2020

Jeu du Quarto en Python

Programmation2

M.SCHLICK

Camille Billouard

Elise Ibanez

03/01/2020

L3 MIASHS

Présentation du projet

- Pour la validation de l'UE de programmation les étudiants sont amenés à réaliser un projet en binôme en plus de l'examen final. Ce projet minimaliste a pour but de mieux appréhender la mise en place de projet informatique. Le jeu du Quarto à était choisi comme sujet.

Principe du Quarto

- Le jeu du Quarto ce joue avec deux joueurs. Le jeu est composé d'un plateau de jeu de 16 cases et une pioche de 16 pièces avec pour chaque pièce 4 attributs différents (Clair, Foncé, Grand, Petit, Avec ou sans trou). Le but est d'aligner, ligne, colonne, diagonale ou en carré, 4 pièces avec au moins un attribut en commun. Le premier joueur réaliser le quarto gagne la partie.
- Choix des technologies :
    - Pygame pour la gestion des « bruitages »
    - Tkinter pour l'interface graphique
    - Sqlite pour la gestion des données utilisateurs

Chronologie

Version	Date	Nom	Description
1.0	14/11/2019	Camille Elise	❖ Répartition des missions
1.1	15/11/2019	Camille	❖ Première version du noyau
1.2	26/11/2019	Camille	❖ Corrections des bugs ❖ Première version sans IHM
2.0	02/12/2019	Elise	❖ Première version avec IHM
2.1	20/12/2019	Elise Camille	❖ Deuxième version avec plusieurs onglets ❖ Mise en place de la base de données de gestion utilisateur (joueur)
2.2	27/12/2019	Elise Camille	❖ Première version finale du jeu ❖ Uniformisation des formats
3.0 - Finale	03/01/2020	Camille Elise	❖ Version finale du Quarto

Fonctionnement de l'appli

- Processus du jeu : utilisateur

  Le programme est lancé

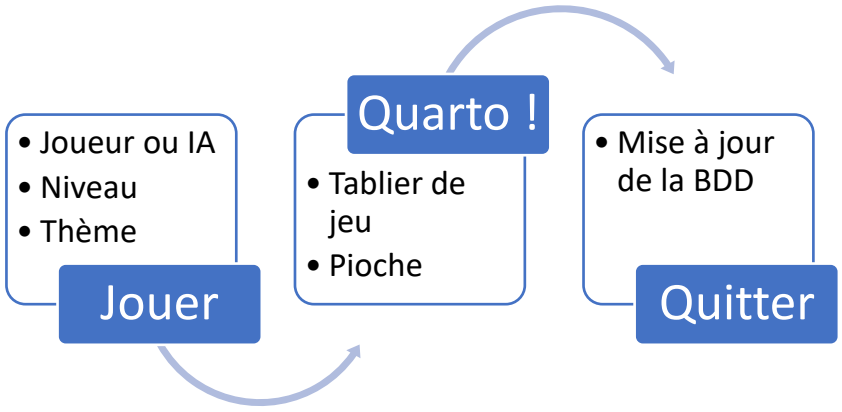
  L'utilisateur renseigne le pseudo des deux joueurs ou s'il souhaite jouer contre la machine il renseigne le pseudo « IA » dans la partie Joueur 1.

  Le programme vérifie l'existence des joueurs dans la base de données (BDD)

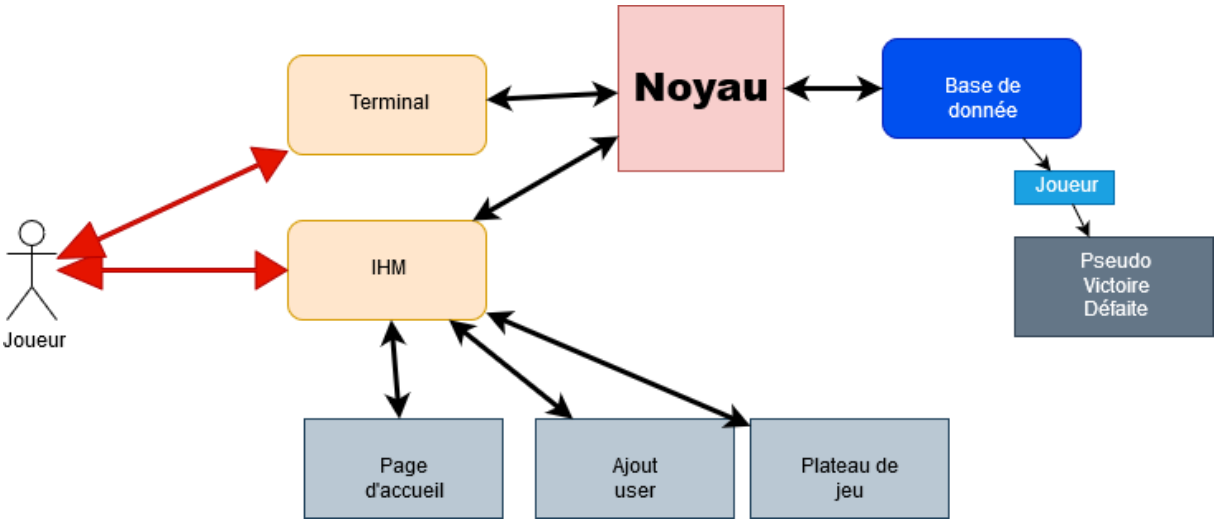
  - Si le joueur existe on lance la fenêtre de jeux
  - Sinon on affiche la fenêtre permettant d'ajouter un joueur à la BDD

