Réalisation d'un jeu d'échec

SALL Amadou BRAZOUSKAYA Darya ARNAUD Alexia GUNTZ Thomas

21 novembre 2014

Cahiers des charges

Description de l'application

Nous nous proposons de réaliser dans ce projet un jeu d'échec en Java. L'affichage du plateau sera réalisé en 2D (affichage sur la console) et on ne prendra pas en compte la possibilité de jouer contre l'IA.

Le jeu est piloté par un interface graphique qui permet de paramétrer les options suivantes :

mode : Par défaut le mode classique est activé mais on peut choisir le mode TODO

timer : Par défaut la limitation du temps est désactivé. S'il est activé le joueur actif a un à temps limité pour jouer sinon il perd son tour.

limitation du nombre de coups : Par défaut il n'y a pas de limite sur le nombre de coups jouables.

Lorsque que l'utilisateur lance une partie. Le système demande d'entrer le nom des deux joueurs concernés.

L'application permet la sauvegarde d'une partie et sa reprise à l'aide d'une fenêtre de login.

Fenêtre de jeu

On se propose un affichage 2D du plateau et des pièces de chacun des deux joueurs.

Lorsqu'une pièce est perdue, elle est affichée sur le côté.

Règles du jeu

Le $Joueur_1$ commence, c'est son tour, il est considéré comme "actif". Il doit sélectionner une pièce pour la déplacer.

Lorsque la pièce est selectionné, il doit choisir une case valide pour s'y déplacer. Une case valide est une case accessible par la pièce.

En fonction du type de la pièce une case est accessible ou pas. une case vide ou une case avec une pièce adverse.

Le joueur "actif" ne peut pas se déplacer sur une case contenant une de ses propres pièces.

Si le joueur déplace sa pièce sur une pièce adverse (sur une case valide), il l'a détruit.

A la fin de chaque tour, on vérifie si un des deux rois est mis en échec. Si le roi du joueur actif est en échec, il doit déplacer une de ses pièces pour rendre son roi hors échec. Un joueur ne peut pas finir son tour si son roi est en échec.

A la fin d'un tour, le joueur dont le roi est mis en échec et mat a perdu.

Fenêtre de fin de partie

En fin de partie, une fenêtre récapitulant les scores s'affichent. Cette fenêtre contient le nom du vainqueur et le nombre de coups joué.

