```
1  using System.Collections.Generic;
2  using UnityEngine;
3
4  /// <summary>
5  /// Logical representation of an XOR (EXCLUSIVE OR) gate.
6  /// </summary>
7  public class XOrGate : Circuit
8  {
9      public XOrGate() : this(Vector2.zero) { }
10
11     public XOrGate(Vector2 startingPos) : base("XOR", 2, 1, startingPos)
         { }
12
13     /// <summary>
14     /// Returns an output to update if the output has changed due to
         alterations in input power statuses.
15     /// </summary>
16     /// <returns>The list of outputs that should have their connections
         called.</returns>
17     protected override List<Output> UpdateOutputs()
18     {
19         bool outputStatus = Outputs[0].Powered;
20         List<Output> outputs = new List<Output>();
21
22         // XOR gate representation
23         Outputs[0].Powered = Inputs[0].Powered && !Inputs[1].Powered || !
             Inputs[0].Powered && Inputs[1].Powered;
24
25         if (outputStatus != Outputs[0].Powered || MaterialNotMatching())
             outputs.Add(Outputs[0]);
26
27         return outputs;
28     }
29
30     /// <summary>
31     /// Checks all outputs to determine if the output node material is not
         matching its power status.<br/><br/>
32     /// This is utilized within custom circuits to force update calls that
         would normally not occur due to the nature of UpdateOutputs().
33     /// </summary>
34     /// <returns>Whether any output material does not match its power
         status.</returns>
35     private bool MaterialNotMatching()
36     {
37         if (Outputs[0].StatusRenderer == null) return false;
38
39         return (Outputs[0].Powered && Outputs
             [0].StatusRenderer.sharedMaterial !=
             CircuitVisualizer.Instance.PowerOnMaterial) ||
```

```
40                  (!Outputs[0].Powered && Outputs
                      [0].StatusRenderer.sharedMaterial !=
                      CircuitVisualizer.Instance.PowerOffMaterial);
41      }
42  }
```