Instrucciones para feature adicional Ghost Piece

1. Crear un nuevo script llamado Ghost.cs, que tendrá código que ya hemos escrito en otros scripts.
2. Importar de Unity lo siguiente:

```
using UnityEngine;
using UnityEngine.Tilemaps;
```

3. Declarar las siguientes variables, que son similares a las que están en el script Piece.cs, la variable trackingPiece se refiere a la pieza que se va a proyectar. También crear las tres propiedades de tilemap, cells y position como en el script Piece.cs:

```
public Tile tile;
public Board board;
public Piece trackingPiece;

public Tilemap tilemap { get; private set; }
public Vector3Int[] cells { get; private set; }
public Vector3Int position { get; private set; }
```

4. En este caso, se requiere inicializar las variables tilemap y cells en la función predeterminada Awake:

```
private void Awake() {
    this.tilemap = GetComponentInChildren<Tilemap>();
    this.cells = new Vector3Int[4];
}
```

5. Ahora, se debe extraer la información de la pieza a proyectar y simular una especie de HardDrop. Vamos a crear cuatro funciones y a utilizar una función de Unity llamada LateUpdate, que se ejecuta después del Update corriente, esto porque los datos de la pieza pueden variar en el Update y la proyección debe actualizarse después de igual manera. Las cuatro funciones se llamarán en orden dentro de LateUpdate.

```
private void LateUpdate() {
    Clear();
    Copy();
    Drop();
    Set();
}

private void Clear() {
}
```

```
private void Copy() {
}

private void Drop() {
}

private void Set() {
}
```

6. Las funciones Set y Clear son similares a los métodos del mismo nombre en el script Board.cs. Únicamente se cambian los piece y piece.data por this. La función Clear tiene el código de Set, pero en lugar de setear el tile, se setea un null:

```
private void Clear() {
    for(int i = 0; i < this.cells.Length; i++) {
        Vector3Int tilePosition = this.cells[i] + this.position;
        this.tilemap.SetTile(tilePosition, null);
    }
}

private void Copy() {
}

private void Drop() {

}

private void Set() {
    for(int i = 0; i < this.cells.Length; i++) {
        Vector3Int tilePosition = this.cells[i] + this.position;
        this.tilemap.SetTile(tilePosition, this.tile);
    }
}
```

7. La función Copy recorre en un ciclo for las cells del script Ghost y les asigna el valor que tienen las celdas respectivas de la variable trackingPiece:

```
private void Copy() {
    for(int i = 0; i < this.cells.Length; i++) {
        this.cells[i] = this.trackingPiece.cells[i];
    }
}
```

8. Por último, la función Drop inicia en la posición de la pieza actual en descenso y recorre las filas hasta que encuentra una posición inválida o el borde inferior del

tablero. En un ciclo for el cálculo de la proyección se va corriendo una fila hacia abajo mientras la función IsValidPosition retorna true. Al inicio, esta función va a retornar false porque la pieza Ghost se coloca en la misma posición de la pieza real, por lo que es necesario quitar esta última, calcular la posición de Ghost y luego volver a colocarla para evitar problemas. El código quedaría así:

```
private void Drop() {
    Vector3Int position = this.trackingPiece.position;

    int current = position.y;
    int bottom = -this.board.boardSize.y / 2 - 1;

    this.board.Clear(this.trackingPiece);

    for(int row = current; row >= bottom; row--) {
      position.y = row;

      if(this.board.isValidPosition(this.trackingPiece, position)) {
        this.position = position;
      } else {
        break;
      }
    }

    this.board.Set(this.trackingPiece);
}
```

9. En el editor de Unity, agregar el script de Ghost.cs al game object Ghost.
10. Completar los siguientes campos con las opciones que el mismo editor reconoce: