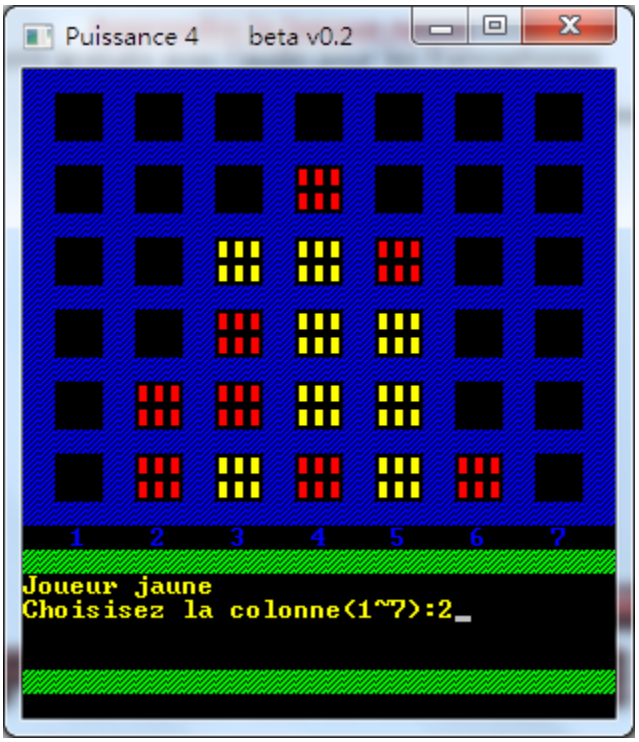
RAPPORT DE PROJET

Puissance 4



REALISATION D'UNE INTERACTION HUMAIN-ORDINATUER PAR LES JEUX 4 EN LIGNE

Par

Kewei HU
09125694

OCTOBRE 2011

Sommaires

1. Remercie 3
2. Introduction 3
3. Structures 3
4. Description 3
   4.1 Structures de données 3
      Le tableau à deux dimensions 4
      Les tableaux à trois dimensions 4
   4.2 Class 4
      Structures 4
      Héritage 4
      Constructeur et Destructeur 5
      D'autre fonctions 6
   4.3 La fonction principale 7
      Objet 7
      Interface 7
      Différentes interactions entre objets 8
   4.4 Intelligence artificielle 8
      1. Calculer des échecs 8
      2. Estimer le DEGREE 9
      3. Faire la réponse 10
5. Conclusion 10
6. Références 10

1. Remercie

Je remercie à Ting SHAO, Cheng Liang, Xueyan JIANG pour des guides d'affichage.

Je remercie aux professeurs de LO02 : Gildas Bayard, Claude Renaud, Guillaume Doyen pour votre guides dans la class. les fiches console.cpp et console.h m'ont donné des grandes aides.

Permettez-moi de vous présenter mes remerciements sincères.

2. Introduction

Puissance 4 est un jeu qui se joue à deux. Le but du jeu est d'aligner le premier 4 pions de sa propre couleur. Les 4 pions peuvent être alignés horizontalement, verticalement ou encore en diagonale.

Dans ma programme, le taille de plateau est 7 fois 6, c'est à dire il y a 7 colonnes, chaque colonne contient 6 pions, on ne peut placer le pion que de bas en haut.

Les principales du programme sont :

1. gérer le plateau de jeu et son affichage.

2. savoir dans quelle colonne il est possible d'ajouter un pion (colonne non pleine).

3. savoir dans quelle colonne il est le plus possible d'ajouter un pion pour gagner ou défendre.

4. pouvoir ajouter un pion jaune ou rouge dans une colonne donnée.

5. déterminer si la partie est et qui a gagné ou si c'est match nul, etc.

Pour abstraire le plateau, on utilise un tableau à deux dimensions pour simuler le plateau, chaque membre du tableau représente de chaque pion du plateau, le data de membre représente de la couleur de pion, c'est-à-dire : 1 représente de couleur jaune, 2 représente de couleur rouge, en particulièrement, 0 représente un pion vide.

3. Structures

Mon programme compose cinq fiches :

main.cpp, jeux.cpp, console.cpp, jeux.h, console.h

console.cpp et console.h sont les fiches donne par professeur, je ne les modifie rien.

Jeux.h et jeux.cpp sont des fiches qui compose de class. Dans ces fiches, je déclare et définis des méthodes et des attributs de la class.

main.cpp compose la fonction principale du programme, il détermine l'interface et l'opération du match.

4. Description

4.1 Structures de données

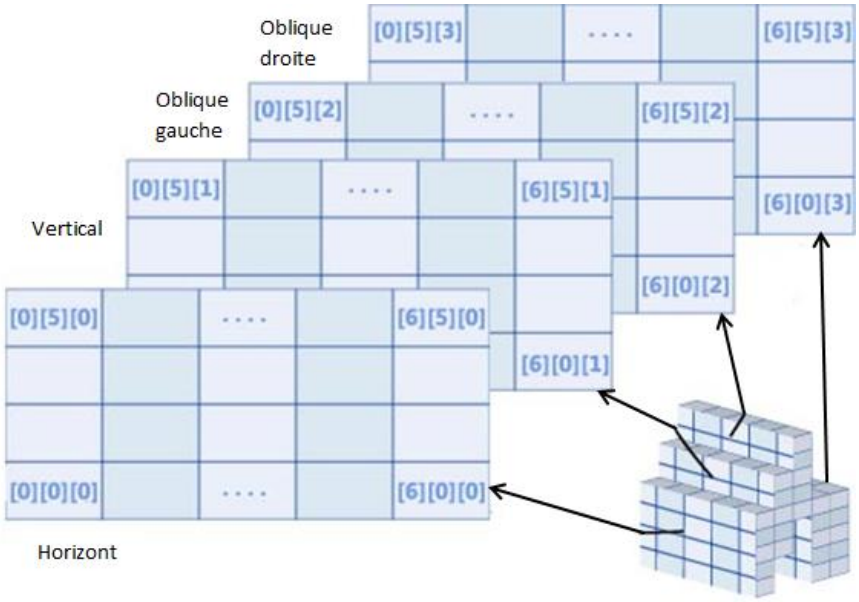
Des données sont dans des tableaux allousés dynamiquement, il y a trois tableaux, un tableau de à deux dimensions, et deux tableaux à trois dimensions. On crée les tableaux et initiale des attributs dans le constructeur de la class, et on détruire les tableaux par le destructeur.

Le tableau à deux dimensions

Elle gère le plateau de jeu, l'indice de la tableau représente de la location de le pion, le data de le membre représente de la couleur de pion, on définit '1' représente de la couleur jaune et '2' représente de la couleur rouge, c'est-à-dire si le membre de la tableau est t[i][j]==1,elle représente un pion dans le colonnei+1 avec un hauteur de j, le couleur de le pion est jaune. la colonne portée de 1 à7, le hauteur portée de 0 à 6,'0' représente de colonne vide, et '6' représente de colonne plein.

Les tableaux à trois dimensions

Elles gèrent de situations des pions de chaque joueur. Les deux premier dimensions représente de la location de pion. La troisième dimension gère la situation, c'est-à-dire les l'opportunité ou risques d'un pion vide sur quatre direction : horizontal, verticale, oblique gauche et oblique droite. Donc, il y a quatre membres de la troisième dimension, chaque membre gère des risques ou opportunité d'une direction. On nomme ce paramètre à<DEGREE>.



4.2 Class

Structures

Il y a deux classes dans le programme : jeux, console.

