**EPIC BATTLE NOOTERS**



**Sommaire**

1. Source d’inspiration  
   a – Noot-Noot  
   b – Combat tour par tour
2. Objectif du projet
3. Organisation du projet  
   a – Git

b – Répartition des taches

1. Grandes parties du projet

a – Création des personnages

1 – types

2 – attaques

b – interface combat

1 – affichage et choix des attaques

2 – personnage actif et inactif

1. Démonstration

**I)Source d’inspiration**

*a) Noot-Noot*

Le jeu que nous avons voulu créer est inspiré de 2 références.   
Tout d’abord, l’univers que nous exploitons est l’univers de « Noot-Noot », qui est une vidéo avec un petit pingouin (vidéo accessible [ici](https://www.youtube.com/watch?v=bnmAi53H520)). Cette vidéo à beaucoup été reprise pour plusieurs blague et nous l’apprécions. Ce personnage vient d’un dessin animé en stop motion, le personnage de base s’appelle Pingu.

*b) Combat au tour par tour*  
La deuxième référence est le jeu vidéo Pokémon. Ce jeu consiste à capturer des créatures nommé Pokémons et de les faires combattre au cours de notre aventure.   
  
Ne pouvant pas recréer un jeu aussi complet que Pokémon, on s’est dit que nous allons uniquement faire un jeu basé sur le système de combat du jeu Pokémon.  
  
Le système de combat en question est un combat au tour par tour, c’est-à-dire que les joueurs ne jouent pas simultanément les attaques mais chacun leur tour. Ce système de combat est bien sûr utilisé dans plein d’autres jeux, mais Pokémon est le plus connue et la référence que plus de gens peuvent avoir.

**II) Objectif du projet**

Une image contenant noir, obscurité

Description générée automatiquementTout d’abord, on pensait faire des combats de 5 Nooters (nom de nos créatures) contre 5 Nooters. Bien sur les 5 ne seront pas sur le terrain en simultané, cela inclus donc un système de remplacement à la mort de chaque Nooter. Chaque Nooter est une référence culturelle, comme le termiNootor inspiré de Terminator ou encore le Nootkia qui est inspiré du Nokia3310.

Nous avons aussi pensé au tout début faire un menu de choix des personnages, mais par manque de temps nous nous contentons de les prédéfinir dans le code. Donc chaque joueur aura ses personnages prédéfinis en brut dans le code. Nous avons donc fait une manière simple à adapter dans le code. Pour que les combats aient un certain sens, chaque Nooter est d’un certain type, et son type sera donc résistant aux attaques de certains types mais sera faible à d’autres attaques.

**III)Organisation du projet**

*a) Git*

On s’est créé un repository git afin de pouvoir tous travailler ensemble efficacement sur le même projet, de plus Qt Creator propose des commandes dans l’application pour lancer les différentes commandes git utiles pour le projet.

*b) Répartition des taches*

Avant de pouvoir commencer correctement à coder, on a fait un diagramme de classe afin de voir toutes les classes que nous devons créer. On a ensuite créé tous ensemble les classes pour avoir la base du projet.   
Une fois toute les classes créés et réunis, nous somme partie avec la répartition suivante :  
Louis s’occupe de l’interface globale de combat, Arthur fait la classe des types qui est complexe, et Théo s’occupe de la création des Nooters.   
Dans cette organisation, on est partie du principe que Arthur et Théo auront fini leurs taches rapidement, afin de faire les fonctionnalités dont Louis a besoin (exemple : fonction récupération du ratio d’attaques selon les types).

**IV)Grandes parties du projet**

*a) Création des personnages*

Pour la création des personnages, cela se fait dans le constructeur de la classe Nooter.



On crée donc un Nooter en lui passant dans le constructeur un QString avec son nom. Avec ce nom, on voulait faire un switch case dans le constructeur pour initialiser le Nooter, mais switch case n’est pas compatible avec les QString. On est donc resté sur un if … else if, dans lequel les personnages sont définis

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

1 – types

Chaque Nooter possède un attribut type. Cet attribut va définir dans quel « famille » se situe le Nooter, et cela va donc définir à quels types d’attaques il va être résistant ou faible.

