Test Setup

Directory Structure

The testing files are located in the `Tests` directory within the project structure. The main test file created is `test_gamelogic.cpp`.

Test Case: `test_gamelogic.cpp`

- `class TestGameScreen : public QObject`: Defines a new class `TestGameScreen` that inherits from `QObject`, the base class for all Qt objects.

- `Q_OBJECT`: Macro from the Qt framework that enables the use of signals and slots in the class.

- `private slots`: Defines private member functions that are test cases. Qt Test runs these functions automatically.

```
void TestGameScreen::initTestCase()
{
    // Initialization code here
}
```

- This function is run once before all other test functions.
- It is typically used for setting up resources or initializing variables that will be used in multiple tests.

testSetAndGetMode

```
void TestGameScreen::testSetAndGetMode()
{
    GameScreen gameScreen;
    gameScreen.setMode("Single Player");
    QCOMPARE(gameScreen.getMode(), QString("Single Player"));
}
```

- This test function verifies that the `setMode` and `getMode` methods work correctly.
- A `GameScreen` object is created.
- The mode is set to "Single Player".
- `QCOMPARE` is used to check if the mode retrieved by `getMode` is "Single Player". If they are not equal, the test fails.

testSetAndGetUsername

```
void TestGameScreen::testSetAndGetUsername()
{
    GameScreen gameScreen;
    gameScreen.setUsername("Player1");
    QCOMPARE(gameScreen.getUsername(), QString("Player1"));
}
```

- This test function checks that the `setUsername` and `getUsername` methods function correctly.
- Similar to `testSetAndGetMode`, but it works with the username.

`testHandleResetButtonClicked`

```
void TestGameScreen::testHandleResetButtonClicked()
{
    GameScreen gameScreen;
    gameScreen.handleResetButtonClicked();
    // Add assertions to check the reset functionality
    // Example: QCOMPARE(gameScreen.someResetState(), expectedValue);
}
```

- This test function verifies that the `handleResetButtonClicked` method works correctly.
- The method is called on a `GameScreen` object.
- Comments indicate where you would add assertions to verify the reset functionality.

`testCheckDraw`

```
void TestGameScreen::testCheckDraw()
{
    GameScreen gameScreen;
    // Setup a draw scenario
    // Example: gameScreen.setBoard(drawBoard);
    QCOMPARE(gameScreen.checkDraw(), true);
}
```

- This test function verifies that the `checkDraw` method works correctly.
- A draw scenario would need to be set up for the test to be meaningful.
- `QCOMPARE` checks if `checkDraw` returns `true` in a draw scenario.

`testCheckWinner`

```
void TestGameScreen::testCheckWinner()
{
    GameScreen gameScreen;
    // Setup a winner scenario
    // Example: gameScreen.setBoard(winnerBoard);
    QCOMPARE(gameScreen.checkWinner(), true);
}
```

- This test function verifies that the `checkWinner` method works correctly.
- A winner scenario would need to be set up.
- `QCOMPARE` checks if `checkWinner` returns `true` in a winner scenario.

`testHandleButtonClick`

```
void TestGameScreen::testHandleButtonClick()
{
    GameScreen gameScreen;
    gameScreen.handleButtonClick(0, 0);
    // Add assertions to check the button click handling
    // Example: QCOMPARE(gameScreen.someButtonState(), expectedValue);
}
```

- This test function verifies that the `handleButtonClick` method works correctly.
- The method is called with specific row and column indices.
- Comments indicate where you would add assertions to verify the button click handling.

`cleanupTestCase`

```
void TestGameScreen::cleanupTestCase()
{
    // Cleanup code here
}
```

- This function is run once after all other test functions.
- It is typically used for cleaning up resources or variables that were set up in `initTestCase`.

`QTEST_MAIN`

```
QTEST_MAIN(TestGameScreen)
#include "test_gamelogic.moc"
```

- `QTEST_MAIN(TestGameScreen)`: Macro that generates the `main` function for the test application.
- `#include "test_gamelogic.moc"`: Includes the Meta-Object Compiler (MOC) file generated by Qt for the test class. This is necessary for signal-slot connections and other Qt-specific features.