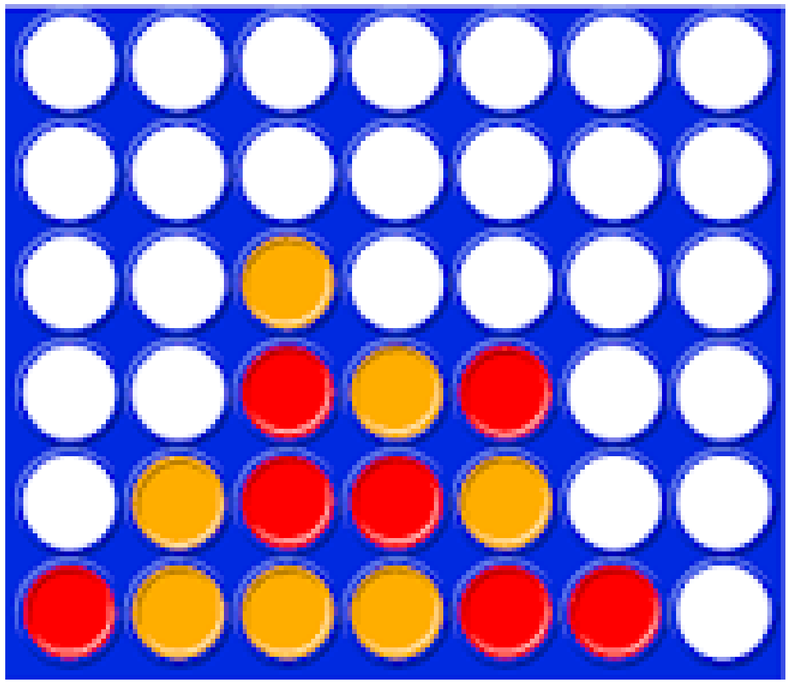
Le Puissance 4



Félix GRISONI

SOMMAIRE

Les Règles du jeu

- But du jeu :Le but du jeu est d'aligner quatre jetons de sa propre couleur (soit horizontalement verticalement ou en diagonale) avant son adversaire.

- Plateau du jeu : le jeu se déroule sur un plateau de 6 rangées par 7 colonnes soit 42 cases.

- Déroulement du jeu : Les joueurs jouent à tour de rôle en insérant un jeton dans la colonne de leur choix. Le jeton tombe au plus bas possible dans la colonne choisie, poussant tout jeton déjà présent vers le bas de la colonne.

- Fin de partie : Lorsqu'un joueur parvient à aligner quatre de ses jetons consécutifs horizontalement, verticalement ou en diagonale, il gagne la partie.

