PROJET ISOLA

L1

Table des matières

Présentation & objectif

Sur un plateau prédéfini, chaque joueur possède un pion qu'il place sur une case blanche qu'il choisit à condition que celle-ci ne soit pas déjà occupée par le pion du joueur adverse.

Chaque joueur joue à tour de rôle. À chaque tour, un joueur déplace son pion vers une case libre qui lui est adjacente puis il interdit une des cases non occupée du plateau et ce, pour le reste de la partie ; une case interdite ne peut plus être utilisée par le joueur ou par son adversaire pendant la partie en cours.

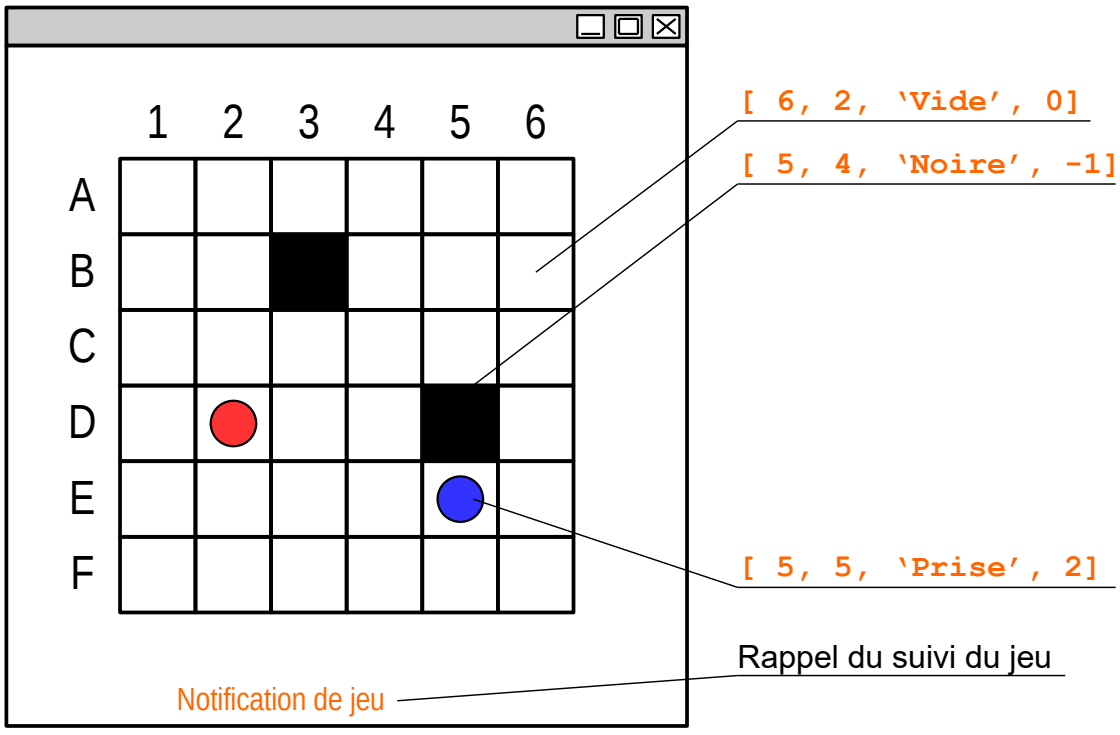
Le but du jeu est de bloquer son adversaire : Le premier joueur qui ne peut plus déplacer son pion perd la partie.

Notre objectif est de recréer ce jeu en programmant en langage Python 3 avec l'aide des fonctions contenues dans le module « fltk », « random » et « doctest » ainsi que les fonctions natives de ce langage.

Démarche logique

En début de partie, qu'il s'agisse du mode « un » ou « deux » joueurs, toutes les coordonnées des cases du plateau sont préparées sont la forme d'une liste de cases, lesquelles sont elles-même définies par une liste contenant les éléments suivants :

- PosX : Position X de la case (numéro de colonne et pas en pixels)
- PosY : Position Y de la case (numéro de ligne et pas en pixels)
- Etat : « Vide », « Noire » ou « Prise »
- Joueur : 0, -1 ou (1 ou 2 pour les pions respectifs des joueurs « Rouge » ou « Bleu »)



Pendant la partie, un pion en cours d'utilisation est enregistré dans un dictionnaire nommé PION_ACTUEL dont les paramètres sont les suivants :

- La clef 'Couleur' contenant les valeurs 'rouge' ou 'bleu'.
- La clef 'Joueur' contenant les valeurs 1 ou 2.

