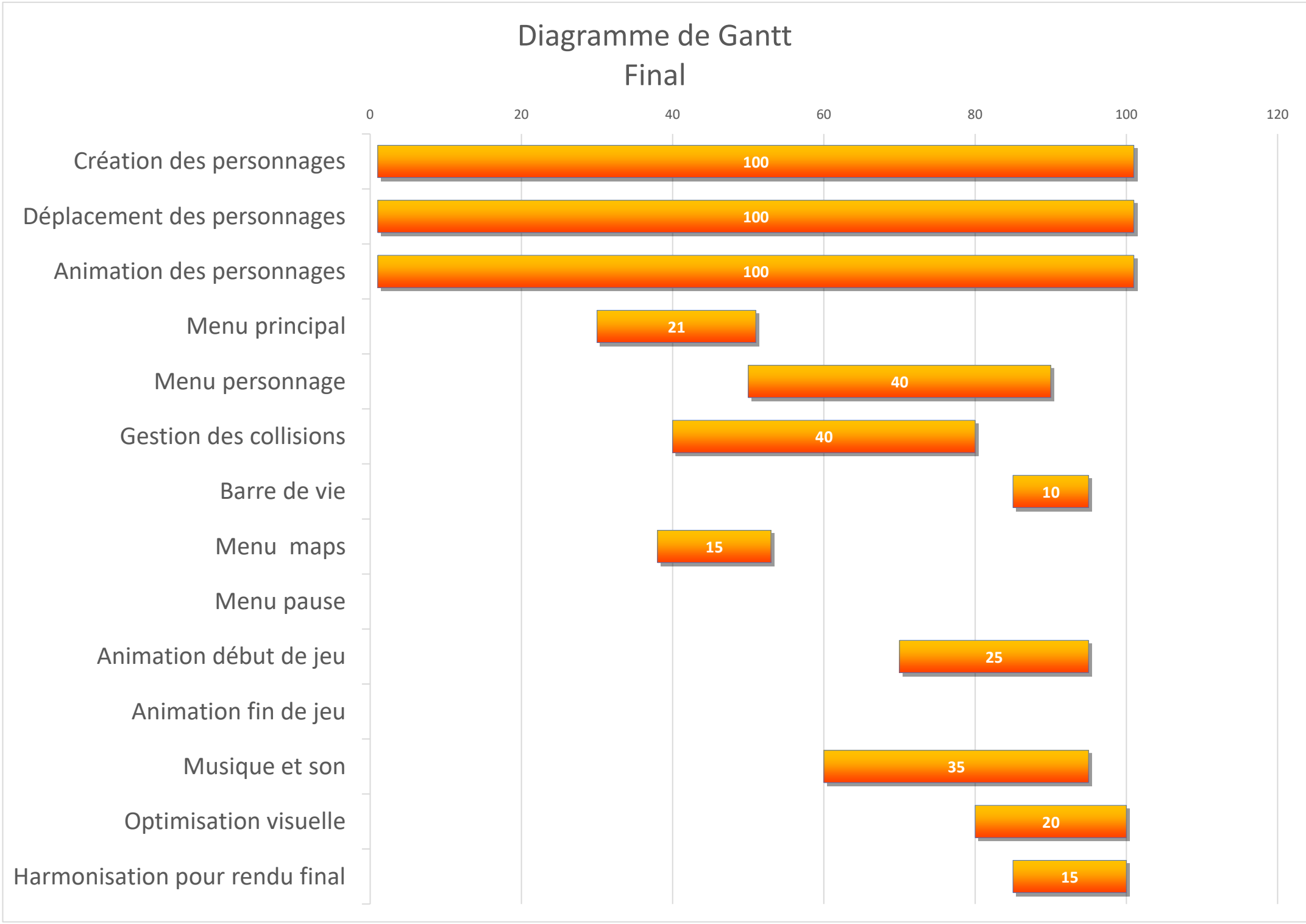


Diagramme de Gantt
Final
0
20
40
60
80
100
120
Création des personnages
Déplacement des personnages
Animation des personnages
Menu principal
Menu personnage
Gestion des collisions
Barre de vie
Menu maps
Menu pause
Animation début de jeu
Animation fin de jeu
Musique et son
Optimisation visuelle
Harmonisation pour rendu final
100
100
100
21
40
40
10
15
25
35
20
15