Le Canvas et les boutons

```python
class Damier(Canvas):                          # Création de la classe "Damier" qui hérite de "Canvas"

    # initialisation

    def __init__(self, parent):                # Méthode spéciale qui est appelée lorsqu'un objet est créé
        self.parent=parent                      # Définition de l'attribut "parent" à partir de l'argument passé à la méthode
        self.matrice=[9,9,9,9,9,9,9,9,9]*6      # Définition de l'attribut "matrice" qui est une liste de 42 éléments initialisée à 9
        self.couleur_pion=['','red','yellow','','','','','','','white'] # Définition de l'attribut "couleur_pion" qui est une liste de 10 éléments initiali
        self.tour=1                             # Définition de l'attribut "tour" à 1
        self.gagne=0                            # Définition de l'attribut "gagne" à 0
        self.last_move = []                     # Définition de l'attribut "last_move" comme une liste vide

        self.can = Canvas.__init__(self, width = 50+700+150, height =10+600+10, bg ='gainsboro') # Initialisation de l'objet Canvas
                                                                                                 # La méthode __init__() de la classe parente (Canvas) est

        self.annuler_bouton = Button(self.parent, text='Annuler', command=self.undo)  # Création d'un bouton pour annuler le dernier coup joué
        self.annuler_bouton.grid(row=13, column=1)                                     # Positionnement du bouton sur la grille avec la méthode grid()
        self.sauvegarde_bouton = Button(self.parent, text='Sauvegardé partie', command=self.sauvegarde) #Création d'un bouton pour sauvegardé la partie en
        self.sauvegarde_bouton.grid(row=8, column=10)              # Positionnement du bouton sur la grille avec la méthode grid()
        self.charge_sauvegarde_bouton = Button(self.parent, text='Chargé partie', command=self.charger_sauvegarde) #Création d'un bouton pour chargé la der
        self.charge_sauvegarde_bouton.grid(row=1, column=10)# Positionnement du bouton sur la grille avec la méthode grid()
        self.quitter_bouton = Button(self.parent, text='Quitter', command=self.quitter) #Création d'un bouton pour quitter le jeu
        self.quitter_bouton.grid(row=9, column=10) # Positionnement du bouton sur la grille avec la méthode grid()
        self.relancer_bouton = Button(self.parent, text='Relancer une partie', command= self.redemarrer) #Création d'un bouton pour relancer une partie
        self.relancer_bouton.grid(row=0, column=10) # Positionnement du bouton sur la grille avec la méthode grid()
        self.affiche_damier()                       # Appel de la méthode "affiche_damier" qui affiche le damier sur la fenêtre

        self.bind('<Button-1>', self.click)         # Appel de la méthode bind() pour associer un événement à une méthode (ici, clic gauche de la souris as

        fenetre_joueur=Label(self.parent, bg ='gainsboro', text = ' Joueur:')         # Création d'une étiquette (Label) pour afficher le joueur courant
        fenetre_joueur.grid(row=0, column=9)                                          # Positionnement de l'étiquette sur la grille
```

```python
menubar = Menu(self)                            # Création d'un menu
menu1 = Menu(menubar, tearoff=0)                # Création d'un sous-menu
menu1.add_command(label="Nouvelle partie", command=self.redemarrer)
menu1.add_command(label="Sauvegarder partie", command=self.sauvegarde)
menu1.add_command(label="Charger partie", command=self.charger_sauvegarde)      # Ajout d'
menu1.add_separator()                           # Ajout d'une séparation entre les commandes d
menu1.add_command(label="Quitter", command=self.quitter)                       # Ajout d'ur
menubar.add_cascade(label="Menu", menu=menu1)                                    # Ajout du
self.parent.config(menu=menubar)  # Configure le menu de la fenêtre parente.
```

Le Canvas et les boutons

```python
class ChoixJoueurs(tk.Frame):  # création de la classe ChoixJoueurs qui hérite de tk.Frame

    def __init__(self, master=None):  # définition du constructeur avec un paramètre optionnel master
        super().__init__(master)  # appel du constructeur de la classe parente (tk.Frame)
        self.master = master  # enregistrement du paramètre master comme attribut de la classe
        self.pack()  # affichage du widget principal (la fenêtre)
        self.create_widgets()  # appel de la méthode create_widgets pour créer les autres widgets

    def create_widgets(self):  # définition de la méthode create_widgets
        # création et affichage des labels et des zones de texte pour la saisie des noms des joueurs
        self.player1_label = tk.Label(self, text="Prénom joueur 1 :")
        self.player1_label.pack()
        self.player1_entry = tk.Entry(self)
        self.player1_entry.pack()

        self.player2_label = tk.Label(self, text="Prénom joueur 2 :")
        self.player2_label.pack()
        self.player2_entry = tk.Entry(self)
        self.player2_entry.pack()

        # création et affichage du bouton pour choisir un joueur aléatoire
        self.random_button = tk.Button(self, text="Choisir aléatoirement", command=self.choose_random)
        self.random_button.pack()

        # création et affichage du label pour afficher le résultat
        self.result_label = tk.Label(self, text="")
        self.result_label.pack()

        # création et affichage du bouton pour fermer la fenêtre
        self.lancer_button = tk.Button(self, text='Lancer Jeu', command=self.fermer)
        self.lancer_button.pack()
```