Class console

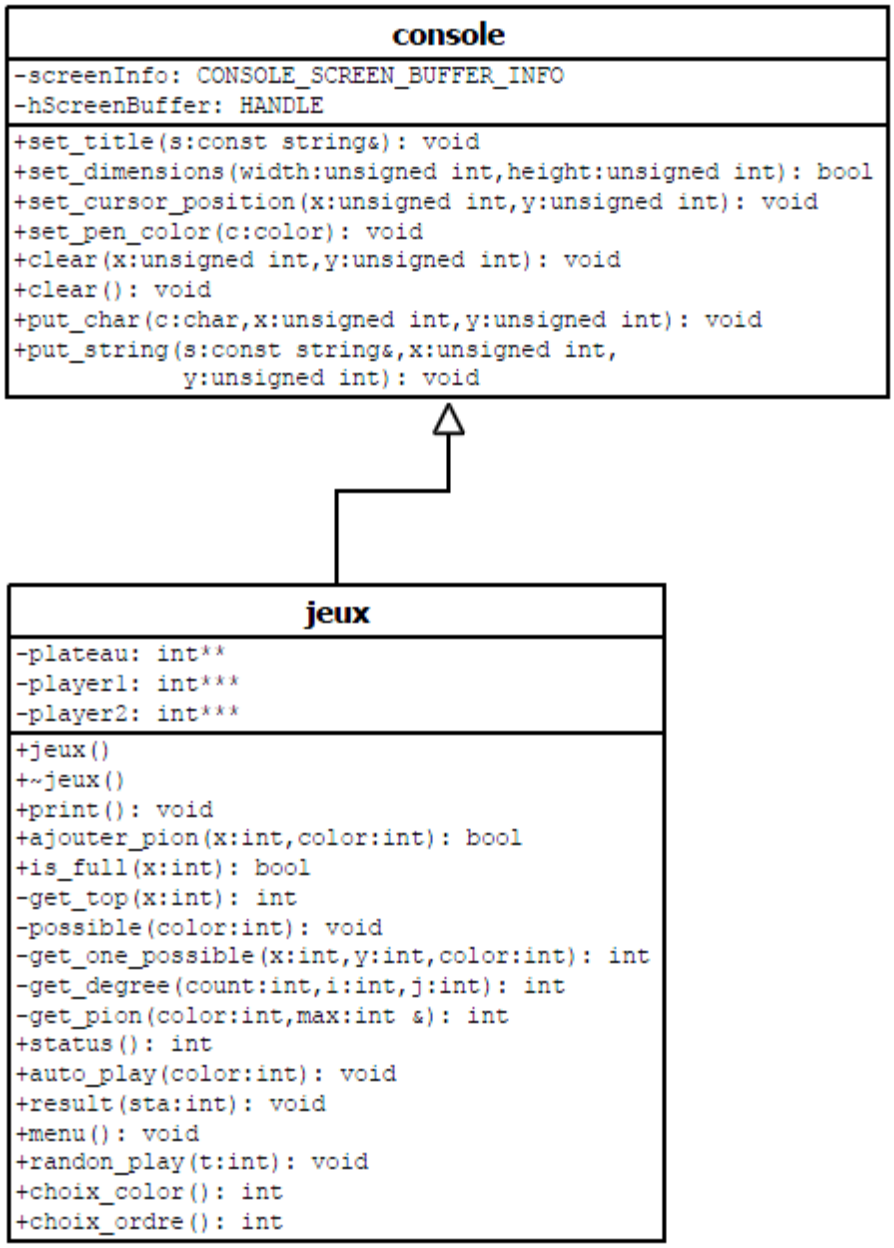
Cette class est donnée par le professeur, elle principalement contrôle d'afficher de programme.

Class jeux

Cette class principalement responsable de gérer des pions et des méthodes pour ajouter, calculer, estimer des pions.

Héritage

La class <jeux> publiquement héritage la class <console>.



Constructeur et Destructeur

Constructeur

Le constructeur de la class jeux crée dynamiquement trois tableaux. Les descriptions précises sont ci-dessus :

Nom : jeux::jeux()

Fonction : crée trois tableaux : un tableau à deux dimensions, deux tableaux à trois dimensions. Elles sont créées par pointeurs, pour créer un tableau à deux dimensions, il faut créer un pointeur :<type **p> ; pour créer un tableau à trois dimensions, il faut créer un pointeur :<type ***p>, ensuite, on utilise multi cycle pour créer les tableaux.

