

# OTHELLO

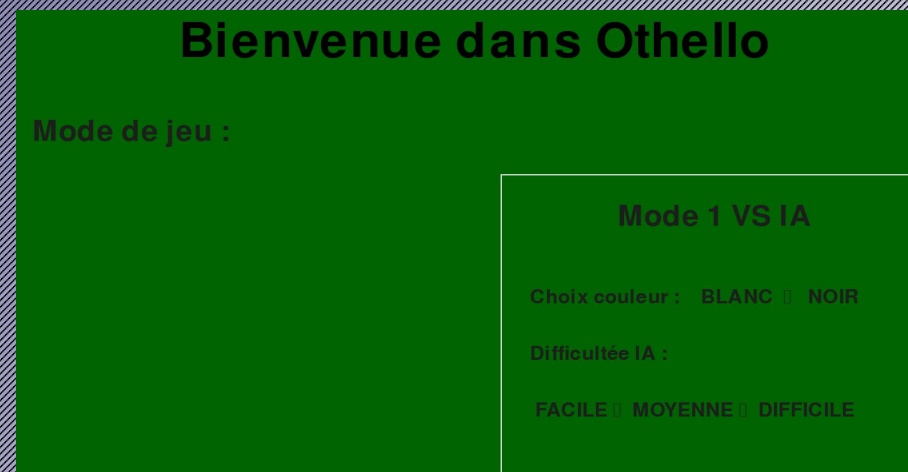
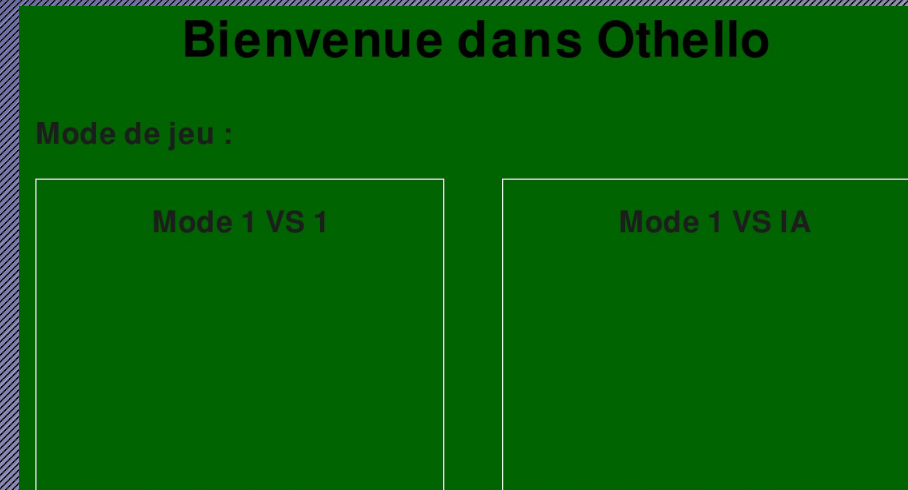
Modélisation informatique  
d'un jeu de plateau



# VUE D'ENSEMBLE DE L'ORGANISATION DU JEU

## Organisation et conventions :

- Deux modes de jeu possibles
- Couleur des joueurs codée en 1 / 2 (respectivement blanc/noir)
- Le joueur blanc commence toujours
- Mode 1vsIA : choix de la difficulté
  - facile : coup au hasard
  - moyen : coup retournant un maximum de pions
  - difficile : prévisions avec 3 coups d'avance



