Projet C++

SurvivorZ

Michele Bona
Rihab Khlifi

Polytech Main 4
Année 2020-2021

Description de l'application développée

Histoire et concept

Nous avons développé un jeu de gestion textuel que l'on a nommé SurvivorZ.

Dans ce jeu on gère un communauté de survivants à l'apocalypse zombie apparue à cause d'une nouvelle étrange mutation du Covid-19, qui après nous avoir poussé à l'isolation nous pousse maintenant à un rapprochement assez aggressif vers tout ce qui vit !

Mais tout le monde n'a pas été infecté ! Et après avoir erré un temps en ne pensant qu'à survivre, les derniers chanceux qui ont encore le contrôle de leur corps et sont privés d'une faim insatiable pour des cerveaux bien juteux se sont regroupés pour assembler leurs forces.

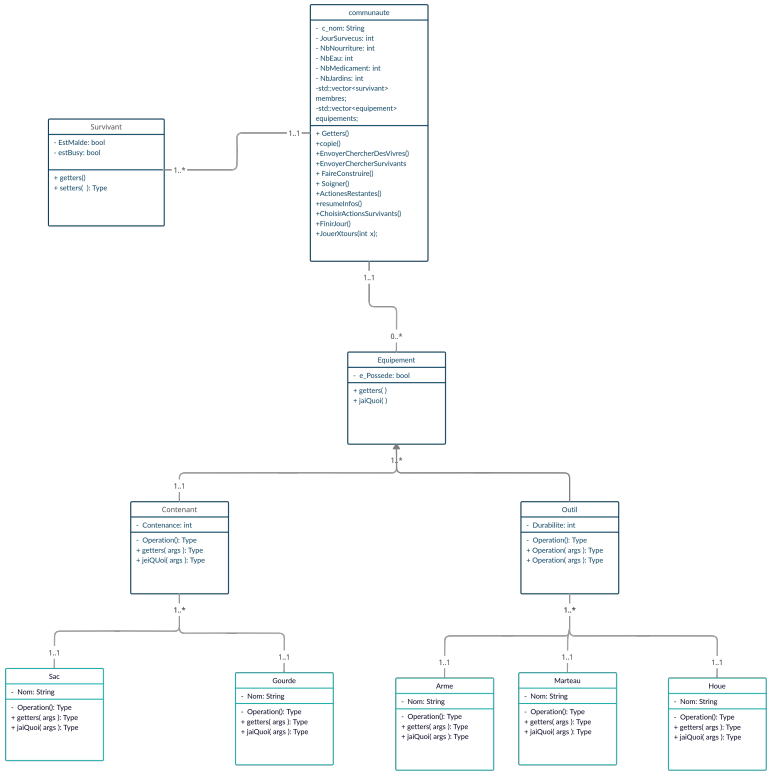
Ils se sont donc sédantarisés et commencent à créer des communautés, mais n'étant plus habitués à réfléchir à force de lutter dans cette apocalypse ils ont besoin d'aide... Heuresement ils ont réussi à contacter un être d'un autre monde pour organiser ce dernier bastion de civilisations, c'est là que vous intervenez !

Chaque survivant suivra aveuglement vos ordres, ils vous font entièrement confiance pour les sortir de ce petrin !

Comment jouer

Vous aurez des actions à faire faire à tous les survivants de la communauté chaque jours. Il faut veuiller à ce que tout le monde aie de qui manger, boire et un toit sur ça tête où dormir à la fin de chaque journée. Le dérouler du jeu est expliquer dans l'application, normalement ça devrait être clairs je vous laisse découvrir !

