Rapport du Projet: Cy Fest

Pour ce projet, notre équipe a choisi Cy Fest. Mathias s'est principalement occupé du mode festivalier, tandis que Romain et Tran se sont concentrés sur le mode manager.

Durant le projet, nous avons rencontré plusieurs problèmes. Tout d'abord, nous avons eu de petites erreurs avec le Makefile que nous avions généré avec ChatGPT pour pouvoir commencer à coder rapidement. La solution trouvée était assez simple : chercher un modèle comprenant les parties principales d'un Makefile, puis le compléter progressivement au fur et à mesure que nous créions des fichiers. Ensuite, des problèmes de compilation lors de la commande « make » nous ont forcés à modifier plusieurs fois le fonctionnement de notre programme en se référant au CM pour les résoudre.

Nous avons également rencontré certains crashs causés par des entrées utilisateur qui faisaient planter le programme si elles n'étaient pas du type voulu. Cela nous a pris beaucoup de temps pour trouver la solution. Nous avons donc décidé de créer deux fonctions pour améliorer la fonction **scanf**. Ces nouvelles fonctions effectuent des tests complets sur le type et les valeurs des données saisies, et suppriment les entrées utilisateur erronées, ce qui avait pour effet, au premier abord, de créer des boucles infinies d'erreurs.

Parmi les erreurs les plus récurrentes, il y avait les erreurs de segmentation, souvent causées par des problèmes d'allocation. Ces erreurs nous plongeaient souvent dans un mystère total, car nous ne savions pas où se situait l'erreur ni pourquoi le code avait des problèmes et ne fonctionnait pas. De plus, il était rarement évident de savoir si le problème venait du mode festivalier ou du mode manager. Un problème de segmentation nous a coûté beaucoup de temps de recherche pour finalement découvrir que la solution était simple : enlever les -1 dans les indices de tableaux. En recherchant sur internet, nous avons découvert l'outil « GDB » qui permet de mettre en pause le programme à certains instants, et de vérifier les données de chaque variable pour identifier où le programme avait un problème.

L'option de compilation **-fsanitize=address** de GCC nous a également permis de corriger les fuites de mémoire présentes dans certaines fonctions de notre programme.

Nous avons également eu des problèmes d'incompréhension des fonctions codées par d'autres membres de l'équipe, ce qui nous a permis de comprendre l'importance des commentaires dans le code et de discuter des fonctions avant leur création. Un autre problème majeur était une mauvaise préparation de la structure du code au début. En effet, même si nous avions établi un schéma de ce qu'il fallait coder, nous avons été obligés de changer la plupart des idées et d'ajouter de nouvelles fonctions en cours de route.

De plus, nous avons mal compris certaines consignes. Par exemple, nous n'avions pas compris que les concerts et les salles de concerts devaient être séparés. Par conséquent, au milieu de l'avancement du projet, nous avons dû changer presque toutes les fonctions et en ajouter de nouvelles.

Nous avons souvent utilisé ChatGPT pour identifier les erreurs « bêtes » sans avoir à solliciter les autres membres du groupe, ou tout simplement comprendre pourquoi notre programme ne fonctionnait pas. Nous avons également organisé des réunions d'équipe pour nous entraider et trouver des

solutions à nos problèmes. Notre principal problème a sûrement été de n'avoir fait qu'une « esquisse » de notre projet sur le logiciel Figma avant de commencer et de ne pas avoir fait un plan plus précis de notre projet, ce qui aurait pu nous faire gagner beaucoup de temps. Néanmoins, nous sommes fiers d'avoir réussi à surmonter de nombreux problèmes pour finalement obtenir le résultat que nous attendions. Les efforts et le temps investis dans la création de notre projet ont été payants. Bien que le résultat final diffère de notre vision initiale, nous sommes satisfaits du produit fini.