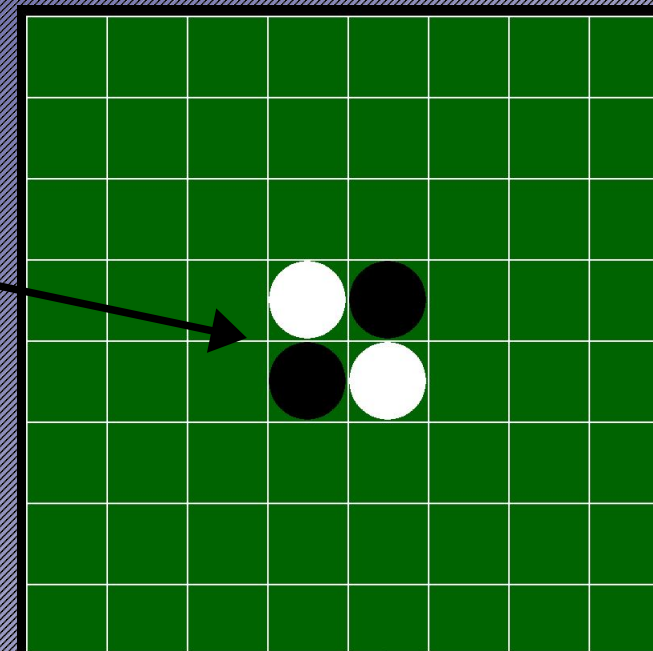


# CREATION DU PLATEAU DE JEU

```
def grille_de_jeu(n):  
    """Entrée un entier n pair et qui renvoie une matrice carré de taille  
    correspondant au plateau de jeu initial"""  
  
    noir=2  
    blanc=1  
    if n%2==1:  
        return "Plateau de jeu uniquement de côté paire"  
  
    else :  
        jeu=np.zeros((n,n))  
        m=n//2-1 # Calcul du milieu de la matrice "jeu"  
  
        jeu[m,m],jeu[m+1,m+1],jeu[m,m+1],jeu[m+1,m]=blanc,blanc,noir,noir  
  
    return jeu
```

```
def matrice_miroir(n):  
    """ Sorties : Création d'un matrice de taille n*n avec des sous listes  
    de longueur 8 pour chaque case. """  
  
    m_initial=[]  
  
    L_vide=[0 for x in range(8)]  
  
    for hz in range (n):  
        for vt in range (n):  
            m_initial.append(L_vide)  
    miroir_joueur=np.array(m_initial)  
    miroir_i=m_initial  
    return miroir_joueur, miroir_i
```

Création de listes  
pour le stockage  
des informations  
concernant les  
cases jouables



## Règles du jeu :

Le but du jeu est d'obtenir le plus de pion de sa couleur à la fin de la partie. Lors de son tour, le joueur pose un pion de manière à encadrer un nombre de pions adverses. Ces derniers se voient alors transformés en pions de la couleur du joueur. Un pion posé doit obligatoirement transformer au minimum 1 pion adverse. Les 8 directions sont jouables. Le joueur blanc est toujours le 1er à commencer. Bon jeu !

## Instructions :

- > CLIQUER sur une case du plateau
- > Si une IA doit commencer, cliquer ICI
- > Appuyer sur ENTRER pour provoquer l'animation du plateau
- > Si la case était jouable, un pion est apparu puis les autres
- > Attention : Une erreur est permise par tour.
- > Au delà, le joueur concerné se voit passer son tour.

Interface graphique sous Pygame



# FONCTION CASE\_LIBRE\_JOUABLE : permet le bon déroulement d'une partie

```
def case_libre_jouable(n,tour_du_joueur,Ia, plateau_jeu):  
  
    Systeme=[1,2]  
    Systeme.pop(tour_du_joueur-1) # Systeme = couleur de l'adversaire  
  
    miroir_joueur, miroir_i = matrice_miroir(n)  
  
    for y in range(n): # Balayage des lignes  
        for x in range(n): # Balayage des colonnes  
            if plateau_jeu[y,x]==Systeme[0]:  
  
                for hz in range (x-1,x+2):  
                    for vt in range (y-1,y+2):  
  
                        if -1<hz<n and -1<vt<n :  
  
                            cmpt_vecteur=1  
                            N=n*vt+hz  
                            sousliste_i=miroir_joueur[N]  
  
                            if plateau_jeu[vt,hz]==0.0:  
                                vecteur=[y-vt, x-hz]  
                                pion_y,pion_x=y+vecteur[0],x+vecteur[1]
```

Matrice destinée à stocker les informations concernant chaque case jouable  
Matrice de comparaison vide

