```
 1  using System;
 2  using System.Collections;
 3  using System.Collections.Generic;
 4  using UnityEngine;
 5
 6  /// <summary>
 7  /// CircuitCaller handles every circuit call after a short delay defined
      in <see cref="Circuit"/>.
 8  /// </summary>
 9  public class CircuitCaller : MonoBehaviour
10  {
11      private static CircuitCaller instance; // Singleton state reference
12
13      // Enforces a singleton state pattern
14      private void Awake()
15      {
16          if (instance != null)
17          {
18              Destroy(this);
19              throw new Exception("CircuitCaller instance already
                  established; terminating.");
20          }
21
22          instance = this;
23      }
24
25      /// <summary>
26      /// Starts a coroutine that shortly accesses the list of provided
          update calls.
27      /// </summary>
28      /// <param name="updateCalls">The list of update calls to pursue.</
          param>
29      public static void InitiateUpdateCalls(List<Circuit.UpdateCall>
          updateCalls) { instance.StartCoroutine(UpdateCalls(updateCalls)); }
30
31      /// <summary>
32      /// Attempts to access the list of provided update calls.
33      /// </summary>
34      /// <param name="updateCalls">The list of update calls to call.</
          param>
35      private static IEnumerator UpdateCalls(List<Circuit.UpdateCall>
          updateCalls)
36      {
37          yield return new WaitForSeconds(Circuit.clockSpeed);
38
39          foreach (Circuit.UpdateCall updateCall in updateCalls)
40          {
41              // Sometime between the call initiation and now, the
                    referenced output was destroyed and should no longer be
```

```
                pursued.
42              if (updateCall.Input.ParentOutput == null) continue;
43
44              if (!CustomCircuitTest(updateCall)) continue;
45
46              // Otherwise, the update call is accessed to update the
                  relevant circuits.
47              Circuit.UpdateCircuit(updateCall.Powered, updateCall.Input,
                  updateCall.Output);
48          }
49      }
50
51      /// <summary>
52      /// Ensures that an update call pertaining to a custom circuit only
          runs if its custom circuit is not deleted.
53      /// </summary>
54      /// <param name="updateCall">The update call to test.</param>
55      /// <returns>Whether this update call should be utilized.</returns>
56      private static bool CustomCircuitTest(Circuit.UpdateCall updateCall)
57      {
58          // In preview scene, therefore not necessary to run the test
59          if (EditorStructureManager.Instance == null) return true;
60
61          // If the input of an update call is under a parent circuit, it is
               guaranteed that its output is as well.
62          bool isInternalConnection =
                updateCall.Input.ParentCircuit.customCircuit != null;
63
64          // Connection does not pertain to the inside of a custom circuit.
65          if (!isInternalConnection) return true;
66
67          // Otherwise, obtain the top-most custom circuit and check to see
                if it is still within the scene.
68          CustomCircuit customCircuitParent =
                updateCall.Input.ParentCircuit.customCircuit;
69
70          while (customCircuitParent.customCircuit != null)
                customCircuitParent = customCircuitParent.customCircuit;
71
72          return !customCircuitParent.ShouldDereference;
73      }
74
75      /// <summary>
76      /// Deletes the specified circuit from the scene.
77      /// </summary>
78      /// <param name="circuit">The circuit to destroy.</param>
79      public static void Destroy(Circuit circuit)
80      {
81          // First disconnects any potential input connections
```

```
82          foreach (Circuit.Input input in circuit.Inputs)
83          {
84              if (input.Connection != null)
85              {
86                  CircuitConnector.Disconnect(input.Connection);
87              }
88          }
89
90          // Then disconnects any potential output connections
91          foreach (Circuit.Output output in circuit.Outputs)
92          {
93              foreach (CircuitConnector.Connection connection in new
                  List<CircuitConnector.Connection>(output.Connections))
94              {
95                  CircuitConnector.Disconnect(connection);
96              }
97          }
98
99          // Ensures all remaining calls within the custom circuit are
              skipped
100         if (circuit.GetType() == typeof(CustomCircuit)) ((CustomCircuit)
              circuit).ShouldDereference = true;
101
102         EditorStructureManager.Instance.DisplaySavePrompt = true; //
              Destroying a circuit triggers the save prompt
103         EditorStructureManager.Instance.Circuits.Remove(circuit); //
              Removes circuit for potential serialization
104         Destroy(circuit.PhysicalObject);
105     }
106 }
```