P1611978 LAFOIX Paul
P1611312 POLOCE Antoine
P1307400 RABUT Théo
P1410782 LANGLADE Maxime

Cahier Des Charges

Nom du projet

I- Presentation

Dans le cadre de l'UE LIFAP4 Conception et développement d'application nous avons choisi de developper un jeu. Avec les contraintes données et nos gouts nous avons choisis de se rapprocher d'une application musicale et ludique. Notre jeu sera composé d'un affichage graphique dynamique d'une partition, la musique qui tournera en audio et sera synchronisé avec l'affichage graphique et une détection de collisions pour gérer le score. Beaucoup de jeux de ce type ont été fait, avec des fonctionnalités ressemblantes à celles de notre projet, mais nous allons être original et rendre <NOM DU PROJET> unique.

Règles du jeux :

Sur votre écran, des « notes » vont defiler et vont avoir une certaine synchronisation avec le fichier audio que vous aurez préalablement choisi.

Le but du jeux étant de taper sur la touche de votre clavier au moment ou la « note » arrivera dans la zone de validation. Et ainsi d'exploser tout les records.

Un jeu se découvre et s'apprend. Si vous voulez plus de description ou de règles, le README qui se trouvera dans le rendu est fait pour ça.

II- Description de la demande

Au terme de cette UE nous voulons avoir un jeu cohérent, ludique et original. Il n'y aurait pas de sens à faire une application de simulation d'un instrument qui serait trop simple et sans difficulté nous allons donc avoir un rendu final amusant.

Pour cela, nous devons avoir :

- Un système de partition qui nous permettrait de lire un fichier audio et le transformer en un fichier lisible par un algorithme simple. (fichier type txt)
- Un menu qui permet de choisir le mode de jeux (Création/Jouer),les paramètres, la chanson, la difficulté…
- Un affichage graphique dynamique des partitions, les « notes » vont devoir arriver de loin et venir sur la barre de validation.
- Une gestion de collision entre la barre de validation et les notes arrivantes.
- Un mode création, à partir d'un fichier audio on peut créer sur une partition vide. Ce mode sera peut être l'unique moyen de créer des partitions qui seront compréhensibles par nos algorithmes de lecture (de jeux)
- Jouabilité via clavier.
- Si possibilité jouabilité avec guitare électronique (type GuitarHero)
- Plusieurs niveaux de difficultés pour un seul fichier audio :

Ex : Dragon Force - through the fire and flame

Niveau Expert, Intermédiaire, Débutant.

Le cahier des charges étant fait en prévisionnel nous allons sans aucun doutes modifier, enlever et rajouter des fonctionnalités dans notre application.

III- Contraintes

Pour réaliser notre application nous devons suivre une méthode de conception qui nous est fourni : la méthode AGILE. Certes cette méthode est la plus optimal mais selon quel critères ? Étant donné que nous n'avons pas vraiment de client.

Le langage utilisé est du C++, le C n'est pas à proscrire mais nous devons tout faire pour nous en passer.

Les librairies utilisées sont SDL2 pour le graphisme et winTxt pour la version texte.

Nous devons fournir une documentation claire et compréhensible, pour cela nous allons utiliser Doxygen.

Nous sommes 4 personnes à travailler sur ce projet, cela peut être vue comme une contraintes puisque nous devons être clair dès le début sur les normes de code (telles que les noms de variables, les noms de fonctions, les casses, etc…) pour être sûr que notre code sera compréhensible par chacun.

Encore une fois notre nombre nous oblige à maitriser l'outil de gestion de version : Mercurial.

Le travail attendu n'étant pas le même pour les groupes de 2/3/4 personnes nous devons rendre un produit fini et fidèle à nos attentes.

La conception d'une telle application nécessite une organisation très précise et une compréhension parfait de la phase de conception pour cela nous allons utiliser un diagramme des classes qui éclaircira chaque partie non triviale.

Une autre contrainte est aussi le temps, l'appréhension de la durée pour un travail donné est différente pour chacun.