Les Fonctions du jeux

```python
def click(self, event):
    self.tombe_un_pion(int((event.x-50)/100), int((event.y-10)/100))  # Appelle la méthode "tombe_un_pion" avec les coordonnées du clic de souris.
    self.affiche_damier()  # Affiche le damier mis à jour.
    self.verifie_alignement()  # Vérifie s'il y a un alignement de pions.


def tombe_un_pion(self, colonne_cliquee, ligne_cliquee):
# Vérifie si la colonne cliquée est valide, si la case en bas de la colonne est vide et si personne n'a gagné
    if colonne_cliquee in [0,1,2,3,4,5,6] and self.matrice[colonne_cliquee+7*5]==9 and self.gagne==0:
        j=5
        arret=False
        # Boucle tant que le pion n'est pas tombé au fond de la colonne ou qu'il y a eu un arrêt
        while arret==False:
            # Vérifie si la case en dessous est vide et si oui, si c'est la bonne case où le pion doit tomber
            if self.matrice[colonne_cliquee+j*7]!=9:
                if (5-ligne_cliquee)==(j+1):
                    # Si le pion tombe dans la bonne case, on met à jour la matrice
                    self.matrice[colonne_cliquee+(j+1)*7]=(self.tour-1)%2+1
                    # On passe au tour suivant
                    self.tour=self.tour+1
                    # On enregistre le dernier coup joué
                    self.last_move.append((colonne_cliquee, j+1))
                    # On arrête la boucle, car le pion a atteint le fond de la colonne ou la case en dessous est occupée
                arret=True
            else:
                # Si la case en dessous est vide, on continue à faire tomber le pion
                # Si on atteint le fond de la colonne, on place le pion dans la première case vide en partant du bas
                if j==0:
                    if (5-ligne_cliquee)==0:
                        self.matrice[colonne_cliquee+j*7]=(self.tour-1)%2+1
                        self.tour=self.tour+1
                        self.last_move.append((colonne_cliquee, j+1))
                        # On arrête la boucle, car le pion a atteint le fond de la colonne
                    arret=True
                j=j-1
```

Les Fonctions du jeux

```python
def verifie_alignement(self):

    for i in range(7): #tester les alignements verticaux
        for j in range(3):
            resultat=self.matrice[i+j*7]+self.matrice[i+(j+1)*7]+self.matrice[i+(j+2)*7]+self.matrice[i+(j+3)*7]
            if resultat==4:
                self.gagne=1
                self.create_line(100+i*100, 560-j*100, 100+i*100, 560-(j+3)*100, width=8, fill = 'green')
                self.afficher_message_victoire(1)
            if resultat==8:
                self.gagne=2
                self.create_line(100+i*100, 560-j*100, 100+i*100, 560-(j+3)*100, width=8, fill = 'green')
                self.afficher_message_victoire(2)

    for j in range(6): #tester les alignements hortizontaux
        for i in range(4):
            resultat=self.matrice[i+j*7]+self.matrice[i+1+j*7]+self.matrice[i+2+j*7]+self.matrice[i+3+j*7]
            if resultat==4:
                self.gagne=1
                self.create_line(100+i*100, 560-j*100, 100+(i+3)*100, 560-j*100, width=8, fill = 'green')
                self.afficher_message_victoire(1)
            if resultat==8:
                self.gagne=2
                self.create_line(100+i*100, 560-j*100, 100+(i+3)*100, 560-j*100, width=8, fill = 'green')
                self.afficher_message_victoire(2)
```

```python
    for i in range(4): #tester les diagonales montantes
        for j in range(3):
            resultat=self.matrice[i+j*7]+self.matrice[i+1+(j+1)*7]+self.matrice[i+2+(j+2)*7]+self.matrice[i+3+(j+3)*7]
            if resultat==4:
                self.gagne=1
                self.create_line(100+i*100, 560-j*100, 100+(i+3)*100, 560-(j+3)*100, width=8, fill = 'green')
                self.afficher_message_victoire(1)
            if resultat==8:
                self.gagne=2
                self.create_line(100+i*100, 560-j*100, 100+(i+3)*100, 560-(j+3)*100, width=8, fill = 'green')
                self.afficher_message_victoire(2)

    for i in range(4): #tester les diagonales descendentes
        for j in range(3,6):
            resultat=self.matrice[i+j*7]+self.matrice[i+1+(j-1)*7]+self.matrice[i+2+(j-2)*7]+self.matrice[i+3+(j-3)*7]
            if resultat==4:
                self.gagne=1
                self.create_line(100+i*100, 560-j*100, 100+(i+3)*100, 560-(j-3)*100, width=8, fill = 'green')
                self.afficher_message_victoire(1)
            if resultat==8:
                self.gagne=2
                self.create_line(100+i*100, 560-j*100, 100+(i+3)*100, 560-(j-3)*100, width=8, fill = 'green')
                self.afficher_message_victoire(2)
```

