



***Projet L2AN1***



***BlackJack (L2AN1)***

***Les informations d’identification du document Les éléments de vérification du document***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Référence du document :** | D3 |  | **Validé par :** | David Janiszek |
| **Version du document :** | 1.00 |  | **Validé le :** |  |
| **Date du document :** | 19/04/2021 |  | **Soumis le :** |  |
| **Auteur(s) :** | Stefanos Alexandre, El Mougharti Hanafi, Couquet Samuel, Vallée Arthur |  | **Type de diffusion :** | Document électronique (.odt) |
|  |  |  | **Confidentialité :** | Réservé aux étudiants UFR Maths-Info de l’université Paris Descartes |
| ***Les éléments d’authentification*** | | | | |
| **Maître d’ouvrage:** | David Janiszek |  | **Chef de projet :** | Frank Sauvage |
| **Date / Signature :** |  |  | **Date / Signature :** |  |

1) PRESENTATION

1. Pourquoi avoir choisi ce projet et qu’est-ce que le blackjack ?

Notre projet est le L2AN, qui correspond au jeu du blackjack. Il correspondait au premier choix de tout le monde, ce qui nous a permis de commencer le développement du jeu très rapidement et nous a motivé tout le long.

Nous sommes tous les quatres des amateurs de BlackJack et nous voulions également apprendre à créer un jeu de A à Z, c’est donc pour cela que notre choix s’est tout naturellement porté vers ce jeu.

Voici un bref résumé du BlackJack, qui est un jeu de casino se jouant avec des paquets de cartes traditionnels :

Le but du blackjack est d’obtenir un meilleur score que le croupier. Pour cela, on peut tirer autant de cartes qu’on le souhaite, mais il est interdit de dépasser le score de 21. Notre score s’obtient en faisant l’addition de nos cartes. Toutes les cartes valent le numéro qui leur est associé (2 vaut 2 points par exemple), le valet, la dame et le roi valent 10 points et l’As vaut 11 ou 1 (Il vaut 11 si avec cette valeur le score du joueur ne dépasse pas 21, sinon il vaut 1). Dans un premier temps, le joueur doit miser, puis il y a la distribution des cartes, on commence avec deux cartes, le croupier commence également avec deux cartes, mais une seule est révélée. Ensuite il y a deux phases. D’abord on peut  choisir de tirer une carte ou de rester avec notre score. Il est également possible de “doubler”, c’est à dire que si la banque le permet, on double notre mise et on pioche une et une seule carte, on ne peut pas en piocher d’autres et vient ensuite le tour du croupier (on utilise doubler généralement quand notre main de départ vaut 11 ou 10).Si l’on dépasse 21 en tirant des cartes, on perd automatiquement. Une fois que l’on a choisi d’arrêter de tirer, vient la deuxième phase, qui est le tour du croupier. Le croupier suit un algorithme très simple : tant qu’il a un score inférieur à 17, il continue de tirer. Si jamais il passe au-dessus de 21 et que le joueur a un score encore réglementaire, alors le joueur gagne. Si le croupier possède un score réglementaire supérieur au joueur, le croupier remporte, et si jamais ils ont le même score il y a égalité.

1. Explication du choix du langage de programmation.

Nous avons choisi de réaliser notre projet en utilisant le langage de programmation Java, en raison de notre UE programmation orientée objet de ce semestre qui est un cours nous apprenant les bases de ce dernier. Ainsi, nous n’avions pas besoin de mélanger deux langages durant notre semestre et nous pouvions progresser dans nos deux cours respectifs. De plus, notre encadrant M.Sauvage nous a également déconseillé de travailler avec deux langages informatiques différents dans le même semestre si cela était possible.

Pour l’environnement de développement du jeu, notre choix s’est porté vers libGDX grâce aux recommandations de M.Sauvage, ce qui s’est avéré par la suite être un très bon choix malgré quelques difficultés passagères au commencement du projet.

