Triominos

Application de bureau — jeu de plateau Triominos

Développeur	COUR Cyrille
Contexte	Projet personnel
Formation	BTS SIO SLAM — Alternance 2025/2027
Stack	Python 3.13 / PyQt6 6.11.0
Date	Septembre 2026

Sommaire

1. Contexte

Triominos est une application de bureau qui rejoue le jeu de plateau du même nom : 56 tuiles triangulaires portant trois chiffres de 0 à 5, que l'on aboute en faisant coïncider les pointes. De 2 à 4 joueurs, humains ou pilotés par l'ordinateur, sur un seul écran.

Le projet vise à disposer du jeu sans sortir la boîte physique, en appliquant fidèlement la règle du livret fourni (Triominos.pdf) — en particulier la règle du pont, dont la définition est régulièrement mal comprise. Il ne s'agit ni d'un jeu en ligne, ni d'un service : aucun compte, aucun serveur, aucune sauvegarde. Une fenêtre, une partie.

Note — Cette procédure s'adresse à toute personne souhaitant installer, lancer, tester ou reprendre le développement du projet Triominos.

- Jouer une partie complète de 2 à 4 joueurs, en mêlant librement humains et adversaires pilotés par l'ordinateur.
- Appliquer automatiquement les règles de score du livret : valeur faciale, pont (+40), hexagone (+50 / +60 / +70), pénalités de pioche, bonus de dernière tuile.
- Choisir une condition de fin : 1, 2 ou 3 manches, tournoi en 400 points, ou mode infini.
- Visualiser les cases jouables et un aperçu de pose au survol, avec système de conseil intégré.

2. Environnement

2.1 Prérequis

- Python 3.10 ou supérieur (testé en production sur 3.13.7) — la contrainte n'est pas déclarée dans un fichier, elle vient de la syntaxe employée (int | None, tuple[int, int] sans typing) avec from __future__ import annotations en tête de chaque module.
- pip (testé en 26.0.1) pour l'installation des dépendances.
- Un environnement graphique capable d'afficher une fenêtre Qt. Aucun serveur, aucune base de données requise.

2.2 Dépendances

Une seule dépendance de production, déclarée dans requirements.txt :

Paquet	Contrainte	Installée	Rôle
PyQt6	>=6.5	6.11.0	Fenêtres, dessin (QPainter), événements souris/clavier, timers
PyQt6-Qt6	(transitif)	6.11.0	Binaires Qt eux-mêmes
PyQt6-sip	(transitif)	—	Couche de liaison C++/Python générée par SIP

Note — Aucune dépendance de développement : les tests sont des scripts Python autonomes à base d'assert, sans pytest ni unittest.

2.3 Installation

```
git clone <url-du-depot> Triomino
cd Triomino

python -m venv .venv
# Windows
.venv\Scripts\activate
# macOS / Linux
source .venv/bin/activate

pip install -r requirements.txt
```

3. Procédure

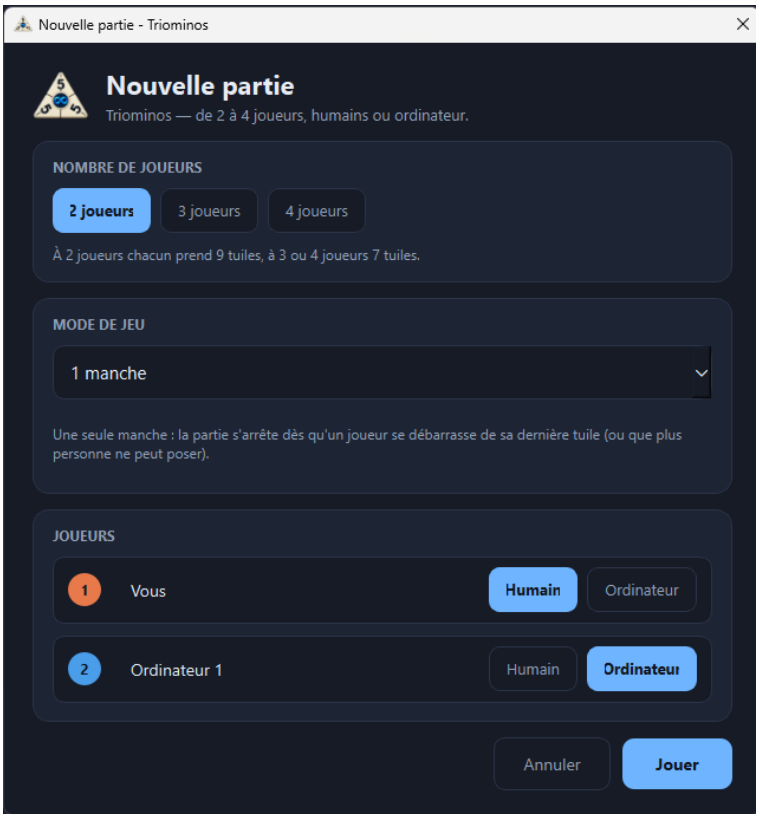
3.1 Étape 1 — Lancer l'application

Détail technique

Depuis la racine du projet, l'environnement virtuel activé, lancer le point d'entrée principal :

```
python main.py
```

Le dialogue de nouvelle partie s'ouvre aussitôt : nombre de joueurs (2 à 4), mode de jeu, noms et nature (humain / ordinateur) de chacun.



3.2 Étape 2 — Jouer une partie

Détail technique

Les commandes clavier / souris principales pilotant une partie sont résumées ci-dessous.

Action	Commande
Choisir une tuile	Clic sur la réglette, ou touches 1–9
Faire tourner la tuile	R, clic droit, ou re-clic sur la tuile choisie
Poser la tuile	Clic sur une case en surbrillance
Piocher	P (3 fois maximum)
Passer son tour	Espace (dernier recours uniquement)
Demander un conseil	C
Recentrer le plateau	F
Nouvelle partie	Ctrl+N
Règles du jeu	F1

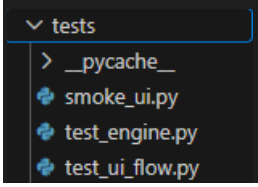
Le moteur de règles (Game.is_valid / Game.play, triomino/game.py) est la seule autorité qui valide un coup et calcule son score ; l'interface se contente de relayer le résultat.

3.3 Étape 3 — Exécuter les tests

Détail technique

Trois suites de tests autonomes, sans framework, sont exécutables individuellement :

```
python tests/test_engine.py            # 27 tests : géométrie, règles, bonus, modes
python tests/test_ui_flow.py           # 5 parties complètes via l'interface
python tests/smoke_ui.py capture.png   # capture du plateau, hors écran
```



Astuce — Les suites d'interface tournent hors écran (QT_QPA_PLATFORM=offscreen, posé par les scripts eux-mêmes) : aucun affichage n'est nécessaire pour les exécuter, y compris en CI.

4. Structure / Données

Aucune base de données : l'état d'une partie tient entièrement en mémoire, porté par des dataclasses gelées (frozen) pour ce qui ne doit plus bouger une fois posé.

Élément	Description	Remarque
Tile	Une tuile : tuple de 3 chiffres (0–5) triés, points calculés	Frozen, ordonnable — 56 instances possibles
Move	Un coup candidat : cellule visée, tuile, rotation	Frozen — valide ou non selon Game.is_valid
Placement	Une tuile effectivement posée sur le plateau	Frozen — porte l'ordre de pose et le joueur
Player	Un joueur : nom, is_ai, main, score, dernière tuile piochée	hand triée sauf tuile piochée en bout
GameMode	Une condition de fin (clé, libellé, manches, cible, infini)	5 instances déclarées dans GAME_MODES
Game	État mutable global : plateau, sommets, pioche, joueurs, manche, journal	Seule porte d'entrée du moteur — play(move)

4.1 Modèle de relations

```
Game 1 ——— 2..4 Player 1 ——— * Tile          (la main)
Game 1 ——— *    Placement 1 ——— 1 Move ——— 1 Tile   (le plateau)
Game 1 ——— 1    GameMode                     (la condition de fin)
```

4.2 Où trouver quoi

Je cherche...	Aller voir
Les règles du jeu, les scores, les modes	triomino/game.py (630 lignes)
La grille triangulaire, les voisins, les hexagones	triomino/geometry.py
Les 56 tuiles et leurs rotations	triomino/tiles.py
L'adversaire ordinateur et les conseils	triomino/ai.py (31 lignes)
L'enchaînement des tours, les raccourcis clavier	triomino/ui/main_window.py
Le dessin du plateau, le zoom, le survol	triomino/ui/board_widget.py
Les couleurs et le dessin d'une tuile	triomino/ui/painting.py

5. Déploiement

Triominos est une application de bureau locale, sans serveur ni service distant : il n'existe pas de pipeline CI/CD ni d'environnement de production au sens des applications web du portfolio. La "mise en production" se limite à récupérer le code et à l'exécuter localement.

5.1 Mise à disposition

- Cloner le dépôt Git sur le poste cible.
- Créer l'environnement virtuel et installer requirements.txt (voir section 2.3).
- Lancer python main.py.

5.2 Ce qui n'est volontairement pas mis en place

Absent	Pourquoi
Packaging (setup.py, PyInstaller)	Le jeu se lance avec python main.py, aucun besoin d'exécutable autonome à ce stade
CI/CD	Application locale, sans environnement à déployer
Base de données / sauvegarde	Une partie vit entièrement en mémoire, la relancer coûte trois clics
Réseau / multijoueur en ligne	Tous les joueurs sont devant le même écran

Note — Si une distribution binaire (PyInstaller ou équivalent) devenait nécessaire, elle ferait l'objet d'une mise à jour dédiée de cette section.