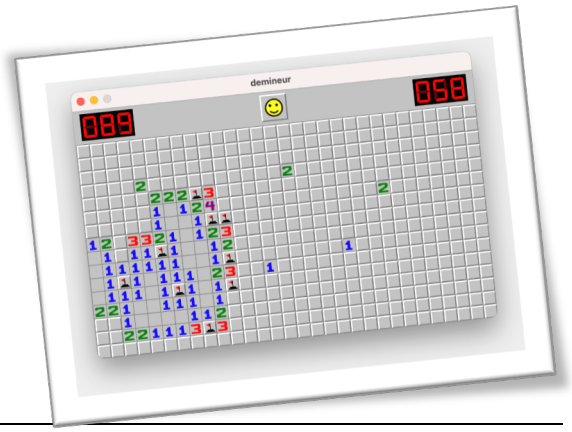
DM
Démineur



Objectif

L'objectif consiste à écrire un jeu ressemblant au démineur tel qu'il a été popularisé par Windows (https://fr.wikipedia.org/wiki/D%C3%A9mineur_(genre_de_jeu_vid%C3%A9o)) :

> *« Le Démineur (Minesweeper) est un jeu vidéo de réflexion dont le but est de localiser des mines cachées dans une grille représentant un champ de mines virtuel, avec pour seule indication le nombre de mines dans les zones adjacentes. ».*

Les dix commandements de maître Yoda

1. **Ce document, avant de commencer, tu liras jusqu'au bout**
2. Les instructions qui te guident pas à pas, tu suivras
3. Les différentes étapes de ton travail, tu conserveras
4. Ton programme, tu indenteras et commenteras
5. La signature des fonctions, tu respecteras
6. De la programmation orientée objet, tu ne feras pas
7. L'utilisation des variables globales, tu limiteras, et les variables locales préfèreras
8. Les tableaux, quand c'est possible, tu utiliseras
9. Seul, tu travailleras
10. Ce que tu fais, tu comprendras, et capable de l'expliquer, tu seras

Prise en main

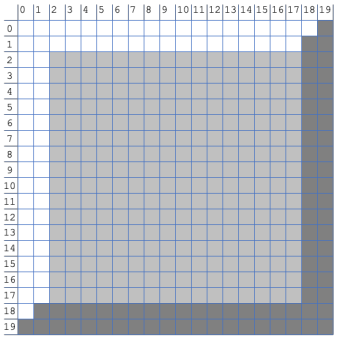
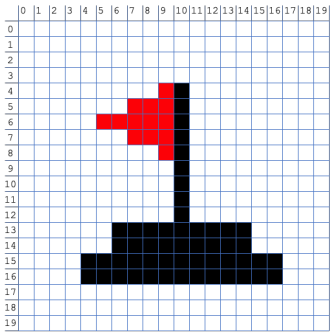
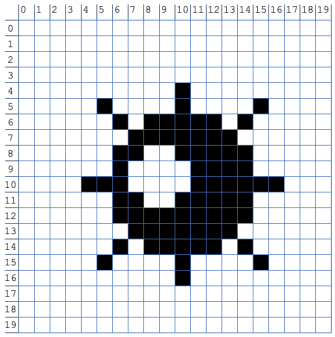
- Téléchargez l'archive demineur.zip et décompressez-là dans un dossier sur votre ordinateur.
- Renommez `demineur_v0.pde` en respectant la convention de nommage suivante : `GXX_NOM_Prenom.pde`, où `XX` doit être remplacé par :
  - `MIn` avec `n` votre n° de groupe si vous êtes dans le portail Maths-Info
  - `MPn` avec `n` votre n° de groupe si vous êtes dans le portail Maths-Physique
  - ME si vous êtes en double licence Maths-Eco
  - BI si vous êtes en licence double diplôme Info-SdV
  - DI si vous êtes en licence double diplôme Droit-Info
- Renommez de même le dossier contenant le programme : `GXX_NOM_Prenom`
- Chargez le programme dans Processing et complétez les informations vous concernant dans le bloc de commentaire en haut du programme
- Observez les différentes fonctions que vous allez devoir programmer
- Prenez connaissance du barème de notation :

$$Note_{finale} = \text{à définir...}$$

I TD1 & 2 = Dessiner les éléments du jeu

I.1. Dessiner un drapeau et une bombe

1. En utilisant les instructions de dessin (rect, point, ellipse, etc.), complétez les fonctions `drawBloc`, `drawFlag` et `drawBomb` qui doivent dessiner respectivement un bloc, un drapeau et une bombe ressemblant aux schémas ci-dessous :

