

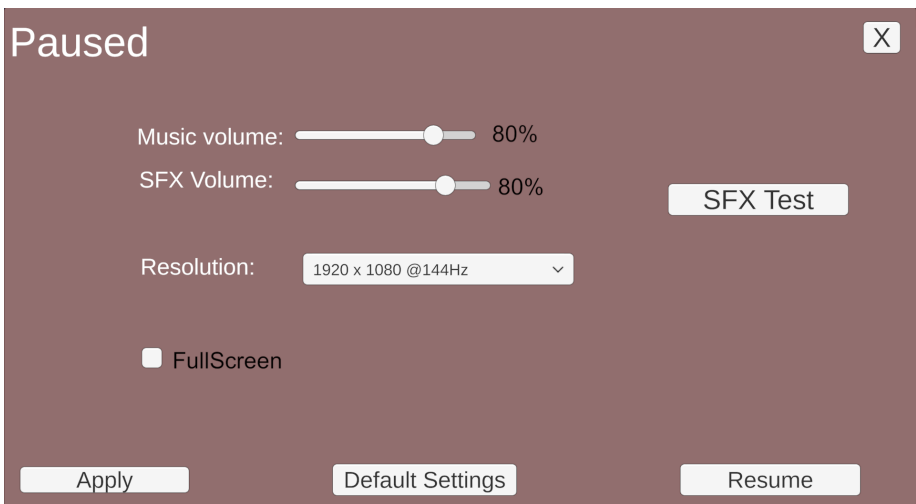
# 1) Objetivo de la actividad

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.

