Guía de Laboratorio 8

Primera parte
UI Vidas del jugador

1) Imagen contenedora "Player Lives"

1. En la jerarquía: Canvas → clic derecho → **UI** → **Image**.
2. Nómbrala **Player Lives**.
3. En **Source Image** asigna:
   Assets/Images/TinySwords/UI/Ribbons/Ribbon_Red_3Slides.png
4. Activa **Preserve Aspect** para que no se deforme.
5. **Anchor Presets**: mantén **Alt** (Option en macOS) y haz clic en **Top-Left** (arriba-izquierda) para anclar posición y pivot.
6. Ajusta su **RectTransform** para que quede visible dentro del margen superior-izquierdo:
   - Pos X ≈ **20–40**, Pos Y ≈ **-20–40**
   - **Width/Height**: deja que Unity lo adapte por el sprite y el Preserve Aspect; si el panel queda muy grande, reduce **Width** hasta que "ribbon" se vea proporcionado.
7. (Opcional) Desactiva **Raycast Target** en el Image si solo es decorativo (optimiza UI).

Comprobación rápida: Al entrar en Play, el ribbon debe estar arriba-izquierda siempre, independientemente de la resolución.

2) Texto "HPText" (etiqueta)

1. Selecciona **Player Lives** → clic derecho → **UI** → **Text – TextMeshPro**.
2. Nombra el objeto **HPText**.
3. En el componente **Text (TMP)**:
   - Texto: **HP:**
   - **Font Style**: **Bold**
   - **Font Size**: **25**
   - **Alignment**: **Middle Left** (centra vertical, alinea a la izquierda)
   - Overflow: **Overflow** (evita cortes si varía el tamaño)
4. Con el **RectTransform** de **HPText**:
   - Ancla: **Middle Left** (dentro del contenedor)
   - Ajusta Pos X (p.ej. **20**) y Pos Y (**0**) para centrarlo verticalmente en el ribbon.

Comprobación: El texto "HP:" debe verse nítido y centrado verticalmente, con algo de padding izquierdo.

3) Texto "HPNumberText" (valor numérico)

1. En **Player Lives → UI → Text – TextMeshPro**.

2. Nómbralo **HPNumberText**
3. En **Text (TMP)**:
    - Texto inicial: **000**
    - **Font Size**: **25**
    - **Alignment**: **Middle Left** (o **Middle Right** si lo anclas al extremo derecho)
4. Posición:
    - Si lo alineas relativo a **HPText**:
        - Ancla: **Middle Left**
        - Pos X ≈ **(HPText.width + 10 a 20 px)** para que quede a la derecha.
    - Si lo anclas al borde derecho del ribbon:
        - **Anchor**: **Middle Right**, **Alignment**: **Middle Right**, Pos X ≈ **-20** para un margen interno.

Comprobación: Debe leerse “HP: 000” de izquierda a derecha sin superponerse.

4) Script UIManager.cs

Cambios resaltados en color amarrillo.

```
using TMPro;
using UnityEngine;

public class UIManager : MonoBehaviour
{
    public GameObject invetory;

    public TMP_Text moneyCountText;
    public TMP_Text woodCountText;
    public TMP_Text meatCountText;
    public TMP_Text livesText;

    public static UIManager Instance { get; private set; }

    private void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
        }
        else
        {
            Destroy(gameObject);
        }
    }

    public void OpenOrCloseInvetory()
    {
```

```
        invetory.SetActive(!invetory.activeSelf);
    }

    public void UpdateMoney(int value)
    {
        moneyCountText.text = value.ToString();
    }
    public void UpdateWood(int value)
    {
        woodCountText.text = value.ToString();
    }
    public void UpdateMeat(int value)
    {
        meatCountText.text = value.ToString();
    }

    public void UpdateLives(int hpValue)
    {
        livesText.text = hpValue.ToString();
    }
}
```

5) Asignación de referencias en el Inspector

1. En el componente **UIManager** del Inspector:
    - Arrastra el objeto **HPNumberText** a Lives Text del script.

6) Script Player.cs

Cambios resaltados en color amarillo.

```
using System;
using UnityEngine;

public class Player : MonoBehaviour
{
  public float speed = 5;
  Rigidbody2D rb2d;
  Vector2 movementInput;
  Animator animator;

  private int currentLives;
  public int maxLives = 100;
  void Start()
  {
```

```
    rb2d = GetComponent<Rigidbody2D>();
    animator = GetComponent<Animator>();
    currentLives = maxLives;
    UIManager.Instance.UpdateLives(currentLives);
}

void Update()
{
    movementInput.x = Input.GetAxisRaw("Horizontal");
    movementInput.y = Input.GetAxisRaw("Vertical");

    movementInput = movementInput.normalized;

    animator.SetFloat("Horizontal", Math.Abs(movementInput.x));
    animator.SetFloat("Vertical", Math.Abs(movementInput.y));

    CheckFlip();

    OpenCloseInvetory();
}

private void FixedUpdate()
{
    rb2d.linearVelocity = movementInput * speed;
}

void CheckFlip()
{
    if ((movementInput.x > 0 && transform.localScale.x < 0) ||
    (movementInput.x < 0 && transform.localScale.x > 0))
    {
        transform.localScale = new Vector3(
            transform.localScale.x * -1,
            transform.localScale.y,
            transform.localScale.z
        );
    }
```

```
    }

    void OpenCloseInvetory()
    {
        if (Input.GetKeyDown(KeyCode.I))
        {
            UIManager.Instance.OpenOrCloseInvetory();
        }
    }
}
```

Segunda parte
Menú pausa

1) Crear el Panel "Pause Menu"

1. Canvas → clic derecho → **UI** → **Panel**.
2. Nombra el objeto **Pause Menu**.
3. En el RectTransform deja el panel con **anchors estirados** (por defecto de Panel), para cubrir toda la pantalla.
4. En el componente **Image** del Panel:
    - Cambia el **Color** a un **rojo** con algo de transparencia (p. ej. RGBA(180, 20, 20, 160)), así se ve un overlay.

