Project Document

This Unity project is designed to facilitate the management of UI elements and their hierarchies through JSON data. It consists of two main classes (UIElement and JsonCreator) and an editor class (JsonInspector). Below is a detailed breakdown of each component:

1. UIElement Class

This class represents a UI element with various properties and methods to create a new instance of UIElement.

```
public string name;
public Vector3 position;
public Vector3 rotation;
public Vector3 scale;
public Color color;
public List<string> componentNames = new List<string>();
public List<UIElement> components = new List<UIElement>();
```

Methods

```
1 reference
public static UIElement CreateNew(string name, Vector3 position, Vector3 rotation, Vector3 scale, Color color)
{
    UIElement newElement = new UIElement
    {
        name = name,
        position = position,
        rotation = rotation,
        scale = scale,
        color = color
    };
    return newElement;
}
```

2. UIElementList Class

This class holds a list of UIElement objects.

```
public class UIElementList
{
    public List<UIElement> elements;
}
```

3. JsonCreator Class

This class extends MonoBehaviour and contains methods to generate JSON data from the UI hierarchy and save it to a file.

```
public string jsonFilePath = "Assets/UIHierarchy.json";
private UIElement rootElement;
```

Methods Start:

Initializes the root element and triggers the JSON data generation and saving process.

```
Unity Message | 0 references
void Start()
{
    rootElement = new UIElement();
    GenerateJSONData(transform, rootElement);
    string jsonData = JsonUtility.ToJson(rootElement, true);
    SaveJSON(jsonData);
}
```

GenerateJSONData: Recursively generates JSON data from the UI hierarchy.

```
2 references
void GenerateJSONData(Transform currentTransform, UIElement currentElement)
{
    currentElement.name = currentTransform.name;
    currentElement.position = currentTransform.localPosition;
    currentElement.rotation = currentTransform.localEulerAngles;
    currentElement.scale = currentTransform.localScale;
    currentElement.color = Color.white;

    Component[] gameComponents = currentTransform.GetComponents<Component>();
    foreach (Component comp in gameComponents)
    {
        currentElement.componentNames.Add(comp.GetType().Name);
    }

    foreach (Transform child in currentTransform)
    {
        UIElement childElement = new UIElement();
        GenerateJSONData(child, childElement);
        currentElement.components.Add(childElement);
    }
}
```

SaveJSON: Saves the generated JSON data to a file.

```
1 reference
void SaveJSON(string jsonData)
{
    File.WriteAllText(jsonFilePath, jsonData);
    Debug.Log("JSON data saved to " + jsonFilePath);
}
```

4. JsonInspector Class

This editor class provides a custom inspector for the JsonCreator class, allowing for loading, editing, and saving JSON data, as well as creating and instantiating templates through the Unity Editor.

```
private string jsonFilePath = "Assets/UIHierarchy.json";
private string jsonData = "{\"elements\":[]}";
private Vector2 scrollPosition;

private UIElementList templateList;
private List<UIElementList> template;
private int selectedTemplateIndex = 0;

// Fields for creating a new template
private string templateName = "New Template";
private Vector3 templatePosition = Vector3.zero;
private Vector3 templateRotation = Vector3.zero;
private Vector3 templateScale = Vector3.one;
private Color templateColor = Color.white;
```

Methods

OnInspectorGUI: Defines the custom inspector UI and handles user interactions.

