Mario Bros (SFML, C++, Makefile)

Description du Projet

Ce projet est une version personnalisée de Mario Bros développée en C++ avec l'utilisation de la bibliothèque SFML . Il inclut un menu de démarrage avec plusieurs modes de jeu :

- 2 vs 2 : Mario contre Luigi (chaque personnage est contrôlé par un joueur).
- Joueur vs IA : Mario est contrôlé par le joueur tandis que Luigi est contrôlé par une IA.

Dépendances

- SFML : Gestion des graphiques, audio et événements.
- C++: Langage de programmation principal.
- Makefile : Pour automatiser la compilation.

Installation

1. Cloner le dépôt

Pour récupérer le code source, ouvrez un terminal et entrez la commande suivante :

```
git clone https://github.com/votre-utilisateur/mario-bros.git
cd mario-bros
```

2. Installer les dépendances

Assurez-vous d'avoir installé SFML, g++ et Make.

Installation sur Linux (Ubuntu/Debian):

```
sudo apt update
sudo apt install libsfml-dev build-essential
```

3. Compilation et Lancement

Pour compiler le projet, utilisez le Makefile inclus :

```
cd Mario_Bros/src
```

make

Puis lancez le jeu :

./mario

Si vous voulez nettoyer les fichiers générés :

make clean

Comment Jouer

Menu Principal

Lorsque le jeu démarre, vous accédez à un menu avec plusieurs options :

Jouer en 2 vs 2:

- Mario et Luigi sont contrôlés par deux joueurs.
- Les contrôles sont les suivants :
 - Mario : Flèches directionnelles + touches de saut/tir.
 - Luigi: Touches WASD + touches de saut/tir.

Joueur vs IA:

- Mario est contrôlé par le joueur.
- Luigi est contrôlé par une IA programmée pour réagir intelligemment à l'environnement du jeu.

But du Jeu

Le but du jeu est de collecter le plus de pièces et d'atteindre le drapeau en premier tout en surmontant les obstacles!

Objectifs principaux:

- Collecter le plus de pièces pour augmenter votre score.
- Éviter les obstacles (trous, blocks, pipe) pour ne pas perdre et continuer à avancer.
- Échapper aux ennemis ou les éliminer si possible en sautant dessus.
- Récupérer des pouvoirs (champignons, étoiles, fleur) pour gagner des avantages temporaires (devenir plus grand, super vitesse, lancer des boules de feu).
- Atteindre le drapeau en premier : Le premier joueur à toucher le drapeau remporte la partie!

Le gagnant est celui qui atteint le drapeau en premier ou qui a le meilleur score en cas d'égalité!

Gestion des Vies et du Score

- Le joueur commence avec 3 vies .
- Le score augmente en :
 - Collectant des pièces
 - Tuant des ennemis
- À chaque 100 points, le joueur gagne une vie supplémentaire.
- Si le joueur meurt mais a encore une vie disponible, il **revient à l'endroit où il est mort** sans recommencer le niveau depuis le début.
- S'il n'a plus de vies, c'est Game Over!

Comment jouer?

Contrôles

(Mode Joueur vs Joueur)

Mario (Joueur 1)	Luigi (Joueur 2)
Flèche \rightarrow	D
$Flèche \leftarrow$	\mathbf{Q}
Flèche ↑	${f Z}$
Espace	${f F}$
	Flèche \rightarrow Flèche \leftarrow Flèche \uparrow

(Mode Joueur vs IA)

• Mario est contrôlé par le joueur :

Avancer \rightarrow Flèche \rightarrow

 $Reculer \rightarrow Flèche \leftarrow$

Sauter → Flèche ↑

Tirer (Boule de feu) \rightarrow Espace

• Luigi est contrôlé par une IA qui se déplace automatiquement.

Description des classes principales

Classe Player

La classe Player représente les joueurs dans le jeu Mario Bros. Elle permet de gérer toutes les fonctionnalités liées au personnage :

- Déplacement (droite, gauche, saut)
- Gestion des collisions (avec les blocs, les tuyaux, les trous)
- Animation du personnage (en fonction de la direction et de l'état)
- Gestion de la vie, des pièces et des pouvoirs
- Lancer de boules de feu (si le joueur a le pouvoir de feu)
- Gestion de l'état (grand, petit, étoile, mort)
- Détection de la victoire (atteinte du drapeau)

Diagramme UML de la Classe Player

```
(C) Player
□ sf::Texture normalTexture
□ sf::Texture fireTexture
sf::Sprite sprite
std::string characterName
□ Mouvement mouvement
sf::Keyboard::Key rightKey
sf::Keyboard::Key leftKey
sf::Keyboard::Key jumpKey
🗖 bool big
■ bool isDead

    bool movingRight
    bool movingLeft

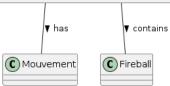
bool hasFirePower
□ int lives
int coins
🛮 int fireballCooldown
std::vector<Fireball> fireballs
□ int currentFrame
int frameCounter
std::string characterType
int hitTimer
bool isStarPowered
□ int starPowerFrames
□ float baseSpeed
float currentSpeed
□ int score
□ float initial×
□ float initialY
□ float deathX
□ float deathY
int lastLifeThreshold
• Player(texturePath: string, name: string, x: float, y: float, speed: float, right: Key, left: Key, jump: Key)
update(blocks: vector<sf::Sprite>&, pipes: vector<sf::Sprite>&, questionBlocks: vector<sf::Sprite>&): void
applyGravity(blocks: vector<sf::Sprite>&, pipes: vector<sf::Sprite>&, questionBlocks: vector<sf::Sprite>&): void
draw(window: sf::RenderWindow&): void
animate(): void
moveRight(): void
moveLeft(): void
isMovingRight(): bool
isMovingLeft(): bool
getPosition(): sf::Vector2f
setPosition(x: float, y: float): void
getbounds(): sf::FloatRect
getCoins(): void
loseLife(): void
gainLife(): void
getLives(): int
o grow(): void
shrink(): void
isBig(): bool
die(): void
loadTexture(): bool
checkForGaps(gaps: vector<sf::FloatRect>&): void

    getSprite(): sf::Šprite&

• checkWin(flag: sf::Sprite&): bool
collectFireFlower(): void
shootFireball(): void
updateFireballs(blocks: vector<sf::Sprite>&): void
drawFireballs(window: sf::RenderWindow&): void
getFireballs(): vector<Fireball>&

    hasFirePowerActive(): bool

loseFirePower(): void
bounce(): void
getScore(): int
getName(): string
setMovingRight(move: bool): void
setMovingLeft(move: bool): void
isInAir(): bool
getMouvement(): Mouvement&
o isSmall(): bool
isHit(): bool
o isInvincible(): bool
getisDead(): bool
setIsDead(dead: bool): void
o collectEtoile(): void
setSpeed(newSpeed: float): void
getSpeed(): float
• getBaseSpeed(): float
• respawn(): void
• updateInitialPosition(x: float, y: float): void
shouldRespawn(): booladdScore(points: int): void
```



Classe Enemy

- Classe de base pour tous les ennemis
- Implémente le comportement de base (mouvement, collision)
- Gère l'état de vie et la logique de défaite
- Inclut des méthodes virtuelles pour les comportements spécifiques

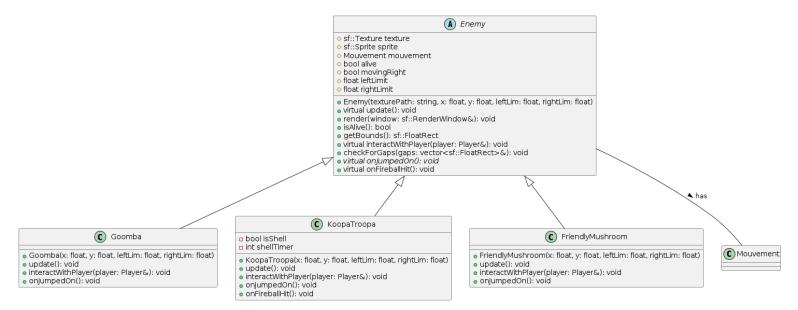


Figure 2: Diagramme UML

Classe FriendlyMushroom

- Hérite de Enemy mais agit comme un power-up
- Fait grandir le joueur lors de la collecte
- Se déplace horizontalement et change de direction aux obstacles

Classe FireFlower

- Power-up qui donne au joueur la capacité de lancer des boules de feu
- Apparaît dans certains blocs spéciaux
- Reste statique et attend d'être collectée

Classe Fireball

- Projectile lancé par le joueur avec le power-up feu
- Se déplace dans la direction du lancement
- Rebondit sur les surfaces et détruit les ennemis au contact

Classe Coin

- Objet collectible qui augmente le score du joueur
- Apparaît dans les niveaux ou sort des blocs frappés

Classe Camera

- Gère la vue du jeu et suit le joueur
- Implémente des limites pour ne pas montrer hors du niveau
- Adapte la vue à la taille de la fenêtre

Classe Background

- Charge et affiche l'environnement du jeu
- Gère les tuiles de sol, les tuyaux, les blocs et le ciel
- Fournit des méthodes pour accéder aux éléments du niveau

Classe AIPlayer

- Contrôle un personnage secondaire (Luigi) par IA
- Prend des décisions basées sur l'environnement et les ennemis
- Implémente différentes stratégies (éviter les ennemis, sauter les obstacles)

Logique du jeu

1. Boucle principale:

- Gestion des entrées utilisateur
- Mise à jour des états des objets
- Détection des collisions
- Rendu graphique

2. Système de collision:

- Détection précise avec des rectangles englobants
- Traitement différent selon le type d'objet
- Gestion des collisions latérales et verticales

3. Système de power-ups:

- Les champignons rendent Mario plus grand
- Les fleurs de feu permettent de lancer des projectiles
- Effets temporaires et permanents

4. Système d'IA:

- Prise de décision basée sur des probabilités
- Détection d'obstacles et d'ennemis
- Comportement adaptatif selon la situation

5. Gestion des niveaux:

- Chargement à partir de fichiers texte
- Placement des objets selon une grille
- Vérification des conditions de victoire