Bilan du projet : Thème, spécifications et organisation

1. Comment tout a commencé

Dès le lancement du projet, notre objectif était clair : réaliser un simulateur à événements discrets en langage C ANSI. Mais plutôt que de reproduire un modèle classique (du type caisse de supermarché ou distributeur de tâches), nous voulions un concept à la fois humoristique, motivant et technique.

C'est ainsi qu'est né Jean-Michel Jounix, professeur illuminé par la théorie du chaos et déterminé à réhabiliter Poincaré dans l'imaginaire collectif.

Ce personnage haut en couleurs nous a permis d'intégrer des mécaniques de jeu variées (PNJ, quêtes, inventaire, temps simulé), tout en respectant les contraintes du simulateur à événements discrets.

2. L'idée du jeu

Objectif du jeu : Faire évoluer Jean-Mi dans sa carrière, écrire des livres, survivre aux absurdités administratives, et atteindre un but ultime : devenir la théorie du chaos incarnée.

Le jeu s'inspire des Sims dans sa logique de services et d'état du personnage. Nous avons ajouté :

- Une carte 2D avec affichage graphique en SDL2 et maps gérées sous Tiled (.tmx)

- Des PNJ avec dialogues contextuels et aléatoires

- Un inventaire d'objets de quête (livre, manuscrit, tampon, etc.)

- Une gestion du temps simulé (jour/nuit, événements conditionnels)

- Un système de quête à étapes

- Un menu admin pour déboguer, tester ou manipuler les variables

Chaque action influe sur les statistiques (faim, énergie, PV), et les déplacements entre maps peuvent déclencher de nouveaux événements ou dialogues.

3. Approche et organisation

Nous avons travaillé en trinôme (Thomas, Michel, Gabriel) avec une répartition souple des rôles :

- Michel s'est focalisé sur la structure du code, la logique C, les structures et les pointeurs.

- Gabriel a pris en charge la partie SDL2, l'intégration des maps via Tiled, et l'affichage graphique.

- Thomas s'est chargé du suivi des livrables, de l'intégration globale et a également pris en main l'ensemble du code principal, assumant naturellement le rôle de chef de projet.

Notre fonctionnement :

- Points hebdomadaires sur Teams / Discord

- Utilisation de GitHub pour le partage de code

- Test unitaire et relecture croisée à chaque étape critique

- Prototypage rapide (menu, système de quête, dialogues) puis refactorisation

4. Outils et technologies

- Langage : C ANSI (respect strict des contraintes du cahier des charges)

- Moteur graphique : SDL2 + SDL_ttf + SDL_image

- Cartographie : Tiled (.tmx), chargement dynamique avec lib tmx

- IDE : VS Code (principalement)

- Versioning : Git + GitHub

- Communication : WhatsApp, Discord, Teams

5. Complexités techniques et défis

Le projet a rapidement gagné en complexité avec :

- Une architecture modulaire avec fonctions dédiées (souvent < 20 lignes) et évitant les variables globales

- La gestion d'événements conditionnels en fonction du temps, des interactions, de l'état du joueur

- Un système de quête progressif (via une structure `Quete` avec checkpoints)

- Une interface texte dans le jeu (via afficher_dialogue_box)

- Des tests sur les limites de faim / énergie / PV pouvant entraîner la mort du personnage

- Des sauvegardes binaires de l'état du jeu (map, inventaire, stats, heure, quête)

- Un menu admin F9 permettant de tester tous les cas (modification des gains, téléportation, inventaire, quête...)

Ce challenge nous a permis de mettre en pratique la mémoire dynamique, les pointeurs, la structuration propre du code et la gestion du cycle de vie complet d'une application.

6. Conclusion et perspectives

Nous avons livré un simulateur 2D fonctionnel, narratif et personnalisé, dans l'esprit d'un mini-RPG. Le projet répond aux spécifications de simulation à événements discrets tout en racontant une histoire unique.

Ce qu'on retient :

- Le jeu est stable, jouable, sauvegardable, avec un début et une fin.

- Tous les concepts vus en cours ont été appliqués (pointeurs, fichiers, structures, affichage, etc.)

- Le code est propre, testé, commenté, et facilement évolutif.

Pistes d'amélioration :

- Intégrer un système d'intelligence plus poussée pour les PNJ

- Ajouter des mini-jeux lors des quêtes (quiz, QTE...)

- Proposer un affichage mobile-friendly (SDL2 + responsive)

- Gérer des logs en 2D visibles à l'écran

- Multiplier les fins possibles (arborescence de quête)

Ce projet a été une véritable expérience de conception de jeu, de gestion de projet, et de programmation système. Une aventure à la hauteur de Jean-Mi.