Destructeur

Le destructeur détruit des espaces qui sont occupés par des tableaux alloués dynamiquement, Les descriptions précises sont ci-dessus :

Nom : jeux::~jeux()

Fonction : on utilise la méthode <delete> pour récupérer des espaces mémoire allouées dynamiquement. Car il y a des tableaux à plusieurs dimensions, on utilise multi cycle pour récupérer tous les mémoires.

D'autre fonctions

La class jeux contient quinze méthodes dont cinq sont privés, maintenant, on introduit sept methodes.

Jeux:is_full
Retour: bool (TRUE ou FALSE)

Paramètre : x :int ;

Fonction: elle vérifie si la colonne dative est pleine, si oui, elle retour TRUE.

Jeux ::get_top
Retour: int(0…6)

Paramètre : x :int

Fonction : elle retour le haut de la colonne datif(x~1…7), s'il est vide, elle retour 0.

Jeux :: ajouter_pion
Retour : bool

Paramètres : x :int,color :int

Fonction : on ajoute un pion dans la colonne datif(x~1…7), et la couleur de pion est le type int : 1 représente de jaune, 2 représente de rouge, 3 représente vide. D'abord, on utilise la méthode <is_full> pour vérifier si la colonne dative est pleine, on n'ajoute un pion que la colonne n'est pas plein. Ensuite, on utilise la méthode < get_top > pour obtenir le haut du colonne, après, on changer le date de membre de 0 à <color> pour représente que ce pion est occupé par le pion avec couleur <color>, enfin, on vide le paramètre <DEGREE> parce que c'est nul pour un pion occupé.

Jeux ::status()
Retour : int

Paramètre : nul

Fonction : elle juge la fin du jeu， si le jaune a gagné, elle retour 1, si le rouge a gagné, elle retour 2, si le match est nul, retour 3, si le match est en progrès, elle retour 0

Jeux ::result()
Retour :void

Paramètre :sta :int

Fonction : elle imprime le résultat de match.le paramètre <sta> est transmis par la méthode <status>.

Jeux :print()
Retour : void

Paramètre :void

Fonction : elle crée l'Interface graphique pour l'utilisateur. La fonction principale est ci-dessous:

1. Imprimer un rectangle avec couleur bleue qui représente des bornes de plateau.

2. Traverser horizontalement et de haut en bas sur la tableau <plateau[i][j] >.si le date est 1, elle imprime un pion jaune, s'il est 2, elle imprime un pion rouge, s'il est 0, elle imprime un vide.

3. refaire l'étape 1 et 2 jusqu'a le dernière row du plateau.
4. refaire l'étape 1.

Jeux::auto_play(int color)

Retour: void

Paramètre: color : int

Fonction: cette méthode utilise d'autre quatre méthode pour jouer automatiquement avec intelligents. Elle ajoute un pion de couleur <color> automatiquement.

Jeux ::randon_play()
Retour :void

Paramètre : m : int

Fonction : elle ajoute des pions de deux couleurs alternativement. Le paramètre <m> représente de l'étape de ajouter par hasard. Par exemple, si m est 1, elle ajouter un pion jaune et un pion rouge par hasard. Si m est 2, elle ajouter un pion jaune, un pion rouge, un pion jaune et un pion rouge par hasard.

4.3 La fonction principale

Objet

Game

Il gère les pions de plateau, les fonctions pour jouer et la méthode pour afficher.

