

«Talento Tech»

Desarrollo de Videojuegos

Unity 3D

Clase 09



Clase N° 9 | Interacción con UI en 3D II

Temario:

- Crearemos una barra de vida en el enemigo
- ProBuilder

Objetivos de la clase

En esta clase nos enfocaremos en dos áreas fundamentales del desarrollo de videojuegos: la visualización del estado de los enemigos y el diseño estructural del entorno.

1. **Implementar una barra de vida visual** para un enemigo, que proporcione feedback claro al jugador durante el combate.
2. **Diseñar una barra de vida que siempre se alinee con la cámara**, asegurando su visibilidad desde cualquier ángulo.
3. **Optimizar la interfaz** haciendo que la barra desaparezca cuando el jugador se aleje, evitando una sobrecarga visual en pantalla.
4. **Explorar el uso básico de ProBuilder en Unity** como herramienta para crear entornos 3D directamente desde el editor.
5. **Construir figuras básicas y personalizarlas** para diseñar prototipos de niveles, plataformas o estructuras jugables.
6. **Editar vértices, bordes y caras** para adaptar la geometría a las necesidades específicas del diseño del mundo de *Nexus*.

UI Ingame

Barra de vida enemiga

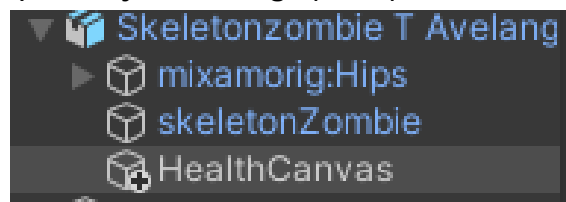
En la clase de hoy seguiremos trabajando con las capacidades de una UI ingame. Como primer ejemplo crearemos una barra de vida sencilla que se ubique arriba de los enemigos que:

- Esté siempre orientada hacia la cámara.
- Se active únicamente cuando el jugador se acerque.
- Mejore la experiencia de combate sin sobrecargar visualmente la pan

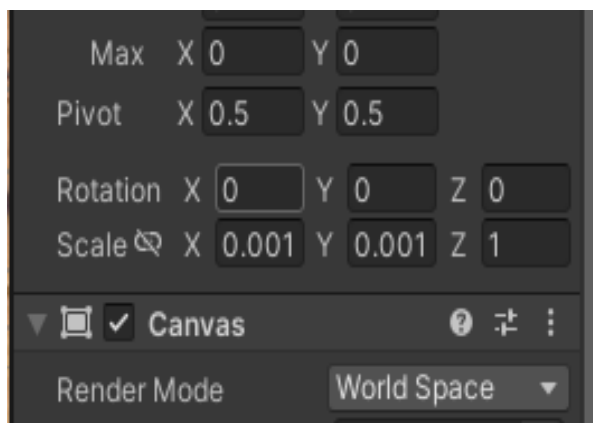
Configuración inicial: Canvas en World Space

1. Crear la barra de vida:

- Dentro de Unity, agregá un nuevo Canvas configura su propiedad de Render Mode en **World Space**.
- Hacelo hijo (**child**) del **objeto enemigo** para que se mueva junto con él.



- Escalalo a 0.001 y posicionalo en las coordenadas (0, 0, 0) para ajustarlo fácilmente sobre su cabeza.



- Acomoda la barra de vida para que quede centrada



Al darle play, se ve que si vamos por el costado del enemigo, la barra de vida no apunta a nosotros/as y por lo tanto, nos dificulta su visualización. Vamos a corregirlo generando un script asignado a nuestro enemigo:



Script LookAt. Mejora visual: Siempre mirando a la cámara

Para garantizar que la barra siempre esté visible sin importar desde qué ángulo se mire al enemigo, usaremos el siguiente script:

```
Transform target;
// Start is called before the first frame update
void Start(){
    target = GameObject.Find("Player").transform;
}
// Update is called once per frame
void Update(){
    transform.LookAt(target);
}
```


Explicación:

- la variable target es el jugador, etiquetado como "Player"
- En cada frame (Update), la barra se orienta hacia el jugador, manteniéndose visible.



