AdvancedPlayerPrefs Documentation

Introduction

The AdvancedPlayerPrefs package provides an extended functionality for Unity's built-in PlayerPrefs system. It allows you to store and retrieve various data types such as bools, strings, colors, integers, floats, vectors, quaternions, and even Unity Transforms persistently across game sessions.

Namespace

The AdvancedPlayerPrefs package is contained within the `GreenLeaf` namespace.

Usage

To use AdvancedPlayerPrefs in your Unity project, follow these steps:

1. Import the AdvancedPlayerPrefs namespace:
   csharp

   Copy code

   ```
   using GreenLeaf;
   ```

2. Use the provided static methods to set and get data:
   csharp

   Copy code

   ```
   // Example: Setting and getting a bool AdvancedPlayerPrefs.SetBool("isSoundOn", true); bool soundOn = AdvancedPlayerPrefs.GetBool("isSoundOn", false);
   ```

Supported Data Types

- Bool
  - `SetBool(string key, bool value)`
  - `GetBool(string key, bool defaultValue)`
- String
  - `SetString(string key, string value)`
  - `GetString(string key, string defaultValue)`
- Color
  - `SetColor(string key, Color value)`
  - `GetColor(string key, Color defaultValue)`
- Int
  - `SetInt(string key, int value)`
  - `GetInt(string key, int defaultValue)`
- Float
  - `SetFloat(string key, float value)`
  - `GetFloat(string key, float defaultValue)`
- Vector2
  - `SetVector2(string key, Vector2 value)`

    - `GetVector2(string key, Vector2 defaultValue)`
- Vector3
    - `SetVector3(string key, Vector3 value)`
    - `GetVector3(string key, Vector3 defaultValue)`
- Vector4
    - `SetVector4(string key, Vector4 value)`
    - `GetVector4(string key, Vector4 defaultValue)`
- Quaternion
    - `SetQuaternion(string key, Quaternion value)`
    - `GetQuaternion(string key, Quaternion defaultValue)`
- Transform
    - `SetTransform(string key, Transform value)`
    - `GetTransform(string key, Transform defaultValue)`

Additional Features

- Deleting Data
    - `DeleteKey(string key)`: Deletes the specified key and its associated data.
    - `DeleteAll()`: Deletes all keys and their associated data.
- Checking Key Existence
    - `HasKey(string key)`: Checks if a key exists.
- Saving Changes
    - `Save()`: Saves all modified PlayerPrefs to disk.

Note

- Make sure to save changes using `Save()` after modifying PlayerPrefs to persist the changes.

Example

csharp
Copy code
```
using UnityEngine; using GreenLeaf; public class ExampleUsage : MonoBehaviour { void Start() { // Set and get a bool
AdvancedPlayerPrefs.SetBool("isSoundOn", true); bool soundOn = AdvancedPlayerPrefs.GetBool("isSoundOn", false);
// Set and get a string AdvancedPlayerPrefs.SetString("playerName", "John"); string playerName =
AdvancedPlayerPrefs.GetString("playerName", "Unknown"); // Set and get a color Color playerColor = Color.blue;
AdvancedPlayerPrefs.SetColor("playerColor", playerColor); Color savedColor =
AdvancedPlayerPrefs.GetColor("playerColor", Color.white); // Other data types follow similar patterns... } }
```

Conclusion

With AdvancedPlayerPrefs, you can conveniently store and retrieve various data types in Unity with enhanced functionality compared to Unity's built-in PlayerPrefs system.