Travail fait par Matis et Gabriel

**NEON NIGHTBEAT**

**Document de conception**

Travail remis à Jean-Christophe Demers

1er Mars 2023

**Document de conception**

*Introduction*

Ce document est rédigé pour présenter une réflexion approfondie de notre projet ainsi que d’expliquer intelligemment nos choix. Nous allons séparer ces réflexions en plusieurs parties : l’interface utilisateur, les données externes, les structures de données, les patrons de conception, l’expression régulière, l’algorithme et l’équation mathématique. Ces parties vont être des idées détaillées sur ce que nous planifions de faire pour ce projet et elles peuvent occasionnellement être accompagnées d’images ou de liens pour approfondir notre raisonnement.

*Interface utilisateur*

Pour l’interface utilisateur, il va y avoir plusieurs maquettes à incorporer puisque notre projet contient un site web ainsi qu’un jeu fait avec Unity. Pour commencer, notre site web va avoir une page principale, qui va simplement être une page de connexion, accompagné d’une page pour s’inscrire si l’on clique sur un bouton dédié à cet effet sur la page de connexion. Cette page de connexion consiste d’un champ de texte pour le nom d’utilisateur, que nous allons vérifier dans la base de données pour s’assurer qu’il n’existe pas déjà et qui va être insensible aux lettres (case insensitive). Ensuite, juste en dessous, il va y avoir un autre champ de texte pour écrire le mot de passe, mais celui-ci va avoir un type d’input « password » pour garder la confidentialité du mot de passe pendant l’insertion. De plus, dans le cas de l’inscription, il va y avoir un champ additionnel pour confirmer le mot de passe. Finalement, il y a un bouton pour se connecter ou s’inscrire en bas de ces champs de texte, et voici deux maquettes que nous avons fait qui illustre bien ces aspects :

Graphical user interface

Description automatically generated

Graphical user interface, text

Description automatically generated

Site web

Ensuite, nous allons avoir deux autres pages sur notre site, et ces pages ne sont accessibles que si l’utilisateur est connecté. En effet, après s’être connecté, nous allons être redirigé sur une page, la page d’accueil, montrant une liste de toutes les musiques existantes sur le jeu accompagné d’informations, comme le nom de la musique, ainsi que les trois meilleurs scores du joueur, incluant le nom d’utilisateur s’il l’avait oublié. De plus, quand l’utilisateur clique sur la boîte qui regroupe ces informations, il va être redirigé vers la deuxième page, qui est celle des statistiques globales spécifiques aux musiques. Je parle de cette page au singulier puisqu’elle est faite pour s’adapter à la musique choisie, puisque la page commence vide et va se remplir en demandant de l’information à la base de données de la musique choisie. Cette page consiste à afficher la liste des meilleurs joueurs d’une musique et va être ordonné par le meilleur score, mais pas que puisque nous allons aussi pouvoir les ordonner par ordre alphabétique des noms d’utilisateur, par ordre de meilleure précision des notes, ou bien par meilleur « combo » de la partie. Enfin, cette page va aussi nous permettre à réutiliser la première page puisque les noms d’utilisateur du classement vont être cliquables et vont ramener vers la page d’accueil, mais avec les informations de ce joueur. Voici les maquettes pour ces deux pages, commençant par la page d’accueil :Graphical user interface

Description automatically generated

Page de statistiques globales :Graphical user interface

Description automatically generated

Jeu

Passons maintenant à l’interface utilisateur, qui va consister de 5 interfaces. Pour commencer, il y a la page d’accueil. Cette page est simplement composée de trois boutons : « Jouer », « Paramètres » et « Quitter ». Graphical user interface, text, application, chat or text message

Description automatically generated

Ensuite, quand nous cliquons sur « Paramètres », nous allons être redirigé vers une autre interface, où il y a un bouton « Retour » et où nous pourrons changer les paramètres du jeu, dont le volume général (%), le volume de la musique (%), le volume des effets du jeu (%), les touches du clavier reliés aux quatre touches du jeu, ainsi que la couleur primaire et secondaire des rangées, comme illustré dans la maquette suivante : A screenshot of a computer

Description automatically generated with medium confidence

Pour le cas du bouton « Jouer », nous allons être redirigé vers une interface qui nous demande de se connecter. Cette interface a un bouton, pour revenir au menu, ainsi que deux champs texte pour le nom d’utilisateur et le mot de passe, accompagné d’un bouton « se connecter » et « s’enregistrer ». Le bouton « s’enregistrer » va ouvrir la page internet de notre site web dédiée à la création de comptes décrite plus tôt pour que l’utilisateur se fasse un compte. Le bouton « se connecter » va vérifier si les champs ne sont pas vides et s’assurer que les informations sont correctes, dans quel cas le joueur va pouvoir continuer. Voici un exemple de cette interface :