Première recherche : cases vides autour des pions adverses



# FONCTION CASE\_LIBRE\_JOUABLE : permet le bon déroulement d'une partie

```
interrupteur=False # Signature par défaut
```

```
while -1<pion_y<n and -1<pion_x<n and \
plateau_jeu[pion_y, pion_x]==Systeme[0]:
```

```
    interrupteur=True # Signature de la boucle
```

```
    pion_x+=vecteur[1]
```

```
    pion_y+=vecteur[0]
```

```
    cmpt_vecteur+=1
```

```
if not interrupteur:
```

```
    if -1<pion_y<n and -1<pion_x<n and \
    plateau_jeu[pion_y,pion_x]==tour_du_joueur:
```

```
        sousliste_miroir= \
```

```
        codage_miroir([vecteur[0],vecteur[1]],cmpt_vecteur,sousliste_i)
```

```
        miroir_joueur[N]=sousliste_miroir
```

```
else: #Cas général (passage par la boucle while)
```

```
    if -1<pion_y<n and -1<pion_x<n and \
    plateau_jeu[pion_y,pion_x]==tour_du_joueur:
```

```
        sousliste_miroir=\
```

```
        codage_miroir([vecteur[0],vecteur[1]],cmpt_vecteur, sousliste_i)
```

```
        miroir_joueur[N]=sousliste_miroir
```

```
Liste_case_jouable=lecture_miroir(n,miroir_joueur,miroir_i)
```

```
if Ia: # mode de jeu avec Intelligence Artificielle
```

```
    case_joueur=reecriture_case_jouable(n,Liste_case_jouable)
```

```
    return case_joueur, miroir_joueur
```

Cas où plusieurs pions  
adverses se succèdent

Recherche approfondie : 1  
pion de la couleur du joueur à  
la fin de la succession de pions  
adverses = case jouable

Liste des cases jouables



# FONCTION TOUR DE JEU : Simule les actions du joueur pendant 1 tour

```
def tour_jeu_CLIC(n,tour_du_joueur,arret_jeu,case,avec_Ia,compteur_erreur,validation):  
  
    plateau_jeu=jeu  
    Liste_pion_modifiee=[]  
  
    Systeme=[1,2]  
    Systeme.pop(tour_du_joueur-1)  
  
    possibilite,miroir_joueur=\  
    case_libre_jouable(n,tour_du_joueur,False,plateau_jeu)  
    if possibilite==[]:
```

