Getting started

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

Core components

The core components that control the game logic have no dependencies on Unity. These components include:

- `Game` - Controls the game logic by handling user input and updating the board state.
- `Board` - Contains a collection of blocks placed on the board and allows for moving them within the defined bounds.
- `Block` - A unit-sized rectangle with a specified type that the `Board` renders.
- `Piece` - A piece is a collection of blocks moving together on the `Board`.

Separating from Unity API facilitates game logic testing, as dealing with `MonoBehavior` dependencies is unnecessary.

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 responsible for controlling the main logic by handling user input and updating the board state. 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);
    }
```

```
}
```

After configuration, you can start the `Game` using the `game.Start()` method.

Rendering the `Board`

Both the `Game` and the `Board` represent the gameplay's current (in memory) state. These classes are separated from Unity APIs, meaning other components, such as `BoardView`, `PieceView` or `ScoreView` handle rendering.

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

Theming

Tomino supports customizing colours it uses to tint individual game and UI components. The theming system comprises `Theme`, `ThemeProvider` and `ThemeController` components.

The `Theme` is a `ScriptableObject` containing the colour scheme. Tomino provides four pre-defined themes (found in the Tomino/Theme folder): Default Autumn Pink Teal You can create a custom theme using the Assets -> Create -> Tomino -> Theme menu.

The `ThemeProvider` is a `ScriptableObject` that references the current `Theme`.

The `ThemeController` is a `MonoBehaviour` that listens to the `Settings` change events and updates the `ThemeProvider.currentTheme` value accordingly.