

Rapport Qualité de programmation

Sujet : Robot Ecraseur

Groupe 9 : Yassine Goeller, Léandre Demuth, Victor Sauter, Victor Brungard

Sommaire :

I - Introduction

II - Cahier des charges

III - Méthodes de travail

IV - Réalisation

V - Conclusion

Cahier des charges :

Le langage utilisé sera du C++.

Les fonctionnalités attendues sont :

* Le joueur peut se déplacer dans l’une des huit cases autour de lui
* Un robot cherche à écraser le joueur
* Deux types de robots : des robots première génération qui ne se déplacent qu’horizontalement et verticalement, et des robots deuxième génération qui se déplacent aussi en diagonales
* Les robots ne sont capables de détecter qu’un être humain
* Quand un robot se déplace sur la case du joueur, il écrase le joueur et celui-ci meurt
* Quand un robot se déplace sur la case d’un autre robot, les deux robots dont détruis et la case contient alors un débris
* Quand un robot se déplace sur la case d’un débris, il est aussi détruit
* La partie se termine soit quand tous les robots sont détruits, soit quand le joueur est écrasé
* Créer un terrain
* Sauvegarder un terrain
* Charger la configuration d’un terrain
* Organiser une partie de jeu avec ce terrain avec un affichage en mode texte
* Affichage graphique (wingbi)
* Ajouter d’autre types de robots
* Ajouter des débris dans le terrain
* Avoir un joueur expert qui ne se déplace qu’horizontalement et verticalement

Introduction :

Ce projet est dans la carde de la matière qualité de programmation. Nous allons donc répondre aux attentes à l’aide de toutes les connaissances.

Le travail est à réaliser en groupe. Celui-ci est composé de Yassine Goeller, Léandre Demuth, Victor Sauter et Victor Brungard. Nous nous sommes réunis par affinité mais aussi car nous avons l’opportunité de pouvoir se voir facilement.

Il est donc indispensable de s’organiser. Pour cela, il faut une bonne communication et une bonne méthode de travail.

Méthodes de travail

1. Git

Lors de la création du projet nous avons introduit directement le logiciel de version Git.

Ce logiciel permet de faciliter le partage des ﬁchiers et permettre à tout le monde d’avoir en permanence la dernière version du projet.

1. Communication

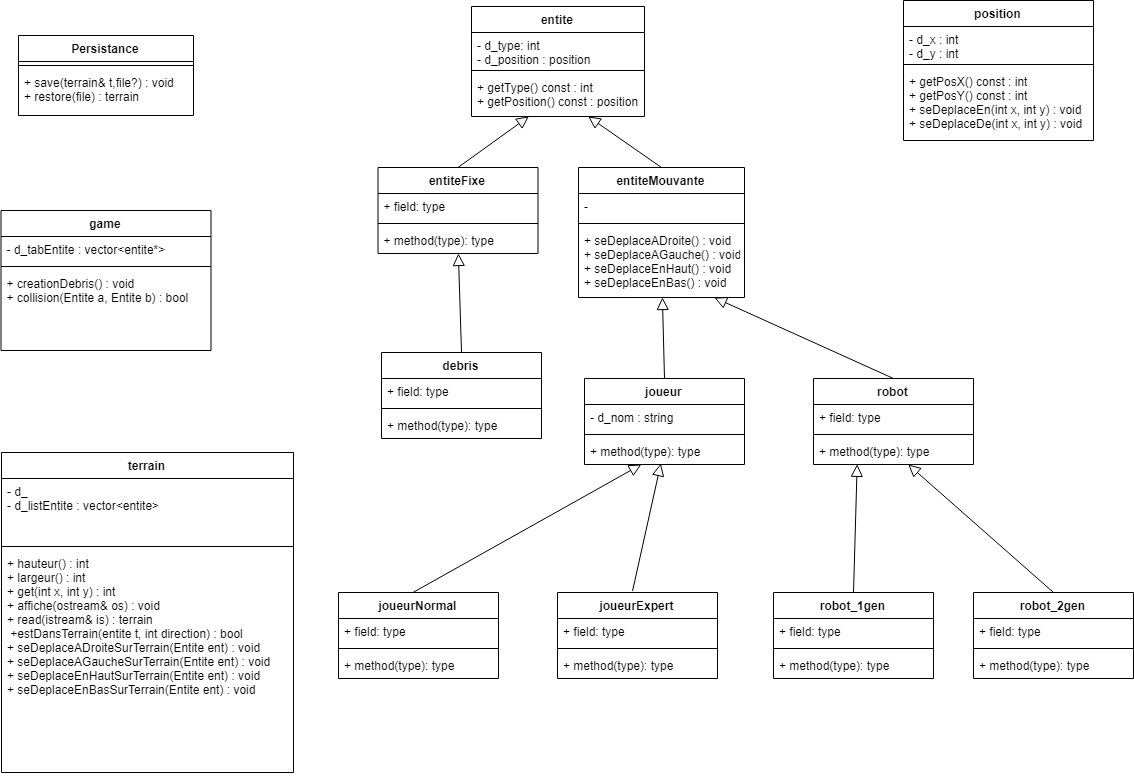
La messagerie instantané Messenger fut l’outil le plus utilisé dans ce projet. Elle nous a permis de communiquer et surtout d’avoir un historique des messages pour les personnes n’étant pas disponible à ce moment-là.

