```
using System;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

/// <summary>
/// MenuLogicManager handles all scene transitions as well as validation tests for deleting custom circuits.
/// </summary>
public class MenuLogicManager : MonoBehaviour
{
    // Singleton state reference
    private static MenuLogicManager instance;

    /// <summary>
    /// Whether the current editor scene has just been created or not.<br/><br/>
    /// If it is the first time, this value indicates that default values must be initialized in the editor scene.
    /// </summary>
    private bool firstOpen;

    /// <summary>
    /// The index of the current editor scene to open.
    /// </summary>
    private int currentSceneIndex;

    /// <summary>
    /// The current preview structure in the preview scene.
    /// </summary>
    private PreviewStructure currentPreviewStructure;

    /// <summary>
    /// Utilized within <seealso cref="TraversalTest(PreviewStructure, PreviewStructure)"/> to sort all preview structures into these respective lists.<br/>
    /// If a circuit is invalid, it means that it is being utilized somewhere and cannot be deleted. A valid circuit can be deleted.
    /// </summary>
    private List<PreviewStructure> invalidCircuits, validCircuits;

    // Enforces a singleton state pattern
    private void Awake()
    {
        if (instance != null)
        {
            Destroy(this);
            throw new Exception("MenuLogicManager instance already established; terminating.");
```

```
43          }
44
45          instance = this;
46      }
47
48      /// <summary>
49      /// Attempts to open an editor scene by index.
50      /// </summary>
51      /// <param name="sceneIndex">Index of the editor structure to open.</
          param>
52      public void OpenScene(int sceneIndex)
53      {
54          bool isCreated = MenuSetupManager.Instance.EditorStructures
            [sceneIndex] != null;
55
56          firstOpen = !isCreated;
57
58          // Opens the scene only if it has not been just created; otherwise
             completes the setup process.
59          if (isCreated) ImportScene(sceneIndex); else
            MenuInterfaceManager.Instance.BeginSceneNameSubmission
            (sceneIndex);
60      }
61
62      /// <summary>
63      /// Creates an editor scene.
64      /// </summary>
65      /// <param name="sceneIndex">Index of the editor structure to
          create.</param>
66      /// <param name="name">Name of the editor structure to create.</param>
67      public void CreateScene(int sceneIndex, string name)
68      {
69          MenuSetupManager.Instance.EditorStructures[sceneIndex] = new
            EditorStructure(name);
70          ImportScene(sceneIndex);
71      }
72
73      /// <summary>
74      /// Opens a preview structure.
75      /// </summary>
76      /// <param name="previewStructure">The preview structure to open.</
          param>
77      public void OpenPreview(PreviewStructure previewStructure)
78      {
79          currentPreviewStructure = previewStructure;
80          SceneManager.LoadScene(2);
81      }
82
83      /// <summary>
```

```
84      /// Opens a scene that has already been created beforehand.
85      /// </summary>
86      /// <param name="sceneIndex"></param>
87      private void ImportScene(int sceneIndex)
88      {
89          currentSceneIndex = sceneIndex;
90          SceneManager.LoadScene(1);
91      }
92
93      /// <summary>
94      /// Runs several validation tests that determine whether a custom
          circuit can be deleted.<br/><br/>.
95      /// If there are no error messages returned, then the custom circuit
          can be deleted.
96      /// </summary>
97      /// <param name="_previewStructure">The prospect custom circuit/
          preview structure to delete.</param>
98      /// <returns></returns>
99      public static List<string> CanDeleteCustomCircuit(PreviewStructure
          _previewStructure)
100     {
101         List<string> errorMessages = new List<string>();
102         instance.invalidCircuits = new List<PreviewStructure>();
            instance.validCircuits = new List<PreviewStructure>();
103
104         // Marks each preview structure as either valid or invalid
105         foreach (PreviewStructure previewStructure in
            MenuSetupManager.Instance.PreviewStructures)
            instance.TraversalTest(previewStructure, _previewStructure);
106
107         // Tests #1/#2: not placed or part of a bookmark directly (top-
            most custom circuit) or indirectly (inside of another custom
            circuit) within an editor scene.