2 – attaques

Les attaques sont très importantes pour les combats, car il faut bien choisir quelle attaque utiliser selon son type pour faire encore plus de dégâts à l’adversaire.   
Les attaques sont donc gérées dans la classe attack, et est caractérisé par :

- un nom

- une description (pour décrire l’attaque sur l’interface)

- quantité de dégâts infligés par l’attaque

-cout de l’attaque

-temps de rechargement de l’attaque (pour éviter de trop utiliser la même attaque)

-le type de l’attaque

*b) interface combat*

L’interface se compose en différentes parties. Le canevas compose l’entièreté de la fenêtre afin de pouvoir positionner nos éléments comme bon nous semble et en pouvant dessiné des formes dessus puisque l’esthétique des boutons et autres composants ne nous convenait pas et n’était pas bien adaptés à notre application. La partie supérieure contient les nooters ainsi que leurs attributs (barre de vie et d’énergie). La barre d’énergie de l’adversaire est masquée au joueur courant afin de ne pas dévoiler de potentielles informations précieuses ! Le fond est une image que l’on affiche avant tout le reste afin de ne pas écraser ce qu’on dessine sur le canevas.

Voici un aperçu de l’application :

Une image contenant texte, dessin humoristique, illustration

Description générée automatiquement

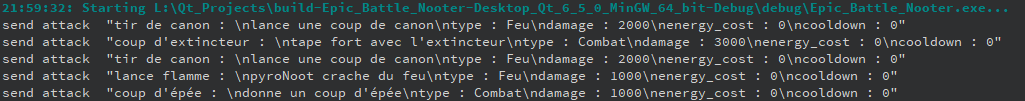
J’ai rencontré quelques difficultés notamment pour l’affichage d’animations, je ne savais pas vraiment comment m’y prendre alors j’ai essayé plusieurs choses avec des threads, etc... Mais au final je suis resté sur un bon vieux entier et une boucle while qui appelle la fonction repaint() pour faire quelque chose de sympa en fonction du temps courant de l’application et d’une durée d’animation .

Une image contenant texte, Police, logiciel, Logiciel multimédia

Description générée automatiquement

Entre les tours des joueurs, une fenêtre noire indiquant de passer au joueur suivant est affichée pour pouvoir effectuer le roulement. (Les animations et autres seront plus simple à comprendre après avoir vu la démonstration vidéo)

Pour sélectionner les attaques qui sont sur la partie basse de l’écran, on utilise les touches du clavier (les flèches directionnelles pour bouger et la touche espace pour sélectionner l’attaque à lancer). Durant le projet, on s’est beaucoup aidé des fonctions toString() développées pour afficher les données dans nos classes. Donc à chaque attaque lancée, un print en console est fait en utilisant qDebug().



Par manque de temps la gestion de l’énergie a été laissée tombée mais les attributs pour l’utiliser sont présent.

Je vous laisse regarder le code afin que vous constatiez l’organisation de celui-ci.

La gestion des images à afficher a aussi été un petit défi avec la gestion des ressources de Qt mais on s’en est sorti avec nos fichiers en inscrivant même des alias sur nos fichiers pour éviter du code à rallonge supplémentaire à l’appel de l’image pour l’afficher.

Sur la fin en voyant que le rendu approchait, et que les fonctionnalités n’étaient pas au rendez-vous, on a mis en place une initialisation aléatoire des nooters pour commencer une nouvelle partie différente à chaque lancement de l’application.

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Les éléments sont disposés comme vous pouvez le voir dans un gridLayout avec un spacer sur la première ligne afin de laisser la place pour l’affichage de la partie supérieure de l’écran.

Pour une question de lisibilité et de facilité de modification, on a défini nos variables utiles dans le .h afin d’y accéder facilement dans tout le fichier .cpp. Ça a été très pratique notamment pour le positionnement des éléments en fonction de la taille du background ce qui nous a permis de changer de fond facilement après que notre dessinateur attitré nous a fourni les versions finies des nooters et fond d’écran utilisés dans la version « finale ».

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

**V)Démonstration**

Pour que vous puissiez avoir un aperçu du projet, voici donc un lien vers une vidéo que nous avons fait. Cette vidéo sert donc de démonstration :