Récupération des coordonnées  
des cases qui sont jouables

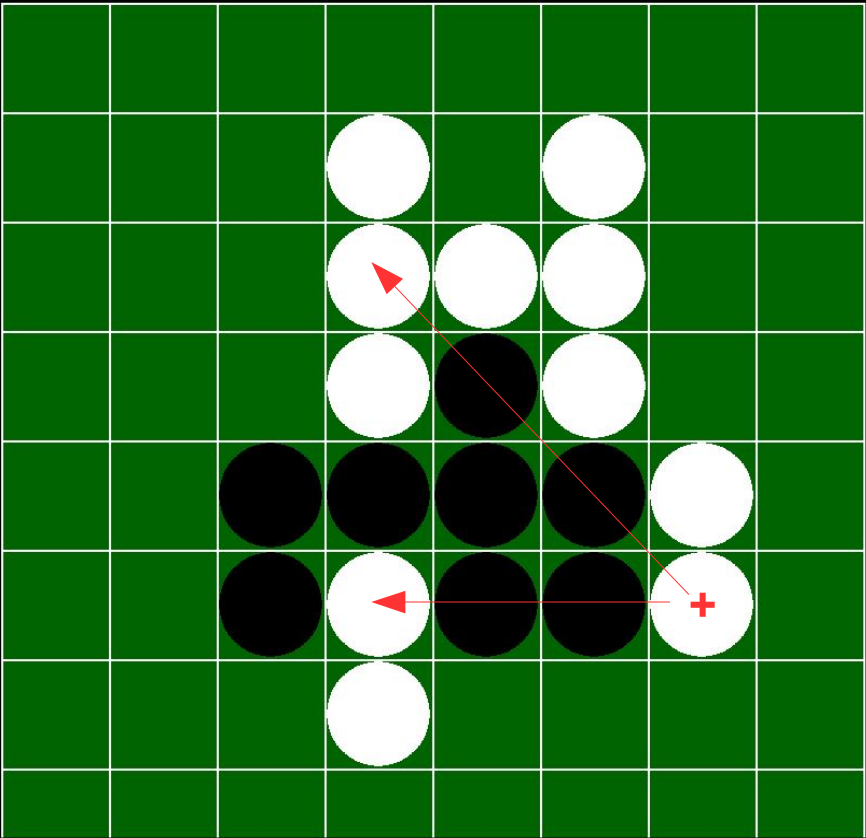
```
else:  
    nouv_y,nouv_x=reecriture_case_jouable_CLIC(n,case)  
    # Coordonnées de la case cliquée sous forme [y,x]  
    compteur_erreur,validation=\  
    verification_CLIC(possibilite,case,compteur_erreur,validation)  
  
    if compteur_erreur<2 and validation :  
        jeu[nouv_y,nouv_x]=tour_du_joueur  
        Liste_pion_modifiee=\  
        retourne_pions(n,tour_du_joueur, True, nouv_y, nouv_x, miroir_joueur,plateau_jeu)  
        arret_jeu=0  
    elif compteur_erreur>=2: # Le joueur passe son tour : réinitialisation  
        Fenetre_passe_tour = pygame.Surface((550,175))  
        Fenetre_passe_tour.fill(gris_taupe)  
        font = pygame.font.Font(None, 35)  
  
        if tour_du_joueur==1:  
            texte = font.render("Blanc passe son tour", 1, blanc)  
        else:  
            texte = font.render("Noir passe son tour", 1, blanc)  
        Fenetre_passe_tour.blit(texte,(25,0))  
  
        if not avec_Ia:  
            Fenetre.blit(Fenetre_passe_tour, (1000,420))  
            compteur_erreur=0  
        else:  
            texte_IA = font.render("Tour de l'IA ", 1, blanc)  
            Fenetre_passe_tour.blit(texte_IA,(295,0))  
            pygame.draw.rect(Fenetre_passe_tour,(0,100,0),[294,0,140,30],2)  
            Fenetre.blit(Fenetre_passe_tour, (1000,495))  
  
    tour_du_joueur=Systeme[0]  
  
return tour_du_joueur, arret_jeu, compteur_erreur, validation, Liste_pion_modifiee
```

Condition à vérifier pour pouvoir  
jouer : la case choisie doit être dans  
la liste des cases jouables

Seconde chance en cas d'erreur du joueur



# Exemple en pratique de la fonction TOUR DE JEU



**Instructions :**

- > CLIQUER sur une case du plateau
- > Si celle-ci est jouable, un pion de votre couleur apparaît
- > Appuyer sur ENTRER pour provoquer l'animation du plateau
- > Attention : Une erreur est permise par tour.
- > Au delà, le joueur concerné se voit passer son tour.

**Mode 1 vs 1, Tour de jeu :**

**Joueur blanc**



# FONCTIONS PERMETTANT DE JOUER CONTRE UNE I.A. :

- La création d'une Intelligence Artificielle permet à un utilisateur de jouer contre un adversaire virtuel.

```
def tour_IA_CLIC(n,difficulte,tour_Ia,arret_jeu,joueur):  
    if difficulte==[0]:  
        tour du joueur,arret_jeu,Liste_pion_modifiee,joueur=\  
            IA_facile_CLIC(n,tour_Ia,arret_jeu,joueur)  
    if difficulte==[1]:  
        tour du joueur,arret_jeu,Liste_pion_modifiee,joueur=\  
            IA_moyen_CLIC(n,tour_Ia,arret_jeu,joueur)  
    if difficulte==[2]:  
        tour du joueur,arret_jeu,Liste_pion_modifiee,joueur=\  
            IA_difficile_CLIC(n,tour_Ia,arret_jeu,joueur)  
  
    return tour du joueur,arret_jeu,Liste_pion_modifiee, joueur
```

## Mode 1 VS IA

Choix couleur : BLANC □ NOIR

Difficulté IA :

