BONNET Kloé (nom de commit : kbonnet001)
DONAT-BOUILLUD Lallie (nom de commit : LallieDB, Boursicotte)
Groupe 1

Projet d'introduction à la programmation

Document de justification techniques

Sommaire :

INTRODUCTION

Notre projet est un jeu du mémory qui possède 3 modes de jeu : le mode solo, le mode multijoueur et le mode contre l'ordinateur aussi appelé mode IA. Ce jeu du mémory a comme particularité que le joueur décide du nombre de cartes qui forme un n-uplet et du nombre de n-uplet présent dans le jeu.
Le mode solo permet de jouer seul au jeu du mémory, en jouant avec un nombre de coups limité ou pas (avec une limite de 100 coups limités).
Le mode multijoueur permet de jouer jusqu'à 10 les uns contre les autres.
Le mode IA comporte différents niveaux et permet de jouer contre l'ordinateur modélisé par le chat "Patate" qui a pour caractéristiques :
- Au niveau 1 : de choisir toutes ses cartes aléatoirement (aucune mémoire)
- Au niveau 2 : de choisir ses cartes en prenant en compte les dernières actions du joueur (petite mémoire)
- Au niveau 3 : de choisir ses cartes de manière intelligente (très bonne mémoire, se rappelle de toutes les actions)

LANCER LE JEU

Pour lancer le jeu, il faut cloner le lien de notre projet Github, puis lancer le fichier cloné sur VS code et écrire "dotnet run" dans le terminal. Le jeu se lancera alors et vous demandera avec quel mode de jeu vous souhaitez jouer.

LES FONCTIONS ET PROCÉDURES COMPOSANTS LE CODE

I. Les fonctions avant le début de la partie

Jeux
Cette procédure permet de lancer le jeu en demandant au joueur le mode de jeu (voir fonction choixModeDeJeu) puis le(s) nom(s) du/des joueur(s) (voir fonction DetermineNbJoueurs) et retourne le mode de jeu voulu par le joueur (voir fonction ModeSolo, NombreCoupsLimite, ModeIA, ModeMultijoueur)

->ChoixModeDeJeu
Cette fonction permet de renvoyer un entier qui représente le mode de jeu choisi par le joueur :
1=Mode solo
2=Mode Multi
3=Mode IA
Elle utilise la fonction VerifieSynthaxe pour savoir si le joueur tape bien un entier compris entre 1 et 3, et la fonction DemandeNomJoueur

-> VerifieSynthaxe
Cette fonction retourne un booléen false si la réponse n'est pas un entier qui se situe dans l'interval souhaité, retourne true sinon.
->DemanedNomJoueur()
Cette fonction demande au(x) joueur(s) son(leur) nom(s)

->DetermineNbJoueurs
Cette fonction affiche le mode de jeu choisi et retourne un tableau de taille égale au nombre de joueur (1 pour le mode solo, 2 pour le mode IA et demande
aux joueurs pour le mode multi)
Elle utilise la fonction VerifieSyntaxe

->NombreCoupsLimite
Cette fonction retourne un entier qui est le nombre de coups limites souhaité par le joueur. Si le joueur ne veut pas un nombre de coups limites, elle retourne 0

II. Les différents mode de jeu

ModeSolo

Le mode solo prend en argument le nombre de coups limite (0 si le joueur ne souhaite pas en avoir), le nom du joueur et lance la partie de mémory pour un joueur seul.
Elle se compose d'une phase de création et d'initialisation des paramètres, puis d'une boucle qui s'arrête quand le joueur retourne toutes les cartes ou que son nombre de coups limite est épuisé, elle-même composée d'une phase de réinitialisation partielles de certains paramètres et d'une boucle qui demande à l'utilisateur de retourner des cartes tant qu'il ne se trompe pas, la dernière phase du programme permet d'afficher un message de félicitation au joueur ou d'échec dans le cas où il ne réussit pas à retourner toutes les cartes avant le nombre de coups limites.
Elle utilise comme fonction :
DemanderCartesIndent, DemanderNbUplet, CreerTableauInitial, NewPlateauSolution, VerifieCartes, AfficherTableau, DemandePositionCartes, RetourneCarte, PaireRetourne, Felicitations