  Le jeu est lancé.

- Processus du jeu : noyau
  - A chaque tour le joueur choisis une pièce puis l'adversaire choisis la case et une pièce pour le joueur 1.
  - Au bout de 3 tours on vérifie si un quarto à lieu.



Architecture



Modèle de données

Le programme permet une gestion des utilisateurs avec une base de données : Sqlite. Cette BDD permet d'enregistrer, supprimer et afficher les utilisateurs par ordre de victoire.

Limites & ouvertures

Limites

Pour le moment le programme ne propose pas une « IA » au sens propre du terme. C'est un choix aléatoire des pièces et des cases qui est fait pour chaque tour par l'ordinateur.

La partie « graphique » sera aussi à revoir pour rendre le jeu plus attractif sur les fenêtres de choix d'utilisateur, de niveau et de thème.

Lorsqu'une partie est finie, on ne peut pas relancer une nouvelle partie sans fermer le jeu. Et on ne peut pas annuler une partie en cours.

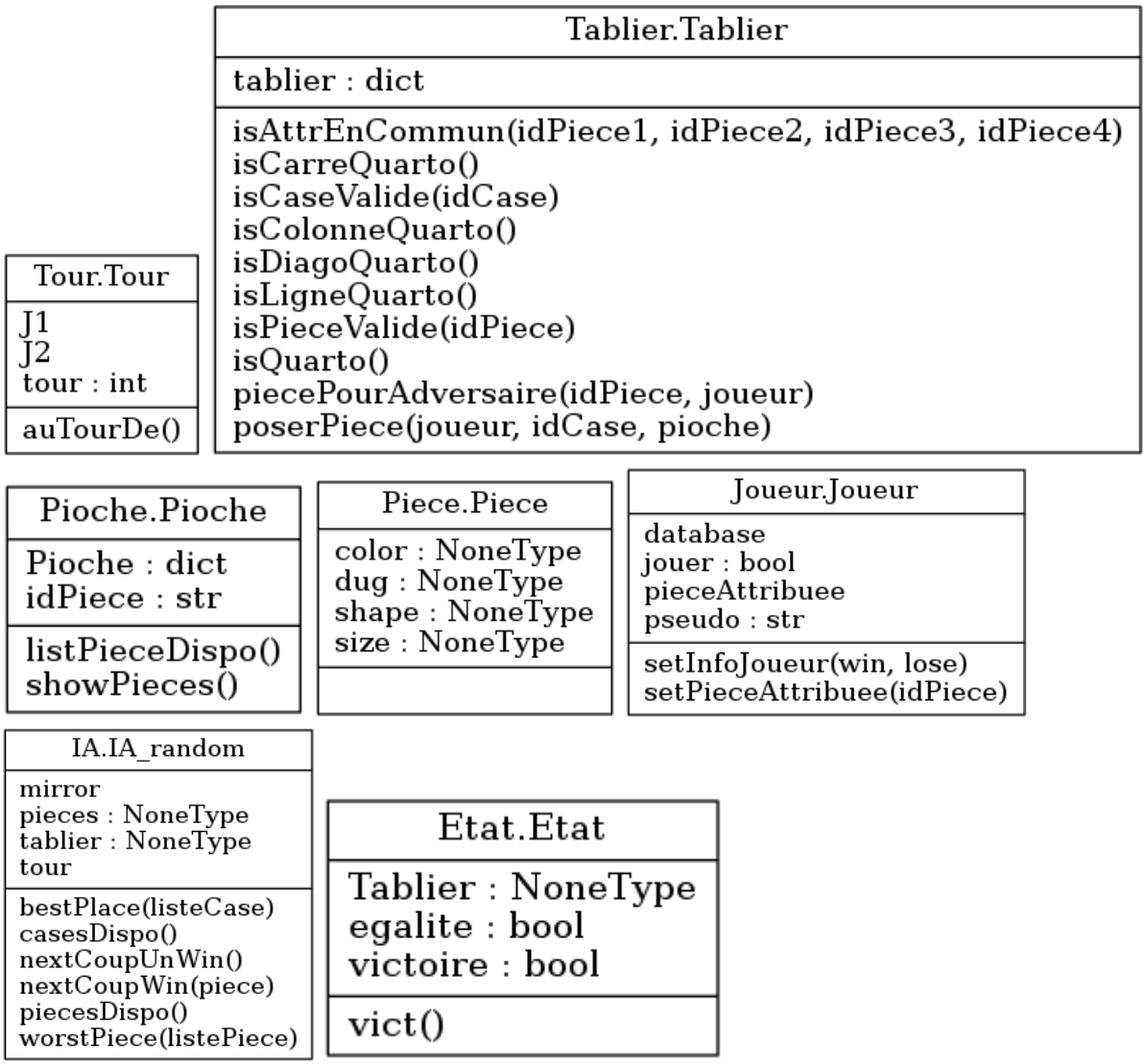
Ouvertures

Il serait envisageable de mettre en place un système de jeux à distance avec une architecture client-serveur. De plus avec les données enregistrées dans la BDD à chaque partie pourrait permettre de réaliser des statistiques sur les joueur et ainsi personnaliser l' « IA » en fonction des joueurs.

