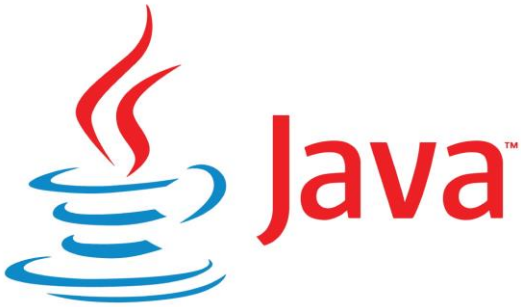
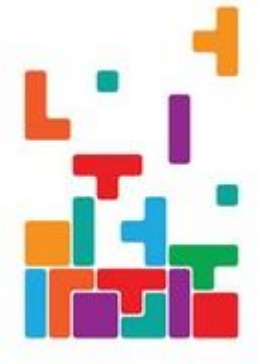
Projet Android - Tetris

LéCubKiTomb

Un jeu Tetris like.

Par Loriane L'Hostis et Mathieu Suchet.





IUT DE CLERMONT-FD
Aurillac - Le Puy-en-Velay - Clermont-Ferrand
UNIVERSITÉ
Clermont Auvergne

2ème année DUT informatique 2021-2022.

Sommaire:

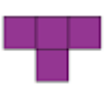
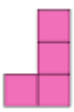


Contextualisation

Vous avez déjà eu envie de jouer à un jeu sans vous prendre la tête ou sans vraiment savoir à quoi jouer ?
Venez essayer notre jeu “LéCubKiTomb”. Il reprend le principe du très célèbre jeu Tetris. Notre application vous propose de jouer très facilement à ce jeu en un clic seulement !
Notre jeu propose quelques fonctionnalités en plus, comparé à un Tetris “classique”.
Vous pouvez faire bouger et tourner votre pièce en déplaçant votre téléphone vers la direction voulue, ou en faisant tourner votre téléphone vers la rotation voulue.
Notre jeu propose une gestion de la taille de la grille pour que la difficulté soit modulaire. Un slider est disponible dans la page d’options, ce qui vous permet de modifier la difficulté en fonction de vos envies.
Vous pouvez aussi consulter votre score en fin de partie mais vous apparaitrez dedans seulement si vous faites partie de l’élite des joueurs !

L’application respecte la “Separation of Concerns” (SoC). C’est un concept de conception visant à séparer un programme informatique en plusieurs parties. Chacune des parties doit donc gérer un aspect, une partie de l’application. C’est une bonne pratique qui permet de segmenter le code et de le rendre plus simple pour notamment la maintenance mais aussi le débogage.
Plus largement, cela permet de donner une responsabilité à un “projet”.

Dans notre application, la partie fonctionnelle de jeu est donc codée en Java mais le Java ne peut pas s’occuper de la partie visuelle. Il est doté d'une partie “visuelle” très simple en console mais cette partie se limite ici. Le jeu doit intégralement fonctionner en mémoire sans que la partie visuelle soit présente.
La partie graphique est donc quant à elle codée en XML. Elle vient se superposer à la partie Java. C’est une sorte de surcouche permettant d’avoir d’une part un aspect graphique pour notre jeu mais aussi de modifier certains paramètres comme la taille de la grille, etc.
Le jeu est relié avec l’affichage pour que ces deux concepts soient synchronisés.

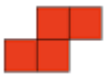
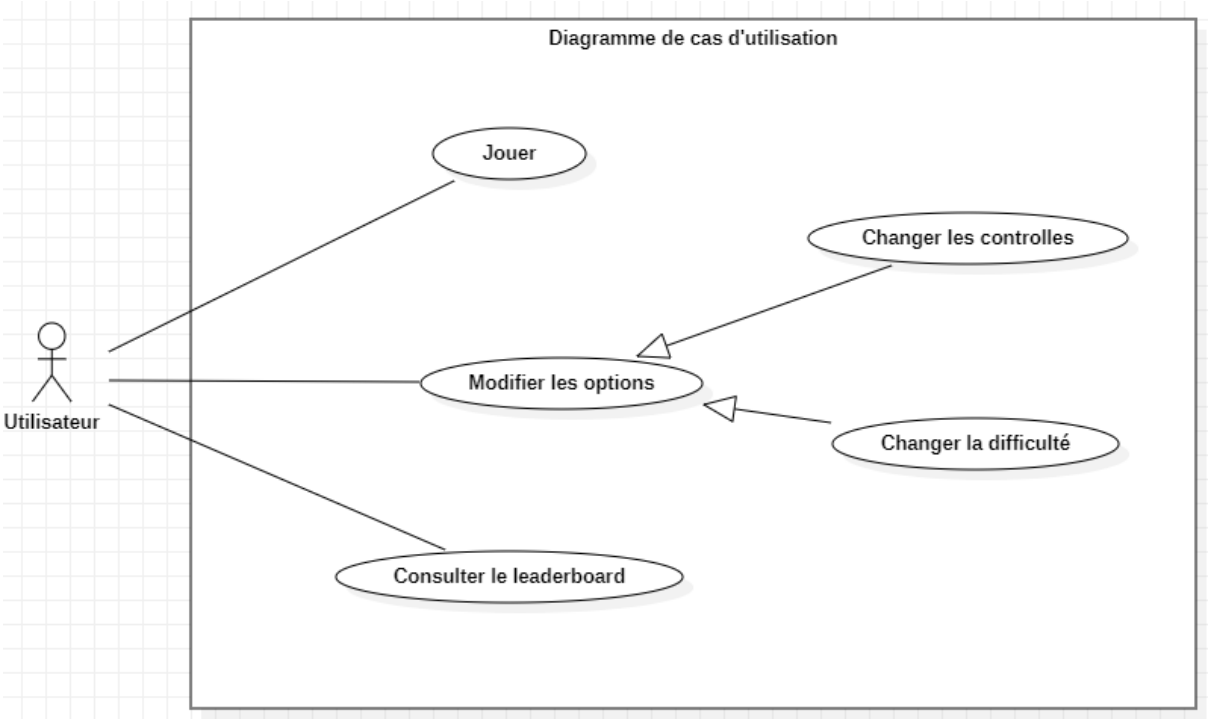


Diagramme de cas d'utilisation



Le diagramme ci-dessus présente les différents cas d'utilisation de notre jeu. Ce sont les différentes raisons pour lesquelles une personne serait amenée à lancer notre jeu. Une personne peut tout simplement vouloir jouer, vouloir modifier les options pour améliorer son expérience utilisateur.
Elle peut aussi vouloir consulter son classement une fois des parties effectuées, ou tout simplement consulter le classement général.

