Eliott BABET

Alexandre POULAIN

Romain LANCELOT

Julien CERVELLERA

Charles SAURY

Promo 2021

Rapport : projet transverse

Semestre 4

Introduction

Nous avons réalisé un jeu de type RPG le but consiste en un héros choisit par l'utilisateur qui combat des monstres nommé Slimes. Ces slimes ont la capacité de se diviser. Il y a 3 catégories : un gros slime, qui se divise en deux moyens slimes, un slime moyen, qui se divise en 4 petits slimes. Pour que les 4 apparaissent il faut tuer les deux moyens. Chaque héros et monstre est attribué un système de points de vie et de dégâts qu'ils s'infligent mutuellement. Ce jeu un peu rétro force l'imagination et nous invite à une réflexion sur les possibilités que nous apporte le C++ et le système de classe dans nos codes futurs.

Il y a un système de fuite qui permet de reset le jeu afin de retenter sa chance !

I. Présentation des sources et explication du code

Tout d'abord, notre code est surtout orienté autour de l'utilisation des classes, l'explication des classes et de leurs fonctionnalités se fera dans la partie suivante.

Notre jeu se déroule de la manière suivante : au début de chaque tour une fonction, "Affichage()" est appelée afin de montrer au joueur l'état de chaque entité en vie, cette fonction est appelé jusqu'à ce que le combat se termine.

En premier lieu, notre joueur affronte un 1er slime, ce slime est le monstre le plus puissant présent sur ce code mais le heros n'aura aucune difficulté à l'éliminer. Ce combat se passe à l'intérieur d'une boucle while où il y a 2 conditions de sortie. La 1ère est si la vie du héros atteint 0 ou moins et la seconde condition est comme la première, si la vie du slime atteint 0 ou moins.

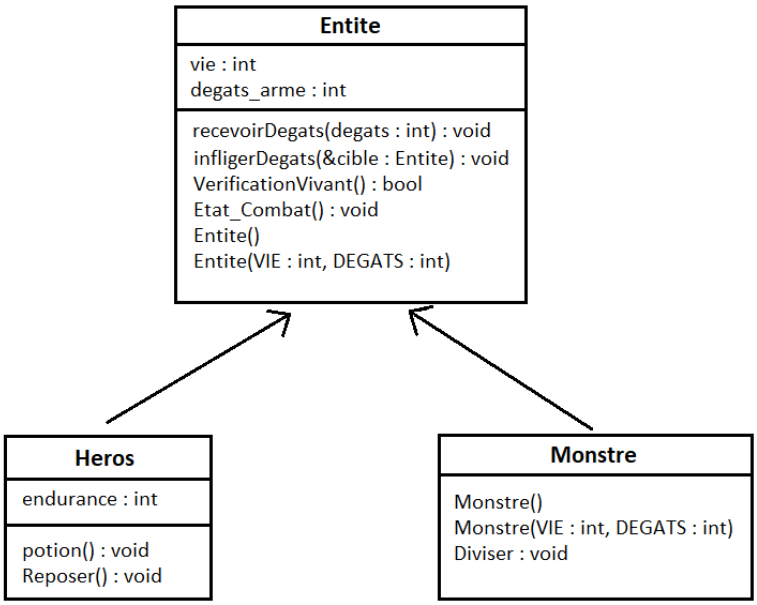
Ensuite, si le joueur élimine ce slime, on sort de cette 1er boucle et nous fait entrer dans une deuxième boucle où le joueur doit éliminer deux slimes qui sont moins puissants que le premier rencontré. Les conditions de sortie restent les mêmes à l'exception qu'il faut que les deux slimes meurent afin de pouvoir passer à l'étape suivante.

Enfin, l'étape finale est l'affrontement conte 4 slimes. Les conditions de sortie respectent la même logique que pour les deux boucles précédentes.

Un deuxième monstre fait ensuite son apparition, celui-ci a la particularité de pouvoir s'énerver lorsqu'il passe sous un certain seuil de point de vie.

Si le joueur a réussi à vaincre tous les monstres, il a gagné, dans le cas contraire il a perdu.

II. **Présentation du diagramme de classes**



Pour appliquer les fonctionnalités vues en cours et pratiquées en TD. Nous avons choisi de créer 3 classes distinctes. Au départ, nous avions seulement 2 classes : Heros et Monstre. Mais suite aux idées de vouloir rendre le héros et les monstres uniques nous avons voulu leur attribuer des caractéristiques propres. Dès lors, nous avons créé une 3eme classe, Entite, qui regroupe les méthodes et attributs communs aux Heros et Monstre qui vont hériter de la nouvelle classe. L'héritage nous a offert la possiblité d'avoir un code structuré qui pourra être compris beaucoup plus simplement.

III. **Comportement des classes**

Concernant les méthodes et attributs établies dans les classes nous avons fait en sorte qu'on comprenne leurs fonctionnalités à partir de leur nom.

A. La classe "Entite"

La fonction "recevoirDegats(degats : int)" renvoie une valeur qui correspond au point de vie moins les dégâts subis.

La fonction "infligerDegats(&cible : entite)" permet, en complément de la fonction "recevoirDegats(degats : int)", d'infliger un des dégâts à une cible. Ici l'utilisation de "&" permet de pouvoir directement modifier le paramètre de la fonction et non pas de modifier une copie du paramètre.

