Le Mans Université

Institut d'Informatique Claude Chappe Le Mans Université

Le Mans Université

Licence Informatique 2^e *année*

Module 174UP02 Rapport de Projet

Tomogashi Island

Alban Vallee, Elona Pierre, Kilian Pousse

6 avril 2024

Résumé

Dans le cadre de nos études, nous avons réalisé un jeu vidéo qui nous a permis de mettre en avant tout ce que nous avions appris depuis le début de notre licence en informatique.

Lien du projet : https://github.com/KilianPousse/Tomogashi_Island/

Table des matières

1 Introduction

1.1 Principe du jeu

Notre projet porte sur un jeu à l'aspect mignon avec de petits personnages représentant des animaux, le tout dans un univers coloré. Le joueur incarne un lapin nommé Haricot qui arrive sur une île sans savoir pourquoi ni comment il a atterri ici. Il y fera la rencontre de plusieurs personnages qui l'aideront à répondre à ses questions.

Le jeu que nous avons imaginé est inspiré de plusieurs jeux comme Stardew Valley, Tomagochi, Animal Crossing et Harvest Moon.

1.2 Règles

Le joueur doit effectuer différentes tâches afin de découvrir comment il a atterri ici. Il le découvre au fur et à mesure grâce à des livres disséminés partout sur la carte. Le joueur peut également développer son économie ce qui lui permettra d'acheter des cosmétiques et différents items au cours de la partie. Il y a plusieurs moyens de récolter de l'argent que ce soit avec la pêche ou l'agriculture puisque le joueur pourra cultiver des plantations ou pécher du poisson pour obtenir des collectibles revendables sur la place du marché.

1.3 Fonctionnalités prévues

Nous avions prévu une liste de fonctionnalités différentes que voici :

1. Faire pousser et récolter des cultures
2. La pousse des cultures en fonction du temps
3. Pêcher
4. Acheter et vendre des objets
5. Une carte dynamique contenant des éléments animés
6. Des musiques pouvant changer en fonction d'où se situe le joueur sur la carte
7. Des PNJ avec qui interagir
8. Un menu
9. La possibilité de sauvegarder la partie
10. Une page de paramètre
11. Une barre de raccourcis d'items
12. Un inventaire
13. Différentes quêtes
14. Un système jour et nuit en fonction du temps qui passe
15. Des crédits dans les menus
16. Un tutoriel pour chaque interactions possibles
17. Des coffres de rangement
18. Une boutique de cosmétique
19. Acheter et équiper des cosmétiques
20. Créer un filtre pluie et faire varier la météo
21. Une barre d'énergie qui descend plus vite si le joueur fait des actions
22. Téléportation au lit du joueur lorsque qu'il est trop tard

Nous n'avons malheureusement pas pu implémenter les fonctionnalités 17 à 22 pour des raisons multiples. Pour les fonctionnalités 18 et 19 qui concernent les cosmétiques, il y avait un manque crucial de temps pour les réaliser puisque nous aurions dû refaire les sprites du personnage et toutes ses animations pour faire bouger les cosmétiques avec ses mouvements. En gérer la boutique rajoutée d'autant plus de temps nécessaire. D'autres fonctionnalités étaient prévues dans un second plan comme la 20 ou la 21 ce qui explique pourquoi elles ne sont pas encore disponibles dans le jeu.

2 Organisation

L'organisation d'un projet a une importance primordiale puisqu'elle définit dès le départ le bon déroulement du projet. Elle se base sur les compétences et l'expérience de chacun pour pouvoir avancer dans de bonnes conditions. C'est en s'aidant des cours de *Conduite de projets* que l'on a pu planifier le projet en respectant les différentes phases d'un cycle de vie d'un projet à l'aide d'outils comme un *Diagramme de Gantt*.

2.1 Diagramme de Gantt

Au commencement du projet, nous nous sommes rassemblés pour établir un diagramme de Gantt précis avec toutes les fonctionnalités prévues(voir annexe 22).

Cependant, pour une meilleure compréhension générale, nous avons établi un diagramme de Gantt simplifié énonçant seulement les tâches principales.

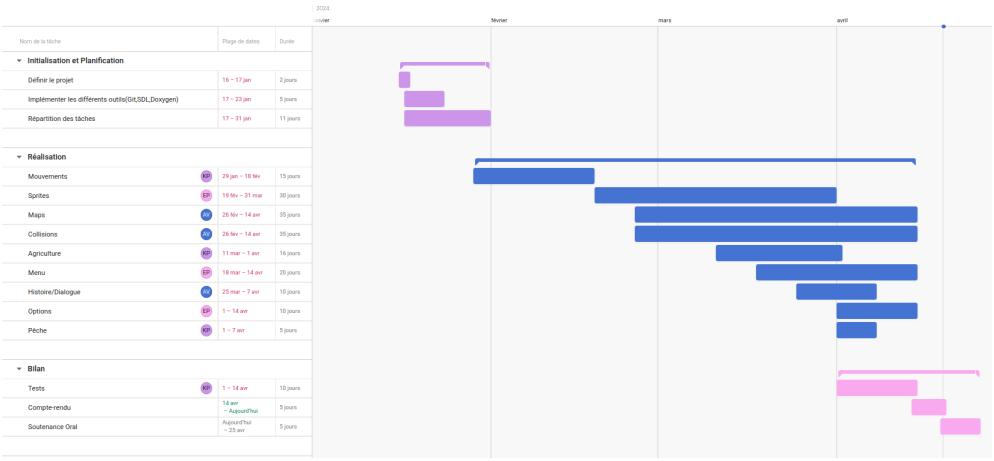

FIGURE 1 – Diagramme de Gantt simplifié

2.2 Répartition des tâches

Nous nous sommes alors repartit les taches dès le départ puis tout au long du projet pour toujours avancer de manières égales.

Tâches	Alban	Elona	Kilian
Diagramme de Gantt			
Implémentation de Doxygène			
Sprites des Personnages			
Cartes / Collisions			
Mouvements du personnage			
Interface du Menu			
Histoire			
Système horaire			
Activité : Agriculture			
Activité : Pêche			
Activité : Marché			
Musiques			
Barre des tâches			
Gestion des items/inventaires			
Monnaie virtuelle			
Gestion des dialogues			
Options			
Filtre jour/nuit			
Tests			
Oral de soutenance			
Rédaction du LaTeX			

FIGURE 2 – Tableau présentant la répartition des tâches.

2.3 Respect des deadlines

Il a fallu repenser plusieurs fois notre organisation durant le projet afin de pouvoir arriver à la finalité de notre projet. Malheureusement, nous n'avons pas pu développer certaines fonctionnalités à cause du manque de temps. Nous avions prévu cette éventualité et avons donc réalisé le projet de manière à implémenter les objectifs les plus importants en priorité de sorte à ce que le projet puisse être fonctionnel tout en ayant une physionomie de jeu intéressante.

2.4 Division des tâches

Les taches du projet ont été divisées en plusieurs blocs pour prévoir le manque de temps possible pour l'implémentation des dernières fonctionnalités du jeu. Nous avons commencé par fabriquer la base du jeu avec les cartes, les sprites puis nous avons découpé chaque fonctionnalité en partie de code : le menu, les options, l'agriculture, la pêche, le marché, l'histoire, etc.

3 Conception

Nous avons décidé d'organiser nos fonctionnalités en différents blocs afin de pouvoir les développer séparément sans se marcher dessus. Cette méthode nous a permis d'avancer étape par étape sur les taches, de la plus importante à la plus optionnelle. Ce modèle de travail nous a octroyé la possibilité de rendre un jeu avec les fonctionnalités principales, comme les collisions, l'agriculture, etc. Le tout dans un temps définit. Nous avons pu également en ajouter certaines moins primaires comme le filtre nuit ou bien les dialogues avec les PNJs.

3.1 Conception du projet

Notre espace de travail est structuré comme vu durant le coura de conduite de projet. Les fichiers sont regroupés selon leur extensions et leur utilités. Par exemple, les fichiers **.h* se retrouvent dans le dossier *include*. Cette arborescence permet une bonne organisation de notre travail. Chaque section, chaque fonctionnalité se trouve dans un fichier adéquat. Si nous prenons l'exemple des fonctions de gain et de suppression d'objets. Elles consistent à modifier le contenu de ce dernier. On a donc décidé de mettre les fonctions *inventary_give* et *inventary_remove* dans le fichier *src/inventary.c*.

- assets
- bin
- include
  - header.h *(Entête principal pour tous les scriptes)*
  - item.h *(Liste des items)*
  - style.h *(Lien vers les images dans les dossiers)*
  - ...
- src
  - main.c *(Programme principal*
  - player.c *(Gestion, affichage du joueur)*
  - display.c *(Affichage des arrières plans, calques, ...)*
  - map.c *(Gestions des cartes)*
  - farm.c *(Gestion de l'agriculture)*
  - ...
- test

FIGURE 3 – Arborescence

3.2 Les sprites

Tout nos sprites animés ont été réalisé par nous-mêmes :
— Les PNJ
— Les mouvements de l'eau
— Les objets décoratifs (le bateau, les arbres en mouvement,...)
— Le filtre pluie
— L'image du menu
— Le logo de notre jeu
— Le logo de la fenêtre

3.2.1 Les personnages

Pour réaliser ce genre de sprites de personnages il faut mettre bout à bout plusieurs images et les faire coïncider ensemble.

Tout nos sprites de personnages ont été créé sur la même base, c'est-à-dire quatre images par animation, le personnage principal est le seul à avoir des animations pour toutes les directions, les autres sont animés pour rester sur place et bouger de manière à les rendre plus vivant.

FIGURE 4 – Exemple de sprite d'un pnj volant

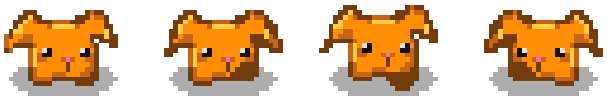

FIGURE 5 – Exemple de sprite de notre personnage principal

Les autres animations de notre personnage principal et PNJ sont retrouvable a la figure 24 de l'annexe.

3.2.2 Les overlays

Nous avons créé un système d'overlay afin de rendre la carte plus dynamique, cette couche de la carte affichera l'eau animée et les items de décorations animés comme le bateau.

FIGURE 6 – Exemple d'overlay

Ce système permet de faire passer le personnage derrière ces éléments de décors. Il faut donc décomposer les cartes en quatre couches : les animations de l'eau (disponible en annexe sur 14 frames), les textures du sol et les ombres, les PNJ et enfin le personnage principal.

FIGURE 7 – Exemple d'overlay d'une carte

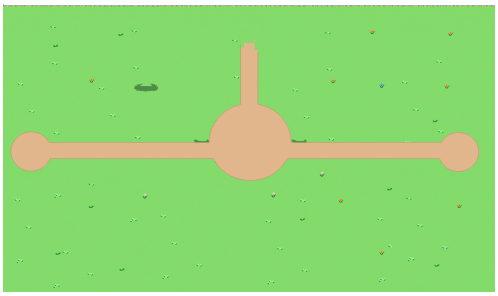

FIGURE 8 – Exemple de texture de sol

3.2.3 Les logo et image de fond

Enfin, pour donner une identité visuelle à notre jeu nous avons crée nous-même le logo du jeu, l'icône de la fenêtre et les images de fond du menu. Nous avons également animé ces images à certains endroits dans notre jeu comme pour le logo dans les menus afin de rendre le jeu plus dynamique ainsi que certains éléments sur l'image de fond. Nous avons également réutilisé les personnages que nous avons créés et certains éléments reconnaissables qui définissent notre jeu à différents endroits comme dans le menu ou les options.

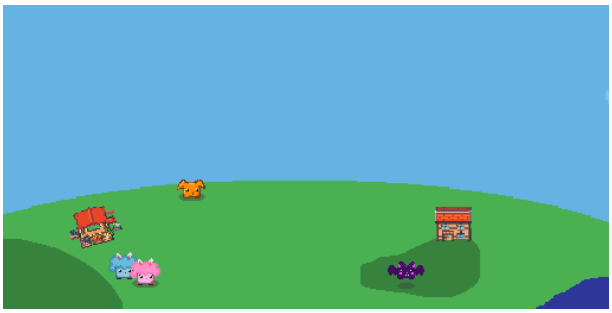

FIGURE 9 – Fond d'écran du menu

FIGURE 10 – Logo du jeu

4 Développement

Le développement est la partie la plus conséquente de notre jeu projet. Ici, chaque fonctionnalité y sera détaillée. Chaque outil utilisé y sera aussi retranscrit afin de réaliser un développement global de cette partie.

4.1 Logiciel et outils utilisés

On a utilisé la seconde version de la librairie graphique Simple DirectMedia Layer (SDL) afin d'établir une interface graphique agréable et ergonomique au joueur. SDL apporte une aide considérable à la programmation d'un jeu utilisant le langage C. Il permet d'afficher des images, jouer des musiques, détecter des événements comme le déclenchement d'une touche.

On a également utilisé Visual Studio Code comme outil de développement. Ce dernier nous a permis de programmer avec beaucoup d'aide et de raccourcis.

4.2 Détails des fonctionnalités

Comme dit précédemment dans la partie *Conception du projet*, on a décidé de développer notre projet en bloque. Nous allons voir la méthode utilisée pour réaliser ces fonctionnalités.

4.2.1 La carte

La carte de Tomogashi Island utile un fonctionnement particulier que l'on peut retrouver dans différents jeux rétros. Le premier opus d'*Animal Crossing*, *Zelda 1* sont des jeux qui utilisent ce principe dans la présentation de leur carte. En effet, celle de Tomogashi Island est séparée en neuf parties distinctes qui ont une fonctionnalité toute différente. Nous pouvons y retrouver une zone d'agriculture, de pêche, de commerce, de plage, d'habitations. Ce choix s'explique par une volonté de rappeler l'époque de ces jeux pour apporter une touche de nostalgie.

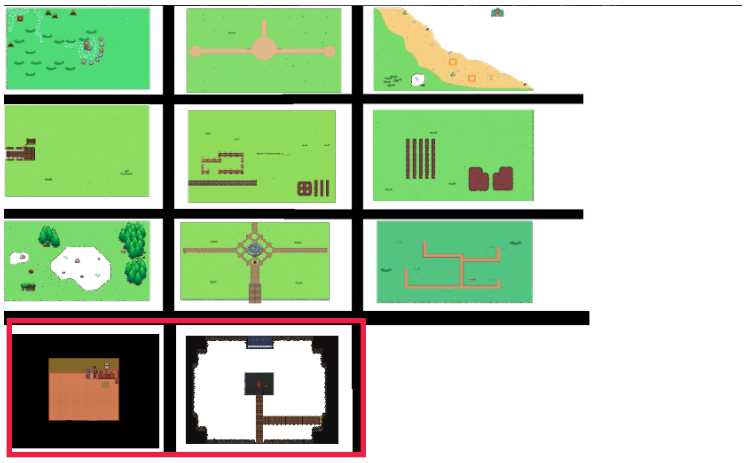

FIGURE 11 – Exemple de collision

Nous avons donc organisé les différentes parties des cartes dans un tableau se situant dans une structure *map_ t*. Où pour chaque partie de cartes, on a une texture d'arrière-plan et de calque.

4.2.2 Les collisions

Chaque parcelle de la carte est découpée en un damier de tuiles. Chaque tuile possède une valeur qui représente une interaction avec le joueur. Si la valeur est négative, alors il ne pourra pas y aller. Cependant, la valeur peut indiquer qu'une action est possible par le joueur. Par exemple, les tuiles numérotées 33 signifient qu'à cet emplacement, il est possible de planter ou de récolter des ressources. Cette case est appelée « terre cultivable ».

La structure *map_t* contient les cartes à afficher ainsi que les valeurs des tuiles. Alors que *player_t* contient des informations sur le joueur comme sa position sur la carte. Cela permet de calculer la position des angles du spirite du personnage principal. Si le joueur souhait avancer alors qu'une des valeurs d'un des coins est négative, alors il y a collision et le joueur ne pourra pas avancer dans cette direction. (*Figure 12*)

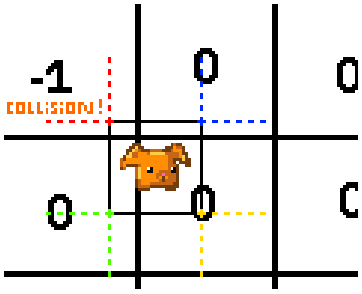

FIGURE 12 – Exemple de collision

4.2.3 L'agriculture

Tomogashi Island a une économie basée sur la collecte de ressources agricoles. Le joueur doit acheter des graines pour les planter et les récolter afin de vendre le résultat.

Afin de gérer la croissance des plantations, on a utilisé une mécanique de cycle. Le jeu est réparti en journée de vingt-quarte heures. Chaque heure dure en réalité une minute. Cela signifie que si vous jouez durant vingt-quatre minutes, vous passez une journée dans le jeu. Afin de diminuer le temps d'attente entre la plantation et la récolte, on a décidé de sous-diviser une journée en quatre cycles de six heures. À la fin d'un cycle, toutes les plantations passent au stade de croissance supérieure. Quand une plante atteint son âge maximal, elle est enfin récoltable.

Un changement stade de croissance se remarque par un changement d'apparence. Par exemple, la tomate pousse en quatre stades qui se symbolise par une pousse qui grandit pour donner un pied de tomate.

FIGURE 13 – Exemple de pousse de blé

FIGURE 14 – Exemple de pousse de tomate

4.2.4 La pêche

La pêche est un autre moyen pour de gagner de l'argent. La zone pêche se situe à gauche de la maison du joueur. Si le joueur se positionne au bout du ponton, il lui sera proposé de pouvoir pêcher. En appuyant sur une touche d'interaction, « espace » ou « e », le jeu se mettra en phase de pêche.

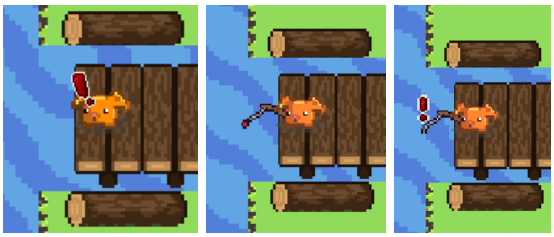

FIGURE 15 – Caption

Après l'animation du personnage, un délai est choisi aléatoirement pouvant vaciller de trois à quinze secondes. Ce délai correspond au temps à attendre entre la fin du lancer et le plongeon du bouchon. Quand est submergé, le joueur à une seconde pour appuyer sur une touche d'interaction afin de remonter la ligne et de récupérer le poisson. Si le joueur n'appuie pas, alors un prochain délai sera calculer. Si le joueur réagis au bon moment, alors un poisson lui sera donné aléatoirement parmi cinq poissons plus ou moins rares.

Race de Poisson	Taux d'obtention
Poisson	40%
Thon	30%
Poisson Lanterne	20%
Carpe Koï	7%
Requin	3%

FIGURE 16 – Exemple de collision

4.2.5 L'inventaire

Il y a d'abord une barre de tâche toujours visible en jeu par l'utilisateur qui sert de mini inventaire avec quelques places disponible pour que le joueur puisse changer rapidement les objets qu'il tient en main.

FIGURE 17 – Barre des tâches

L'inventaire est une fonctionnalité très importante dans Tomogashi Island. Cela permet de stocker des éléments appelés « *item* ». Cet objet possède sa propre structure. Elle est composée de deux entiers : un pour connaître l'identifiant et l'autre le nombre d'un item. L'inventaire a également une structure constituée de deux tableaux d'*items*, l'inventaire principal et la barre des tâches, ainsi qu'une variable indiquant l'objet sélectionne par le joueur.

La barre des tâches permet d'accéder plus rapidement aux items afin que le joueur puisse les utiliser en les sélectionnant.

Le joueur peut toujours avoir accès au reste de l'inventaire en appuyant sur la touche « I ». Il pourra échanger les objets de place comme il le souhaite, afin d'utiliser celui de son choix.

L'inventaire utilise des fonctions lui permettant l'ajout et la suppression d'items avec respectivement *inventory_ give(item_ t * item)* et *inventory_ remove(item_ t * item)*.

Le fonctionnement global et la représentation de l'inventaire sont une inspiration du jeu vidéo *Minecraft*.

FIGURE 18 – Inventaire

4.2.6 Achat et vente d'objets

L'économie est l'un des objectifs les plus important que propose le jeu. L'argent permet d'acheter des objets pouvant être utilisés dans l'agriculture par exemple. L'achat et la vente d'objets se font auprès des villageois se situant sur la place du marché de la zone commerçante. À l'aide d'une interface, le joueur peut choisir la quantité d'objets qui souhaite vendre ou bien acheter (*Figure25*).

Il existe différents types de marchand. Il y a le vendeur de graine, l'acheteur de plants, l'acheteur de poisson. À l'origine, nous avions pour objectif de proposer une boutique de cosmétique qui servirait au joueur pour dépenser son argent virtuel dans des accessoires qui seront portés par leur personnage.

4.3 Logiciel

Durant le développement d'un projet, il est normal d'utiliser différents logiciels, lors de ce projet nous avons utilisé *Pixilart* ce logiciel nous a permis de dessiner les cartes et de les séparer en différentes couches.

FIGURE 19 – Logo pixilart

L'utilisation de *Piskel* nous a permis d'animer les sprites du personnage principal et des PNJ.

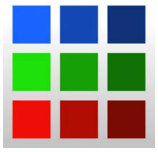

FIGURE 20 – Barre des tâches

Le site internet *Itch.io* a permis de gagner du temps sur certains éléments graphiques comme les objets de décorations type arbre, fleurs, ou encore champs et collectible (graine, légumes,etc.).

FIGURE 21 – Barre des tâches

4.4 Les bibliothèques utilisée

On a utilise la seconde version de la librairie graphique Simple DirectMedia Layer (SDL) comme énoncée plus haut pour l'interface graphique.

5 Tests et débogage

5.1 Les tests

Nous avons réalisé tout un ensemble de test qui se trouvent sur notre github dans le dossier */test*. (*Figure26*)

Chaque fichier source de chaque test se trouve dans le chemin suivant : *./test/src*. Nous pouvons y retrouver le crypte des tests sur les fonctions *outWindow*, *outMap* et *LoadSave*.

— outWindow(int x, int y) : *Figure27*
Cette fonction est une fonction utilisé régulièrement dans notre jeu. En effet, elle est utilisé pour tester si à chaque déplacement, le joueur sort de l'écran et dans quelle direction.

Si l'écran fait : 1000x1000

valeurs d'entrées (x,y)	valeurs attendues	valeurs reçues
-100, 100	1	1
100, -100	2	2
1001, 100	3	3
100, 1001	4	4
100, 100	0	0

— outMap(map_t m) : *Figure28*
La fonction outMap est très utile pour savoir si le joueur est hors de la carte. Cela est utile pour gérer les collions.

Si la map fait : 3x3

valeurs d'entrées map avec (x,y)	valeurs attendues	valeurs reçues
-1, 1	1	1
1, -1	2	2
4, 1	3	3
1, 4	4	4
1, 1	0	0

— LoadSave( ) : *Figure29*
Cette fonction permet de charger une sauvegarde enregistrée dans un fichier.

5.2 Débogages

Nous avons utilisé l'outil *GDB* pour déboger notre programme. Par exemple, nous avons eu un problème à la sortie de grotte. En effet, le jeu nous faisait sortir sur la mauvaise partie de la carte. Nous avons donc utilisé *watch map.x* et *watch map.y* pour regarder l'évolution des variables correspondant à la position du joueur. Nous avons également utilisé un *breakpoint* avec *break action* pour bloquer l'exécution à la fonction *action(...)* (*Figure30*, *Figure31*).