108         foreach (EditorStructure editorStructure in
            MenuSetupManager.Instance.EditorStructures)
109         {
110             if (editorStructure == null) continue;
111
112             // Placed circuit test
113             foreach (CircuitIdentifier circuitIdentifier in
                editorStructure.Circuits)
114             {
115                 if (circuitIdentifier.previewStructureID == -1) continue;
116
117                 PreviewStructure previewStructure =
                    MenuSetupManager.Instance.PreviewStructures
                    [MenuSetupManager.Instance.PreviewStructureIDs.IndexOf
                    (circuitIdentifier.previewStructureID)];
118
```

```
119                 if (instance.invalidCircuits.Contains(previewStructure))
                      { errorMessages.Add("Circuit is directly and/or
                      indirectly referenced by a placed circuit in an editor
                      scene"); break; }
120             }
121
122             // Bookmark test
123             foreach (int customCircuitID in editorStructure.BookmarkIDs)
124             {
125                 if (customCircuitID == -1) continue;
126
127                 PreviewStructure previewStructure =
                      MenuSetupManager.Instance.PreviewStructures
                      [MenuSetupManager.Instance.PreviewStructureIDs.IndexOf
                      (customCircuitID)];
128
129                 if (instance.invalidCircuits.Contains(previewStructure))
                      { errorMessages.Add("Circuit is directly and/or
                      indirectly referenced by a bookmark in an editor
                      scene"); break; }
130             }
131         }
132
133         // Test #3: not directly (top-most custom circuit) or indirectly
              (inside of another custom circuit) within a custom circuit.
134         foreach (PreviewStructure previewStructure in
              MenuSetupManager.Instance.PreviewStructures)
135         {
136             if (previewStructure == _previewStructure) continue;
137
138             if (instance.invalidCircuits.Contains(previewStructure))
                  { errorMessages.Add("Circuit is directly and/or indirectly
                  referenced by another custom circuit"); break; }
139         }
140
141         // Returns the list of all obtained error messages, if any.
142         return errorMessages;
143     }
144
145     /// <summary>
146     /// Sorts the current preview structure as invalid (contains the
          target circuit) or valid by running a modified depth-first
          search.<br/><br/>
147     /// Essentially, the user inputs a starting preview structure. Until a
           starting circuit (e.g. AND) is reached or there is nothing else to
          explore, this method:<br/>
148     /// - Recursively calls this method on all of its circuit components
          that are custom circuits granted they have not already been sorted
          into valid or invalid circuits.
```

```
149     /// - Adds the current circuit as an invalid preview structure if any of its children contained the target custom circuit, and returns.<br/>
150     /// - Adds the current circuit as a valid preview structure if none of its child custom circuits contained the target custom circuit.
151     /// </summary>
152     /// <param name="current"></param>
153     /// <param name="target"></param>
154     private void TraversalTest(PreviewStructure current, PreviewStructure target)
155     {
156         // If the target preview structure was recursively called by TraversalTest, then add it.
157         if (current == target) { if (!invalidCircuits.Contains(current)) { invalidCircuits.Add(current); } return; }
158 
159         // If already designated as an invalid/valid circuit, return.
160         else if (invalidCircuits.Contains(current) || validCircuits.Contains(current)) return;
161 
162         // Traverse through each custom circuit and recursively call this method.
163         // Once done, check to see if the target preview structure was detected. If so, then also add this preview structure.
164         foreach (CircuitIdentifier circuitIdentifier in current.Circuits)
165         {
166             if (circuitIdentifier.previewStructureID != -1)
167             {
168                 PreviewStructure previewStructure = MenuSetupManager.Instance.PreviewStructures[MenuSetupManager.Instance.PreviewStructureIDs.IndexOf(circuitIdentifier.previewStructureID)];
169 
170                 TraversalTest(previewStructure, target);
171 
172                 if (invalidCircuits.Contains(previewStructure)) { invalidCircuits.Add(current); return; }
173             }
174         }
175 
176         // If not invalid, then must be valid.
177         validCircuits.Add(current);
178     }
179 
180     // Getter methods
181     public static MenuLogicManager Instance { get { return instance; } }
182 
183     public bool FirstOpen { get { return firstOpen; } }
184 
```

```
185     public int CurrentSceneIndex { get { return currentSceneIndex; } }
186
187     public PreviewStructure CurrentPreviewStructure { get { return
          currentPreviewStructure; } }
188 }
```