Documents d'analyse

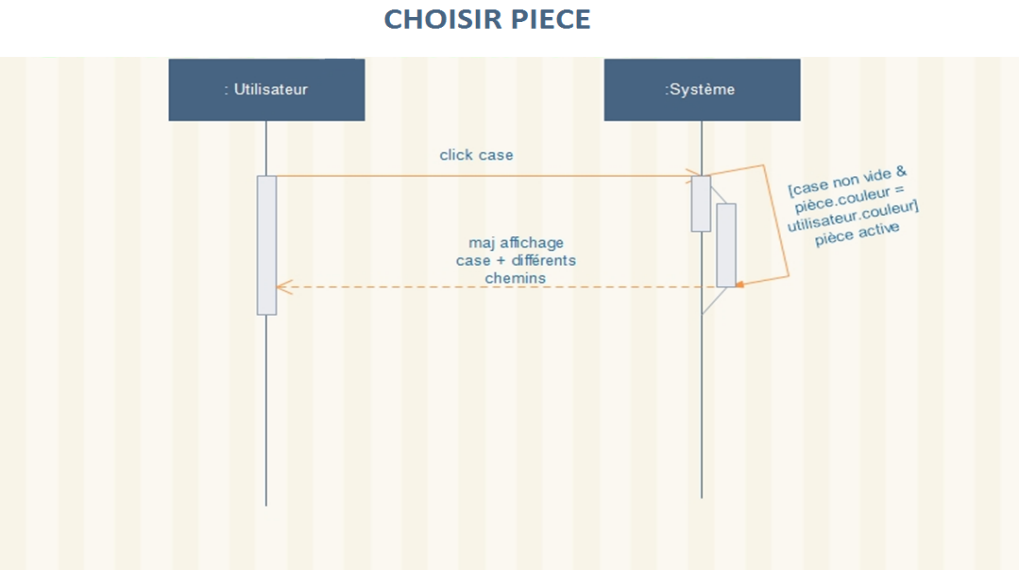
Cas d'utilisation du jeu

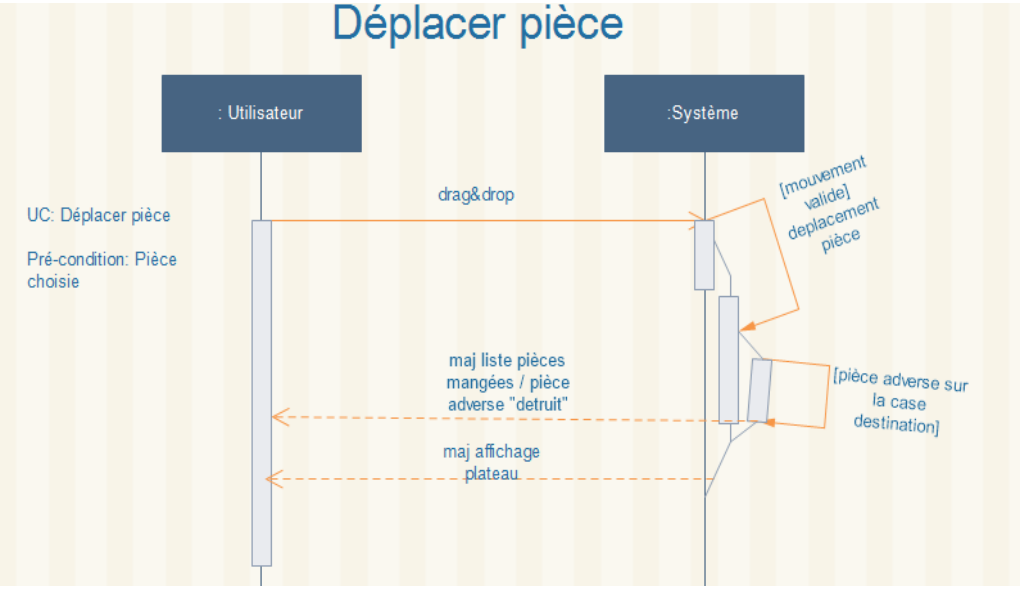
Interface graphique :
- Quitter système
- Pause
- Abandonner partie
- Modifier les options
- Ecrire les logins des deux utilisateurs

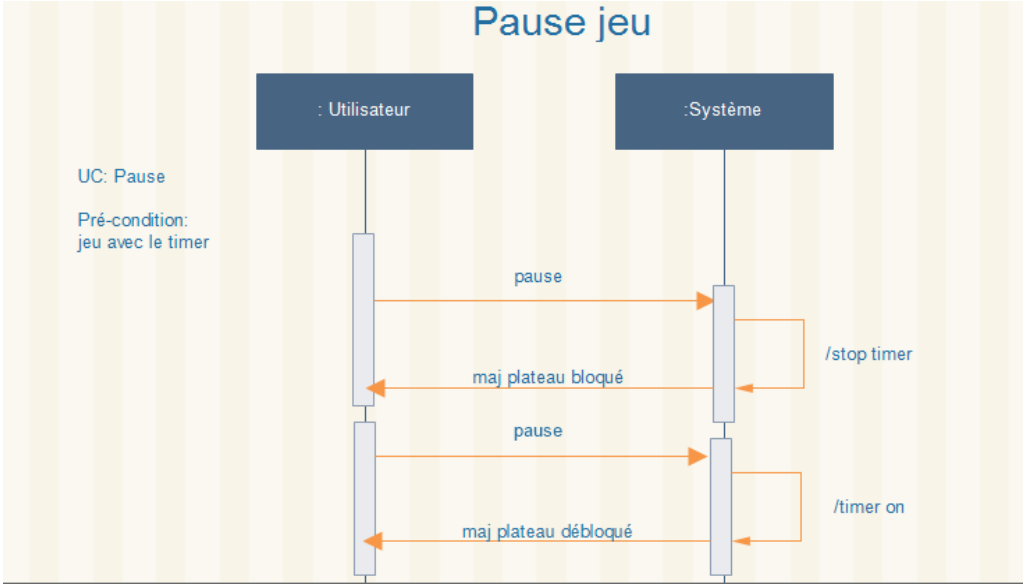
Moteur de jeu :
- Choisir une pièce
- Déplacer une pièce
- Prendre une pièce adversaire
- Choisir case valide

