

```
1 using System;
2 using System.IO;
3 using UnityEngine;
4
5 /// <summary>
6 /// ConnectionSerializer stores mesh and index information pertaining to
7 the assigned connection for serialization.
8 /// </summary>
9 [Serializable]
10 public class ConnectionSerializer
11 {
12     /// <summary>
13     /// Whether <seealso cref="startingMesh"/> is equal to <seealso
14     cref="endingMesh"/>.
15     /// </summary>
16     [SerializeField]
17     bool singleWired;
18
19     /// <summary>
20     /// Contains relevant index information used to identify the
21     connection's input and output circuit(s).
22     /// </summary>
23     [SerializeField]
24     CircuitConnectorIdentifier circuitConnectorIdentifier;
25
26     /// <summary>
27     /// Serialized mesh data for the starting wire.
28     /// </summary>
29     [SerializeField]
30     MeshSerializer startingMesh;
31
32     /// <summary>
33     /// Serialized mesh data for the ending wire.
34     /// </summary>
35     [SerializeField]
36     MeshSerializer endingMesh;
37
38     /// <summary>
39     /// Serialized mesh data for the non-starting/ending wires.<br/><br/>
40     /// If there is only a starting and ending wire, its value will be
41     null.
42     /// </summary>
43     [SerializeField]
44     MeshSerializer parentMesh;
45
46     // Private constructor; a ConnectionSerializer can only be instantiated
47     through its primary constructor.
48     private ConnectionSerializer() { }
```

```
45     /// <summary>
46     /// Instantiates and populates a <seealso cref="ConnectionSerializer"/> ↗
47     /// with the assigned values; saves to the provided path.
48     /// </summary>
49     /// <param name="connection">The connection to serialize.</param>
50     /// <param name="circuitConnectorIdentifier">The obtained <seealso ↗
51     ///     cref="CircuitConnectorIdentifier"/> representing this connection.</ ↗
52     ///     param>
53     /// <param name="path">The directory to save the serialized ↗
54     ///     information.</param>
55     public static void SerializeConnection(CircuitConnector.Connection ↗
56     connection, CircuitConnectorIdentifier circuitConnectorIdentifier, ↗
57     string path)
58     {
59         ConnectionSerializer connectionSerializer = new ↗
60         ConnectionSerializer();
61
62         // Assigns relevant values
63         connectionSerializer.circuitConnectorIdentifier = ↗
64         circuitConnectorIdentifier;
65         connectionSerializer.startingMesh = new MeshSerializer ↗
66         (connection.StartingWire.transform.GetChild(0).GetChild ↗
67         (0).GetComponent<MeshFilter>().mesh, ↗
68         connection.StartingWire.transform);
69         connectionSerializer.endingMesh = new MeshSerializer ↗
70         (connection.EndingWire.transform.GetChild(0).GetChild ↗
71         (0).GetComponent<MeshFilter>().mesh, ↗
72         connection.EndingWire.transform);
73         connectionSerializer.singleWired = connection.StartingWire == ↗
74         connection.EndingWire;
75
76         // If the connection has a parent mesh, serialize its information ↗
77         and save to parentMesh.
78         if (connection.GetComponent<MeshFilter>() != null) ↗
79         {
80             connectionSerializer.parentMesh = new MeshSerializer ↗
81             (connection.GetComponent<MeshFilter>().mesh, ↗
82             connection.transform);
83         }
84
85         // Writes object to directory
86         File.WriteAllText(path, JsonUtility.ToJson(connectionSerializer));
87     }
88
89     // Getter methods
90     public bool SingleWired { get { return singleWired; } }
91
92     public CircuitConnectorIdentifier CircuitConnectorIdentifier { get ↗
93     { return circuitConnectorIdentifier; } }
94
95     public MeshSerializer StartingMesh { get { return startingMesh; } }
```

```
74
75     public MeshSerializer EndingMesh { get { return endingMesh; } }
76
77     public MeshSerializer ParentMesh { get { return parentMesh; } }
78 }
```