6 Conclusion et résultats

6.1 Les difficultés rencontrées

Ce projet aussi passionnant que chronophage nous a permis de gagner de l'expérience que ce soit en termes d'organisation, de communication, et d'apprentissage (SDL, Git, Doxygene, CUnit). Il nous a montré à quel point le partage des tâches est important dans un travail de groupe plus long que ceux que l'on fait habituellement. Pour notre jeu, la majorité des implémentations voulues ont été ajoutées, pour celles qui ne le sont pas, il n'y avait pas de difficultés particulières rencontrées, il s'agissait surtout d'un manque de temps. Ce fut assez prévisible dès le départ, c'est pour cela que nous avons émis la possibilité de beaucoup de tâches pour être sûr de ne pas repasser par la phase de planification. L'organisation via des outils comme le diagramme de Gantt n'a pas totalement fonctionné. S'adapter à ce format en étant en période de cours avec des horaires variables et du temps de travail personnel plus ou moins conséquents n'était pas une tâche facile. Ce n'est pas pour autant que le partage des tâches ne s'est pas fait sans soucis entre notre groupe, simplement pas grâce à un Trello ou à un Gantt.

Nous avons aussi dû apprendre à utiliser des outils comme SDL ou CUnit que nous ne connaissions pas.

6.2 Notre expérience

Réaliser ce projet nous a permis d'utiliser des compétences acquises tout au long de nos études en informatique, ce projet nous a également permis d'apprendre à gérer le temps d'une manière plus optimale afin de respecter des délais.

7 Annexes

7.1 Organisation

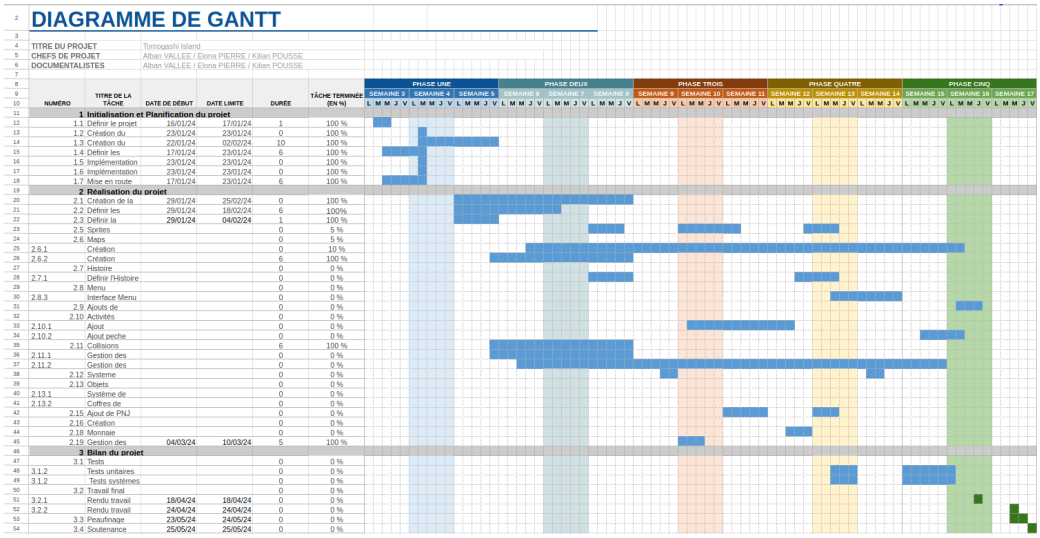

DIAGRAMME DE GANTT

TITRE DU PROJET	Tomogashi Island
CHEFS DE PROJET	Alban VALLEE / Elona PIERRE / Kilian POUSSE
DOCUMENTALISTES	Alban VALLEE / Elona PIERRE / Kilian POUSSE

NUMÉRO	TITRE DE LA TÂCHE	DATE DE DÉBUT	DATE LIMITE	DURÉE	TÂCHE TERMINÉE (EN %)
	1 Initialisation et Planification du projet				
	1.1 Définir le projet	16/01/24	17/01/24	1	100 %
	1.2 Création du	23/01/24	23/01/24	0	100 %
	1.3 Création du	22/01/24	02/02/24	10	100 %
	1.4 Définir les	17/01/24	23/01/24	6	100 %
	1.5 Implémentation	23/01/24	23/01/24	0	100 %
	1.6 Implémentation	23/01/24	23/01/24	0	100 %
	1.7 Mise en route	17/01/24	23/01/24	6	100 %
	2 Réalisation du projet				
	2.1 Création de la	29/01/24	25/02/24	0	100 %
	2.2 Définir les	29/01/24	18/02/24	6	100%
	2.3 Définir la	29/01/24	04/02/24	1	100 %
	2.5 Sprites			0	5 %
	2.6 Maps			0	5 %
2.6.1	Création			0	10 %
2.6.2	Création			6	100 %
	2.7 Histoire			0	0 %
2.7.1	Définir l'Histoire			0	0 %
	2.8 Menu			0	0 %
2.8.3	Interface Menu			0	0 %
	2.9 Ajouts de			0	0 %
	2.10 Activités			0	0 %
2.10.1	Ajout			0	0 %
2.10.2	Ajout peche			0	0 %
	2.11 Collisions			6	100 %
2.11.1	Gestion des			0	0 %
2.11.2	Gestion des			0	0 %
	2.12 Système			0	0 %
	2.13 Objets			0	0 %
2.13.1	Système de			0	0 %
2.13.2	Coffres de			0	0 %
	2.15 Ajout de PNJ			0	0 %
	2.16 Création			0	0 %
	2.18 Monnaie			0	0 %
	2.19 Gestion des	04/03/24	10/03/24	5	100 %
	3 Bilan du projet				
	3.1 Tests			0	0 %
3.1.2	Tests unitaires			0	0 %
3.1.2	Tests systèmes			0	0 %
	3.2 Travail final			0	0 %
3.2.1	Rendu travail	18/04/24	18/04/24	0	0 %
3.2.2	Rendu travail	24/04/24	24/04/24	0	0 %
	3.3 Peaufinage	23/05/24	24/05/24	0	0 %
	3.4 Soutenance	25/05/24	25/05/24	0	0 %

