Jeu de la vie — Tests unitaires

Projet C++17 (Grid / GameOfLife / ConwayRule)

1 Objectif

Documenter les tests unitaires ajoutés pour la logique du Jeu de la vie (sans SFML). Les tests valident la règle de Conway, la stabilité et quelques cas limites courants.

2 Architecture du runner

- Fichier `tests/main.cpp` : harness minimaliste (pas de framework externe).
- Compilation via `make test` qui produit `tests/test_runner` en liant uniquement les fichiers de logique (`Grid.cpp`, `GameOfLife.cpp`, `InitialStateLoader.cpp`, `GridExporter.cpp`).
- Aucun lien avec SFML : exécution en ligne de commande uniquement.
- Arrêt immédiat sur échec (`std::exit(1)`) pour remonter clairement la première assertion en défaut.

3 Harness et helpers

- `expect(cond, message)` : vérifie une condition, affiche "Test failed: ..." et quitte en cas d'échec.
- `logCase`/`logOk` : traces lisibles en console pour chaque cas de test.
- `makeGrid(vector<string>)` : construit une grille en mémoire à partir de lignes "0/1" ("1" = cellule vivante).
- `expectGrid(grid, rows, msg)` : compare une grille avec un motif attendu (dimensions et états des cellules).

4 Cas de test couverts

1. **Bloc (still life) stable**
   Motif 2×2 vivant entouré de cellules mortes. Après un pas de simulation :
   - Le motif reste identique (vérification par `expectGrid`).
   - `isStable()` retourne vrai (grille inchangée).
   - `hasFinished()` retourne vrai (stabilité atteinte).

2. **Blinker (oscillateur période 2)**
   Ligne de trois cellules vivantes. Après un pas, elle devient colonne; après deux pas, revient à l'horizontale. Les deux phases sont comparées à des motifs attendus.

3. **Cellule isolée qui meurt**
   Une cellule vivante sans voisins doit disparaître au pas suivant (mort par sous-population).

5 Extrait de code (harness)

```
static void expect(bool condition, const std::string& message) {
    if (!condition) {
        std::cerr << "Test failed: " << message << "\n";
        std::exit(1);
    }
}

static Grid makeGrid(const std::vector<std::string>& rows) {
    Grid g(rows.size(), rows.front().size(), false);
    for (int y = 0; y < g.rows(); ++y)
        for (int x = 0; x < g.cols(); ++x)
            g.at(y, x).setState(rows[y][x] == '1'
                ? static_cast<CellState*>(new AliveState())
                : static_cast<CellState*>(new DeadState()));
    return g;
}
```

6 Exécution

- `make test` pour compiler le binaire `tests/test_runner`.
- `tests/test_runner` pour exécuter les cas. Sortie attendue :

```
Running Game of Life unit tests...
[CASE] Still life (block) remains stable... OK
[CASE] Blinker oscillates with period 2... OK
[CASE] Isolated live cell dies in one step... OK
All tests passed.
```

7 Extensions possibles

- Ajouter un test de mode torique (`Grid` avec `toroidal=true`) pour vérifier le comptage des voisins.
- Ajouter des tests d'échec du chargeur de fichier (`InitialStateLoader`) pour détecter un format invalide.
- Couvrir des motifs plus longs (planeur, pulsar) et la détection de stabilité après plusieurs pas.

- Intégrer un framework léger (Catch2/doctest) si les besoins de reporting/fixtures augmentent.