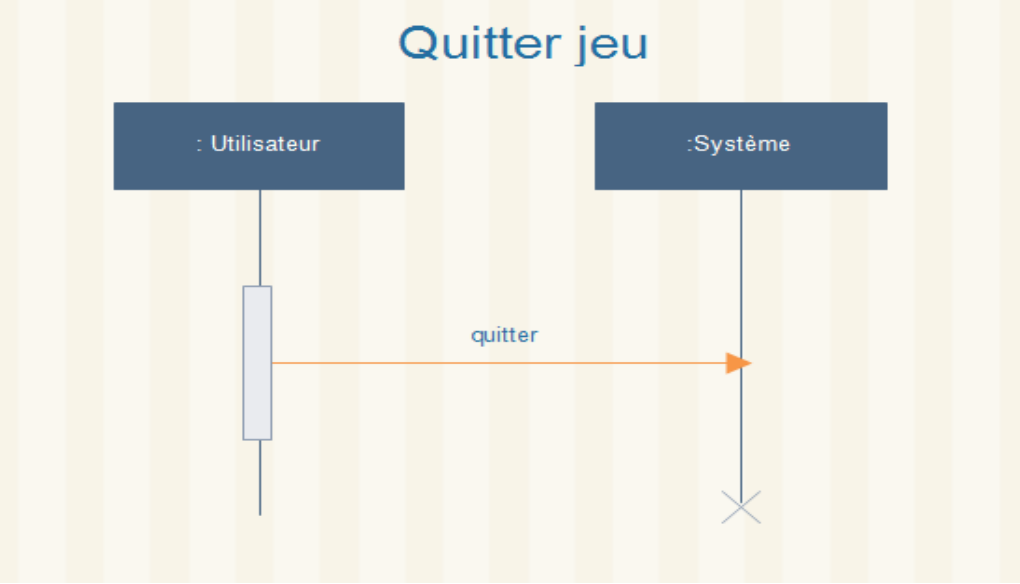
Gestion du temps :
- Décrémenter timer
- Changer le joueur actuel

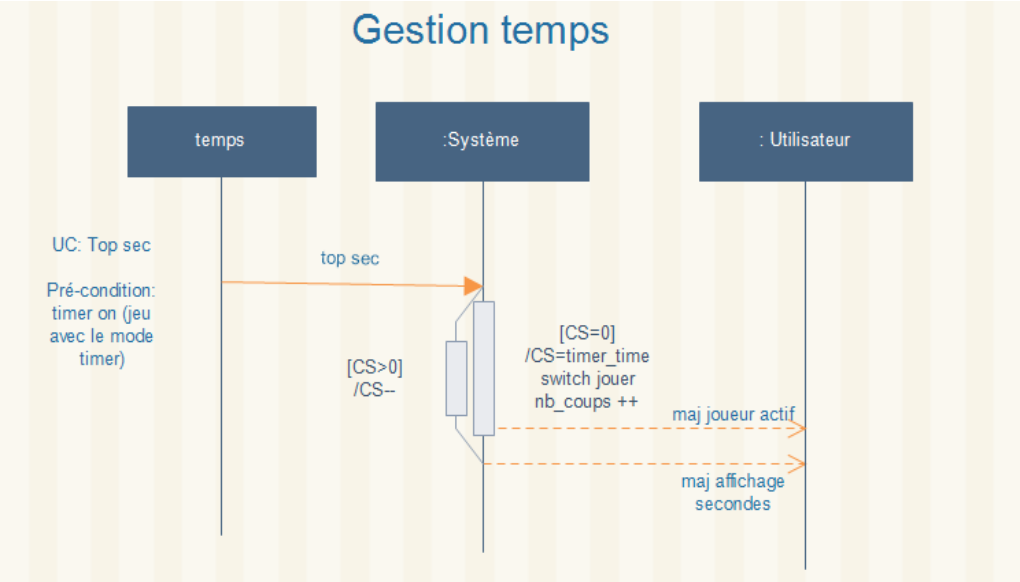
Diagrammes de séquence




Déplacer pièce
: Utilisateur
:Système
UC: Déplacer pièce
Pré-condition: Pièce choisie
drag&drop
[mouvement valide] deplacement pièce
maj liste pièces mangées / pièce adverse "detruit"
[pièce adverse sur la case destination]
maj affichage plateau


Pause jeu
: Utilisateur
:Système
UC: Pause
Pré-condition: jeu avec le timer
pause
/stop timer
maj plateau bloqué
pause
/timer on
maj plateau débloqué


Quitter jeu
: Utilisateur
:Système
quitter


Gestion temps
temps
:Système
: Utilisateur
UC: Top sec
Pré-condition:
timer on (jeu
avec le mode
timer)
top sec
[CS>0]
/CS--
[CS=0]
/CS=timer_time
switch jouer
nb_coups ++
maj joueur actif
maj affichage
secondes