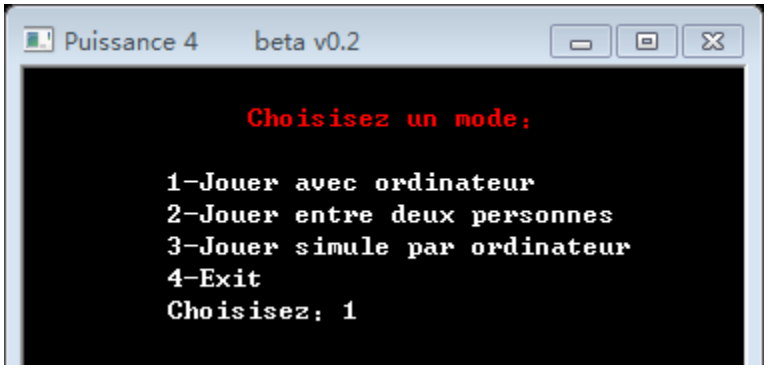
Screen

Il contrôle le titre de la barre de la fenêtre de la console, les dimensions de la console et la couleur de l'écriture

Interface

Menu

Dans cette fenêtre, on peut choisir le mode du jeu :



-1 joueur humain et 1 joueur simulé par l'ordinateur

-2 joueurs humains

-2 joueurs simulés par l'ordinateur

Interface de jouer

Il y a deux parties de la fenêtre : zone du plateau et zone de manier.

Dans la zone du plateau, on peut voir directement la situation du match.

Dans la zone de manier, on peut voir les informations : qui doive ajouter un pion, quel colonne on peut ajouter le pion, etc.

Opération

L'opération du programme est très simple.

Jouer avec ordinateur :

D'abord, vous devez choisir la couleur de votre pion, ensuite, vous devez choisir si vous voulez jouer avant de l'ordinateur ou pas, après, vous pouvez jouer avec l'ordinateur, vous devez taper le numéro de la colonne.

Jouer entre deux personnes :

C'est comme jouer avec l'ordinateur, les deux joueurs ajoute son pion alternativement.

Jouer simule par ordinateur :

Si vous choisissez ce mode, le résultat de match peut imprimer sur l'écran tout de suite.

Différentes interactions entre objets

D'abord on faire instance de un objet <screen> pour contrôler la taille et la titre de la fenêtre. Il aussi contrôle des affichages de menu dans le programme.

Ensuite, on demande l'utilisateur pour choisir un mode de jeu.

Puis, on faire instance d'autre objet <game> pour gérer le plateau et contrôler le processus du jeu. Apres chaque action d'ajouter un pion, on renouvèle l'écran et on calculer le DEGREE de tous les pions et on juge si le match est fini, si non, on continue le processus, si oui, on interrompe le processus et imprime le résultat du match.

Enfin, on retourne à menu pour demander le choix.

4.4 Intelligence artificielle

Dans ce partie, on parle des méthodes pour jouer automatiquement avec intelligents. Les méthodes sont classifiées pour trois étapes :

1. Calculer des échecs

Il y a deux méthodes pour cette étape : < void possible (int color)> et <int get_one_possible(int x,int y,int color)>

get_one_possible

Retour: int

Paramètre: x,y,color:int

Fonction : Elle calculer le nombre de pion en même couleur en ligne sur quatre directions : horizontal, verticale, oblique gauche et oblique droite. Pour chaque direction, elle appelle la méthode < get_degree>

pour estimer le DEGREE sur tout direction et les écrire dans les tableaux à trois dimensions, c'est-à-dire les tableaux<player[i][j][k]>.Elle retour le Maximin de pion en même couleur et en ligne.

possible (int color)
Retour: void

Paramètre: color:int

Fonction:elle utilise la methode<get_one_possible> pour calculer le DEGREE de tous les pions vides.