Code de l'application



La base du code sont les classes communaute et survivant. C'est la base du jeu.

communaute.hh La déclaration de la classe communauté, cette classe a de nombreux attributs :

- c_nom : (String) du nom de la communauté choisi par le joueur
- JourSurvecus : (int) nombres de jours survécus par la communauté
- NbNourriture (int) nombres de "rations" de nourritures possédés par la communauté
- NbEau (int) nombres de "rations" d'eau possédés par la communauté
- NbMedicament (int) nombres de médicaments possédés par la communauté (sert à soigner un survivant malade)
- NbJardins (int) Pas implémenté mais aurait servi à générer de la nourriture automatiquement chaque jour (à faire construire par un survivant si on a une houe)

— std : :vector<survivant> membres ; vecteur des survivants vivants dans la communauté

— std : :vector<equipement> equipements ; vecteur des equipements possédés par la communauté

communaute.cc Le constructeur initialise tous les attributs avec les valeurs de départ, ensuite c'est dans cette classe qu'il y a toutes les méthodes essentielles au jeu (elles sont décrites dans le codes je passe vite) :

— les getters : sert dans le code pour récupérer les attributs privés

— La surcharge de l'opérateur «

— Les méthodes d'actions : la base du jeu, ce sont les actions permisent au joueur chaque jour

— Les méthodes d'infos : Permet d'afficher les infos sur la commu pour le joueur

— Les méthodes de jeu : permettent le déroulement du jeu (dont finirJour), elles sont appelées dans le main

survivant La classe survivant a les attributs estMalade et estBusy (occupé en anglais). On peut donc créer des objets survivants dans la communauté et savoir a tout moment qui est malade ou occupé.

main Les fonctions dans le main sont assez équivoques. C'est un jeu en format texte sur le terminal, on attendra une réponse correqpondant à une action (comme décrit dans le terminal à chaque fois).

La base du jeu est "solide" (on a beaucoup testé) mais un peu simple pour l'instant, en effet si on crée 2 tentes au début du jeu on peut envoyer nos survivants chercher des vivres chaque jours sans problèmes et le jeu est (presque) impossible à perdre... Bon jeu !

Suite

J'ai malheuresement conscience que ça ne sera pas assez, le gros du code a été fait pendant les vacances de Noël et je rattrapais les tps en mêmes temps... Ensuite j'ai manquer de temps pendant les partiels pour implémenter plein de concepts (problème d'organisation). Par exemple les équipements aurait été ajoutés théoriquement, grosse partie du jeu malheuresement qui n'a pas été implémenté par manque de temps (et aussi par manque d'organisation, on a voulu essayé beaucoup de concepts pour cette partie car on trouvait le code du jeu de base trop simple et malheuresement on n'a pas réussi à les implémenter à temps...). Le concept est simple, si la communauté à un des équipement il peut faire une nouvelle action :

- le sac permet de faire ramener plus de nourriture à un survivant (de type contenant donc incassable)
- la gourde permet de faire ramener plus d'eau à un survivant (de type contenant donc incassable)
- le marteu permet de construire des maisons (c'est un abri comme la tente mais qui peut habriter 5 personnes au lieu de 2), limité à un nombre d'utilisation définie par sa durabilité
- la houe permet de créer des jardins, limité à un nombre d'utilisation définie par sa durabilité
- l'arme permettait de survivre plus facilement à la recherche de vivre (mais au final on ne peut pas mourrir de la recherche de vivres)
- la radio aurait permis de trouver des survivants (non malades) qui rejoindrait la communauté

Quelques unes de ces classes ont été codées mais ne servent pas dans le jeu. Les classes sac et gourdes héritent de contenant (qui aurait dû etre une classe abstraite), les classes outil, arme, houe héritent de outil (qui aurait dû etre une classe abstraite) et enfin contenant et outil héritent de equipement (qui aurait dû etre une classe abstraite aussi).

Procédure d'installation

Ce jeu utilise seulement les bibliothèques de base de C++, il y a un makefile qui marche et des tests.
Il suffit donc de télécharger le dossier et ensuite de taper *make* puis *./main* dans un terminal.

Conclusion

On est surtout fiers que la base du jeu marche, malgré le fait que le jeu est simple tel qu'il est. En effet la bonne structure de la classe communauté nous a pris du temps mais ensuite l'implémentation des différentes actions une fois que la classe marchait était simple et pratique. C'est pourquoi même si on a pas pu implémenter toutes nos idées on sait comment faire et on sait que ça ne "cassera" pas la base du jeu (les seules erreurs possibles seront liés au code). L'interface de jeu aussi nous plait, même si ce n'est que le terminal (l'idée de créer une interface nous a longtemps intéréssé) le déroulé du jeu est ergonomique et visuellement propre.

Merci de votre attention.