FACILE □ MOYENNE □ DIFFICILE



# FONCTION IA DIFFICILE :

## Présentation de la fonction et de ses « outils » associés

```
def IA_difficile_CLIC(n,tour_du_joueur,arret_jeu,joueur):  
    plateau_0=jeu  
    Liste_pion_modifiee=[]  
  
    Systeme=[1,2]  
    tdj=Systeme.pop(tour_du_joueur-1)  
    # Stockage de la couleur du joueur avant d'effectuer la fonction.  
  
    possibilite_0,miroir_joueur=\  
    case_libre_jouable(n,tour_du_joueur,True,plateau_0)  
    if possibilite_0==[]:
```

Réutilisation des fonctions de base

```
    else:  
        tour_du_joueur,arret_jeu, Liste_plateau_1=\  
        simulation_plateau(n,tour_du_joueur,arret_jeu,plateau_0)  
        tour_du_joueur,arret_jeu, Liste_plateau_2=\  
        simulation_du_max(n,tour_du_joueur,arret_jeu,Liste_plateau_1)  
        tour_du_joueur,arret_jeu, Liste_plateau_3=\  
        simulation_du_max(n,tour_du_joueur,arret_jeu,Liste_plateau_1)  
  
        Liste_Nb1=[]  
        Liste_Nb1_max=[]  
        reference=0  
  
        for k in range(len(Liste_plateau_3)):  
            Nb_1,Nb_2,Vide=0,0,0  
  
            for y in range(n):  
                for x in range(n):  
                    if Liste_plateau_3[k][y,x]==0:  
                        Vide+=1  
                    else:  
                        if Liste_plateau_3[k][y,x]==1:  
                            Nb_1+=1  
                        else:  
                            Nb_2+=1  
            Liste_Nb1.append(Nb_1)
```

Simulation de plateaux à t+1  
Simulation de plateaux à t+2  
Simulation de plateaux à t+3

Comptage des pions noirs, blancs et des cases vides sur les plateaux fictifs à t+3



# FONCTION IA DIFFICILE :

## Présentation de la fonction et de ses « outils » associés

```
for l in range(len(Liste_Nb1)):
    X=Liste_Nb1[l]
    if X>reference:
        reference=X
        Liste_Nb1_max=[l]
    elif X==reference:
        Liste_Nb1_max.append(l)

# Retour au jeu réel

hasard=random.choice(Liste_Nb1_max)
choix_Ia=possibilite_0[hasard]
nouv_y, nouv_x=choix_Ia[0],choix_Ia[1]

-----
plateau_0[nouv_y, nouv_x]=tdj
Liste_pion_modifiee=\
retourne_pions(n,tdj,True,nouv_y, nouv_x, miroir_joueur,plateau_0)

arret_jeu=0
tour_du_joueur=Systeme[0]

return tour_du_joueur,arret_jeu
```

Recherche du plateau à  $t+3$   
avec un maximum de pions  
de la couleur de l'IA

Choix du pion menant au  
meilleur coup pour l'IA à  $t+3$   
Choix au hasard s'il en existe  
plusieurs



# FONCTION SIMULATION\_PLATEAU :

Un plateau «fictif» est créé par possibilité de case jouable (t+1)

```
def simulation_plateau(n,tour_du_joueur,arret_jeu,plateau_0):  
# Création de plateaux fictifs au tour_+1  
  
Systeme=[1,2]  
Systeme.pop(tour_du_joueur-1)  
  
possibilite,miroir_joueur=case_libre_jouable(n,tour_du_joueur,True,plateau_0)  
  
if possibilite==[]:  
    arret_jeu+=1  
    tour_du_joueur=Systeme[0]  
else:  
    Liste_plateau_jeu=[] # Possibilités de plateaux jouables à t=0  
    for k in range(0,len(possibilite)):  
        plateau_jeu=copy.deepcopy(plateau_0)  
        # Copie profonde pour pouvoir manipuler plusieurs copies du plateau de jeu à t=0  
  
        choix_Ia=possibilite[k]  
        nouv_y, nouv_x=choix_Ia[0],choix_Ia[1]  
  
        plateau_jeu[nouv_y, nouv_x]=tour_du_joueur  
        #SIMULATION: Ajout d'un pion sur le plateau fictif avec les coordonnées choisies par l'IA
```