Retour d'expérience

Lors du commencement du projet, nous avons précisément défini les tâches à réaliser et les objectifs à atteindre. Cependant, nous n'avons pas correctement anticipé la demande de travail pour la programmation des méthodes. C'est-à-dire que, sachant nos connaissances alors limitées, nous n'avons pas pu avant de nous documenter et de réaliser le code, avoir conscience de la quantité de travail pour chaque tâche. Si l'implémentation de la musique n'a pas été complexe, en revanche il a été très compliqué d'intégrer plusieurs personnages et nous avons dû utiliser du polymorphisme par exemple, et il nous était difficile au début du projet de comprendre les mécaniques à utiliser pour coder ces fonctionnalités. D'autre part nous avons construit notre code progressivement : nous avons d'abord créer un seul personnage et fait évoluer le jeu avec ce seul personnage pendant plus de la moitié de la durée du projet, et ensuite nous avons ajouté de nouveaux personnages. La tâche « Création des personnages », originellement prévue pour une durée très limitée, s'est étalée tout le long du projet. Ainsi, Miguel et Anthime ont travaillé toute la durée du projet sur le moteur de combat, les collisions et sur les personnages dans le but de toujours optimiser le jeu pour qu'il soit le plus qualitatif possible.

Aussi si nous avons finaliser le projet avec des fonctionnalités supplémentaires que prévu, nous n'avons pas coder de menu pause par manque de temps, en tant que fonctionnalité la moins importante du jeu, et dans notre cas, moins utile que le reste des fonctionnalité.

Cependant nous sommes arrivés à coder 90% des fonctionnalités prévues. Les réunions hebdomadaires ainsi que la liste des tâches très complètes nous ont permis de ne pas perdre nos objectifs de vue. Nous nous sommes beaucoup investi dans le projet et cela nous a permis de principalement rebondir à chaque problème rencontré, en adaptant la quantité de travail nécessaire pour les tâches afin de ne pas retarder le projet.

UNIVERSITÉ PARIS SUD | IUT D'ORSAY

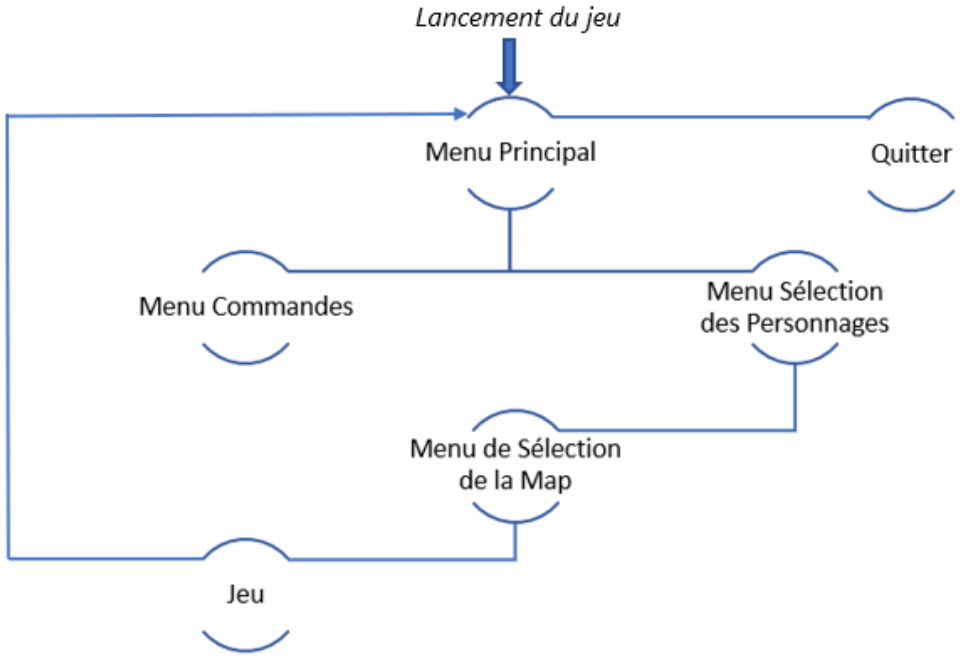
université PARIS-SACLAY

Réalisation technique

DEVELOPPEMENT DU JEU « LA BAGARRE » »