1. Pourquoi github et non la forge ?

Pour ce projet nous avons choisi de nous orienter sur GitHub, qui est un service d'hébergement et de gestion de développement de logiciels. Notre choix s’est porté vers GitHub car l’un des membres de notre groupe, Alexandre avait l’habitude de ce service. Il a donc introduit au reste du groupe la façon d’utiliser GitHub dès le début pour que le développement du jeu commence le plus rapidement possible. Cela nous a fait gagner beaucoup de temps.

2) Avancement du Projet

1. Initiation aux Classes libGDX et à l’organisation des dossiers
2. Apprentissage de Git

Pour apprendre facilement et rapidement à utiliser GitHub à ces coequipiers Alexandre nous a introduit les commandes que l’on devait absolument connaitre et maitriser. Elles sont les suivantes: git push( push : mettre à jour le projet sur GitHub) et git pull( pull : permet de récupérer les mises à jour faites par les camarades sur le GitHub) . En plus de ces commandes, il s’est rajouté d’autres qui était aussi essentiels pour réaliser un push. Par exemple: git restore(permet de supprimer le code rajouté depuis le dernier le pull), git log (qui permet de voir les derniers pull), git add (ajoute le code pour ensuite le push).

1. SplashScreen et Initiation au Gradle

Le développement du projet a commencé avec la création d’un écran de chargement qui apparaît dès le lancement du jeu. Cela nous a apporté les premières difficultés. Le premier problème a été de rajouter un Engine à notre projet pour pouvoir coder le l’écran de chargement. On est passé par la modification du Gradle qui nous a apporté encore plus de soucis. Finalement on a réussi à régler cela assez rapidement en créant un dossier libs qui contient deux fichier en .jar de l’engine Tween Engine dans le projet.

1. Utilisation et compréhension des boutons

Pour n’importe quel jeu, et encore plus pour le BlackJack, nous étions obligés d’utiliser et de comprendre les boutons. Nous avons donc dans un premier temps utiliser les boutons pour naviguer à travers différents écrans du jeu. Cela nous a beaucoup aidé pour travailler plus efficacement. Ensuite l’utilisation des boutons était surtout indispensable pour avoir un jeu fonctionnel. On a dû passer par le logiciel BitMapFont pour mettre la police que l’on voulait sur les boutons. Nous avons créé les boutons grâce à gdx-texturepacker. On a dû s'y reprendre à plusieurs fois pour avoir de beaux boutons mais le résultat en vaut la peine. Donc pour résumer on a dû apprendre à utiliser plusieurs logiciels pour la création des boutons.

1. Création du MainMenu et des sous-menus

Après avoir terminé l’écran de chargement cela nous envoie directement sur le menu donc on a commencé la création du Menu. Pour une question d'esthétique on crée un premier background du menu. Grâce à la création du menu on a commencé à travailler chacun de son côté en créant toutes les autres classes nécessaires pour la division du travail.

1. Gestion du son et sa sauvegarde

Au départ on a ajouté les musiques et les différents sons des effets sans se soucier du niveau de volume. Ensuite on a défini des boutons pour gérer le volume de la musique. Mais la manière dont était géré le volume ne nous plaisait pas. C’est pour cela qu’on a utilisé de nouvelle sorte de bouton les sliders qui permet de choisir un volume très précis entre 1 et 100. Nous étions arrivés à notre but. Malgré cela on a été encore plus loin en sauvegardant le volume choisi par le joueur même s' il ferme le jeu.

Pour la sauvegarde on a utilisé les Préférences de libgdx. On a manipulé les données à sauvegarder dans un tableaux associatifs qui créer fichier où sont situé les données sauvegardées dans un dossier .prefs dans le répertoire public de notre pc qui contient. Ici le nom du fichier créer  pour le volume est AudioSettings. Chemin d’accès : C:\Users\pc\.prefs

En conclusion, notre système de gestion de l’audio est très complet ce qui permet au joueur une bonne expérience de jeu.