Toute les possibilités de jeu dont  
prises en compte

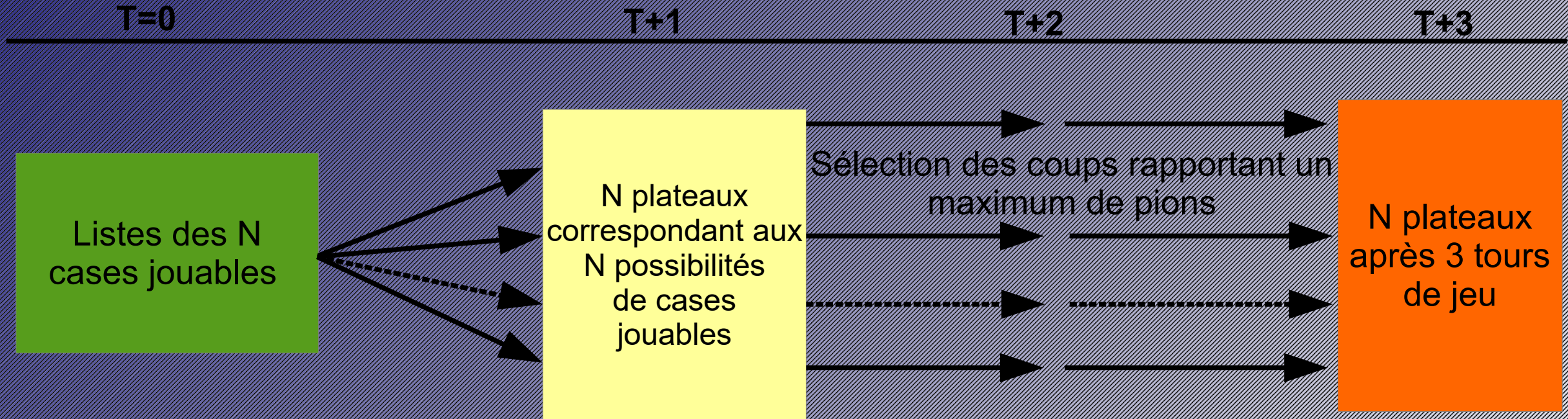
```
# BOUCLE pour retourner des pions  
  
sousliste=miroir_joueur[nouv_y*n + nouv_x]  
  
for l in range(8): # 8 vecteurs possibles autour d'une case  
    y=nouv_y  
    x=nouv_x  
  
    if sousliste[l] !=0 :# Case est jouable dans cette direction  
  
        vecteur=affectation_vecteur(l)  
        longueur_vecteur=sousliste[l]  
  
        l=0  
        while l < longueur_vecteur: # Modification de la couleur  
            #des pions adverses encadrés par deux pions de la couleur du joueur  
  
            plateau_jeu[y+vecteur[0],x+vecteur[1]]= tour_du_joueur  
            y=y+vecteur[0]  
            x=x+vecteur[1]  
            l+=1  
        Liste_plateau_jeu.append(plateau_jeu)  
  
    arret_jeu=0  
    tour_du_joueur=Systeme[0]  
  
return tour_du_joueur,arret_jeu, Liste_plateau_jeu
```



Création de plateaux fictifs  
Changement de couleur et liste des plateaux  
fictifs pour ensuite simuler le plateau retournant  
un maximum de pions à t+2 ; puis t+3



