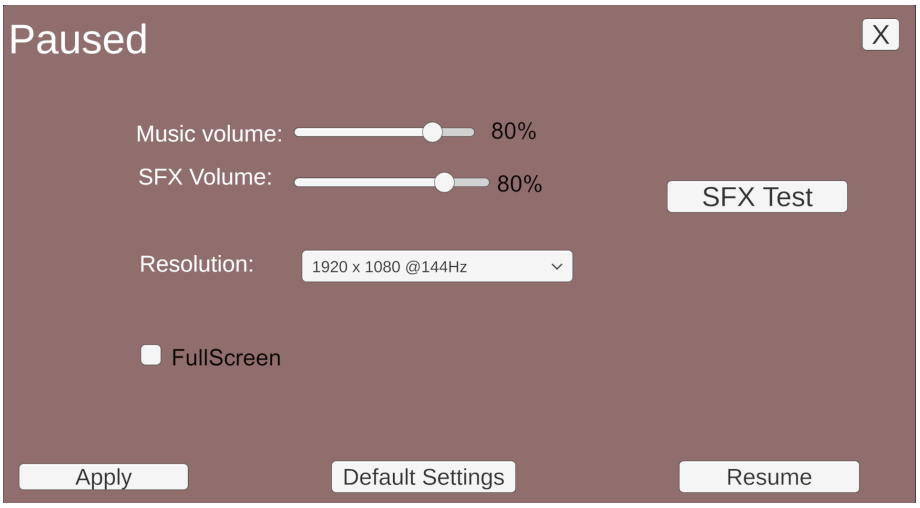
1) Objetivo de la actividad

Desarrollar un menú de Pausa real en Unity que permita modificar ajustes de audio y resolución de juego, aplicando buenas prácticas de UX, persistencia y diseño responsive.
El menú debe abrirse en juego (ESC), animarse, detectar cambios sin guardar y gestionar sonido UI.

2) Funcionalidades implementadas

Funcionalidad	Estado
Pausa/Resume con ESC	OK
Fade animación con CanvasGroup	OK
Control volumen música y sfx con sliders	OK
Botón test SFX y Toggle Music Pause	OK
Guardado persistente con PlayerPrefs	OK
Dropdown resolución TMP y fullscreen	OK
Modal Unsaved Changes Yes / No	OK
SFX click en todos los botones	OK
Responsive UI adaptable 16:9 / 16:10	OK

INSERTAR CAPTURA MENU PAUSA

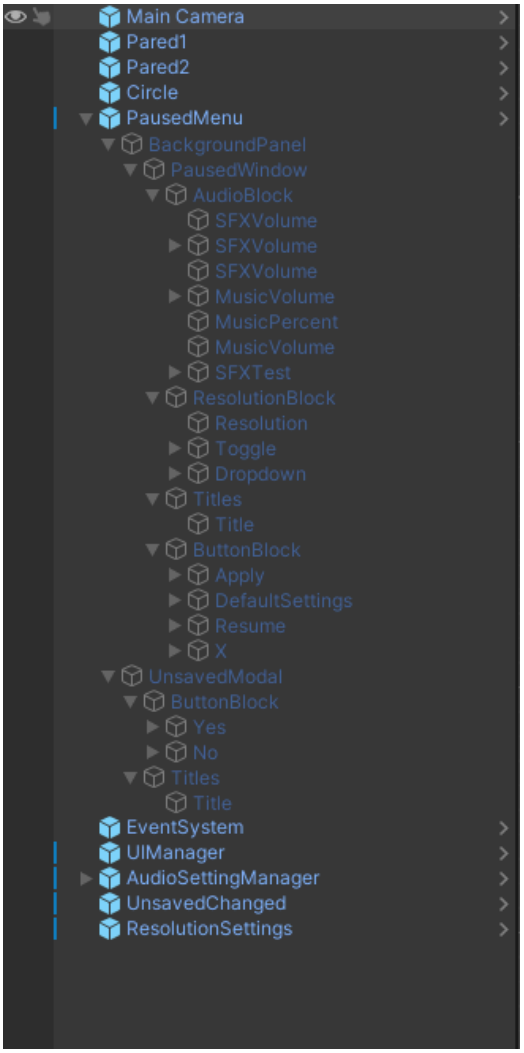


3) Diseño UI Responsive

- Anchors del PauseWindow centrados 0.2 / 0.8 para adaptarse a resoluciones distintas.

