Project TicTacToe

Introduction

Pour mettre en pratique les notions de programmation vues au cours de ce semestre, je vous propose de réaliser un petit jeu dans le terminal : le **TicTacToe** (ou **Morpion** en français).

- Le projet est à réaliser individuellement

Objectifs du projet

- Évaluer votre niveau en programmation en C++
- Réaliser un projet de A à Z, organiser vos idées, structurer vos fichiers et votre code
- Vous familiariser avec l'utilisation de Git et GitHub

Fonctionnalités demandées (Cahier des charges)

Structure de joueur

Vous devez réaliser une **structure** `Player` qui contient les informations suivantes :

- `name` : nom du joueur
- `symbol` : symbole du joueur (e.g. `'X'` ou `'O'`)

Cette structure doit être utilisée pour stocker les informations des joueurs et placée dans des fichiers dédiés `Player.hpp` et `Player.cpp`.

Vous devez réaliser une fonction `create_player` qui ne prend aucun paramètre et qui retourne une structure `Player` initialisée avec les informations du joueur. Cette fonction doit demander le nom du joueur à l'utilisateur (`std::cin`) et lui demander de choisir un symbole (X ou O généralement mais vous pouvez choisir d'autres symboles si vous le

souhaitez).

Affichage du plateau de jeu

Vous devez réaliser une fonction `draw_game_board` (vous pouvez modifier légèrement le nom de la fonction si vous le souhaitez) qui prend en paramètre un tableau de caractères (`char`) et affiche le plateau de jeu dans la terminal.

Le plateau de jeu doit être affiché de la manière suivante :

```
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
```

où chaque chiffre représente une case du plateau de jeu.

Le plateau doit être représenté dans le code sous forme d'un tableau de **caractères** (`char`) représentant les **symboles** des joueurs ou un caractère "vide" (par exemple un `.`) pour une case vide.

⚠️ Ici le nombre de cases du plateau est connu à l'avance, vous devez donc de préférence utiliser un tableau **statique** plutôt qu'un tableau dynamique pour représenter le plateau de jeu.

Mode de jeu

Deux joueurs

Vous devez réaliser un mode de jeu où deux joueurs peuvent s'affronter sur le jeu.

Le jeu doit **alterner** entre les deux joueurs pour leur permettre de jouer un coup à tour de rôle.

Un joueur et IA

Vous devez réaliser un mode de jeu où un joueur affronte une IA (Intelligence Artificielle) qui joue **aléatoirement sur une case vide**.

Dans ce cas de figure vous réutiliserez la structure `Player` pour l'IA en initialisant le nom de l'IA à "IA" et un symbole pré-défini (différent de celui du joueur).

Le jeu doit alterner entre le joueur et l'IA pour leur permettre de jouer un coup à tour de rôle.

Il est possible d'améliorer l'IA mais ce n'est pas requis (voir section "Améliorations possibles").

Menu de démarrage

Vous devez réaliser un menu de démarrage qui permet à l'utilisateur de choisir le mode de jeu (deux joueurs ou un joueur contre l'IA).

Exemple:

```
Bienvenue dans le jeu du TicTacToe
Veuillez choisir un mode de jeu :
1. Deux joueurs
2. Un joueur contre l'IA
```

Fin de partie

Le jeu doit vérifier si un joueur a aligné 3 symboles sur une ligne, une colonne ou une diagonale. Si c'est le cas, le jeu doit afficher le nom du joueur gagnant et terminer la partie.

Si aucun joueur n'a aligné 3 symboles et que le plateau de jeu est complet, le jeu doit afficher un message de fin de partie indiquant que c'est un match nul.

> **ASTUCE**
>
> Idéalement, vous devez écrire une fonction de vérification de victoire qui ne dépend pas des symboles, pour pouvoir réutiliser cette fonction peu importe le symbole des joueurs.

Améliorations

Vous **devez** également réaliser **au moins une** des améliorations listées ci-dessous (ou alors me proposer une autre amélioration que je dois valider).

⚠️ Attention, pour toute amélioration, vous devez réaliser le jeu de base avant de commencer à ajouter des fonctionnalités supplémentaires.

Améliorer l'IA

Il existe plusieurs façons de réaliser une IA pour le jeu du TicTacToe, vous pouvez choisir de réaliser (par ordre de difficulté croissante) :

- Une IA capable de bloquer les coups de l'adversaire: l'ordinateur joue un coup aléatoire sur une case vide, mais si l'adversaire a 2 symboles alignés, l'ordinateur joue sur la case restante pour bloquer l'adversaire
- Une IA basée sur un algorithme de recherche: l'ordinateur envisage beaucoup (ou toutes) les possibilités, et joue le meilleur coup trouvé (cf. les algos minimax, alpha-beta pruning, etc.)

Ajouter une interface graphique

Vous pouvez réaliser une interface graphique pour le jeu du TicTacToe en utilisant une bibliothèque graphique comme SFML, SDL, raylib, etc.

⚠️ Attention, l'interface graphique n'est pas requise pour ce projet, Cela peut être intéressant mais vous êtes responsables de la gestion de l'interface graphique et du temps que cela vous prendra. Vous devez réaliser le jeu en terminal avant de commencer à ajouter une interface graphique. Concernant la compatibilité, si vous êtes sur MacOS, venez m'en parler avant de vous lancer dans cette amélioration que je sois bien en mesure de corriger votre projet.

Évolution vers un jeu plus complexe

Vous avez la possibilité de réaliser un jeu similaire au TicTacToe mais avec des règles plus complexes (plateau de jeu plus grand, alignement de symboles plus grand, etc.)

Si vous partez sur un jeu plus complexe, vous devez réaliser le nouveau jeu dans des fichiers séparés du jeu de base pour ne pas impacter la réalisation du jeu de base.

Dans ce cas, une IA n'est pas requise et vous pouvez réaliser seulement un mode deux joueurs.

Quelques exemples de jeux plus complexes :

- Gomoku (aligner 5 symboles)
- Puissance 4 (aligner 4 symboles)

- Ultimate tic-tac-toe (Tic-tac-toe imbriqué sur 2 niveaux (plateau de jeu de 9x9 cases))

Librairies

Si vous voulez utiliser des librairies tierces, demandez-moi avant pour que je puisse valider votre choix.

Manipuler le terminal

Ce n'est pas requis mais si vous voulez pouvoir effacer entièrement le terminal (entre chaque tour de jeu par exemple) ou déplacer le curseur dans le terminal, vous pouvez utiliser cette librairie que j'ai réalisée pour vous : Terminal Ctrl.

Voilà comment l'ajouter au projet via CMake:

```
include(FetchContent)
FetchContent_Declare(
    terminal_ctrl
    GIT_REPOSITORY https://github.com/dsmtE/terminal_ctrl
    GIT_TAG "origin/main"
)

FetchContent_MakeAvailable(terminal_ctrl)
target_link_libraries(${PROJECT_NAME} PUBLIC terminal_ctrl)
```

Voilà un exemple d'utilisation de la librairie:

```
#include <terminal_ctrl/terminal_ctrl.h>

int main() {
    terminal_ctrl::clear_terminal();
    terminal_ctrl::move_cursor(1, 3);
    std::cout << "Hello World" << std::endl;
    return 0;
}
```

Rendu

Le projet est à rendre sur **GitHub**(ou **Gitlab**), vous devez m'envoyer le lien de votre dépôt avant la date limite de rendu: le **Dimanche 21 Décembre 23h59**.

⚠️ Seul les commits avant la date limite seront pris en compte pour la notation.

Un document vous sera partagé pour que vous puissiez noter le lien de votre dépôt.

(Évidemment je dois avoir accès à votre dépôt pour pouvoir le consulter, faites bien attention à ce qu'il soit en public (ou, si vous tenez à le garder privé, ajoutez moi en collaborateur (dsmtE))).

Vous devez ajouter un fichier `README.md` à la racine de votre dépôt et qui fera office de **rapport** et contiendra :

- Des indications pour exécuter votre programme si nécessaire (librairies, système d'exploitation testé).
- Des explications sur l'organisation de votre code (fichiers, structure, fonctions, etc.) et vos choix d'implémentation.
- Si améliorations il y a, une description des fonctionnalités supplémentaires que vous avez réalisées (et comment les utiliser si besoin).
- Un rapide bilan sur les problèmes rencontrés et solutions trouvées.

⚠️ Attention: Un commit unique la veille de la date limite ne sera pas accepté, vous devez montrer que vous avez travaillé régulièrement sur le projet.

Barème

Le jeu de base comptera pour **16** points, le reste des points sera attribué sur les améliorations réalisées ainsi que le **Rapport** (Fichier `README.md`).