2) Título "PAUSE MENU"

1. Selecciona **Pause Menu** → clic derecho → **UI** → **Text – TextMeshPro**.
2. Nombra el objeto **PauseTitle**.
3. En **Text (TMP)**:
    - Texto: **PAUSE MENU**
    - **Font Size**: **80**
    - **Font Style**: **Bold** (si quieres mayor presencia)
    - **Alignment**: **Center** (centro horizontal y vertical)
4. En el RectTransform, ancla **Middle Center** y deja Pos X = 0, Pos Y ≈ **+120** para que quede un poco más arriba del centro (y así haya espacio para el subtítulo).

3) Subtítulo "Press P to Resume"

1. En **Pause Menu** → **UI** → **Text – TextMeshPro**.
2. Nómbralo **PauseHint**.
3. En **Text (TMP)**:
    - Texto: **Press P to Resume**
    - **Font Size**: **36**
    - **Alignment**: **Center**
4. RectTransform:
    - Ancla: **Middle Center**
    - Pos X = 0, Pos Y ≈ **+40** (o **0** si quieres exactamente en el centro).
5. (Opcional) Para estructura más limpia, podrías añadir un **Vertical Layout Group** en **Pause Menu** y meter **PauseTitle** y **PauseHint** dentro; ajusta Child Alignment = Middle Center y Spacing = 20. Si haces esto, quita offsets manuales de Pos Y.

4) Script UIManager.cs (con pausa)

Cambios resaltados en color amarillo

```
using TMPro;
using UnityEngine;

public class UIManager : MonoBehaviour
{
    public GameObject invetory;
    public GameObject pauseMenu;

    public TMP_Text moneyCountText;
    public TMP_Text woodCountText;
    public TMP_Text meatCountText;
    public TMP_Text livesText;

    public static UIManager Instance { get; private set; }

    private void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
        }
        else
        {
            Destroy(gameObject);
        }
    }

    public void OpenOrCloseInvetory()
    {
        invetory.SetActive(!invetory.activeSelf);
    }

    public void UpdateMoney(int value)
    {
        moneyCountText.text = value.ToString();
    }
    public void UpdateWood(int value)
    {
        woodCountText.text = value.ToString();
    }
    public void UpdateMeat(int value)
    {
        meatCountText.text = value.ToString();
    }
```

```
    public void UpdateLives(int hpValue)
    {
        livesText.text = hpValue.ToString();
    }

    public void PauseGame()
    {
        pauseMenu.SetActive(true);
        Time.timeScale = 0;
    }

    public void ResumeGame()
    {
        pauseMenu.SetActive(false);
        Time.timeScale = 1;
    }
}
```

5) Asignar el Panel al UIManager

1. Selecciona el GameObject que tiene el componente **UIManager**
2. Arrastra el objeto **Pause Menu** (Panel) al campo **pauseMenu** del UIManager en el **Inspector**.

6) Script Player.cs (abrir/cerrar pausa)

Cambios resaltados en amarillo.

```
using System;
using UnityEngine;

public class Player : MonoBehaviour
{
    public float speed = 5;
    Rigidbody2D rb2d;
    Vector2 movementInput;
    Animator animator;

    private int currentLives;
    public int maxLives = 100;
    private bool gameIsPaused = false;
    void Start()
    {
        rb2d = GetComponent<Rigidbody2D>();
        animator = GetComponent<Animator>();
        currentLives = maxLives;
        UIManager.Instance.UpdateLives(currentLives);
    }
```

```
    void Update()
    {
        movementInput.x = Input.GetAxisRaw("Horizontal");
        movementInput.y = Input.GetAxisRaw("Vertical");

        movementInput = movementInput.normalized;

        animator.SetFloat("Horizontal", Math.Abs(movementInput.x));
        animator.SetFloat("Vertical", Math.Abs(movementInput.y));

        CheckFlip();

        OpenCloseInvetory();
        OpenClosePauseMenu();
    }

    private void FixedUpdate()
    {
        rb2d.linearVelocity = movementInput * speed;
    }

    void CheckFlip()
    {
        if ((movementInput.x > 0 && transform.localScale.x < 0) ||
        (movementInput.x < 0 && transform.localScale.x > 0))
        {
            transform.localScale = new Vector3(
                transform.localScale.x * -1,
                transform.localScale.y,
                transform.localScale.z
            );
        }
    }

    void OpenCloseInvetory()
    {
        if (Input.GetKeyDown(KeyCode.I))
        {
            UIManager.Instance.OpenOrCloseInvetory();
        }
    }

    void OpenClosePauseMenu()
    {
        if (Input.GetKeyDown(KeyCode.P))
        {
            if (gameIsPaused)
            {
```

```
                UIManager.Instance.ResumeGame();
                gameIsPaused = false;
            }
            else
            {
                UIManager.Instance.PauseGame();
                gameIsPaused = true;
            }
        }
    }
}
```