Faculté des sciences Jean Perrin

Puissance 4

Patrick PLOUVIN et Patrice PLOUVIN

Année universitaire : 2022-2023

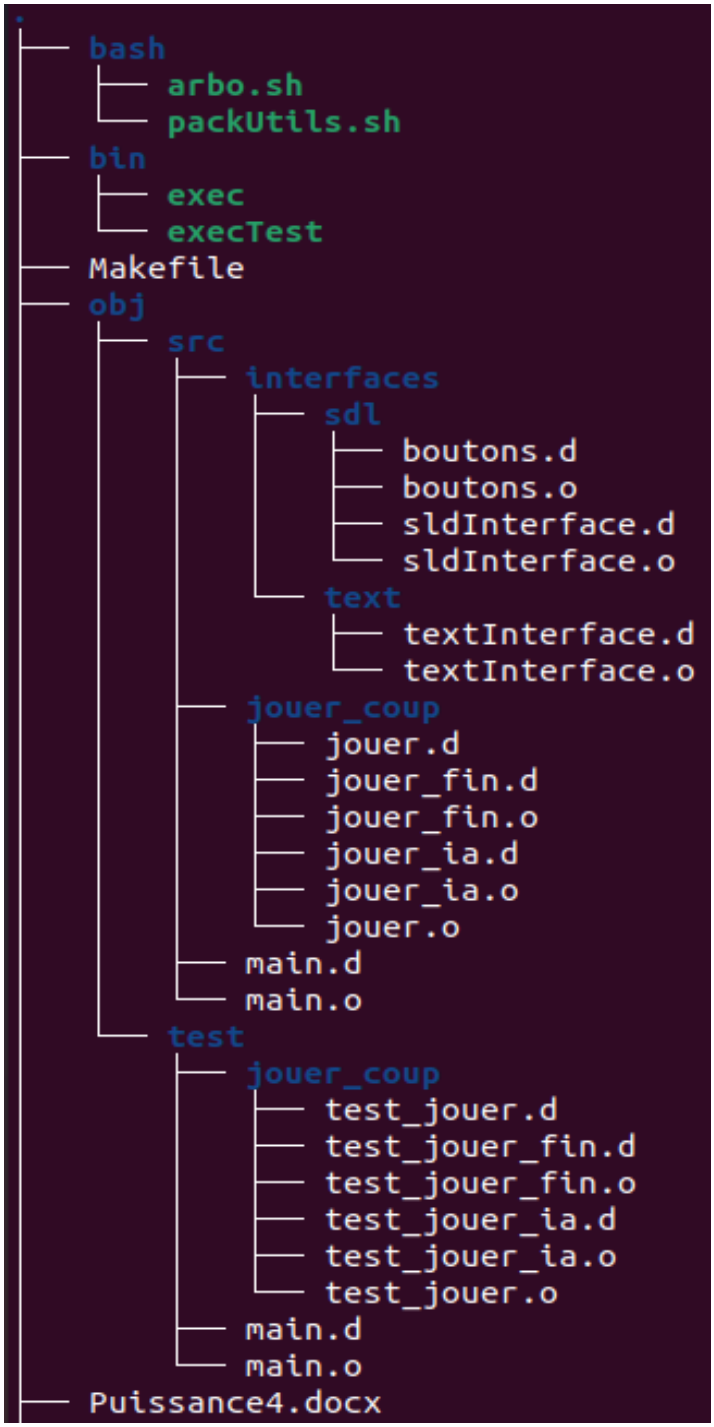
Tables des matières

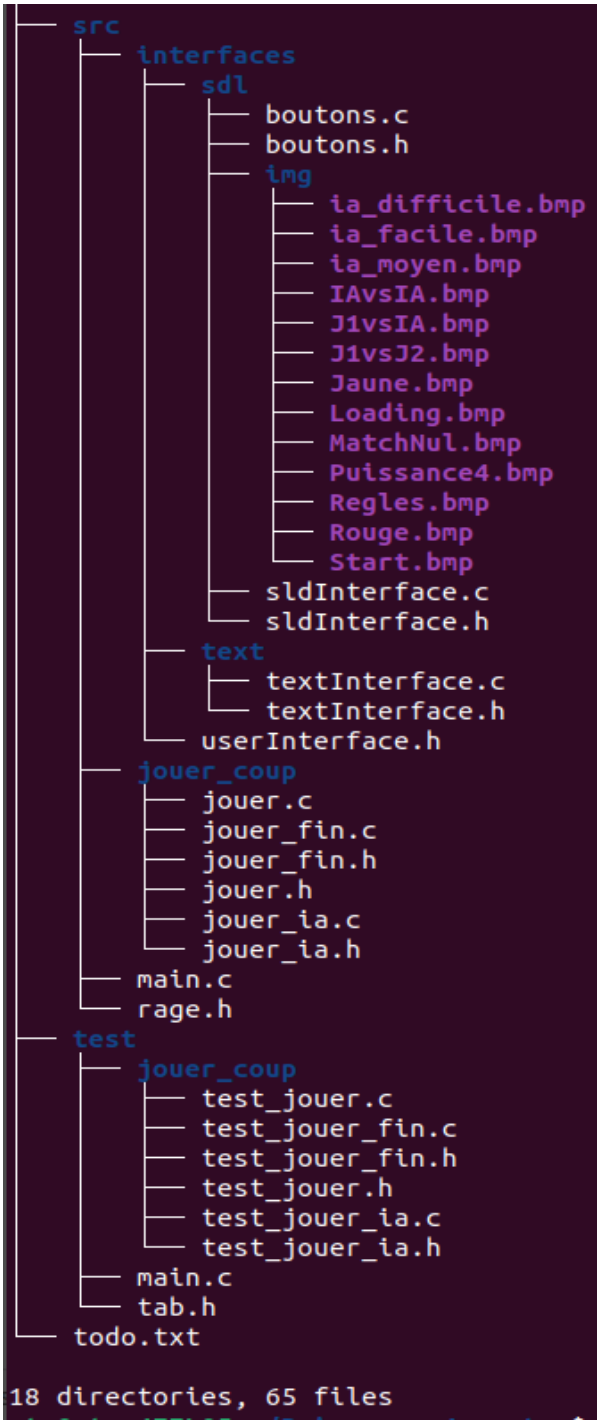
Tables des matières 2
Introduction 2
Structure du projet 3
Le Makefile 4
L'interface utilisateur 5
Choix sur les IA 6
Choix de la SDL 6
Ce que nous n'avons pas réussi à faire 7
Pour aller plus loin… 8

Introduction

Le projet consiste en la réalisation d'un programme permettant de jouer au jeu "Puissance 4". Ce jeu se joue sur une grille de 6 rangées et 7 colonnes, et le but est d'aligner 4 pions de même couleur de manière consécutive (horizontalement, verticalement ou en diagonale). Chaque joueur dispose de 21 pions d'une couleur, et ils doivent tour à tour placer leur pion dans une colonne libre. Le premier joueur à réaliser un tel alignement remporte la partie. Si la grille est remplie sans qu'aucun joueur n'ait réalisé d'alignement, la partie est déclarée nulle. Pour ce projet, il sera possible de jouer en mode console ou en mode graphique grâce à la librairie SDL2, et plusieurs modes de jeu seront proposés : humain vs. humain, humain vs. IA, IA vs. IA. L'IA proposera 3 niveaux de difficulté : facile, normal et difficile. Le code sera commenté de manière à pouvoir générer la documentation automatiquement, et des tests unitaires seront écrits pour vérifier le bon fonctionnement du programme. Enfin, le programme devra être compatible avec les machines de salle de TP sans avoir besoin de librairies externes supplémentaires.

Structure du projet





Accessible via un make arbo.

