UFR Sciences et techniques
Université du Maine

RAPPORT DE PROJET

RogueLike

13 décembre 2016

Emeric Mottier
Valentin Pelloin
Titouan Teyssier

L2 Sciences pour l'ingénieur

Table des matières

Chapitre 1

Introduction

Nous avons choisi le jeu *Roguelike* car c'est un jeu que nous trouvons intéressant, puisqu'il est complet, et que c'est un jeu aux possibilités infinies : il est toujours possible d'ajouter de nouvelles actions que le joueur pourra effectuer.

Notre jeu se déroule dans le bâtiment IC^2. Nous sommes un étudiant, nous partons du rez-de-chaussée, et nous devons aller chercher QUELQUE CHOSE tout en haut, pour le ramener.
Nous devons cependant faire attention aux monstres : des L1, L2, L3, des masters, des doctorants, et certains fantômes : Claude et Chappe.
Sur notre chemin, nous pouvons trouver quelques pièges : des flaques d'eau laissées par les femmes de ménages qui nous font glisser, des trous entre les étages qui nous font tomber d'un étage à un autre inférieur, ou des cartes à jouer qui nous sont jetées dessus par des L1.
Durant notre parcours, nous devons aussi tenir compte de notre faim. Nous possédons une barre de vie, lorsqu'elle est à zéro, nous mourrons. Pour régénérer de la vie, il y a deux possibilités : ne plus avoir faim (en mangeant de la nourriture, attention, certaines sont empoisonnées), et des seringues de soin à s'injecter directement. Ces objets peuvent être consommés directement sur place quand le joueur le trouve, ou plus tard, en les gardant dans son inventaire.
Enfin, lorsque le joueur apparait, il ne voit pas entièrement la carte, il doit la découvrir pour cela. Lorsque le joueur a trop faim, en plus de perdre de la vie, il s'évanouit : il se déplace plus difficilement, et perd connaissance de ce qu'il a découvert.

Pour le projet, nous devions au minimum effectuer un jeu qui génère des niveaux (ici, des étages dans notre bâtiment) aléatoires, avec une taille variant en fonction de l'étage où se trouve le joueur. Il était aussi demandé, en fonction de l'avancement du projet, d'ajouter des fonctionnalités supplémentaires : des armes, des monstres, des pièges, ou autres.

Chapitre 2

Organisation

2.1 Répartition des tâches

Mettre ici notre répartition des tâches.

2.1.1 Pourquoi cette répartition

2.2 Utilisation d'un gestionnaire de versions

Pour gérer les versions du projet nous avons utilisé Git et Github. Git pour toute la partie locale à chacune de nos machine. Quand nous codons, nous pouvons ainsi faire des versions du projet régulièrement et revenir en arrière si besoin. Github pour la mise en commun des modifications apportées, ce qui nous permet de travailler ensemble sans nécessairement coder en même temps ou au même endroit. Afin que chacun puisse librement ajouter ses modifications au projet, un seul dépôt Github a été créé et chaque membre de l'équipe a reçus le droit en écriture sur le dépot.

2.3 Utilisation des projets Github

2.4 Notre boîte à outils

Notre projet a été réalisé en plusieurs modules différents, et l'un d'entre eux est notre boite à outil. Dedans se trouve de nombreuses fonctions qui nous sont utiles, mais qui ne sont pas pour autant liées à notre projet en particulier : des fonctions de comparaison d'intervalles, d'aléatoires, de caractères, de log d'erreurs, de fichiers, ...
Nous avons aussi les fonctions essentielles pour l'accès à des listes et des files.

2.5 Doxygen, CUnit, GDB

Durant la réalisation de notre projet, nous avons utilisé divers outils d'aide à la programmation et au débogage.

La première chose que nous avons mis en place est la documentation à l'aide de *Doxygen*. C'est un programme qui génère une documentation automatiquement, en fonction des fichiers

d'en-têtes et sources. La documentation peut être générée de plusieurs formats, nous avons choisi au format HTML car il est plus facile de s'en servir. Celle-ci est sur internet, à l'adresse suivante : `https://roguelike.vlntn.pw/`. Elle se met à jour automatiquement en fonction de notre code (via un webhook mis en place sur Github).

Ensuite, nous avons utilisé *CUnit*, un framework de tests unitaires pour le C. Toutes nos fonctions de notre boîte à outil ont été testés, avec des assertions que nous jugeons pertinentes (sur des valeurs qui pourraient poser problème dans certaines fonctions, comme des valeurs nulles, négatives, sur des fichiers inexistants, ...).

Enfin, lorsque nous avions certains bogues que nous n'arrivions pas à résoudre, nous avons utilisé le logiciel de debogage *GDB* (*GNU DeBugger*). Nous n'avions pas réussi à le faire fonctionner dès le début, car nous utilisons la libraire d'affichage *ncurses*, qui utilise déjà le terminal pour afficher notre jeu. En le combinant avec *GDB*, le terminal n'était plus utilisable.
La solution à été d'utiliser deux téléscripteurs (TTY) differents : un pour le jeu, et un pour le débogueur.
Dans l'annexe, vous pouvez retrouver 3 exemples de cas où nous nous sommes servis du débogueur.