- Uso de VerticalLayoutGroup + ContentSizeFitter para distribuir bloques.
- Tres bloques principales: AudioBlock, ResolutionBlock, ButtonsBlock.
- Layout Element en cada bloque para mantener alturas consistentes.

INSERTAR CAPTURA UNITY HIERARCHY / BLOQUES



4) Scripts principales (explicación + código clave)

4.1 PauseMenu.cs

Responsable de abrir/cerrar el menú y pausar el juego físicamente.

```
4 references
public void TogglePause()
{
    if (isFading) return;
    isPaused = !isPaused;
    if (isPaused) StartCoroutine(FadeIn());
    else StartCoroutine(FadeOut());
}
```

`Time.timeScale = 0;` dentro de FadeIn → pausa física real.

INSERTAR CAPTURA SCRIPT

```
public class PauseMenu : MonoBehaviour
{
    public AudioSettingsManager audioManager;

    public GameObject pausePanel; // asignar PausePanel
    public CanvasGroup canvasGroup; // CanvasGroup del pausePanel
    public float fadeDuration = 0.25f;
    bool isPaused = false;
    bool isFading = false;

    void Start()
    {
        pausePanel.SetActive(false);
        canvasGroup.alpha = 0f;
        canvasGroup.interactable = false;
        canvasGroup.blocksRaycasts = false;
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            TogglePause();
        }
    }

    public void TogglePause()
    {
        if (isFading) return;
        isPaused = !isPaused;
        if (isPaused) StartCoroutine(FadeIn());
        else StartCoroutine(FadeOut());
    }

```

```
        1 reference
    IEnumerator FadeIn()
    {
        isFading = true;
        pausePanel.SetActive(true);
        audioManager.RegisterButtonsForPanel(pausePanel);

        float t = 0f;
        while (t < fadeDuration)
        {
            t += Time.unscaledDeltaTime;
            canvasGroup.alpha = Mathf.Lerp(0f, 1f, t / fadeDuration);
            yield return null;
        }
        canvasGroup.alpha = 1f;
        canvasGroup.interactable = true;
        canvasGroup.blocksRaycasts = true;
        Time.timeScale = 0f; // pausa física
        isFading = false;
    }

        1 reference
    IEnumerator FadeOut()
    {
        isFading = true;
        canvasGroup.interactable = false;
        canvasGroup.blocksRaycasts = false;
        float t = 0f;
        while (t < fadeDuration)
        {
            t += Time.unscaledDeltaTime;
            canvasGroup.alpha = Mathf.Lerp(1f, 0f, t / fadeDuration);
            yield return null;
        }
        canvasGroup.alpha = 0f;
        pausePanel.SetActive(false);
        Time.timeScale = 1f; // reanuda física
        isFading = false;
    }

    // Llamar desde botón Resume
        0 references
    public void OnResumeButton()
    {
        if (isPaused) TogglePause();
    }
}
```