ModeMulti

Le mode multi prend en argument les noms des joueurs et lance la partie de mémory pour plusieurs joueurs. Elle se compose d'une phase de création et d'initialisation des paramètres, puis d'une boucle qui s'arrête quand toutes les cartes sont retournées, composée d'une boucle for qui permet aux joueurs de jouer les uns après les autres, elle-même composée d'une phase de réinitialisation partielles de certains paramètres et d'une boucle qui demande à l'utilisateur de retourner des cartes tant qu'il a raison. La dernière phase du programme étant d'afficher un message de félicitation au joueur(s) qui a/ont gagné(s).
Elle utilise comme fonction :
DemanderCartesIndent, DemanderNbUplet, CreerTableauInitial, NewPlateauSolution, VerifieCartes, AfficherTableau, DemandePositionCartes, RetourneCarte, PaireRetourne, Felicitations, Egalite, DeterminationGagnantEtMessage .

ModeIA

Le mode IA prend en argument le nom du joueur et lance la partie de mémory.
Elle se compose d'une phase de création et d'initialisation des paramètres, puis d'une boucle qui s'arrête quand toutes les cartes sont retournées, qui est composée d'une boucle for qui permet alternativement au joueur et à l'ordinateur de jouer, elle-même composée d'une phase de réinitialisation partielles de

certains paramètres et d'une boucle qui demande à l'utilisateur ou à l'IA de retourner des cartes tant qu'il a raison, la dernière phase du programme
étant d'afficher un message de félicitation ou de défaite au joueur.
Elle utilise comme fonction :
DemanderCartesIndent, DemanderNbUplet, CreerTableauInitial, NewPlateauSolution, VerifieCartes, AfficherTableau, DemandePositionCartes, CartesPatate, RetourneCarte, PaireRetourne, Felicitations, Egalite,DeterminationGagnantEtMessage

III. L'initialisation des grilles utilisées pour le mémory

DemandeCarteIdent
Cette fonction demande le nombre de nuplet (=nombre de cartes identiques) avec lesquel que le joueur souhaite jouer

DemandeNbUplet
Cette fonction demande le nombre de cartes que compose un n-uplet au joueur. 1=un singulet, 2= une paire, 3= un triplet,...

OptimisationPlateau
Cette fonction permet d'optimiser le plateau de jeu du mémory en retournant si possible un plateau carré, sinon un plateau rectangulaire avec le plus petit écart
possible entre la largeur et la longueur

CreerTableauInitial
Cette fonction permet de retourner un tableau optimisé remplis uniquement de d'astérisques. Elle utilise la fonction OptimisationPlateau().

NewPlateauSolution
Cette fonction renvoie un tableau optimisé contenant le nombre de uplet et la longueur de n-uplet pris en argument. Les caractères possibles vont de 'a' à 'z',
de 'A' à 'Z' et de '0' à '9'. Les valeurs du tableau sont remplies aléatoirement. Elle utilise la fonction OptimisationPlateau.

AfficherTableau
Cette procédure permet d'afficher sur le terminal le tableau entré en argument

IV. Fonctions et procédures utilisées en cours de partie

RetourneCarte
Cette fonction permet de retourner la carte située à la position choisie par l'utilisateur dans le tableau donné en argument.

DemandePositionCartes
Cette fontion demande à l'utilisateur de choisir une carte qu'il souhaite retourner. Si cette carte est déjà retourner (ie si elle n'a pas la valeur d'un astérisque)
ou si les valeurs entrées ne correspondent pas à celle du tableau, on redemande à l'utilisateur de choisir une position de carte.

