Le Mans Université

Institut d'Informatique Claude Chappe
Le Mans Université

Le Mans Université

Licence informatique 2e année

Module 174UP02 – Rapport de projet

Dig & Rush

Matthieu Boulanger
Ania Garoui
Yohan Harison
Jacques-Gérard Mpondo Toutou

15 avril 2024

Lien vers le dépôt du projet

Table des matières

1 Introduction

Les objectifs de ce projet en groupe sont de mettre en pratique nos connaissances acquises durant cette deuxième année de licence informatique, notamment en algorithmique en code C. Ce projet permet également de nous introduire à la gestion de projet qui nous sera utile dans le monde professionnel. Enfin, il permet de commencer à agir en groupe et en autonomie, deux points qui nous seront également utiles en tant que professionnels.

Le principe du jeu Dig & Rush consiste à l'instar du jeu Once upon a tower, jeu mobile, de descendre dans une tour en creusant des blocs qui font office de murs et de sols comme dans un labyrinthe. Le joueur a pour outil une pelle qui lui permet de détruire les blocs. Il faut en plus de cela éviter ou tuer les ennemis afin de progresser le plus rapidement dans la tour. Si le personnage meurt la partie se termine. Le joueur a le choix entre plusieurs personnages qui ont chacun une particularité. Attention, chaque tour est différente ! Cela vous permet d'explorer à chaque nouvelle partie une nouvelle facette du jeu.

Ce rapport se déroule en quatre parties. L'organisation du projet, comment les tâches ont été définies et attribuées ; la conception, qui présentera nos choix de conception et les règles du jeu ; le développement avec les outils utilisés, les sturctures de données et les algorithmes et enfin la conclusion qui mettra en avant nos remarques, nos points d'amélioration et ce que nous avons tiré de ce projet. En annexe, se trouvera les tests unitaires et tentatives de debogage, les captures d'écran du jeu et les diagrammes de Gantt.

2 Organisation

2.1 Définition des tâches

Avant de commencer à coder l'équipe a effectué un brainstorming afin de connaître toutes les phases du projet et les tâches qui seraient susceptibles d'entrer dans ces dernières. En effet, en citant exhaustivement les tâches cela nous permet d'avoir une estimation temporelle du projet. Nous avons également pris en compte les attentes du corps enseignant.

Nous avons détecté 7 phases principales sur le projet général : la définition du projet, la planification, la conception et l'analyse, la phase de réalisation, la phase de documentation, le rapport du projet et la soutenance qui font parti de la phase de livraison.

La définition du projet est l'étape post conception elle permet de faire une esquisse du projet et ainsi se le représenter plus facilement. On y détermine les limites du projet. La conception entre dans l'architecture du projet. La planification, qui sera évoquée dans la partie suivante, sert à se répartir les tâches. La phase de réalisation est le codage ainsi que les tests. La documentation est la création de documents permettant la compréhension du projet. Le rapport et la soutenance sont réservés à la présentation du projet.

2.2 Répartition des tâches

Suite à l'élaboration de la liste des tâches nous nous sommes réparties les tâches en fonction de deux critères. Tout d'abord les choix et affinités de chacuns, en effet chaque personne du groupe a préféré certaines tâches plutôt que d'autres. Ensuite, nous nous sommes réparties le reste des tâches en équilibrant afin que tout le monde ait à peu près le même temps de travail.

Pour rendre visuel ces deux étapes précédentes l'équipe utilise l'outil GANTT. Il s'agit d'un graphe qui représente les tâches étalées sur le temps. Google Sheet est utilisé afin de pouvoir le mettre à jour et collaborer à distance. Sur le graphe sont présentes les tâches, les dates encadrantes ces dernières et une représentation graphique de la durée de la date entre la semaine du 8 janvier 2024 et celle du 22 avril 2024. L'équipe adopte deux diagrammes de Gantt, un diagramme prévisionnel qui représente l'avancement idéal du projet et un diagramme réel qui est mis à jour régulièrement. Ce dernier sert principalement à voir les tâches restantes en fonction de la deadline.

2.3 Outils consacrés à la communication et à l'organisation

Nous utilisons Discord pour communiquer. Il permet de créer des canaux en fonction de nos besoins. Dans le cas de projet il y a un canal pour les ressources, un pour la conversation générale et des canaux vocaux si nous avons besoin de nous appeler. Nous utilisons principalement le canal général pour suivre l'avancement de chacun.

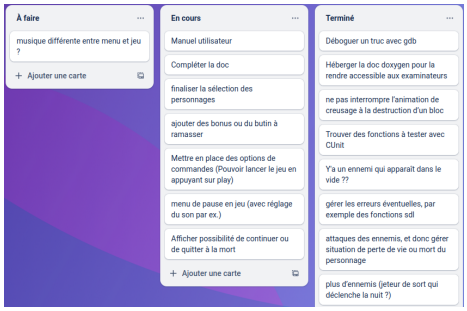

FIGURE 1 – *Mettre capture Discord*

Sur le mois d'avril un Trello est mis en place, il a permis de voir plus facilement les tâches restantes et de les prioriser. Dans ce Trello est présent huit listes pour : les ressources, les idées, ce qu'il faut faire mais qui n'est pas urgent, ce qu'il faut faire, les tâches en cours, celles terminées, les problèmes et enfin ce qu'il ne faut pas oublier. Contrairement au diagramme de Gantt, il se concentre principalement sur les tâches liées au codage.

FIGURE 2 – *Une partie du Trello*

Afin de gérer les différentes versions du code sources, Git est utilisé. Il permet d'avoir un espace de stockage commun que ce soit pour la documentation que le code. De plus, cela permet à chacun de travailler sur une partie du projet sans modifier ce que font les autres membres du groupe, notamment grâce à la création de branches qui permet la création un clone du code principal et pouvoir travailler dessus. A peu près une branche a été créée pour chaque fonction principale du jeu. Une fois le code respectif fonctionnel nous le fusionnons dans le code principal.

En plus du diagramme de Gantt et du Trello l'équipe utilise l'outil "Projects" dans Git. Il sert, dans le cas de ce projet, à communiquer notre avancement au corps enseignant. Il permet également, pour chaque semaine, de mettre la tâche effectuée par chaque membre de l'équipe.

3 Conception

Développé en C en utilisant la bibliothèque SDL, notre jeu a d'abord connu une phase de conception.

3.1 Objectif du jeu

3.2 Règles de jeu

3.3 Graphismes

Les graphismes de Dig&Rush sont fortement inspirés du jeu mobile Once Upon a Tower. Pour les menus (menu principal, menu de personnages, menu de paramètres, ...), les couleurs chaudes telles que le rouge et le orange, mixé à une pointe de violet nous semblaient plus adaptés à l'esprit combatif et dangereux du jeu. En ce qui concerne l'interface du jeu, nous avons opté pour une palette de couleurs vivantes dans les tons bleu, vert et blanc pour attirer et retenir l'attention des joueurs. Une fois les couleurs définies, il nous restait à trouver les bonnes ressources en libre de droit. Nous avons donc fini par opter pour des images d'intérieur médiéval pour les menus, un fond de ciel bleu pour l'interface de jeu et un fond uni blanc pour le fond de la tour. La tour elle à son tour, est entourée de murs de blocs de pierres.

3.4 Niveaux

Les niveaux, avec le placement des blocs qui devait être aléatoire, était l'une des parties les plus complexes à concevoir, car il nous fallait pour chaque ligne une suite de blocs de pierres, de blocs de terre et de vides générés automatiquement

FIGURE 3 – *Menu principal*

mais tout en faisant attention à toujours laisser au moins une issue possible au joueur afin qu'il puisse avancer.

3.5 Personnages

3.6 Mécaniques

Comme mentionné précédemment, notre jeu est composé d'une tour, de blocs de pierres, de terre, de personnages (joueurs et non joueurs) et le petit plus : un effet de transition entre le jour et la nuit.

- Les blocs de pierres délimitent la tour et les endroits par où le joueur ne peut pas passer.
- Les blocs de terre sont destructibles et permettent donc de creuser un chemin pour le joueur, mais non pour le PNJ.
- Les personnages joueur creusent et attaquent les ennemis pour les vaincre.
- Les personnages non-joueur (ennemis) se déplacent sur un axe horizontal et attaquent le joueur dès qu'il est détécté à leur proximité.
- Effet d'ombre qui permet la transition entre le jour et la nuit pour varier la difficulté et rajouter du défi.

Chacune de ces mécaniques est agrémenté d'un son pour soit un joueur masculin ou féminin (son de blessure, d'attaque, de mort, ...).

3.7 Analyse des difficultés

4 Développement

Dans cette section, nous traitons des aspects techniques du développement de notre jeu en 2D utilisant la bibliothèque SDL et programmé en langage C. Elle englobe plusieurs sujets, notamment les librairies employées, l'architecture du code, la création des personnages...