PHASE UNE | PHASE DEUX | PHASE TROIS | PHASE QUATRE | PHASE CINQ

SEMAINE 3 | SEMAINE 4 | SEMAINE 5 | SEMAINE 6 | SEMAINE 7 | SEMAINE 8 | SEMAINE 9 | SEMAINE 10 | SEMAINE 11 | SEMAINE 12 | SEMAINE 13 | SEMAINE 14 | SEMAINE 15 | SEMAINE 16 | SEMAINE 17

L M M J V L M M J V L M M J V L M M J V L M M J V L M M J V L M M J V L M M J V L M M J V L M M J V L M M J V L M M J V L M M J V

FIGURE 22 – Diagramme de Gantt prévisionnel

7.2 Les sprites

FIGURE 23 – Les sprites PNJ

FIGURE 24 – Exemple de sprite du personnage principal

FIGURE 25 – Exemple d'échange

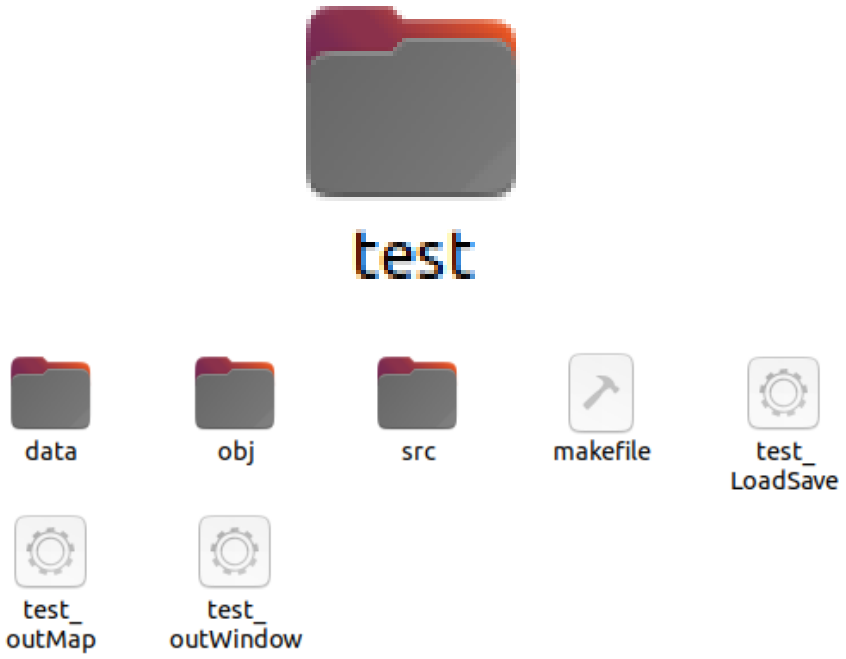

FIGURE 26 – Image de nos tests

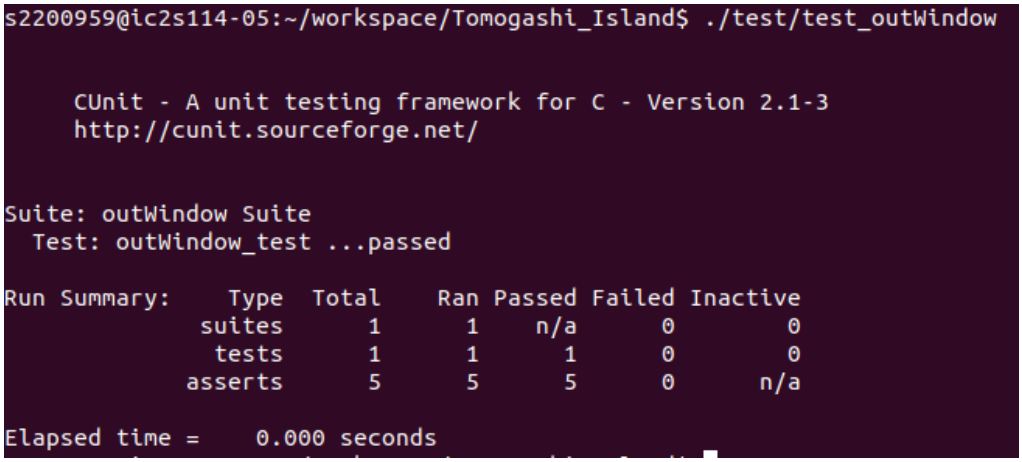

FIGURE 27 – Image de nos tests

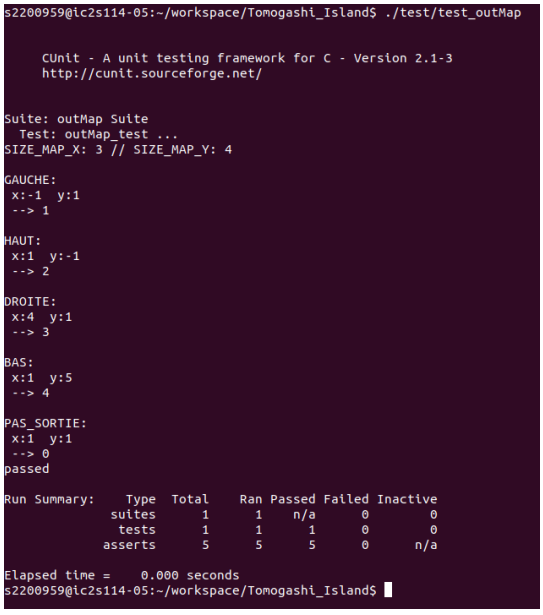

FIGURE 28 – Image de nos tests

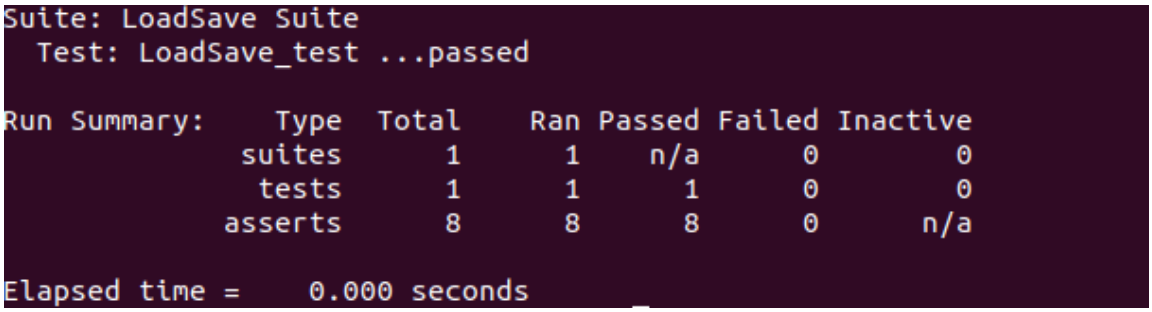

FIGURE 29 – Image de nos tests

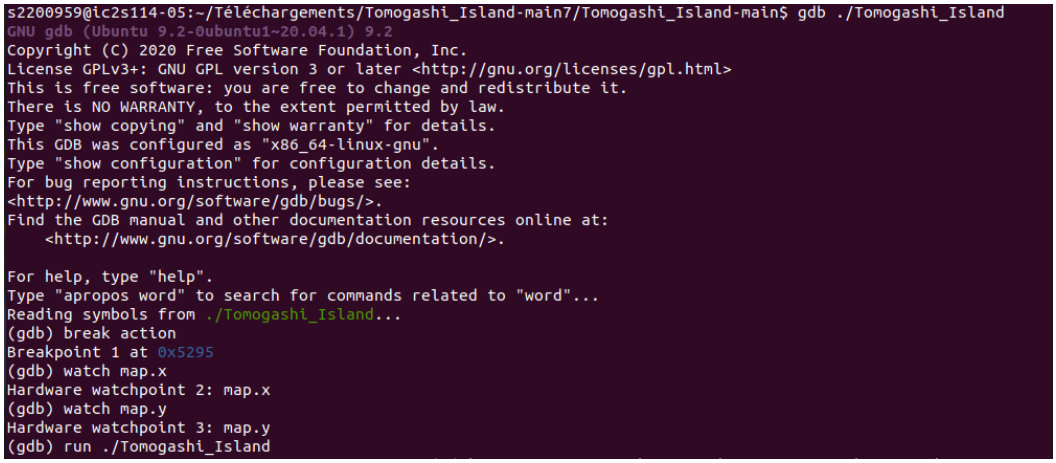

FIGURE 30 – Image de nos tests

FIGURE 31 – Image de nos tests