

8805

WINNERS  
PLAY HERE



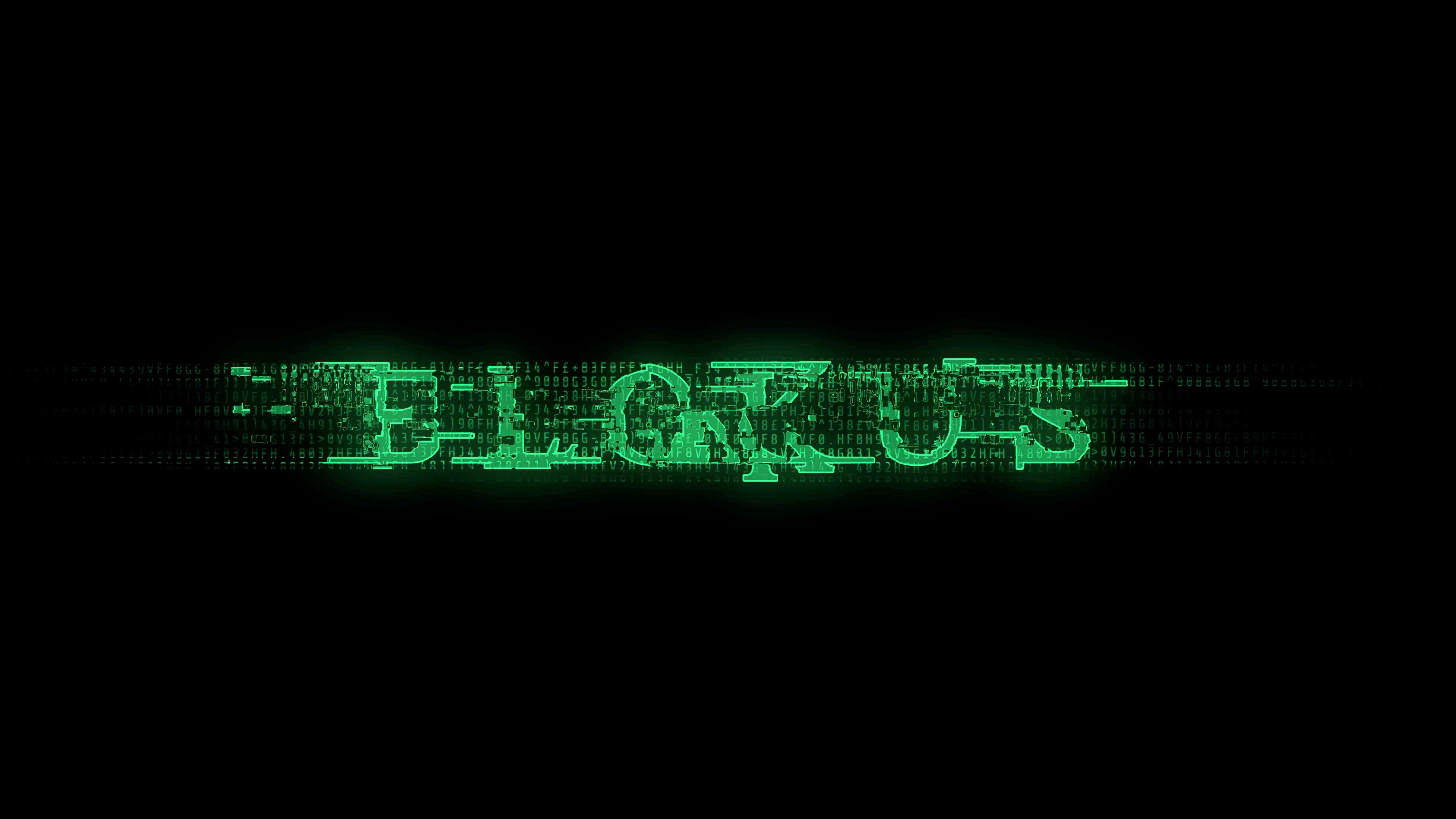
NUMBER OF SLOTS AND TABLES

Do not hold doors

Do not hold doors

Do not hold doors

Do not hold doors

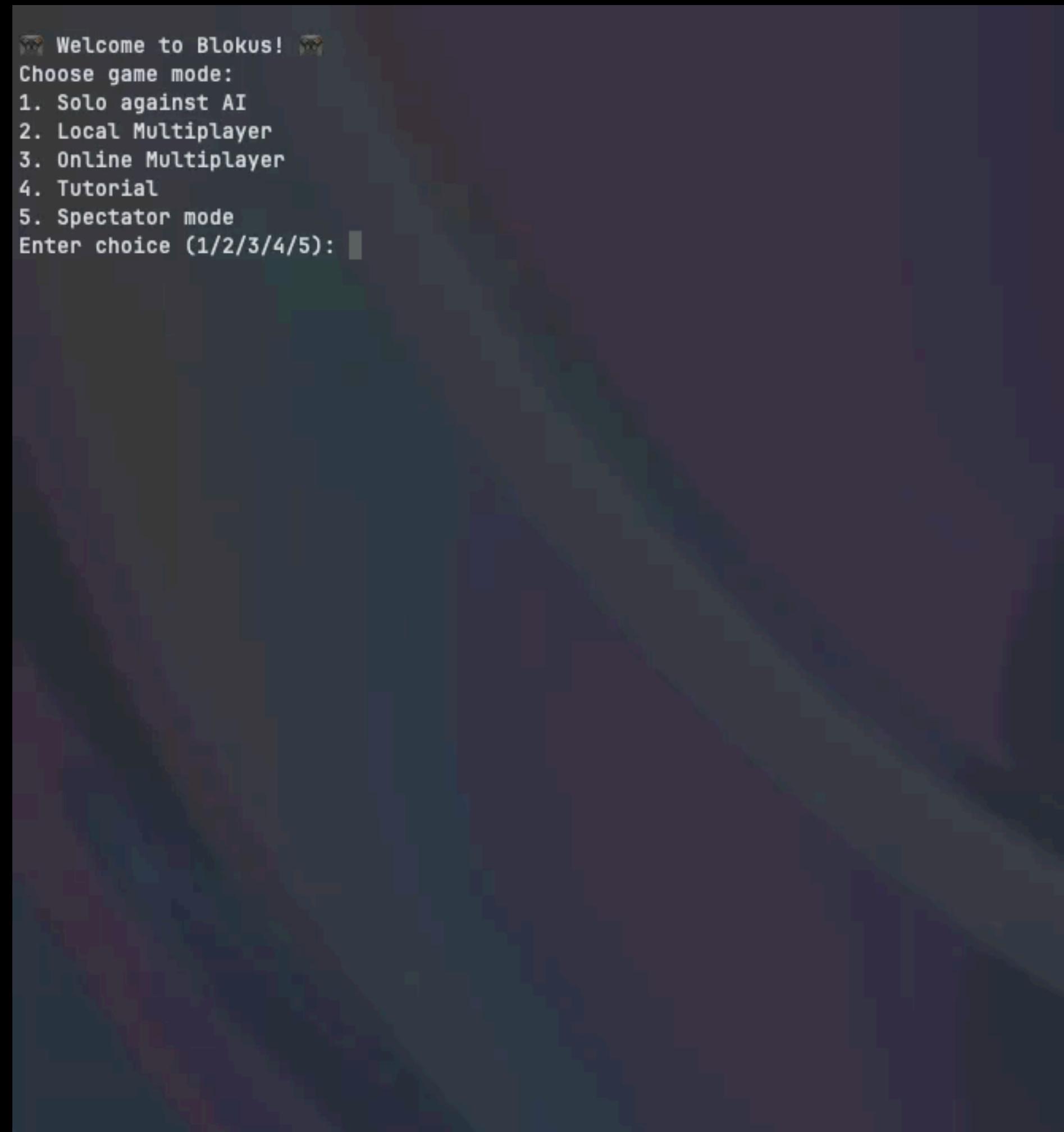


# PLAN

- I. Fonctionnalités développées
  - I. Fonctionnalités de base
  - II. Fonctionnalités additionnelles
- II. Couverture des tests
- III. Organisation du groupe
- IV. Apports du projet

# Fonctionnalités de base.

1. Mise en place du jeu
2. Gestion des joueurs
3. Validation des coups
4. Gestion des tours
5. Détection de fin de partie



# Fonctionnalités add.

1. IA
2. Multijoueur
  1. Local
  2. En ligne
3. Sauvegarde

=====

# Welcome to Blokus!

=====

Choose game mode:

1. Solo against AI
2. Local Multiplayer
3. Online Multiplayer
4. Tutorial
5. Spectator mode

Enter choice (1/2/3/4/5): █

# Easy

- Basé sur l' aléatoire
- Évaluation de tous les mouvements possibles
- Sélection d'un mouvement valide

```
# Fonction principale pour jouer un mouvement aléatoire
def _play_random_move(board):
    pieces = obtenir_pieces_restantes()
    if not pieces:
        return False

    mouvements_possibles = []

    # Boucle pour chaque pièce restante
    for piece in pieces:
        for variant in obtenir_varianter(piece):
            positions_valides = trouver_positions_valides(variant, couleur)
            for x, y in positions_valides:
                mouvements_possibles.append((piece, variant, x, y))

    # Choisir un mouvement aléatoire parmi les mouvements possibles
    if mouvements_possibles:
        piece_choisie, variante_choisie, x, y = choisir_au_hasard(mouvements_possibles)
        print(f"Placer pièce à ({x}, {y})")
        faire_mouvement(board, variante_choisie, (x, y))
        return True

    print("Aucun mouvement disponible")
    return False

# Fonction pour effectuer un mouvement
def faire_mouvement(board, piece, position):
    x, y = position
    placer_piece_sur_plateau(board, piece, x, y, couleur)
    retirer_piece_des_restantes(piece)

# Fonction pour obtenir toutes les variantes d'une pièce
def obtenir_varianter(piece):
    variantes = []
    for _ in range(4):
        piece = tourner(piece)
        variantes.append(piece)
        variantes.append(inverser(piece))
    return variantes
```

# Medium

- Trier les pièces par taille
- Évaluer tous les mouvements possibles
- Choisir et effectuer le meilleur mouvement
- Evaluation heuristique

```
# Fonction principale pour jouer un mouvement de niveau moyen
def _play_medium_move(board):
    pieces = trier_par_taille_descendante(obtenir_pieces_restantes())
    if not pieces:
        return False

    meilleur_mouvement = None
    meilleur_score = -inf

    # Calculer le centre du plateau
    centre_x = taille_plateau // 2
    centre_y = taille_plateau // 2

    # Boucle pour chaque pièce restante
    for piece in pieces:
        for variant in obtenir_varianter(piece):
            positions_valides = trouver_positions_valides(variant, couleur)
            for x, y in positions_valides:
                score = évaluer_mouvement(board, variant, x, y, centre_x, centre_y)
                if score > meilleur_score:
                    meilleur_score = score
                    meilleur_mouvement = (variant, x, y)

    # Choisir le meilleur mouvement parmi les mouvements possibles
    if meilleur_mouvement:
        piece_choisie, x, y = meilleur_mouvement
        faire_mouvement(board, piece_choisie, (x, y))
        return True

    return False
```

# Hard

- Contrôle de territoire avancé
- Blocage de l'adversaire
- Facteurs modifiables afin d'ajuster la difficulté

```
# Fonction principale pour jouer un mouvement de niveau difficile
def _play_hard_move(board):
    pieces = trier_par_taille_descendante(obtenir_pieces_restantes())
    if not pieces:
        return False

    meilleur_mouvement = None
    meilleur_score = -inf

    # Calculer le centre du plateau
    centre_x = taille_plateau // 2
    centre_y = taille_plateau // 2

    # Boucle pour chaque pièce restante
    for piece in pieces:
        for variant in obtenir_variantes(piece):
            positions_valides = trouver_positions_valides(variant, couleur)
            for x, y in positions_valides:
                score = évaluer_mouvement(board, variant, x, y, centre_x, centre_y)
                if score > meilleur_score:
                    meilleur_score = score
                    meilleur_mouvement = (variant, x, y)

    # Choisir le meilleur mouvement parmi les mouvements possibles
    if meilleur_mouvement:
        piece_choisie, x, y = meilleur_mouvement
        faire_mouvement(board, piece_choisie, (x, y))
        return True

    return False

# Fonction pour évaluer un mouvement
def évaluer_mouvement(board, piece, x, y, centre_x, centre_y):
    score = 0

    # Facteur 1 : Taille de la pièce
    taille_piece = calculer_taille(piece)
    score += taille_piece * POIDS_TAILLE

    # Facteur 2 : Distance au centre
    distance_centre = calculer_distance(x, y, centre_x, centre_y)
    score += distance_centre * POIDS_CENTRE

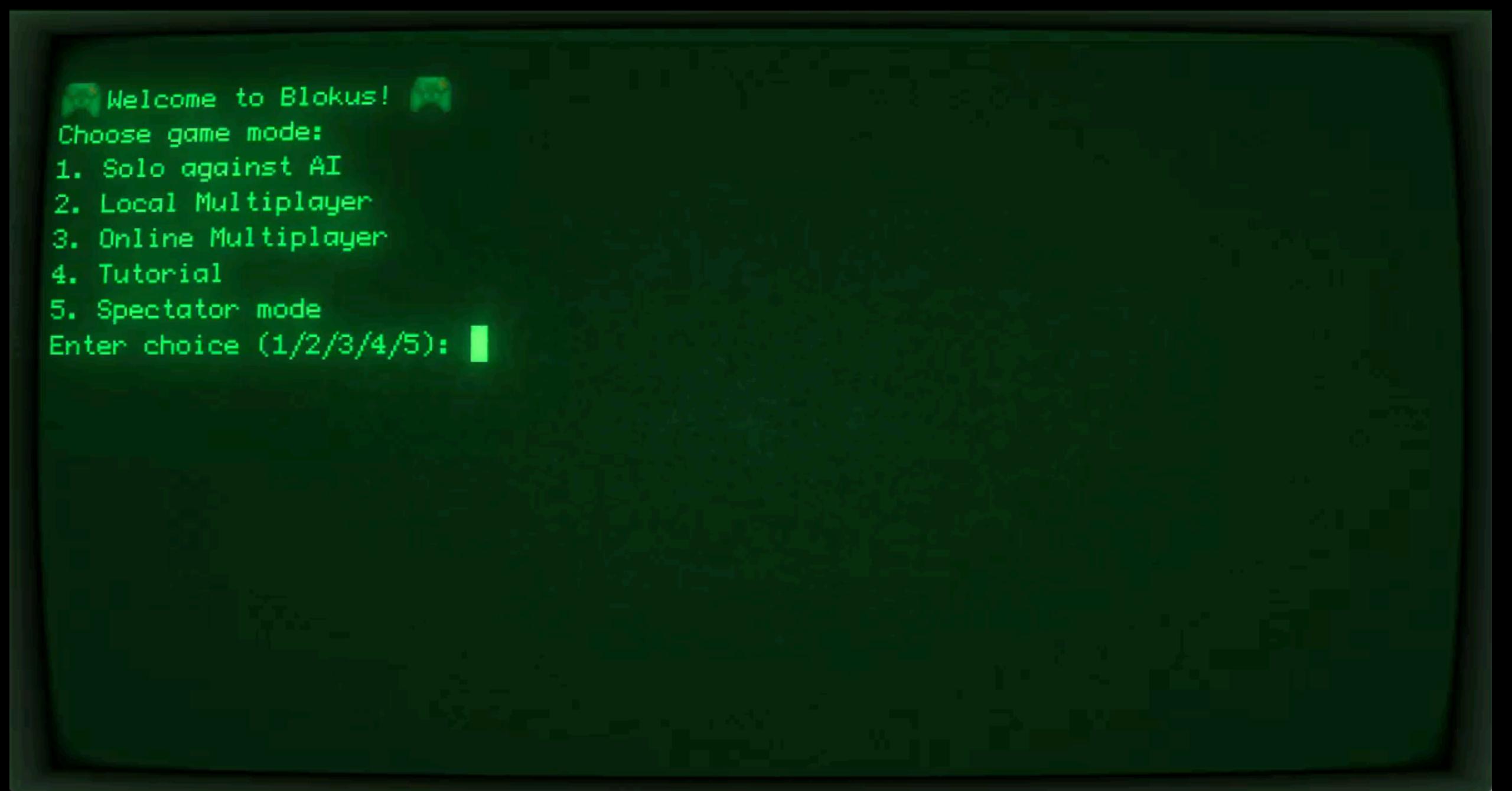
    # Facteur 3 : Contrôle du territoire
    territoire = calculer_territoire(board, piece, x, y)
    score += territoire * POIDS_TERRITOIRE

    return score
```



# Mulijoueur local

- possibilité de choisir entre 2 - 4 joueurs
- Placement des pieces comme dans les autres modes de jeu



# Multijoueur en ligne

- Un serveur distant lance simplement le fichier **server.sh**
- Possibilité d'héberger plusieurs parties en simultané
- Basé sur socket

```
> ls
-- [] application.sh
-I [] blokus_autosave.json
-I [] blokus_save.json
-- [] docs
-- [] LICENSE
-- [] README.md
-- [] server.sh
-- [] src
-- [] test
-- [] test.sh
C ~/Documents/project-s1 on main
at 12:52:44
```

```
C ~/Documents/project-s1 on main
at 12:54:04
```



# Tutoriel interactif

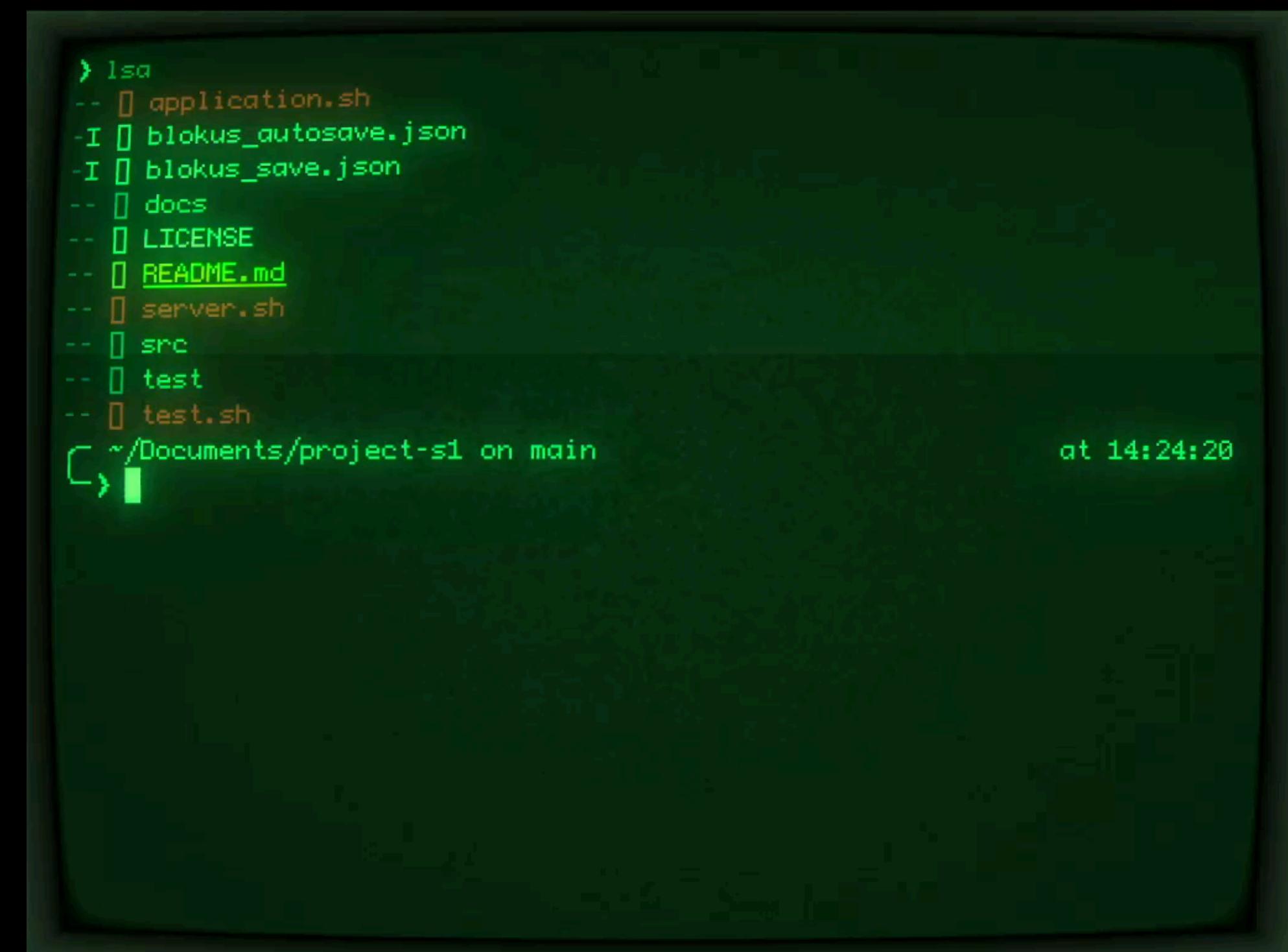
- Un tutoriel textuel avec des choix
- Plus simple à adapter à chaque langue



```
~/Documents/project-s1 on main
at 13:39:15
```

# Couvertures des tests

- 62 tests unitaires ont été implémentés avec **unittest**
- 12 tests de validation



```
❯ ls
-- [] application.sh
-I [] blokus_autosave.json
-I [] blokus_save.json
-- [] docs
-- [] LICENSE
-- [] README.md
-- [] server.sh
-- [] src
-- [] test
-- [] test.sh
~/Documents/project-s1 on main
at 14:24:20
```

# Organisation du groupe

- Nous nous sommes repartis le projet à 3
- Communication organisée au sein d'un serveur Discord
- Issues assignées à chacun organisé avec des labels
- Utilisation des IDE : NeoVim et VSCode



# Apports personnels

Henrique FRANCO DE OLIVEIRA

- Développement de l'aspect multijoueur
- Développement des base du jeu (collaboration)
- Correction des bugs

# Apports personnels

Gabriel FREISS

- Creation du mode spectateur
- Developpement collaboratif bot medium et hard
- Creation du tutoriel textuel

# Apports personnels

Anselme GARNIER

- développement de la base du jeu (prototype initial)
- Creation de l'IHM
- Creation bot medium et hard
- Mise en oeuvre tests unitaires
- Correction de bugs

# Apports du projet

1. Compétences
  1. Techniques
  2. Organisationnelles

└ ~Documents/project-s1 on main

at 15:17:33

# **DEMO**