Ce dictionnaire permet de savoir, à tout moment de la partie, le joueur en cours de jeu. Une fonction spéciale permet l'alternance des joueurs à chaque fin de tour complet {déplacement de pion + interdiction de case}.

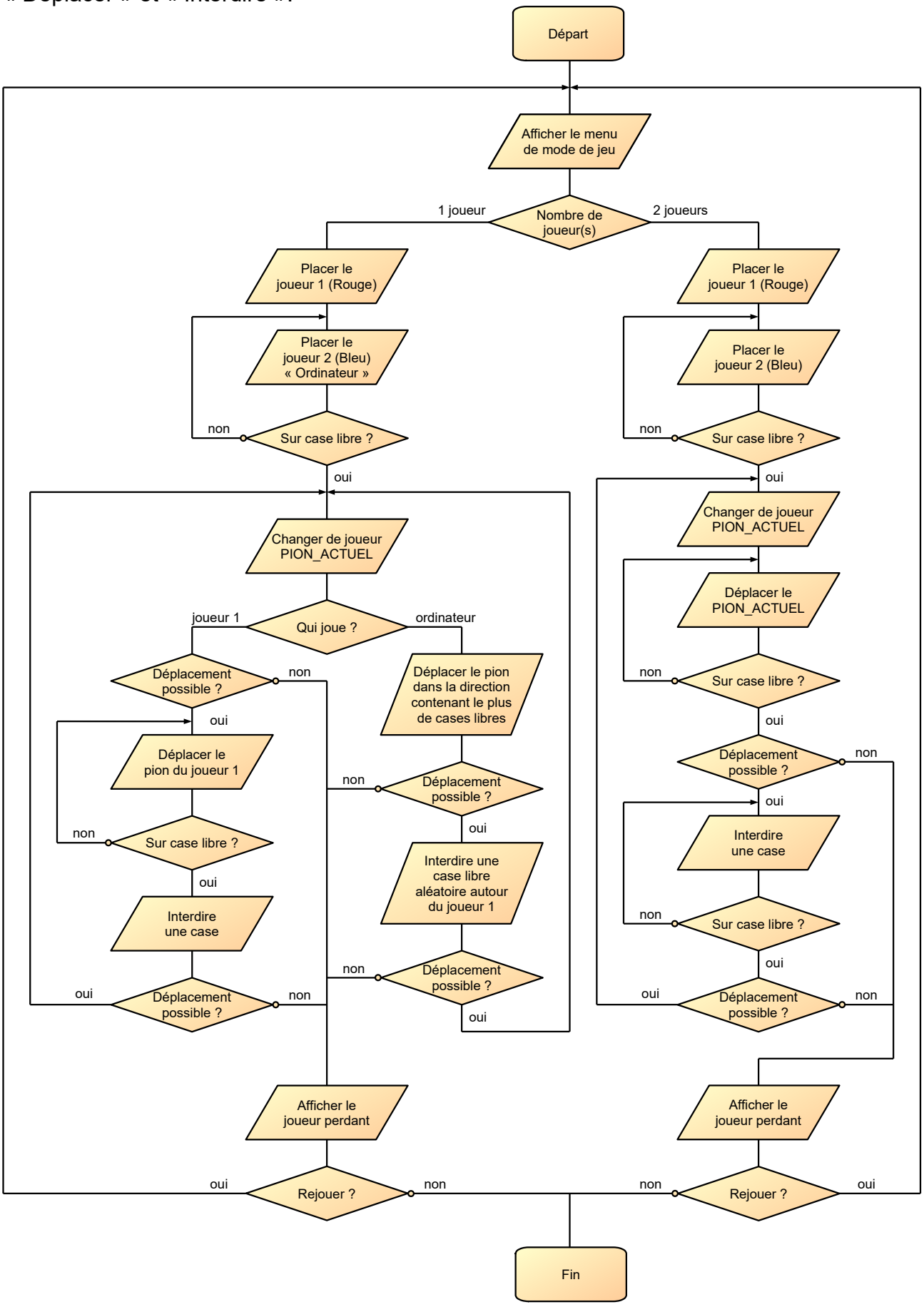
Une amélioration importante a été apportée : Au lieu d'entrer les coordonnées des déplacement et d'interdiction au clavier en tapant C-3, E-5…, une détection évènementielle a été mise en place et les différentes coordonnées sont finalement saisies par des « clics souris » sur le plateau graphique, que ce soit pour le déplacement d'un pion ou l'interdiction d'une case. Les coordonnées des « clics » captées en pixels sont systématiquement converties en positionnement de case sur le plateau. En revanche, une case occupée par un pion ou une interdiction n'est pas prise en compte par le « clic » et le jeu reste en pause tant qu'une case blanche, à fortiori « libre », n'est pas cliquée.