Le code sera fourni à la société InfoGamesProf.

Encore une fois, les fonctionnalités de <NOM DU PROJET> vont évoluer, on ne peut pas vraiment prévoir quelle seront les librairies que nous allons utiliser.

Tâches

0 Version jouable (Win Txt)

- 0 Analyse de la version jouable minimale
  - 0.1 Les classes indispensables et le coeur du jeux
  - 0.2 Le format des fichiers de lecture/ecriture
  - 0.3 La synchronisation audio/visuelle
  - 0.4 UML et premier diagramme de Gantt
- 1 Conception architecturale et premiers Tests
  - 1.0 .h généraux à faire avec priorité sur le coeur du jeux défini en 0
  - 1.1 Affichage fichier avec defilement (Test format 0.2)
  - 1.2 .h plus spécifique et type de données efficaces
- 2 Conception détaillé, début de code et Tests
  - 2.0 Faire le jeu.cpp minimale
  - 2.1 Test et vérification des besoins
  - 2.2 Utiliser « afficher fichier avec defilement » (1.1) pour coder le reste des .cpp
- 3 Fin de code et upgrade
  - 3.0 Finition du code, compilation et test d'utilisation.
  - 3.1 Si validation, on passe à la version avec affichage propre et ludique.

(SDL,OpenGl…)

L'ordre des tests et de l'implémentation de nouvelles fonctionnalités est difficile a prévoir, mais nous avons ici le minimum à faire pour avoir un jeu fonctionnel, ce qui est notre but dans un premier temps.

1 Version affichage propre (SDL,OpenGL..)

- 0 Recherche librairies SDL, openGL.. selon les besoins.
  - 0.0 Recherche sur la librairies SDL (nous suffira t-elle?)
  - 0.1 declaration des fonctions d'affichage dans les .h et creation des nouveaux modules si besoin est.
- 1 Implementation de la version Affichage propre
  - 1.0 Code des .cpp correspondants au .h crées ou modifiés en 1.0.1
  - 1.1 Debug et test d'affichage basique
  - 1.2 Verifications de la cohérence audio/video
- 2 Test d'utilisation et validations
  - Test d'utilisation, en lien avec 1.1.2
  - Si validation, on passe en 2.0.0 avec une analyse des besoins orientés vers la jouabilité et le game design (le cycle pour implementer de nouvelles fonctionnalités ressemblera surement au cycle 0)

On arrive à un niveau prévisionnel un peu trop abstrait pour être précis mais on aura une idée générale du travail à faire et de l'organisation générale des taches.

2 Version creation

- 0 Analyse de la version avec mode création
  - 0.0 Analyse de la compatibilité possible avec la version précédente
  - 0.1 Possibilité de modifications de classes pour obtenir la fonctionnalités de creation
  - 0.2 Modification UML et Diagramme de Gantt
- 1 Conception Architecturale et premiers tests
  - 1.0 Creation des .h inexistant et modifications des .h utilisables
  - 1.1 Verification qu'il n'y est pas de double declarations.
  - 1.2 .h spécifiques et types de données efficaces
- 2 Conception détaillée, debut de code et tests
  - 2.0 Code des .cpp correspondants
  - 2.1 Verifications qu'il n'y est pas de double implementations (on ne RE-fait pas.)
  - 2.2 Compilation et debug
- 3 Fin du code et update
  - 3.0 Finition du code, compilation et test d'utilisation.
  - 3.1 Si validation, on retourne en 0 avec une nouvelle fonctionnalités à implementer selon les besoins et le temps restant.

3 Version Dingodingue

- Implementation de la version jouable avec une guitare électronique type Guitar_Hero
- Implementation du mode DingoHero
- …..

Il n'y a pas d'attributions de taches explicites, nous sommes une équipe qui communiquera, qui partagera son travail et qui avancera en tant que tel. Si la repartition du travail venait à être trop inégales, nous vous le ferons savoir lors de notre soutenance.