Ahora nos encontraremos con un posible problema:

😞 ¿Queremos que todas las barras del juego apunten a nuestro jugador y sean visibles constantemente?

Esto supondría un consumo importante de recursos. Por eso, paraa solucionarlo crearemos un Script de activación en nuestro enemigo.

Optimización: Activar la barra solo si el jugador está cerca

```
GameObject bar;
Transform player;
[SerializeField]float MinDistance = 10f;

void Start(){
    bar = transform.GetChild(2).gameObject;
    player = GameObject.Find("Player").transform;
}

void Update(){
    float d = Vector3.Distance(player.position,
transform.position);

    if (d <= MinDistance && !bar.gameObject.activeInHierarchy){
        bar.SetActive(true);
    }else if (d > MinDistance){
        bar.SetActive(false);
    }
}
```

```
}
```

Resumen del comportamiento:

- Si el jugador se encuentra dentro del rango (**MinDistance**), la barra se activa.
- Si se aleja, se oculta automáticamente.

Explicación

Variables

```
GameObject bar;  
Transform player;  
[SerializeField]float MinDistance = 10f;
```

- **GameObject bar**: Representa la barra (u otro elemento del HUD/UI) que se activa o desactiva en función de la distancia.
- **Transform player**: Referencia al transform del jugador (para obtener su posición en el espacio).

[SerializeField] float MinDistance:

- **[SerializeField]**: permite ajustar el valor de la variable desde el Inspector de Unity sin necesidad de hacerlo público.
- **Float MinDistance**: Será la distancia mínima en la que el jugador debe estar para que la barra sea visible.

Método Start()

```
void Start(){  
    bar = transform.GetChild(2).gameObject;  
    player = GameObject.Find("Player").transform;  
}
```

transform.GetChild(2).gameObject:

- Obtiene el tercer hijo (arrancando desde el 0, el índice 2) del objeto al que está vinculado el script y lo asigna a la variable bar.

Asumimos que este hijo representa la barra que se mostrará/ocultará

GameObject.Find("Player").transform:

- Busca en la jerarquía un objeto llamado "Player" y obtiene su Transform, permitiéndonos acceder a la posición del jugador en el mundo.

Método Update()

```
void Update() {  
    float d = Vector3.Distance(player.position, transform.position);  
    if (d <= MinDistance && !bar.gameObject.activeInHierarchy) {  
        bar.SetActive(true);  
    } else if (d > MinDistance) {  
        bar.SetActive(false);  
    }  
}
```

float d = Vector3.Distance(player.position, transform.position): Guardamos en la variable *d* la distancia entre el jugador y el objeto al que está asociado este script.

- **player.position:** Posición del jugador.
- **transform.position:** Posición del objeto actual.

Condicional:

if (d <= MinDistance && !bar.gameObject.activeInHierarchy):

Comprobamos si la distancia entre el jugador y el objeto (*d*) es menor o igual a la distancia mínima (*MinDistance*) y verificamos si la barra (*bar*) no está actualmente activa.

bar.SetActive(true):

- Si ambas condiciones son verdaderas, activa la barra (*bar*), haciéndola visible en el juego.

Condición: else if (d > MinDistance):

- Si la distancia es mayor que *MinDistance*, desactiva la barra.

bar.SetActive(false):

- Desactiva la barra (*bar*), haciéndola invisible.

Finalmente tendremos una barra de vida sobre nuestro enemigo que solo aparecerá cuando nos acerquemos lo suficiente a él.

ProBuilder

Diseño rápido de escenarios en Unity

🤔 ¿Qué pasaría si deseamos crear una mejor pared, puerta o ambiente?

🤔 ¿Tendremos que descargar todo desde la Asset Store o unir cada figura base?

Para esto podemos usar la herramienta de “**Pro Builder**”.

Este package es ideal para agilizar el prototipado de niveles. Con él, crearemos fácilmente componentes básicos, como una pared que incluya el espacio para una puerta.

ProBuilder está integrada en Unity y nos permite:

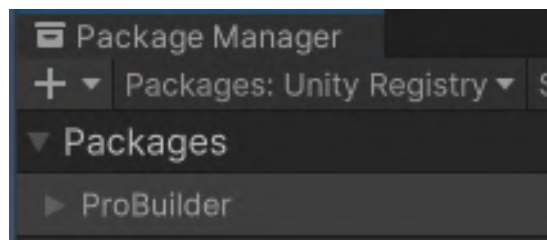
- Crear y editar geometría directamente dentro del editor.
- Diseñar niveles, plataformas o estructuras de forma visual y rápida.
- Prototipar espacios sin necesidad de usar software externo de modelado 3D.

Es ideal para diseñadores y desarrolladores que quieran:

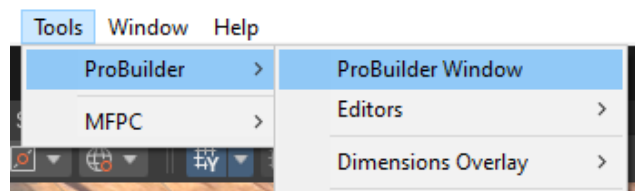
- Crear escenarios jugables sin depender de modeladores.
- Iterar con rapidez en el layout de niveles.
- Probar ideas sin necesidad de texturas o modelos finales.

¿Cómo instalar ProBuilder?

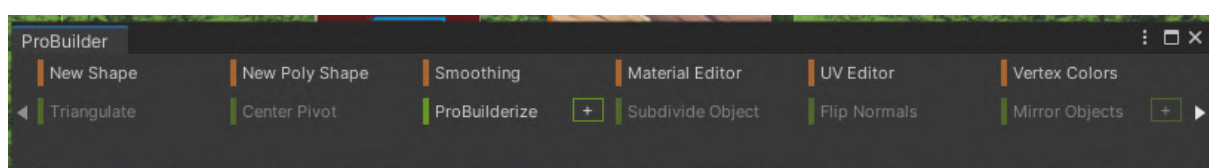
1. Ir a Window → Package Manager.
2. Elegimos Unity Registry para que sea de Unity y en el buscador de la derecha, pondremos “ProBuilder” y le damos Click.



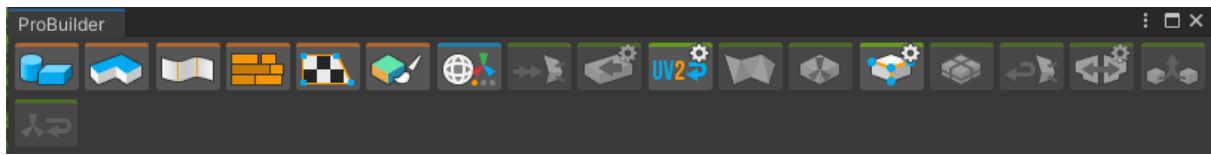
3. Hacemos click en Install.
4. Una vez instalado volvemos a la pantalla principal, y vamos a poder acceder a sus herramientas desde: Tools → ProBuilder → ProBuilder Window



Lograraremos ver una interfaz parecida a esta:



Si no nos gusta esta visualización de la herramienta, a la derecha en los 3 puntos podremos cambiarlo al “Icon Mode”:



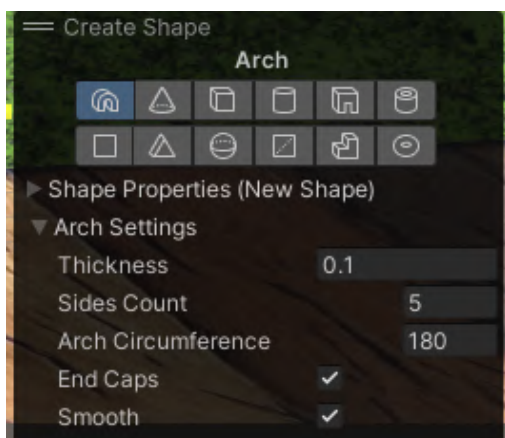
Dentro de esta ventana, tendremos numerosas herramientas. Por el momento nos concentramos en: “**New Shape**” y “**New Poly Shape**”

Usando ProBuilder en la escena

Para usar ProBuilder, vamos a organizarnos en 4 pasos que nos permitan crear, diseñar y organizar nuestras geometrías en la escena.