1. Création du Jeu en Terminal (Cartes.java / Deck.java, etc.)

Au bout de quelques semaines, nous avons décidé de rédiger des classes pour pouvoir jouer au BlackJack dans la console, pour ensuite implémenter ce code dans l’interface graphique. La nom de la toute première classe est Deck.java, elle permet d’initialiser les six paquets de cartes nécessaires au bon déroulement du jeu, à les mélanger et à les retourner sous forme d’ArrayList. Il y a trois ArrayList, l’une est composé des entiers correspondant aux valeurs des cartes (Par exemple le roi est transformé en 10), une autre est composé du noms des cartes sous forme de String (RoiTrefle, DixCoeur), et enfin la dernière est de type Animation afin que l’on puisse s’en servir lors de l’implémentation du code.

Ensuite, vient la classe Player.java qui permet de stocker les informations liée aux joueurs, comme sa banque, son nom, sa mise, et des méthodes permettant de déterminer s’il a fait un blackjack, etc… Nous n’avions pas prévu de rédiger cette classe au début, mais elle s’est avérée obligatoire lorsque nous avons voulu rendre le code viable pour un nombre de joueurs supérieur à 1.

Enfin, la classe la plus importante, Cards.Java. Elle permet de distribuer les cartes, faire piocher les joueurs s’ils le souhaitent, contient l’IA du croupier et détermine les gagnants/perdants à la fin de la partie. Cette classe était l’une des plus complexes à faire, notamment car il fallait l’adapter pour 1 à 6 joueurs (le blackjack se joue à maximum six). Nous avons eu beaucoup de mal à la rendre fonctionnelle pour plus d’un joueur, il a fallu adapter les méthodes, mais nous avons fini par réussir. Cependant, par manque de temps nous n’avons pas réussi à implémenter le code pour plus d’un joueur, ces changements n’ont donc en soi rien apporté au projet, mais cela nous a permis de comprendre et de progresser sur pas mal d’aspect en Java.

1. Implémentation du jeu dans l’interface graphique.

Une fois que le jeu était fonctionnel dans la console, il était nécessaire de l’implémenter graphiquement.Comme nous n’avions pas d’expérience,c’est l’un des autres problèmes complexes que nous avons eu lors de nos séances de travail. Il y avait des soucis, les cartes que l’on affichait ne correspondaient pas aux cartes que le joueur tirait, certaines règles n’étaient pas respectées, etc… L’aboutissement de cette partie est ce qui nous a pris le plus de temps au sein du projet. Au début nous avancions au même rythme sur l’implémentation en SinglePlayer et en Duoplayer, puis nous nous sommes rendu compte que nous n’aurions pas le temps de rendre fonctionnel le jeu pour plus d’un joueur, alors le Singleplayer est devenu notre priorité. Finalement, nous n’avons pas pu faire le jeu pour plus d’un joueur, mais nous avons laissé les boutons pour ces modes de jeu.

1. Transition de Gameplay.java -> SinglePlayer.java / DuoPlayers.java / TrioPlayers.java / QuadraPlayers.java
2. Gestion des Animations remplaçant les Textures

Tout d’abord, nous avons pris la décision de remplacer les textures par des animations dans le but de rendre le jeu plus vivant et qu’il soit plus agréable à jouer. Comme nous ne connaissions rien, il nous a fallu du temps pour comprendre comment celle-ci fonctionnait, et nous avons eu quelques problèmes la dessus, a commencer par la création de celle-ci. Contrairement aux texture ou une simple image que nous avons pu trouver libre de droit sur Internet nous suffisait, les animations nécessitent 6 images (voir plus mais pour ce que nous devions faire, c’était suffisant) et nous n’avons pas trouvé sur Internet nous avons dû créer ces images pour pouvoir avoir nos animations. Une fois toutes les images des animations créées pour chaque carte (6 images par carte), nous devions en faire une SpriteSheet pour chaque carte afin de pouvoir les appeler facilement dans notre code, nous avons fait ça grâce à TexturePacker. Ensuite, une fois que cette étape était terminée, nous avons dû chercher comment faire en sorte à ce que l’animation puisse s’afficher, pour cela nous avons cherché de l’aide sur le site de LIBGDX, ou encore certains tutoriel sur YouTube pour nous aider mais, aucun ne marchait car nous pensons qu’entre temps (Les sujets dataient de 2014-2015) il y a eu une mise à jour et la méthode était obsolète. Nous avons fini par chercher dans des forums et nous avons trouvé un sujet de 2017 qui présentait une autre manière de faire qui fonctionnait.

