TWELVE MEN'S MORRIS

SUPINFO – Projet de fin d'année

*Documentation générale_*

Projet : TWELVE MEN'S MORRIS

Étudiant : Aguibou Sow

École : SUPINFO

Année académique : 2024 – 2025

TWELVE MEN'S MORRIS

Sommaire

1. Introduction.....
2. Objectifs du projet.....
3. Technologies utilisées.....
4. Fonctionnalités développées.....
   - 4.1. Création du plateau.....
   - 4.2. Phase 1 – Pose des pions.....
   - 4.3. Détection des moulins.....
   - 4.4. Suppression d'un pion adverse.....
   - 4.5. Passage automatique à la phase suivante.....
   - 4.6. Phase 2 – Mouvement des pions. .....
   - 4.7. Phase 3 – Le saut (vol libre) .....
   - 4.8. Phase 4 – Détection de la victoire.....
5. Erreurs et limitations.....
6. Structures de données et composants graphiques
   - 6.1. Structures de données utilisées.....
   - 6.2. Composants graphiques Tkinter.....
7. Conditions de fin du jeu.....
8. Choix techniques.....

TWELVE MEN'S MORRIS

9. Présentation des algorithmes.....

9.1. Algorithme de la pose.....

9.2. Algorithme du mouvement.....

9.3. Algorithme du saut.....

9.4. Algorithme du moulin.....

9.5. Algorithme de suppression.....

9.6. Algorithme de victoire.....

10. **Manuel du jeu Twelve Men's Morris**

10.1. Installation et lancement.....

10.2. Règles du jeu.....

10.3. Déroulement d'une partie.....

10.4. Interface.....

11. Conclusion.....

TWELVE MEN'S MORRIS

1 Introduction

Ce document présente la documentation technique du projet *Twelve Men's Morris*, un jeu de plateau développé dans le cadre du projet de fin d'année à SUPINFO.

Le jeu fonctionne sur les systèmes Windows, Mac OS et Linux. Il propose une interface interactive, un tour par tour, avec la gestion des phases (pose, mouvement, saut) et la détection des conditions de victoire.

Le but du projet est de proposer une version bureau complète de ce jeu stratégique traditionnel, en respectant les règles officielles. Le développement a été réalisé en Python, avec l'utilisation de la bibliothèque Tkinter pour la création de l'interface graphique.

TWELVE MEN'S MORRIS

2 Objectifs du projet

- Comprendre les règles du jeu **Twelve Men's Morris** et les modéliser en code.
- Implémenter une interface graphique fonctionnelle avec Tkinter.
- Décomposer le jeu en plusieurs phases de logique (pose, mouvement, saut).
- Offrir une expérience de jeu fluide et fidèle aux règles.
- Documenter toutes les étapes du développement pour permettre la lecture, la maintenance et l'évolution du projet.

3 Technologies utilisées

- Langage : Python 3
- Bibliothèque graphique : Tkinter
- Éditeur de code : Visual Studio Code
- Plateformes visées : Windows, Mac OS, Linux

TWELVE MEN'S MORRIS

3.1 Fonctionnalités développées

3.2 Création du plateau

Le plateau a été modélisé sous forme de grille de coordonnées avec des emplacements valides. Chaque case correspond à une position définie, ce qui permet une gestion précise des actions du joueur.

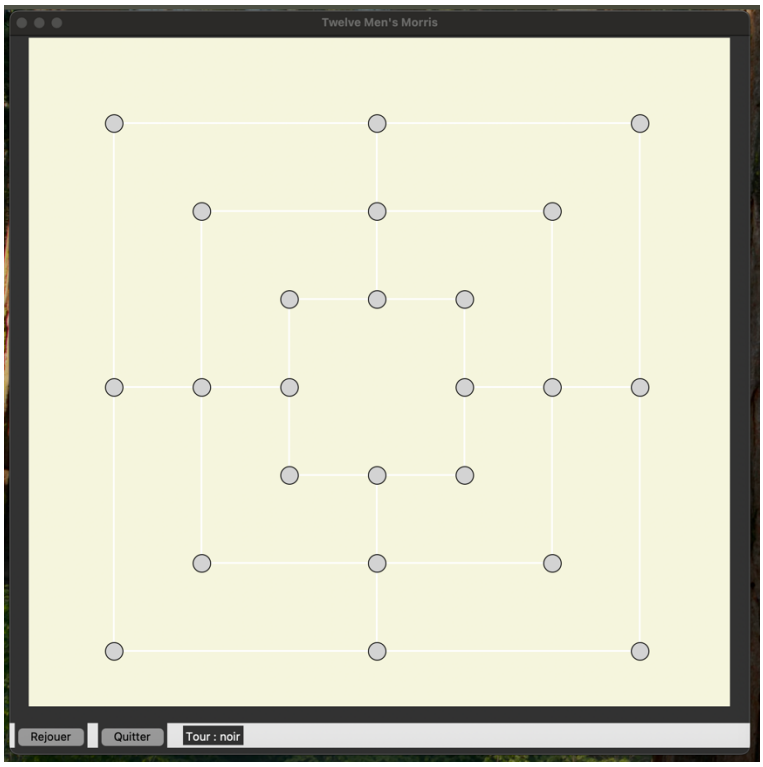


Figure 1:Le plateau est dessiné avec Tkinter (Canvas) : carrés concentriques, intersections valides et points de placement.