Vous devez respecter l'échelle : une ligne = une colonne = 1 pixel). Pour le drapeau et la bombe, ne faites pas de fond blanc, mais dessinez uniquement l'élément en lui-même. Dans un premier temps, ne tenez pas compte des paramètres `x` et `y`.

2. Les paramètres `x` et `y` correspondent en fait aux coordonnées du bloc dans la grille.
   a. Déclarez une variable globale `cote`, que vous initialiserez à 20 dans la fonction `settings` ;
   b. Sachant que les blocs ont une dimension 20x20 pixels, il faut multiplier `x` et `y` par la valeur de la variable `cote` pour obtenir leur position dans la fenêtre graphique. Utilisez les paramètres `x` et `y` des fonctions ci-dessus et la variable `cote` pour calculer la position du dessin dans l'interface ;
   c. Sans modifier votre dessin, déplacez-le à l'endroit voulu en utilisant les fonctions `pushMatrix`, `translate` et `popMatrix`.

I.2. Happy face / sad face

Toujours en utilisant les instructions de dessin (rect, point, ellipse, etc.), complétez les fonctions `drawHappyFace`, et `drawSadFace` qui doivent dessiner respectivement un smiley content et un smiley mécontent ressemblant aux images ci-contre :  

I.3. Un début d'interface

On veut ajouter un bandeau au-dessus de la grille des blocs, de hauteur 50 pixels. C'est dans ce bandeau que s'afficheront le visage (content ou mécontent), le nombre de mines restant à localiser (à gauche), et le temps écoulé depuis le début de la résolution (à droite).

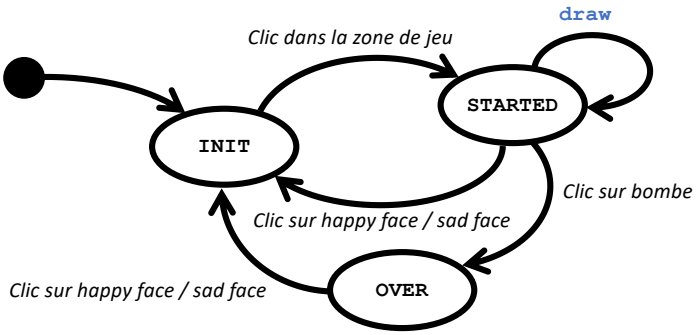
1. Déclarez une variable globale `bandeau`, que vous initialiserez à 50 dans la fonction `settings`, correspondant à la hauteur du bandeau (en pixels) ;
2. Mettez à jour les fonctions `drawBloc`, `drawFlag` et `drawBomb` pour prendre en compte ce bandeau en décalant le dessin de 50 pixels vers le bas.

3. Déclarez 2 variables `colonnes` et `lignes` définissant le nombre de cases du jeu en largeur et en hauteur ; Initialisez-les à 30 et 16 dans la fonction `settings`.
4. Déduisez-en la taille de la fenêtre graphique et appelez la fonction `size` dans la fonction `settings` pour définir la taille de la fenêtre graphique.
5. Complétez la fonction `display` en faisant un fond d'un niveau de gris à 192 puis en appelant `drawHappyFace` dessinant un smiley content au milieu du bandeau et en dessinant deux blocs au milieu de la grille, avec respectivement un drapeau ou une bombe dessus, en faisant des appels aux fonctions définies ci-dessus (voir capture d'écran ci-dessous).