Pour remplacer toutes les Textures en Animations, il a d’abord fallu créer une nouvelle classe Animations qui à permit de pouvoir définir facilement les animation de TOUTES les cartes et qu’on a ainsi pu stocker dans une liste pour pouvoir ensuite les appeler dans notre code Principale.

Une fois que toutes nos animations étaient prêtes à l’utilisation, nous avons pu remplacer toutes nos textures par les animations et elles étaient fonctionnelles.

1. Gestion de l’Historique

Pour respecter les consignes de notre projet et pour notre propre envie constante d’améliorer le jeu, on a décidé de créer un historique des parties jouez dans tous les modes. Cela fonctionne pour tous les modes de jeu même s'ils ne sont pas tous disponibles. On a passé pas mal de temps sur ce sujet pour créer un algorithme de sauvegarde des parties le plus optimal possible. Le problème qui a été très dur a régler a été la gestion de toutes ces données créer. Finalement nous avons réussi à créer cet algorithme très compact et qui gère les données(c’est-à-dire supprime les données inutiles au fur et à mesure ) de la façon la plus optimale possible. Pour conclure, la création de cet algorithme a été très dure mais on a réussi grâce à la persévérance à proposer un historique parfait !

1. Exportation : Pourquoi ça n’a pas pu être possible

3) Contribution de Chacun + ce que ça nous a apporté

1. Alexandre Stefanos
2. Arthur Vallée

Pour ma part, j’ai décidé de m’investir le plus possible sur les animations du jeu, mais j’ai aussi aidé sur certains points ou mes camarades avaient certains problèmes. Le projet m’a apporté beaucoup de connaissances en Java et m’a fait découvrir certaines choses qui ne m'intéressaient pas forcément avant mais que maintenant j’apprécie.

1. Hanafi El-Mougharti

Pour commencer je voudrais dire que ce projet m’a beaucoup apporté en connaissance et m'a appris à coder et travailler en groupe ce qui sera très utile dans le monde du travail. Par rapport a m’a contribution dans ce projet j’ai pu travailler au début en duo sur la création de l’écran de chargement. On a ensuite décidé de se séparer des différentes tâches de travail. J’ai donc continué individuellement sur la création des boutons au niveau des textures et de l'animation finale des boutons. Puis je me suis orienté sur la gestion du son ainsi que la sauvegarde de ce dernier. Par la suite, j'ai réussi à réaliser entièrement l’historique des parties du jeu. Pour terminer j’ai aidé mes camarades dès lors qu’il avait besoin de moi pour pouvoir participer à la réussite de ce projet. Les attentes non réalisées par manque de temps nous déçoivent tous un peu j’imagine. Malgré cela nous sommes finalement tous très contents du résultat final !

1. Samuel Couquet

La majeure partie de mon travail au sein de ce projet, en plus de diverses tâches mineures, consistait en la rédaction du code pour rendre le jeu fonctionnel dans la console, avant de l’implémenter dans l’interface graphique. La création de ce code m’a permis de bien me familiariser avec le langage de programmation Java, auquel je n’avais jamais touché auparavant. Grâce à ce projet, j’ai pu me familiariser avec des interfaces graphiques, apprendre diverses composantes de la programmation qui me seront sans aucun doute utiles pour la suite de mes études.

1. Conclusion de Groupe