Des réunions hebdomadaires ont été mis en place. Nous parlons des problèmes rencontrés, de l’avancement des tâches qui avaient été distribués et des prochaines tâches pour la semaine à venir. Elles permettent donc un suivi du projet, et surtout de faire avancer le projet en résolvant le problème ensemble.

Réalisation

1. UML

Avant de se lancer tête baissée nous avons réfléchis aux différentes classes que nous auront besoin pour réaliser le programme.

Nous avons réalisé un UML :

1. Implémentation du jeux

La classe joueurNormal va créer un joueur avec nom, un position x et y.

La classe persistance va permettre de sauvegarder un terrain ainsi que de le restaurer.

La classe position va nous permettre de sauvegarder les coordonner x et y et avec des méthode mettre déplacer ces position a partir d’un vecteur ou de déplacer la position a une autre position.

La classe robot première génération va permettre de créer un robot avec une position x et y et avec une méthode on va pouvoir déplacé le robot sur le terrain et une autre méthode on pourra trouver la meilleure déplacement a faire pour que le robot suive le joueur.

La classe robot va permettre de créer une entité mouvante avec une position x et y.

La classe terrain va créer un terrain avec une largeur et une hauteur et on pourra avec une méthode afficher le terrain avec une autre méthode regarder si les entités sont dans le terrain enfin on pourra déplacer les entités mouvante dans le terrain avec tous les méthode déplacement.

La classe entite représente la classe mère de nombreuses sous-classes. Elle a une position noté d\_pos et a un entier noté d\_type.

La classe entite possède :

* Deux constructeurs, un qui prend en paramètre x et y qui sont deux entiers permettant de mettre à jour la position de l’entité. L’autre qui prend un paramètre une position.
* Un destructeur
* Une méthode getPosition() qui renvoie la position de l’entite
* Une méthode getType() qui renvoie un entier c’est-à-dire le type de l’entite

La classe entiteFixe est une classe dérivée de la classe entite. Elle permet de structurer les différentes classes qui vont suivent. L’entite ne pourra donc pas bouger. C’est pour cela que la classe débris dérive de cette classe.

Ensuite, il existe la classe entiteMouvante. A l’inverse d’une entiteFixe, cette classe possède des méthodes permettant de se déplacer. Les 4 méthodes sont seDeplaceADroite(), seDeplaceEnHaut(), seDeplaceAGauche(), seDeplaceEnBas().

Les fonctionnalités réalisées :

* Sauvegarder une partie
* Reprendre une partie
* Charger un terrain avec un fichier
* Fin de parties
* Affichage graphique et texte