Paso 1: Crear una forma básica

- Abrimos la ventana de **ProBuilder** desde:
Tools → ProBuilder → ProBuilder Window.
- Hacemos click en el botón **New Shape** y elegimos una forma básica como cubo, plano, cilindro o esfera (cube, plane, cylinder, sphere)
- Ajustamos los parámetros de tamaño, rotación o subdivisiones si lo necesitás.
- Presionamos **Build** para colocar el objeto en la escena.

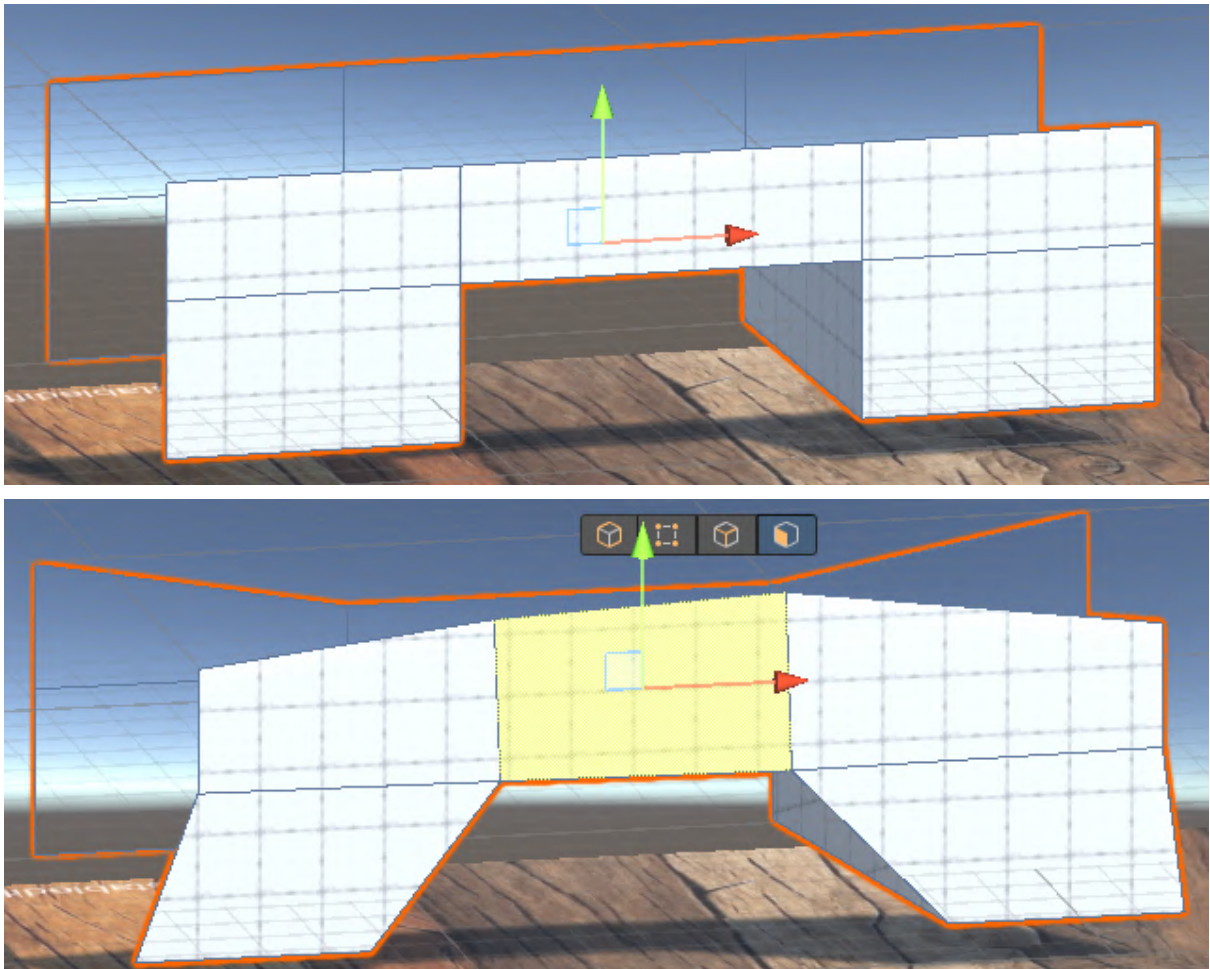


Cada forma tiene una variedad de propiedades que nos permiten modificarla a gusto.

Paso 2: Editar la geometría del objeto

1. Seleccioná el objeto creado.
2. Desde la **barra de herramientas de ProBuilder**, elegí el modo de edición:
 - Vértices (puntos)
 - Bordes (líneas)

- Caras (superficies)
3. Con las herramientas estándar (Move, Scale, Rotate) podés modificar la forma como quieras.



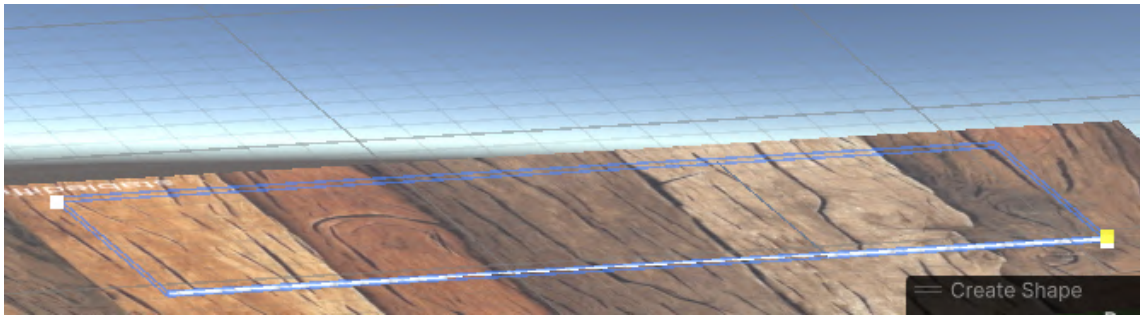
Paso 3: Aplicar materiales o colores

1. Seleccioná la cara o el objeto completo.
2. En el panel de ProBuilder, usá la herramienta **Material Editor**.
3. Arrastrá un material desde tu carpeta de Assets o aplicá un color plano.

Paso 4: Organizar tu escena

- Recomendamos agrupar los objetos de ProBuilder dentro de **Empty Objects** para mantener la jerarquía ordenada.
- Podés renombrar cada forma según su función (ej. "PlataformaInicio", "ZonaSecreta", "EscaleraCentral").

- Es útil **duplicar y ajustar** objetos en lugar de crearlos desde cero cada vez.



New Poly Shape

La herramienta **New Poly Shape** de ProBuilder permite crear formas **totalmente personalizadas**, dibujadas a mano alzada dentro del editor. Es ideal para diseñar:

- Caminos irregulares
- Plataformas con bordes personalizados
- Superficies naturales o estructuras no convencionales

Esta opción, nos dejará crear una figura “a mano” realizando el dibujo sobre una superficie y luego darle una altura determinada.