3.3 Phase 1 – Pose des pions

Chaque joueur dispose de 12 pions. À tour de rôle, ils posent un pion sur une case vide du plateau.

La logique mise en place :

- Vérification que la case est libre.
- Vérification du tour du joueur.
- Compte du nombre total de pions posés.

TWELVE MEN'S MORRIS

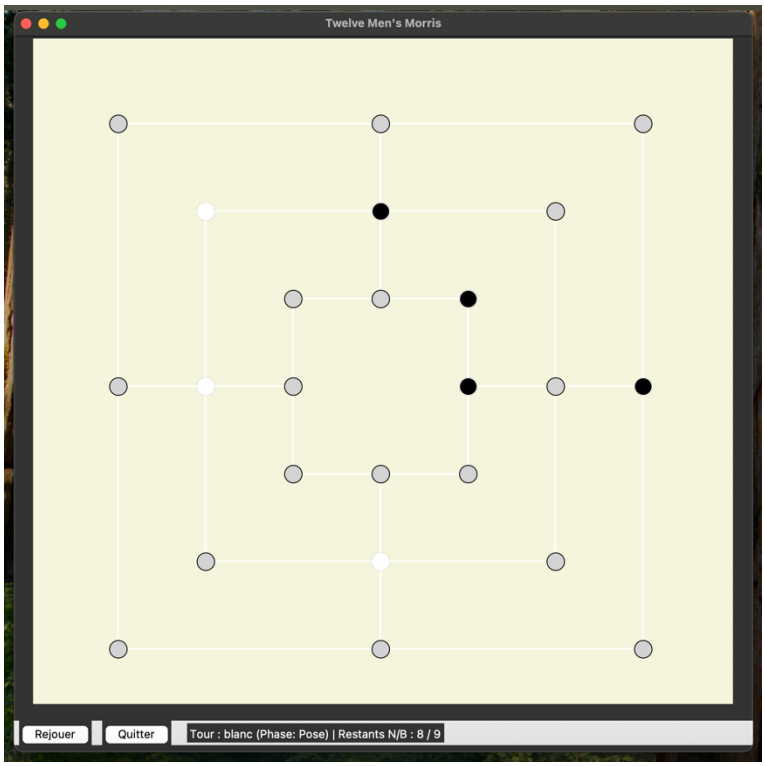


Figure 2: À tour de rôle, chaque joueur place un pion sur une case libre ; l'interface met à jour l'état et le tour en cours.

3.4 Détection des moulins

Un moulin est formé lorsqu'un joueur aligne trois de ses pions horizontalement ou verticalement. Le système détecte automatiquement cette situation après chaque pose ou déplacement.

TWELVE MEN'S MORRIS

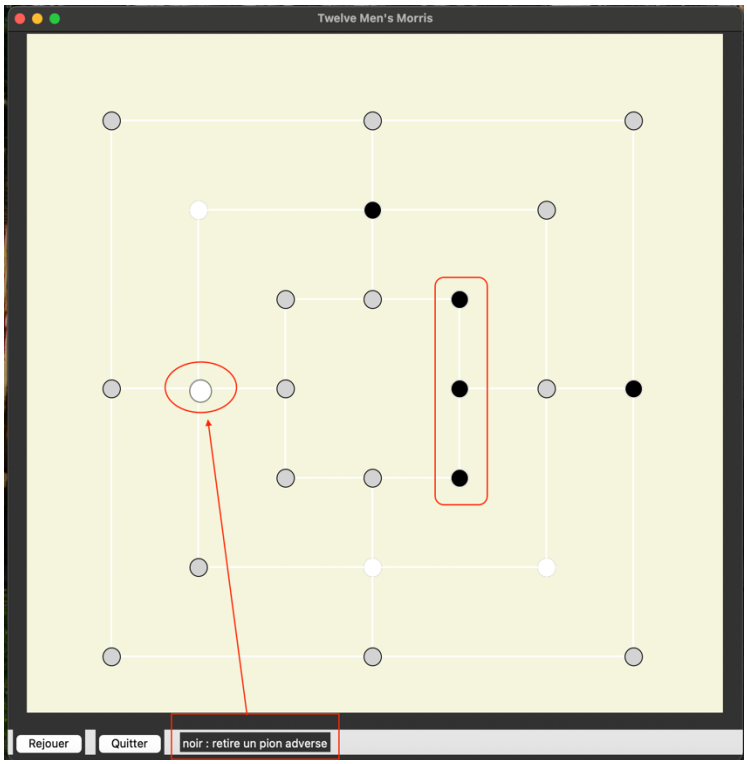


Figure 3: Après chaque pose/déplacement, le système vérifie les alignements de trois pions et signale la formation d'un moulin.