1. Navigation au sein du jeu

Notre jeu est composé de différentes pages dans lesquelles nous pouvons naviguer de la façon suivante :



Chaque menu ainsi que le combat, a son propre développement et son codage : il affiche ses éléments et réagit à ses interactions. Pour cette raison, chacun des états du jeu affiché ci-dessus dispo de sa propre classe pour les différents menus, et le jeu de combat est orchestré par plusieurs classes et sous-classes. Il nous faut donc : passer d'un état à l'autre du jeu au sein d'une même fenêtre, et, mémoriser les informations transmises tout le long de la navigation entre les fenêtres. En effet, en choisissant un personnage dans le menu de sélection des Personnages, il faut que celui-ci apparaisse lors du combat, et que le décor choisit dans le Menu de Sélection de la Map soit celui du combat.

Nous avons donc créer une classe permettant de gérer tous les états 1 à 1, et les uns entre les autres. Elle est nommé GestionFenetre et contient dans son code tout d'abord la création de la fenêtre en mode plein écran, et les déclarations des 4 objets des classes menus correspondants. Une variable initialisée préalablement est à l'origine de la navigation entre les différents états : chaque valeur de la variable entre 0 et 4 correspond à l'utilisation de méthodes spécifiques à un état, et donc à une classe spécifique tout au moins. Ainsi, la variable est initialisée à 0, on appelle donc le menu principal et toutes les

méthodes permettant sa gestion, son affichage et les interactions avec l'utilisateur, jusqu'à ce que le joueur utilise la commande pour passer au menu de sélection des personnages et que la variable soit attribuée à 1 : alors ce sont les méthodes du menu de sélection des personnages qui sont appelées. Si le joueur veut revenir en arrière grâce à une méthode du menu de selection des personnages, la variable se voit alors affectée la valeur 0. De même pour les 3 autres états du jeu ayant chacun correspondant valeur propre de la variable.

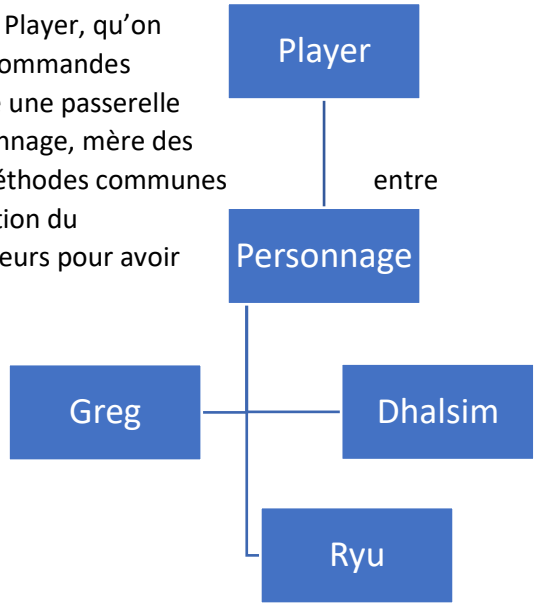
Les deux joueurs choisissent leur personnage respectif, un décor au sein des menus, navigue dans le jeu jusqu'à l'état du combat. Toutes les informations ont été retenues et peuvent alors être utilisées pour les trois classes de l'état combat : Player, Scene et Personnage. 2 objets de la classe Player sont créés, un objet Scene est créé ainsi que 2 objets de la classe Personnage prenant en paramètre les variables enregistrées le long de la navigation.

A la fin du combat, la variable de gestion d'état du jeu se remet à zéro et le jeu revient au menu principal. Toutes les informations enregistrées de chaque classe sont réinitialisées pour pouvoir recommencer une nouvelle partie pour ne pas avoir les sélections, les personnages, et le décor toujours enregistrés.