Par ces évènements, les joueurs n'intéragissent pas directement graphiquement mais modifient les listes des cases du plateau. Une fonction spéciale met ensuite à jour l'affichage graphique en prenant en charge les changements qui ont été apportés préalablement par les joueurs.

Enfin, il faut savoir que dès qu'une action est menée et qu'un rafraichissement a lieu, une vérification de fin de partie est réalisée. Si l'un des deux joueurs voit son pion complètement immobilisé, un message de fin de partie s'affiche pour annoncer le joueur perdant et si une autre partie doit être rejouée : Si tel est le cas, le jeu repart au choix du nombre de joueur, dans le cas contraire la fenêtre du jeu se ferme.

Démarche de programmation

Afin de faciliter la lecture de l'algorigramme de programmation, celui-ci ne présente pas la détection des évènements bien que chacun d'eux servent de déclencheur pour les actions « Déplacer » et « Interdire ».



Fonctions littérales

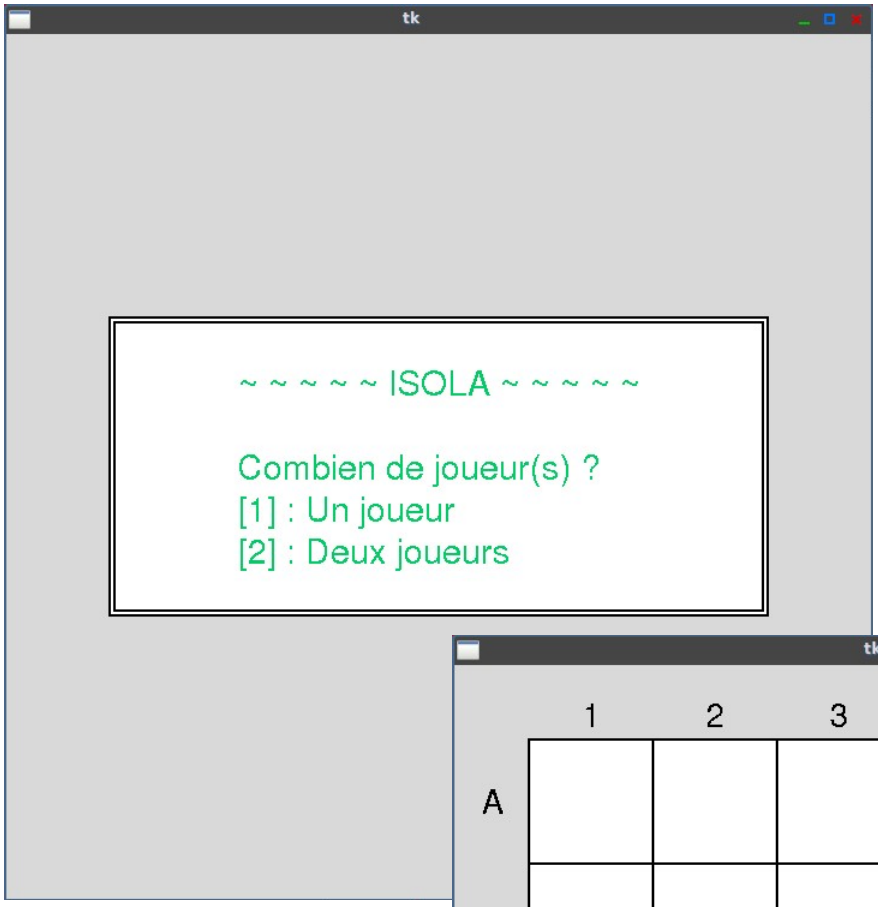
Nom	Rôle	Entrée	Sortie
pixel_vers_case	Converti des coordonnées données en pixels en des coordonnées de position de case	Tuple	Liste
case_vers_pixel	Converti des coordonnées de position de case en coordonnées du centre de la case en pixel	Tuple	Liste
case_possible	Détermine les cases possible pour se déplacer	Entier Liste de listes	Liste de listes
case_possible_hasard	Fourni les coordonnées d'une case au hasard autour du joueur	Entier Liste de listes	Liste de listes
position_joueur	Détermine la position d'un joueur	Entier Liste de liste	Liste
multi_list	Multiplie tous les élément de la liste par un nombre	Liste Entier	Liste
add_list	Additionne deux liste entre elles élément à élément	Liste Liste	Liste
pos_max_list	Détecte la position du maximum d'une liste	Liste	Entier
choix_vide	Détermine la case la plus judicieuse pour se déplacer	Entier Liste de listes	Liste
init_pateau	Crée le plateau		Liste de listes
change_joueur	Permet d'alterner les joueurs		