3.5 Suppression d'un pion adverse

Lorsqu'un joueur forme un moulin, il peut supprimer un pion adverse. Cette suppression est conditionnée : il est interdit de supprimer un pion faisant partie d'un moulin actif, sauf si aucun autre pion n'est disponible.

TWELVE MEN'S MORRIS

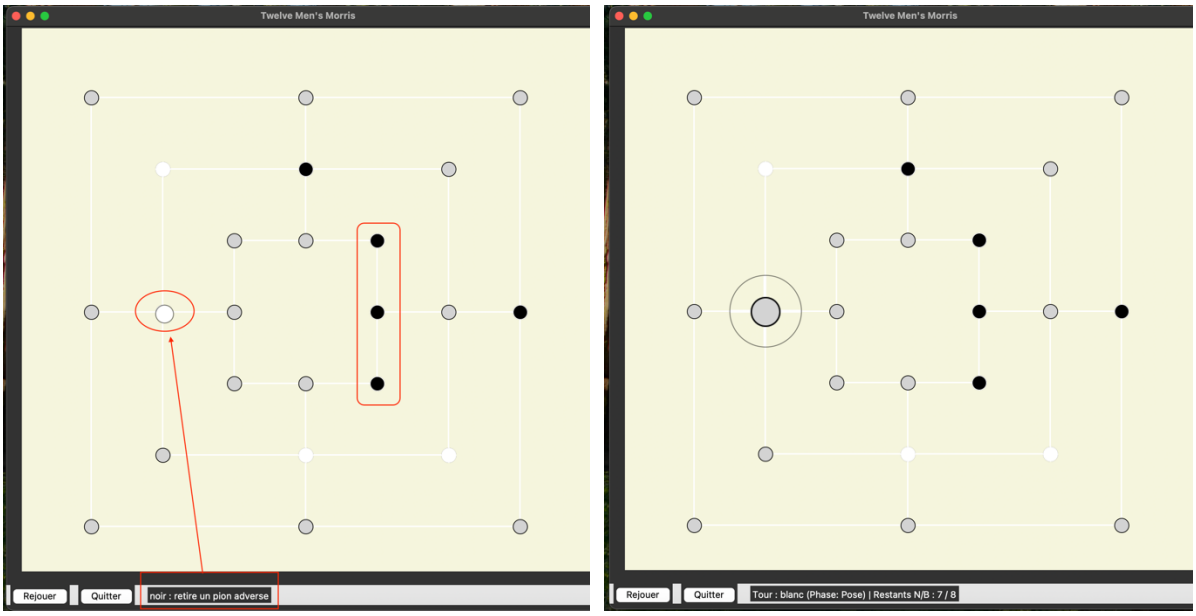


Figure 4: Quand un moulin est formé, le joueur supprime un pion adverse (hors moulin si possible)

3.6 Passage automatique à la phase suivante

Une fois les 24 pions posés (12 par joueur), le jeu passe automatiquement à la phase 2 : **le mouvement des pions.**

3.7 Phase 2 – Mouvement des pions

Une fois la phase de pose terminée, les joueurs déplacent leurs pions à tour de rôle.

Fonctionnement :

- Le joueur clique sur l'un de ses pions déjà placés.
- Puis clique sur une case vide adjacente (voisine).
- Si le déplacement est valide, le pion est déplacé.
- Si un moulin est formé, le joueur peut à nouveau supprimer un pion adverse.

Une fonction dédiée vérifie si la case cible est une voisine directe de la position initiale.

Cette phase est stratégique car elle permet aux joueurs de construire des moulins ou de bloquer ceux de l'adversaire.