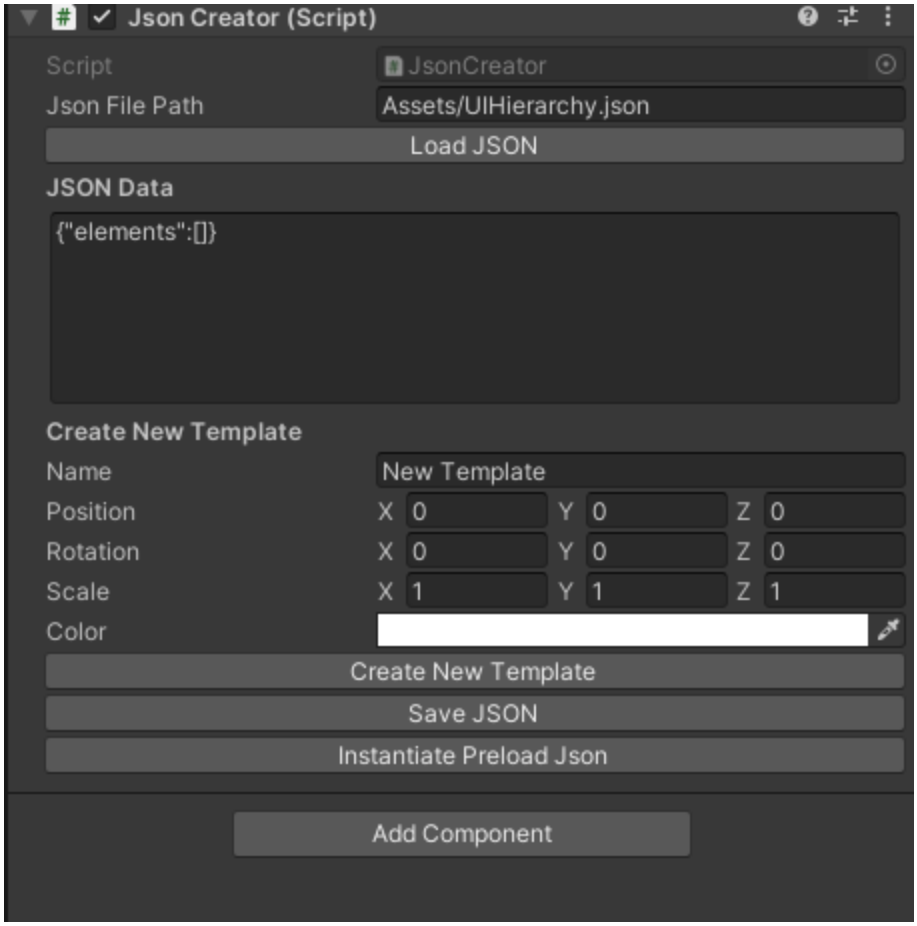
LoadJSON: Loads JSON data from a file.

SaveJSON: Saves JSON data to a file.

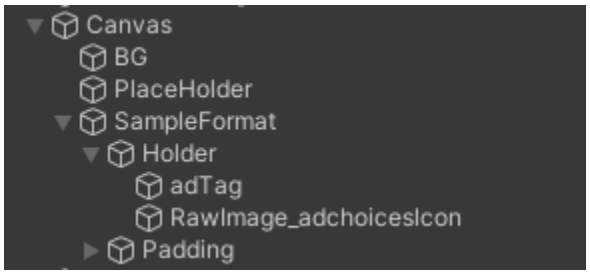
CreateNewTemplate: Creates a new template and adds it to the JSON data.

InstantiateTemplate: Instantiates a template in the Unity Editor.

CreateUIHierarchy: Recursively creates UI elements from a template.



Once you start the game , It will Automatically create a json file format on a given path based on gameobject hierarchy.



LoadJson Button : It will load a json file and show what properties and fields it contains inside the json data field , also u can edit the json properties from inspector and save it by using Save JSON Button.

Create NewTemplate : You can create a new template by rename the template name and save it on the path.

Instantiate Preload Json: you can instantiate the preload json gameobject and also you can instantiate your new template also.

Error Handling…

```
void SaveJSON(string jsonData)
{
    try
    {
        File.WriteAllText(jsonFilePath, jsonData);
        Debug.Log("JSON data saved to " + jsonFilePath);
    }
    catch (Exception e)
    {
        Debug.LogError("An error occurred while saving JSON data: " + e.Message);
    }
}
```

The method contains a try-catch block to handle any exceptions that might occur during the file writing process: Try Block: Attempts to write the JSON data to a file at the specified path. If successful, a message confirming the successful saving of the data is logged to the console.

Catch Block: If an exception occurs (for instance, due to an invalid file path or permission issues), the catch block is executed. It catches the exception and logs an error message to the console, which includes the message from the exception to help diagnose the issue.