ENSTA

École Nationale Supérieur de Techniques Avancées.

Rapport du projet IN204

Andrés Eloy RUBIO CARVAJAL

Table des matières

À propos du projet

La tâche principale était de créer un jeu Tetris, pour un joueur unique et multijoueur avec un réseau, développé en C++ avec l'aide de l'API appelée SFML, où vous pouvez trouver des moyens de gérer des choses comme les fenêtres, les couleurs, la connexion TCP, les événements, les variables de temps, entre autres.

Compression pour accomplir cette tâche

Au début, il était impossible de comprendre comment écrire le code sans d'abord essayer de comprendre comment le jeu fonctionne en général, il nécessite différentes fenêtres, pièces, terrain de jeu, réseau, etc.

Pour pouvoir comprendre cela, je me suis donné pour tâche de faire un diagramme des classes à la main où je différenciais les classes du terrain de jeu, avec sa taille, son emplacement, ses méthodes comme la révision des impacts des pièces, le mouvement des pièces, la révision des points pour faire une ligne dans le jeu, a également trouvé claire l'idée des classes des pièces, avec ses différentes formes et sa position qui avec une petite multiplication résout le problème de la rotation ayant une nouvelle position pour la montrer plus tard. Entre la rotation et le mouvement, il faut tenir compte de l'état futur, effectuer l'action et voir si elle ne respecte aucune restriction, c'est-à-dire une solution pour vérifier que les pièces ne sortent pas de leur limite et n'entrent pas en collision avec d'autres.

J'ai essayé de mettre en place ces deux classes que j'avais définies, en partant de la classe des pièces, où j'avais des difficultés à comprendre comment imprimer les pièces, à trouver une solution avec des vecteurs unitaires, à définir les différents points des pièces, quand je me suis retrouvé bloqué parce que je ne savais pas comment traiter les pièces, je me suis consacré à revoir ce dont j'avais besoin d'autre parmi lesquels il fallait aussi une classe de fenêtres où l'on trouverait les différentes fenêtres à afficher comme le menu, la connexion multijoueur et le jeu, à la fois solo et multijoueur.

J'ai essayé de mettre en place ces deux classes que j'avais définies, en partant de la classe des pièces, où j'avais des difficultés à comprendre comment imprimer les pièces, à trouver une solution avec des vecteurs unitaires, à définir les différents points des pièces, quand je me suis retrouvé bloqué parce que je ne savais pas comment traiter les pièces, je me suis consacré à revoir ce dont j'avais besoin d'autre parmi lesquels il fallait aussi une classe de fenêtres où l'on trouverait les différentes fenêtres à afficher comme le menu, la connexion multijoueur et le jeu, à la fois solo et multijoueur.

les prochaines étapes

En parlant du code et des choses nécessaires pour l'accomplissement de cette tâche, il serait idéal d'avoir une classe d'événements qui soit le père des différentes fenêtres, dans les fenêtres de forme générique on peut trouver des étiquettes, des boutons. Par conséquent, avoir une classe qui affiche un bouton avec sa taille et son texte serait utile, ainsi qu'une classe qui permet d'avoir des étiquettes à égalité avec le texte ce qui est réalisé en tenant compte de l'aide de la SFML lors de l'implémentation d'une police au projet et ensuite de l'appeler depuis le projet selon le chemin où le fichier de Fonts a été enregistré. Pour le terrain de jeu et les pièces, nous aurions besoin d'une classe de jeu, qui peut créer à l'intérieur les objets, les pièces et le terrain de jeu, ici nous prendrions en compte l'ajout de points après chaque fois que la classe du terrain de jeu supprime une ligne, nous devons prendre en compte le fait que nous avons besoin d'une variable d'accumulation de temps à l'intérieur de l'événement du jeu qui permet de mettre à jour l'écran chaque fois que des changements se produisent dans une certaine période de temps, l'utilisation d'étiquettes de texte permet normalement d'afficher les points du joueur ainsi que le temps de jeu, qui est une variable qui s'incrémente à chaque fois qu'une ligne est supprimée et l'autre est la variable de temps à l'intérieur de l'événement du jeu, ce qui diffère du temps de l'événement de l'application, les différents événements nécessitant une variable de temps pour être mis à jour.

Naturellement, tout cela serait l'essence même d'un joueur unique, le multijoueur qui envisage la connexion à un réseau Lan nécessite l'aide de TcpSocket et de Packet que la SFML fournit pour établir des connexions entre une classe Host et une classe de client, Host est créé de telle sorte que les informations envoyées sous forme de paquets à l'adresse IP sélectionnée soient partagées entre ceux qui font partie de Host, ce sont les clients, qui s'envoient des informations entre eux, ce serait le joueur un et le joueur deux, en prenant comme suggestion la création d'une classe joueur un et deux qui différencie les attributs du jeu, pour la mise en place d'un événement multi-joueurs est plus facile à utiliser comme base ; la même création de l'événement d'un joueur, un terrain de jeu plus petit est créé à côté du terrain de jeu original, qui recevra des informations de la connexion avec le réseau lan correspondant aux mises à jour de cette nouvelle matrice ainsi que les points, de cette façon la partie multi-joueurs n'aurait qu'à recevoir des informations et mettre à jour la fenêtre afin que vous puissiez voir les terrains de jeu de deux joueurs différents.

Conclusions

J'ai appris de cette expérience comme mon premier projet de faire un jeu en programmation, de ne pas essayer de tout faire parfaitement du premier coup, les classes que l'on peut arriver à penser peuvent être très bonnes mais l'implémentation du code et les problèmes que cela représente ont beaucoup plus de poids dans les décisions de comment structurer les classes, personnellement j'ai essayé de faire les classes que je pensais depuis le début d'une manière parfaite, où aucun détail ne manque à l'extérieur, avoir des conflits logiques pour cette raison, où j'avais des blocs juste pour définir une fonction, en considérant à quoi elle servirait et où je l'utiliserais, en tenant compte de la première vue du projet est la première étape pour commencer, diviser cela en petites tâches pour vérifier son utilité pourrait être l'étape numéro deux, et aller mettre en œuvre de petites fonctions qui exécutent des tâches simples du projet avec une autre classe et ainsi vérifier qu'il est bien mis en œuvre.

Il n'est pas possible de faire un projet seul sans la base de sa gestion claire, c'est ce que j'ai rencontré quand j'ai voulu faire ma première classe concernant les pièces, j'ai eu une confusion logique, je ne savais pas comment le mettre en œuvre après tout je pensais à le faire, puis j'ai pensé à faire un . cpp pour faire les choses que je voulais à partir d'une classe pour montrer les pièces et les déplacer, en utilisant les vecteurs2d que possède la SFML Je pouvais comprendre que je pouvais utiliser les objets à partir de positions, et avec ces positions pour les montrer, je l'ai fait sans classes à l'intérieur de ce fichier test, mais ensuite je ne comprenais pas comment l'implémenter, cela me bloquait constamment surtout quand je trouvais des erreurs de syntaxe que je ne comprenais pas et que je ne pouvais pas trouver sur internet. Naturellement, les classes que je pensais avoir subi de nombreux changements constants avec ces tests parce que je ne pouvais pas implémenter les objets dans un autre fichier, bien que je pense que c'était une erreur de manipulation des dimensions mais je ne pouvais pas en comprendre la cause.