La fonction "VerificationVivant()" permet de vérifier si la vie de l'entité est supérieur à 0. Cette fonction permet dans le cas de l'héros de savoir si on continue le combat ou non et dans le cas du monstre de savoir s'il doit se diviser ou alors mettre fin au combat.

La fonction "Etat_Combat()" permet d'afficher, à la fin de chaque tour, l'état des points de vie de chaque entités.

B. La classe "Heros"

Ici en plus des caractéristiques héritées de la classe "Entite", on a donné à la classe "Heros" un attribut et deux méthodes.

La fonction "potion()" consiste à permettre au joueur de pouvoir regagner des points de vie.

La fonction "Reposer()" est une mécaniques dans le jeu qu'on aurait voulu ajouter. Cette mécanique repose sur le fait que le héros consomme plus ou moins d'endurances en fonction de l'attaque du joueur et lorsque cette endurance serait à 0, le joueur devra se reposer pendant 1 tour.

C. La classe "Monstre"

Dans le cas d'un projet plus conséquent, cette classe aurait servi de classe mère aux autres monstres qu'on aurait pu ajouter. Mais ici notre objectif était d'implémenter un seul type de monstre. Tout comme la classe "Heros" cette classe hérite des attributs et méthodes de la classe "Entite" mais la singularité de cette classe est la présence d'une méthode "Diviser()"

La fonction "Diviser()" est une fonction qui permet aux slimes, à leur mort, de pouvoir se diviser en deux slimes de rang inferieur donc qui possède moins de vie et moins de dégâts.

IV. Avantages et inconvénients du projet

A. Avantages

D’une part, nous avons pu nous familiariser avec le langage C++. Avec nos connaissances en C du semestre précédent, nous avons trouvé que le C++ était pour le sujet de notre projet un avantage majeur, en effet le lien entre les classes et plus particulièrement la fonction héritage qui facilitait la

création de personnage. Pouvoir scinder son code en plusieurs classes qui s'appellent les unes aux autres et l'orientation objet répondaient parfaitement au but de notre de jeu.

Ce langage nous offrait la possibilité de voir clair dans notre code et nous permettait de cibler et de corriger facilement nos erreurs.

La découverte du C++ nous a aussi permis de comprendre et d'utiliser la sécurisation des données dans le code. Cette notion de sécurisation des données pourrait nous permettre, dans le cadre de la création d'un jeu plus aboutie, d'empêcher toutes sorte de triche venant de l'utilisateur.

Aussi, les connaissances que nous a permis d'acquérir ce projet sur ce langage nous servira pour le futur, comme le projet Transverse, en effet, ces deux travaux ont pour but de faire un jeu. Ce mini projet nous a apporté beaucoup quant aux mécaniques à apporter sur un jeu.

B. **Inconvénients**

Comme vous pouvez le savoir, ce projet s'étendait sur un temps court. De ce fait, nous n'avons pas pu réaliser l'ensemble des idées que nous avions imaginés au sein de notre groupe. Cependant cela nous a donné l'envie de poursuivre en C++ et nous a donné des idées sur notre projet « Transverse » et a permis d'accroitre nos connaissances.

V. **Travail en équipe**

Le langage C++ a réellement facilité la division du travail en équipe. En effet, nous nous répartissions les taches lors des séances d'avancement en effectuant un « brainstorming » des idées que nous avions. Ainsi, chaque semaine nous faisions une réunion afin de savoir où nous en étions et pour savoir quelle idée appliquée pour la prochaine séance. Cela nous a permis d'établir un tableau des fonctionnalités à intégrer et de repartir les taches afin d'optimiser notre temps. Pour l'envoie de fichier ou pour partager des idées en dehors de ces réunions, nous utilisions Github pour partager le code ou nous communiquions à l'aide de logiciels comme Discord ou par Messenger.

Le système de classes nous a alors permis de nous répartir des taches autant dans le code que dans la présentation du projet. Nous avons trouvé que coder en C++ en équipe était un moyen bien plus efficace que le langage C. Cela s'explique du fait que c'est un langage bien plus divisé et organisé.

VI. **Problèmes rencontrés**

Lorsque nous codions, nous avons dû faire face à une ligne de code corrompue. Celle-ci nous a encombrées car malgré nos efforts, il nous a été impossible de trouver l'origine de l'erreur. De ce fait, nous avons dû modifier notre code afin que celui-ci puisse fonctionner sans problème.

De plus nous avons rencontrée certaines difficultés lorsque l'on devait permettre au joueur d'utiliser des fonctions contenant des variables qu'on a sécurisées au préalable. Comme l'utilisation d'une potion de vie qui avait pour objectif de modifier la vie du héros qui était une variable privée donc non modifiable par le joueur mais on a pu surmonter ce problème à l'aide du tutoriel d'Openclassroom concernant les "accesseurs".

Conclusion

Ce projet fut un moyen d'accroitre nos connaissances de façon interactives. Cela change des séances de TP classique et nous avons trouvé que c'est un moyen plus ludique et efficace d'apprendre le C++.

Dans l'avenir, nous aimerions continuer ce projet. Dans cet optique nous pourrons d'une part créer une interface qui rendrait le jeu plus interactif. De plus, nous voudrions continuer à développer la profondeur de nos personnages. Pour se faire, nous ajouterions un système de choix de classe, d'ajouter davantage de compétences, que ce soit pour les monstres ou pour notre héros. Enfin, pour envisager de partager ce programme à une échelle plus importante, nous voudrions ajouter un aspect graphique propre à cet univers.