Glasshouse Test

Task

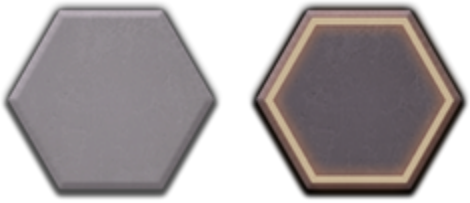
Implement the same puzzle as seen in the vertical slice demo of Glasshouse.

Puzzle Rules

The puzzle takes place on a NxM grid of hexagons.

There are several types of hexagons:

1. Empty: these hexagons can't be interacted with, they can only be powered up with other nodes. If a Empty node is being powered up, it should show the value of the power it has.



The node has two possible sprites: the left one when the node has no power, the right one when the node has power. In the 2nd state, there should be a number inside the hexagon to represent the amount of power that node is receiving

2. Target: these nodes have a digit inside, which represents the number of nearby powered up nodes it requires. The objective of the puzzle is to satisfy the conditions of all Target nodes.

This node is quite complex in terms of sprite representations. The 1st sprite is the default state at the start of the game, the 2nd one is for when the node is receiving power but it is still lower than the target power, the 3rd one is used when the Target node is receiving more power than needed, and the 4th one is for when the target power has been reached. In the first 3 states, a number should be inside the hexagon, to show the amount of power needed

3. Single direction 1x: this node is interactable
    a. Left click: rotate the node counterclockwise (6 possible directions)
    b. Right click: power on and off the node

What the node does is to power up empty nodes in the direction it is facing (1 node only), when it is actually powered up. This is the type of node that you have to manipulate in order to solve the puzzle. Remember these node don't power up Target nodes directly, Target nodes have to be powered up through nearby Empty powered up nodes.

This is the node in unpowered and powered state.

4. Double direction 1x: same as the previous node but it powers up in 2 directions at the same time.

5. Single direction 2x: this is the same as the 1x variant but it powers up 2 nodes along the direction it is facing.

When you power on/off a node, a sound effect should be played, same when the node is rotated (the sounds are provided).

When all target nodes have been correctly powered up, the corresponding “puzzle complete” sound effect should play, along with a shake of the puzzle UI.

Guidelines

The test should be developed following good programming practices and patterns, trying to avoid creating unnecessary dependencies and separating the UI logic from the puzzle logic (or, to make a web development analogy, separating the Front-End from the Back-End).
Bonus points are awarded for a system that is open to the addition of new types of puzzles, with completely different mechanics and representations.