Fonctions graphiques et évènementielles

Nom	Rôle	Entrée	Sortie
affiche_plateau	Affiche graphiquement le plateau	Liste de liste	
affiche_texte	Affiche un message dans le canvas	Chaîne de caractères	
cree_pion	Modifie le plateau en fonction des coordonnées du curseur durant le clic gauche	Liste de liste	Liste de liste
case_en_moins	Noirci les cases en fonctions des coordonnées du curseur pendant le clic gauche	Liste de liste	Liste de liste
case_en_moins_ordi	Noirci une case aléatoirement autour du joueur 1	Liste de liste	Liste de liste
deplacement	Déplace un joueur sur la case où se trouve le curseur pendant le clic gauche	Liste de liste	Liste de liste
deplacement_ordi	Déplace automatiquement le joueur 2 sur une case possible	Liste de liste	Liste de liste
efface_position	Supprime le pion de son ancienne case et la rend vide	Entier Liste de liste	Liste de liste
mort	Affiche un message de fin de jeu avant la fin réelle		False
menu_depart	Affiche le menu de mode de jeu		
nb_joueur	Attend la touche correspondant au nombre de joueur		Entier
jeu	Lance le jeu	Bool	Bool
jeu1	Lance le jeu a un seul joueur	Bool Liste de liste	Bool
jeu2	Lance le joueur a deux joueurs	Bool Liste de liste	Bool

Manuel d'utilisation

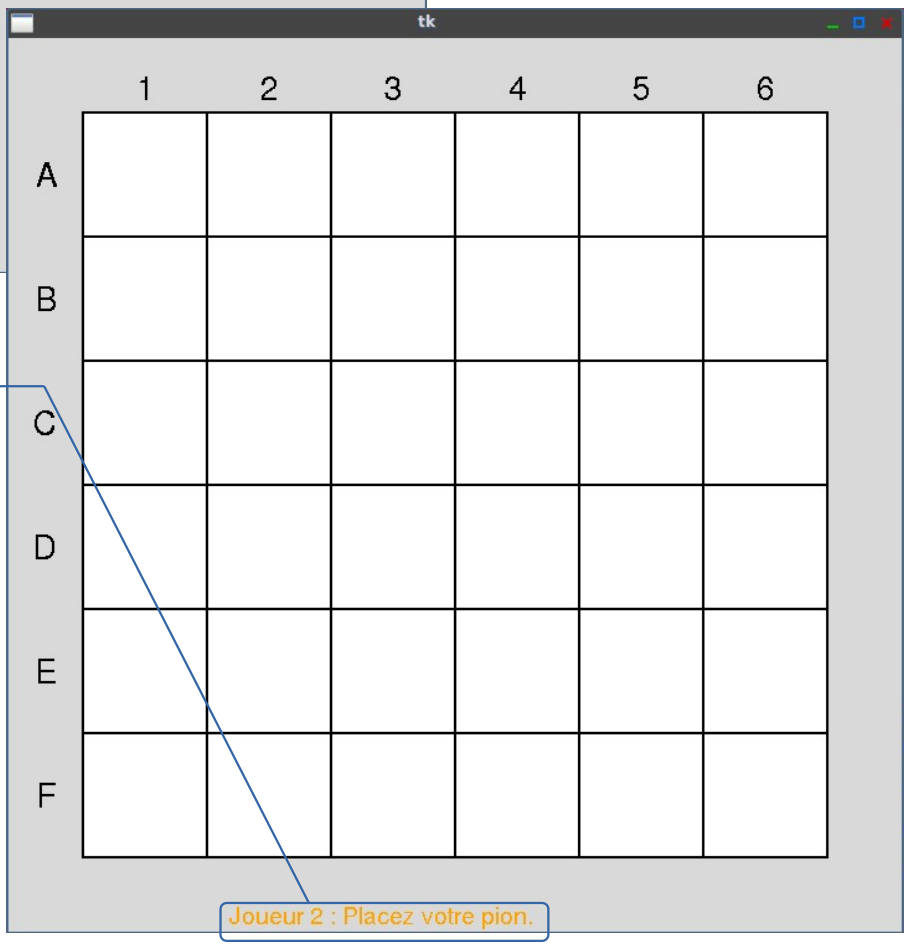
Après avoir lancer l'exécution du programme, un menu s'affiche pour connaître le nombre de joueur(s).