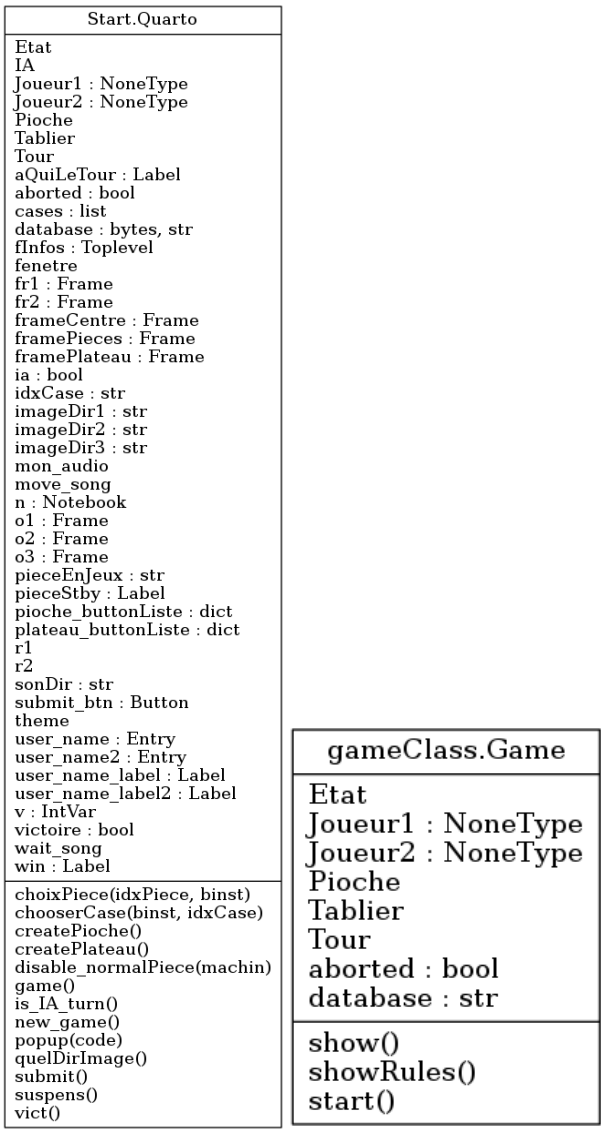
Annexe

Diagrammes de classes

Module Model :



Module View :


Start.Quarto
Etat
IA
Joueur1 : NoneType
Joueur2 : NoneType
Pioche
Tablier
Tour
aQuiLeTour : Label
aborted : bool
cases : list
database : bytes, str
fInfos : Toplevel
fenetre
fr1 : Frame
fr2 : Frame
frameCentre : Frame
framePieces : Frame
framePlateau : Frame
ia : bool
idxCase : str
imageDir1 : str
imageDir2 : str
imageDir3 : str
mon_audio
move_song
n : Notebook
o1 : Frame
o2 : Frame
o3 : Frame
pieceEnJeux : str
pieceStby : Label
pioche_buttonListe : dict
plateau_buttonListe : dict
r1
r2
sonDir : str
submit_btn : Button
theme
user_name : Entry
user_name2 : Entry
user_name_label : Label
user_name_label2 : Label
v : IntVar
victoire : bool
wait_song
win : Label
choixPiece(idxPiece, binst)
chooserCase(binst, idxCase)
createPioche()
createPlateau()
disable_normalPiece(machin)
game()
is_IA_turn()
new_game()
popup(code)
quelDirImage()
submit()
suspens()
vict()
gameClass.Game
Etat
Joueur1 : NoneType
Joueur2 : NoneType
Pioche
Tablier
Tour
aborted : bool
database : str
show()
showRules()
start()