Wacky Breakout Increment 4
Detailed Instructions

Overview

In this project (the fourth increment of our Wacky Breakout game development), you're adding more functionality to the game.

To give you some help in approaching your work for this project, I've provided the steps I implemented when building my project solution. I've also provided my solution to the previous increment in the materials zip file, which you can use as a starting point for this project if you'd like. As you can see, the steps for this increment are less detailed than those for previous Wacky Breakout increments; that's because you're maturing as a game programmer and should be able to figure out how to do more without detailed instructions.

Step 1: Add Bonus blocks

For this step, you're adding the bonus blocks.

Add a value in the configuration data CSV file for the Bonus block points (I used 2 for this) and add the appropriate fields and properties to the `ConfigurationData` and `ConfigurationUtils` class so the `BonusBlock` class (see next paragraph) will be able to access this value.

Add a sprite for the Bonus block and create a new BonusBlock script as a child class of the `Block` class. Add code to the `BonusBlock Start` method to set the points the block is worth to the Bonus block value from the `ConfigurationUtils` property.

Make a BonusBlock prefab using the Bonus block sprite and the BonusBlock script. Add a field for the BonusBlock prefab to the LevelBuilder script and populate that field in the Inspector. Change the LevelBuilder script to only use Bonus blocks so you can make sure the scoring works properly.

Step 2: Add Freezer and Speedup blocks

For this step, you're adding the effect (freezer and speedup) blocks. You won't be adding the actual effects until the next increment, but you're adding the effect blocks to the level now.

Add a value in the configuration data CSV file for the Effect block points (I used 5 for this) and add the appropriate fields and properties to the `ConfigurationData` and `ConfigurationUtils` class so the `EffectBlock` class (see next paragraph) will be able to access this value.

Add sprites for the Freezer and Speedup blocks and create a new EffectBlock script as a child class of the `Block` class. Add code to the `EffectBlock Start` method to set the points the block is worth to the Effect block value. Add fields to the script to store the two effect sprites and populate them in the Inspector.

Make an EffectBlock prefab using the Freezer block sprite and the EffectBlock script. Add a field for the EffectBlock prefab to the LevelBuilder script and populate that field in the Inspector.

Copy the EffectName.cs file from the zip file into your Scripts/Gameplay folder. That file defines a EffectName enum with two values: Freezer and Speedup. Add a field to the EffectBlock script to hold which kind of effect the block applies and add a public property for setting the field. Set the sprite for the block appropriately within the set accessor for the property.

Change the LevelBuilder script to only use Freezer blocks so you can make sure the scoring and sprite setting works properly. Do the same for Speedup blocks. Hint: You'll need to save the block game object you instantiate as you add each block to the level so you can get the EffectBlock component attached to that game object so you can access the property to set the effect for the block.

Note: You do NOT have to actually implement the freezer and speedup effects; that happens in the next increment. The points should be correct when you hit a effect block, though.

Step 3: Randomizing the blocks

For this step, you're randomly adding the four types of blocks to the scene based on probabilities.

Add values in the configuration data CSV file for the different block probabilities (I used 70 for standard blocks, 20 for bonus blocks, 5, for freezer blocks, and 5 for speedup blocks). Add the appropriate fields and properties to the `ConfigurationData` and `ConfigurationUtils` class so the `LevelBuilder` class will be able to access those values.

Add code to the LevelBuilder script to randomly decide which block to place each time a block is added to the level based on the probabilities for each block. My `Start` method was getting somewhat large at this point, so I added a `PlaceBlock` method that randomly picks a block to place at a given location.

That’s it for this increment.