Graphical user interface

Description automatically generated

Après s’être identifié, l’utilisateur va pouvoir accéder à l’interface de choix de musique. Cette interface consiste en une liste de longueur horizontale, qui pourrait éventuellement être descendue verticalement et qui ne prend pas toute la longueur de l’écran. Les musiques vont être affichés avec ces informations : nom de la musique, groupe de la musique, longueur de la musique et difficulté de la musique. Quand l’utilisateur va cliquer sur une musique, il va y avoir un menu qui va apparaître à côté de la liste des musiques et qui va afficher des informations, comme le classement personnel, le classement global (onglet pour changer de personnel à global). Ce menu va aussi afficher la liste de modificateurs du jeu pour ajouter de la difficulté et qui va notamment ajouter un multiplicateur de score pour récompenser le joueur qui rend le jeu plus dur. Enfin, il va y avoir un bouton « Jouer » pour commencer une partie sous ces éléments, donc voici un schéma simple représentant la description faite dans ce paragraphe :

A picture containing graphical user interface

Description automatically generated

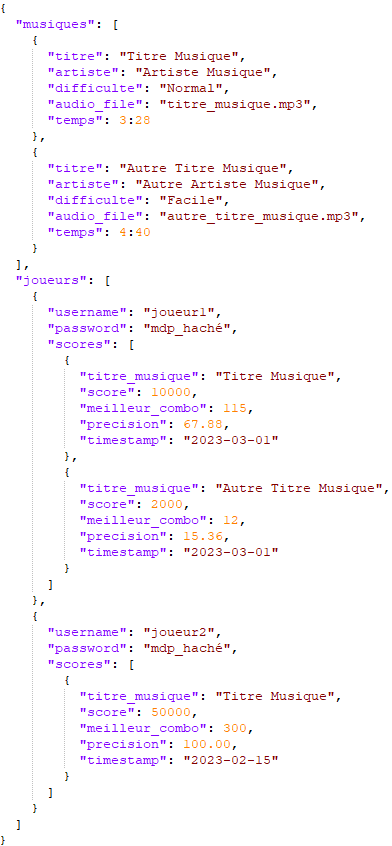
Finalement, après avoir choisi la musique, les modificateurs et d’avoir commencé la partie, nous arrivons dans l’interface de jeu. Cette interface affiche le jeu ainsi que les informations utiles, comme le score, le combo et la précision, comme dessiné dans cette maquette simplifiée :

Graphical user interface

Description automatically generated

*Données externes*

Pour échanger des informations à l’externe en entrée et en sortie dans notre projet, nous avons opté pour une base de données non relationnelle utilisant « NoSQL ». Notre choix pour le système de gestion de base de données est « MongoDB » et nous pensons que c’est un bon choix puisque ce système est orienté documents et est très flexible tout en supportant de grands volumes de données. De plus, nous apprenons comment utiliser MongoDB cette session-ci dans le cours d’administration de base de données, donc c’est une décision intelligente puisque la gestion de celle-ci va être plus efficace que si c’était un système de gestion de base de données que nous ne connaissions pas.

Pour la structure de cette base de données, nous allons avoir deux collections : les musiques et les joueurs. Les musiques vont contenir ces champs : « titre », « artiste », « difficulte », « audio\_file » et « temps ». Dans le cas des joueurs, leur collection va contenir : « username », « password » et « scores ». Scores est un objet qui contient plusieurs champs, comme « titre\_musique », « score », « meilleur\_combo », « precision » et « timestamp », mais il n’est pas une collection. Voici une représentation :

*Structure de données*

Pour nos trois structures de données, nous avons décidé de parler de l’utilité du array, du stack et de la liste. Pour commencer, un tableau va être utile pour storer nos objets « Note », puisque pour faire apparaître les notes avec notre algorithme, l’itération d’un tableau va être très utile pour filtrer les notes qui ont besoin d’être affichées. Par ailleurs, notre deuxième structure de donnée, la pile, va être utilisée pour stocker les notes touchées, ainsi ajoutant le combo du joueur qu’il avait quand il a raté dans une liste en comptant le nombre de notes qu’il y a dans la pile pour ensuite vider la pile, ou bien le combo qu’il avait quand le jeu s’est terminé. De plus, nous pourrons faire une deuxième pile appelée « max », qui va stocker les combos en même temps que la pile qui stocke les notes, mais celle-ci ne va stocker la donnée que si elle est plus grande que celle qui est dans la pile. Donc, en fin de partie, nous allons avoir le meilleur combo, ainsi que le nombre de notes que le joueur a réussi à avoir en additionnant toutes les valeurs dans la liste. La troisième structure de données va être simplement la liste, puisque comme nous venons de le mentionner, on va pouvoir utiliser la liste pour stocker toutes les valeurs de combo, et ainsi faire une addition de tous les combos pour afficher le nombre de notes total réussies. De plus, la liste va avoir d’autres utilités, dont le stockage de la précision des joueurs à chaque fois qu’il touche une note, pour ensuite faire une moyenne de toutes les valeurs dans la liste.

*Patrons de conception*

Pour les patrons de conceptions, nous avons choisi 3 patrons principaux et allons créer un patron nous-même. Un des patrons choisis est le « State », qui va être très utile dans notre jeu puisque ça va nous permettre d’implémenter un système pour pauser le jeu. Le state pattern va aussi pouvoir être utilisé pour signaler une partie finie, gagnée ou perdue, pour ensuite pouvoir afficher une page avec les statistiques de la partie. Finalement, le state pattern va être indispensable si nous voulons bien organiser le code pour les différents menus qu’il y a dans notre jeu et pour pouvoir ajouter facilement d’autres menus si besoin. Le deuxième patron choisi est le « Factory », qui va aussi être utile dans le cas de notre jeu puisque notre jeu va contenir des notes simples, ainsi que des notes qu’il va falloir maintenir. Dans notre cas, le factory pattern est un bon choix puisque ces types de notes différentes vont pouvoir être encapsulés dans une classe mère pour avoir les mêmes comportements, mais des principes de conception différentes. Comme dernier patron choisi, nous avons pensé à utiliser un patron « Observer », qui va être utile pour mettre à jour plusieurs informations qui seront affichées pendant que le joueur joue. Par exemple, quand une note est touchée, les classes de différents objets qui peuvent être changés en touchant une note, comme le score, le combo ou la précision vont être notifiés, puisque ces objets vont être dans la liste des observer de cette action et vont donc se mettre à jour dans l’interface utilisateur.

*Expression régulière*

Puisque les utilisateurs ont besoin d’entrer un mot de passe pour se créer un compte, nous voulons que ce mot de passe soit sécurisé, donc en plus du hachage du mot de passe, nous allons vouloir vérifier qu’il respecte certains critères. Ces critères sont : au moins huit caractères, au moins une lettre majuscule, au moins une lettre minuscule et au moins un chiffre. Voici à quoi ressemblerait cette expression : « ^(?=.\*?[A-Z])(?=.\*?[a-z])(?=.\*?[0-9]).{8,}$ ». Ensuite, si le mot de passe respecte ces critères, nous allons extraire le mot de passe et le hacher pour le mettre dans la base de données. Cette expression est simple, mais très utile, donc nous pensons que c’est un bon choix.

*Algorithme*

Dans les plusieurs algorithmes qui sont possibles pour ce jeu, celui dont nous allons parler est l’algorithme de vérification d’état de note. Ce que nous voulons dire par là, c’est que puisque nos notes vont être dans une liste qui contient tous les objets note, nous ne voulons pas pour autant toutes les afficher. C’est ici que l’algorithme vient en jeu, puisque notre algorithme va aller vérifier la valeur de la position y d’une note, pour ainsi vérifier si la note est dans la zone de jeu et l’afficher si la valeur y est valide, mais si l’algorithme détecte que la note est descendue trop basse, la note va compter comme manquée et nous allons pouvoir la détruire de la liste. De plus, cet algorithme pourra aussi être utile pour un éventuel modificateur « ghost », qui ferait en sorte que les notes apparaissent plus tard pour donner une fenêtre moins large de réaction à la note, en donnant une limite différente en position y.

*Équation mathématique*

Puisque nous allons pimenter notre projet avec des modificateurs, il y en a qui peuvent utiliser des équations mathématiques. Ce que nous avons décidé pour l’instant, c’est de faire un mod « wave », qui va changer la trajectoire des notes pour qu’à la place de descendre en ligne, elles vont descendre en suivant une forme de vague. L’équation mathématique la plus similaire à cette description est la fonction sinusoïdale. Pour l’instant, si le joueur active le mod, la trajectoire de toutes les notes vont être la même, ce qui veut dire que l’amplitude (hauteur d’une oscillation), ainsi que la période (longueur d’une oscillation) de la fonction vont être la même pour l’entièreté des notes. Éventuellement, si nous avons le temps, nous comptons faire un modificateur appelé « random wave », qui va faire la même chose, mais chaque note va avoir une amplitude et une période différente. Voici à quoi pourrait ressembler le mod « wave » avec la trajectoire d’une note (ligne bleu) et la ligne noire qui représente la trajectoire initiale de la note :