Le répertoire «obj» comporte tous nos fichiers compilés. Dans le dossier «src/interface» nous trouvons deux dossiers, le dossier «sdl» comporte les fichiers qui permettent d'afficher notre interface graphique ainsi que les boutons visibles lorsque l'on joue.Il comporte aussi les messages de victoire ect.. Le fichier text comporte nos fichiers permettant de jouer avec le terminal. Ensuite nous avons un dossier qui contient tous nos fichiers permettant aux joueurs et aux IA de jouer. Le dossier test comporte les tests sur les fichiers 'jouer'.

Pour des soucis de visibilité, les fichiers de la documentation ont été supprimés pour plus de clarté. A savoir qu'en plus dans la structure il y a un fichier doc qui comporte tous les fichiers nécessaires à la documentation en ligne (HTML). Il y a aussi notre fichier Doxyfile qui est notre paramètre pour la documentation.

Le Makefile

Le makefile décrit ci-dessus permet de gérer la compilation et l'exécution du projet de manière automatisée.
Voici les commandes :

La commande <make install> permet de vérifier que les paquets nécessaires à la compilation du projet sont bien installés sur la machine.

La commande <make> permet de compiler le projet. Elle crée un dossier obj où tous les fichiers objets (.o) seront stockés, ainsi que les fichiers de dépendances (.d) qui listent les dépendances de chaque fichier, en conservant l'arborescence du dossier src. Pour lancer le programme, il suffit d'entrer la commande ./bin/exec, ou ./bin/execTest pour les tests.

La commande <make run> compile le programme avec make et exécute immédiatement l'exécutable du programme.

La commande <make runtest> compile le programme avec make et exécute immédiatement l'exécutable des tests.

La commande <make arbo> donne l'arborescence depuis le dossier Puissance_4.

La commande <make memory> exécute valgrind sur l'exécutable du programme avec les options adéquates.

La commande <make clean> supprime le dossier obj.

La commande <make doc> appel doxygene doxyfile, permet de créer la doc.

La commande <make cleandoc> supprimer la doc.

La commande <make mrproper> supprime le dossier bin où sont stockés les exécutables.

La commande make DEBUG=0 permet de compiler en mode débogage.

L’interface utilisateur

Le fichier userInterface.h gère l'interface utilisateur du programme de jeu "Puissance 4". Il contient plusieurs éléments clés pour le fonctionnement de l'interface :

L'énumération Player définit les symboles pouvant être affichés dans une case du plateau de jeu : vide par défaut, croix ou rond selon le joueur qui joue. La structure Bouton gère les boutons de l'interface utilisateur en utilisant la librairie SDL2. Un bouton est représenté par un rectangle de couleur spécifique.

La structure userInterface contient les informations nécessaires pour afficher l'interface : la fenêtre SDL, le renderer, les boutons et le path global. Si l'interface est en mode texte, ces trois arguments sont nuls.

La structure Puissance gère le plateau de jeu et stocke les informations sur le joueur courant, le mode de jeu et la difficulté de l'IA pendant la partie.

Ces structures et énumérations permettent de gérer l'interface utilisateur de manière cohérente, qu'elle soit en mode texte ou graphique. Elles servent de base pour l'implémentation de l'interface utilisateur du programme.

Choix sur les IA

Lors du développement du jeu "Puissance 4", nous avons décidé d'implémenter trois niveaux de difficulté pour l'IA : facile, moyen et difficile.

1. Facile. En mode facile, l'IA est très facile à battre. Elle ne fait aucun calcul et choisit simplement une colonne au hasard pour y placer son pion, tant que cette colonne n'est pas pleine.