II TD3 à 6 = Des variables pour la gestion du jeu

II.1. On distingue les étapes

Il est important de savoir où on en est dans le jeu (initialisation, victoire, défaite). Pour cela, on utilisera une variable dont la valeur correspondra à l'étape de jeu en cours. Les différentes étapes s'enchaîneront selon le schéma suivant (le disque noir correspond au démarrage du programme, les ellipses correspondent aux différentes étapes du jeu, les flèches correspondent aux évènements faisant passer d'un état à l'autre) :

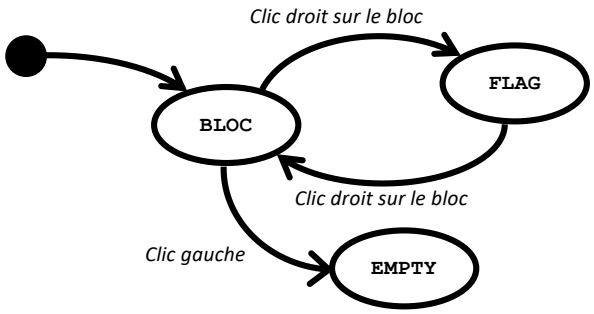


1. Déclarez une variable globale `etat`.
2. Déclarez des constantes (voir le mot clé `final`) `INIT`, `STARTED`, `OVER` pour identifier les différentes étapes possibles et attribuez-leur des valeurs entières croissantes (0 pour `INIT`, 1 pour `STARTED`, 2 pour `OVER`).
3. Initialisez la variable `etat` à la valeur `INIT`

4. En repartant du début d'interface défini en fin de I., complétez la fonction mouseClicked de manière à ce que :
    a. la variable `etat` passe de `INIT` à `STARTED` quand on clique dans la zone en gris foncé (en-dessous du bandeau) ;
    b. la variable `etat` passe de `STARTED` à `INIT` quand on clique sur le bloc du smiley ; la fonction `init` devra alors être appelée ;
    c. la variable `etat` passe de `STARTED` à `OVER` quand on clique sur le bloc de la bombe ;
    d. la variable `etat` passe de `OVER` à `INIT` quand on clique sur le bloc du smiley ; la fonction `init` devra alors être appelée ;
5. Mettez à jour la fonction d'affichage de manière à afficher le smiley « happy face » quand on est dans les états `INIT` ou `STARTED` et le smiley « sad face » quand on est dans l'état `OVER`.

II.2. Alterner l'affichage d'un bloc

Dans les questions suivantes, le but sera de compléter la fonction `mouseClicked`. Pour savoir sur quel bouton de la souris le clic a eu lieu, il suffit de tester la valeur de la variable `mouseButton` et la comparer à `LEFT` ou `RIGHT` (regardez la documentation). Il est important de savoir dans quel état est un bloc (bloc « nu », bloc « découvert », bloc « suspect »). Pour cela, on utilisera une variable dont la valeur correspondra à l'état du bloc. Les différentes étapes s'enchaîneront selon le schéma suivant (le disque noir correspond à l'initialisation du bloc au démarrage du programme, les ellipses correspondent aux différents états du bloc, les flèches correspondent aux évènements faisant passer d'un état à l'autre) :



1. Modifiez la fonction `display` de manière à afficher un bloc au centre de l'interface ;
2. Déclarez une variable globale `etat_bloc`.
3. Déclarez des constantes `BLOC`, `EMPTY`, `FLAG` pour identifier les différents états possibles et attribuez-leur des valeurs entières croissantes (0, 1, 2).
4. Initialisez la variable `etat_bloc` à la valeur `INIT`
5. Complétez la fonction mouseClicked de manière à ce que :
    a. la variable `etat_bloc` passe de `BLOC` à `EMPTY` quand on clique sur le bloc avec le bouton de gauche alors qu'il est dans l'état `BLOC` ;
    b. la variable `etat_bloc` passe de `BLOC` à `FLAG` quand on clique sur le bloc avec le bouton de droite alors qu'il est dans l'état `BLOC` ;
    c. la variable `etat_bloc` passe de `FLAG` à `BLOC` quand on clique sur le bloc avec le bouton de droite alors qu'il est dans l'état `FLAG` ;
6. Modifiez la fonction `display` de manière à n'afficher le bloc que quand la variable `etat_bloc` a la valeur `BLOC` ou `FLAG` (on n'affichera rien sinon), et à afficher un drapeau dessus quand le bloc est dans l'état `FLAG`.
7. Mettez à jour la fonction `init` de manière à réinitialiser la variable `etat_bloc` à la valeur `BLOC`.