Les valeurs placées entre crochets [ ] correspondent à la touche à appuyer pour effectuer votre choix.

Exemple : [1] correspond à la touche « 1 » du pavé numérique ou des nombres supérieurs dans le clavier alpha-numérique.

Appuyez sur la **touche « 1 »** pour commencer une partie seul(e) ou la **touche « 2 »** pour démarrer une partie à deux.

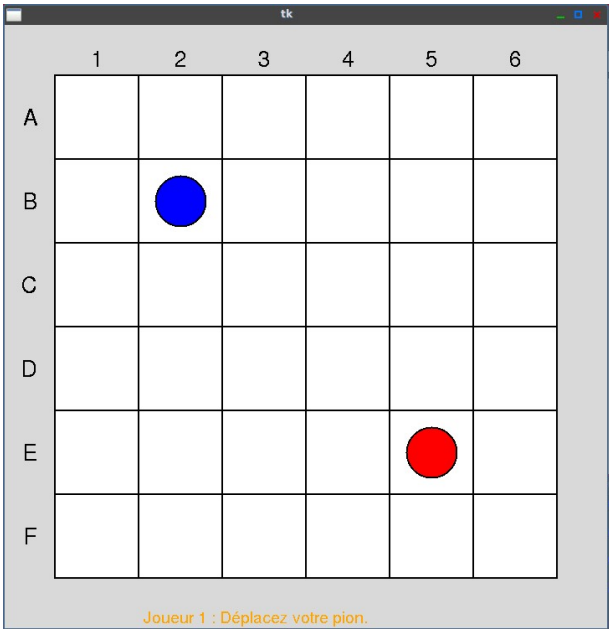


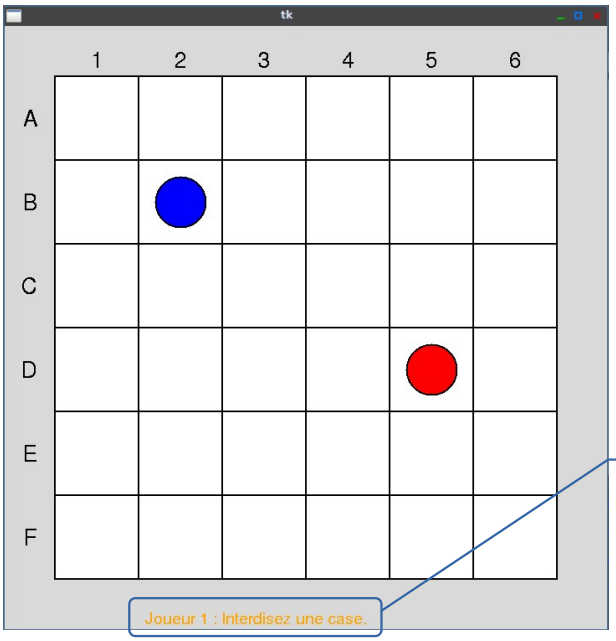
Comme l'indique la **notification** en bas de la fenêtre, il est demandé au **joueur 2** de placer son pion.

Il sera ensuite demandé au **joueur 1** d'en faire de même.

Lorsque les deux pions sont placés, la partie débute.

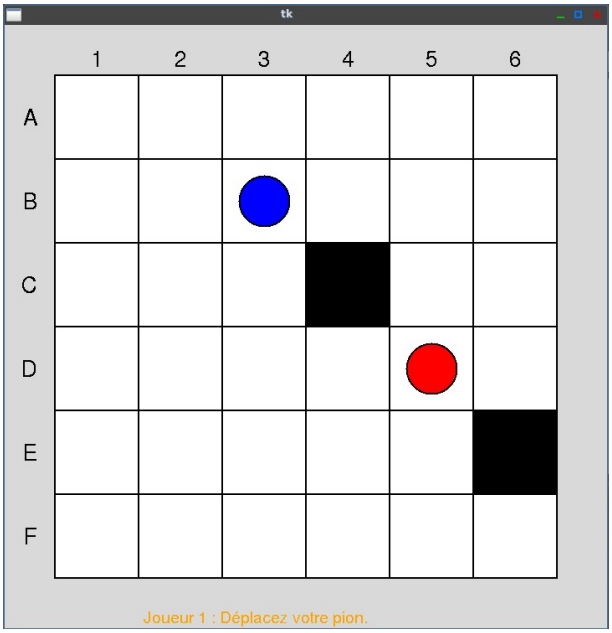
Le **joueur 1 (Rouge)** est le premier à devoir déplacer son pion.

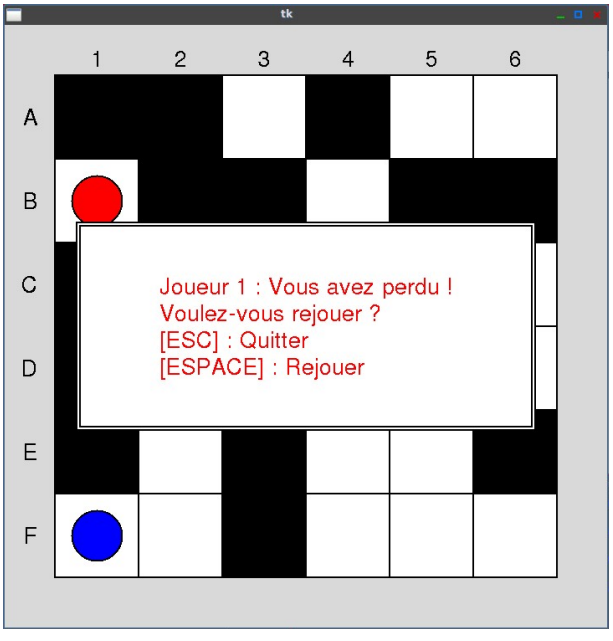




Le jeu demande alors au **joueur 1** d'**interdire** une case du plateau en **cliquant dessus pour la noircir**.

La partie se déroule alors en alternant tout à tour les joueurs ainsi que les déplacements de pion et les interdictions de cases jusqu'à ce qu'un des joueurs ne puisse plus déplacer son pion, tel qu'est présentée la situation ci-dessous.





Lorsqu'un joueur perd, le message ci-contre s'affiche et le jeu propose aux usagers de rejouer en appuyant sur la « barre d'espace » ou de quitter le jeu en appuyant sur la touche « Escape », quelquefois nommée « Echap ».

Améliorations apportées

Dans notre programme nous avons ajouté différentes améliorations :

- Les joueurs peuvent désormais choisir la taille de grilles, le nombre de colonnes est compris entre 5 et 12 cases et le nombre de lignes compris entre 5 et 9 cases.
- Nous avons décidé d'utiliser l'événement « clic gauche » pour le choix des positions, des cases à supprimer et des déplacements fait par les joueurs.
- Pour le raisonnement de l'ordinateur nous avons décidé qu'il se déplacera dans la direction vers lequel il y a le plus de cases vides consécutives. Quant au noircissement des cases, il prend au hasard une case possible autour du joueur 1.

Difficultés rencontrées

Nous avons rencontré des difficultés à trouver une stratégie de programmation la plus simple pour le déplacement voulu quant au pion de l'ordinateur.

De plus pour la fonction case_possible la difficulté réside à trouver un algorithme permettant de vérifier les cases autour du pion car il n'a pas de coordonnées fixe.

Le plus délicat dans la fonction pos_max_list était de trouver l'indice du maximum d'une liste.