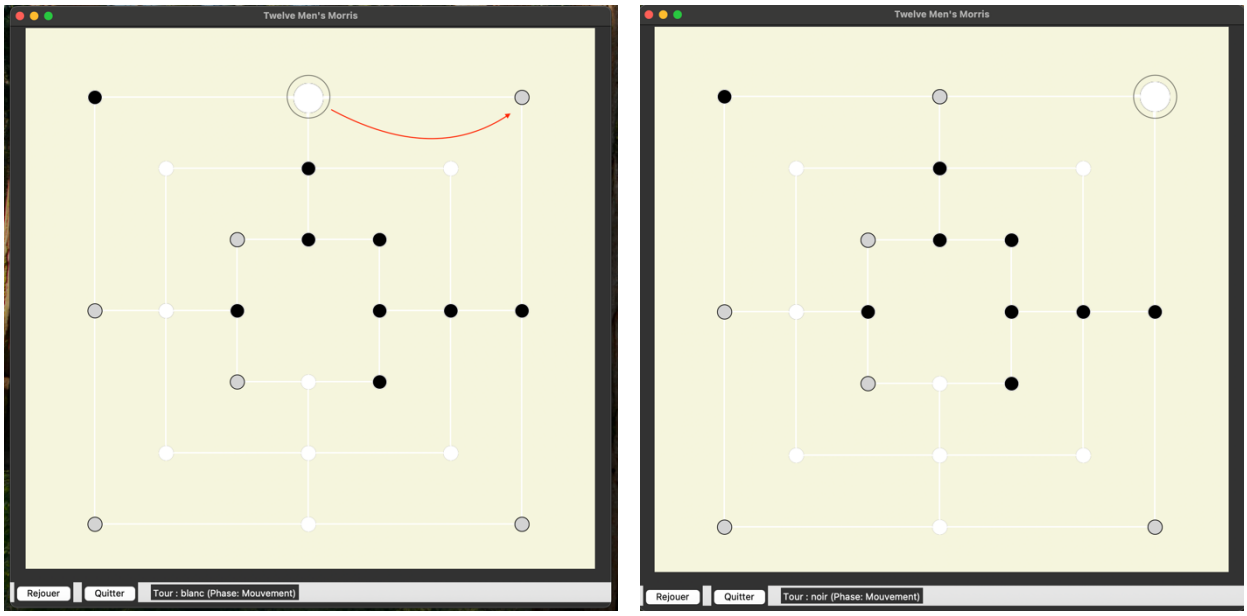


Figure 5: Le joueur sélectionne l'un de ses pions puis le déplace vers une case voisine libre ; les règles de moulin s'appliquent

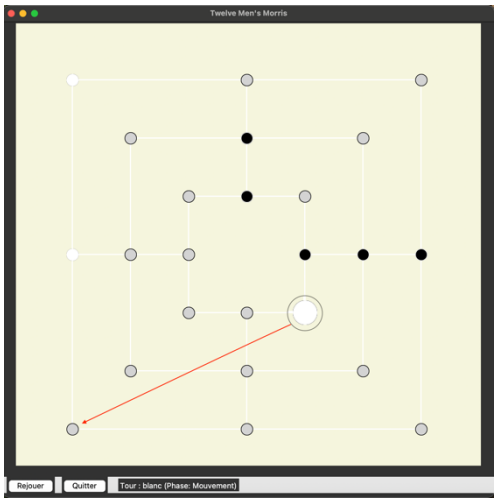
3.8 Phase 3 – Le saut (vol libre)

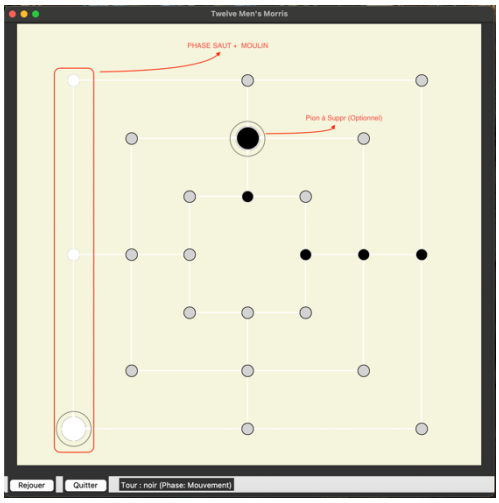
Lorsqu'un joueur tombe à trois pions, il peut les déplacer librement sur n'importe quelle case vide du plateau, sans contrainte de voisinage.

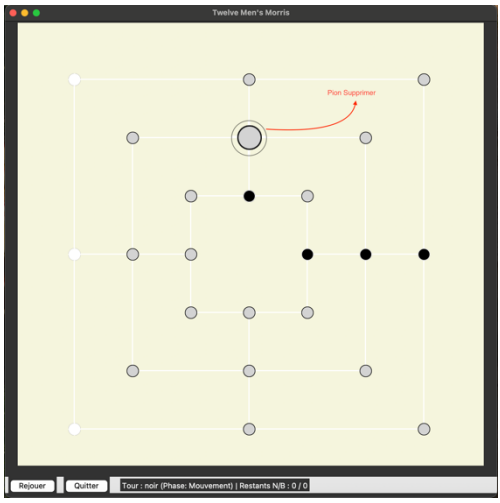
Cette règle spéciale a été ajoutée pour permettre au joueur en difficulté de continuer à jouer. Elle rend le jeu plus équilibré.

Le code détecte automatiquement cette situation et autorise le saut libre uniquement pour le joueur concerné.

