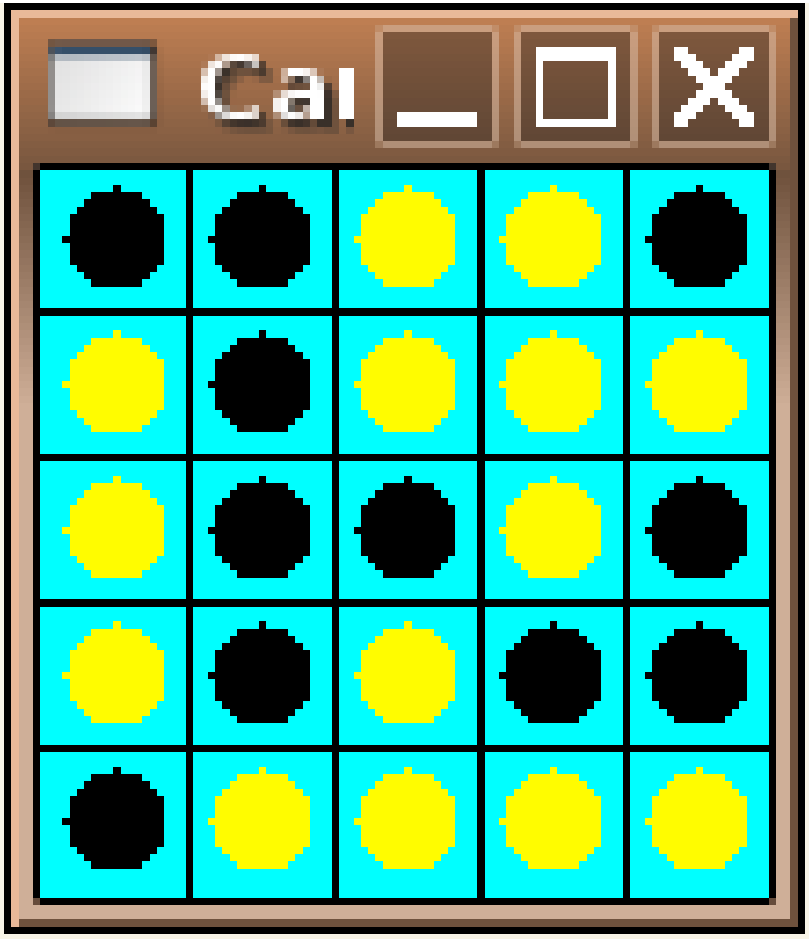
TP3 —Le jeu All Out

Le but de cette séance de TP est de programmer le jeu **All Out**. Ce jeu se compose d'une grille de n x n cases où sont déposées des ampoules. Au début du jeu, les ampoules sont allumées ou éteintes de manière aléatoire.

La copie d'écran suivante montre un exemple de configuration initiale du jeu (pour n = 5) où les ampoules allumées sont représentées par des cercles de couleur claire et les ampoules éteintes par des cercles noirs.



Un joueur de **All Out** ne peut faire qu'une seule action: inverser l'état d'une ampoule. Cependant, cette action entraîne également l'inversion des ampoules adjacentes (i.e celles situées juste une case en haut, en bas, à gauche ou à droite) de l'ampoule modifiée.

Le but du jeu est d'éteindre toutes les ampoules de la grille.

Unités de compilation

Le but du TP est de produire un programme `allout`. Pour cela, nous allons découper la

construction de ce programme en deux unités de compilation : une interface graphique (unité `Ig`) et un programme principal (unité `Allout`).

L'interface graphique aura l'interface suivante (à enregistrer dans un fichier `ig.mli`):

```
exception Solution

val lancer_interface : int -> unit
val fermer_interface : unit -> unit
val dessiner_grille : bool array array -> unit
val attendre_coup : unit -> int * int
```

La fonction `lancer_interface` permet d'ouvrir une fenêtre graphique pour une grille de n x n lampes (n est passé en argument). La fonction `fermer_interface` permet de fermer cette fenêtre.

La fonction `dessiner_grille` dessine une grille passée en argument sous forme d'une matrice de booléens.

La fonction `attendre_coup` attend que l'utilisateur ait soit cliqué sur une lampe, soit tapé sur une touche du clavier. S'il a cliqué sur une lampe, la fonction renvoie un couple d'entiers correpondant aux coordonnées de la case dans la matrice. Sinon, si l'utilisateur a tapé la touche `q`, le jeu se termine. Si l'utilisateur a tapé la touche `s`, la fonction lève l'exception `Solution` pour indiquer que l'utilisateur souhaite connaître la solution du jeu.

Programmation. Écrire les fonctions de l'unité compilation `Ig`. On utilisera un appel `wait_next_event [Button_down;Key_pressed]` pour récupérer les évènements clic de souris et touche pressée (voir cette page du manuel OCaml)

Makefile

Écrire un fichier `Makefile` pour automatiser la compilation de votre programme.

Programmation

L'unité de compilation `Allout` doit permettre non seulement de jouer au jeu de manière interactive, mais également de trouver la solution du jeu (si elle existe) quand l'utilisateur tape sur la touche `s`.

Voici quelques indications pour vous aider à programmer ce jeu.

Indication 0 La taille de la grille sera passée en argument à l'exécution du programme.

Indication 1 Écrire la fonction `nouvelle_grille: int-> bool array array` à l'aide de `Random.int` telle que `nouvelle_grille p` initialise une matrice de taille n x n remplie de booléens ayant p chances sur 100 d'être vrais.

Indication 2 Écrire la fonction `selectionne_cases: int -> int -> (int * int) list` en utilisant la fonction `List.filter` telle que `selectionne_cases i j` retourne les coordonnées de toutes les cases de la matrice qui devront être inversées si l'on inverse l'ampoule située aux coordonnées `(i, j)`. On fera attention à ne pas sélectionner de cases en dehors de la grille.

Indication 3 Écrire la fonction `grille_gagnante : bool array array -> bool` qui détermine si toutes les lampes de la grille passée en paramètre sont éteintes.

Indication 4 En remarquant que la solution (si elle existe) permettant d'éteindre toutes les lampes d'une grille ne contient qu'au plus une inversion par lampe, que l'ordre de ces inversions n'importe pas, et que pour "annuler" l'effet d'un coup il suffit de rejouer ce même coup, écrire la fonction `solution : bool array array -> (int * int) list option` telle que `solution g` retourne `None`

s'il n'y a aucune solution à la grille `g` et `Some l` si une solution existe, où `l` est une liste de paires d'entiers correspondant aux inversions à réaliser pour arriver à éteindre toutes les lampes de `g`.