Jeu du Mastermind

Projet UE-Programmation Orientée Objet

BESQUEUT Corentin et VILLEDIEU Maxance

29/04/2024

Table des matières

Chapitre 1

Architecture du Jeu (Client)

1.1 Présentation du Projet

La production du jeu Mastermind s'est déroulée en deux grandes étapes. D'abord, nous nous sommes occupés de programmer le jeu en lui-même, avec des choix architecturaux simple et qui seront expliqués dans les prochains chapitres. Nous avons ensuite rendu l'affichage en console esthétiquement correct étant donné que nous avons décidé de ne pas proposer d'interface graphique. En effet, nous avons décidé de s'attarder sur la deuxième étape de notre projet : la création d'un mode multijoueur avancé. Il sera détaillé dans un chapitre qui lui est dédié dans ce rapport.

1.2 Fonctionnement général du jeu

Le jeu ci-joint fonctionne à l'aide de plusieurs threads, que ce soit côté client et côté serveur. Dans ce chapitre, nous nous attarderons uniquement sur le côté client.

Au lancement, le jeu (client) s'ouvre sur la page d'accueil, où une boucle va attendre une entrée de commande de la part de l'utilisateur. Une fois avoir analysé la commande, notre programme va, selon les cas, créer une sous-boucle (pour la boucle de jeu par exemple) qui va s'exécuter dans la boucle principale. Dès la fermeture de la boucle fille (dans notre exemple, de la boucle de jeu), l'utilisateur sera reconduit dans la boucle globale/parent, et donc, dans le menu du jeu.

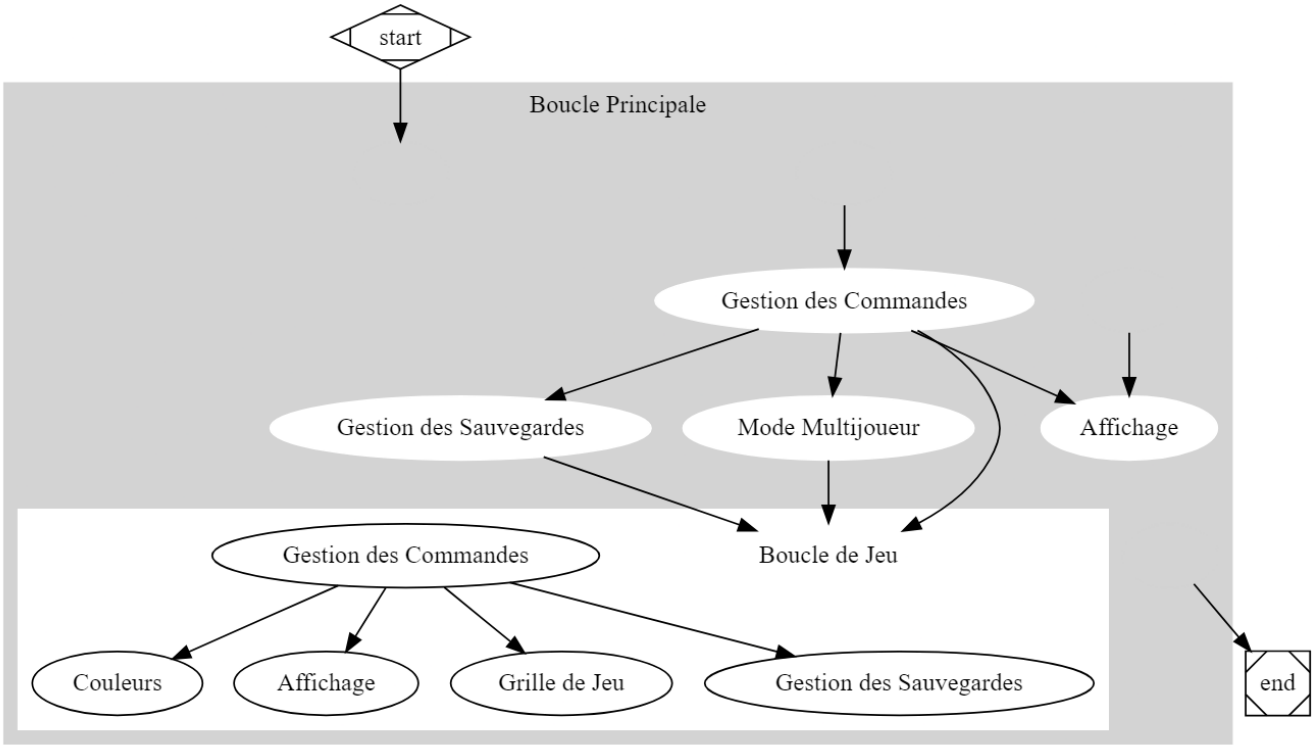
1.3 Décomposition architecturale du jeu

Nous avons choisi de décomposer le jeu en de multiples entités, chacune correspondant soit à une structure soit à un objet du Mastermind. Vous trouverez donc dans le dossier source de notre jeu (client), les différents dossiers comportant toutes les classes du client Mastermind. Leur objectif est expliqué dans le tableau suivant :

Classes du Jeu		
Dossiers	**Classes (*Enum*)**	**But**
utils	ArchiveManager	Gère l'écriture et lecture de fichiers de sauvegarde
	Color (Enum)	*Regroupe les couleurs utilisées par le jeu (pions et marqueurs) ainsi que certaines fonctions d'accès*
	Command	Autonomise la gestion des commandes de l'utilisateur (Lecture, Traitement, Exécution, Erreurs)
	Display	Classe Abtraite regroupant toutes les fonctions d'affichage (erreurs, jeu, aides, header, nom du jeu...)
	PawnType (Enum)	*Catégorise l'identité et la fonction de chaque pion (marqueurs, pion de jeu...)*
	Structurable	Interface permettant de spécialiser toutes les structures de données du jeu (Lignes de pions, essai du joueur, grille de jeu...)
pawn	Pawn	Définit la structure générale d'un pion
	MarkerPawn	Définit les spécificités d'un pion en tant que marqueur
	GamePawn	Définit les spécificités d'un pion en tant que pion de jeu
gamegrid	Grid	Crée la structure de données principale du jeu (regroupant tous les essais du joueur en stockant les pions et les marqueurs associés à chaque essai)
	Line	Implémente une structure de données contenant uniquement des pions afin de stocker des pions ou des marqueurs
	Pattern	Classe étendue de la classe Line permettant de créer le pattern de pions que le joueur doit deviner
game	Game	Implémente la boucle de jeu, ainsi que les fonctions associées (victoire, défaite, paramètres de jeu...)

Table 1.1 – Classes du Jeu

Voici un diagramme représentant non exhaustivement le déroulement principal de notre programme :



Chapitre 2

Mode Multijoueur

2.1 Architecture Serveur

2.1.1 Protocole

Voici les paquets utilisés afin de transmettre les informations essentielles au bon déroulement d'une partie.
Ces paquets implémentent l'interface **Deliverable**.

Type de packet	Description
LOGIN	Packet that allow client to connect with username to server
CUSTOM_MESSAGE	Packet to allow displayed communication
SETTINGS_GAME	Packet to allow client to receive game settings
LAUNCH_GAME	Packet that allows player to launch the game
START_GAME	Packet that allows player to start the game
PLAY_CONTENT	Packet to allow client to send his move
WAIT_FOR_TURN	Packet that allows player to wait for all
CONTINUE_GAME	Packet that allows player to continue the running game
END_CONNECTION	Packet that allows users to be disconnected

2.1.2 Fonctionnement Serveur

Démarrage du Server

Pour démarrer, le serveur à besoin des paramètres suivants (dans l'ordre) :

1. port de connexion.
2. Nombre maximum de joueurs attendus afin de démarrer la partie.
3. Nombre maximum d'essais que les joueurs disposeront afin de trouver le secret.
4. La taille de la solution à trouver (le secret).
5. un booléen **true ou false** désignant si les marqueurs sont mélangés.
6. un booléen **true ou false** désignant le fait que des couleurs peuvent se répéter dans la grille solution.
7. Le nombre de couleurs disponibles pour le joueur pendant la partie.

Par exemple, pour initialiser une partie "classique" de Mastermind côté serveur, en console, nous donnerons la ligne suivante :

```
<acces a la classe Server> 55678 2 10 4 true true 8
```

A noter qu'il s'agit là d'un serveur *TCP* (même si un système client/serveur en **UDP** a également été produit).
Ce serveur admet différents états en fonction du déroulement de la partie ainsi que différentes commandes.

Commande	Description
/kick <ip> <port>	Permet d'éjecter un joueur du serveur
/list	Permet de voir la liste des joueurs en ligne
/solution	Permet de révéler, seulement côté serveur, la solution
/forcestart	Permet de forcer le lancement d'une partie (si la salle d'attente n'est pas pleine)

Le système embarque donc l'API du jeu. Le serveur est au format **ROOM** c'est à dire qu'à la fin de chaque partie, le serveur s'éteint.
Toute connexion avec un client est conservée au moyen d'un Thread géré par un ThreadPool.

2.2 Architecture Client

2.2.1 Démarrage

Pour démarrer un mode multijoueur, il faudra, dans le menu principal du jeu, exécuter la commande /**multi** <**username**> <**ipAddress**> <**port**> (c.f /**help** dans le jeu).

Remarque : Si c'est l'un des joueurs qui va "host" le serveur du jeu, il pourra accéder à une partie multijoueur en renseignant ***localhost*** *à la place de l'adresse Ip. Cela rentre également en compte si plusieurs joueurs jouent en local (sur la même machine).*

2.2.2 Commandes en partie multijoueur

Commande	Description
/color $c_1 \dots c_k$	Permet d'envoyer k couleur au serveur
/chat <msg>	Permet de communiquer avec les autres utilisateurs
/infos	Permet de faire un rappel des règles de la partie en cours (gérée par le serveur)

2.2.3 Déroulement de la partie

Durant la partie, pour chaque tour t, les joueurs jouent sans ordre en sachant que la liste des gagnants ne sera révélée seulement à la fin de ce tour t.

Ce court rapport ne pouvant pas dépasser les trois pages de contenu, nous ne pouvons pas décrire de manière satisfaisante l'ensemble du projet. Si toutefois vous avez des questions à propos du projet en lui-même, du fonctionnement ou de l'architecture du jeu, ou quelconque autre question, vous pouvez nous contacter à l'aide de nos mails étudiants suivants :

— Corentin.BESQUEUT@etu.uca.fr
— Maxance.VILLEDIEU@etu.uca.fr