4.2 AudioSettingsManager.cs

Gestiona sliders de música y sfx, test buttons, y reproduce SFX UI en todos los botones.

```
1 reference
void OnMusicSliderChanged(float v)
{
    tempMusic = v;
    UpdateMusicVolumeUI(v);
    ApplyMusicVolume(v);
    if (unsavedHandler) unsavedHandler.SetUnsaved(true);
}
```

```
public void ToggleMusicPause()
{
   if (musicSource.isPlaying) musicSource.Pause();
   else musicSource.UnPause();
}
```

```
1 reference
void OnMusicSliderChanged(float v)
{
    tempMusic = v;
    UpdateMusicVolumeUI(v);
    ApplyMusicVolume(v);
    if (unsavedHandler) unsavedHandler.SetUnsaved(true);
}

1 reference
void OnSfxSliderChanged(float v)
{
    tempSfx = v;
    UpdateSfxVolumeUI(v);
    ApplySfxVolume(v);
    if (unsavedHandler) unsavedHandler.SetUnsaved(true);
}

2 references
void UpdateMusicVolumeUI(float v) => musicValueText.text = Mathf.RoundToInt(v * 100) + "%";
2 references
void UpdateSfxVolumeUI(float v) => sfxValueText.text = Mathf.RoundToInt(v * 100) + "%";

2 references
void ApplyMusicVolume(float v) { musicSource.volume = v; }
2 references
void ApplySfxVolume(float v) { sfxSource.volume = v; }

2 references
void ApplyAudioVolumes(float music, float sfx)
{
    ApplyMusicVolume(music);
    ApplySfxVolume(sfx);
}

1 reference
public void PlaySfxTest()
{
    sfxSource.PlayOneShot(sfxSource.clip);
}
```


SFX Test

4.3 ResolutionSettings.cs

Rellena dropdown TMP con resoluciones reales y aplica fullscreen.

```
107         public void OnApply()
108         {
109             var r = uniqueResolutions[selectedIndex];
110             PlayerPrefs.SetInt(RES_W_PREF, r.width);
111             PlayerPrefs.SetInt(RES_H_PREF, r.height);
112             PlayerPrefs.SetInt(RES_FULLSCREEN, Screen.fullScreen ? 1 : 0);
113             PlayerPrefs.Save();
114             if (unsavedHandler != null) unsavedHandler.SetUnsaved(false);
115         }
```

4.4 UnsavedChangesHandler.cs

Si hay cambios sin guardar → bloquea cierre y muestra modal Yes/No

```
29      public void TryClosePause()
30      {
31          if (hasUnsavedChanges)
32          {
33              unsavedModal.SetActive(true);
34          }
35          else
36          {
37              pauseMenu.TogglePause();
38          }
39      }
```

