Snog's Interaction System

Welcome to the documentation for the Snog Interaction System, a no-code, modular framework for creating complex interactions in Unity.

This system is designed to be data-driven, allowing you to create, edit, and manage all of your game's interactions through a user-friendly editor window without writing a single line of code.

Table of Contents

1. Core Concepts
2. Initial Setup Guide
3. Creating Your First Interaction
4. Editing and Deleting Interactions
5. Writing Custom Interaction Logic
6. FAQ

Core Concepts

The system is built on a few key components that work together:

- **Interactor**: A component you place on your player. It's responsible for detecting interactable objects in the world using a raycast.
- **InteractibleObj**: A component you place on any object you want the player to interact with (doors, items, NPCs, etc.). It links the object to a specific `InteractionType`.
- **Interaction Creator Window**: The heart of the system. This editor window is where you create, edit, and delete all your interactions without writing code.
- **InteractionType**: A ScriptableObject that bundles together all the data for a single interaction: its name, its behavior script, and its UI prompt.
- **Interaction Registry**: A ScriptableObject that acts as a central database for all `InteractionType` assets in your project.

Initial Setup Guide

1. The Interaction Registry

The system needs an `InteractionRegistry` asset to store all your interaction types.

- Go to **Tools > Interaction Creator**.
- If no registry is found, the window will show a "Create New Registry" button. Click it.
- This will automatically create and assign the registry for you.

2. Player Setup

Your player character needs the `Interactor` component to see interactable objects.

- Select your player GameObject.
- Add the `Interactor` component.
- **Interactor Source**: Assign a Transform that marks where the detection raycast should start (e.g., the player's camera).
- **Interact Range**: Set how far the player can interact from.
- **Interactable Layer**: Choose a layer that all your interactable objects will be on (e.g., "Interactable").

3. UI Setup

The system needs a UI Text element to display prompts like "Press E to open".

- Create a **UI > Text - TextMeshPro** element on your Canvas.
- Select this new Text GameObject.
- Add the `InteractionText` component to it.
- Drag the TextMeshPro component into the **Text Appear** field on the `InteractionText` component.
- By default, the text should be disabled. The system will enable it when needed.

Creating Your First Interaction

1. Open the editor window via **Tools > Interaction Creator**.
2. In the **Create New Interaction** section, fill in the fields:
    - **Interaction Name**: `ChangeColor` (no spaces or special characters).
    - **Prompt Text**: `Press {key} to use`. The `{key}` tag will be automatically replaced by the key you choose.
    - **Interaction Key**: Choose the key, for example, `E`.
3. Click **Create Interaction**.

The tool will automatically generate three new assets in `Assets/Snog/InteractionSystem/Generated/`:

- A C# script: `ChangeColorInteraction.cs`
- A prompt asset: `ChangeColorPrompt.asset`
- An interaction type: `ChangeColor.asset`

The tool will also highlight the new C# script for you, ready to be edited.

Editing and Deleting Interactions

As your project grows, you can manage all your interactions from the same window.

- **To Edit**: Select an interaction from the **Select Interaction** dropdown. Its properties will appear, and you can change the prompt text or key. Click "Update Interaction" to save.
- **To Delete**: Select an interaction from the dropdown and click the "Delete Selected Interaction" button.

Writing Custom Interaction Logic

The system creates the template code for you, all you need to do is add the logic you want.

1. After creating the `ChangeColor` interaction, open the `ChangeColorInteraction.cs` script.
2. You will see an `Execute` method. This is where your game logic goes.

Example: Change object's color

C#

```
using UnityEngine;
using Snog.InteractionSystem.Core.Interfaces;

namespace Snog.InteractionSystem.Behaviors
{
    public class ChangeColorInteraction : MonoBehaviour,
IInteractionBehavior
    {
        public void Execute(GameObject target)
        {
            // Try to get the MeshRenderer component from the
interacted object.
            var meshRenderer = target.GetComponent<MeshRenderer>();
            if (meshRenderer != null)
            {
                Debug.LogWarning($"'{target.name}' has no
MeshRenderer component.");
                return;
            }

            // Generate a new random color.
            Color randomColor = new Color
            (
              Random.value, // Red channel
              Random.value, // Green channel
              Random.value  // Blue channel
            );

            // Apply color to the renderer
            meshRenderer.material.color = randomColor;
            Debug.Log($"Changed the color of {target.name} to
{randomColor}");
        }
    }
}
```

FAQ

- **Pressing the key does nothing?**
  - On the `InteractibleObj` component, double-check that the **Interaction Type Name** string exactly matches the name you gave it in the creator window (e.g., "ChangeColor"). It is case-sensitive.