2. Le Combat

Conception du combat

Le combat oppose deux joueurs, pour pouvoir sélectionner les joueurs, on a du passer un personnage en paramètre de Player, qu'on passera par référence. Le player se contente de lancer les différentes animations de son personnages en fonction des commandes rentrés au clavier ou à la manette. On utilise des booléens pour ces différentes commandes. La classe agit donc comme une passerelle entre l'utilisateur et son personnage. Pour programmer les personnages, nous avons utilisés une classe abstraite personnage, mère des différentes classes de personnages (ryu, greg, dhalsim). Cette classe nous permet de rassembler tous les attributs et méthodes communes entre ces personnages. Cependant, nous avons eu un problème car l'implémentation des méthodes était différentes en fonction du personnages (notamment les méthodes d'animation). Nous avons donc dû utiliser des méthodes virtuelles et des pointeurs pour avoir une résolution dynamique de ces même méthodes.



Programmation du personnage

Pour pouvoir animer un personnage, il suffit de sélectionner les différents Sprite sur la texture chargé par ce même personnage et les afficher à une cadence géré par une horloge.

Ce schéma se repetera pour chaque animation avec un nombre d'images variables.

Pour pouvoir gérer les collisions, nous avons utilisé des zone de détéction, hitbox et hurtbox

3. Menu principal

Le menu principal est l'état du jeu affiché dès l'ouverture de la fenêtre. Il permet au joueur, à l'aide des flèches du clavier pour sélectionner, et à l'aide du bouton entrée pour valider, de choisir entre « Jouer », c'est-à-dire passer au menu de sélection de personnage, « Commandes » pour voir les touches du clavier à utiliser pour jouer, et « Quitter » pour fermer la fenêtre.

Le menu principal est mis en place par une classe qui lui est propre : la classe MenuPrincipal. Celle-ci est composée d'éléments visuels : d'un titre, d'une image de fondet des images des cases sélectionnables en mode « non-selectionnées » et « selectionnées ». Tous les sprites ont été réalisés à l'aide du logiciel GIMP. La classe possède aussi un attribut entier appelé selection qui indique sur quelle case est positionné le joueur et quelle action effectuer si il appuie sur la touche entrée :

-Lorsque le joueur est positionné sur « Jouer » le sprite correspondant est rouge et les autres gris. Si le joueur appuie sur entrée, la variable de selection des états du jeu passe à 1 ; la boucle de gestion de la fenêtre nous sort du menu principal et nous amène au menu de sélection des personnages.

-Lorsque le joueur est positionné sur « Commandes » le sprite correspondant est rouge et les autres gris. Si le joueur appuie sur entrée, la variable de selection des états du jeu passe à 3; la boucle de gestion de la fenêtre nous sort du menu principal et nous amène au menu des commandes.

-Lorsque le joueur est positionné sur « Quitter » le sprite correspondant est rouge et les autres gris. Si le joueur appuie sur entrée, la fenêtre de jeu se ferme.

4. Menu commandes

Le menu commandes consiste uniquement en l'affichage d'éléments visuels statiques pour afficher les sprites des commandes des joueurs créés à l'aide du logiciel GIMP. Une méthode nommée retourMenu nous permet de revenir au menu principal en appuyant sur la touche echap.

5. Menu map

Le menu map offre au joueur le choix entre 6 maps différentes. Une fois la sélection faite , la map choisie est chargée par les méthodes de la classe Scene qui permet de définir le décor du combat et la hauteur des personnages paramétrée selon chaque map.

Evolutions possibles

Par la suite nous pourrions ajouter de nouveaux personnages, de nouvelles attaques, créer nos propres personnages, nos propres game design, et des designs de menu plus attractif.
Nous pourrions aussi créer une fonction pour choisir aléatoirement un personnage ou une map.
Un menu pause pendant le jeu permettrait de mettre en pause le jeu et de revenir au menu.
Modifier le volume de la musique et des sons pourrait rendre le jeu plus qualitatif
Il serait aussi idéal de pouvoir jouer seul contre un robot.

Conclusion

Nous nous sommes tous beaucoup investi lors de ce projet et sommes fiers de vous rendre ce jeu qui, nous l'espérons, vous plaira. Il correspond au cahier des charges identifié au début du projet. Nous avons tous beaucoup appris sur le fonctionnement du langage C++ et SFML et ressortons grandis de ce projet. Le projet fut très plaisant à réaliser.