Unity Hierarchy

* SettingsMenu (Parent Canvas/Prefab) <SettingsManager>
  + Darkness (Background Image)
  + Prompt (Paused Text)
  + Back (Button)
    - Text
  + Music Volume (Volume Slider)
    - Background (Empty Slider Image)
    - Fill Area (Area Inside The Slider)
      * Fill (Image That Adjusts To Fill Slider)
    - Handle Slide Area (Area On Slider For Handle)
      * Handle (Interactable Image)
        + Icon (Overlay Image For Style)
    - Music Volume Text (Changing % Value)
    - Music Text ("Music" before % value)
  + SFX Volume (Same As Music)
    - Background
    - Fill Area
    - Handle Slide Area
    - Music Volume Text
    - Music Text
  + BCMode (Toggle For Invincibility) <BCModeSaver>
    - Background (Empty Checkbox)
      * Checkmark
    - Label (Text)

SettingsManager.cs

using UnityEngine;

using TMPro;

using UnityEngine.UI;

public class SettingsManager : MonoBehaviour

{

private SliderBase musicVolumeSlider; // Slider superclass for music volume

private SliderBase sfxVolumeSlider; // Slider superclass for sound effects volume

private Slider musicSlider; //Actual unity slider object

private Slider sfxSlider; //Actual unity slider object

private void Awake()

{

// Locate the text components in the scene for volume displays

TextMeshProUGUI musicVolumeText = GameObject.Find("MusicVolumeText").gameObject.GetComponent<TextMeshProUGUI>();

TextMeshProUGUI sfxVolumeText = GameObject.Find("SFXVolumeText").gameObject.GetComponent<TextMeshProUGUI>();

//Using text components get the slider components

musicSlider = musicVolumeText.transform.parent.GetComponent<Slider>();

sfxSlider = sfxVolumeText.transform.parent.GetComponent<Slider>();

//DYNAMIC BINDING

// Initialize the volume sliders with the found text components

musicVolumeSlider = new MusicVolumeSlider(musicVolumeText);

sfxVolumeSlider = new SFXVolumeSlider(sfxVolumeText);

// Load saved settings for both volume sliders

musicVolumeSlider.LoadSetting();

sfxVolumeSlider.LoadSetting();

}

//Sets the actual Unity slider values

private void OnEnable()

{

musicSlider.value = musicVolumeSlider.Value;

sfxSlider.value = sfxVolumeSlider.Value;

}

// Method to set the music volume and save the setting

public void SetMusicVolume(float value)

{

//Sets value in in the slider subclass

musicVolumeSlider.Value = value;

//Saves new value

musicVolumeSlider.SaveSetting();

}

// Method to set the sound effects volume and save the setting

public void SetSFXVolume(float value)

{

sfxVolumeSlider.Value = value;

sfxVolumeSlider.SaveSetting();

}

// Getter for the current music volume

public float GetMusicVolume() => musicVolumeSlider.Value;

// Getter for the current sound effects volume

public float GetSFXVolume() => sfxVolumeSlider.Value;

}

BCModeSaver.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class BCModeSaver : MonoBehaviour

{

public bool BCMode = false;

[SerializeField] Toggle BCModeToggle;

public void loadBCMode()

{

//loads player pref and sets box to match

BCMode = PlayerPrefs.GetInt("BCMode", 0) == 1;

BCModeToggle.isOn = BCMode;

}

public void setBCMode()

{

//saves to player prefs and switches box

BCMode = !BCMode;

PlayerPrefs.SetInt("BCMode", BCMode ? 1 : 0);

}

}

Sliders.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

// Base class for slider implementations

public class SliderBase

{

// Protected field to store the slider's current value

protected float value;

// Property for accessing and modifying the slider value

public float Value

{

get => value; // Getter for the value

set

{

this.value = value; // Set the value

OnValueChanged(value); // Notify derived classes of the change

}

}

// Virtual method to handle actions when the value changes

protected virtual void OnValueChanged(float value)

{

Debug.Log("Value changed to: " + value);

}

// Virtual methods for loading and saving slider settings

public virtual void LoadSetting()

{

value = 0f;

Debug.Log("LoadSetting called in base class");

}

public virtual void SaveSetting()

{

Debug.Log("SaveSetting called in base class");

}

}

MusicVolumeSlider.cs

using UnityEngine;

using TMPro;

public class MusicVolumeSlider : SliderBase

{

private void Start()

{

// Initialize the music source volume and update the volume display on start

AudioManager.Instance.musicSource.volume = value;

UpdateVolumeText(value);

}

// Text component to display the volume value

private TextMeshProUGUI volumeText;

// Set the volume text display

public MusicVolumeSlider(TextMeshProUGUI volumeText)

{

this.volumeText = volumeText;

}

// Handle volume changes when the slider value is updated

protected override void OnValueChanged(float value)

{

// Update the audio source volume based on the slider value

AudioManager.Instance.musicSource.volume = value/100;

// Change the volume text display

UpdateVolumeText(value);

}

// Update the displayed volume percentage

private void UpdateVolumeText(float value)

{

volumeText.text = "Volume: " + $"{Mathf.RoundToInt(value)}" + "%"; // Display as a percentage (0–100)

}

// Save the current volume setting to PlayerPrefs

public override void SaveSetting()

{

PlayerPrefs.SetFloat("MusicVolume", value);

}

// Load the volume setting from PlayerPrefs

public override void LoadSetting()

{

value = PlayerPrefs.GetFloat("MusicVolume", 1.0f); // Default to 1.0 if no setting exists

OnValueChanged(value);

UpdateVolumeText(value); // Initialize text display with loaded value

}

}

MusicVolumeSlider.cs

using UnityEngine;

using TMPro;

public class SFXVolumeSlider : SliderBase

{

// Text component to display the volume value

private TextMeshProUGUI volumeText;

// Set the volume text display

public SFXVolumeSlider(TextMeshProUGUI volumeText)

{

this.volumeText = volumeText;

}

// Handle volume changes when the slider value is updated

protected override void OnValueChanged(float value)

{

// Update the audio source volume for sound effects

AudioManager.Instance.fxSource.volume = value/100;

// Refresh the volume text display

UpdateVolumeText(value);

}

// Update the displayed volume percentage

private void UpdateVolumeText(float value)

{

volumeText.text = "Volume: " + $"{Mathf.RoundToInt(value)}" + "%"; // Display as a percentage (0–100)

}

// Save the current SFX volume setting to PlayerPrefs

public override void SaveSetting()

{

PlayerPrefs.SetFloat("SFXVolume", value);

}

// Load the SFX volume setting from PlayerPrefs

public override void LoadSetting()

{

value = PlayerPrefs.GetFloat("SFXVolume", 1.0f); // Default to 1.0 if no setting exists

UpdateVolumeText(value); // Initialize text display with loaded value

}

}