

Rapport du projet Projet : Pickomino



Réalisé par :

Saadaoui Brahim

Années universitaire : 2021/2022

Encadré par : Mme Marie Pelleau



-Petite Intro au jeu

Jeu de dés faisant appel au calcul, à la chance et au sang froid! Les Pickominos sont posés sur la table en ordre croissant.

Chaque joueur à son tour jette les dés. Il choisit une valeur qui revient plusieurs fois et met les dés correspondants de côté. Il procède ainsi à chaque jet de dés jusqu'à ce qu'il décide d'arrêter. La somme totale des dés mis de côté permet de gagner le pickomino correspondant. Mais attention : si un jet de dé ne donne que des valeurs déjà retenues, le tour est terminé! Il est par ailleurs obligatoire d'avoir une série de vers de terre. Les Pickominos gagnés font marquer des points (1, 2, 3 ou 4) qui détermineront le gagnant en fin de partie.

Le nombre de joueurs : 2 – 7 avec un joueur autonome(ordinateur)

Contenu : 16 Pickominos (de 21 à 36), 8 dés.

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4


Première étape : le jeu (Explication de chaque fonctions)

Dans ce projet on a défini des structures qu'on va utiliser toutes au long du projet, sont définies comme suit :

On a utilisé 5 structures :

- **Element** prend en paramètre un entier nombre et le suivant si pour définir une pile.
- **Pile** on a défini une pile, son premier élément s'appelle premier.
- **Tdes** qui contient deux entiers : val pour la valeur du dé et somme pour la somme des dés de même valeur dans une lancée.
- **Joueurs** qui contient deux entiers : Sommeval pour la somme des dés que le joueur a déjà choisi et dernieval pour la dernière valeur du dé et une structure Pile.
- **Pickominos** qui contient un entier Valeur (valeur des) et la structure joueurs qu'on a déjà défini par le pointeur gagnant.

Après la définition des structures, on procède à l'explication des fonctions utilisées dans ce projet et on commence par la fonction :

 **initializer** : prend en paramètre un tableau t et un tableau de structure tdes ; et réinitialise t[0]=0 , t[1]=8 et t[2]=0 et remet à zéro toutes les sommes pour toutes les valeurs.

Exemple :

On a un tableau t : `int t[3]={14,4,0};`

Et un tableau struct tdes ValDG[6]=
`{{1,0},{2,4},{3,0},{4,0},{5,10},{6,0}};`

Si on met `initializer(t,ValDG);` la fonction est de type void mais change les valeurs du tableau t en {0,8,0} et ValDG en {{1,0},{2,0},{3,0},{4,0},{5,0},{6,0}}

Remarque :

Pour le tableau t :

t[0] représente la somme total des valeurs de dés gardées par le joueurs

t[1] le nombre des dés à lancer à chaque fois ,(la premier fois le joueurs lance 8 dés , et les autres fois lance que le reste des dés)

t[2] peut avoir deux valeurs possible, soit 0 ou 1 si le joueur a choisi la valeur v ,0 sinon.

 **Lancer** : elle prend en paramètre deux tableau d'entiers et un tableau de struct tdes

Au cours de premier lancer, la fonction **rand ()** nous donne des valeurs aléatoires entre 1 à 6, et on représente le chiffre 6 par v, avec v prend la valeur 5 et cela c'est le rôle de la première boucle de la fonction lancer.

Et à chaque fois que le joueur prend une valeur parmi les valeurs aléatoires, il lance de nouveau tant qu'il y'a des valeurs et les valeurs restant sont donner par la variable **po** dans la deuxième boucle, et le tour est raté si (**po <=0**) c'est-à-dire si le joueur a déjà choisi une valeur.

+Gagnant : la fonction gagnant prend en paramètre une structure joueur et un entier Nombrejoueurs,

Et détermine si un joueur est gagné en calculant la somme des valeurs de ses dés.

+Jouer : pour la fonction jouer, on commence par initialiser un tableau de pickominos , intituler tableaupicko qui initialise les 16 pickominos en donnant a chaque pickominos un nombre de point.

Après on demande à l'utilisateur d'entrer le nombre de joueurs, si le nombre de joueurs égale 2,

On demande à l'un des joueurs de commencer, en tapant le chiffre 1 sinon taper n'importe quelle touche.

Après nous avons fait une boucle tant que **re < 16** (cad tant qu'il nous reste des dés), on fait appel à la fonction initializer, et on affiche un message

"le joueurs **n** , c'est a toi de jouer " et le joueur il faut au moins qui lance une fois.

Ensuite on vérifie si on a encore des dés à disposition, et on demande au joueur de relancer l'un dés a chaque tour, ou s'arrêter, et pour lancer un dés il faut appuyer sur **O**. En fin la dernière boucle calcule la somme des dés lacer par le joueur, et donne le pickomino gagner,et si

(`tableaupicko[z].gagnant -> dernierval==z`) , le joueur a le droit de prendre le pickomino de son adversaire si le Pickomino est au sommet de la pile , le joueur le récupère et le place au sommet de sa pile. et le programme affiche le joueur gagnant et la valeur gagnée.

Deuxième étape : IA

Dans cette seconde partie, notre objectif est de construire une petite intelligence artificielle. Notre programme va donc devoir demander au joueur (humain) de lancer, sinon taper n'importe quelle touche, et le robot(ordinateur) commence le premier à lancer.

Pour cela on a besoin de deux fonctions `lancerRobo` et `Jouerseul`.

Pour la fonction `lancerRobo` c'est le même principe que la fonction `lancer` mais cette fois ci c'est le robot (l'ordinateur) qui lance les dés, `lancerRobo` au premier lancer la fonction `rand ()` donne des valeurs aléatoires, et à chaque fois le programme prend la valeur maximale des dés et affiche un message comme quoi il a choisi cette valeur, et lorsque l'IA a déjà choisi une valeur parmi les valeurs disponibles, dans ce cas le tour est raté.

Pour la fonction `jouerseul` elle permet à l'utilisateur de jouer tout seul avec le robot (ordinateur) en appuyant sur le chiffre 1, pour cela nous avons fait des conditions pour que l'IA réagit d'une manière intelligente.

La première des choses il doit choisir la valeur v en priorité, et après il choisit le dé et la multiplication de chaque même valeur de ce dé, et prend le max entre toutes ses valeurs des dés lancés.

Le robot lance tant qu'il le reste des dés et tant qu'il n'a pas atteint la valeur minimale des pickomino sur la table.

Troisième et dernière étape : un système de sauvegarde/restauration

À présent, durant la partie, un joueur peut enregistrer une sauvegarde de la partie courante au sein d'un fichier qui s'appelle «sauvegarde.txt». Les données sont sauvegardées sous forme de texte.