Unity Level Editor Tool Documentation

Introduction

The Level Editor Tool is a Unity custom editor tool designed to make level creation intuitive and flexible. With it, you can manage various level components, including environment objects, particle effects, sound effects, and word-based game elements directly from within the Unity Editor. This tool leverages the **Model-View-Controller (MVC)** pattern and **Single Responsibility Principle (SRP)** to ensure maintainability, modularity, and ease of use.

---

1. Setting Up the Level Editor Tool

Prerequisites

1. **Unity 2021.1 or later**: The tool is built to work seamlessly with recent Unity versions.
2. **TextMeshPro**: The tool uses TextMeshPro for displaying text on UI elements. Ensure this package is installed via Unity's Package Manager.

Initial Setup Steps

1. **Import the Tool**: Add the Level Editor Tool scripts and assets into your Unity project.
2. **Organize Assets**: Place the tool's assets (such as `LevelData` and `EnvironmentData` `ScriptableObject`s) in appropriate folders. It's recommended to keep them in a structured folder hierarchy for easy access.

Creating Essential Assets

1. **Create LevelData Assets**: Right-click in your Project window, select **Create > Level Data** to create a new `LevelData` asset. This asset will hold all the information for a specific level, including environment setup, words, particle effects, and sounds.
2. **Create EnvironmentData Assets**: Similarly, create environment assets by selecting **Create > Environment Data**. Each `EnvironmentData` asset represents a type of environment object that you can add to your levels.

---

2. Using the Level Editor Tool

Opening the Level Editor Window

1. Go to **Tools > Level Editor Plugin > Level Editor** in the Unity menu bar.
2. The `LevelEditorWindow` will open, displaying all tools and options for customizing levels.

Creating and Selecting Levels

- **Create New Level**: Click on the **"Create New Level"** button at the top of the editor window. Specify a save location for the new `LevelData` asset. This asset will automatically load into the editor for customization.
- **Select Existing Level**: Use the **Level Data** dropdown to select an existing `LevelData` asset. This will load the selected level's configuration into the editor for modification.

---

3. Editor Tool Layout and Usage

Level Settings

- **Level Name**: Enter a name for the level, which helps in identifying it in the Project view.
- **Background Image**: Select a sprite as the background image for the level. Click **Apply Background** to preview the selected background in the Scene view.
- **Question Text**: Type the main question or prompt for the level. This text will appear on the UI during gameplay.
- **Animated Scene Prefab**: Select a prefab for an animated scene that will play upon level completion. This could be any animation or effect to signify the end of the level.
- **Success Animations**: Add animation clips that will play when the player successfully completes the level.

Manage Words

- **Add New Word**: Add words to the level's word list. These words will appear as clickable buttons in the UI.
- **Add Correct Word**: Designate specific words as correct answers. These words trigger specific actions when selected during gameplay.
- **Remove Words**: Use the **Delete** buttons to remove unwanted words or correct words.

Environment Management

This section allows you to add, replace, or delete environment objects from the level.

- **Add Environment Object**: Each environment object defined in `EnvironmentData` will have a button here. Clicking a button will instantiate the corresponding object in the scene.
- **Replace Selected Object**: Select an environment object in the Scene view and replace it with another object from the list.
- **Delete Selected Object**: Deletes the currently selected environment object from the scene.
- **Delete All Environments**: Removes all environment objects from the scene.

Particle System Prefabs

This section allows you to define particle effects for visual enhancements.

- **Add Particle Prefab**: You can add particle effect prefabs (e.g., smoke, fire, spark) to the level by selecting prefabs.
- **Remove Particle Prefab**: Remove specific particle prefabs if they are no longer needed.

Sound Effects

Manage audio clips for different gameplay events.

- **Add Sound Effect**: Assign sound effects for various level actions. For example, you can specify a sound effect for correct or incorrect answers.
- **Remove Sound Effect**: Remove sound effects by clicking the **Remove** button next to the clip.

Preview and Save

- **Preview/Update Level**: Click this button to update and preview the level configuration in real time in the Unity Editor. It will instantiate words, set the question text, and load the animated scene (if any) for immediate feedback.
- **Save Level Settings**: This button saves all changes made in the editor to the `LevelData` asset. Use this button frequently to ensure your modifications are stored persistently.

---

4. Script Descriptions

Here's a breakdown of each script in the tool and its specific purpose.

LevelEditorWindow.cs

- **Purpose**: The main editor window that organizes and displays all components of the Level Editor Tool.
- **Functionality**:
    - Provides UI for creating or selecting levels.
    - Displays sections for level settings, environment management, particles, and sounds.
    - Integrates with `UIManager`, `EnvironmentUIController`, and `LevelPreview` to manage various parts of the editor.

UIManager.cs

- **Purpose**: Manages the UI elements and fields for level settings, words, animations, particles, and sounds.
- **Functionality**:
    - Draws fields for editing basic level settings like name, background, question text, and animations.
    - Provides functionality for managing word lists and correct words.
    - Manages particle and sound effect entries, allowing users to add or remove them.

EnvironmentModel.cs

- **Purpose**: Acts as a data container for environment objects, ensuring they are organized within a designated parent object in the scene.
- **Functionality**:
    - Manages the environment container in the scene, where all environment objects are stored.
    - Stores and retrieves environment objects, positions, and rotations, ensuring consistent setup across levels.

EnvironmentManager.cs

- **Purpose**: Handles the logic for adding, replacing, and deleting environment objects in the scene.
- **Functionality**:
    - Instantiates new environment objects and places them within the environment container.
    - Replaces an existing environment object with another type while maintaining the same position and rotation.
    - Deletes specific or all environment objects as per user action.

EnvironmentUIController.cs

- **Purpose**: Provides the UI for managing environment objects within the LevelEditorWindow.
- **Functionality**:
  - Draws buttons for each environment action (add, replace, delete).
  - Communicates with `EnvironmentManager` to apply actions based on user inputs in the editor window.

LevelPreview.cs

- **Purpose**: Allows for real-time preview of level configurations within the Unity editor.
- **Functionality**:
  - Updates the scene with the current level settings, displaying question text and instantiating words.
  - Instantiates the animated scene preview (if assigned) to give visual feedback on how the level will look.

LevelData.cs

- **Purpose**: Stores all level-specific data, such as settings, environment configurations, words, particles, and sounds.
- **Functionality**:
  - Holds information like level name, background, word lists, correct words, particle effects, and sound effects.
  - Acts as a `ScriptableObject` asset that can be saved and reloaded to maintain level configurations.

EnvironmentData.cs

- **Purpose**: Defines each environment object available in the editor with attributes like name, prefab, and type.
- **Functionality**:
  - Stores the prefab reference and metadata for each environment object.
  - Used in the environment section of the editor to allow users to add these objects to levels.

Summary

The Level Editor Tool provides an organized, intuitive way to create and customize levels in Unity. By following the steps outlined above, you can create levels with customized backgrounds, environment objects, word lists, particle effects, and sounds. Each component adheres to the **MVC pattern** for ease of use, with clear separation between data, logic, and UI.

Key Points to Remember

- **Save** your changes often by clicking **Save Level Settings**.
- Use **Preview/Update Level** to verify how your setup will look in-game.
- Keep environment objects organized by using the **Environment Management** section effectively.
- **Customize words** and **correct words** to control game behavior and player interaction.
- **Add animations and sounds** to enhance the player experience.

With this tool, level creation is streamlined, making it easy to design and test game levels directly in the Unity Editor.

○