Getting started

This document describes steps required for using Tomino in your Unity project.

Core components

The core logic of the game is controlled by components that have no dependencies on Unity. This components include:

- `Game` - Controls the game logic by handling user input and updating the board state.
- `Board` - Contains collection of blocks placed on the board and allows for moving them within the defined bounds.
- `Block` - A block with specified type that can be placed (and moved) on a board.
- `Piece` - A piece is a collection of blocks that all move together on the board.

Separating from Unity API facilitates game logic testing as there is no need to deal with `MonoBehavior` dependencies.

Creating the `Board` and the `Game`

The first step is to create a `Board` which will define boundaries and positions for blocks and the piece controlled by the player.

```
Board board = new Board(10, 20);
```

The next step is to instantiate the `Game` object that is responsible for controlling the main logic by handling user input and updating the board tate. The game pools user input from the provided `IPlayerInput` parameter.

```
var game = new Game(board, new KeyboardInput());
```

The game needs to receive update events and because it's not a `MonoBehavior` it has to be done manually, e.g. by the parent controller class.

```
class GameController: MonoBehavior
{
    void Update()
    {
        game.Update(Time.deltaTime);
    }
}
```

When configuration is finished the game can be started by calling the `game.Start()` method.

Rendering the `Board`

Both `Game` and the `Board` represent the current (in memory) state of the gameplay. This classes are separated from Unity APIs which means that endering has to be handled by other components, such as `BoardView`, `PieceView` or `ScoreView`.

In addition, the `BoardView` contains an instance of the `TouchInput` which can be passed to the `Game` constructor.