Rapport d'analyse : Démineur

Gautier TABORDET

October 2023

1 Introduction

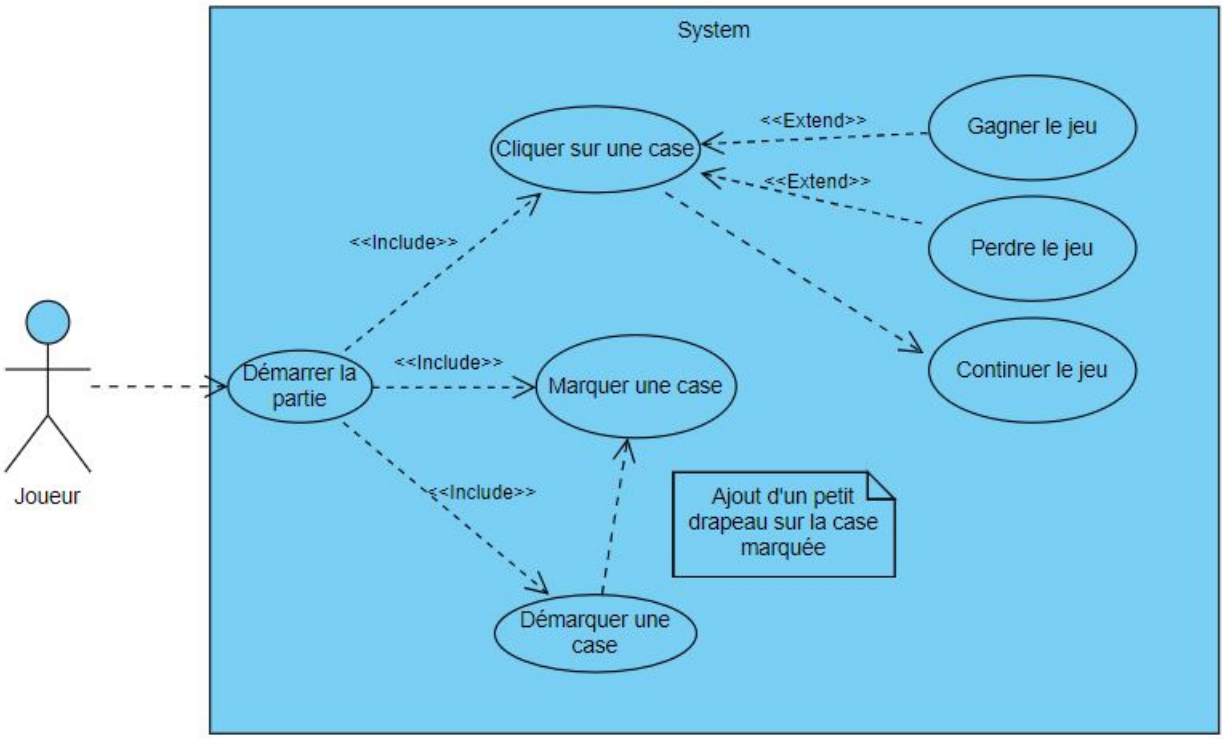
Le jeu du démineur est un jeu de réflexion classique sur ordinateur qui consiste en la recherche de mines cachées dans un champ de cases. Le joueur est présenté avec une grille de cases, certaines contenant des mines et d'autres étant vides. L'objectif est de découvrir toutes les cases vides sans cliquer sur une mine. Le jeu donne des indices sous forme de chiffres pour indiquer combien de mines se trouvent dans les cases adjacentes. Le joueur doit utiliser ces indices pour déterminer l'emplacement des mines et marquer les cases suspectes avec des drapeaux. Le jeu peut être gagné en découvrant toutes les cases vides sans faire exploser de mines, mais il peut aussi être perdu si le joueur clique sur une mine. Le démineur est un jeu de logique et de déduction qui demande de la prudence et de la stratégie pour réussir.

2 Modélisation

Avant de débuter le code, il faut modéliser l'ensemble du jeu sous forme de diagrammes UML, permettant ainsi d'accélérer la compréhension des différentes étapes à suivre lors de l'implémentation.

2.1 Diagramme d'utilisation

Le diagramme d'utilisation est un outil de modélisation qui permet de représenter comment les acteurs interagissent avec un système ou un logiciel, en montrant les fonctionnalités ou les cas d'utilisation auxquels ils accèdent.



Le joueur début sa partie, et possède 3 choix : Il peut marquer et démarquer plusieurs cases, avec l'ajout d'un symbole sur la case lorsqu'elle est marquée.

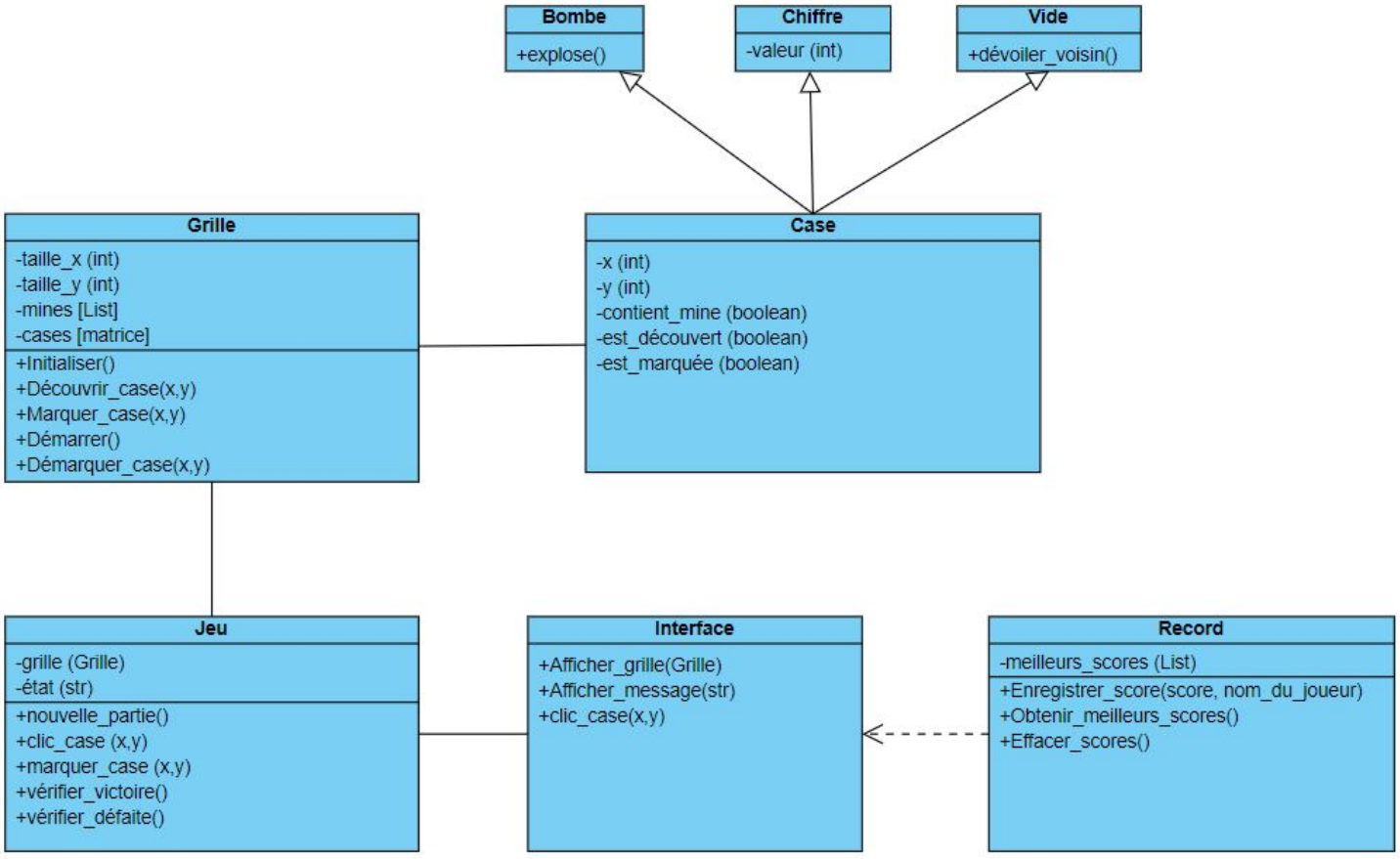
Il peut également cliquer sur une case.

Si c'est la première case à être cliquer, alors la grille se génère et l'endroit cliqué est forcément une case vide, dévoilant d'autres cases aux alentours.

Si ce n'est pas la première case, alors son contenu est dévoilé. Dès lors, soit

2.2 Digramme de classe

Le diagramme de classe est une représentation visuelle des classes d'objets, de leurs attributs et de leurs relations dans un système logiciel, aidant à comprendre la structure et l'architecture du système.



Le diagramme présent ci-dessus représente l'ensemble des classes pour le jeu du démineur. En commençant en haut, on observe une classe "mère" **Case** suivie de 3 classes "filles" : **Bombe**, **Chiffre** et **Vide**.
La classe mère **Case** prend comme paramètre ses coordonnées (int) x et y dans la grille. Egalement, chaque case est suivie de 3 boolean :

- *contient mine* : ce boolean renvoie True si cette case cache une mine.
- *est découvert* : ce boolean renvoie True si la case à déjà été découverte précédement.
- *est marquée* : ce boolean renvoie True si cette case est marquée.

Les 3 classes filles **Bombe**, **Chiffre** et **Vide** héritent de la classe mère **Case**, et ont des éléments supplémentaires :

- **Bombe** possède une méthode *+explode()* permettant de déclencher la mine, et de terminer la partie.
- **Chiffre** possède un attribut - *valeur (int)* qui correspond au nombre de mines adjacentes à cette case.
- **Vide** possède une méthode *+dévoiler voisin()* qui affiche l'ensemble des cases vides autour de cette case, sur la seule condition que les cases se touchent mutuellement.

La classe **Case** est reliée à la classe **Grille**, car la grille comporte plusieurs cases. Cette grille possède comme attributs ses dimensions, *taille x* et *taille y*. Egalement, elle possède la liste des positions des mines présentent dans cette grille.

Cette classe **Grille** possède également plusieurs méthode, comme celle permettant d'initialiser *+Initialiser()* la grille au début du jeu. On peut retrouver aussi *+Découvrir case (x,y)* afin de rendre une case découverte. Il faut une méthode *+Démarrer()* pour créer une nouvelle grille, et lancer le jeu. Enfin, on peut retrouver les méthodes *+Marquer case (x,y)* et *+Démarquer case (x,y)* afin de marquer ou démarquer la case en position (x,y).

On trouve aussi la classe **Jeu**. Cette classe possède comme attributs une *grille (Grille)* et un *état (str)*. Cette classe a également plusieurs méthodes comme *+nouvelle partie()* afin de démarrer une nouvelle partie, *+clic case (x,y)* pour cliquer sur une case en position (x,y). On retrouve également *+marquer case (x,y)* pour mettre un marqueur sur une case en position (x,y). Enfin, la classe **Jeu** contient 2 méthodes *+vérifier victoire()* et *+vérifier défaite()* pour lancer la vérification de victoire ou de défaite sur la partie en cours.

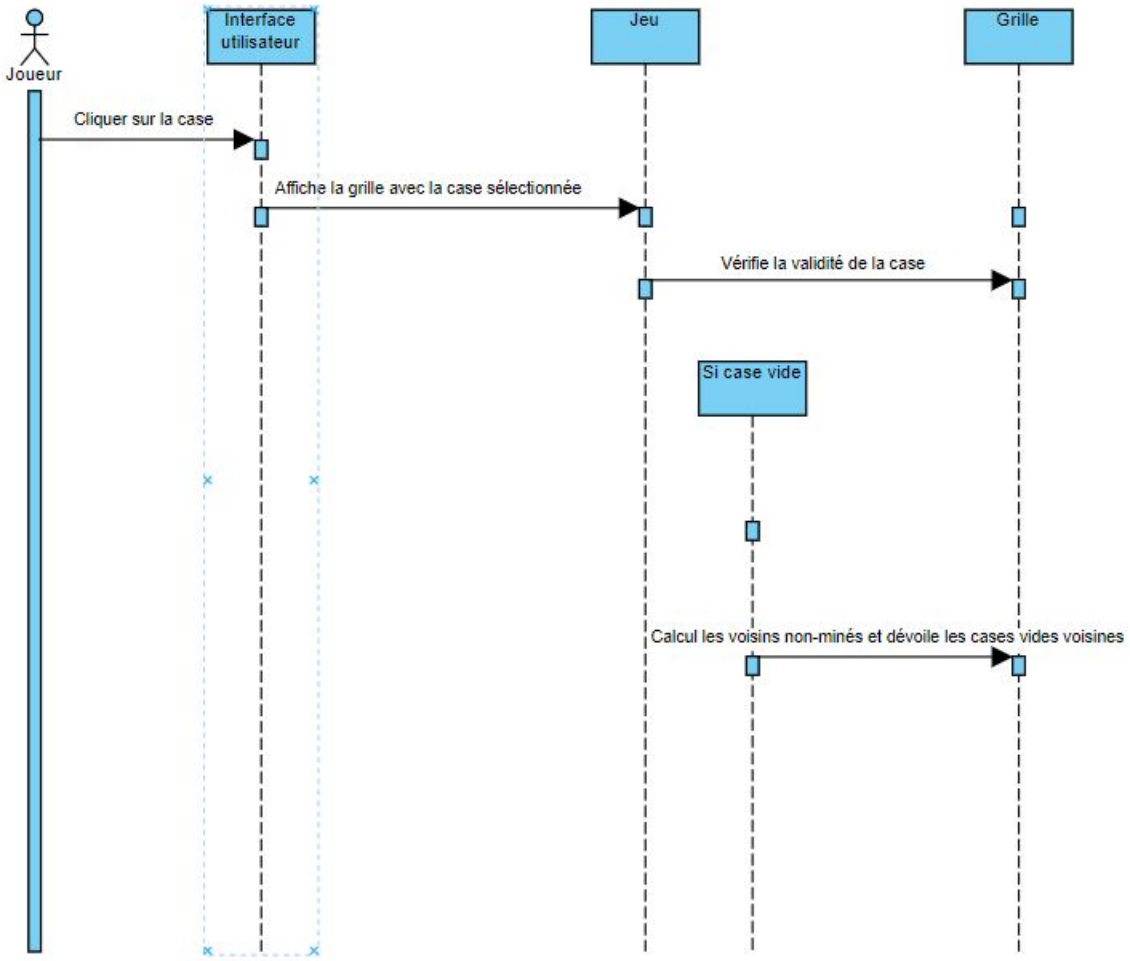
J'ai intégré également la classe **Interface** dans ce diagramme de classe. Cette classe permet d'abord d'afficher la grille sur l'interface grâce à la méthode *+afficher grille(Grille)*. De plus, elle permet d'afficher des messages, comme "Victoire !" ou "Défaite :(" grâce à la méthode *Afficher message (str)*. Enfin, on peut cliquer sur une case à travers l'interface graphique *+clic case (x,y)*. Cette méthode est à retravailler, car elle est déjà présente dans la classe **Grille**.

Pour finir, la classe **Record** est une extension de la classe **Interface** et permet, en autre, d'afficher un tableau des scores à la fin de la partie, une sorte de "hall of fame". Pour cela, il nous faut une liste *meilleurs scores (list)* comme attribut, contentant l'ensemble des pseudos et score des joueurs. Pour remplir ce tableau, on a donc 3 méthodes : *+Enregistrer score (score, nom du joueur)* pour enregistrer dans la liste les informations du joueur, *+Obtenir meilleurs scores()* pour sélectionner les 10 meilleurs scores à afficher par exemple, et *+Effacer scores()* afin de réinitialiser la liste des meilleurs scores.

Ce diagramme résume l'essentiel des classes du démineur, cependant il risque d'être modifié par la suite lors de l'implémentation du jeu.

2.3 Digramme de séquence

Le diagramme de séquence illustre la séquence temporelle des messages échangés entre les objets dans un système logiciel, permettant de visualiser le déroulement des interactions entre les composants du système.



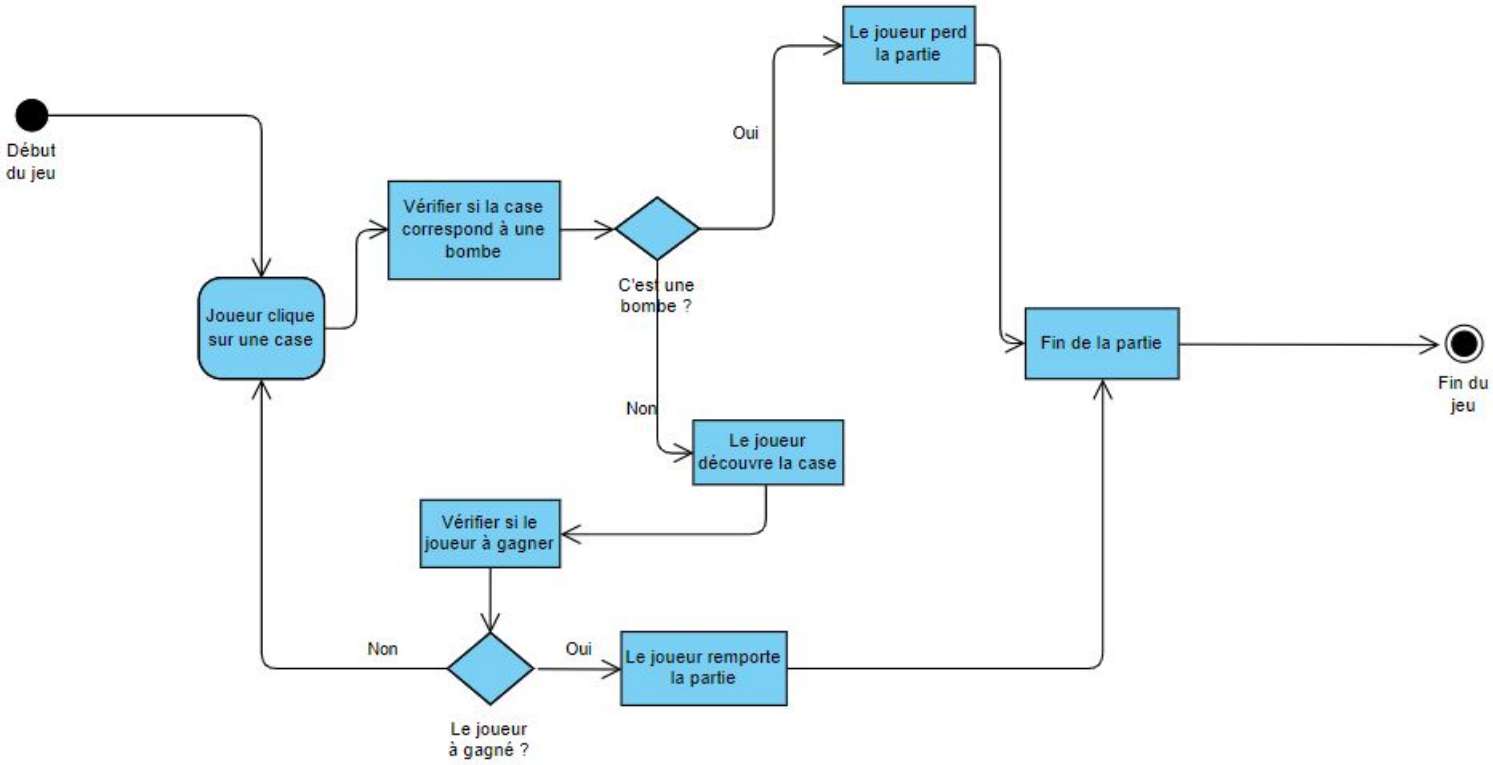
Le joueur commence par cliquer sur une case vide dans la grille pour la découvrir.

L'interface utilisateur réagit en affichant la grille mise à jour avec la case découverte. Le jeu calcule les voisins non-minés de la case découverte et dévoile les cases vides voisines.

L'interface utilisateur réagit en affichant les cases découvertes voisines. La séquence se termine, et le joueur peut continuer à jouer ou à effectuer d'autres actions, comme marquer/démarquer une case.

2.4 Digramme d'activité

Le diagramme d'activité est utilisé pour modéliser le flux d'activités, de décisions et de transitions dans un processus, permettant de visualiser les étapes et les interactions entre les éléments du système ou du processus.



Le jeu démarre lorsque le joueur clique sur une case pour la première fois. Le système vérifie si la case sélectionnée est une bombe ou non.
Si la case est une bombe, le joueur perd la partie, et le jeu se termine.
Si la case n'est pas une bombe, le joueur découvre la case, et le jeu continue.

Après chaque découverte de case, le jeu vérifie si le joueur a gagné en vérifiant s'il a découvert toutes les cases non piégées.

Si le joueur a gagné, le jeu affiche un message de victoire, et la partie se termine.

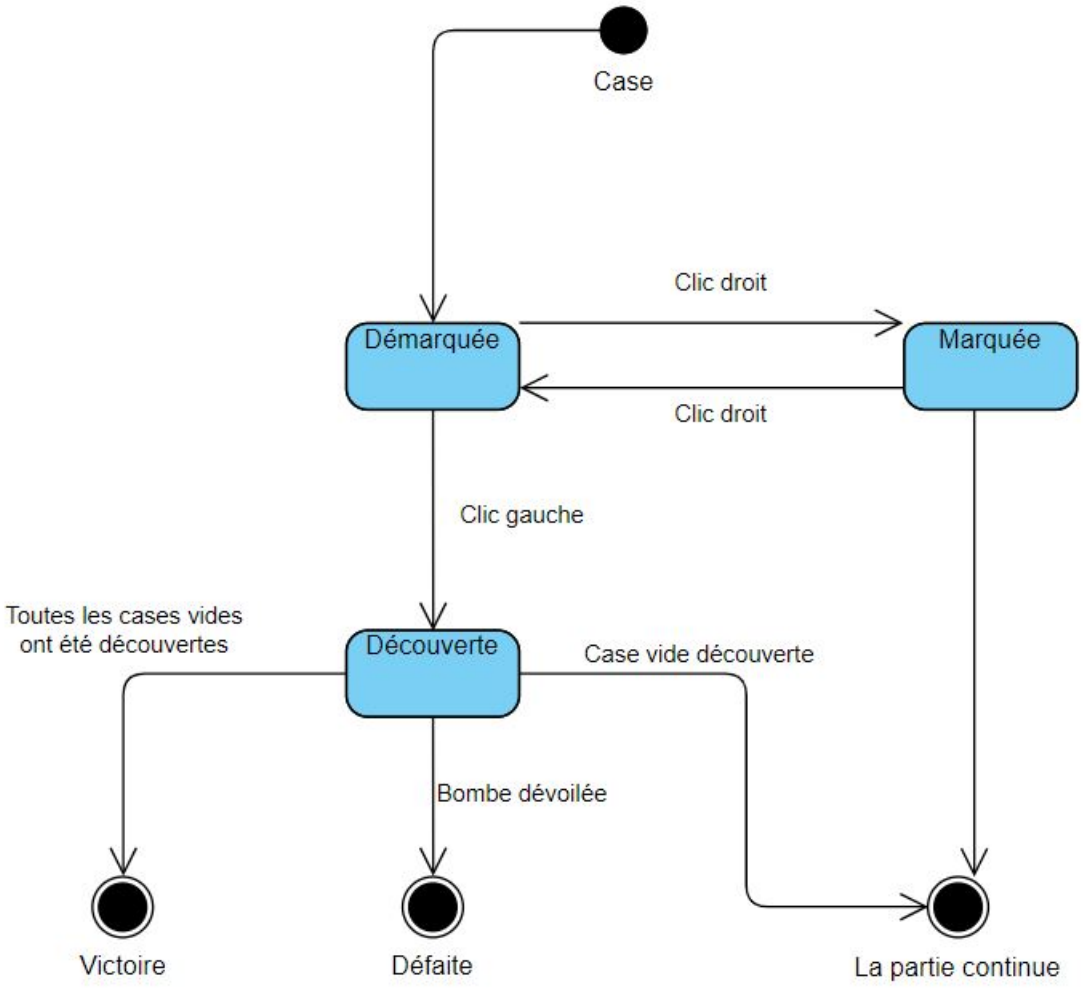
Si le joueur n'a pas encore gagné, le jeu continue jusqu'à ce que toutes les cases non piégées soient découvertes ou qu'une bombe soit découverte, entraînant la fin de la partie.

Dans ce diagramme, il est possible d'ajouter les 2 conditions suivantes en dessous de la vérification "Non" si la case est une bombe :

- La case est un chiffre ? Alors afficher le chiffre, puis vérifier la condition de victoire
- La case est vide ? Alors afficher toutes les cases vides aux alentours, et proposer à nouveau au joueur de cliquer sur une case.

2.5 Digramme d'état-transition

Le diagramme d'état-transition représente les différents états qu'un objet ou un système peut prendre, ainsi que les transitions entre ces états en réponse à des événements, permettant de modéliser le comportement dynamique d'un élément du système.



Dès le début du jeu, l'ensemble des cases sont démasquées. On a donc le choix entre masquer la case avec un clique droit, et la découvrir avec un clic gauche.

Si la case est masquée, alors on peut la démasquer avec de nouveau un clic droit.

Si la case est découverte avec un clic gauche, alors 3 états finaux sont possibles :

- Soit la case découverte est vide, alors le jeu continue
- Soit la case découverte est une bombe, alors c'est une défaite
- Soit la case découverte était la dernière case vide à être découverte, alors

toutes les cases vides ont été découvertes, et donc c'est une victoire.

3 Conclusion