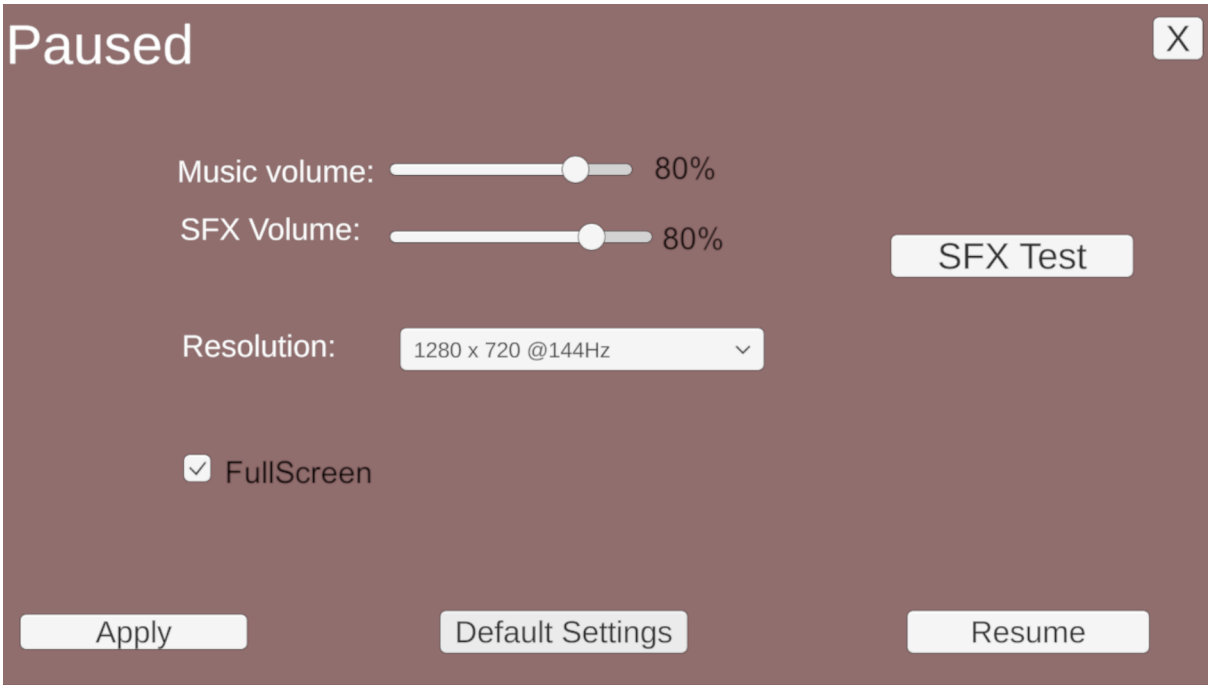
5) Persistencia y UX

- Solo al pulsar **Apply** se guardan cambios.
- Uso de PlayerPrefs para que se mantengan entre sesiones.
- UX real: el usuario controla cuándo confirma cambios.
- Modal evita perdida accidental de configuración (UX profesional real de videojuegos).

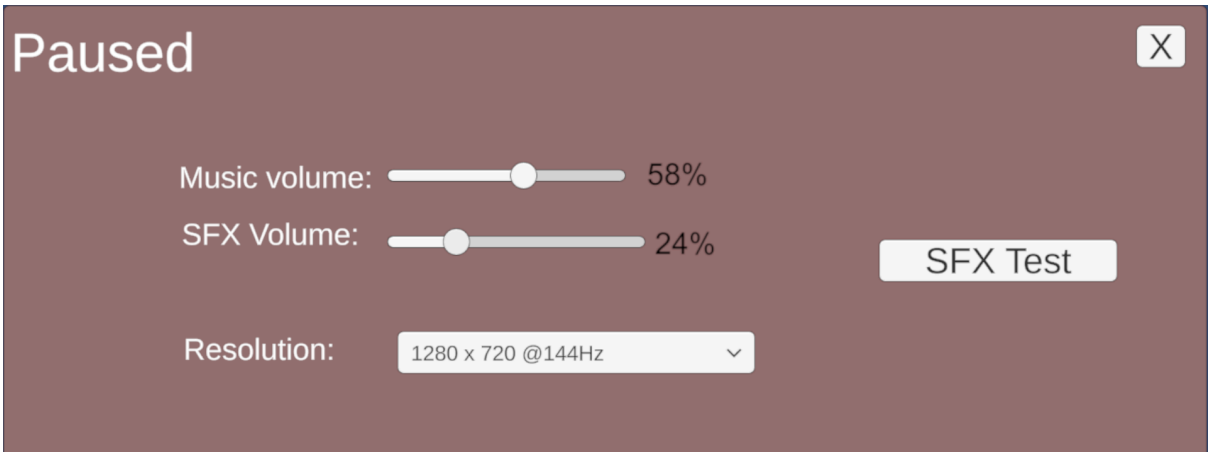
6) Tests realizados

Test	Resultado
modificar volúmenes y cerrar sin Apply	aparece modal Yes/No
aplicar sonidos → cerrar menú → volver a abrir	valores guardados correctamente
fullscreen ON / OFF	cambia inmediatamente y persiste
test buttons música / sfx	funcionan correctamente
responsive 1920x1080 / 1680x1050	correcto sin romper layout