Chapitre 3

Analyse et Conception

3.1 Notre cahier des charges

3.2 Règles détaillées du jeu

3.3 Comment jouer ?

— Lancement du jeu :
Pour commencer à jouer, vous devez télécharger le jeu à partir de l'adresse suivante : rogueLike ; avec la commande suivante : git clone https ://github.com/TitouanT/rogueLike/ (sur linux) et git clone git@github.com :TitouanT/rogueLike.git (sur Mac).
Vous faîtes : «cd rogueLike».
Puis vous compilez grâce au makefile : «make install».
Le jeu commence dès que vous faîtes : «./rogueLike»

— Les déplacement :
Nous pouvons gérer nos déplacements sur la carte grâce aux flèches de direction.

— Interactions avec des objets :
Les interactions avec un objet (serringues de soins, nourriture, escalier) se font avec la touche «entrée».

— Ouvrir et fermer une porte :
Si vous souhaitez ouvrir une porte, déplacez-vous devant la porte, appuyer sur la touche «o» et marqué la direction de la porte avec les flèches de direction. Mais pour fermer, c'est le même principe que pour ouvrir une porte sauf que la touche est «c» au lieu de «o».

— Gestion de l'inventaire :
Pour voir votre inventaire, vous devez appuyer sur la touche «i». Pour prendre un objet (serringues de soins, nourriture), vous devez appuyez sur la touche «g» mais pour poser votre inventaire, vous devez appuyez sur la touche «d» et indiquer la case se trouve l'objet.

— Sauvegarder sa partie :
Vous pouvez sauvegarder la partie à tout moment avec la touche «s», cette manoeuvre n'arrêtera pas votre expérience de jeu.

Chapitre 4

Codage, méthode et outil

4.1 Structures et énumérations

Nous avons utilisé un certain nombre de structures et d'énumération pour modéliser le rogue-Like.

4.1.1 Une case de la carte

La structure la plus importante est celle qui représente une case de la carte. Cette structure contient un champ qui définit sont type, un autre qui défini son état ainsi qu'un tableau listant les objet présent sur la case et deux entiers pour savoir si le joueur a découvert cette case et pour connaitre le nombre d'objets présent.

Le type est une énumération des différents type de case possible : un vide, un mur, un encadrement de porte, une piece ou un couloir.

L'état est une énumération utilisé pour les encadrements de portes afin de savoir si il y a une porte, et si elle est ouverte ou fermée. L'état est aussi utilisé pour les pieces afin de savoir si il y a de la lumière ou non dans la pièce.

Un objet est décrit par une structure qui est composé d'un type et d'un entier pour savoir si l'objet a été découvert par le joueur. Le type d'un objet est lui décrit par une énumération et définit les escaliers ascendants et descendants, la nourriture, les kits de santé et les pièges.

4.1.2 Les être vivants

Le joueur controle un personnage que l'on représente à l'aide d'une structure. Elle permet d'enregistrer son nom, sa position (ligne, colonne et niveau), ses points de vie et d'attaque, son agilité, son expérience, son appétit, sa santé, le nombre de mouvement effectué, un booléen pour savoir si il a trouvé le QUELQUE CHOSE ainsi qu'in tableau d'objet qui représente son inventaire.

Les monstres ont eux aussi une structure qui enregistre leur nom, leur types, leur position (ligne, colonne et niveau), leur points de vie et d'attaque, leur champ de vision, leur agilité et quelque champs supplémentaire qui est utilisé comme mémoire pour leur IA. Les différents types de monstres sont L1, L2, L3, master, doctorant et fantome.

4.1.3 Et le reste

La représentation en matrice des niveau n'étant pas dans tous les cas la plus efficace, nous avons créé une structure de niveau qui contient un tableau de pièces et le nombre de pieces que contient l'étage.

Une pieces est elle-même représentée par une structure qui contient la position de son angle supérieur gauche (ligne et colonne) ainsi que ses dimensions (largeur et hauteur).

Quelque fois, nous avons besoin de manipuler des position (ligne, colonne). Nous avons donc créé une structure de position qui contient une ligne et une colonne.

4.2 Séparation du code en modules

bla

4.3 Détail des modules

4.3.1 La génération des niveaux

bla

4.3.2 La sauvegarde

4.3.3 Les changements d'étages

4.3.4 Les interactions et déplacements

4.3.5 L'affichage

4.3.6 Les monstres

bla

4.3.7 La nourriture et la vie

4.3.8 Les pièges

Chapitre 5

Résultat et conclusion

Faire une comparaison avec les objectifs.

5.1 Améliorations possibles

5.2 Apport personnel du projet

Chapitre 6

Annexe