# Schéma Bilan du fonctionnement de IA\_DIFFICILE

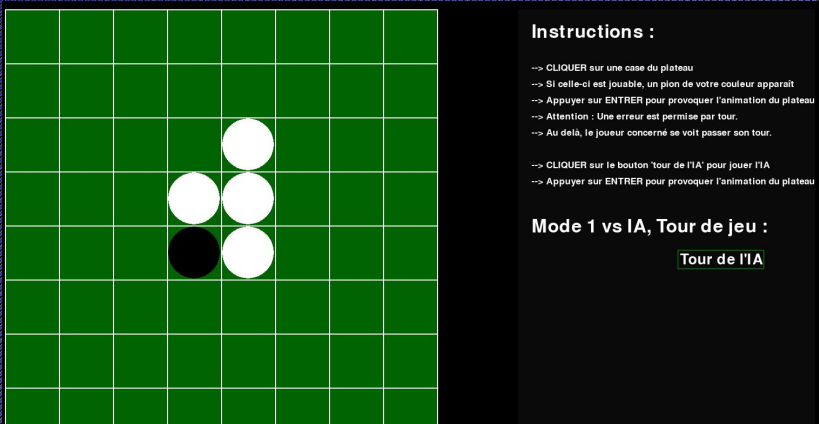
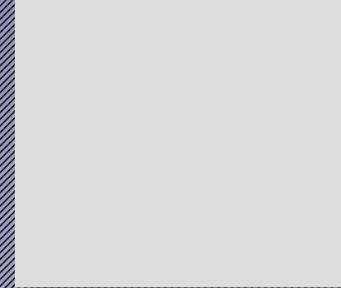


Après 3 tours de jeu : sélection du coup de l'IA à jouer à  $t+1$   
qui amène au plateau avec le maximum de pions de sa couleur à  $t+3$ .

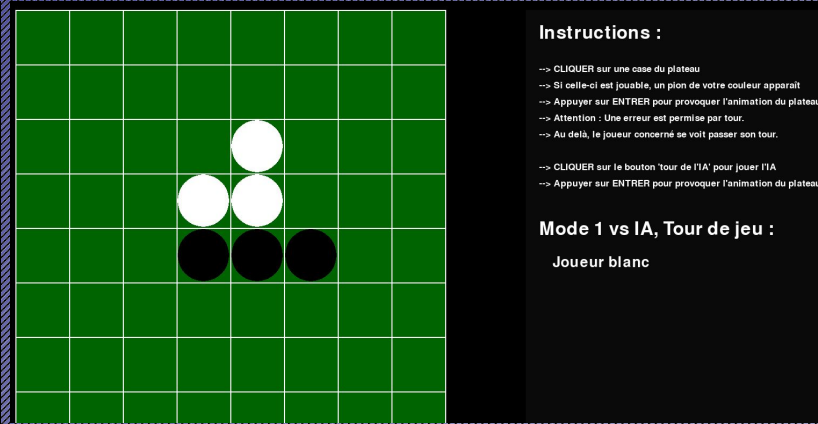


# FONCTION OTHELLO-GRAPHISME :

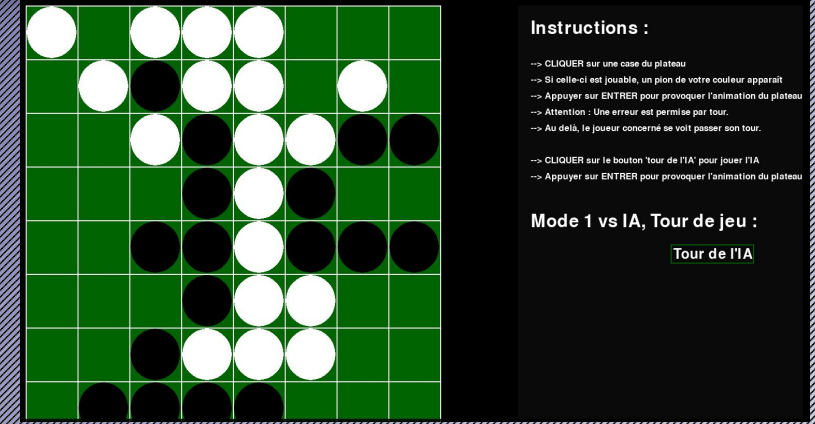
La fonction principale qui permet l'affichage et l'interface de jeu