Les Fonctions du jeux

```python
# Fonction Quitter

    def quitter(self):
        if askyesno('Confirmation Quitter', 'Quitter vraiment ?'): # Afficher une boîte de dialogue pour
            self.parent.destroy() # Fermer la fenêtre du jeu

    def undo(self):
        if self.last_move:  # Vérifier s'il y a un coup à annuler
            colonne, ligne = self.last_move.pop()  # Récupérer les coordonnées du dernier coup
            self.matrice[colonne + ligne * 7] = 9  # Retirer le pion de la matrice
            self.tour -= 1  # Réduire le compteur de tour de 1
            self.gagne = 0  # Réinitialiser le statut de victoire
            self.affiche_damier()  # Mettre à jour l'affichage du damier

    def sauvegarde(self):
        with open('sauvegarde', 'wb') as f:
            pickle.dump((self.matrice, self.tour, self.last_move), f)

    def charger_sauvegarde(self):
     if askyesno('Confirmation Charger partie', 'Charger vraiment ?'): # Afficher une boîte de dialogue pou
        with open('sauvegarde', 'rb') as f:
            matrice, tour, last_move = pickle.load(f)
            self.matrice = matrice
            self.tour = tour
            self.last_move = last_move
            self.affiche_damier()
```

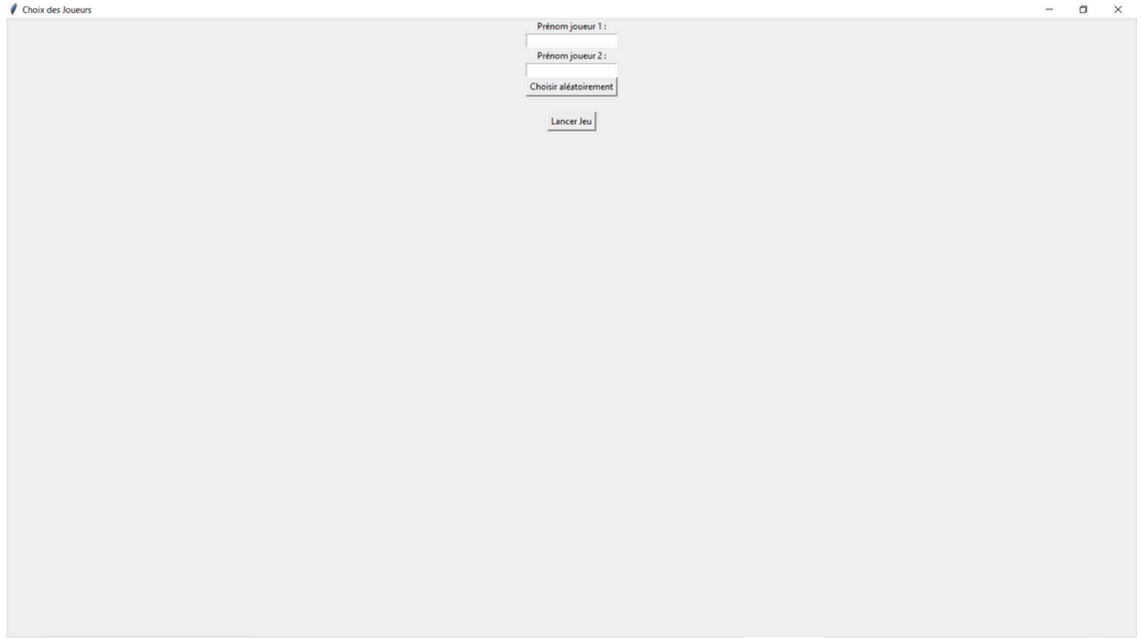
Les Fonctions du jeux

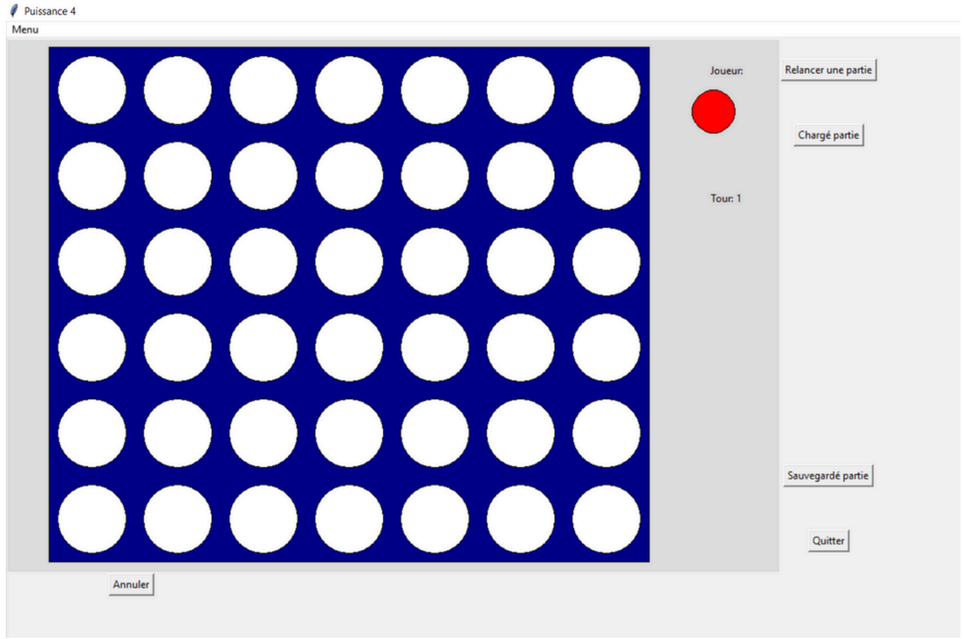
```python
