Supreme Checkers

Overview

- A networked Unity-2D Checkers game for Drexel's **`SE-181`**: *Intro to Software Engineering*.
- **Click Here to Get Started**
- In general, check the **Wiki** if you have a question, or refer to one of the developers on **[Discord]** (contact `peter201943#8017` for access)
- This project can be accessed on *GitHub*, where it is hosted along with its *Issues* (Bug Trackers), *Releases* (Builds), *Wiki* (Discussion/Knowledge), and other features

File Structure

- This is a conglomerate of separate concerns:
  - Documentation
  - Third Party Libraries
  - Learning
- As such, the **root** file structure reflects this:
  - The **Game Files** are stored in **`Unity`**
  - The **Document Source Code** is stored in **`docs`**
  - The **Rendered Documents** are in **`gen`**
  - The **Code Coverage** is in **`CodeReport`**
  - Within **`Unity/Assets`**, there are **two** folders
    - The **Checkers Game** in **`Checkers`**
    - The **Networking Tutorial** in **`RW`**
    - The Network Tutorial, out of a lack of time, is also where the **Networking Libraries** are located:
      - The **Core Networking** in **`Unity/Assets/RW/Photon/PhotonUnityNetworking`** (*BIG* folder, lots of useful scripts)
      - The **Realtime Networking** in **`Unity/Assets/RW/Photon/PhotonRealtime`**
- There are some miscellanious folders that need to be cleaned up:
  - **`Unity/Assets/Photon`**: Empty, nothing important in here
  - **`Unity/Assets/StreamingAssets`**: Again, nothing important
  - **Most of `Unity/Assets/RW`**: There are many files in here belonging to the *Tutorial*, that are not needed for the *Checkers* game

Game File Structure

- Each of the *Elements* of the game (*Board*, *Piece*, *Player*, *Game*, *Tests*) gets its own folder, where a *script* and/or *scene*/*prefab* is stored
- A better understanding of each class can be had by visiting the comments in the source code
- The *Elements* are:
  - **`Board`**
    - A *Prefab* and a *Script*
    - The *Prefab* contains an `8x8` 3D Grid of Cubes with `Tile` components attached to make the "board"
    - The *Script* Handles almost everything, from *Cell Highlighting*, to *Networking*, to *Turn Control*, and so on
    - Potentially too *Big*
    - Also contains the *non-GameObject* **`Space`** class, which has various stats for a cell/tile/grid/square/space on the board
  - **`Game`**
    - Just a *Scene* with an instance of **`Board`** and many **`Pieces`**
    - Is the **"Main Scene"** that gets loaded *after* the **Launcher**
  - **`Piece`**
    - Nothing?
  - **`Player`**
    - Mostly stats, such as whether the player is the *"Current"* Player
  - **`Tests`**
    - The *Unit*, *Integration*, and other Tests required by the course
  - **`Launcher`**
    - Taken from the *Tutorial*, is a simple matchmaking menu
    - This should be the **"First Scene"** that gets loaded on opening the app

Testing

- Unity has a built-in testing framework that uses "Assembly Definition Files" to *"see"* other scripts
- These are `.json` files that must be added to whatever directories with scripts in them that you want to be able to test
- An `.asmdef` file exists in the major script locations:
    - `Unity/Assets/Checkers/Board/Board.asmdef`
    - `Unity/Assets/Checkers/Piece/Piece.asmdef`
    - `Unity/Assets/Checkers/Player/Player.asmdef`
    - `Unity/Assets/RW/Scripts/RW.asmdef`
- We assume that the included *Photon* libraries work, and so no `.asmdef` files have been created for them
- For more on testing inside *Unity*, visit these pages on the wiki
- There are **two** kinds of tests:
    - **`EditMode`**
        - These are tests that run in the *editor*, and *not* during *play*
        - Similar to **Unit Tests**, these tests **cannot** access the *Scene*, but run faster and at any time
        - Better to test the individual methods of a class
    - **`PlayMode`**
        - These are tests that run *in game*, and *not in editor*
        - Similar to **Integration Tests**, these tests can access the **Scene** and talk to other **GameObjects**
        - Better to test the behavior of multiple GameObjects interacting with each other
- The Tests should be located in `Unity/Assets/Checkers/Tests/`
    - There are **two** subfolders, labelled `PlayMode` and `EditMode`
    - There is an example test in each folder
- For Code Coverage/Static Analysis, we used **Roslyn** with **Visual Studio**

Coding

- Each of the scripts has some documentation
- Please add your notes to them as you write them out, what issues you are having, etc

Issues

- If you have the time, please add any persistent issues to the *Github Issues Tracker*
- Otherwise, note the issue in the script location of the problem, eg on a **Method** or on a **Class** as a *comment**

Branches

- (last updated 2020-11-30T09:25:00-05)
- **`master`:** The current branch
- **`Sound-Highlight`:** Code Improvements, but broken
- **`TurnFixes`:** Move fixes, integrated

Releases

`Version 1.0.0`

Logic and networking code for game reached completion point.

`Version 1.0.1`

Functionality for the sound and credits were added.

`Version 1.0.2`

Code and sounds credits added to project. As well the functionality for implementing sounds were added.

`Version 1.0.3`

Piece path highlighting was added to the game.

`Version 1.0.4`

Process of adding in unit tests was began.

`Version 1.0.5`

Added in the rest of the unit tests to the project.

`Version 1.0.6`

Fixed logic issues with jumps.

`Version 1.0.7 - Final Release`

Implemented final missing UI items