A logo with a black border

Description automatically generated

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***--FOX--- AND--- GEESE--***

**Documentation Technique**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Auteur : DAHOUI David

## Campus : Tours

## Date : 10/08/2023

## Année scolaire 2022-2023

# SOMMAIRE :

1. **Introduction**
   * 1.1. Présentation du jeu "Fox and Geese"
   * 1.2. Objectifs de la documentation
2. **Justification du choix du langage et de la librairie graphique**
   * 2.1. Avantages de Python pour le développement de jeux
   * 2.2. Pourquoi Tkinter ? Ses atouts pour le développement d'interfaces graphiques
   * 2.3. Comparaison avec d'autres langages et librairies
3. **Description des structures de données par fichiers**

* main.py
* start\_page.PY
* PLAYER\_SELECTION\_PAGE.PY
* animal\_selection\_page.py
* game\_page.py

1. **Présentation des algorithmes de gestion de la grille**
   * 4.1. Initialisation de la grille
   * 4.2. Mise à jour de la grille après chaque mouvement
   * 4.3. Vérification de la validité des mouvements sur la grille
2. **Présentation des algorithmes implémentant les déplacements et les captures**
   * 5.1. Algorithmes de déplacement pour le renard
   * 5.2. Algorithmes de déplacement pour les oies
   * 5.3. Gestion des captures
   * 5.4. Conditions de victoire et de défaite
3. **Conclusion**
   * 6.1. Retour sur les défis rencontrés lors du développement
   * 6.2. Pistes d'améliorations et évolutions futures
4. **Annexes**
   * 7.1. Code source complet
   * 7.2. Bibliographie et ressources

**1. Introduction**

*1.1. Présentation du jeu "Fox and Geese"*

"Fox and Geese" est un jeu de société traditionnel joué en Europe depuis le Moyen Âge. Le jeu se déroule sur un plateau de jeu en forme de croix, généralement divisé en 33 intersections ou "cases". Un joueur contrôle un seul renard, tandis que l'autre joueur contrôle 13 oies. L'objectif pour le renard est de capturer les oies, tandis que l'objectif pour les oies est de bloquer le renard afin qu'il ne puisse plus bouger.

*1.2. Objectifs de la documentation*

Cette documentation vise à fournir un aperçu technique détaillé du jeu "Fox and Geese" développé en Python en utilisant la librairie graphique Tkinter. Elle couvrira les raisons derrière le choix du langage et de la librairie, une description des structures de données utilisées, ainsi que les algorithmes mis en œuvre pour gérer la grille de jeu, les déplacements et les captures.

**Règles du jeu "Fox and Geese"**

1. **Plateau de jeu**:
   * Le plateau est constitué de 33 intersections ou "cases".
   * Les oies commencent le jeu sur une des extrémités de la croix, tandis que le renard commence au centre.
2. **Objectifs**:
   * **Renard**: Capturer autant d'oies que possible, de manière à ce que les oies restantes ne puissent pas le bloquer.
   * **Oies**: Bloquer le renard afin qu'il ne puisse plus bouger.
3. **Mouvements**:
   * **Renard**: Peut se déplacer dans n'importe quelle direction (horizontalement, verticalement ou en diagonale) d'une seule case à la fois. Il peut également sauter par-dessus une oie pour la capturer, à condition qu'il y ait une case vide de l'autre côté de l'oie.
   * **Oies**: Peuvent se déplacer d'une seule case à la fois, mais seulement vers l'avant (verticalement ou en diagonale vers l'avant).
4. **Capture**:
   * Le renard capture une oie en sautant par-dessus elle, à condition qu'il y ait une case vide de l'autre côté. Une oie capturée est retirée du plateau.
5. **Fin du jeu**:
   * Le jeu se termine lorsque le renard est bloqué et ne peut plus bouger ou lorsque suffisamment d'oies ont été capturées, rendant impossible pour elles de bloquer le renard.