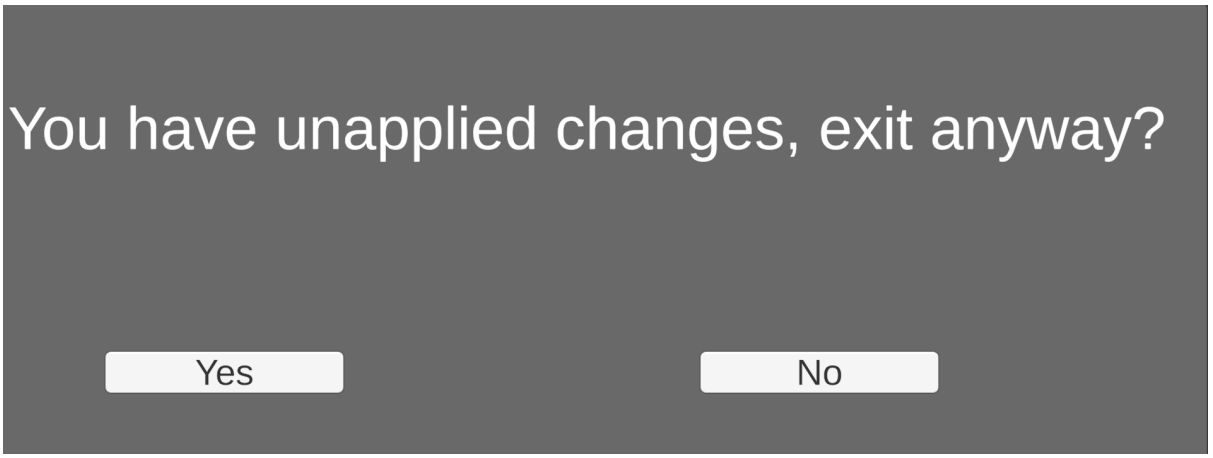
Configuració Per defecte



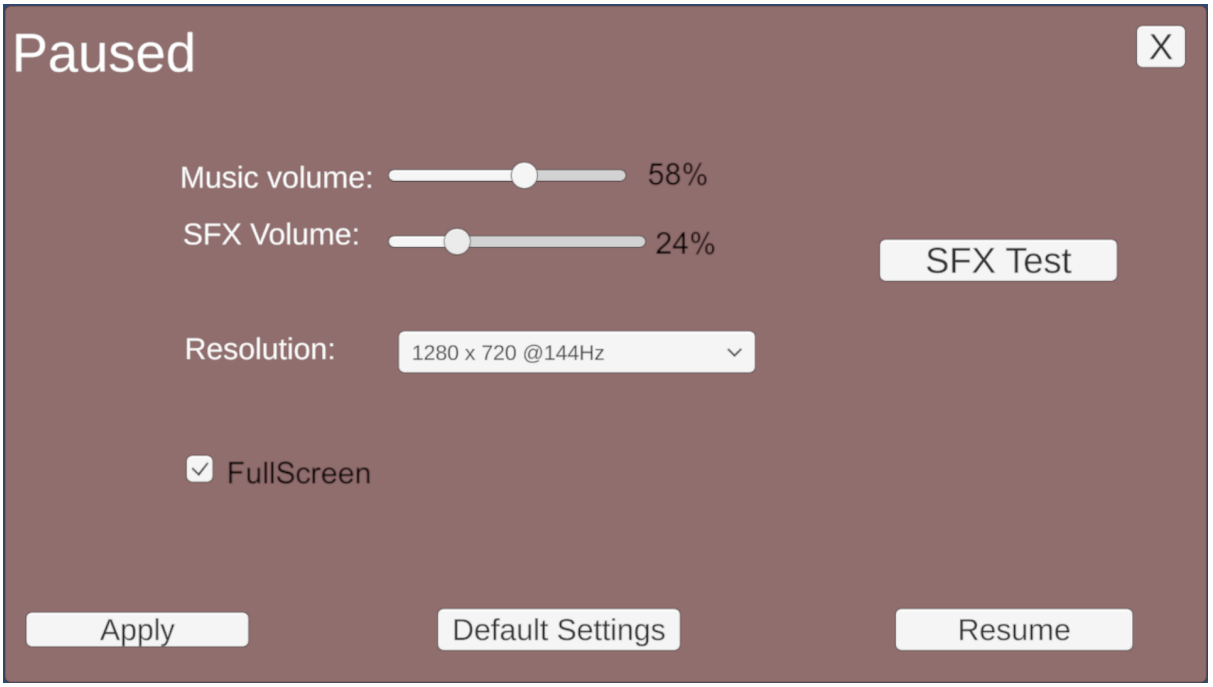
Configuracion cambiada de volumenes



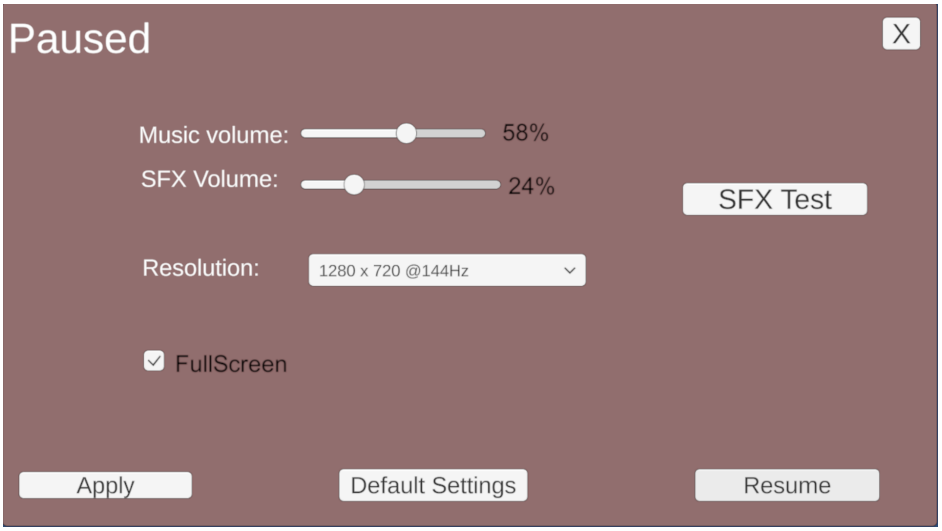
Configuracion sin aplicar cambios



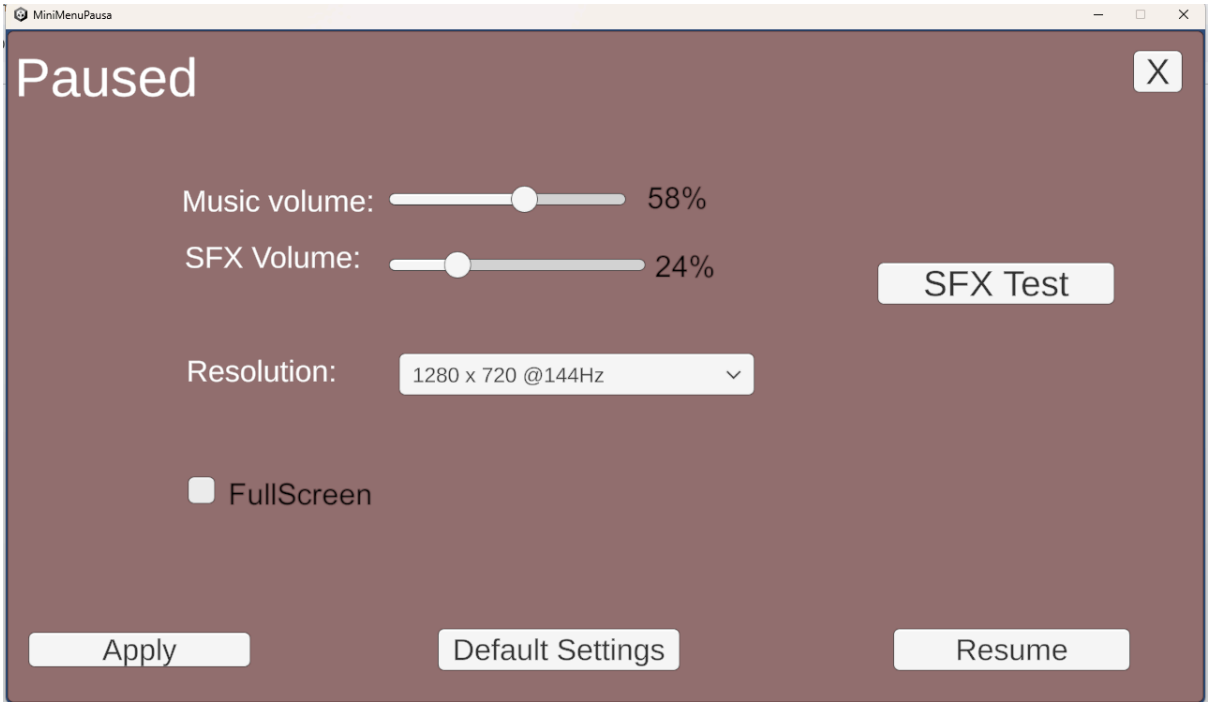