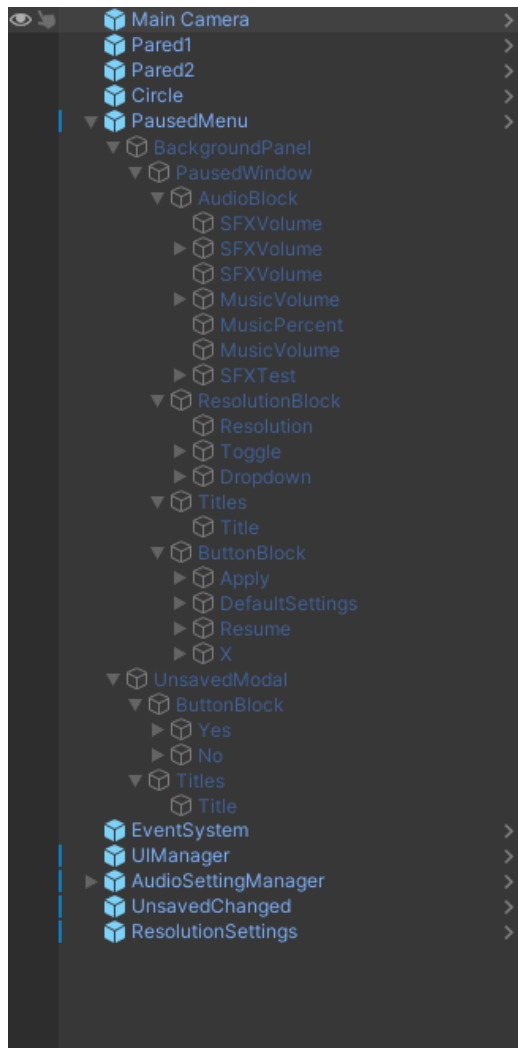
## 2) Funcionalidades implementadas

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



### 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.



## 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.

```
6 public class PauseMenu : MonoBehaviour
7 {
8     public AudioManager audioManager;
9
10    public GameObject pausePanel; // asignar PausePanel
11    public CanvasGroup canvasGroup; // CanvasGroup del pausePanel
12    public float fadeDuration = 0.25f;
13    bool isPaused = false;
14    bool isFading = false;
15
16    void Start()
17    {
18        pausePanel.SetActive(false);
19        canvasGroup.alpha = 0f;
20        canvasGroup.interactable = false;
21        canvasGroup.blocksRaycasts = false;
22    }
23
24    void Update()
25    {
26        if (Input.GetKeyDown(KeyCode.Escape))
27        {
28            TogglePause();
29        }
30    }
31
32    public void TogglePause()
33    {
34        if (isFading) return;
35        isPaused = !isPaused;
36        if (isPaused) StartCoroutine(FadeIn());
37        else StartCoroutine(FadeOut());
38    }
39
```

```

40 1 reference
41 IEnumerator FadeIn()
42 {
43     isFading = true;
44     pausePanel.SetActive(true);
45     AudioManager.RegisterButtonsForPanel(pausePanel);
46
47     float t = 0f;
48     while (t < fadeDuration)
49     {
50         t += Time.unscaledDeltaTime;
51         canvasGroup.alpha = Mathf.Lerp(0f, 1f, t / fadeDuration);
52         yield return null;
53     }
54     canvasGroup.alpha = 1f;
55     canvasGroup.interactable = true;
56     canvasGroup.blocksRaycasts = true;
57     Time.timeScale = 0f; // pausa fisica
58     isFading = false;
59 }
60
61 1 reference
62 IEnumerator FadeOut()
63 {
64     isFading = true;
65     canvasGroup.interactable = false;
66     canvasGroup.blocksRaycasts = false;
67     float t = 0f;
68     while (t < fadeDuration)
69     {
70         t += Time.unscaledDeltaTime;
71         canvasGroup.alpha = Mathf.Lerp(1f, 0f, t / fadeDuration);
72         yield return null;
73     }
74     canvasGroup.alpha = 0f;
75     pausePanel.SetActive(false);
76     Time.timeScale = 1f; // reanuda fisica
77     isFading = false;
78 }
79 // Llamar desde botón Resume
80 0 references
81 public void OnResumeButton()
82 {
83     if (isPaused) TogglePause();
84 }
85 }

```

## 4.2 AudioManager.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();
}

65 void OnMusicSliderChanged(float v)
66 {
67     tempMusic = v;
68     UpdateMusicVolumeUI(v);
69     ApplyMusicVolume(v);
70     if (unsavedHandler) unsavedHandler.SetUnsaved(true);
71 }
72
73 1 reference
74 void OnSfxSliderChanged(float v)
75 {
76     tempSfx = v;
77     UpdateSfxVolumeUI(v);
78     ApplySfxVolume(v);
79     if (unsavedHandler) unsavedHandler.SetUnsaved(true);
80 }
81
82 2 references
83 void UpdateMusicVolumeUI(float v) => musicValueText.text = Mathf.RoundToInt(v * 100) + "%";
84 2 references
85 void UpdateSfxVolumeUI(float v) => sfxValueText.text = Mathf.RoundToInt(v * 100) + "%";
86
87 2 references
88 void ApplyMusicVolume(float v) { musicSource.volume = v; }
89 2 references
90 void ApplySfxVolume(float v) { sfxSource.volume = v; }
91
92 2 references
93 void ApplyAudioVolumes(float music, float sfx)
94 {
95     ApplyMusicVolume(music);
96     ApplySfxVolume(sfx);
97 }
98
99 1 reference
100 public void PlaySfxTest()
101 {
102     sfxSource.PlayOneShot(sfxSource.clip);
103 }
```

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     }
116
```