**2. Justification du choix du langage et de la librairie graphique**

**2.1. Avantages de Python pour le développement de jeux**

Python est un langage de programmation polyvalent, offrant de nombreux avantages pour le développement de jeux, notamment :

* **Simplicité et lisibilité :** Le langage est conçu pour être facile à lire et à écrire. Cette simplicité aide les développeurs à écrire du code plus rapidement, ce qui est essentiel lors du prototypage ou du développement de jeux.
* **Richesse des bibliothèques :** Python dispose d'une vaste bibliothèque standard et de nombreuses bibliothèques tierces, y compris pour le développement de jeux, ce qui réduit la nécessité de coder des fonctionnalités à partir de zéro.

**2.2. Pourquoi Tkinter ? Ses atouts pour le développement d'interfaces graphiques**

Tkinter est l'une des bibliothèques GUI (Graphical User Interface) les plus anciennes pour Python, et elle offre plusieurs avantages :

* **Intégration native avec Python :** En tant que partie intégrante de la bibliothèque standard de Python, Tkinter ne nécessite pas d'installation supplémentaire, ce qui facilite le démarrage du développement.
* **Facilité d'utilisation :** Tkinter est conçu pour être simple et intuitif, permettant aux développeurs de créer rapidement des interfaces utilisateur sans avoir besoin d'une expertise approfondie en développement d'interfaces graphiques.
* **Portabilité :** Les applications développées avec Tkinter peuvent être exécutées sur la plupart des plateformes sans modification, ce qui facilite leur distribution.

**2.3. Comparaison avec d'autres langages et librairies**

Bien que Python et Tkinter offrent de nombreux avantages, il est important de mentionner comment ils se comparent à d'autres options populaires dans le domaine du développement de jeux :

* **Python vs. C++ :** Alors que C++ est souvent cité pour sa performance, Python offre une rapidité de développement inégalée. Pour des jeux comme "Fox and Geese", où les performances ultra-élevées ne sont pas essentielles, Python est un excellent choix.
* **Tkinter vs. Pygame :** Pygame est une autre bibliothèque populaire pour le développement de jeux en Python. Cependant, pour des jeux basés sur des grilles avec des interactions simples, Tkinter est souvent suffisant et plus facile à utiliser pour les débutants.

**3. DESCRIPTION DES STRUCTURES DE DONNEES**

**3.1. MAIN.PY**

**Modules et classes importés** :

* **tkinter** : Module utilisé pour créer des interfaces graphiques (GUI) avec Python.
* **ttk** : Un sous-module de **tkinter** qui offre des widgets supplémentaires et un moyen de styliser l'interface.
* **IntVar** : Une classe spéciale de **tkinter** pour déclarer et manipuler des variables entières.
* **StartPage, PlayerSelectionPage, AnimalSelectionPage, GamePage** : Ces classes représentent les différentes pages (ou écrans) du jeu.

**Classe Game** :

* **Attributs** :
  + **window** : Fenêtre principale de l'application.
  + **frames** : Dictionnaire pour stocker les différents écrans du jeu.
  + **container** : Conteneur principal pour les frames/pages.
  + **player\_count** : Variable pour stocker le choix de l'utilisateur concernant le nombre de joueurs.
  + **fox\_count** : Variable pour stocker le choix de l'utilisateur concernant le nombre de renards.
  + **goose\_count** : Variable pour stocker le choix de l'utilisateur concernant le nombre d'oies.
* **Méthodes** :
  + **show\_frame(cont)** : Affiche le frame/page spécifié.
  + **start\_game()** : Démarre une nouvelle partie en créant une nouvelle instance de **GamePage**.
  + **run()** : Exécute la boucle principale de l'application.

**Fonction main()** : Cette fonction initialise et exécute le jeu.

**Bloc conditionnel if \_\_name\_\_ == "\_\_main\_\_":** : Ce bloc conditionnel garantit que le jeu est initialisé et exécuté uniquement lorsque le script est exécuté directement (et non importé en tant que module).