Cas Jouer

Nom	Jouer
Objectif	Lancer une partie de jeu
Acteur Principal	Utilisateur
Condition initiale	- L'utilisateur doit avoir lancé l'application
Scénario d'utilisation	- L'utilisateur lance l'application - L'utilisateur clique sur jouer et la partie de jeu se lance
Condition de fin	- L'utilisateur est sur sa partie de jeu

Cas Modifier les options

Nom	Modifier les options
Objectif	Modifier les contrôles ou la taille de la grille, autrement dit la difficulté
Acteur Principal	Utilisateur
Condition initiale	- L'utilisateur doit être sur la page d'accueil de l'application
Scénario d'utilisation	- L'utilisateur clique sur les paramètres
Condition de fin	1) L'utilisateur change ses contrôles 2) L'utilisateur change la taille de la grille 3) L'utilisateur revient à l'accueil sans faire de modifications

Cas consulter le leaderboard

Nom	Consulter le leaderboard
Objectif	Consulter le classement des joueurs
Acteur Principal	Utilisateur
Condition initiale	- L'utilisateur doit être sur la page d'accueil de l'application
Scénario d'utilisation	- L'utilisateur clique sur le bouton de leaderboard
Condition de fin	- L'utilisateur est sur la page de leaderboard

Fonctionnalité

Boucle temporelle

Le fonctionnement de notre application est basé sur une boucle de temps. Une boucle est matérialisée par le déplacement d'une pièce sur une case de la grille. La boucle n'a pas tout le temps la même durée, elle s'adapte en fonction de la durée de la partie. La boucle s'accélère petit à petit tout au long de la partie pour augmenter la difficulté de manière linéaire.
La boucle peut aussi être "finie prématurément", c'est-à-dire que quand le joueur décide de faire descendre la pièce avec la touche dédiée à cette action, la boucle est en quelque sorte avortée et la pièce fait son déplacement de manière instantanée.

Le déplacement des pièces

Chaque pièce peut se déplacer de gauche à droite sur la grille. Elle est stoppée lorsqu'elle rencontre le bord de la grille ou alors une autre pièce. Une pièce peut aussi se déplacer vers le bas, mais elle ne peut pas remonter la grille.

Rotation des pièces

Une pièce peut aussi tourner avec les mêmes contraintes que précédemment. Lorsqu'elle rencontre le bord de la grille ou une autre pièce, ses déplacements sont restreints si une collision est détectée.

Changement de la difficulté

De la même manière, il est aussi possible de personnaliser la taille de la grille. Dans les paramètres, l'option est disponible sous l'intitulé "difficulté". Le changement de difficulté provoquera le changement de la taille de la grille.



Fixation des pièces

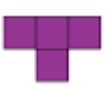
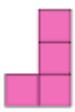
Une fois qu'une pièce atteint le bas de la grille, elle se fixe à cette dernière. Elle se fixe aussi à cette dernière lorsque cette pièce touche une autre pièce, autrement dit une fois qu'elle ne peut plus descendre.

Système de points

Lorsqu'une pièce est accélérée, elle rapporte des points. Elle en rapporte aussi une fois qu'elle se pose. Mais le système de points est basé sur la destruction de lignes. Le but étant de remplir tous les trous, une fois que les pièces qui sont fixées à la grille forment une ligne, cette dernière est détruite et elle rapporte plus de points.
Les lignes peuvent être détruites au maximum par 4 car la pièce la plus haute fait 4 de hauteur.

Fin de partie

Une partie se finit lorsqu'une pièce apparaît et qu'elle ne peut pas descendre. Cela signifie que l'accumulation des pièces est montée jusqu'à l'endroit où elles apparaissent. Plus aucune pièce ne peut alors s'ajouter à la grille ; la partie est donc finie.

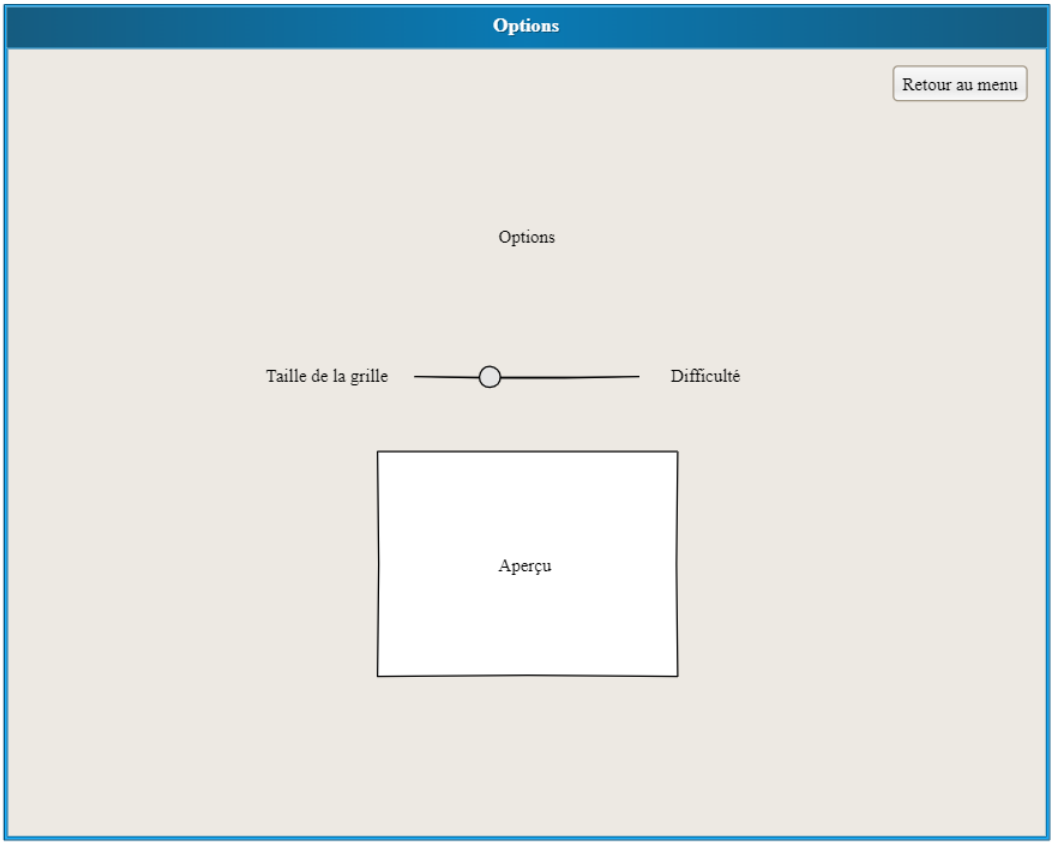



Sketchs

Menu

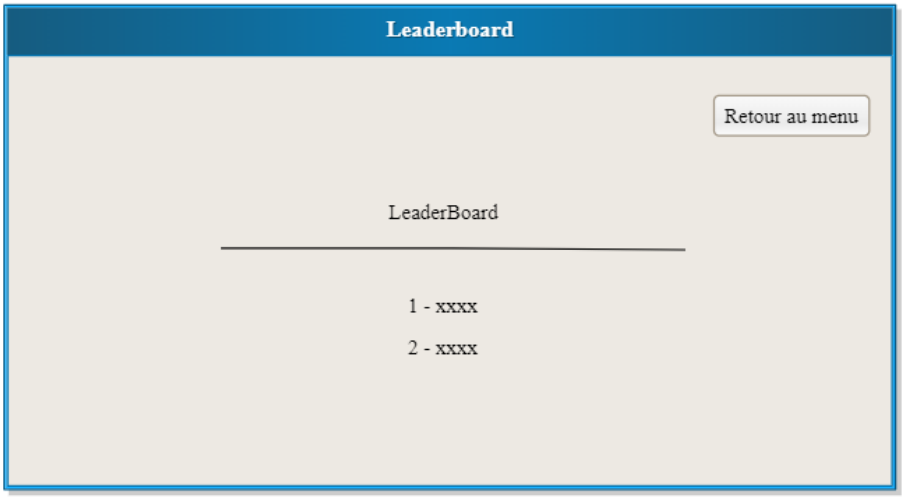
Cette vue présente simplement la page d'accueil. On peut voir le nom de l'application avec le texte intitulé "titre".
On a ensuite le bouton des paramètres, celui du leaderboard et enfin celui de jeu. C'est la page principale qui permet d'accéder aux différentes pages du jeu.