### 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     }

```

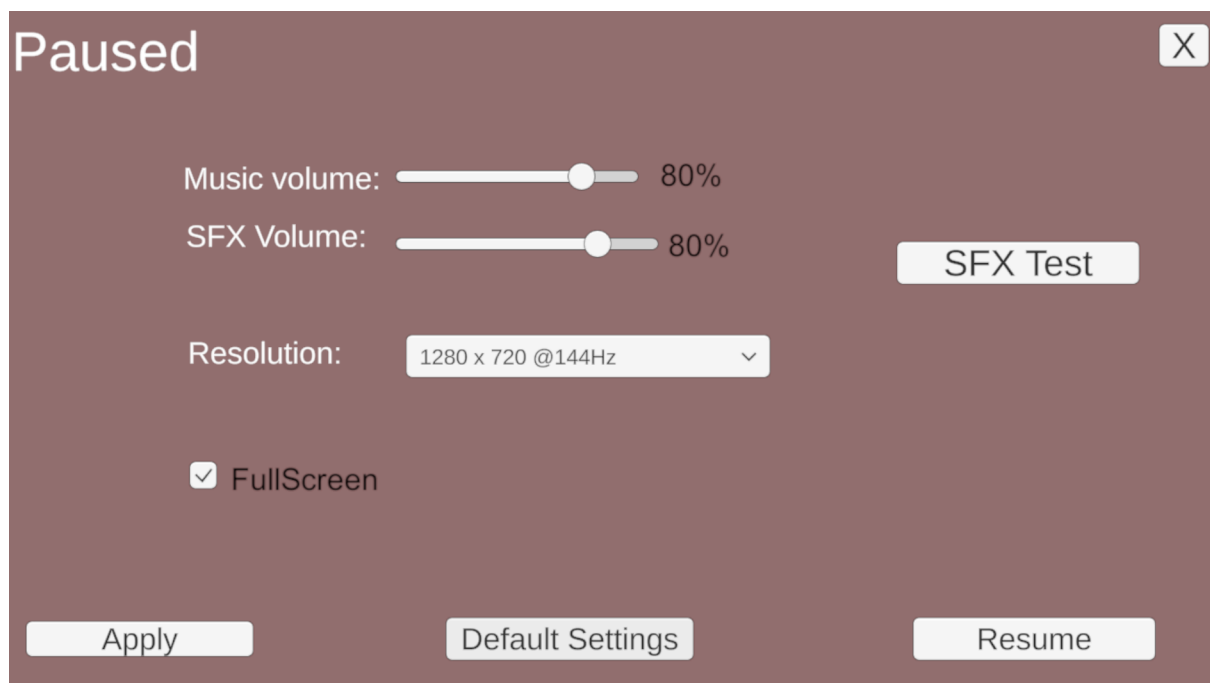
## 5) Persistencia y UX

- 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 pérdida accidental de configuración (UX profesional real de videojuegos).

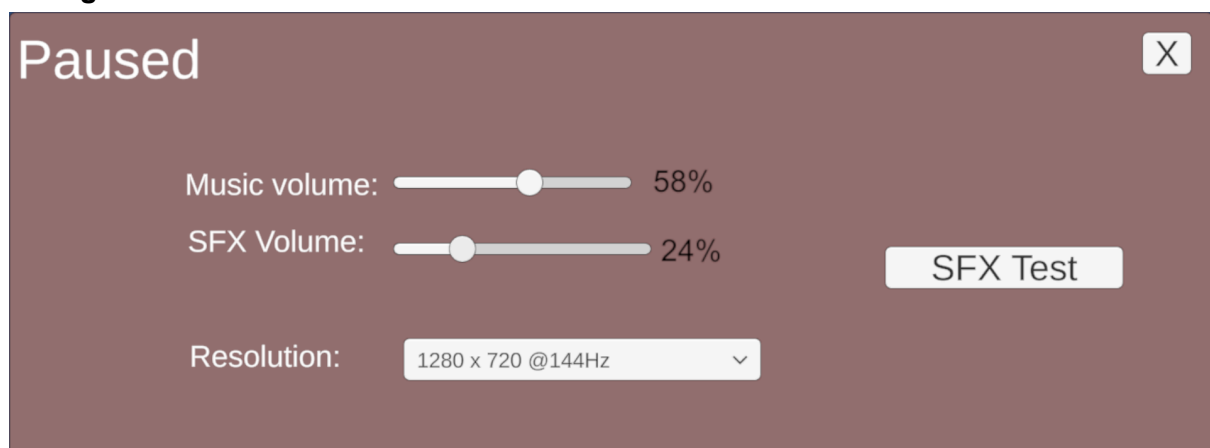
## 6) Tests realizados

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

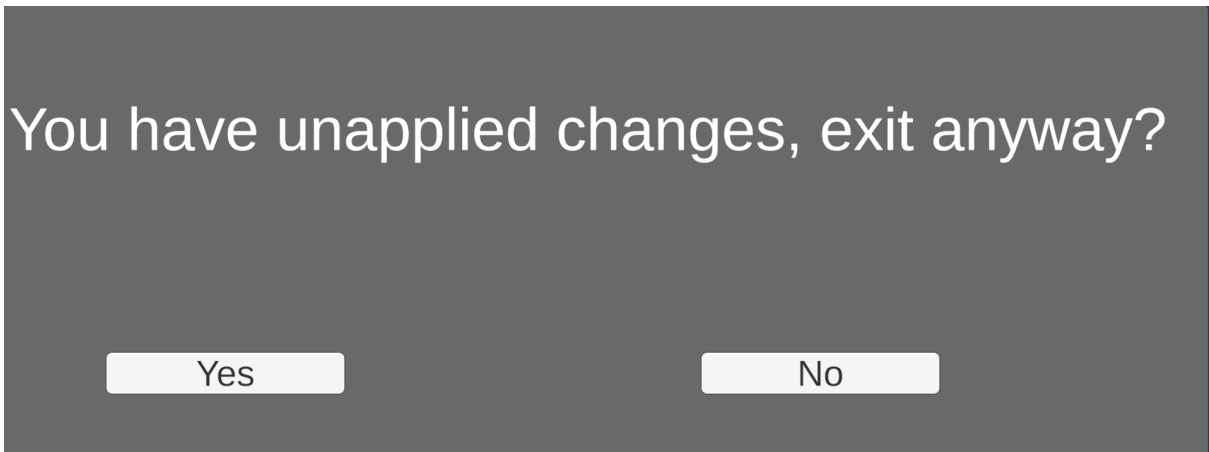


### Configuración cambiada de volúmenes

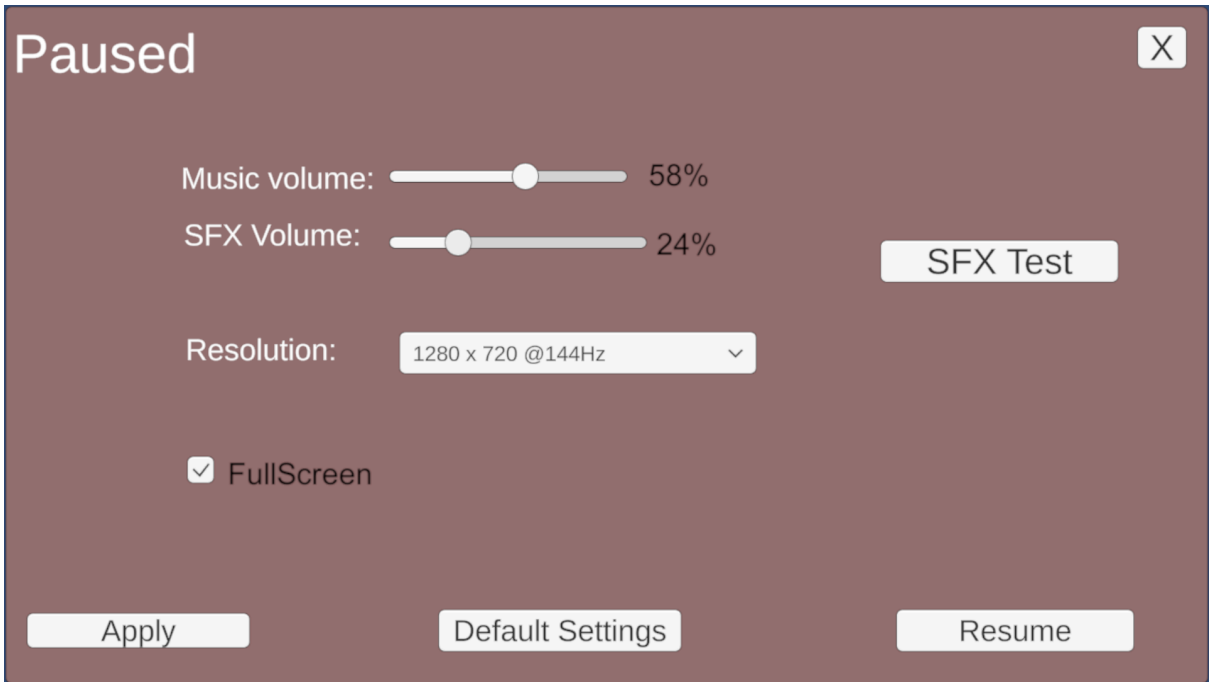




Configuración sin aplicar cambios



Configuración antes de guardar



## Configuración guardada

Paused

X

Music volume: 58%

SFX Volume: 24%

SFX Test

Resolution: 1280 x 720 @144Hz

☒ FullScreen

Apply

Default Settings

Resume

## Configuración de pantalla completa antes de aplicar

MiniMenuPausa

Paused

X

Music volume: 58%

SFX Volume: 24%

SFX Test

Resolution: 1280 x 720 @144Hz

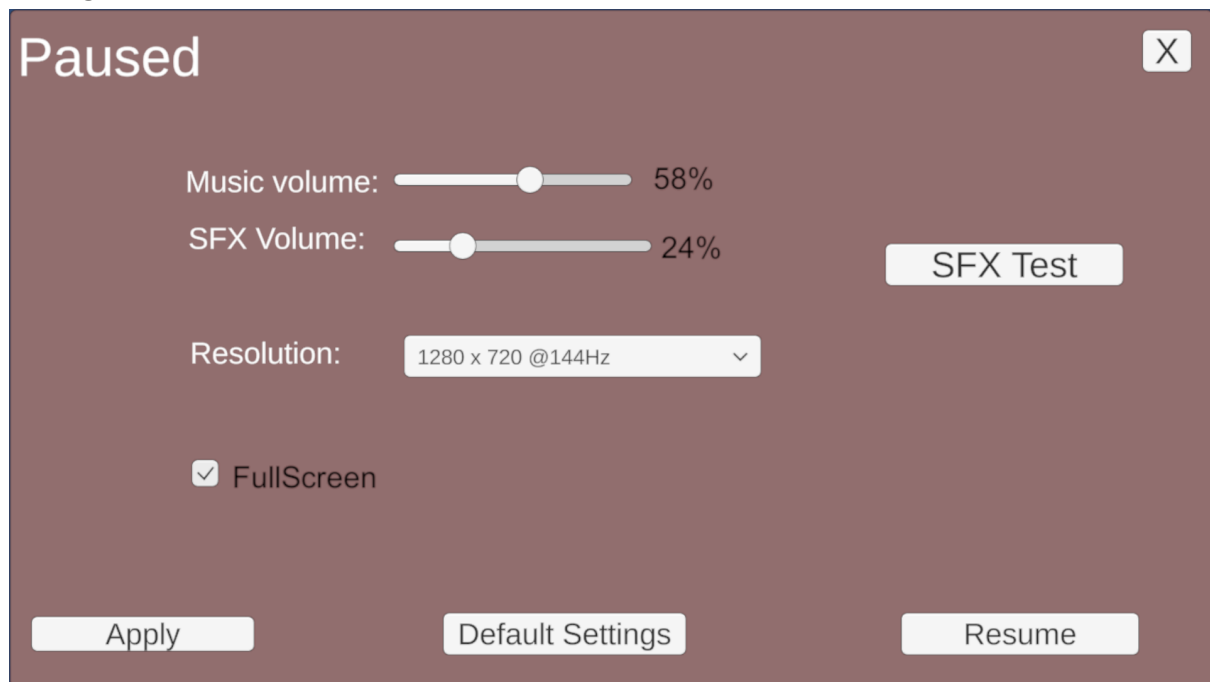
☐ FullScreen

Apply

Default Settings

Resume

## Configuración 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.