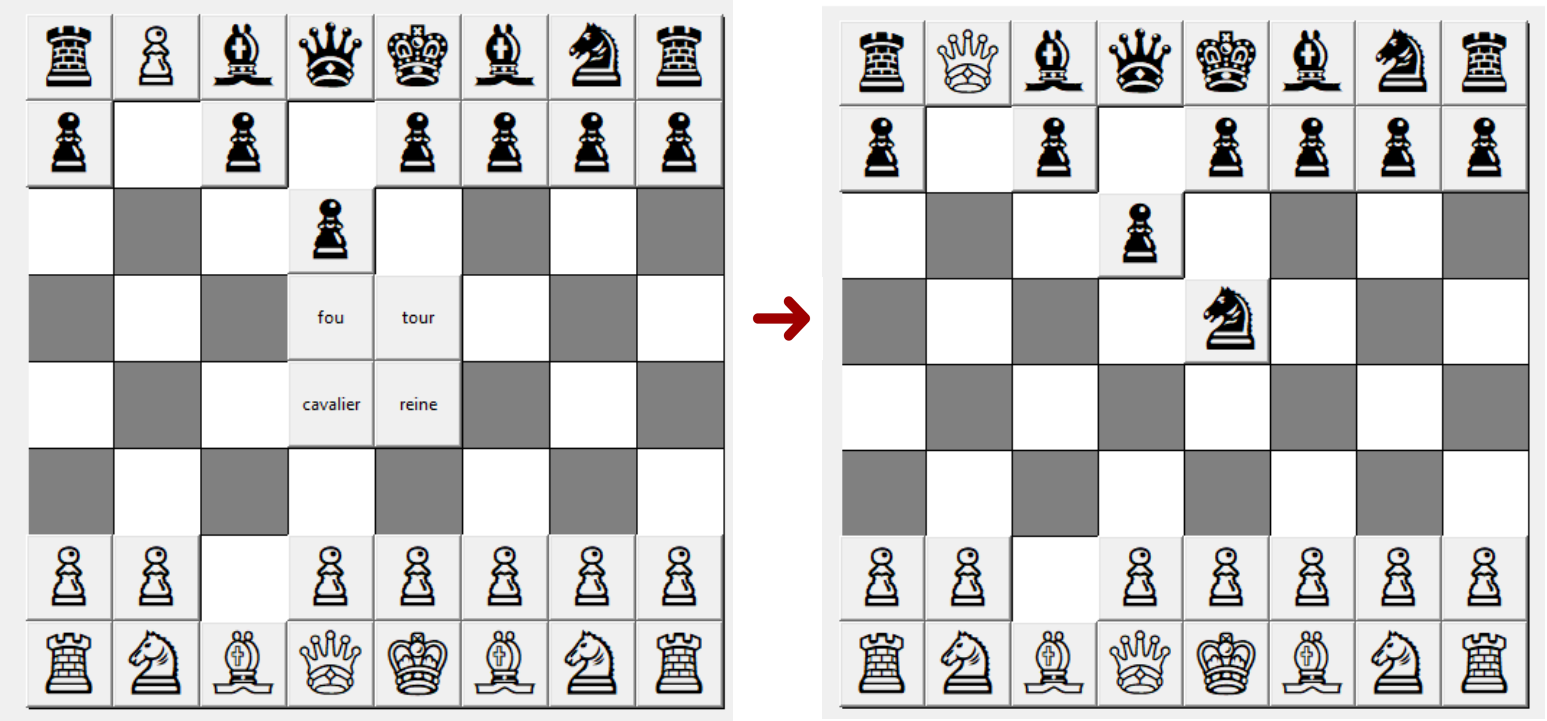
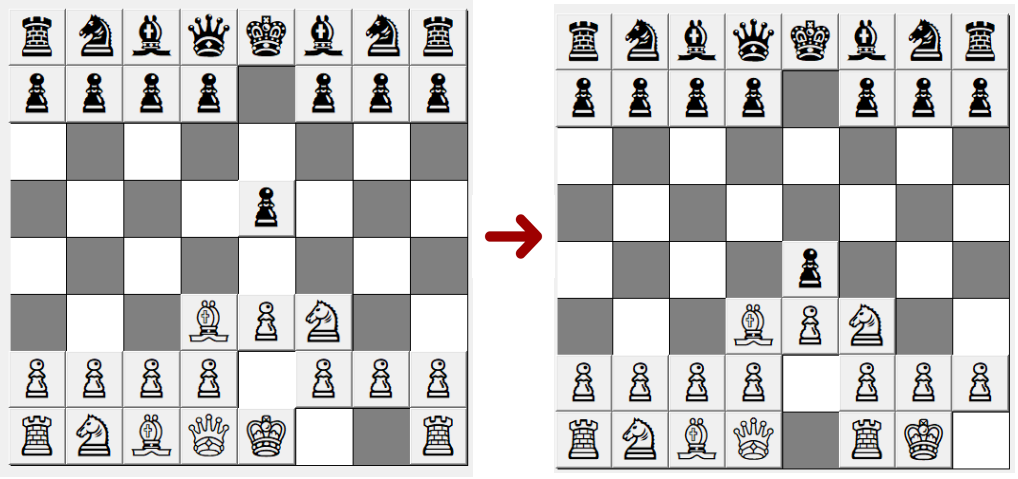
Compte-rendu

Objectifs et résultats :

Nous sommes amplement satisfaites de notre projet. Comme nous l'avions dit dans le document pré-projet, nous nous étions fixées comme objectif de pouvoir déplacer les pièces et d'appliquer les règles de base des échecs. Nous avons donc largement dépassé nos ambitions puisque nous avons mis en place un contrôle des déplacements, la possibilité de roquer, la promotion de pion ou encore la détection des échecs et mat et des cas d'égalité.



Promotion du pion blanc



Roque des blancs

Construction globale du projet :

Nous avons commencé par créer l'interface graphique de base, l'échiquier (nous nous sommes inspirées d'Internet pour créer l'échiquier) et les pièces, grâce à la classe Piece. Puis nous avons cherché à pouvoir déplacer une pièce quand on la sélectionne, grâce aux boutons. Nous avons ensuite codé la notion de tour de jeu (fonction tour_de_jeu) et ajouté petit à petit toutes les fonctions nécessaires au jeu : d'abord un premier contrôle des mouvements basique, ensuite la capture des pièces puis la fonction échec,...

Quelques difficultés rencontrées:

Les difficultés rencontrées sont nombreuses :

- Interagir avec l'échiquier : nous avions prévu à l'origine de représenter les 64 cases par des boutons mais nous avons vite changé d'avis car, avec les 32 pièces en plus , cela faisait trop de boutons à gérer et la solution ne paraissait pas optimale. Nous avons donc opté pour un échiquier uniquement graphique en créant les cases avec la fonction create_rectangle de Tkinter. Nous avons ensuite eu du mal à récupérer les coordonnées du clic pour le déplacement souhaité des pièces. Après quelques recherches, nous avons trouvé la fonction winfo_pointerxy() mais elle renvoyait les coordonnées par rapport à la fenêtre et pas au sein de l'échiquier. Nous avons finalement trouvé une solution : soustraire à cette fonction les coordonnées de l'origine de l'échiquier, qu'on obtient avec la fonction winfo_rootx().
- Nous n'arrivions pas à coder le déplacement d'une pièce dans le bon ordre : sélectionner d'abord la pièce, puis la case de destination. Notre problème était de stocker la pièce sélectionnée pour pouvoir la bouger après avoir cliqué sur la case voulue. Un moyen envisagé était d'utiliser les fonctionnalités command et lambda du bouton de la pièce pour passer en argument la pièce à déplacer comme dans le programme ci-dessous :

```
#on associe au bouton la fonction deplacement en passant la pièce comme argument
cav=Button(echiquier,text='♞',command=lambda: deplacement(cav))
def deplacement(piece):
      while clic==False: #on attend jusqu'à ce qu'une case soit sélectionnée
            if clic==True:   break
      piece.place(x=x_clic,y=yclic) #puis on déplace la pièce
```

Mais cela ne fonctionnait pas car tant que le bouton était “en attente”, il était enfoncé et on ne pouvait cliquer nulle part ailleurs donc il était impossible de sélectionner la case voulue. Nous avons finalement réussi en stockant la pièce dans une variable globale (voir fonction tour_de_jeu et Piece.dep).

- Plus généralement, nous avons dû apprendre à maîtriser Tkinter et les classes, que nous n’avions jamais manipulées auparavant.

<u>Améliorations possibles :</u>

- Tourner l’échiquier à chaque tour pour que chaque joueur soit dans le sens de ses pièces
- Permettre la prise en passant (pour les pions)
- Montrer les cases des déplacements possibles
- Choisir la situation de départ des pièces (merci Monsieur pour l’idée)