ELDER INTERNAL IGNITION

RAPPORT DE PROJET DE C

BRUNET-CARTEAUX M. – DRIEU C. – JEGAT R.

MONTJOIE H. – LE MOIGNE L.

3 EII B – S6

Sommaire

[Introduction 2](#_Toc452028051)

[Cahier des charges 2](#_Toc452028052)

[Structure du jeu 3](#_Toc452028053)

[Spécificités du jeu 4](#_Toc452028054)

[Conclusion 4](#_Toc452028055)

# Introduction

Elder Internal Ignition est un jeu de stratégie au tour par tour codé en langage C sous l’environnement Code::Blocks et fonctionnant sous Windows. Nous nous sommes tournés vers l’usage de librairies telles que SDL et libfreetype pour développer la partie graphique et avons usé d’un grand nombre de structures et d’énumérations pour coder le moteur du jeu.

Le jeu permet à deux joueurs de s’affronter sur une carte vue de dessus. Chaque joueur dispose de trois personnages dont il choisit le type (mage, voleur, guerrier …). Ces personnages disposent d’un certain nombre de points de vie, de déplacements et d’attaque propres à leur classe et s’affrontent au tour par tour. Il existe deux principaux types d’attaques (lancer de sort ou corps-à-corps) ; certains lancers de sort peuvent être lancés à distance. Le perdant est le premier à avoir ses trois personnages avec un nombre de points de vie nul.

Nos premiers objectifs étaient de créer un jeu proposant une IHM agréable et simple avec jusqu’à quatre joueurs et permettant de jouer en réseau. Si certains de ces objectifs étaient trop ambitieux pour que nous réussissions à les atteindre, nous avons tout de même été en mesure de développer une première version jouable stable.

# Cahier des charges

Bien qu’il ait évolué au fil du développement du jeu, nous avons rapidement établi un cahier des charges séparant les tâches à accomplir en différentes *milestones* et disponible dans le dossier *doc* du dépôt Git des développeurs.

Nous avons

# Structure du jeu

Afin de simplifier le codage pour l’ensemble du groupe, nous avons décidé de répartir chaque point distinct dans un fichier source spécifique. Chaque fichier source possède un fichier header qui lui est propre, à part le main.c. Au total, le jeu utilise 19 fichiers sources et 18 structures.

|  |  |
| --- | --- |
| **Structures** | **Contenu** |
| boolean | Définition du type boolean |
| Skins\_Charac | Ensemble de noms de fichiers indiquant les textures des personnages |
| Skins\_GUI | Ensemble de noms de fichiers indiquant des textures relatives à l'interface |
| Terrain | Un terrain permet de faire le choix entre une case forêt, montage, plaine, rivière, et de savoir si un personnage ou un sort peut le traverser |
| Case | Permet de connaître le type de terrain, ses coordonnées et si un personnage est déjà dessus ou non |
| Carte\_Jeu | Permet de connaître l'identifiant d'une carte et est composé d'un tableau de Case à deux entrées |
| Effet | Permet de connaître l'ID et le nom d'une attaque, ainsi que ses effets (ex: nombre de tours restant avant que l'attaque ne s'estompe, nombre de dégats subis, nombre de PV régénérés, etc...) |
| Attaque | Permet de connaître le nom d'une attaque et son ID, la mana utilisée par celle-ci, sa portée et la durée des dégats appliqués |
| Classe | Permet de connaître la classe d'un personnage, ses attaque, son nombre de PD, de PV et sa mana maximum |
| Personnage | Permet de connaître la classe du personnage, ses PD, PV, mana actuels, son score actuel de coups critiques, la case où il se trouve et les effets qui l'affectent actuellement |
| Competence | Permet de connaître l'ID et le nom d'un compétence, les points à dépenser pour acquérir une compétence, la compétence suivante sur le même niveau, les effets d'une compétence et si celle-ci est connue ou non |
| Effet\_Competence | Permet de connaître le nom d'un effet de compétence et son ID |
| Carte\_A | Permet de connaître le nom et l'ID d'une carte ainsi que sa taille |
| Game\_Package | Permet de connaître le numéro de la carte, les classes des personnages et leurs compétences, et les types de terrains de la partie initialisée |
| Joueur | Permet de connaître le nom du joueur et les personnages qu'il a choisi |
| Node Joueur | Permet de gérer une liste de joueurs |
| ListJoueur | Une liste de nœuds pointant vers des joueurs |
| Partie | Informations concernant la partie ( liste des joueurs et le nombre de tours). Permet de gérer les tours |
|  | |
| **Fichier.c** | **Contenu** |
| main.c | Initialisation du jeu et lancement de l'affichage |
| affichage.c | Code des fonctions liées à l'affichage de la SDL |
| attaques.c | Code des fonctions liées aux attaques |
| carte.c | Code des fonctions liées aux cartes |
| case.c | Code des fonctions liées aux cases |
| classe.c | Code des fonctions liées aux classes |
| ClassesXML.c | Code des fonctions d'acquisition des classes |
| competence.c | Code des fonctions liées aux arbres de compétences |
| controleur.c | Code des fonctions du controleur du jeu |
| deplacements.c | Code des fonctions liées aux déplacements des personnages |
| gamePackage.c | Code des fonctions liées aux gamePackage |
| Interface.c | Code des fonctions d'acquisition des classes |
| joueur.c | Code des fonctions liées à la structure Joueur |
| joueurList.c | Code des fonctions liées aux structures NodeJoueur et ListJoueur |
| partie.c | Code des fonctions liées à la structure Partie |
| personnage.c | Code des fonctions liées aux personnages |
| resultat\_aleatoire.c | Code des fonctions liées à la strcure de résultat aléatoire pour les coups critiques |
| terrain.c | Code des fonctions liées aux types de terrains |
| TerrainsXML.c | Code des fonctions d'acquisition des terrains |

# Spécificités du jeu

# Conclusion