4.1 Librairies et outils utilisés

Pour le développement de notre jeu, nous avons principalement opté pour l'utilisation de la bibliothèque SDL (Simple DirectMedia Layer) en langage C. SDL s'est révélée être un choix judicieux, offrant une interface simple et efficace pour la gestion des graphiques, du son et des périphériques d'entrée. Cette bibliothèque a grandement facilité le processus de développement de notre jeu en 2D. En complément, nous avons intégré GitHub comme plateforme de gestion de version, nous permettant ainsi de collaborer efficacement sur le code source et de suivre son évolution au fil du temps. De plus, nous avons utilisé des outils de développement standard tels que GCC (GNU Compiler Collection) pour la compilation du code source et GDB (GNU Debugger) pour le débogage.

4.2 Création des personnages

4.3 Architecture du code

`main.c`	point d'entrée du programme
`ressources.c`	fonctions et structures de chargement des fichiers de ressources
`tour.c`	cœur du jeu avec notamment la boucle principale
`entite.c`	structure utilisée pour tout objet devant être affiché, déplacé et animé
`entite_obstacle.c`	spécialisation des entités qui agissent comme obstacles
`entite_destructible.c`	spécialisation des entités qui peuvent être détruites
`entite_pnj.c`	spécialisation des entités représentant un personnage non joueur
`entite_perso.c`	spécialisation des entités représentant le personnage du joueur
`texte.c`	API[A] pour simplifier la gestion des textes
`nuit.c`	structure et fonctions implémentant une transition jour/nuit

Table 1 – Rôle des différents modules

5 Conclusion

La conclusion de ce projet met en lumière les apprentissages, les défis surmontés et les perspectives pour l'avenir. Nous avons réussi à concrétiser notre vision du jeu Dig & Rush grâce à une collaboration efficace et à une répartition équilibrée des tâches. En utilisant des outils tels que Discord, Trello et Git, nous avons pu organiser notre travail de manière transparente et coordonnée, facilitant ainsi la communication et la gestion des versions du code.

A Lexique

PNJ	Personnage Non Joueur
API	*Application Programming Interface*, ensemble de classes et fonctions servant d'interface vers un service

TABLE 2 – Définitions des termes techniques employés dans le document

B Tests unitaires

Les tests unitaires sont essentiels pour assurer la qualité et la fiabilité. Parmis les différentes fonctions implémentées, il nous a semblé important de vérifier le bon fonctionnement grâce à une suite de tests disponibles sur la librairie CUnit.
Fonctions testées : init_ressources, recuperer_texture, recuperer_spritesheet, recuperer_son, recuperer_musique, recuperer_audio, recuperer_police, jouer_audio et detruire_ressources.
Voici le résultat des tests disponibles dans le fichier test_unit.c

```
-------------------------   Lancement des tests...   -------------------------

bin/test_unit


     CUnit - A unit testing framework for C - Version 2.1-3
     http://cunit.sourceforge.net/


Suite: Suite_Test_Ressources
  Test: test du chargement des ressources ...passed
  Test: test de destruction des ressources ...passed

Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      1      1    n/a      0        0
               tests      2      2      2      0        0
             asserts      5      5      5      0      n/a

Elapsed time =    1.935 seconds
```

FIGURE 4 – *Résultats des tests unitaires*

Les résultats des cinq assertions sont logiques et indiquent que les fonctions se comportent comme prévu, ce qui signifie que les ressources sont fiables.
Nous avons jugé important de mettre en place une suite de tests unitaires efficace

qui présente une bonne base pour ajouter plus de tests au fur et à mesure que le projet s'agrandit. Cela aide au maintient de la qualité du code tout au long du développement.

C Exemple de débogage

Lors du développement du jeu, plus précisément de la fonction de génération d'ennemis, un bug revenait souvent : un ennemi était généré dans le vide. Pour régler ce problème, il a donc fallu déboger.
La fonction en question (generer_ennemi(x,y)) est déclarée et appelée dans le fichier source src/morceaux_niveau.c.
Etapes de débogage :

FIGURE 5 – *Bug du squelette*

- Lancer GDB
- Poser un point d'arrêt (*(gdb) break src/morceaux_niveau.c :generer_ennemi*).
- Appuyer sur *PLAY*.
- On avance dans le jeu jusqu'à rencontrer le point d'arrêt.
- On execute *step* pour rentrer dans l'appel de fonction.
- A l'aide des options de débogages telles que textitprint, display, next, ..., nous avons constaté que l'algorithme de base n'était pas bon.

Note : les instruction de débogage sont renseignées dans le manuel d'installation et d'utilisation fourni.

,õ,