```
1  using System;
2  using UnityEngine;
3
4  /// <summary>
5  /// CircuitIdentifier provides circuit serialization and deserialization
     by storing relevant enum values.
6  /// </summary>
7  [Serializable]
8  public class CircuitIdentifier
9  {
10     /// <summary>
11     /// CircuitType is the serialized representation of any circuit
         located within a scene.
12     /// </summary>
13     public enum CircuitType { CUSTOM_CIRCUIT, INPUT_GATE, DISPLAY, BUFFER,
          AND_GATE, NAND_GATE, NOR_GATE, NOT_GATE, OR_GATE, XOR_GATE }
14
15     /// <summary>
16     /// The circuit type to be serialized by this CircuitIdentifier
         instance.
17     /// </summary>
18     [SerializeField]
19     public CircuitType circuitType;
20
21     /// <summary>
22     /// Contains a valid custom circuit ID if the referenced CircuitType
         value is <seealso cref="CircuitType.CUSTOM_CIRCUIT"/>.
23     /// </summary>
24     [SerializeField]
25     public int previewStructureID = -1;
26
27     /// <summary>
28     /// The location of the circuit within the scene.
29     /// </summary>
30     [SerializeField]
31     Vector2 circuitLocation;
32
33     /// <param name="circuit">The circuit to serialize.</param>
34     public CircuitIdentifier(Circuit circuit)
35     {
36         Vector3 pos = circuit.PhysicalObject.transform.position;
37
38         circuitType = CircuitToCircuitType(circuit);
39         circuitLocation = new Vector2(pos.x, pos.z);
40
41         // If the circuit is a custom one, store its ID
42         if (circuitType == CircuitType.CUSTOM_CIRCUIT) previewStructureID
             = ((CustomCircuit)circuit).PreviewStructure.ID;
43     }
```

```
44
45      /// <summary>
46      /// Instantiates and returns a <see cref="Circuit"/> based on the
          provided CircuitIdentifier (using default visibility settings).
47      /// </summary>
48      /// <param name="circuitIdentifier">The CircuitIdentifier to access
          and reference.</param>
49      /// <returns>The instantiated <seealso cref="Circuit"/>.</returns>
50      public static Circuit RestoreCircuit(CircuitIdentifier
          circuitIdentifier)
51      {
52          return RestoreCircuit(circuitIdentifier, true);
53      }
54
55      /// <summary>
56      /// Instantiates and returns a <see cref="Circuit"/> based on the
          provided CircuitIdentifier.
57      /// </summary>
58      /// <param name="circuitIdentifier">The CircuitIdentifier to access
          and reference.</param>
59      /// <param name="visible">False if the circuit is located inside of a
          custom circuit, otherwise true.</param>
60      /// <returns>The instantiated <seealso cref="Circuit"/>.</returns>
61      public static Circuit RestoreCircuit(CircuitIdentifier
          circuitIdentifier, bool visible)
62      {
63          Vector2 pos = visible ? circuitIdentifier.circuitLocation :
              Vector2.positiveInfinity;
64
65          switch (circuitIdentifier.circuitType)
66          {
67              case CircuitType.CUSTOM_CIRCUIT:
68                  return new CustomCircuit
                      (MenuSetupManager.Instance.PreviewStructures
                      [MenuSetupManager.Instance.PreviewStructureIDs.IndexOf
                      (circuitIdentifier.previewStructureID)], pos, visible);
69              case CircuitType.INPUT_GATE:
70                  return new InputGate(pos);
71              case CircuitType.DISPLAY:
72                  return new Display(pos);
73              case CircuitType.BUFFER:
74                  return new Buffer(pos);
75              case CircuitType.AND_GATE:
76                  return new AndGate(pos);
77              case CircuitType.NAND_GATE:
78                  return new NAndGate(pos);
79              case CircuitType.NOR_GATE:
80                  return new NOrGate(pos);
81              case CircuitType.NOT_GATE:
```

```
82                  return new NotGate(pos);
83              case CircuitType.OR_GATE:
84                  return new OrGate(pos);
85              case CircuitType.XOR_GATE:
86                  return new XOrGate(pos);
87              default:
88                  throw new Exception("Invalid circuit type.");
89          }
90      }
91
92      /// <summary>
93      /// Converts the provided <see cref="Circuit"/> to a valid <seealso cref="CircuitType"/> for serialization.
94      /// </summary>
95      /// <param name="circuit">The circuit to convert.</param>
96      /// <returns>The converted <seealso cref="CircuitType"/>.</returns>
97      private static CircuitType CircuitToCircuitType(Circuit circuit)
98      {
99          Type type = circuit.GetType();
100
101         if (type == typeof(CustomCircuit)) return CircuitType.CUSTOM_CIRCUIT;
102
103         if (type == typeof(InputGate)) return CircuitType.INPUT_GATE;
104
105         if (type == typeof(Display)) return CircuitType.DISPLAY;
106
107         if (type == typeof(Buffer)) return CircuitType.BUFFER;
108
109         if (type == typeof(AndGate)) return CircuitType.AND_GATE;
110
111         if (type == typeof(NAndGate)) return CircuitType.NAND_GATE;
112
113         if (type == typeof(NOrGate)) return CircuitType.NOR_GATE;
114
115         if (type == typeof(NotGate)) return CircuitType.NOT_GATE;
116
117         if (type == typeof(OrGate)) return CircuitType.OR_GATE;
118
119         if (type == typeof(XOrGate)) return CircuitType.XOR_GATE;
120
121         throw new Exception("Invalid circuit type.");
122     }
123 }
```