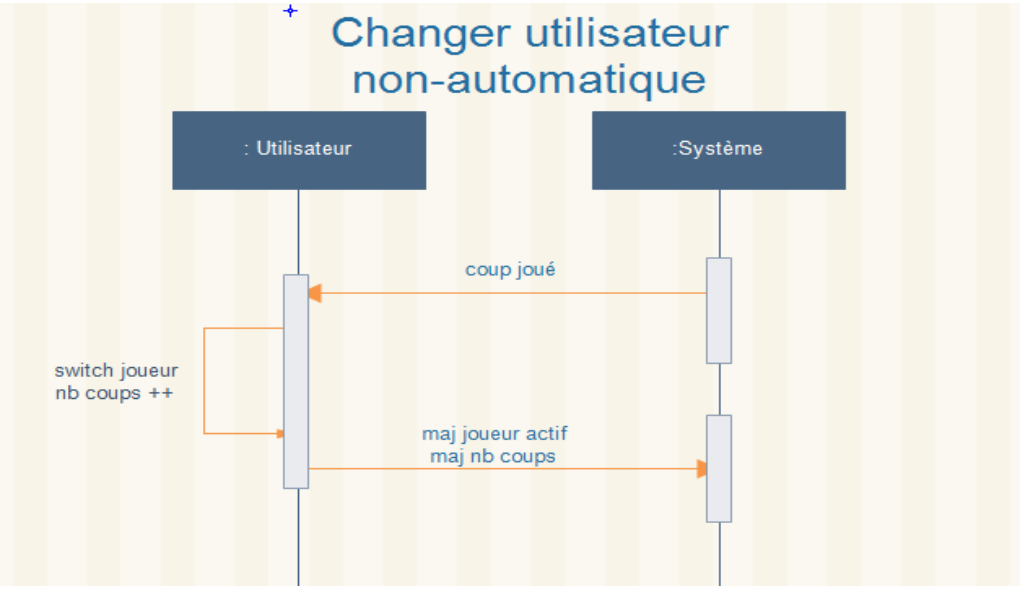
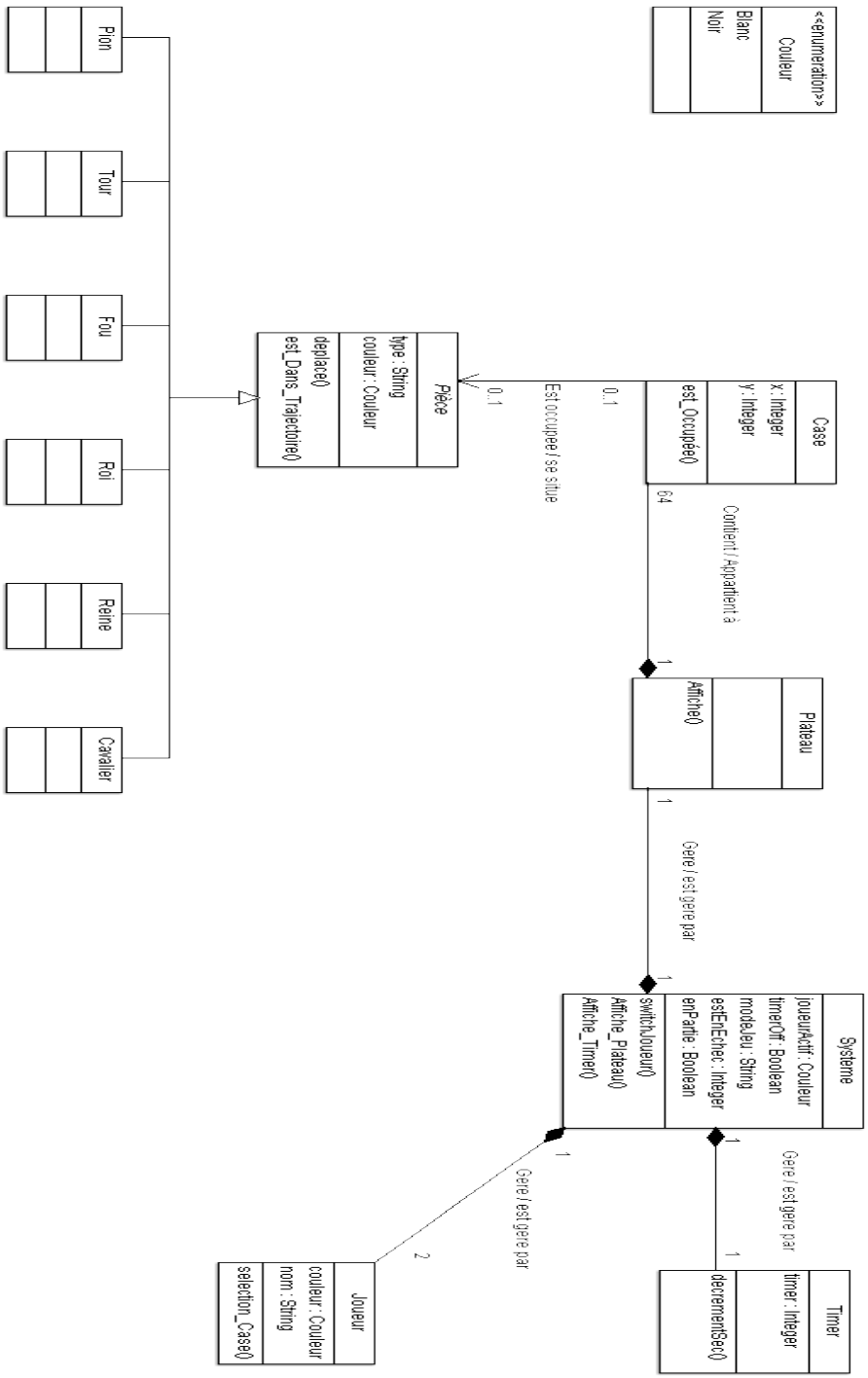

Changer utilisateur
non-automatique
: Utilisateur
:Système
coup joué
switch joueur
nb coups ++
maj joueur actif
maj nb coups

Diagramme de classe



Documents de conception

Architecture logique du logiciel : MVC

On a choisi ce modèle car :
— il y a beaucoup d'interactions avec l'utilisateur à travers l'IG susceptible de modifier la vue en interprétant les requêtes
— certains mécanismes (règles du jeu) doivent interagir entre contrôleur et la vue, ce qui n'est pas possible avec une architecture en couche.

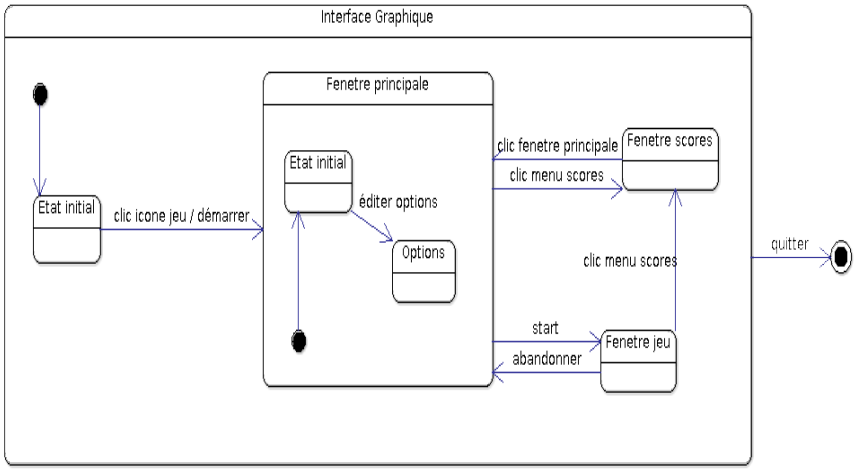
Description de l'incrément choisi

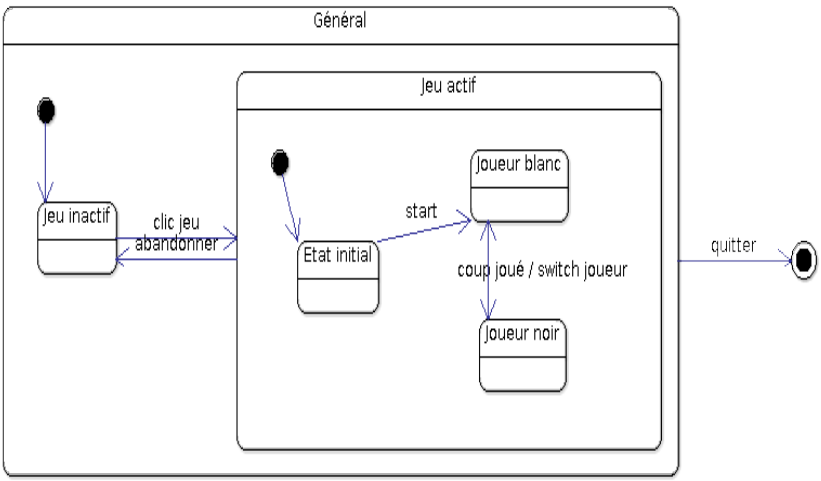
IG : en 2D ; fenêtre des scores ; fenêtre des règles du jeu ; fenêtre de jeu

Mode de jeu : 2 joueurs

Moteur de jeu : sélection d'une case jouable ; prise d'une pièce adverse ; mouvements classiques ; échec/échec et mât

Diagramme d'états-transition




Général
Jeu actif
Jeu inactif
clic jeu
abandonner
Etat initial
start
Joueur blanc
coup joué / switch joueur
Joueur noir
quitter