Options



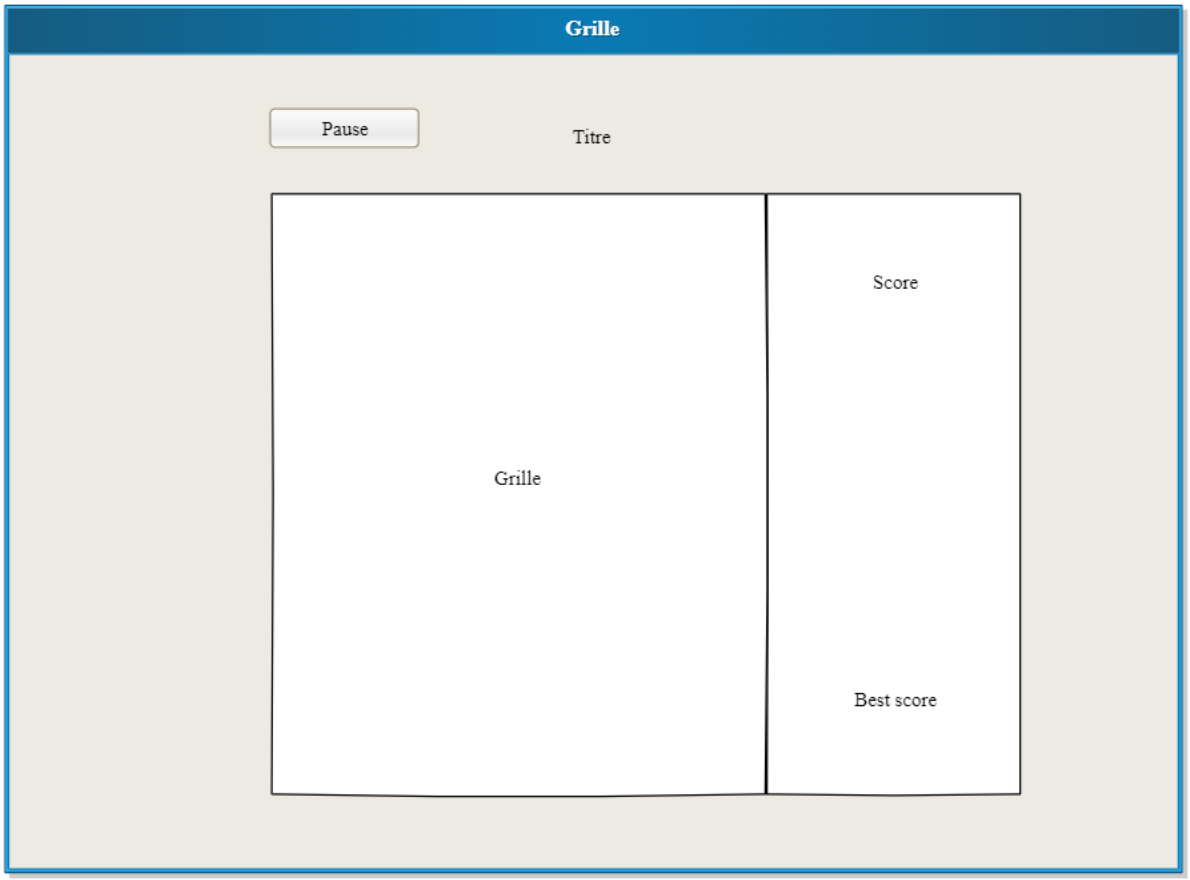
On peut voir ci-dessus la page des options. Cette page nous permet de modifier la difficulté. Cette dernière est modélisée par la taille de la grille. On a un petit aperçu juste en dessous des modifications que l'on est en train d'effectuer.
Pour finir, on a un bouton "retour au menu" qui, comme son nom l'indique, permet de retourner au menu principal.

Leaderboard



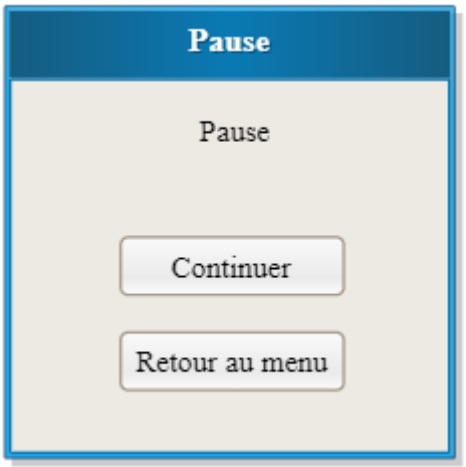
La page de leaderboard présente simplement une liste des joueurs ayant fait les meilleurs scores sur le jeu.
Elle dispose aussi d'un bouton permettant de retourner à l'accueil.

Grille



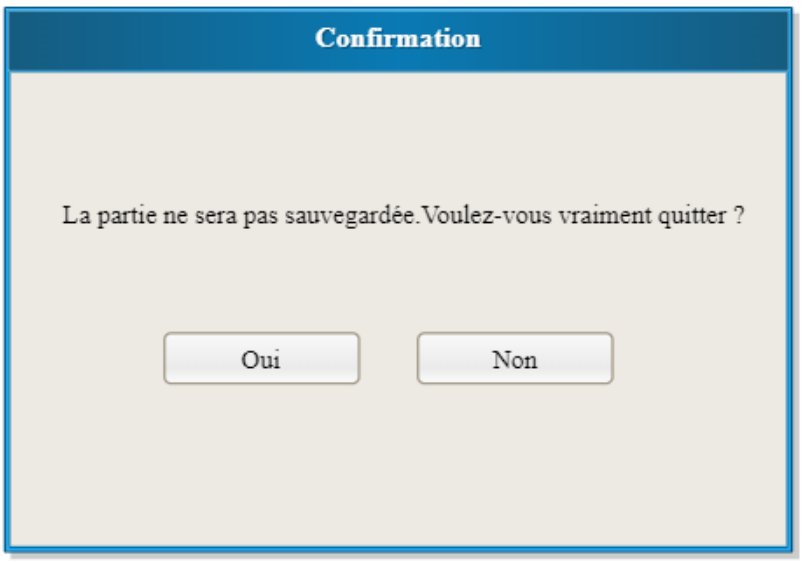
La page de la grille est la page de jeu principale. C'est là où le jeu se déroule. Sur la partie de gauche, on peut voir la grille de jeu. Sur la partie droite, on peut voir les pièces à venir, le score courant de la partie et le meilleur score de l'application.

Pause



La page de pause se présente comme ci-contre. Elle est accessible en appuyant sur le bouton “echap” depuis la page de jeu et uniquement depuis la page de jeu. Elle permet de mettre en pause le jeu le temps d’aller prendre un café par exemple. Depuis cette page, on peut décider de continuer à jouer. Dans ce cas, la page de pause disparaît et on est de retour sur le jeu.
On peut aussi vouloir retourner au menu. Dans ce cas, la page de confirmation apparaît.

Confirmation

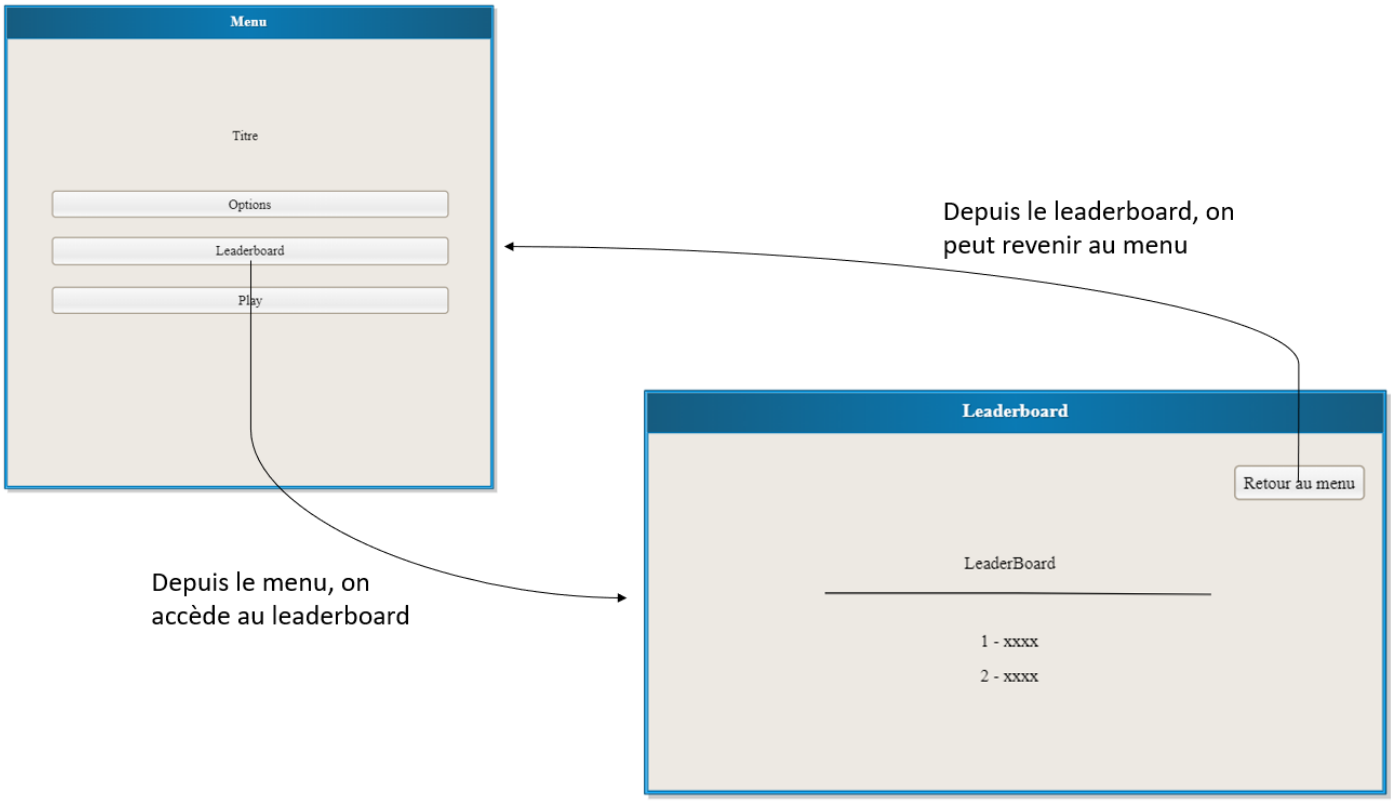


La page de confirmation se présente comme ci-contre. Elle permet d’instaurer une dernière validation avant de quitter le jeu. On peut valider en cliquant sur “oui”. On peut aussi revenir à la page de “pause” en cliquant sur “non”.

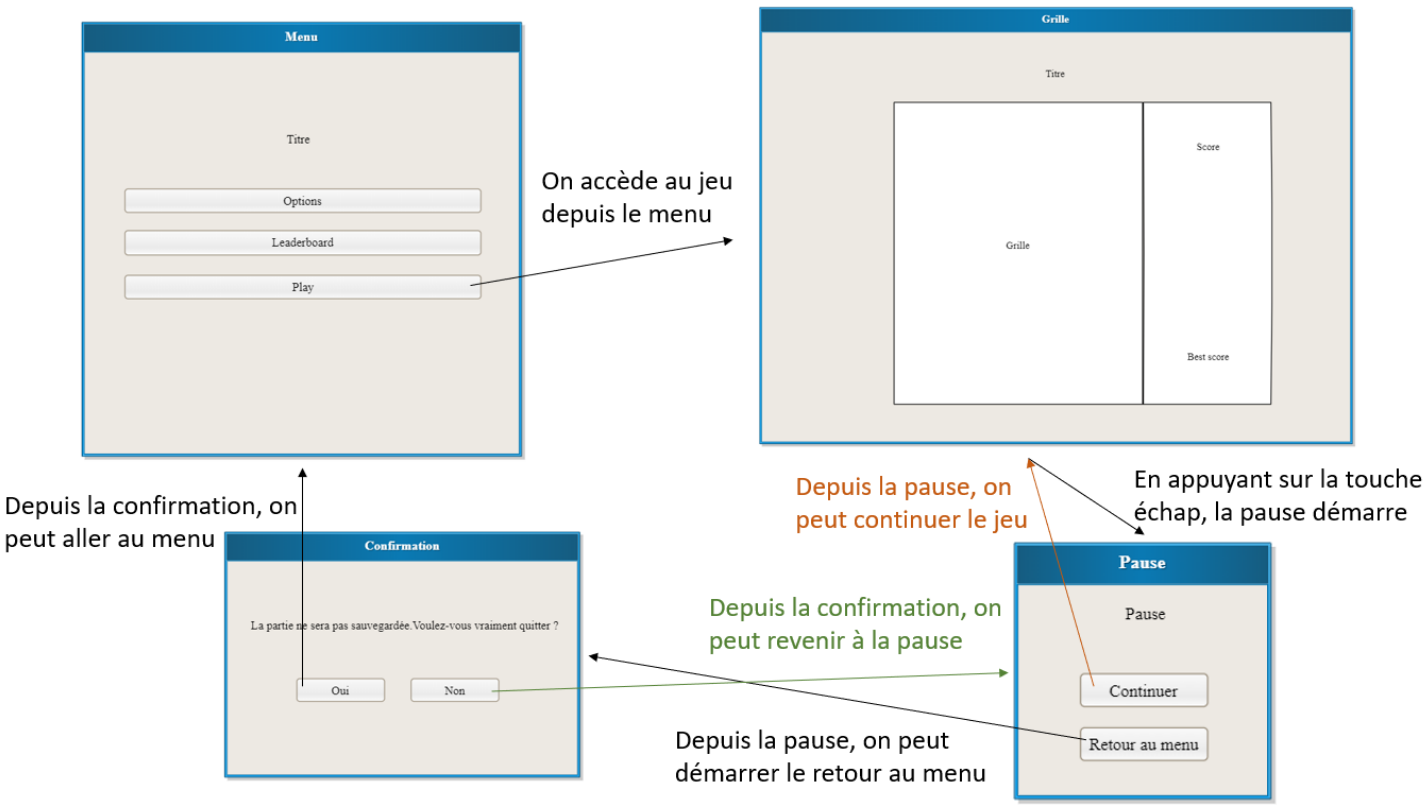
GameOver

La page de game over permet de rejouer ou de revenir au menu selon le choix de l'utilisateur

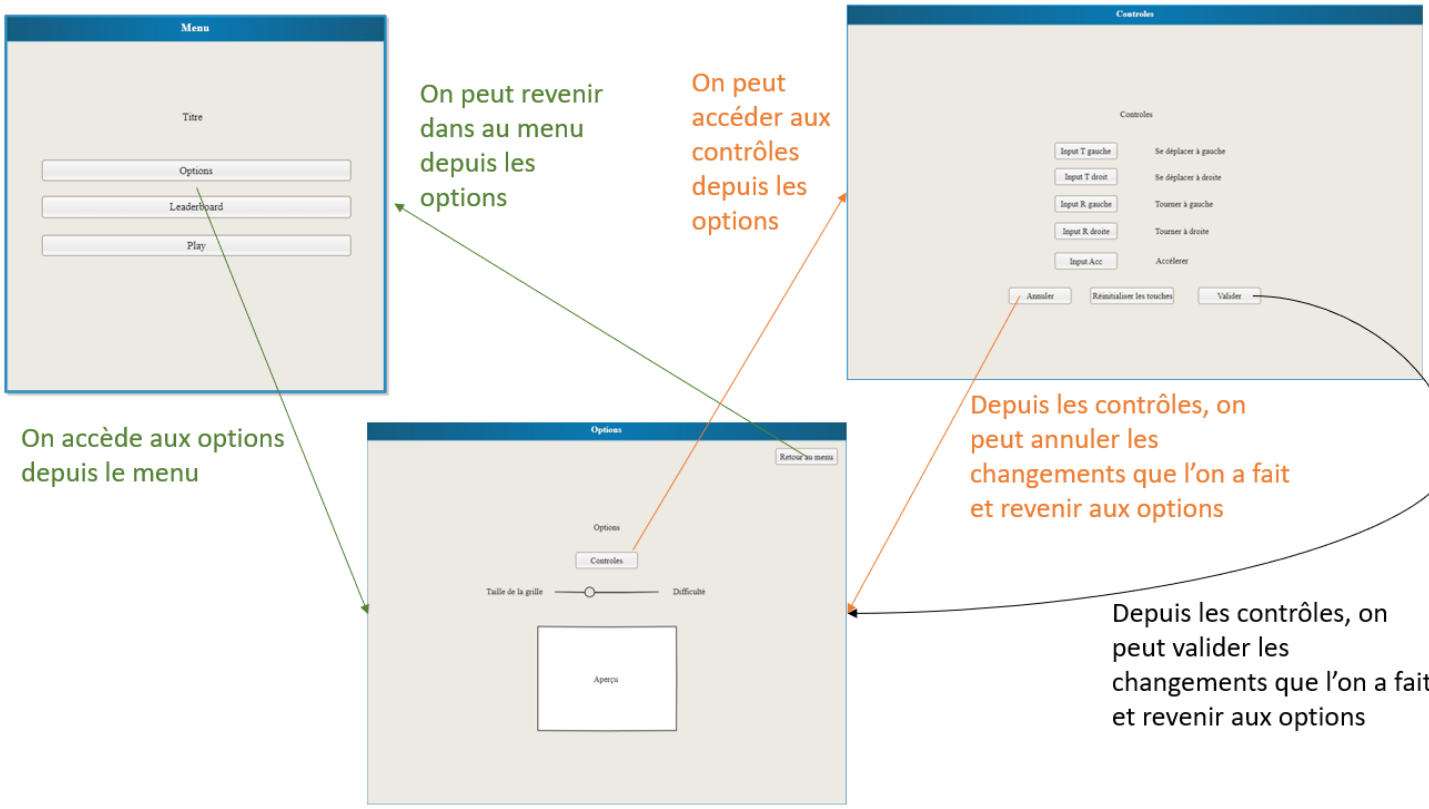
Storyboards



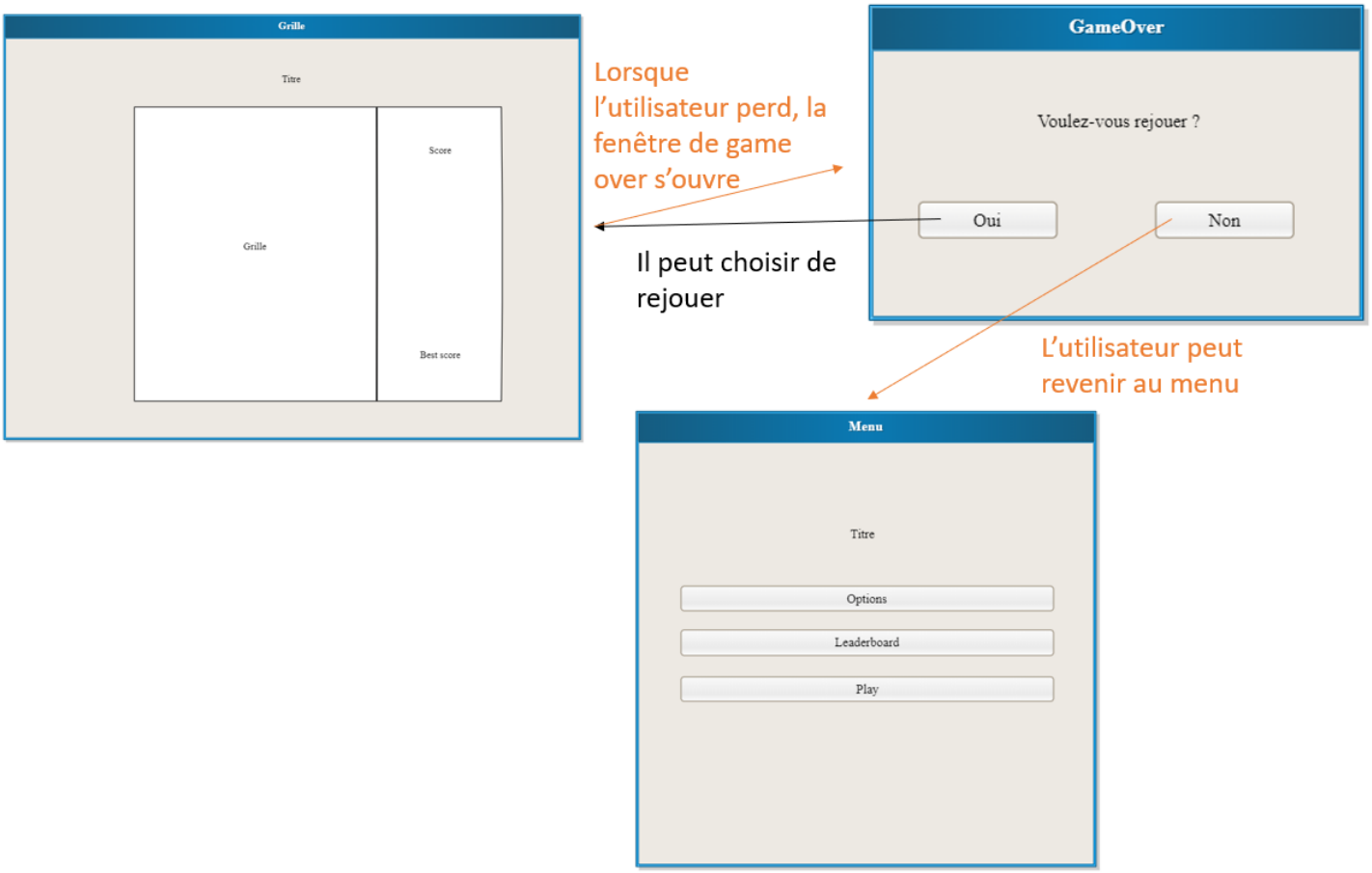
Ce storyboard explique comment l'utilisateur peut accéder au leaderboard. C'est une option directement accessible depuis le menu du jeu. On peut, de manière réciproque, revenir au menu depuis le leaderboard.



Pour jouer, l'utilisateur dispose d'une option "Play" dans le menu qui l'amène jusqu'à la grille de jeu. Une fois dans cette grille, il peut mettre sa partie en pause en appuyant sur la touche "Echap", ce qui a pour effet de créer une fenêtre de pause. A ce stade, l'utilisateur a deux chemins possibles, soit il appuie sur "Continuer" et la partie reprend soit il appuie sur "Retour au menu" et le processus de retour au menu commence. En appuyant sur ce bouton "Retour au menu", une autre fenêtre s'ouvre : la fenêtre de confirmation, qui propose 2 choix, soit "Oui", soit "Non". Dans le cas où l'utilisateur appuie sur "Oui", la fenêtre de pause et de confirmation se ferme, et la grille est remplacée par le menu. Dans le cas où l'utilisateur appuie sur "Non", la fenêtre de confirmation se ferme et l'utilisateur revient sur la fenêtre de pause.

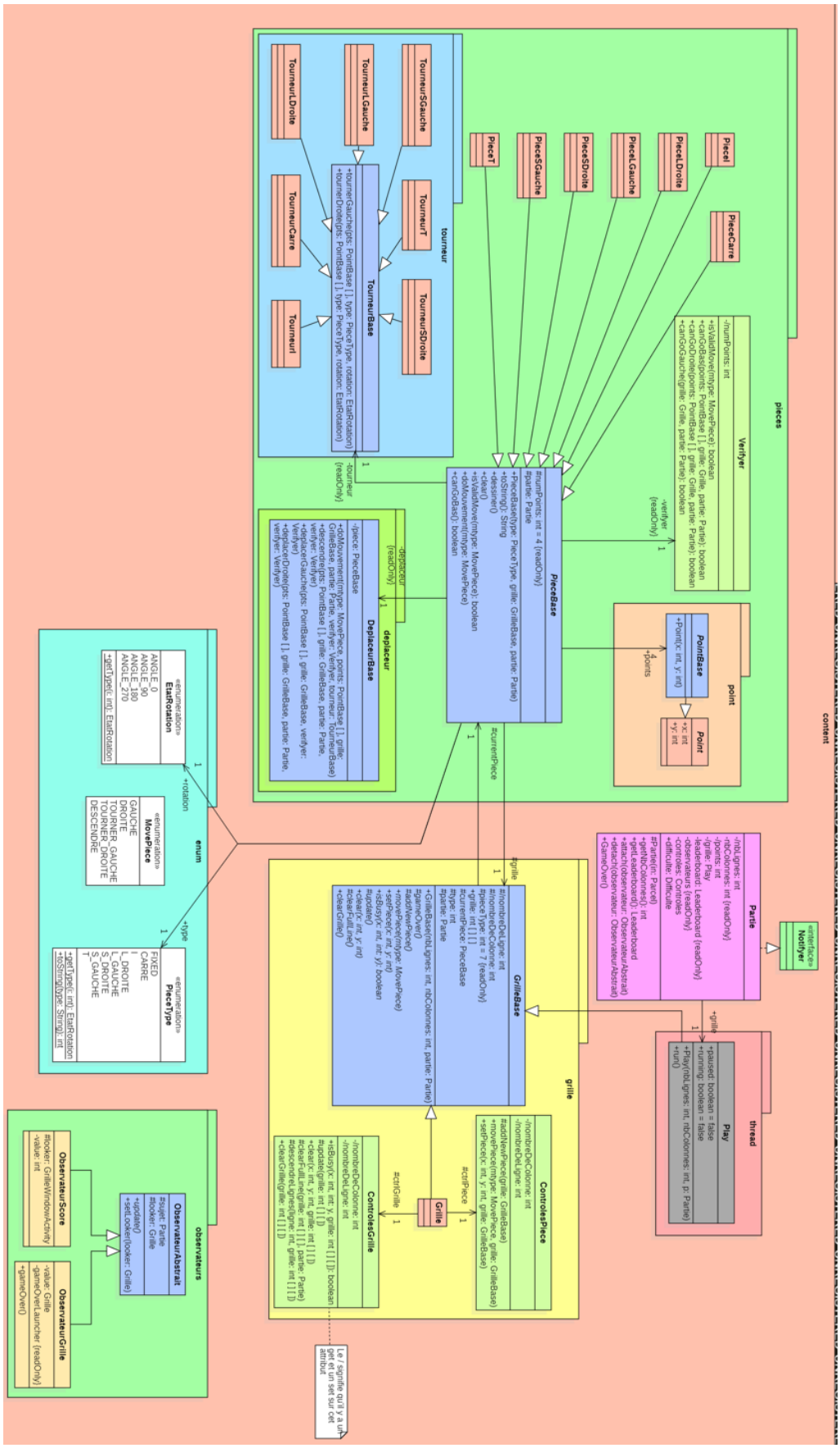


Pour accéder aux options, l'utilisateur a un bouton dans le menu nommé "Options", il peut (comme avec le leaderboard) revenir sur le menu depuis la page des options. Dans cette page des options, l'utilisateur peut accéder à la page des contrôles grâce au bouton "Contrôles". Ce bouton crée une nouvelle page dans laquelle seront exposés les différents contrôles utilisés dans le jeu. Depuis cette page, on peut revenir à la page des options de 2 manières : soit par le bouton "Annuler", ce qui aura pour effet de revenir au menu en ne prenant pas en compte les changements fait par l'utilisateur, soit par le bouton "Valider", qui revient sur la page des options en ayant sauvegardé ce que l'utilisateur a potentiellement modifié.



Depuis la grille, si l'utilisateur perd, une fenêtre de game over s'ouvrira pour lui demander s' il souhaite rejouer ou non, si l'utilisateur répond "Oui", la fenêtre de game over se ferme, et une nouvelle partie se lance, sinon, l'utilisateur revient au menu principal.

Diagramme de classes

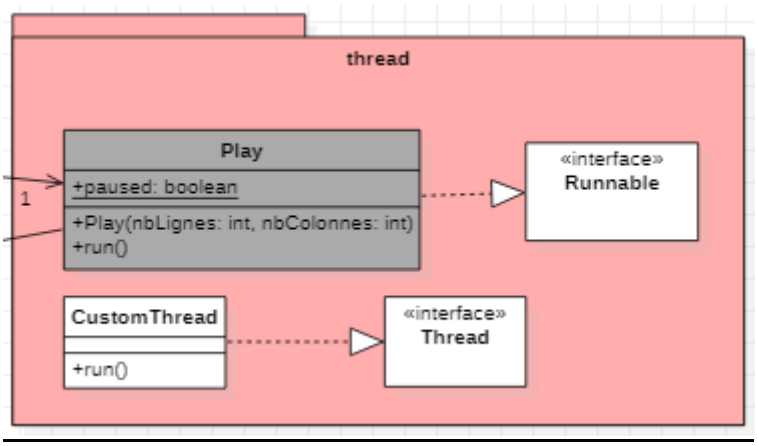


Le diagramme de classes ci-dessus représente l'agencement de la totalité de nos classes nécessaires au bon fonctionnement de notre jeu. Ce diagramme de classe concerne uniquement la partie « logique » de notre jeu. C'est-à-dire que ce diagramme concerne uniquement le jeu et non sa représentation graphique. Nous pouvons observer sur ce diagramme que toute l'application est manipulée par une classe « mère », qui est la classe « Partie ». Le diagramme est aussi organisé en différents packages. Ces derniers sont des facteurs importants pour la qualité du code car, pour chaque entité, par exemple une pièce, toutes les classes là concernant sont regroupées dans le même package. De la même manière, les packages nous permettent de mieux gérer les visibilités d'attributs ou de méthodes au sein des différentes classes. Dans notre application, nous avons décidé d'utiliser différents patrons de conception. Nous avons aussi cherché à respecter le plus possible les principes S.O.L.I.D.

Partie

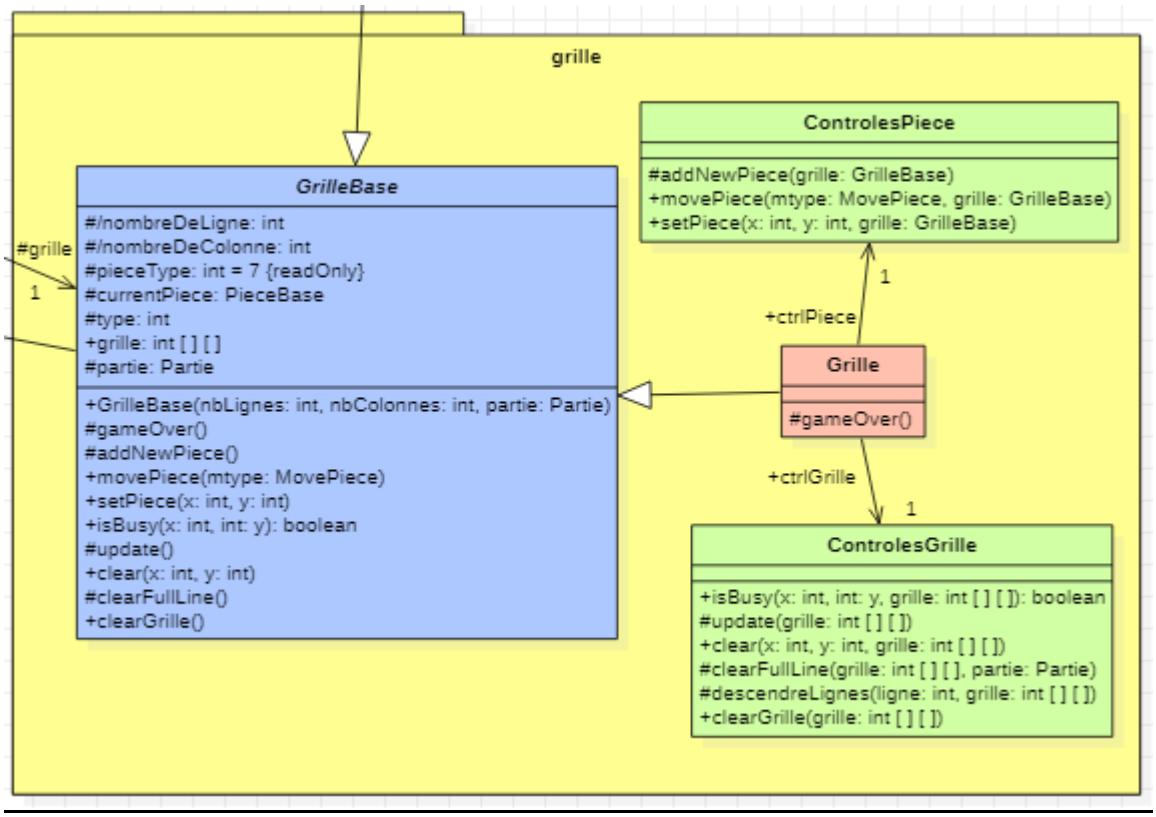
La classe Partie est, comme dit précédemment, la classe centrale de notre application. Nous utilisons ici le **patron de conception « Façade »**. La classe Partie étant la classe permettant de coordonner toutes les autres classes les unes entre les autres. C'est la classe qui rend plus facile l'utilisation de toutes les autres classes du jeu. Pour pouvoir lancer notre jeu, il suffit donc de créer un partie grâce à son constructeur, et ensuite lancer la méthode main(), l'application est conçue de façon que tout fonctionne correctement lorsqu'on la lance depuis cet endroit. Le patron de conception est donc justifié car il simplifie l'utilisation de l'application dans le but de centraliser son fonctionnement. La classe Partie ne possède qu'un seul attribut étant de type d'une autre classe de l'application, mais permet tout de même de jouer le rôle de chef d'orchestre parmi l'ensemble.

Package thread



Ce package est uniquement là pour définir notre boucle de jeu. Elle est présente dans la classe Play, qui implémente l'interface Runnable. On retrouve dans cette classe le constructeur d'un Play, qui a pour but d'initialiser les différents paramètres nécessaires au bon fonctionnement de notre application. Elle possède aussi la méthode run(), elle permet de lancer le thread, et donc, de lancer notre jeu.