2. Estimer le DEGREE

Dans cette étape, elle contient une méthode :< int get_degree(int a, int b,int i,int j)>

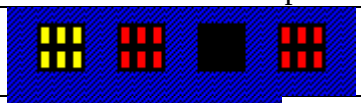
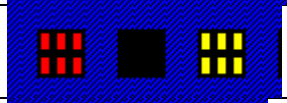
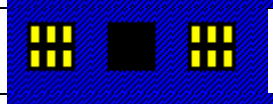
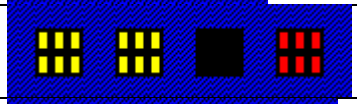
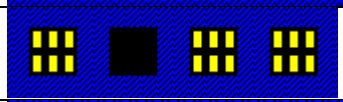
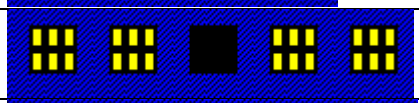
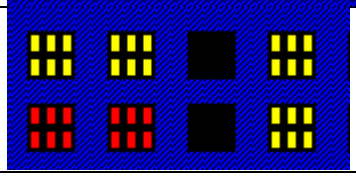
get_degree
Retour : int

Paramètre: a,b,x,y :int

Fonction : elle estime le DEGREE du pion dans le plateau[x][y] par les paramètres a et b qui sont transmis par la méthode < get_one_possible>.a et b représentes des nombre de pion en même couleur et en ligne à gauche et à droite pour horizontal (Au-dessus et au-dessous pour vertical) du pion datif. Ci-dessous est le tableau d'algorithmes :

<P> représente de la possibilité d'ajouter un pion d'ici. (<X> représente Oui, <O> représente Non) <T> représente Quelle que valeur soit là

On suppose d'être la couleur Jaune.

a	b	P	Exemple	DEGREE
0	0	X		0
1	0	X		1
1	1	X		20
2	0	X		5
2	1	X		100
2	2	X		
3	T	X		
T	T	O		0

3. Faire la réponse

On utilise la méthode <int get_pion (int color, int & max)> pour faire la réponse.

Get_pion

Retour : int

Paramètre : color, &max : int

Fonction : cette méthode ajoute un pion dans le colonne sur le priorité ci-dessous :

Priorité	DEGREE	Couleur	Réponse
4	0<D≤100	Soi	Défendre : Ajouter le pion tout de suite
3	0<D≤100	Moi	Attaquer : Ajouter le pion tout de suite
2	≥100	Soi	Défendre : Ajouter le pion tout de suite
1		Moi	Attaquer : Ajouter le pion tout de suite

Elle retour le numéro de la colonne et le paramètre <&max> est affecté du DEGREE de le pion datif.

5. Conclusion

Nous avons réalisé une programme pour le jeu : puissance 4 qui permet d'utilisateur à joueur avec l'ordinateur ou avec d'autre humaine. Il y a trois problèmes principales : afficher le plateau, ajouter un pion et jouer par l'ordinateur avec intelligent.

Pour afficher le plateau, on utilise les méthodes dans la class <console> pour changer le cursor, changer la couleur de caractères, et utilise deux caractères dans la expansion de ASCII (caractère de numéro 178 et 254), c'est très pratique pour le affichage. Dans la première version de la programme, on utilise les caractères <▒> pour imprimer les bornes et <■> pour imprimer des pions, mais il est un problème de l'affichage et il est aussi des problèmes dans le OS la langue étrangères, donc on utilise enfin des ASCII pour afficher le plateau.

Pour ajouter un pion, c'est moins difficile, pour ajouter un pion, il y a trois conditions : d'abord, la colonne existe, ensuite la colonne n'est pas plein, enfin il y a déjà un pion dessous ou c'est le premier pion dans la colonne.

Pour jouer avec l'ordinateur en intelligent, on créer deux tableaux pour gérer des DEGREE pour tous les pions, l'ordinateur va ajouter des pions par la calcule de ceux tableaux.

Dans le programme, on utilise principalement de programmation oriente objet, c'est la première programme pour moi de créer des class, définir des méthodes et décider des algorithmes pour artificiel intelligent. On comprend comment créer un class, faire l'héritage, définir le constructeur et destructeur, utiliser l'allocation dynamique, etc. c'est très pratique et plus intéressant que les devoirs on a déjà fait en classe de TD.

6. Références

[W0] http://wenku.baidu.com/

[W1]http://www.google.com/

[W2] http://www.stratozor.com/puissance-4/puissance-4-classique.php