CartesPatates
Cette fonction permet au niveau 1 du mode IA de choisir la position d'une carte non retournée. La position de la carte est choisie aléatoirement.

VerifieCartes
Cette fonction renvoie un booléen de valeur true si le joueur qui a la main peut continuer de retourner des cartes ou si il a retourné un n-uplet.
Elle renvoie false si le joueur s'est trompé et a retourné une carte qui n'était pas identique à la précédente.

PaireRetourne
Cette fonction intervient quand un joueur retourne un n-uplet. Elle met à jour le tableau du joueur en ajoutant les valeurs du n-uplet retourné. Puis, si on ne
se situe pas dans le mode solo, elle affiche un message de félicitation au joueur ayant retourné le n-uplet et affiche son score. Enfin, elle vérifie si le tableau
du joueur est égal au tableau de solution (cas de fin de partie).
Tant qu'il reste des n-uplets à retourner elle affiche false. Dès que la fin de partie est atteinte elle retourne true.

V. Détermination du gagnant

RetournePosEtMax
Cette fonction retourne la position et la valeur de la valeur maximale d'un tableau. Elle permet de savoir quel joueur gagne la partie dans le mode multi et IA.
Cette fonction ne marche que dans le cas d'une seule valeur maximale, le cas d'égalité est précédemment traité dans la fonction Egalite.

Egalite
Cette fonction retourne un tableau de 10 avec une valeur de 1 à la postion des joueurs ayant un score égal et supérieur à tous les autres joueurs dans un
cas d'égalité. Sinon, il renvoie un tableau de 10 composé uniquement de 0. Cette fonction ne marche que pour un tableau d'entrée de dimension inférieure ou égale
à 10. Nous avons choisi 10 car c'est le nombre maximal de joueur acceptés dans le mode multijoueur (valeur fixée arbitrairement).

DeterminationGagnantEtMessage
Cette fonction utilise les deux fonctions précédentes (Egalite et RetournePosEtMax) pour déterminer à la fin de la partie si il y a un ou plusieurs gagnant et pour
l'/les identifié(s). Puis elle envoye un message de fin en fonction du résultat.
Elle n'apparait pas dans le mode solo