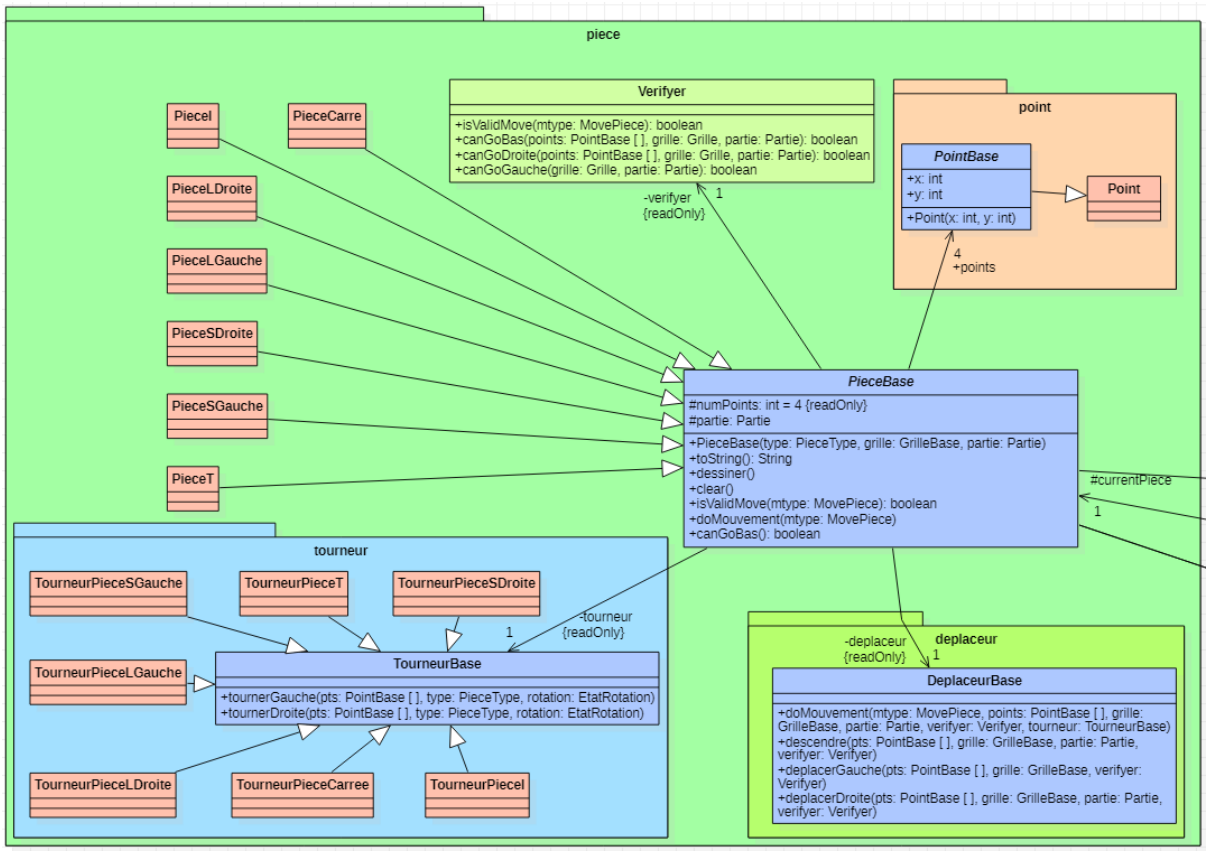
Package grille



Le package “grille” permet de regrouper toutes les classes concernant la grille. La classe principale de cette classe est la classe abstraite *GrilleBase.* C’est classe est abstraite, dans le but de respecter le principe S.O.L.I.D, O (Open/Close principle). Ce principe stipule qu’une application doit être ouverte à l’extension mais fermée à la modification. En bref, pour un souci de qualité, une fois notre application terminée, il ne faudra plus modifier son code, pour éviter le plus possible d’implémenter de nouveaux bugs. C’est donc pour cela que nous devons créer des points d’extensibilité par le biais de classes abstraites. Le but étant de développer une couche d’abstraction sur l’intégralité de l’application pour la rendre le plus flexible possible.
Nous pouvons donc voir la classe principale avec toutes les méthodes. Cette classe utilise le **patron de conception « Patron de méthode »**. Cette classe définit le squelette d’une grille et de ces algorithmes, qui devront être redéfinis dans une autre classe. Ce patron nous permet de redéfinir dans des sous-classes différentes parties des algorithmes, nous permettant ainsi de créer différentes grilles avec différents comportements.

La grille que nous avons créée dispose uniquement de la méthode gameOver(). Les autres méthodes ont été redistribuées dans 2 autres classes, de manière à respecter le principe S.O.L.I.D, S (Single Responsability principle). Ce principe dit qu'une classe doit avoir une seule responsabilitée. Nous avons donc la classe ControlesPiece, qui a donc la responsabilité des différents contrôles concernant une pièce au sein de la grille. Il y a aussi la classe ControlesGrille, qui a la responsabilité des contrôles de la grille.

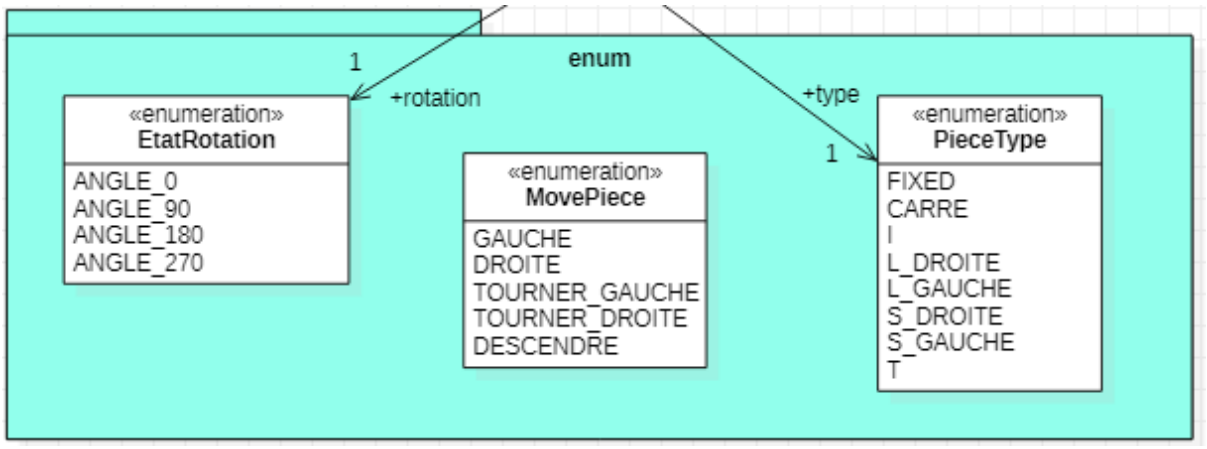
Package pièce



La package "pièce" regroupe toutes les classes concernant directement une pièce. La classe principale de ce package est la classe abstraite *PieceBase*. Cette classe repose elle aussi sur le **patron de conception « Patron de méthode »**. Elle repose sur la même stratégie que la classe *GrilleBase*. Le principe S.O.L.I.D, S (Single Responsability principle) est lui aussi respecté. Une pièce est composée avant tout de points, car pour faire une pièce, il nous faut différents points. Notre classe est donc composée de plusieurs PointBase, se trouvant dans le package point, qui est une classe abstraite, dans un souci de respect du principe S.O.L.I.D, O (Open/Close principle). Elle est aussi composée de la classe Verifyer, qui a la responsabilité de faire des vérifications sur la pièce. Ensuite, la classe DeplaceurBase, se trouvant dans

le package deplaceur. Cette classe a la responsabilité de déplacer une pièce au sein d'une grille. Pour finir, la pièce est composée d'un *TourneurBase* (package tourneur), qui est elle aussi abstraite pour respecter le principe S.O.L.I.D, S (Single Responsability principle). Nous pouvons voir que différentes classes héritent de cette classe abstraite. Nous mettons ici en place un **patron de conception « Stratégie »**. Il est intéressant de l'utiliser ici car il y a uniquement un changement au niveau des comportements des différentes méthodes de la classe *TourneurBase*. Nous avons besoin de différents comportements car chaque pièce étant agencée de différente manière les unes des autres, elles auront donc besoin d'un comportement spécifique à chaque fois.
Pour finir, nous ne pouvons pas définir une *PieceBase* car c'est une classe abstraite. C'est donc ici que nous voyons que 7 classes différentes héritent la classe *PieceBase*. On voit donc apparaître une seconde fois, un **patron de conception « Stratégie »**. Chacune des pièces se diffèrent uniquement par leur comportement, c'est donc le patron adéquat ici. Ce patron nous permet de redéfinir les algorithmes de création des différentes pièces. Nous pouvons donc créer différentes pièces et les manipuler toutes de la même manière.

Package enum



Le package enum nous permet simplement de regrouper les différents enum nécessaires au bon fonctionnement de l'application. Nous avons l'enum EtatRotation, permettant d'énumérer les différents angles de rotation que peut avoir une pièce. L'enum MovePiece permet d'énumérer les différents mouvements possibles sur une pièce. Pour finir, l'enum PieceType est une énumération des différents types de pièce, le type FIXED étant destiné aux pièces ne pouvant plus bouger sur la grille. Tous les autres types étant les types des 7 différentes pièces.