Matière : Informatique
Pré-ing 1, MI 02, Groupe Trie-k

Rapport
Projet Informatique

EL JARJINI Hicham

HALHOUTE Alexandre

NGUYEN Mathieu

Année universitaire à CY TECH 2023/2024

Sujet et description de l’équipe

Notre projet est basé sur le jeu Cosmic Yonder en 2D, intégrant une génération procédurale de salles. Le joueur incarne un accro de l’argent qui est en quête d’obtenir toujours plus. Il se retrouve donc dans son casino habituelle appelé “L’Aleatorium”. Dans ce casino, plusieurs portes s’offrent au joueur donnant sur de nouvelles salles. Mais pour les ouvrir, ce dernier doit obtenir des clés via des mini-jeux ou quêtes et ainsi progresser dans l’histoire.

Nous sommes une équipe constituée de trois personnes : Hicham EL JARJINI, Alexandre HALHOUTE et Mathieu NGUYEN. Nous avons ainsi chacun contribué à réaliser ce projet ambitieux qu’est “L’Aleatorium”. Hicham EL JARJINI s’est occupé de la génération des salles, Alexandre HALHOUTE et Mathieu NGUYEN ont réalisé les mini-jeux. Quant à l’histoire du jeu et son déroulement ont été créée par les trois membres de l’équipe.

Problèmes rencontrés et les solutions

Problèmes rencontrés :	**Solutions apportées et résultats**
Après avoir réussi à générer et afficher une salle, un problème s'est posé : comment faire pour afficher deux salles ? Étant donné qu'une salle est représentée par un tableau, il n'est possible d'afficher qu'un seul tableau à la fois. Comment faire pour afficher une salle à gauche et une autre à droite ?	Pour pouvoir afficher plusieurs salles à la suite, nous avons eu l'idée de faire comme dans les jeux vidéo : le joueur apparaît sur une carte, et lorsqu'une salle est créée, celle-ci est copiée et collée sur la carte. Ainsi, au lieu d'avoir plusieurs tableaux à gérer, il suffit simplement de gérer un seul grand tableau contenant toutes les informations sur les salles.
Une fois les salles créées, un nouveau problème s'est posé : comment éviter la superposition des salles ?	Pour résoudre ce problème, lorsqu'une porte est créée, elle doit passer par des vérifications. Premièrement, elle vérifie s'il y a assez de place derrière elle pour créer une salle de taille maximale. Ensuite, la porte est associée à des marqueurs. Ces marqueurs permettent d'empêcher les salles voisines de se créer au-dessus d'une salle pas encore générée.

Au tout début, nous codions uniquement sur Windows, mais quelle ne fut pas notre surprise lorsque nous avons appris qu'il existait des terminaux Windows et des terminaux Linux. Ainsi, notre bibliothèque « conio.h » ne fonctionnait pas sur les ordinateurs de l'école ! Le déplacement du joueur était donc à refaire.	Pour résoudre ce problème, nous avons dû nous rendre à l'école pour pouvoir utiliser la bibliothèque 'ncurses.h'. Grâce à cette dernière, nous avons pu refaire le déplacement du joueur malgré les problèmes d'affichage que posait la bibliothèque « ncurses.h ». Pour continuer à coder, nous avons dû ajouter des #ifdef _WIN32 et #ifdef __linux__, ce qui nous a permis de distinguer les parties du code qui peuvent s'exécuter sous Windows et celles qui s'exécutent sous Linux.
LINUX : Lorsqu'on lance le jeu sur Linux, il y a un problème : la première image du jeu ne s'affiche pas. Il faut appuyer sur une touche pour pouvoir afficher le jeu et commencer à s'amuser.	Ce problème est lié à « ncurses.h ». En effet, étant donné que « ncurses.h » est également une bibliothèque qui gère l'affichage, elle a quelque peu perturbé notre code. Nous avons réussi à tout régler en gérant les événements entre chaque image, mais nous n'avons pas réussi à résoudre le problème de la première image du jeu.
WINDOWS : Lors d'une animation utilisant des fonctions de temporisation telles que « sleep() », l'utilisateur peut saisir des données qui seront prises en compte par le programme. Ainsi, certains messages peuvent être manqués car le programme va traiter les données envoyées pendant le « sleep() ».	Ce problème a été résolu sur la version Linux grâce à la bibliothèque « termios », mais cette dernière n'est pas disponible sur Windows. Par conséquent, le problème persiste sur Windows.
NAVI BATUM : Création des trois bateaux aléatoirement de l'ordinateur avec beaucoup de vérifications.	Il a fallu créer une fonction qui initialise les caractéristiques du bateau : le sens, la direction, la taille, les coordonnées (nombres aléatoires). Ensuite, la création d'une fonction qui vérifie si le bateau est possible (s'il n'y a pas déjà un bateau dessus, ou s'il y a de la place). Enfin, si c'est possible, placer le bateau.
NAVI BATUM : Vérification du placement des bateaux par l'utilisateur : il ne fallait pas que l'utilisateur puisse superposer des bateaux ou se retrouver bloqué.	Pour la vérification des bateaux, il a fallu soustraire les coordonnées de la dernière case placée par l'utilisateur avec celles de la première, afin de vérifier si elles étaient alignées et que le bateau placé était droit. Il a également fallu introduire une fonction permettant à l'utilisateur d'effacer et de recommencer.
Création d'un combat avec des attaques et surtout des sorts que l'on peut utiliser tous les temps de tours.	Nous avons mis en place un système avec des compteurs qui se réinitialisent chaque fois que le joueur utilise ses compétences.
Nous avions décidé de réaliser la sauvegarde vers la fin du projet. Cependant, une fois arrivé à ce stade, nous nous sommes rendu compte que notre méthode de gestion des variables n'était pas efficace pour la sauvegarde.	Pour résoudre cela, nous avons créé une structure qui contient toutes les informations importantes à sauvegarder. Nous avons dû modifier nos variables existantes pour les remplacer par les nouvelles variables (par exemple, remplacer « map » par « joueur->map »). Ainsi, la sauvegarde des fichiers est devenue plus simple à réaliser.
Avec notre connaissance du langage C, il était difficile de coder certains aspects du projet, notamment les déplacements dynamiques.	Pour pouvoir gérer la notion de déplacement, nous nous sommes aidés d'Internet et de l'outil « ChatGPT ».

Organisation et flux de travail

L'organisation de l'équipe a été cruciale dans la réussite de ce projet. Une répartition des tâches entre les membres fut la manière que nous avions privilégiée comme étant la plus efficace. Notamment grâce aux outils de communication tels que « Whatsapp » et « Discord » qui nous ont permis de discuter régulièrement sur l'avancée du projet et d'organiser les prochaines échéances. Cependant, ces outils ne sont pas suffisants. Ainsi nous nous sommes souvent réunis en présentiel, ce qui nous a grandement aidé afin de tester, modifier et s'entraider pour réaliser ce projet avec l'utilisation de « Github » pour le partage du code entre les membres de l'équipe.

Flux de travail appliqué

- Analyse des besoins : Comprendre en détail les exigences du cahier des charges par des réunions afin de clarifier les attentes.
- Planification : Développer un plan de projet avec des échéances notamment par la planification des tâches pour tous les membres, en commençant à coder dès le choix du sujet effectué.
- Conception : Brouillon afin de comprendre et réaliser certaines exigences comme la génération procédurale de salles.
- Développement : Écriture du code et développement / amélioration des fonctionnalités.
- Tests : Vérifier que chaque mini-jeux, interactions et compilation fonctionne correctement.