POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

LOG2810

STRUCTURES DISCRÈTES

TP2 : AUTOMATES ET LANGAGES

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

ROY BOUTROS (2012197)

JEAN-PAUL KHOUEIRY (2011397)

REMIS LE DIMANCHE 7 DÉCEMBRE 2020

Introduction :

Les jeux de société ont beaucoup évolué depuis le début de leur invention. De nos jours, et surtout pendant le contexte de la pandémie, les avides de jeux de sociétés sont de plus en plus portés à jouer à ces jeux en ligne.

Dans ce contexte socio-culturel présent, il est de notre devoir d'adapter les jeux de société pour les rendre en ligne. Dans notre cas, nous allons adapter le jeu populaire *Mastermind* avec un mode de jeu automatique et versus. Nous allons utiliser les notions apprises d'automate et de machine à états afin d'implémenter et d'optimiser ce code.

2. Présentation des travaux :

Nous avons conceptualisé le programme en utilisant une approche orientée objet. La solution contient les classes: Automate et Etat. Le diagramme de classes se présente ainsi :

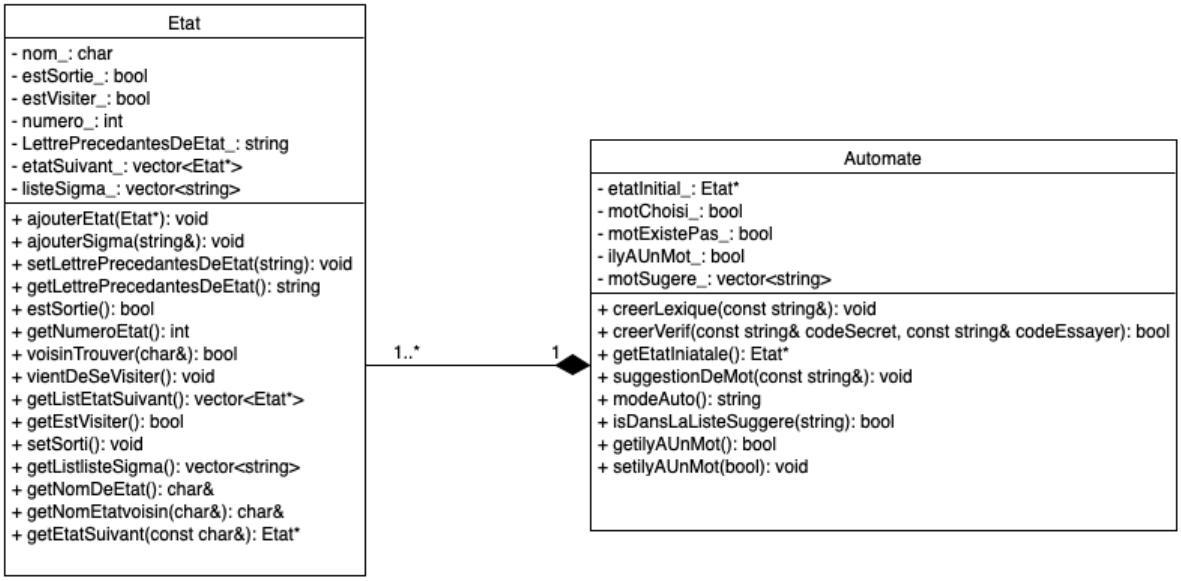

Fig. 1 : *Diagramme de classes complet de la solution*

2.1 Automate :

La classe automate permet de lire un fichier texte de lexique et de créer un automate de tous les mots dans chaque lexique. La classe automate contient les méthodes de base, soit les constructeurs et le destructeur. Les méthodes importantes sont creerLexique(), qui génère un automate à partir de chaque lexique et creerVerif(), qui utilise Mealy pour vérifier et comparer les mots. Les attributs privés sont le nombre d'états et un état initial.

2.2 Etat :

La classe Etat, qui est reliée à la classe automate, représente les états différents dans l'automate. Elle contient les constructeurs et le destructeur, évidemment ainsi que les getters. Cette classe contient aussi les fonctions : ajouterEtat(), qui rajoute un état à l'automate, ajouterSigma(), qui rajoute une lettre dans notre liste de sigma et voisinTrouver(), qui affirme si un voisin à une lettre mis en paramètre est trouvé. Les attributs privés représentent le nom de l'état, un bool qui affirme si l'état est une sortie ou non, le numéro de l'état ainsi que d'un vecteur d'états suivants et un autre de sigma.

2.3 Fonction principale (main) :

C'est là que l'important du jeu se déroule. Le main contient le menu, et fait appel aux fonctions creerLexique(), modeAuto() et modeVersus(). Quand l'usager démarre le programme, il voit le menu, qui lui propose trois options différentes. L'option (1) permet de lire un lexique et d'en créer un automate. Cette option doit obligatoirement être effectué avant de continuer dans le programme. L'option (2) initialise le jeu en mode auto, où l'usager va pouvoir jouer contre l'ordinateur, qui lui choisira un mot secret au hasard dans l'automate construit à partir du lexique. L'option (3) permet d'initialiser le jeu en mode versus, où deux joueurs pourront s'affronter. D'abord, le premier joueur va choisir un mot à deviner. Le programme se charge de lui dire si le mot voulu se trouve dans le lexique et peut aussi proposer des choix de mots si l'usager entre des caractères préfixe à un mot dans le lexique. Le deuxième joueur pourrait ensuite entrer un mot pour deviner le code secret; le programme le laissera savoir combien de lettres dans le mot sont mauvaises, tant que l'usager ne devine pas le code (jusqu'à 15 fois).

2.3.1 Fonctions

- creerLexique() :

Cette fonction créer un automate à partir d'un fichier texte. Nous lisons tout le fichier texte, ligne par ligne pour obtenir tous les mots. Chaque lettre différente de chaque mot est représentée par un état unique. Nous évitons de créer un nouvel état pour les mêmes lettres avec la condition de la méthode voisinTrouver() en vérifiant tous les voisins sortant de chaque état (soit chaque lettre) et si on n'a pas encore créé un état à cette nouvelle lettre on le créer.

- creerVerif() :

Cette fonction vérifie, à l'aide des états de l'automate, que le code secret et le code entré par l'utilisateur sont pareils. Il initialise les états du code secret et les compare avec les caractères du code essayé. Cette fonction se charge aussi de compter le nombre de lettres erronées dans le mot entré par l'utilisateur.

- modeAuto() :

Cette fonction utilise l'automate du lexique afin de trouver un mot au hasard dans l'automate déjà créer à l'aide la fonction creerLexique(). La fonction traverse aléatoirement tous les états jusqu'à temps qu'elle arrive à un état final et parfois si cet état final a encore des états suivants elle pourra aussi ne pas s'arrêter au premier état final et prendre un mot qui est plus grand.

- modeVersus() :

Cette fonction fait appel à la fonction de suggestionDeMot(). La fonction appelée s'occupe de prendre en paramètre une chaine de caractères quelconque et s'occupe de traverser l'automate crée pour suggérer à l'usager des mots qui ont la chaine entrée en paramètre comme préfixe. On d'ailleurs une queue et le principe de BFS pour itérer dans l'automate afin de trouver la liste de suggestion de mot.

3. Difficultés rencontrées :

Initialement, nous avons eu de la difficulté pour implémenter la fonction creerLexique(). C'était la première fois que nous implémentons un automate et nous avons fait beaucoup de recherche au début pour bien s'orienter. Nous avons aussi fait face à des problèmes similaires en ce qui concerne l'implémentation du Mealy. Le reste du code se faisait plutôt bien.

Conclusion :

Pour conclure, ce travail nous a donné l'opportunité d'utiliser nos connaissances apprises en structures discrètes à propos des langages pour recréer le jeu populaire *Mastermind* en code. Nous avons donc pu améliorer nos connaissances en orienté objet C++ tout en réutilisant les notions apprises en cours.