II.3. Un chronomètre

1. Déclarez une variable `start` pour enregistrer la date à laquelle on commence la résolution du démineur.
2. Utilisez la fonction `millis` pour initialiser la variable `start` au lancement de la résolution du jeu, c'est-à-dire au moment du premier clic dans la zone des blocs.
3. Déclarez une variable `time` pour calculer le nombre de secondes écoulées depuis le début de la résolution. Le principe est, à chaque appel de la fonction `draw`, de calculer le nombre de millisecondes écoulées depuis l'initialisation de la variable `start` puis de convertir ce nombre en secondes.

II.4. Afficher le chronomètre

Dans les questions suivantes, le but sera de compléter la fonction `drawTime`.

1. On veut afficher le chrono en utilisant la police de caractères `DSEG7Classic-Bold.ttf` fournie dans le dossier data. Pour cela :
   a. Déclarez une variable de type `PFont`
   b. Créez la fonte, en taille 32, grâce à `createFont`
   c. Au moment d'afficher un texte grâce à la commande `text`, choisissez la bonne police de caractères grâce à `textFont`
2. En haut à droite de la fenêtre, affichez un rectangle noir, de taille 80x40.
3. Affichez d'abord le texte « 888 » avec la couleur RGB (90, 10, 10), puis affichez la valeur de la variable time avec la couleur RGB (255, 0, 0).

4. Pour faire en sorte que le chrono s'affiche toujours sur 3 chiffres, il faut ajouter "0" à gauche du chrono s'il est strictement inférieur à 100, "00" à gauche s'il est strictement inférieur à 10, et bloquer le chrono à 999 (ou réinitialisez-le à 0 quand il atteint la valeur 1000) ;

II.5. Afficher le nombre de mines à trouver

Pour pouvoir jouer au démineur, il faut pouvoir cliquer sur un bloc avec le bouton de gauche pour le faire disparaître ou avec le bouton de droite pour afficher un drapeau sur le bloc si on pense qu'il cache une mine. Un compteur indique le nombre de mines à localiser. Chaque fois qu'un drapeau est positionné, ce compteur est décrémenté (indépendamment du fait que le drapeau soit bien localisé ou non). Il est à nouveau incrémenté si le drapeau est enlevé. Dans les questions suivantes, le but sera de compléter la fonction `drawScore`.

1. Créez une variable pour stocker le nombre de mines restant à localiser ;
2. En haut à gauche de la fenêtre, affichez un rectangle noir, de taille 80x40.
3. Affichez d'abord le texte « 888 » avec la couleur RGB (90, 10, 10), puis affichez le nombre de mines restantes avec la couleur RGB (255, 0, 0).
4. Comme pour le chronomètre, pour faire en sorte que le nombre de mines s'affiche toujours sur 3 chiffres, il faut ajouter "0" à gauche du nombre s'il est strictement inférieur à 100, "00" à gauche s'il est strictement inférieur à 10.

III TD7 à 10 = On affiche plein de blocs

III.1. Représenter les infos de tous les blocs

Pour gérer une grille de $n \times m$ blocs, on va devoir utiliser des tableaux, de manière à représenter l'état de chaque bloc.

1. Déclarez deux variables `lignes` et `colonnes` pour mémoriser le nombre de lignes et de colonnes du jeu et initialisez-les à 16 et 30 ;
2. Déclarez un tableau à deux dimensions d'entiers `paves` pour enregistrer l'état de chacun des blocs (BLOC, FLAG ou EMPTY). Dans la fonction `init`, initialisez tous les blocs dans l'état BLOC.
3. Déclarez un tableau à deux dimensions de booléens `bombes` pour enregistrer la position des bombes dans la grille. Dans la fonction `init`, initialisez toutes les valeurs à `false` puis créez aléatoirement une centaine de bombes dans la grille en changeant la valeur de la case correspondante à `true`.
4. Déclarez un tableau à deux dimensions d'entiers `nb_bombes` pour enregistrer le nombre de bombes présentes dans les huit cases autour de la case. Dans la fonction `init`, initialisez toutes les valeurs à 0 puis calculez les valeurs du tableau à l'aide du tableau `bombes`.

III.2. Adapter l'affichage

On veut vérifier que les tableaux `bombes` et `nb_bombes` ont été correctement initialisés. Pour ce faire, mettez à jour la fonction `display` pour afficher une grille de 30x16 blocs (= 16 lignes de 30 blocs). Chaque bloc a une largeur égale à `cote`. Un espace libre de hauteur `hauteur_bandeau` doit être laissé en haut de la fenêtre. Pour chaque bloc :

1. affichez une bombe si la case contient une bombe ;
2. affichez le nombre de bombes autour si la case ne contient pas de bombe et que le nombre de bombes autour est non nul ; Pour cela, utilisez la police de caractères `"mine-sweeper.ttf"` et affichez la valeur du tableau `nb_bombes` correspondante avec les couleurs suivantes :

nb	R	G	B
1	0	35	245
2	55	125	35
3	235	50	35
4	120	25	120
5	115	20	10
6	55	125	125
Autre	0	0	0

3. n'affichez rien si la case ne contient pas de bombe et que le nombre de bombes autour est nul.

Le résultat doit ressembler à ceci :

III.3. On peut enfin jouer !

On veut maintenant pouvoir jouer effectivement au démineur. Pour cela :

1. modifiez la fonction `display` de manière à :
    a. afficher un bloc si la case est dans l'état BLOC ou FLAG ;
    b. afficher un drapeau sur le bloc si la case est dans l'état FLAG ;
    c. afficher le nombre de bombes autour si la case est dans l'état EMPTY et que le nombre de bombes autour est non nul ;
    d. ne rien afficher si le bloc est dans l'état EMPTY et que le nombre de bombes autour est nul.
2. modifiez la fonction `mouseClicked` de manière à :
    a. vérifier si le clic a eu lieu dans la zone de blocs ;
    b. calculer le n° de ligne de la case cliquée : `(mouseY - hauteur_bandeau) / cote`
    c. calculer le n° de colonne de la case cliquée : `(mouseX / cote)`
    d. si on fait un clic sur une case avec le bouton de gauche (`mouseButton` égal à la valeur 37) :
        i. si la case contient une bombe, faire passer le jeu de STARTED à OVER, et arrêter le chrono
        ii. sinon faire passer la case de BLOC à EMPTY
    e. si on fait un clic sur une case avec le bouton de droite (`mouseButton` égal à la valeur 39) :
        i. si la case est dans l'état BLOC, la faire passer dans l'état FLAG et décrémenter la variable indiquant le nombre de drapeaux à trouver ;
        ii. si la case est dans l'état FLAG, la faire passer dans l'état BLOC et incrémenter la variable indiquant le nombre de drapeaux à trouver ;
        iii. si la case est dans l'état EMPTY, ne rien faire

III.4. En bonus…

1. Complétez la fonction `decouvre` de manière à ce que, si on clique sur une case vide et sans aucune bombe autour, on découvre les 8 cases autour. Si dans les cases autour, certaines sont elles-mêmes vides et sans bombe autour, il faut également découvrir ces cases ainsi que les voisines vides de ces cases, et les voisines des voisines, et les voisines des voisines des voisines…
   hint : une fonction récursive peut être adaptée pour le faire simplement.

2. Complétez la fonction `decouvre2` de manière à ce que, si on clique sur une case vide, avec un nombre `n` de bombes autour et un nombre `n` de drapeaux bien placés autour, les cases non encore découvertes et ne portant pas de drapeau soient découvertes. Si on s'est trompé dans le placement de l'un des drapeaux autour de la case, le programme doit passer dans l'état OVER.