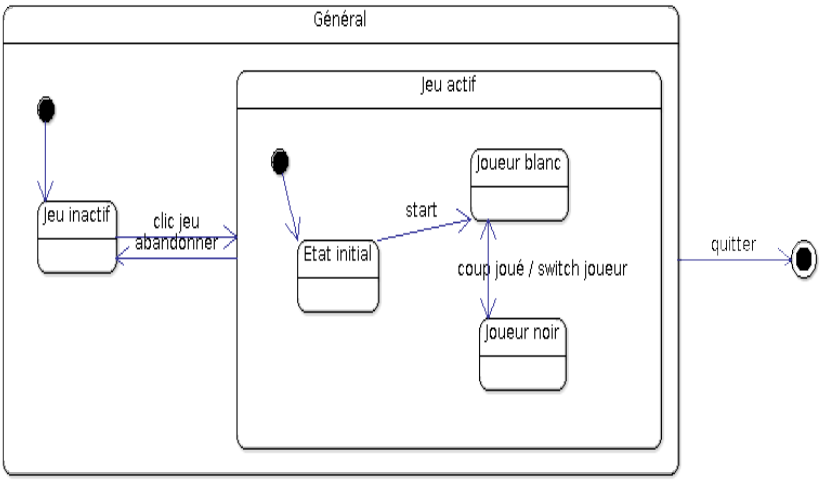
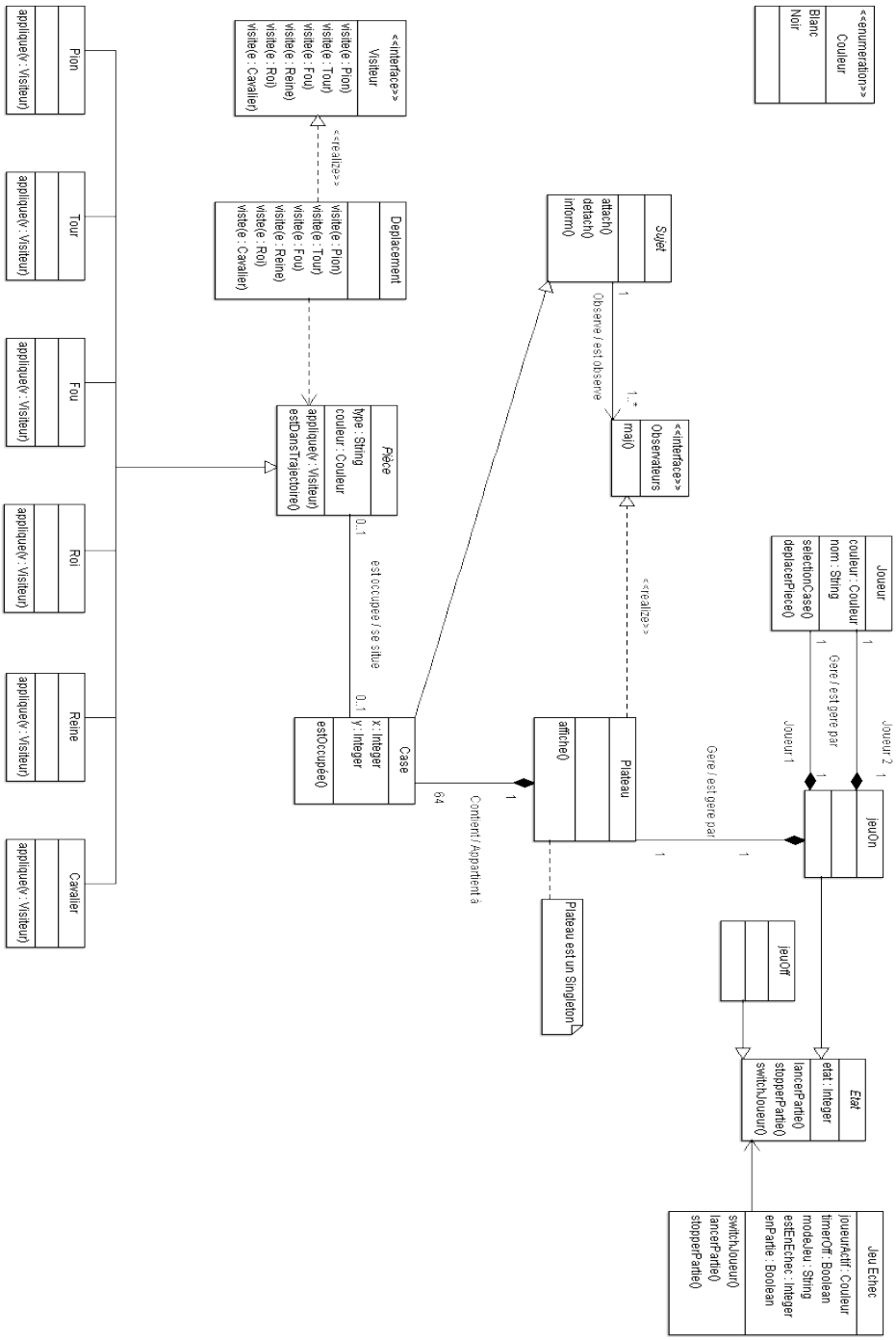