Tour n° 1



Tour n° 2



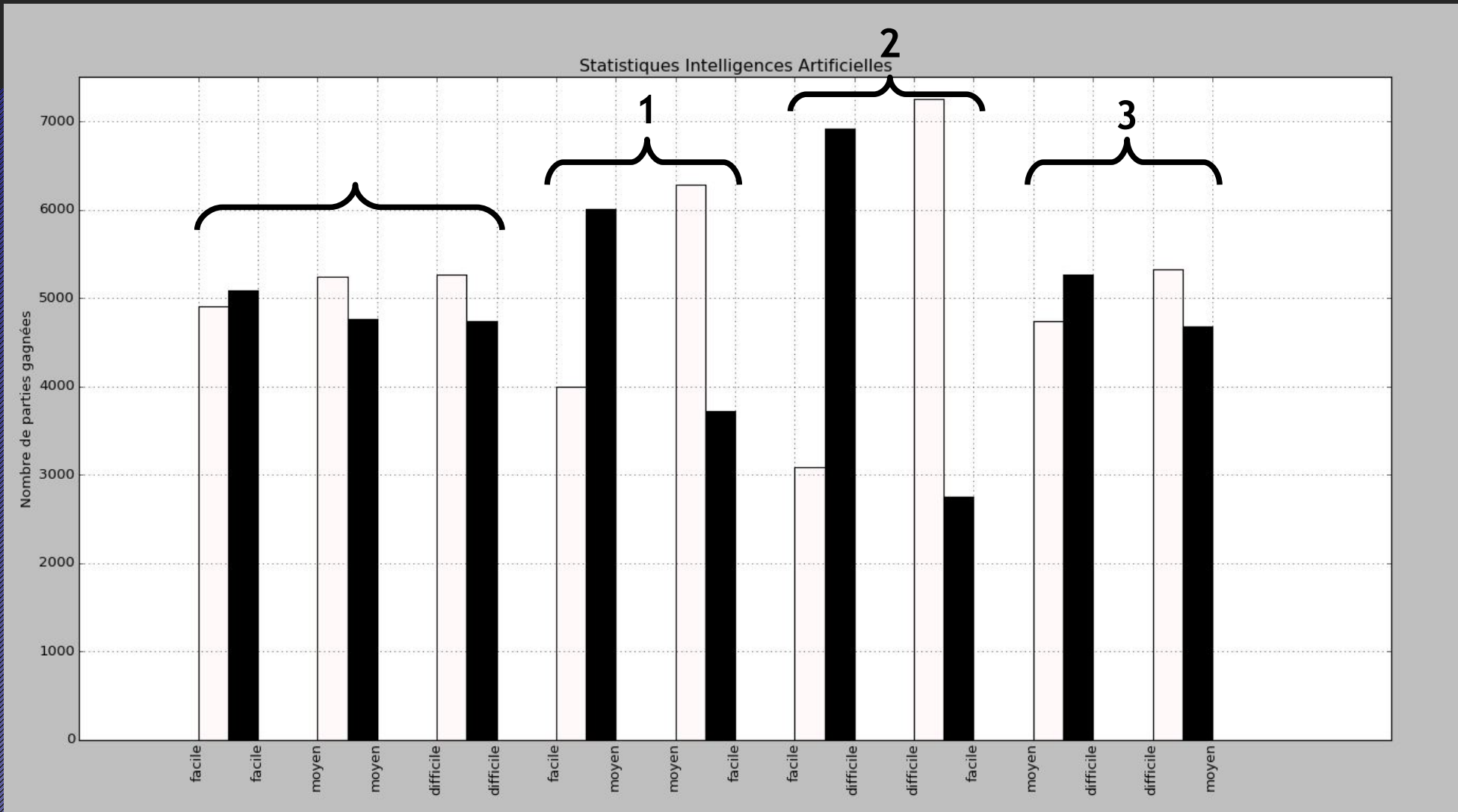
Tour n° 30

→ Partie 1vs IA en  
mode Difficile

→ Interaction avec l'interface graphique :  
clic souris et touche « Entrée »



# Comparaison des différentes IA et conclusion :



1 :  
Facile // Moyen  
38 // 62

2 :  
Facile // Difficile  
29 // 71

3 :  
Moyen // Difficile  
48 // 52