**3.2. START\_PAGE.PY**

**Modules et classes importés** :

* **tkinter** : Module utilisé pour créer des interfaces graphiques (GUI) avec Python.
* **Image, ImageTk** : Classes du module **PIL** (Pillow) pour le traitement d'images.
* **messagebox** : Sous-module de **tkinter** pour afficher des boîtes de dialogue.
* **json** : Module pour le traitement des données au format JSON.
* **os** : Module pour interagir avec le système d'exploitation.
* **pygame.mixer** : Module pour jouer des sons.
* **sys** : Module pour accéder à certaines variables utilisées ou maintenues par l'interpréteur.

**Classe StartPage** :

* **Attributs** :
  + **controller** : Référence à la fenêtre principale pour naviguer entre les pages.
  + **background\_photo\_image** : Image d'arrière-plan pour cette page.
* **Méthodes** :
  + **load\_game\_state()** : Charge l'état du jeu depuis un fichier sauvegardé.
  + **delete\_save()** : Supprime la sauvegarde du jeu.

**Fonction resource\_path(relative\_path)** : Cette fonction renvoie le chemin d'accès de la ressource. Elle est conçue pour fonctionner à la fois en mode développement et lorsqu'une application est compilée (par exemple, avec PyInstaller).

**Variables globales** :

* **path\_to\_mp3** : Chemin d'accès au fichier MP3 pour la musique d'arrière-plan.
* **path\_to\_start\_page\_image** : Chemin d'accès à l'image pour la page de démarrage.

Dans cette page de démarrage (**StartPage**), on retrouve une image d'arrière-plan ainsi que trois boutons :

1. **Commencer le jeu** : Dirige l'utilisateur vers la page de sélection du joueur.
2. **CHARGER** : Tente de charger un jeu sauvegardé depuis un fichier **game\_save.json**.
3. **SUPPRIMER SAUVEGARDE** : Tente de supprimer la sauvegarde du jeu (fichier **game\_save.json**).

La musique d'arrière-plan est également jouée en boucle grâce au module **pygame**.

**3.3. PLAYER\_SELECTION\_PAGE.PY**

**Modules et classes importés** :

* **tkinter, ttk** : Modules utilisés pour créer des interfaces graphiques (GUI) avec Python.
* **Image, ImageTk** : Classes du module **PIL** (Pillow) pour le traitement d'images.
* **FOX** : Constante importée du module **game\_page**, probablement représentant un joueur renard.
* **pygame.mixer** : Module pour jouer des sons.
* **sys, os** : Modules pour interagir avec le système d'exploitation et accéder à certaines variables utilisées par l'interpréteur.

**Classe PlayerSelectionPage** :

* **Attributs** :
  + **controller** : Référence à la fenêtre principale pour naviguer entre les pages.
  + **background\_photo\_image** : Image d'arrière-plan pour cette page.
* **Méthodes** :
  + **select\_one\_player()** : Gère la sélection d'un seul joueur et navigue vers la page de sélection des animaux.
  + **select\_two\_players()** : Gère la sélection de deux joueurs et navigue vers la page de sélection des animaux.

**Fonction resource\_path(relative\_path)** : Cette fonction renvoie le chemin d'accès de la ressource. Elle est conçue pour fonctionner à la fois en mode développement et lorsqu'une application est compilée (par exemple, avec PyInstaller).

**Variables globales** :

* **path\_to\_mp3** : Chemin d'accès au fichier MP3 pour la musique d'arrière-plan.
* **path\_to\_game\_window\_image** : Chemin d'accès à l'image utilisée comme arrière-plan pour cette page.

Dans cette page de sélection des joueurs (**PlayerSelectionPage**), deux options sont présentées à l'utilisateur :

1. **1 joueur** : Dans ce mode, le joueur jouera contre l'IA (Intelligence Artificielle).
2. **2 joueurs** : Dans ce mode, deux joueurs humains joueront l'un contre l'autre.

L'utilisateur est invité à choisir l'un de ces modes en cliquant sur l'un des deux boutons. Une fois le choix effectué, l'utilisateur est dirigé vers la page de sélection des animaux (**AnimalSelectionPage**).

De plus, tout comme dans la page de démarrage, une musique d'arrière-plan est jouée en boucle.

**3.4. ANIMAL\_SELECTION\_PAGE.PY**

**Modules et classes importés** :

* **tkinter, ttk** : Modules utilisés pour créer des interfaces graphiques (GUI) avec Python.
* **Image, ImageTk** : Classes du module **PIL** (Pillow) pour le traitement d'images.
* **pygame.mixer** : Module pour jouer des sons.
* **sys, os** : Modules pour interagir avec le système d'exploitation et accéder à certaines variables utilisées par l'interpréteur.

**Classe AnimalSelectionPage** :

* **Attributs** :
  + **background\_photo\_image** : Image d'arrière-plan pour cette page.
  + **controller** : Référence à la fenêtre principale pour naviguer entre les pages et pour accéder aux variables du jeu.
  + **start\_game\_button** : Bouton qui démarre le jeu une fois que les animaux sont sélectionnés.
  + **goose\_count\_combobox** : Combobox pour sélectionner le nombre d'oies.
* **Méthodes** :
  + **update\_goose\_values()** : Met à jour les valeurs possibles pour la sélection des oies en fonction du nombre de renards choisi.
  + **check\_start\_game\_button()** : Vérifie si le bouton de démarrage du jeu doit être activé ou non.
  + **select\_foxes(event)** : Sélectionne le nombre de renards en fonction du choix de l'utilisateur.
  + **select\_geese(event)** : Sélectionne le nombre d'oies en fonction du choix de l'utilisateur.

**Fonction resource\_path(relative\_path)** : Cette fonction renvoie le chemin d'accès de la ressource. Elle est conçue pour fonctionner à la fois en mode développement et lorsqu'une application est compilée (par exemple, avec PyInstaller).

**Variables globales** :

* **path\_to\_mp3** : Chemin d'accès au fichier MP3 pour la musique d'arrière-plan.
* **path\_to\_game\_window\_image** : Chemin d'accès à l'image utilisée comme arrière-plan pour cette page.

Dans cette page de sélection des animaux (**AnimalSelectionPage**), les joueurs ont la possibilité de choisir le nombre de renards et d'oies qu'ils souhaitent utiliser pour la partie. Deux combobox sont fournis pour effectuer ces sélections. Une fois les sélections effectuées, le bouton "Commencer le jeu" devient actif, permettant aux joueurs de démarrer le jeu avec le nombre choisi d'animaux.

Lorsque l'utilisateur change la sélection du nombre de renards, les choix disponibles pour le nombre d'oies sont également mis à jour en conséquence.

**3.5 GAME\_PAGE.PY**

**Modules Importés** :

1. **tkinter**: C'est un module qui fournit des outils pour créer des interfaces graphiques.
2. **PIL (Python Imaging Library)**: Utilisé pour la manipulation d'images.
3. **json**: Permet de travailler avec des données au format JSON.
4. **pygame.mixer**: Utilisé pour la gestion sonore (ici pour la musique de fond).
5. **random**: Utilisé pour générer des nombres aléatoires.
6. **sys et os**: Fournissent diverses fonctions pour interagir avec le système d'exploitation, y compris la recherche de chemins de fichiers.

**Constantes** :

* **INVALID**: représente une zone non jouable du plateau.
* **FOX**: représente la position d'un renard sur le plateau.
* **GOOSE**: représente la position d'une oie sur le plateau.
* **EMPTY**: représente une case vide sur le plateau.

**GamePage** :

* **Attributs de Classe** : Ces tableaux définissent les configurations de départ du plateau selon le nombre de renards et d'oies.
* **Méthodes** :
  + **\_\_init\_\_**: Initialisation de la page du jeu.
  + **on\_canvas\_click**: Gère les clics sur le plateau.
  + **select\_piece** et **deselect\_all**: Sélectionne ou désélectionne une pièce.
  + **get\_possible\_moves**: Obtient tous les mouvements possibles pour une pièce.
  + **switch\_turn**: Change le joueur actuel.
  + **move\_piece**: Gère le déplacement des pièces.
  + **check\_victory**: Vérifie les conditions de victoire.
  + **draw\_board**: Dessine le plateau sur le canvas.
  + **update\_timer**: Met à jour le chronomètre du jeu.
  + **pause\_game**: Met en pause ou reprend le jeu.
  + **catch\_goose**: Met à jour le compteur d'oies attrapées.
  + **start\_game**: Lance une nouvelle partie.
  + **set\_game\_state**: Charge un état de jeu sauvegardé.
  + **save\_game\_state**: Sauvegarde l'état actuel du jeu.
  + **show\_save\_animation**: Affiche une animation lors de la sauvegarde.
  + **initialize\_game\_from\_data**: Initialise le jeu à partir de données sauvegardées.
  + **quit\_game**: Quitte le jeu.
  + **ai\_move**: Gère le mouvement de l'intelligence artificielle (si activée).

**Fonction resource\_path** : Cette fonction aide à trouver le chemin d'accès des ressources (comme des images ou des fichiers audio) que vous utilisez dans votre jeu. Elle est conçue pour fonctionner à la fois lorsque le script est exécuté normalement et lorsqu'il est compilé en une application avec des outils comme PyInstaller.

Sur la page GamePage, l'utilisateur peut réaliser plusieurs actions. Voici une liste détaillée des fonctionnalités accessibles à l'utilisateur :

* Jouer une partie : L'utilisateur peut démarrer une nouvelle partie de jeu.
* Clic sur le plateau : L'utilisateur peut cliquer sur une pièce du plateau (renard ou oie) pour la sélectionner, puis cliquer sur une autre case pour déplacer cette pièce, si le mouvement est valide.
* Mettre en pause / Reprendre : Il y a un bouton ou une fonctionnalité qui permet à l'utilisateur de mettre le jeu en pause ou de le reprendre.
* Visualiser le temps écoulé : Un chronomètre est disponible, affichant le temps écoulé depuis le début de la partie.
* Voir le nombre d'oies attrapées : L'utilisateur peut voir combien d'oies ont été attrapées par les renards.
* Sauvegarder la partie : L'utilisateur a la possibilité de sauvegarder l'état actuel du jeu pour y revenir plus tard. Une petite animation s'affiche lorsqu'une sauvegarde est effectuée.
* Charger une partie précédente : L'utilisateur peut charger un état de jeu précédemment sauvegardé pour continuer une partie.
* Quitter le jeu : L'utilisateur peut quitter le jeu à tout moment.
* Jouer contre l'IA : Si l'intelligence artificielle est activée, l'utilisateur peut jouer contre elle.

**4.Présentation des algorithmes de gestion de la grille**

**4.1 INITIALISATION DE LA GRILLE**

La grille est une représentation du plateau de jeu. Elle est constituée de différentes configurations en fonction du nombre de renards et d'oies. Chaque configuration est une liste de listes représentant les lignes du plateau.

* **INVALID** : Zone non jouable du plateau.
* **FOX** : Position d'un renard sur le plateau.
* **GOOSE** : Position d'une oie sur le plateau.
* **EMPTY** : Case vide sur le plateau.

Plusieurs configurations possibles pour le plateau sont définies au début de la classe **GamePage**, À savoir **BOARD\_1\_13**, **BOARD\_1\_15**, **BOARD\_1\_17,** **BOARD\_2\_20**, **BOARD\_2\_27**.

A screenshot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

**4.2 MISE A JOUR DE LA GRILLE APRES CHAQUE MOUVEMENT**

La mise à jour du plateau est effectuée grâce à plusieurs méthodes :

* **on\_canvas\_click**: Cette méthode est appelée lorsque l'utilisateur clique sur le plateau. Elle détermine si le clic a eu lieu sur une case valide, si une pièce a été sélectionnée ou si un mouvement est possible.
* **select\_piece**: Sélectionne une pièce sur le plateau.
* **deselect\_all**: Désélectionne toutes les pièces sélectionnées.
* **get\_possible\_moves**: Retourne une liste de mouvements possibles pour une pièce donnée.
* **move\_piece**: Déplace une pièce d'une case à une autre. Elle met également à jour le plateau en conséquence.

**4.3 VERIFICATION DE LA VALIDITE DES MOUVEMENTS SUR LA GRILLE**

La validation des mouvements est principalement gérée par la méthode **get\_possible\_moves**. Cette méthode vérifie :

* Si le déplacement est dans les limites du plateau.
* Pour le renard : Si une oie est à côté et si la case derrière cette oie est vide (permettant au renard de sauter par-dessus l'oie et de la capturer).
* Pour l'oie : Si le mouvement est valide (l'oie peut seulement se déplacer vers l'avant et horizontalement).

De plus, la méthode **check\_victory** est utilisée pour vérifier si l'une des conditions de victoire a été atteinte (par exemple, toutes les oies ont été capturées).

**5.Présentation des algorithmes implémentant les déplacements et les captures**

**5.1. ALGORITHMES DE DEPLACEMENT POUR LE RENARD**

Le déplacement du renard est géré par la méthode **get\_possible\_moves**:

A computer screen with text

Description automatically generated

Cette méthode détermine les mouvements possibles pour un renard à partir d'une position donnée **(i, j)** sur le plateau. Le renard peut se déplacer dans n'importe quelle direction, à condition qu'il ne sorte pas du plateau et que la case cible soit vide. De plus, s'il y a une oie adjacente, le renard peut la capturer en sautant par-dessus elle dans la direction correspondante, à condition que la case derrière l'oie soit également vide.

**5.2. ALGORITHMES DE DEPLACEMENT POUR LES OIES**

Toujours dans la méthode **get\_possible\_moves**, le déplacement des oies est géré comme suit:

A computer screen shot of text

Description automatically generated

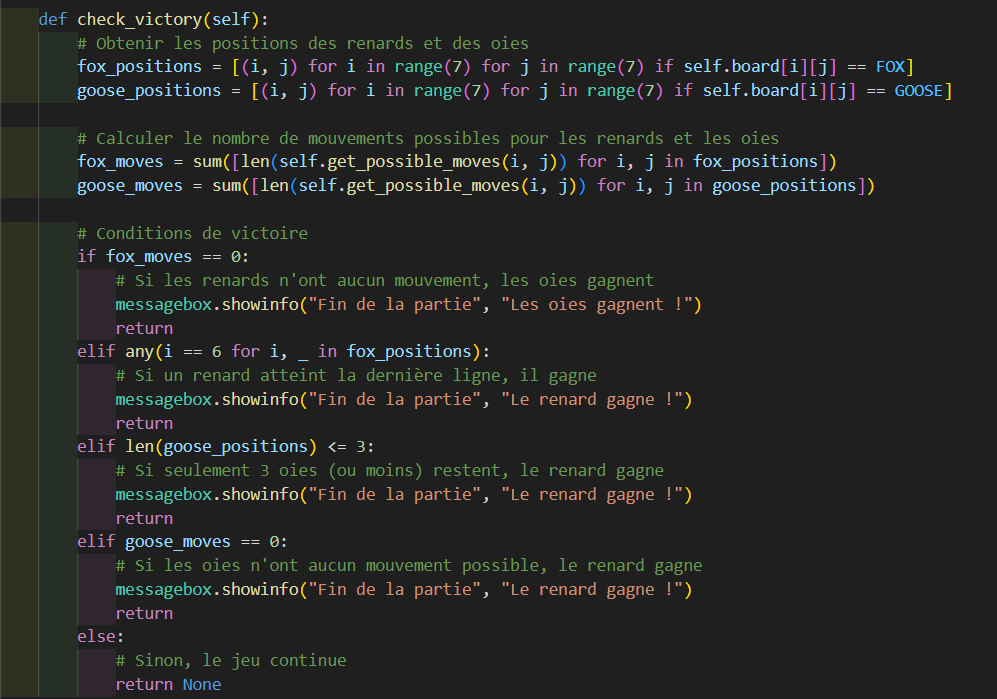
Les oies peuvent se déplacer vers le haut et horizontalement, mais pas vers le bas. Cette limitation est représentée par la liste **forward\_and\_horizontal\_moves**.

**5.3. GESTION DES CAPTURES**

La capture est également gérée dans **get\_possible\_moves** où, si une oie est adjacente à un renard et qu'il y a un espace vide de l'autre côté de l'oie, le renard peut sauter par-dessus et capturer cette oie.

**5.4. CONDITIONS DE VICTOIRE ET DE DEFAITE**

Les conditions de victoire et de défaite sont gérées dans la méthode **check\_victory**:



* Il existe plusieurs conditions de victoire:
* Si le renard n'a pas de mouvements disponibles, les oies gagnent.
* Si un renard atteint la dernière ligne du plateau, le renard gagne.
* Si seules 3 oies (ou moins) restent, le renard gagne.
* Si les oies n'ont pas de mouvements disponibles, le renard gagne.

**6. CONCLUSION**

**6.1. RETOUR SUR LES DEFIS RENCONTRES LORS DU DEVELOPPEMENT**

Développer un jeu comme le Renard et les Oies nécessite une combinaison de compétences en matière de programmation, de logique et de design. Certains des défis qui peuvent être rencontrés lors du développement incluent :

Logique du jeu : Étant donné les règles uniques du jeu, l'implémentation de la logique de déplacement des pièces (renard et oies) peut s'avérer complexe.

Interface utilisateur : Créer une interface utilisateur intuitive et réactive pour le jeu nécessite une réflexion approfondie sur la conception et l'ergonomie.

Gestion des erreurs : Assurer que le jeu gère correctement toutes les erreurs possibles pour éviter les plantages ou les comportements inattendus.

Optimisation : Le jeu doit être optimisé pour garantir des temps de réponse rapides et une expérience utilisateur fluide.

**6.2. PISTES D'AMELIORATIONS ET EVOLUTIONS FUTURES**

Mode Multijoueur en ligne : Ajouter un mode multijoueur en ligne pour permettre aux joueurs de s'affronter à distance.

Intelligence Artificielle améliorée : Développer une IA plus avancée pour le renard ou les oies, pour offrir différents niveaux de difficulté.

Graphismes et sons : Améliorer les graphismes et ajouter des effets sonores pour enrichir l'expérience utilisateur.

Tutoriel et niveaux : Intégrer un tutoriel pour les nouveaux joueurs et proposer différents niveaux de difficulté ou scénarios pour varier le gameplay.

Statistiques et scores : Ajouter un système de scoring et de statistiques pour suivre les progrès et les performances des joueurs.

**7. ANNEXES**

**7.1. CODE SOURCE COMPLET**

(VOIR FICHIER ZIP)

**7.2. BIBLIOGRAPHIE ET RESSOURCES**

"Introduction à la programmation avec Python", par Charles Severance.

"Python GUI Programming with Tkinter", par Alan D. Moore.

Documentation officielle de Python : https://docs.python.org/3/

Tkinter documentation : https://docs.python.org/3/library/tkinter.html

Forums et discussions en ligne sur Stack Overflow et autres sites similaires, pour des questions spécifiques liées à la programmation du jeu.