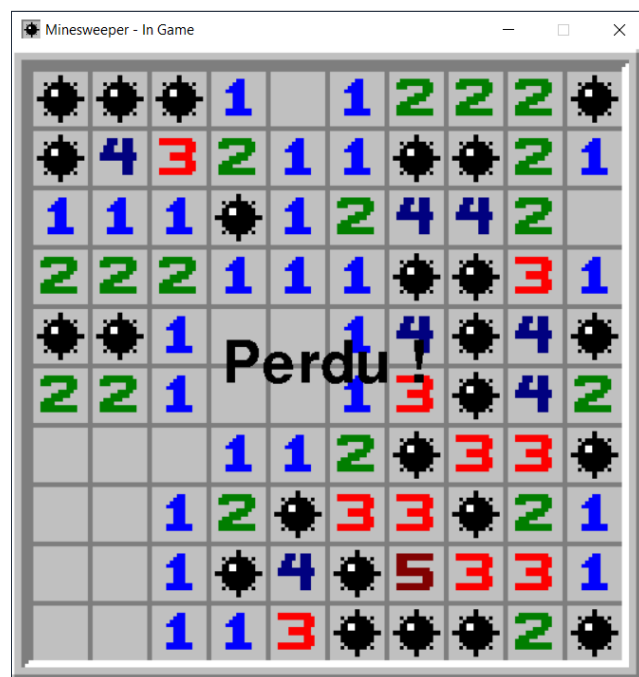


RAPPORT - PROJET TUTORÉ

Démineur



Alan GARCIA CALZADA, Logan DELPORTE, Jean DELMAS
DUT Informatique 2ème année

Sommaire

Sommaire	2
Présentation du projet	3
Cahier des charges	4
1ère Phase	4
2ème Phase	5
3ème Phase	5
Organisation du projet	6
Calendrier	6
Conception du projet	7
Bilan	9

Présentation du projet

Le but du projet est de réaliser une réplique du célèbre jeu du démineur avec une petite particularité : faire un démineur "intelligent"

Le but du jeu est de trouver des bombes sur un tableau à l'aide d'indications sur certaines cases dévoilées. On dévoile le contenu d'une case en cliquant dessus et les indications obtenues sont le nombre de bombes placées à côté de cette dernière (diagonales incluses). Lorsque l'on clique sur une case contenant une bombe, la partie est terminée.

La particularité de notre démineur "intelligent" est qu'il s'adapte à ce que fait le joueur. En effet, si le joueur clique au hasard (c'est à dire qu'il clique sur une case où il n'y a aucune indication à proximité alors qu'il pouvait jouer ailleurs), alors une bombe sera placée à cet endroit, signifiant la fin de la partie (dans la limite du nombre de bombes non indiquées restantes).

A l'inverse, si le joueur se trouve dans une situation ambiguë (d'après les indications fournies au joueur, la bombe peut se situer sur 2 cases différentes), alors le jeu aidera le joueur en déplaçant la bombe sur l'autre case.

Cahier des charges

1ère Phase

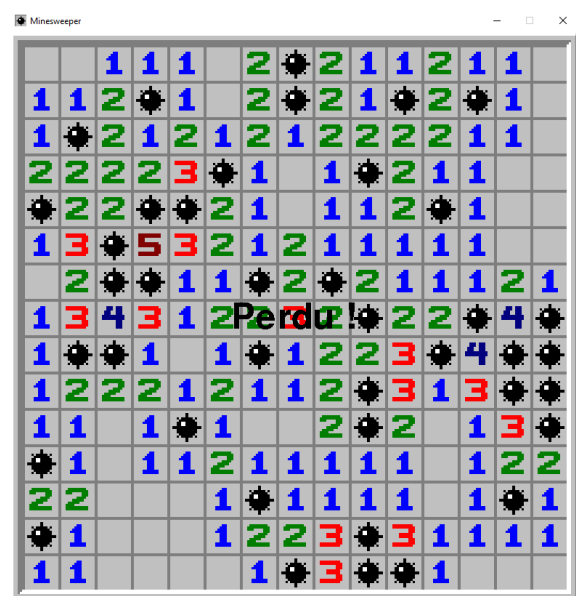
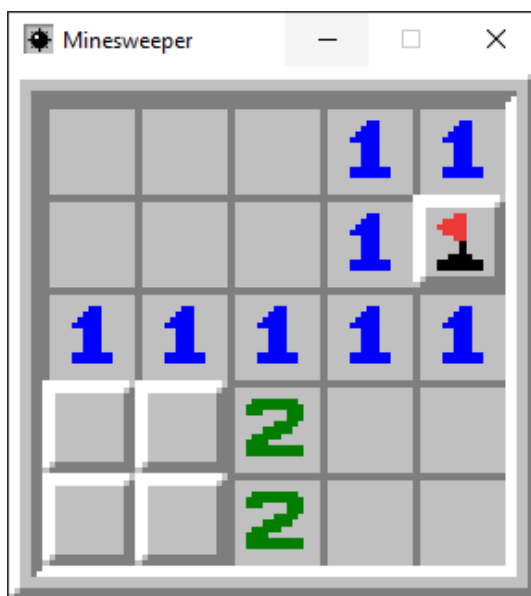
Le but de cette première phase est de recréer le jeu de base du démineur à l'aide du langage Python et de la librairie PyGame.

Le jeu doit pouvoir créer une grille d'une taille et d'un nombre de bombes rentrés en paramètre.

Les bombes doivent être placées aléatoirement en début de partie. Lorsque le joueur clique sur une case qui n'est pas une bombe, elle doit afficher le nombre de bombes à proximité.

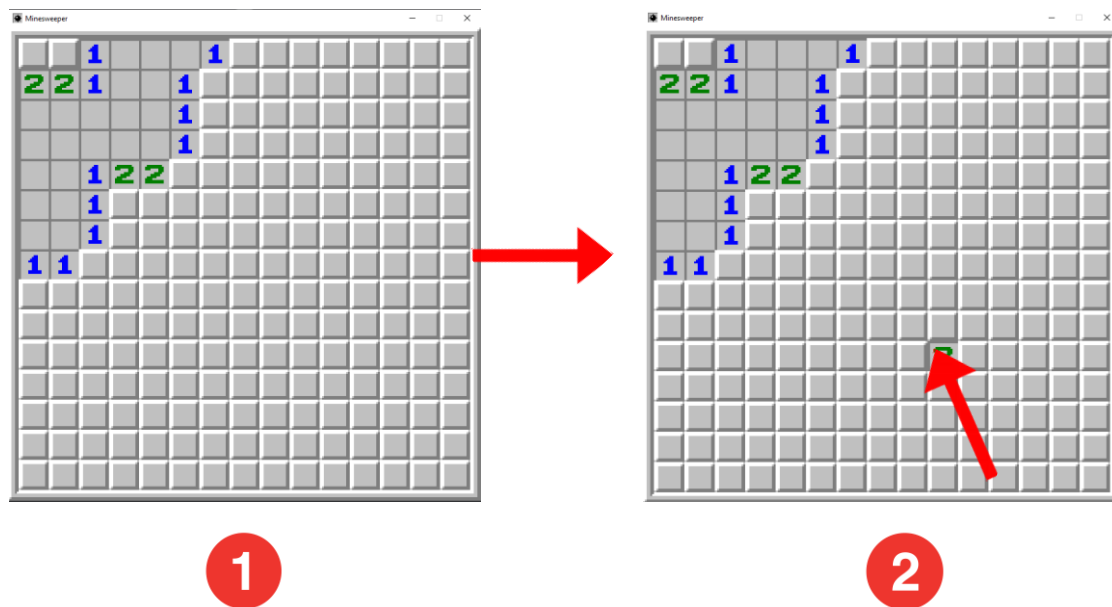
La partie doit se terminer s'il ne reste plus de cases non activées autre que les bombes (partie gagnée) ou si le joueur clique sur une bombe (partie perdue).

Interface :



2ème Phase

La 2ème phase consistera en l'implémentation d'un algorithme **exhaustif** qui teste toutes les positions de bombes possible (en se basant uniquement sur les informations accessibles par l'utilisateur évidemment). Cet algo aura pour but de déterminer si une bombe est **obligatoirement** présente sur une case (x, y) et s'il **est possible** (cf. *Partie 1*) qu'une bombe soit présente sur une case (x, y)



Ici, le joueur qui a cliqué "au hasard", sera puni, et le jeu s'arrêtera.

3ème Phase

La 3ème phase a pour but de passer au niveau suivant et implémenter un SAT solver dans un algorithme afin de rendre le programme plus rapide qu'avec l'utilisation de l'algorithme exhaustif précédemment réalisé.

C'est également lors de cette phase qu'aura lieu le rendu final du code ainsi que de la documentation liée au projet.

Organisation du projet

Nous n'avons pas décidé d'une répartition des tâches fixes à cause de problèmes de communications et d'organisation du projet. Cependant Logan a réfléchi à ce qui était demandé ainsi que les enjeux et aboutissants, Jean a réinterprété et réexpliqué ce qui était demandé pour bien comprendre ce qu'il fallait faire et Alan a codé le jeu tout seul et établi un schéma du type MVC dans un premier temps puis avec l'aide du groupe a implémenté les différentes fonctions et manipulation de données. Bien évidemment, et surtout quand tout le monde était présent, nous avons pu discuter et essayer de trouver des solutions à nos problèmes. Enfin, le rapport et le Powerpoint de présentation ont été effectués par Logan et Alan.

Calendrier

Octobre 2020							
n°	Lu	Ma	Me	Je	Ve	Sa	Di
40				1	2	3	4
41	5	6	7	8	9	10	11
42	12	13	14	15	16	17	18
43	19	20	21	22	23	24	25
44	26	27	28	29	30	31	

Novembre 2020							
n°	Lu	Ma	Me	Je	Ve	Sa	Di
44							1
45	2	3	4	5	6	7	8
46	9	10	11	12	13	14	15
47	16	17	18	19	20	21	22
48	23	24	25	26	27	28	29
49	30						

Décembre 2020							
n°	Lu	Ma	Me	Je	Ve	Sa	Di
49		1	2	3	4	5	6
50	7	8	9	10	11	12	13
51	14	15	16	17	18	19	20
52	21	22	23	24	25	26	27
53	28	29	30	31			








Janvier 2021							
n°	Lu	Ma	Me	Je	Ve	Sa	Di
53					1	2	3
1	4	5	6	7	8	9	10
2	11	12	13	14	15	16	17
3	18	19	20	21	22	23	24
4	25	26	27	28	29	30	31

Février 2021							
n°	Lu	Ma	Me	Je	Ve	Sa	Di
5	1	2	3	4	5	6	7
6	8	9	10	11	12	13	14
7	15	16	17	18	19	20	21
8	22	23	24	25	26	27	28

Mars 2021							
n°	Lu	Ma	Me	Je	Ve	Sa	Di
9	1	2	3	4	5	6	7
10	8	9	10	11	12	13	14
11	15	16	17	18	19	20	21
12	22	23	24	25	26	27	28
13	29	30	31				

Avril 2021							
n°	Lu	Ma	Me	Je	Ve	Sa	Di
13				1	2	3	4
14	5	6	7	8	9	10	11
15	12	13	14	15	16	17	18
16	19	20	21	22	23	24	25
17	26	27	28	29	30		

Mai 2021							
n°	Lu	Ma	Me	Je	Ve	Sa	Di
17						1	2
18	3	4	5	6	7	8	9
19	10	11	12	13	14	15	16
20	17	18	19	20	21	22	23
21	24	25	26	27	28	29	30
22	31						

Analyse projet : 
 Réalisation phase 1 : 
 Débug phase 1 : 
 Analyse phase 2 : 
 Réalisation phase 2 : 
 Débug phase 2 : 
 Phase 3 : 

Conception du projet

Pour l'affichage graphique du jeu nous avons opté pour PyGame car c'est une bibliothèque Python reprenant la bibliothèque SDL du langage C (permettant alors d'être plus performant en plus d'être bien documentée sur internet).

De plus, la partie "affichage" du jeu ne fait que reprendre les informations du plateau et de ses cases et ne fait que demander l'activation de la case aux coordonnées calculées en fonction de la position du clic du joueur.

Afin de représenter un jeu de démineur, nous créons une grille sous la forme d'une matrice. La taille de cette dernière ainsi que le nombre de bombes est saisie par le joueur avant le début de la partie. Les bombes sont ensuite placées aléatoirement en fonction du nombre de bombes à placer.

Chaque case peut être de plusieurs types :

- ACTIVATED : le joueur a cliqué sur la case et sait donc ce qu'il y a dedans (affiche le nombre de cases piégées à proximité), ne peut pas être désactivée
- INACTIVATED : le joueur ne sait pas encore ce qu'il y a dans cette case (il n'a pas cliqué dessus)
- NO_BOMB : la case ne contient pas de bombe
- BOMB : la case contient une bombe
- FORBIDDEN : il est impossible de poser une bombe sur cette case

Toutes les cases sont inactivées au démarrage et deviennent activées si le joueur clique dessus en changeant de type comme vu avant. Elle est vérifiée avant de l'activer si elle ne l'est pas déjà ou bien si elle n'est pas marquée d'un drapeau (il s'agit d'une variable liée à la case), rien ne se passera dans ces cas.

Pour vérifier si le joueur avait un meilleur choix à faire, nous utilisons un objet (nommé "Board"), contenant deux tableaux, appelé à chaque clic. ces tableaux sont:

- user_view : état actuel de ce que le joueur voit de la matrice
- bomb_view : état actuel de la matrice dont ce que le joueur ne voit pas (emplacement des bombes)

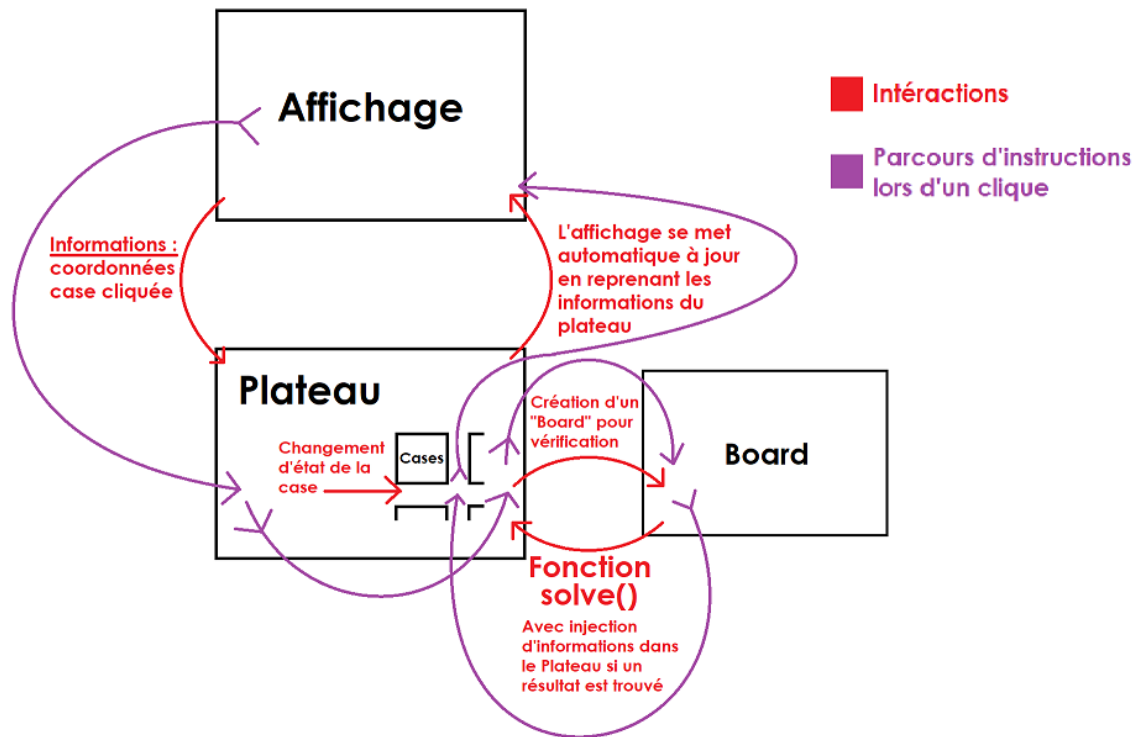


Schéma du fonctionnement du programme

Lorsque le joueur clique sur une case, les coordonnées de la case cliquée sont calculées et l'affichage demande au Plateau d'activer la case aux dites coordonnées.

Lors de l'activation, on teste d'abord si la case avait obligatoirement une bombe (s'il est évident que telle case ait ou n'ait pas une bombe) puis s'il y a une ambiguïté, alors on passe au niveau supérieur.

Un objet *board* est créé afin de vérifier si les bombes peuvent être placées autrement à l'aide de sa fonction *solve()*, puis on injecte les nouvelles données dans le plateau. La case changée est maintenant activée avec l'effet qu'elle aura. L'affichage se met à jour automatiquement depuis les données du Plateau.

Bilan

Bien que le projet n'a pas pu être mené à terme correctement, ce dernier nous a permis de voir ce qui n'allait pas dans notre organisation.

En effet, nous aurions gagné à avoir une meilleure gestion du temps et surtout une meilleure communication (il a été difficile de réunir tout le monde afin de faire le point régulièrement).

Une autre difficulté rencontrée est que nous avons eu beaucoup de mal à cerner ce qu'il fallait faire avec les SAT solvers. Cela ajouté à une mauvaise gestion du temps fait que nous ne sommes pas parvenus à ajouter les SAT solvers à notre programme.