2. Moyen. En mode moyen, l'IA commence à réfléchir avant de jouer. Elle suit plusieurs étapes pour déterminer son coup : si elle peut gagner la partie en jouant un coup précis, elle le joue ; si elle peut bloquer un coup gagnant de l'adversaire, elle joue ce coup ; sinon elle joue le coup que l'adversaire aurait pu jouer s'il avait pu aligner un grand nombre de pions. Elle s'assure alors que son adversaire ne gagne pas en plaçant un pion au-dessus du sien. Si plusieurs choix de colonnes sont possibles pour aligner beaucoup de pions, elle en sélectionne une au hasard parmi celles-ci. Par exemple, si le joueur 1 joue dans la colonne 2, l'IA jouera dans la colonne 1, 2 ou 3. Dans le cas où le plateau de jeu est bien rempli et qu'il existe un risque de voir son adversaire gagner, l'IA peut refuser de jouer pour essayer de faire avancer le jeu. Si le coup qui aurait aligné le plus de pions est joué dans une colonne pleine, l'IA cherche alors une autre colonne où elle peut jouer.

3. Difficile. En mode difficile, l’IA fonctionne comme l’IA moyen à la différence qu’elle génère deux coups avant de jouer le troisième. On peut donc dire qu’elle réfléchit 3 coups à l’avance.

Choix de la SDL

Le choix de la SDL (Simple DirectMedia Layer) permet de créer une interface graphique pour notre jeu. Lorsque le mode de jeu SDL est lancé, une fenêtre s'ouvre affichant les règles du jeu ainsi que des informations sur le fonctionnement des boutons de la page suivante. Cela permet à l'utilisateur de comprendre immédiatement comment jouer sans avoir à chercher comment fonctionnent les différents boutons.

La génération du tableau étant un peu long, un bouton “loading” apparait pour spécifier à l’utilisateur que sa demande a bien été prise en compte. Au relancement d’une

partie, il se peut que le joueur doive attendre avant de sélectionner un bouton car le plateau est regénéré à chaque fin de partie.

Pour faciliter l'utilisation de l'interface, les boutons de sélection de mode de jeu et de difficulté de l'IA sont plus petits que les autres boutons. Cela permet de mettre en valeur leur sélection si l'utilisateur en choisit un. Si l'utilisateur sélectionne un mode de jeu avec l'IA, il doit également choisir la difficulté de l'IA. Une fois que les modes ont été choisis, le plateau attend l'instruction pour placer le premier pion.

Pour afficher les cercles sur le plateau de jeu, plutôt que de gérer une surface pour afficher chaque cercle, de traiter les points de pivot et de réaliser des calculs superflus pour l'allocation de mémoire, nous avons opté pour une solution plus simple en traçant simplement des traits de couleur depuis le centre de chaque case.

Ce que nous n’avons pas réussi à faire

Pour l’IA difficile nous avons essayé d’implémenter l’algorithme Minimax. L’algorithme amène l'ordinateur à passer en revue toutes les possibilités pour un nombre limité de coups et à leur assigner une valeur qui prend en compte les bénéfices pour le joueur et pour son adversaire. Le meilleur choix est alors celui qui minimise les pertes du joueur tout en supposant que l'adversaire cherche au contraire à les maximiser (le jeu est à somme nulle).

Les problèmes rencontré étaient pour la plus part le fait que l’IA jouait de manière aléatoire. De plus l’algo Minimax renvoie une valeur de « score » or ce que nous voulions était la valeur de la colonne où ce score était le plus profitable. Cependant l’IA jouait très souvent sur les côtés. Il n’était pas évident de savoir si l’erreur venait d’une des fonctions que nous avons dû implémenter pour Minimax ou de Minimax.

Nous avons donc opté pour la solution expliqué au-dessus.

Pour aller plus loin…

Nous aurions pu créer une page paramètre pour pouvoir, par exemple, changer la vitesse de l'IA. Nous aurions aimé rendre l'IA beaucoup plus compétitive avec l'algorithme Minimax et la rendre plus optimisée grâce à l'algorithme Alpha-Beta. L'ajout d'un timer aurait été envisageable, comme aux échecs avec un temps pour le premier joueur et un temps pour le second joueur : si l'un des deux timer descend à zéro c'est perdu. Pour finir, nous aurions pu implémenter un nombre de série de victoires ou de points en face à face.