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

/// <summary>
/// PreviewManager deserializes the current preview structure within the preview scene.
/// </summary>
public class PreviewManager : MonoBehaviour
{
    // Singleton state reference
    private static PreviewManager instance;

    /// <summary>
    /// Material used for empty inputs and outputs respectively.
    /// </summary>
    [SerializeField]
    Material inputMaterial,
        outputMaterial;

    /// <summary>
    /// Displays the current input or output label being hovered on, if any.
    /// </summary>
    [SerializeField]
    TextMeshProUGUI nameText;

    /// <summary>
    /// List of instantiated circuits in the scene.
    /// </summary>
    private List<Circuit> circuits = new List<Circuit>();

    /// <summary>
    /// List of all inputs from circuits in the scene.
    /// </summary>
    private List<Circuit.Input> inputs = new List<Circuit.Input>();

    /// <summary>
    /// List of all outputs from circuits in the scene.
    /// </summary>
    private List<Circuit.Output> outputs = new List<Circuit.Output>();

    /// <summary>
    /// The preview structure to deserialize and load into the preview scene.
    /// </summary>
    private PreviewStructure previewStructure;

```

```
47      // Enforces a singleton state pattern
48      private void Awake()
49      {
50          if (instance != null)
51          {
52              Destroy(this);
53              throw new Exception("PreviewManager instance already
                  established; terminating.");
54          }
55
56          instance = this;
57      }
58
59      // Begins the deserialization process
60      private void Start()
61      {
62          CursorManager.SetMouseTexture(true);
63          previewStructure =
                MenuLogicManager.Instance.CurrentPreviewStructure;
64          nameText.text = previewStructure.Name;
65          Deserialize();
66          UpdateIOMaterials();
67      }
68
69      /// <summary>
70      /// Deserializes the current preview structure.<br/><br/>
71      /// The restored values include the circuits, connections, and camera
            position.
72      /// </summary>
73      private void Deserialize()
74      {
75          foreach (CircuitIdentifier circuitIdentifier in
                previewStructure.Circuits) circuits.Add
                (CircuitIdentifier.RestoreCircuit(circuitIdentifier));
76
77          MenuSetupManager.Instance.RestoreConnections(previewStructure);
78          CameraMovementPreview.Instance.PlayerCamera.transform.position =
                previewStructure.CameraLocation;
79      }
80
81      /// <summary>
82      /// Iterates through each circuit in the scene to change the material
            of empty inputs and outputs.
83      /// </summary>
84      private void UpdateIOMaterials()
85      {
86          int inputIndex = 0, outputIndex = 0;
87
88          foreach (Circuit circuit in circuits)
```

```
89          {
90              foreach (Circuit.Input input in circuit.Inputs)
91              {
92                  // If this current input exists in InputOrders, it is
                      guaranteed to be an empty input.
93                  if (previewStructure.InputOrders[inputIndex] != -1)
                      input.Transform.GetComponent<MeshRenderer>().material =
                      inputMaterial;
94
95                  inputs.Add(input);
96                  inputIndex++;
97              }
98
99              foreach (Circuit.Output output in circuit.Outputs)
100             {
101                 // If this current output exists in OutputOrders, it is
                      guaranteed to be an empty output.
102                 if (previewStructure.OutputOrders[outputIndex] != -1)
                      output.Transform.GetComponent<MeshRenderer>().material =
                       outputMaterial;
103
104                 outputs.Add(output);
105                 outputIndex++;
106             }
107         }
108     }
109
110     // Getter methods
111     public static PreviewManager Instance { get { return instance; } }
112
113     public List<Circuit> Circuits { get { return circuits; } }
114
115     public List<Circuit.Input> Inputs { get { return inputs; } }
116
117     public List<Circuit.Output> Outputs { get { return outputs; } }
118
119     public Material InputMaterial { get { return inputMaterial; } }
120
121     public Material OutputMaterial { get { return outputMaterial; } }
122 }
```