TWELVE MEN'S MORRIS





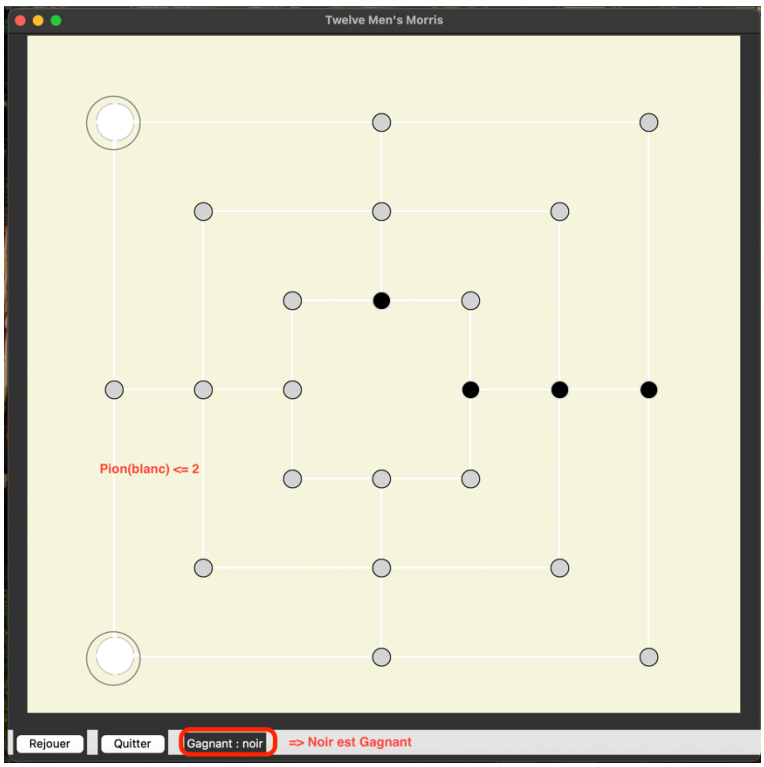


Fifure6: Avec 3 pions, un joueur peut se déplacer librement vers n'importe quelle case vide pour rester en jeu.

3.9 Phase 4 Détection de la victoire

Le jeu détecte une fin de partie lorsque :

- Un joueur possède moins de 3 pions.
- Un joueur ne peut plus effectuer de déplacement valide.
- Le système affiche alors un message de victoire ou d'égalité et propose de recommencer.



TWELVE MEN'S MORRIS

4 Erreurs et limitations

- Les pions ne peuvent pas être posés sur une case occupée.
- Aucun joueur ne peut dépasser les 12 pions.
- Les clics sont bloqués si une suppression est en cours ou si le jeu est terminé.

5 Structures de données et composants graphiques

1. Structures de données utilisées

- positions_occupees : dictionnaire qui indique qui occupe chaque case (noir/blanc).
- pions_a_poser : dictionnaire qui gère combien de pions chaque joueur doit encore poser.
- voisins : dictionnaire des cases voisines pour valider les déplacements.
- moulins : liste des combinaisons de trois positions qui forment un alignement.

1. Composants graphiques Tkinter

Composants graphiques utilisés

- Canvas : pour dessiner le plateau (carrés, lignes, cases).
- Ovales : pour représenter les pions (noirs ou blancs).
- Label : pour afficher le joueur en cours ou le gagnant.
- Boutons : ***Rejouer*** **et** ***Quitter.***

---

6 Conditions de fin du jeu

Le jeu se termine automatiquement dans les cas suivants :

- Un joueur a moins de trois pions sur le plateau.
- Un joueur ne peut plus se déplacer.
- Aucun mouvement possible pour les deux joueurs (match nul).

Dans chaque cas, un message est affiché et un bouton permet de recommencer une nouvelle partie ou Quitter

7 Choix techniques

- Tkinter a été choisi pour sa simplicité, sa portabilité et son intégration directe dans Python.
- Les actions de jeu sont gérées dans une fonction centrale → clic_souris(event) qui interprète les clics selon l'état actuel du jeu.
- Des variables globales contrôlent l'état du jeu : phase, joueur actif, pions restants, etc.
- Le code est commenté de manière simple, sans complexité inutile, pour faciliter la compréhension et la relecture.
- Chaque fonctionnalité a été validée progressivement afin de corriger les bugs au fur et à mesure et assurer la stabilité du jeu.

Nous avons choisi un fond beige et des points noirs visibles pour garder la lisibilité. Une musique de fond a été ajoutée (pygame) afin de rendre l'expérience plus immersive. Le design reste minimaliste mais pourra être enrichi par des thèmes visuels dans une future version.

8 Présentation des algorithmes

8.1 Algorithme de la pose :

1. Vérifier que la case cliquée est vide.
2. Placer le pion du joueur actuel.
3. Vérifier si un moulin est formé.
4. Si moulin, activer la suppression d'un pion adverse.
5. Décrémenter le compteur de pions restants à poser.

TWELVE MEN'S MORRIS

```
if phase_pose and pions_a_poser[joueur_actuel] == 0:
    autre = joueur_adverse(joueur_actuel)
    if pions_a_poser[autre] > 0 :
        print(f"{joueur_actuel} a fini de poser. tour auto passe a {autre}.")
        joueur_actuel = autre
        label_tour.config(text=f"Tour : {joueur_actuel} (Phase: Pose)")
        return
    else:
        mettre_a_jour_phase() # les 2 ont fini de poser
if i in positions_occupees:
    print(f"c mort frérot, position {i} déjà prise par {positions_occupees[i]}")
    return
if pions_a_poser[joueur_actuel] <= 0 :
    print(f"{joueur_actuel} n'a plus de pions a poser")
    return
couleur = "black" if joueur_actuel == "noir" else "white"
canvas.create_oval(px - rayon, py - rayon, px + rayon, py + rayon, fill=couleur)
positions_occupees[i] = joueur_actuel
print(f"{joueur_actuel} a mis un pion en {i}")
print(f"Nombre total de pion actifs: {len(positions_occupees)}")
pions_a_poser[joueur_actuel] -= 1
if pion_in_moulin(i, joueur_actuel):
    print(f" MOULIN ! {joueur_actuel} peut retirer un pion adverse")
    mode_suppresion = True
    joueur_en_suppression = joueur_actuel
    suppression_effectuee = False
    label_tour.config(text=f"{joueur_actuel} : retire un pion adverse")
    return
mettre_a_jour_phase()
joueur_actuel = joueur_adverse(joueur_actuel) #changement de joueur
label_tour.config(
    text=f"Tour : {joueur_actuel} "
        f"{'(Phase: Mouvement)' if not phase_pose else '(Phase: Pose)'} | "
        f"Restants N/B : {pions_a_poser['noir']} / {pions_a_poser['blanc']}"
)
if not phase_pose:
    verifier_victoire()
```

Pendant la phase de pose, on vérifie que la case est libre et que le joueur a encore des pions à poser. On dessine le pion, on détecte un éventuel moulin (puis suppression), on met à jour la phase (passage auto au mouvement quand les 24 pions sont posés) et on change de joueur

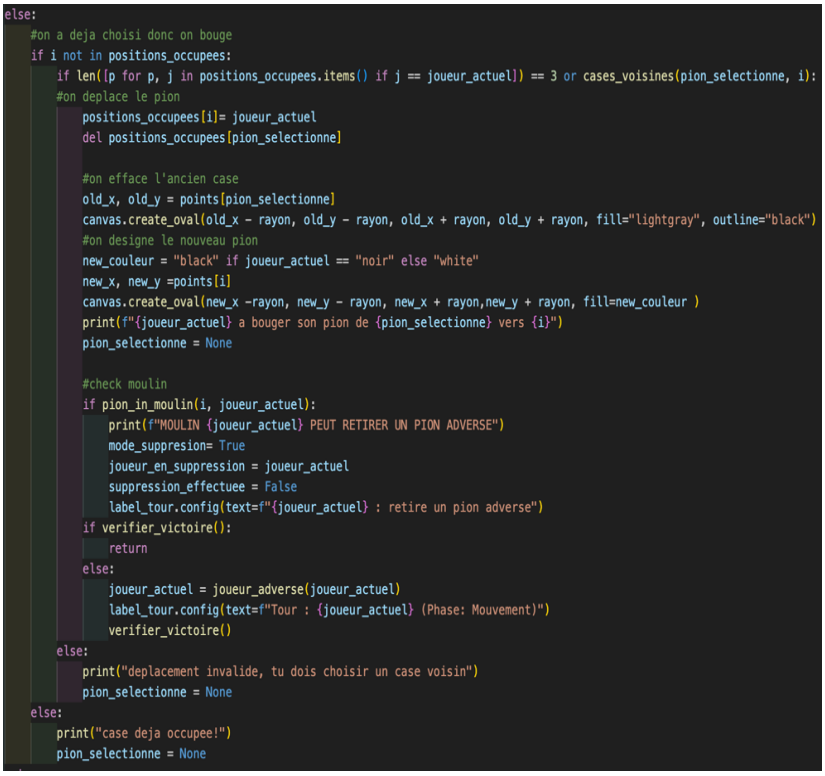
8.2 Algorithme du mouvement :

6. Le joueur sélectionne un de ses pions déjà posés.
7. Vérifier que la case cible est une voisine libre.
8. Déplacer le pion.
9. Vérifier si un moulin est formé et activer la suppression si besoin.

```
if not phase_pose :
    if pion_selectionne is None:
        #choisit un pion a bouger
        if i in positions_occupees and positions_occupees[i] == joueur_actuel:
            pion_selectionne= i
            print(f"{joueur_actuel} a selectionne le pion en {i}")
        else:
            print("selection invalide, faut choisir un de tes pion")
```

TWELVE MEN'S MORRIS

Légende **:** *En phase de mouvement, le joueur doit d'abord sélectionner l'un de ses pions déjà posés. Toute sélection d'une case vide ou d'un pion adverse est refusée*



```
else:
    #on a deja choisi donc on bouge
    if i not in positions_occupees:
        if len([p for p, j in positions_occupees.items() if j == joueur_actuel]) == 3 or cases_voisines(pion_selectionne, i):
        #on deplace le pion
            positions_occupees[i]= joueur_actuel
            del positions_occupees[pion_selectionne]

            #on efface l'ancien case
            old_x, old_y = points[pion_selectionne]
            canvas.create_oval(old_x - rayon, old_y - rayon, old_x + rayon, old_y + rayon, fill="lightgray", outline="black")
            #on designe le nouveau pion
            new_couleur = "black" if joueur_actuel == "noir" else "white"
            new_x, new_y =points[i]
            canvas.create_oval(new_x -rayon, new_y - rayon, new_x + rayon,new_y + rayon, fill=new_couleur )
            print(f"{joueur_actuel} a bouger son pion de {pion_selectionne} vers {i}")
            pion_selectionne = None

            #check moulin
            if pion_in_moulin(i, joueur_actuel):
                print(f"MOULIN {joueur_actuel} PEUT RETIRER UN PION ADVERSE")
                mode_suppresion= True
                joueur_en_suppression = joueur_actuel
                suppression_effectuee = False
                label_tour.config(text=f"{joueur_actuel} : retire un pion adverse")
            if verifier_victoire():
                return
            else:
                joueur_actuel = joueur_adverse(joueur_actuel)
                label_tour.config(text=f"Tour : {joueur_actuel} (Phase: Mouvement)")
                verifier_victoire()
        else:
            print("deplacement invalide, tu dois choisir un case voisin")
            pion_selectionne = None
    else:
        print("case deja occupee!")
        pion_selectionne = None
```

Le déplacement est autorisé vers une case voisine libre. Exception : si le joueur n'a plus que 3 pions, il peut « sauter » vers n'importe quelle case vide. Après déplacement : redessin du pion, détection de moulin, éventuelle suppression, puis changement de joueur et vérification de victoire

8.3 Algorithme du saut :

10. Détecter qu'un joueur n'a plus que 3 pions.
11. Permettre à ce joueur de déplacer un pion vers n'importe quelle case libre.
12. Vérifier moulin et suppression comme pour le mouvement.

8.4 Algorithme du moulin :

13. Parcourir toutes les combinaisons possibles (liste moulins).
14. Vérifier si les 3 positions d'une combinaison sont occupées par le même joueur.

15. Si oui, retourner “moulin formé”.

```python
def pion_in_moulin(pos, joueur):
    for moulin in moulins:
        if pos in moulin:
            if all(positions_occupees.get(p) == joueur for p in moulin):
                return True
    return False
```

Cette fonction parcourt toutes les combinaisons possibles de trois positions (moulins).
Si la position courante appartient à une combinaison et que les trois cases sont occupées par le même joueur, la fonction retourne True → un moulin est formé.

8.5 Algorithme de suppression :

16. Si moulin formé, attendre que le joueur clique sur un pion adverse.
17. Vérifier que ce pion ne fait pas partie d'un moulin adverse (sauf si tous ses pions sont dans des moulins).
18. Retirer le pion choisi.

```python
if mode_suppresion:

    pion_adverse = joueur_adverse(joueur_en_suppression)
    tous_dans_moulin = all(pion_in_moulin(pos, pion_adverse) for pos in positions_occupees if positions_occupees[pos] == pion_adverse)
    for i, (px, py) in enumerate(points):
        distance = ((px - event.x) ** 2 + (py - event.y) ** 2) ** 0.5
        if distance <= 10:
            if (
                not suppression_effectuee
                and i in positions_occupees
                and positions_occupees[i] == pion_adverse
            ):
                if not pion_in_moulin(i, pion_adverse) or tous_dans_moulin:
                    canvas.create_oval(px - 10, py - 10, px + 10, py + 10, fill="lightgray", outline="black")
                    del positions_occupees[i]
                    print(f"pion adverse retiré en {i}")
                    suppression_effectuee = True
                    mode_suppresion = False
                    if verifier_victoire():
                        return

                    joueur_actuel = joueur_adverse(joueur_en_suppression)
                    joueur_en_suppression = None
                    label_tour.config(
                        text=f"Tour : {joueur_actuel} "
                            f"{'(Phase: Mouvement)' if not phase_pose else '(Phase: Pose)'} | "
                            f"Restants N/B : {pions_a_poser['noir']} / {pions_a_poser['blanc']}"
                    )
                    return
```

Légende : *Après la formation d'un moulin, le joueur peut retirer un pion adverse. On interdit de retirer un pion dans un moulin sauf si tous les pions adverses sont en moulin. Mise à jour de l'UI et passage de tour, avec vérification immédiate de la victoire*

8.6 Algorithme de victoire :

19. Vérifier si un joueur a moins de 3 pions.
20. Vérifier si un joueur ne peut plus se déplacer.
21. Si l'une des conditions est remplie, déclarer la victoire de l'adversaire.

TWELVE MEN'S MORRIS

```python
def verifier_victoire():
    # 1) Victoire "moins de 3 pions
    for joueur in ["noir", "blanc"]:
        pions_joueur = [pos for pos, j in positions_occupees.items() if j == joueur]
        if len(pions_joueur) < 3:
            gagnant = joueur_adverse(joueur)
            print(f"Victoire de {gagnant} ! (l'adversaire n'a plus que {len(pions_joueur)} pions)")
            label_tour.config(text=f"Gagnant : {gagnant}")
            canvas.unbind("<Button-1>")
            return True

    # 2) Victoire par "blocage"
    if not phase_pose:
        # On test le blocage uniquement pour le joueur qui va jouer (joueur_actuel)
        joueur = joueur_actuel
        pions_joueur = [pos for pos, j in positions_occupees.items() if j == joueur]

        # avec 3 pions, il peut sauter => pas de blocage possible
        if len(pions_joueur) >= 4:
            a_un_coup = any(
                any((v not in positions_occupees) and cases_voisines(pos, v) for v in range(24))
                for pos in pions_joueur
            )
            if not a_un_coup:
                gagnant = joueur_adverse(joueur)
                print(f"Victoire de {gagnant} ! (joueur {joueur} bloqué)")
                label_tour.config(text=f"Gagnant : {gagnant}")
                canvas.unbind("<Button-1>")
                return True

    return False
```

Cet algorithme vérifie deux conditions de fin de partie :

1. **Moins de 3 pions** → le joueur perd automatiquement.
2. **Blocage** → si un joueur ne peut plus effectuer de coup valide (pas de voisin libre, sauf en phase de saut), il perd.

Dans les deux cas, le gagnant est affiché et la partie est arrêtée.

---

9 Manuel du jeu TWELVE MEN'S MORRIS

9.1 Installation et lancement

- Installer Python 3.
- Aller dans le dossier du projet.
- Lancer la commande : python3 src/main.py

9.2 Règles du jeu

Le jeu Twelve Men's Morris se joue à deux. Chaque joueur dispose de 12 pions (noirs ou blancs). L'objectif est de réduire l'adversaire à moins de 3 pions ou de le bloquer.

Chaque joueur dispose de 12 pions (noirs et blancs).

- Phase 1 – Pose des pions :

Les joueurs posent leurs pions à tour de rôle sur une case vide.

TWELVE MEN'S MORRIS

Si un joueur aligne **3 pions consécutifs**, il forme un moulin et peut retirer un **pion adverse.**

(On ne peut retirer un pion d'un moulin adverse que si aucun autre pion n'est disponible).

- Phase 2 – Mouvement :

Quand tous les pions sont posés, les joueurs déplacent leurs pions à tour de rôle vers une **case voisine libre**.

Un moulin formé permet encore de retirer un pion adverse.

- Phase 3 – Saut :

Lorsqu'un joueur n'a plus que **3 pions**, il peut les déplacer vers **n'importe quelle case libre** du plateau.

- Victoire :

Un joueur gagne si son adversaire n'a plus que **2 pions** ou s'il ne peut plus jouer.

- Match nul :

Si le plateau est plein et que personne n'a gagné, la partie est déclarée nulle.

1. Déroulement d'une partie

- Le joueur noir commence.
- Phase 1 : poser les pions.
- Phase 2 : déplacer les pions (cases voisines).
- Phase 3 : sauter quand il reste 3 pions.
- Former un moulin → retirer un pion adverse.
- Victoire si l'adversaire a < 3 pions ou est bloqué.

2. Interface

- Clic souris pour poser/déplacer.
- Bouton Rejouer : recommencer une partie.
- Bouton Quitter : fermer le jeu.

10Conclusion

Ce projet m'a permis d'appliquer mes connaissances en programmation Python dans un cadre concret et complet. Le fait de devoir structurer un jeu par étapes m'a appris à raisonner comme un développeur, à anticiper les erreurs, à organiser mon code, et à construire une application graphique interactive.

Le projet est complet et fonctionnel.

Il respecte les règles officielles et propose une base solide pour des améliorations futures (IA, graphismes avancés, sauvegarde de partie).

Ce projet représente pour moi une étape importante dans ma progression, car je l'ai réalisé entièrement en comprenant chaque ligne de code. La documentation me permet de garder une trace claire de mon raisonnement et de mon avancement.