Général
Jeu actif
Jeu inactif
clic jeu
abandonner
Etat initial
start
Joueur blanc
coup joué / switch joueur
Joueur noir
quitter

Diagramme de classes logicielles



Manuel de l'utilisateur pour l'incrément

Débuter une partie

L'utilisateur doit cliquer sur le menu déroulant *File* puis sur le sous menu *New Game* afin de commencer une nouvelle partie.

Terminer une partie

L'utilisateur a plusieurs façons de terminer la partie :
la plus brutale est de fermer la fenêtre de jeu,
la plus naturelle est d'attendre la fin du jeu en gagnant ou en perdant la partie.

Déplacer une pièce

Sélectionner votre pièce en cliquant dessus avec la souris puis en maintenant le bouton enfoncé, déplacer celle-ci jusqu'à la case souhaitée (drag and drop).
Si cette case est valide, le mouvement est validé et votre pièce déplacée.
Une case est valide si elle respecte les possibilités de mouvement de la pièce sélectionnée.

Prendre une pièce adverse

Déplacer votre pièce (cf section précédente), si son mouvement est correct et que sa case d'arrivée correspond à une case ou se situe une pièce adverse alors vous "prenez" la pièce adverse.
Attention au cas particulier des pions qui prennent en diagonale, ie sur la case de la ligne supérieure et de la colonne droite ou gauche.

Voir les règles

Cliquer sur le menu déroulant *File* puis sur le sous menu*Rules*.

Voir les scores

Cliquer sur le menu *Scores*.

Bilan sur les outils de modélisation utilisés

Problèmes rencontrés

Nous avons utilisé ArgoUML pour construire les diagrammes de classes et les diagrammes d'états-transitions. Ce logiciel est facile à prendre en main mais le tracé des diagrammes de classes a été laborieux. En effet le déplacement des flèches est délicat à réaliser.

Pour les diagrammes de séquence nous avons utilisé , il était plus facile de tracer les diagrammes séquences qu'avec ArgoUML.

Le programme n'a pas été entièrement réalisé suite à des problèmes d'implémentations. Nous avons rencontré de nombreaux problèmes liés au langage Java (affichage/ré-affichage) pour l'interface graphique. Beaucoup de temps a été perdu pour l'interface graphique.C'est pourquoi l'ensemble des fonctionnalités de l'incrément n'ont pas pu toutes êtres réalisées : toutes classes du diagramme de classe ont été implémentées.Cependant il manque quelques fonctionnalitées dans les classes "jeuON" et dans "pièces" pour le déplacement des pièces.

Les patrons Visiteurs/Observateurs/Etats ont été implémentés. Nous avons remarqué que, pour le patron visiteur, Case devait elle aussi être Observateur de Pièce, et pas seulement Pièce. Nous avions besoin de pouvoir informer le plateau qu'une pièce a été déplacée, et le seul lien entre Pièce et Plateau était case. C'est donc logique qu'une case observe si il y une pièce dessus ou pas . On peut déplacer les pièces, même si elles réagissent parfois bizarrement. La vérification des déplacements spéciaux pour chaque pièce mais n'a pas été utilisée dans le programme par manque de temps. On peut afficher un tableau des scores et une page des règles.

Solutions apportées

ArgoUML : beaucoup de patience et de self-control.
: