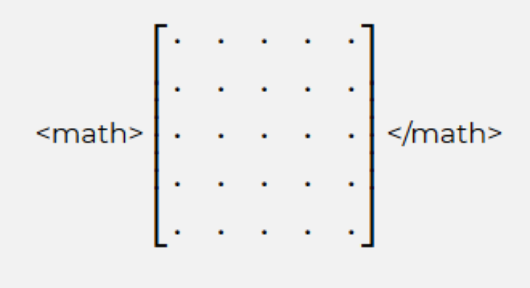
Rappel des règles du jeu

Pour ceux qui ne connaitraient pas encore ce petit jeu aussi simple qu'addictif, je vous en rappelle rapidement le principe.
Vous incarnez un agent de l'équipe de déminage chargé de sécuriser un champ de mines anti-personnelles (eh oui, malheureusement, ces horreurs existent encore à notre époque), en disposant des petits drapeaux « Attention, ne marchez pas ici, ça va vous péter à la tronche » sur chacune d'entre elles.

Le champ de mines se présente sous la forme d'un **tableau à L lignes et C colonnes**. Chacune des cases de ce tableau peut contenir une mine ou non. Au début de la partie, toutes les cases sont masquées (il y a de la terre partout).
La suite des explications prendra en exemple la petite matrice suivante à 5 lignes et 5 colonnes.



Dans le jeu, vous n'avez que deux actions possibles. **Creuser** ou **poser un drapeau**.

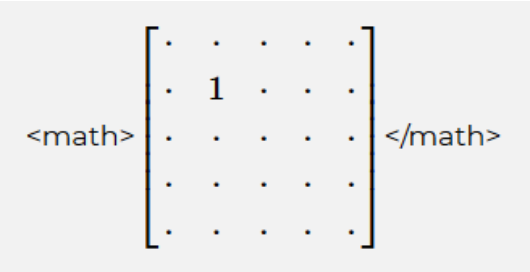
Creuser

Creuser constitue l'action principale du jeu : c'est la seule action indispensable pour terminer la partie, et elle est irréversible.
Lorsque vous creusez sur une case du tableau, il n'y a que deux possibilités :

- **La case contient une mine**, elle vous pète à la tronche et la partie est terminée
- **La case ne contient pas de mine**, et elle vous indique le nombre de cases adjacentes qui contiennent une mine.

Ici, on travaille en 8-connexité, c'est-à-dire que chaque case a **8 cases adjacentes** : au-dessus, en-dessous, à gauche, à droite, au-dessus à gauche, au-dessus à droite, en-dessous à gauche, et en-dessous à droite d'elle.

Par exemple, je décide de commencer la partie en creusant la case se trouvant à la seconde ligne, seconde colonne de la matrice (donc de coordonnées (1, 1)):

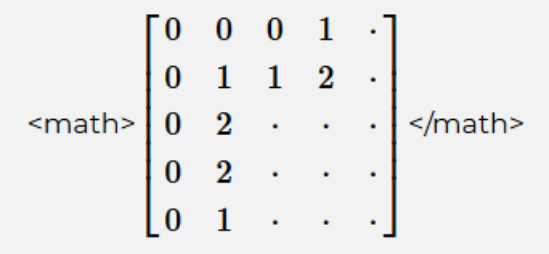


Ouf! Pas de mine, à cette position, je suis toujours en vie, et la case m'indique qu'il se trouve une mine (et une seule) dans son voisinage direct...

Un petit raccourcis

Pour éviter que le jeu ne soit trop répétitif, on ajoute une petite règle très pratique. Si la case où l'on a creusé ne contient aucune mine dans son voisinage (une case 0), alors le jeu creuse automatiquement sur toutes les cases autour d'elle, ce qui permet de dégager les grands espaces libres d'un seul coup.

Par exemple, je tente ma chance, et je creuse maintenant au-dessus à gauche de la case précédente (position (0,0), donc).



Cette case ne contenait pas de mine dans son voisinage direct, le jeu a donc creusé automatiquement partout autour d'elle, ce qui, récursivement, a découvert tout un champ libre.

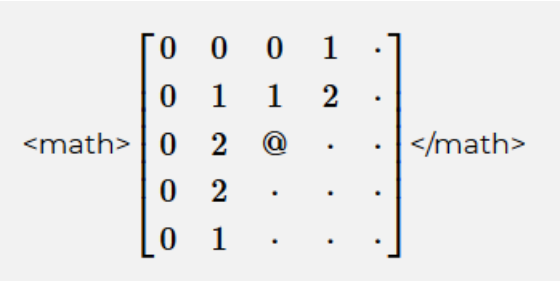
La fin de la partie

Pour terminer la partie (et gagner), il faut et il suffit que **toutes les cases du tableau ne contenant pas de mine aient été creusées**.

Poser un drapeau

La seconde action possible dans le jeu, **poser un drapeau**, n'est absolument pas indispensable pour jouer, mais elle constitue une sécurité pour le joueur (ce qui est extrêmement pratique lorsque l'on joue avec un grand champ de 1000 cases). Lorsque l'on pose un drapeau sur une case, celle-ci affiche un symbole "attention", et on ne peut plus creuser dessus tant que le drapeau s'y trouve. Cela évite donc de se faire péter une mine à la tronche par erreur alors que l'on avait deviné qu'elle était là.

Par exemple, je suis sûr que la case (2,2) contient une mine, je décide donc de poser un drapeau dessus (symbolisé ici par une arobase).



Ainsi, si j'essayais par erreur de creuser cette case, le programme l'ignorerait. C'est une sécurité.
Le fait de poser un drapeau doit être RÉVERSIBLE.

En effet, on peut tout à fait poser un drapeau sur une case parce où l'on soupçonne la présence d'une mine à un moment donné, mais on doit pouvoir aussi retirer ce drapeau par la suite si l'on en a envie.

Consignes

Je vous propose de programmer ce jeu en plusieurs phases, de manière à ce que les débutants ne se perdent pas dès le départ, et que vous ayez la possibilité d'**apprendre** des choses avec ce mini-projet quel que soit votre niveau.

Phase 1 : votre base de travail.

Dans un premier temps, il va vous falloir coder le cœur du programme, à savoir **la version jouable du jeu la plus simple possible**.

Si vous débutez dans la POO, je vous recommande très chaudement de vous forcer à programmer objet pour ce projet : il est suffisamment simple pour que vous vous en sortiez avec 3 classes.

Si vous n'avez pas encore appris la POO, qu'à cela ne tienne, vous n'êtes pas obligé d'utiliser ce paradigme pour obtenir une version jouable du jeu, mais c'est peut-être une bonne occasion de s'y coller.

Pour valider cette première phase, vous devrez obtenir la version la plus simple possible, qui vous servira ensuite de base pour le reste du développement :

- **Déterminer un moyen pertinent de représenter les données** avec lesquelles vous devrez interagir. Si je rappelle cette analyse préalable obligatoire (et évidente), c'est simplement parce que cette modélisation va influencer tout le reste du mini-projet.

Plus les structures de données que vous utiliserez seront pertinentes, plus le jeu sera facile à programmer.

- **Coder la fonctionnalité creuser** de base. Comme je viens de l'expliquer dans les règles du jeu, l'action creuser est la seule indispensable pour finir la partie. C'est donc la seule requise dans cette phase. Il n'est pas non plus obligatoire de coder pour le moment la petite règle supplémentaire qui creuse automatiquement autour des casses nulles : c'est gadget, et vous pouvez tester le fonctionnement de votre programme sans ça.
- **Une interface utilisateur rudimentaire**. Dans cette première phase, **ne vous focalisez surtout pas sur l'interface**. Elle peut être moche, on s'en fout... À vrai dire elle DOIT être moche. Le but, c'est simplement que vous puissiez afficher l'état du champ (la grille du jeu), **Y COMPRIS LES MINES**, et creuser une case donnée en donnant ses numéros de ligne et numéro de colonne. Cette interface rudimentaire va vous permettre de tester votre jeu, voir si les règles sont bien programmées, qu'il n'y a pas de bug, que tout est sous contrôle. Rien de plus ! Attention cependant à ce que cette interface rudimentaire soit bien séparée du reste du programme (bien encapsulée), parce qu'évidemment, elle va changer dans la suite.
- **Une partie minimaliste**. Contentez-vous pour le moment d'un tableau à 5 lignes et 5 colonnes grand maximum, et de quelques mines (2 ou 3) (de toute façon avec l'interface "console print" ce serait trop fastidieux de faire plus). Encore une fois, votre jeu ne doit pas à ce niveau vous offrir un challenge, au contraire, les parties doivent être finies rapidement, de manière à vérifier que vous SAVEZ détecter quand une partie est perdue ou gagnée, sans passer 20 minutes à rentrer des coordonnées au clavier à chaque fois. 

Une fois cette phase achevée, vous devriez être capable d'avoir une version "jouable minimale" du démineur, avec les mines qui s'initialisent au hasard dans le champ (mais sont affichées, pour vous aider à débugger), la possibilité de creuser (et donc l'affichage de nombre de mines autour de la case dans laquelle vous venez de creuser) et la détection de la fin de la partie. Une fois que tout ceci fonctionne et est bien en place, vous pouvez passer à la phase 2.

Phase 2 : La totalité des règles du jeu.

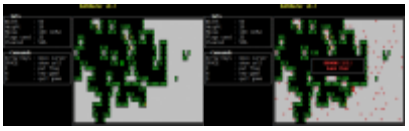
Dans cette phase, votre modèle du démineur va être complété, avec toutes les règles du jeu. Si votre code a bien été organisé dans la première partie, vous ne devriez avoir que du code à rajouter, et très peu à modifier.
Il vous faudra donc coder :

- **La fonctionnalité creuser intelligente**. Si la case que vous creusez ne contient aucune mine dans son entourage, alors vous creusez automatiquement partout autour d'elle afin de dégager les grands champs vides.
- **La pose de drapeaux**. Si un drapeau est posé sur une case, on ne peut plus la creuser, mais on peut retirer le drapeau pour creuser ensuite.
- **Pour l'interface utilisateur**. Pas de changement exceptionnel, mais comme vous avez deux actions différentes, je vous conseille (c'est le plus simple) de faire en sorte que l'interface soit « modale » : On appuie sur "D" pour passer en mode drapeau, et sur "C" pour passer en mode creusage. Le reste est identique.

Une fois que tout ceci fonctionne bien, vous pouvez masquer les mines et avoir un "vrai" jeu.

Phase 3 : l'interface

Maintenant, et seulement maintenant que le cœur du jeu est en place avec toutes les règles, vous pouvez penser à l'interface.
Ici, vous avez le choix des armes. PyQt, PyGTK, tkInter, PyGame... Profitez-en peut-être pour choisir un module ou une bibliothèque d'interface à laquelle vous désirez vous essayer, c'est un bon moyen d'apprendre.
Personnellement, en préparant ce mini-projet, je me suis souvenu que j'avais toujours voulu essayer ncurses (les interfaces console riches, pour les terminaux Unix), donc j'ai codé mon interface en ncurses avec le module **curses** de l'API standard de Python (par chance, le jeu s'y prête bien). Voici deux screenshots :



Séparez, de préférence, votre code d'interface du moteur du jeu (collez-les dans 2 modules différents).
Personnellement, avant de soumettre une proposition de correction, je vais refaire un module d'interface utilisant autre chose que ncurses, histoire d'avoir une correction portable.

Phase 4 : Lâchez-vous !

Maintenant que vous avez un démineur complet avec une belle interface, lâchez-vous, rajoutez des parties en temps limité, un système de score, une liste des highscores, des options modifiables, ou même, que sais-je ? un mode multi-joueur coopératif en ligne ou une IA qui vous donne un indice si vous êtes bloqué... Bref, amusez-vous, et venez nous montrer vos belles idées !

Bon courage !