def afficher_message_victoire(self, joueur):
    if joueur == 1:
        showinfo("Victoire", "Le joueur 1 (rouge) a gagné !")
    elif joueur == 2:
        showinfo("Victoire", "Le joueur 2 (jaune) a gagné !")

def afficher_message_egalite(self):
            if all(pion != 9 for pion in self.matrice):  # Vérifie si la grille est pleine
                if self.gagne == 0:  # Vérifie s'il n'y a pas de gagnant
                    showinfo("Égalité")


def affiche_damier(self):
    self.create_rectangle(50,10,750,610, fill='dark blue')  # Dessine un rectangle bleu pour représenter le plateau de jeu.
    self.grid(row=0, column=0, rowspan=10, columnspan=10)  # Place le canevas dans la fenêtre parente.
    self.create_oval (800, 60, 850, 110, fill=self.couleur_pion[(self.tour-1)%2+1])  # Dessine un cercle représentant le pion du joueur cou
    for i in range (7):
        for j in range (6):
            self.create_oval (50+i*100+10, 10+600-j*100-10, 50+i*100+90, 10+600-j*100-90, fill=self.couleur_pion[self.matrice[i+7*j]])  # D
    texte_tour='Tour: '+str(self.tour)
    fenetre_tour=Label(self.parent, text = texte_tour, bg ='gainsboro')  # Crée un label contenant le numéro de la tour courante.
    fenetre_tour.grid(row=2, column=9)  # Place le label dans la fenêtre parente.
```

Le Compte Rendu





Conclusion

Testons le jeu !