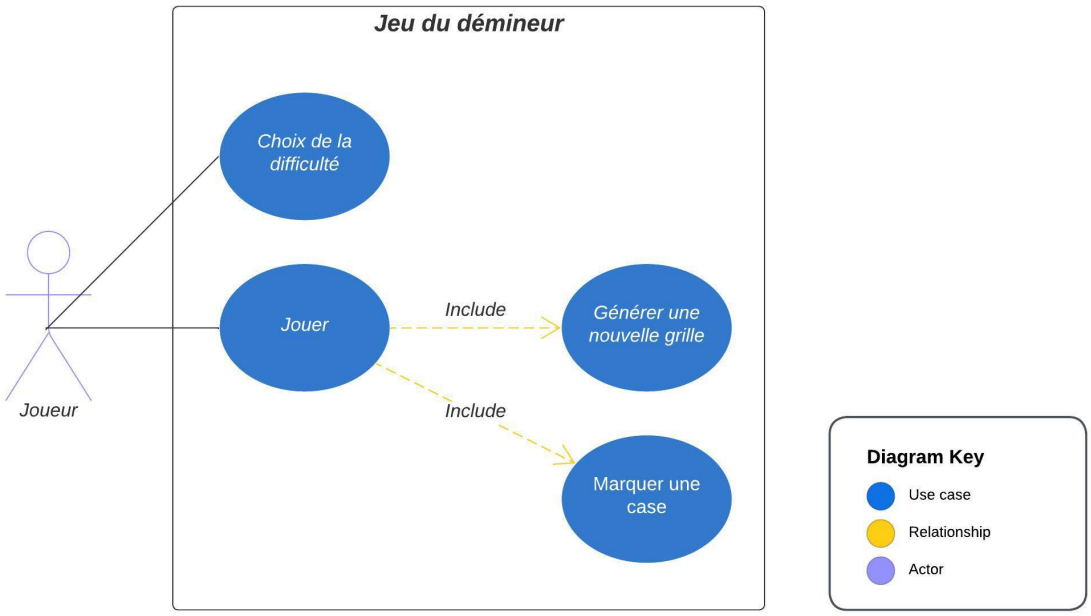
Projet démineur en orienté-objet sur Python

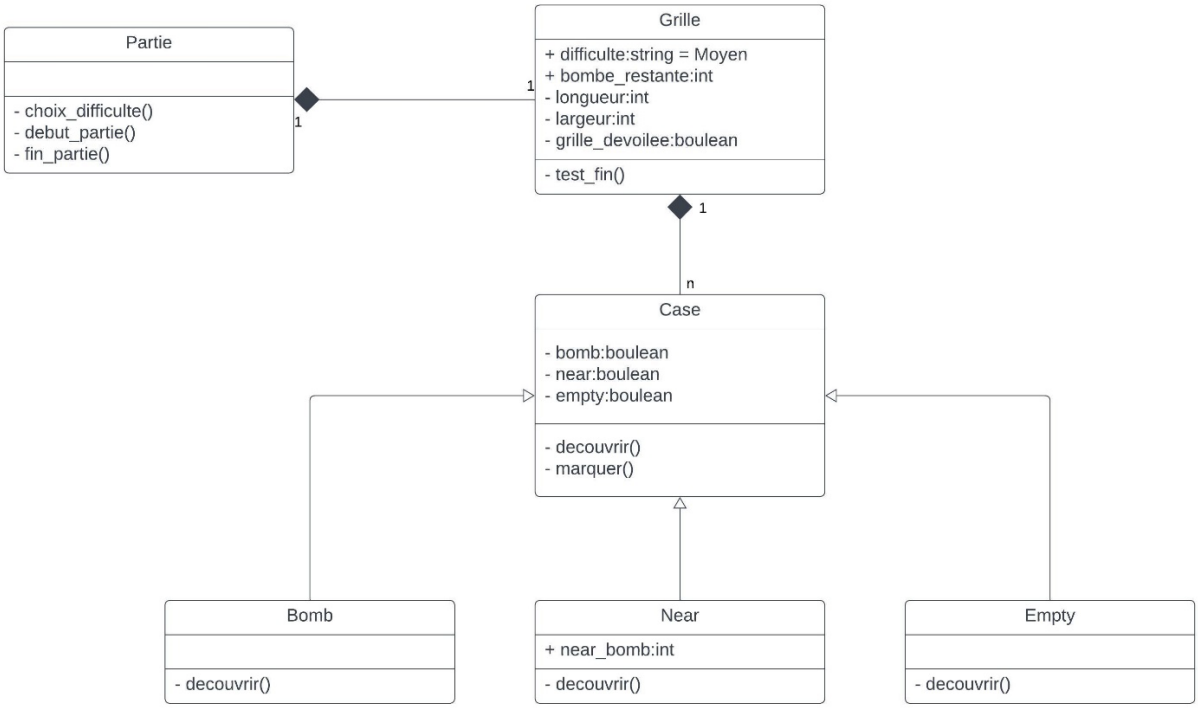
Analyse

Diagramme de cas d'utilisation :



Le diagramme de cas d'utilisation présente notre système avec les principales fonctionnalités attendu. Notre système est une application permettant de jouer du démineur. Le seul acteur est donc le joueur qui va pouvoir réaliser plusieurs choses. Pour commencer, il peut choisir de démarrer une nouvelle partie et jouer au jeu. Cette action va alors générer une grille sur laquelle il sera possible pour l'utilisateur de marquer chaque case pour indiquer la présence ou non d'une bombe. Aussi, le joueur pourra choisir un niveau de difficulté (facile, moyen, difficile et personnalisé) ce qui aura un impact sur la génération de la grille. En effet cette dernière sera plus ou moins grande, avec plus ou moins de bombes en fonction de la difficulté choisit.

Diagramme de classes :

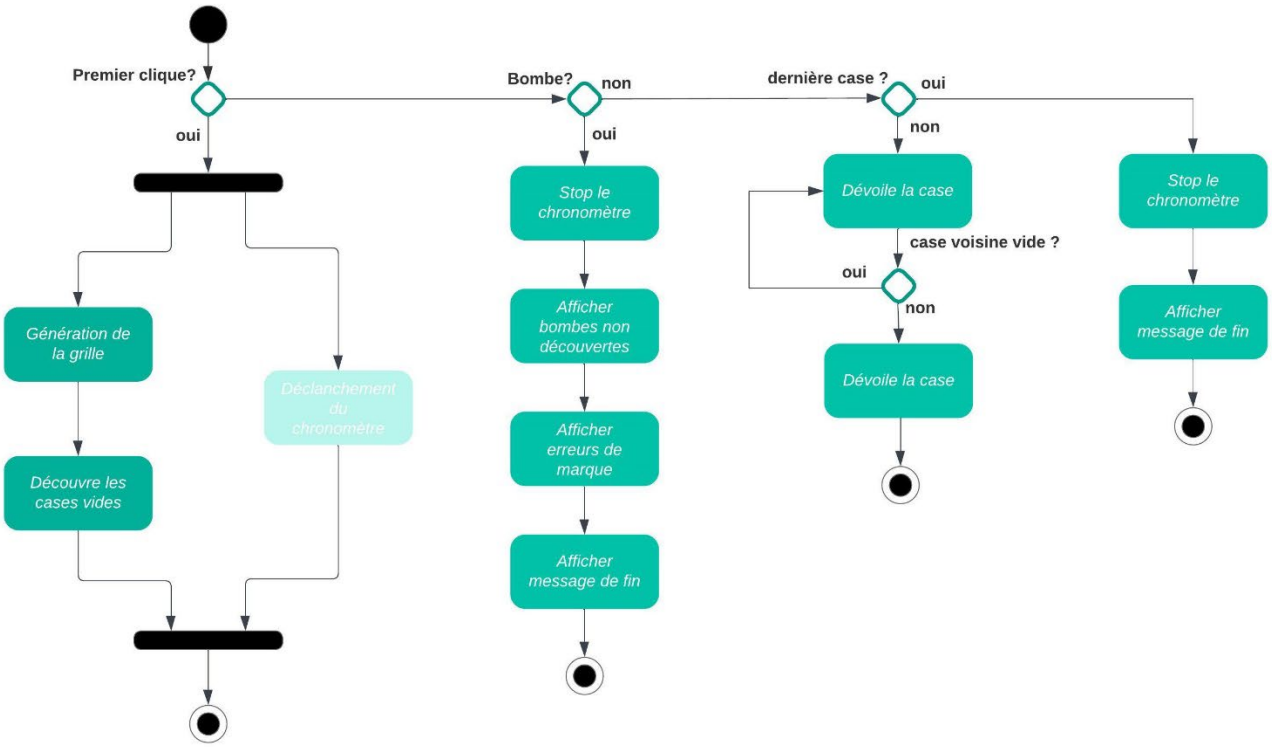


La première classe à décrire est la classe "Partie". Cette dernière contient trois fonctions. Une qui permet de choisir la difficulté de la partie, impactant ainsi la taille et le nombre de bombes contenu dans la grille générée dans la fonction suivante : "début_partie()". Cette dernière se déclenchera au premier clic sur une case de la grille. Il sera aussi possible d'ajouter un chronomètre qui se déclenchera aussi au premier clic. La dernière fonction permet de récupérer l'attribut "grille_devoilee" de la classe "Grille", qui indique si toutes les cases à découvrir ont bien été découverte, auquel cas on lance la procédure de fin notamment en proposant au joueur de commencer une nouvelle partie.

Pour la classe suivante, "grille", nous retrouvons les attributs comme la longueur et la largeur de cette dernière, qui correspond au nombre de case qui constituent la grille. Aussi nous voulons afficher au joueur la difficulté sélectionné ainsi que le nombre de bombes restantes à placer. Ce dernier ne test ni n'indique les erreurs faites par le joueur, s'il a mal positionné une bombe, sans la découvrir par exemple. Ces deux paramètres sont par conséquent publics. La fonction "test_fin()" compte le nombre de case à dévoiler restantes, et fait passer l'attribut "grille_devoilee" à "True" lorsque la condition de fin est respectée.

Enfin, la grille est constituée de cases pouvant être de trois types : une bombe, une case voisine à une bombe ou enfin une case vide. Pour ces trois dernière, l'action marquer est la même : un petit drapeau indique à l'utilisateur qu'il l'a déjà marqué, et cela bloque cette dernière : on ne peut pas la découvrir tant qu'il y a un drapeau sur celle-ci. C'est pourquoi il faut aussi que la fonction marquer puisse aussi enlever un drapeau déjà placé en cas d'erreur. Cependant, la fonction "decouvrir()"dépend du type de case: pour une case vide, cette action va découvrir toutes les cases vides environnantes, découvrir une case "near" (voisine) ne fait rien d'autre que découvrir la case en question. Enfin, découvrir une case bombe mets fin au jeu, et affiche toutes les bombes restantes à découvrir et les potentiels erreurs de marques.

Diagramme d'activité :



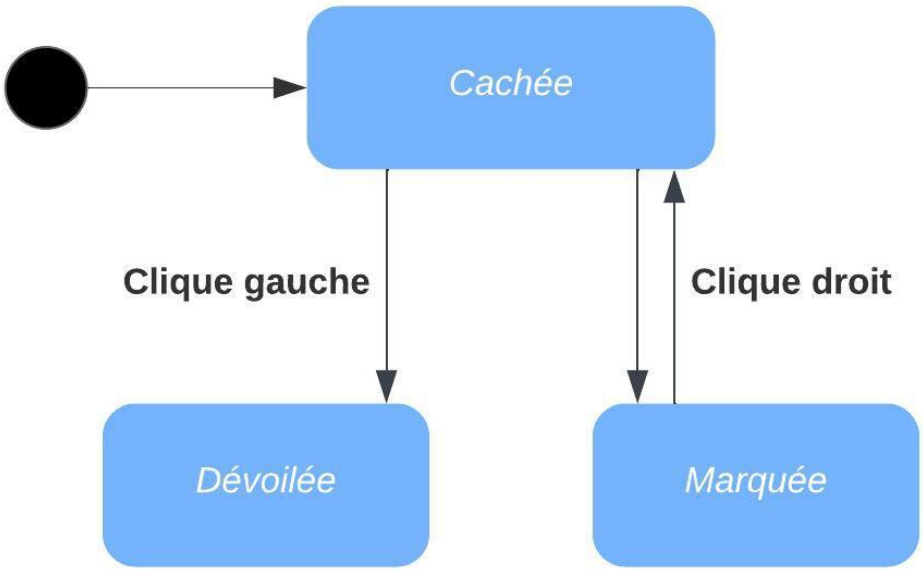
Ce diagramme d'activité présente la fonctionnalité déclenchée pour chaque clic gauche (fonction découvrir) sur une case encore non découverte.

Donc, tout d'abord on regarde si c'est le premier clic. Si oui, on génère la grille en découvrant les premières cases et on déclenche le chronomètre. Ce dernier est facultatif, et sera réalisé si nous disposons d'assez de temps pour le faire. L'objectif principal de générer la grille à cette étape est de rendre impossible de tomber sur une bombe dès le premier clic.

Si ce n'était pas le premier clic, on regarde si la case dévoilée est une bombe. Si oui, alors le jeu est perdu. On arrête donc le chronomètre et on affiche les bombes non marquées et les éventuelles erreurs de marquage. Finalement on affiche le message de fin qui propose au joueur de recommencer une nouvelle partie.

Si ce n'est pas une bombe on se demande si c'est la dernière case à découvrir. Si c'est le cas, alors la partie se termine. Sinon, on dévoile la case et on dévoile les cases voisines qui sont vide.

Diagramme d'états-transitions :



Ce diagramme présente les trois états possibles d'une case. Soit elle est cachée, et le joueur peut alors soit la découvrir soit la marquée pour signaler une bombe et bloquée la case. On passe d'une case cachée à dévoiler par clic gauche de la souris, puis, une fois dévoilée la case ne peut plus changer d'état. Pour une case que l'on veut marquer, il faut clic avec le clic droit de la souris. Cependant, si le joueur se rends compte qu'il a marqué à tort une case, il peut encore enlever la marque en cliquant une nouvelle fois sur le clic droit