Configuración antes de guardar



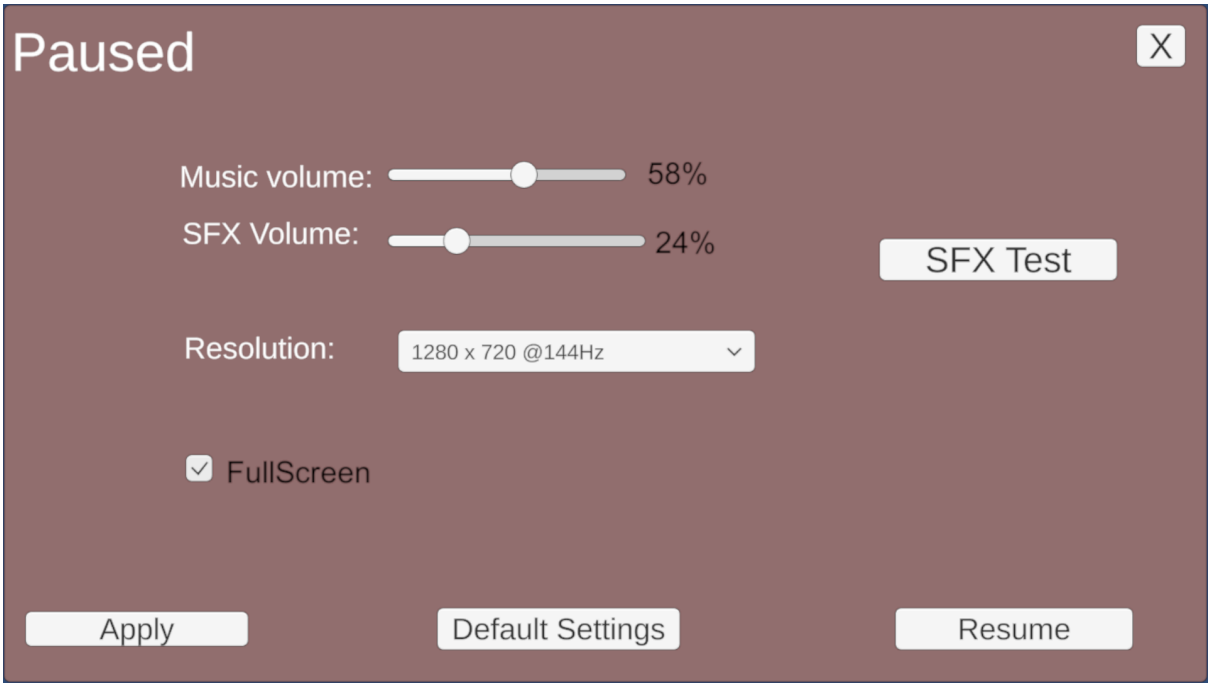
Confirguracion guardada



Configuración de pantalla completa antes de aplicar



Configuracion con pantalla completa



7) Conclusión Final

El menú pausa implementado recrea el comportamiento real de videojuegos profesionales: persistencia, test de audio, resoluciones reales, detección de cambios sin guardar, sonido UI para feedback y diseño responsive.

Cumple los objetivos del RA1 de Diseño de Interfaces y demuestra comprensión de arquitectura UI + UX + lógica de estado en Unity.