VI. Procédures d'affichage

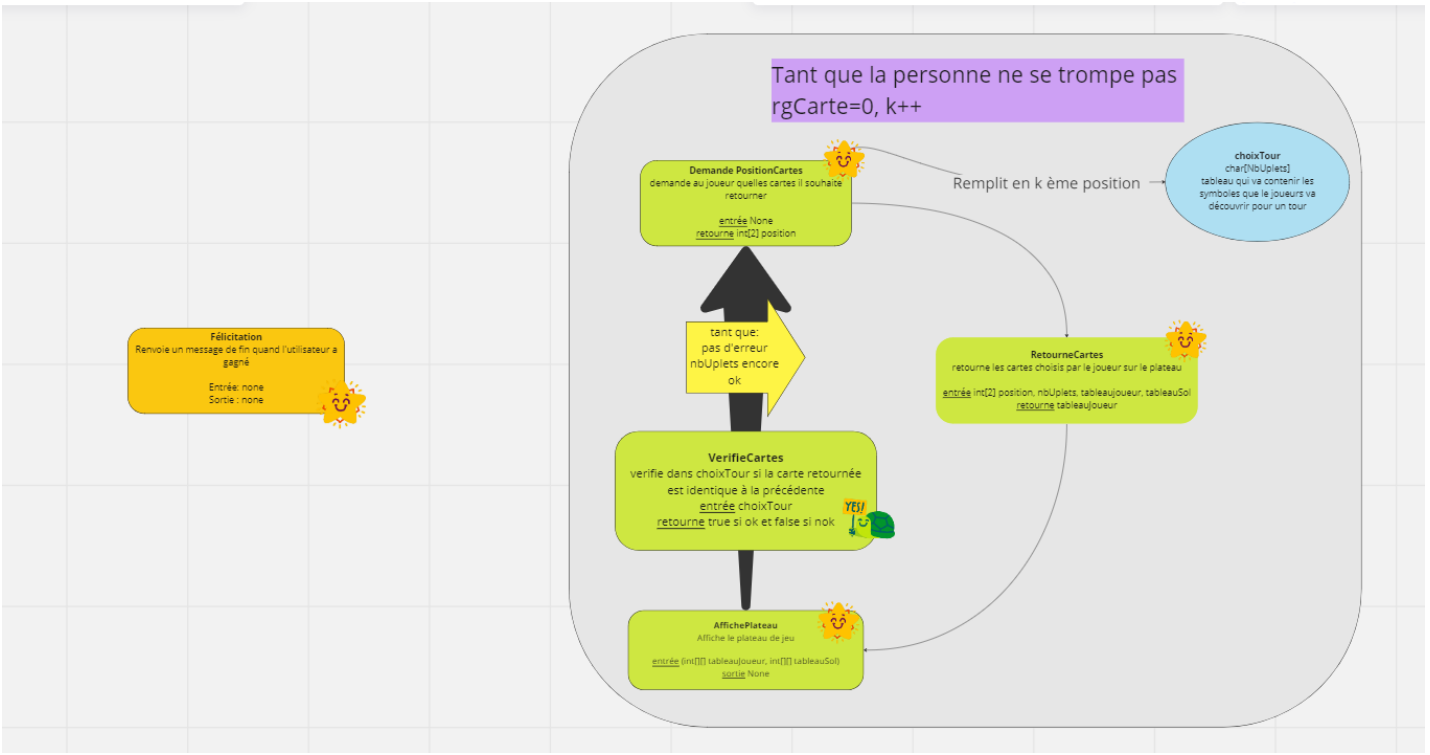
Felicitations
Cette procédure renvoie un message de félicitation au gagnant.

ORGANISATION DU TRAVAIL

Pour effectuer ce mémory, nous étions en binôme. Nous avons toutes les deux étudié en prépa avant (intégrée et CPGE) et notre niveau en informatique est semblable.

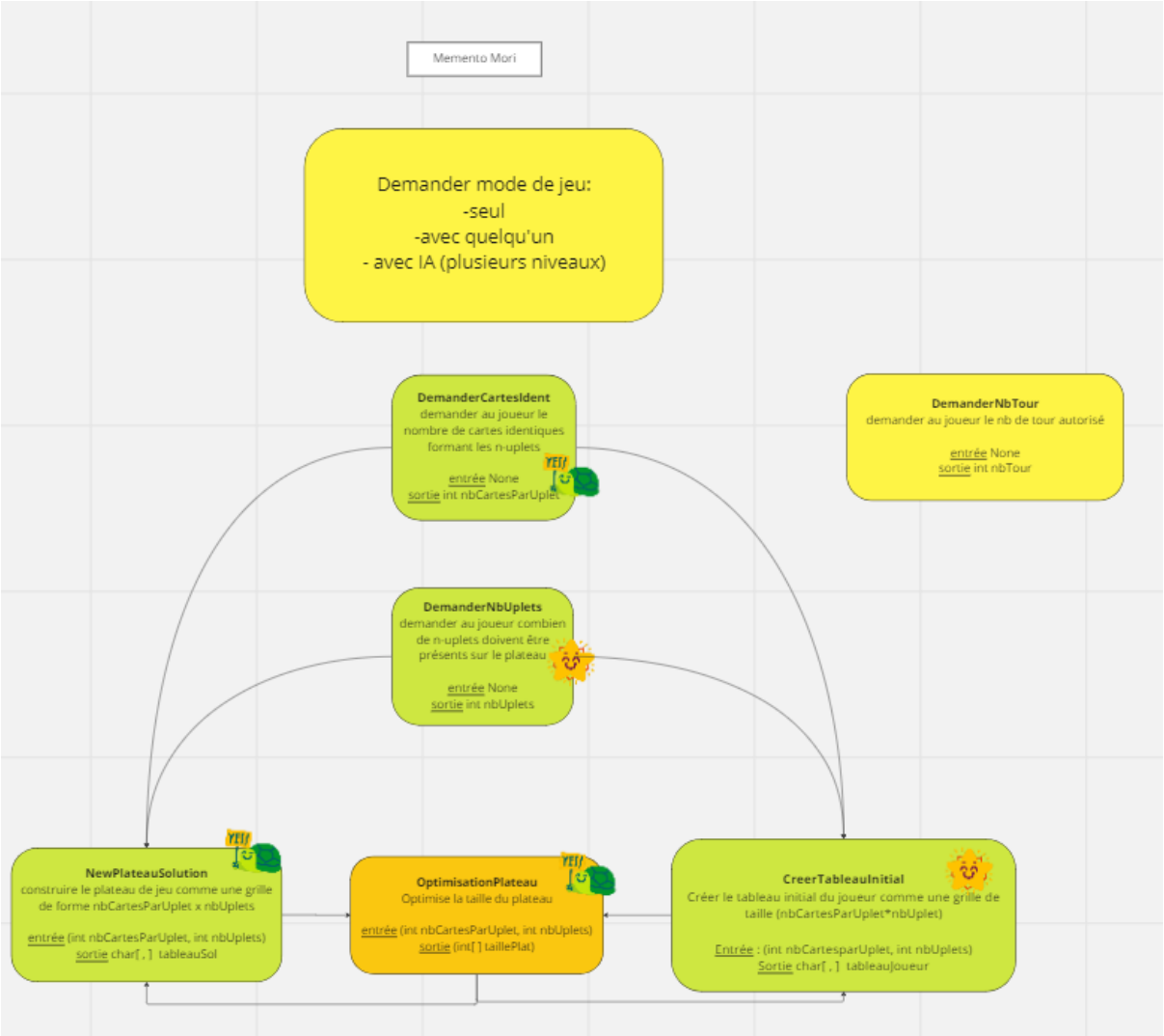
A la première semaine de projet, nous avons discuté des différentes fonctions dont allait avoir besoin notre programme. Une fois ces fonctions écrites, nous avons travaillé sur Miro pour se répartir les fonctions et voir rapidement ce qui était fait et ce qui restait à faire.

Voici une capture d'écran de ce que fait la boucle de notre programme :



La tortue représente l'avancée de Kloé et l'étoile représente celle de Lallie. Les carrés représentent les fonctions. Quand ils sont verts, ce sont les fonctions obligatoires alors quand jaune ce sont les fonctions facultatives.

Voici la capture de nos début de recherches :



Au niveau des branches, nous avons chacune mis nos modifications sur nos branches respectives (Kloé et Lallie). Puis nous avons eu d'autres branches pour les projets spécifiques tels que interface, IA pour le niveau 1 de l'IA et IA2 pour le niveau 2 et 3 que nous n'avons pas eu le temps de finir.

Dans cette dernière branche, le but était d'améliorer la mémoire de l'IA, plus précisément, de créer une tableau qui représente la mémoire de l'IA. Pour le niveau 2, l'IA aurait dû se rappeler seulement de la dernière action du joueur. Pour le niveau 3, l'IA aurait dû se rappeler de toutes les actions.

MiseAJourMemoireIA permet de mettre à jour la mémoire active de l'IA. Pour le niveau 2, cette fonction remplace l'ancienne mémoire par la nouvelle. Pour le niveau 3, la fonction fusionne l'ancienne mémoire avec la nouvelle.

VerifieMemoireVictoire est une fonction qui permet de vérifier de vérifier dans la mémoire de l'IA s'il existe une combinaison gagnante. Si oui, la fonction retourne un tableau contenant la

position des cartes à jouer pour gagner. Sinon, elle retourne un tableau rempli de positions [-1,-1].