Paso a paso para usar New Poly Shape

1. Abrir la herramienta

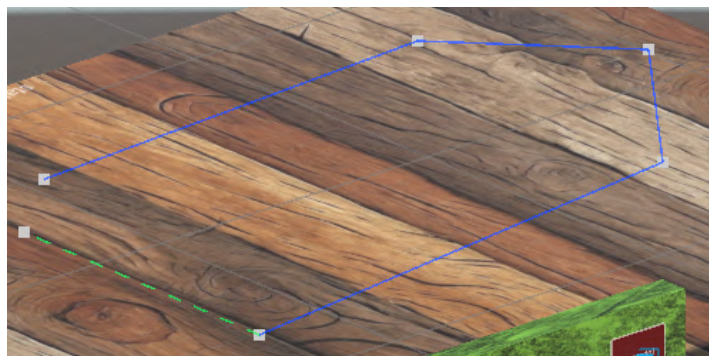
Desde el menú superior: **Tools** → **ProBuilder** → **New Poly Shape**

Esto abrirá un nuevo modo de edición en la escena, activando la herramienta de dibujo.

2. Dibujar la forma en el plano

- **Clic izquierdo:** agrega puntos que definen el contorno de tu forma.
- Podés crear curvas, ángulos o líneas rectas según cómo distribuyas los puntos.
- A medida que clickeás, vas viendo cómo se forma el perímetro.

💡 **Sugerencia:** Mantené presionada la tecla **Shift** para alinear puntos fácilmente.



3. Cerrar la forma

Cuando termines de marcar el contorno:

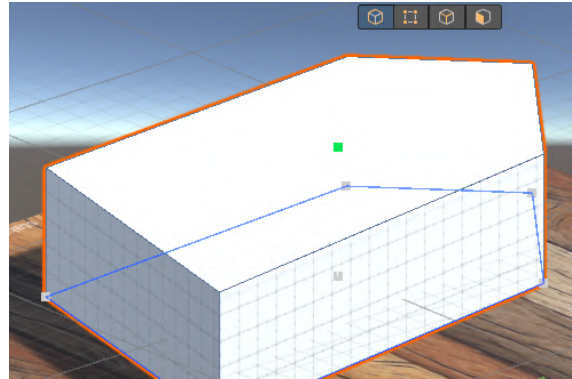
- Presioná **Enter** para cerrar la forma y convertirla en una superficie editable.
- Unity generará automáticamente un objeto ProBuilder plano con base en tu trazo.

4. Extruir para dar volumen

Con la forma seleccionada:

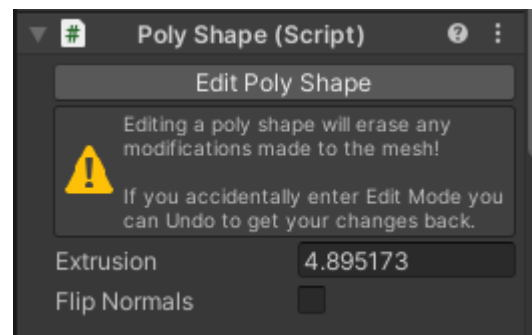
1. Activá el modo de selección de **cara**.
2. Seleccioná la cara superior.
3. Usá la herramienta de **extrusión** para darle altura al objeto.

Esto convierte tu forma 2D en un volumen 3D completamente personalizado.



● Si la figura ya fue creada y desean seguir modificando, pueden seleccionarla ir al Script “Poly Shape (Script)” y hacer click en “Edit Poly Shape”

Esto nos habilitará nuevamente para modificar las propiedades/variables de nuestra figura.



¿Por qué usarlo?

- Es una forma **rápida y flexible** de crear geometrías adaptadas al diseño de nivel.
- Ideal para plataformas con formas únicas, estructuras naturales o espacios no convencionales.
- Compatible con las herramientas de edición y materiales de ProBuilder.

Materiales y recursos adicionales.

ProBuilder

<https://docs.unity3d.com/Packages/com.unity.probuilder@6.0/manual/index.html>

Vida y Maqueta:



Después de implementar el panel dinámico de datos para el jugador, Talento Lab da un giro en el enfoque del proyecto: *"Nuestro mundo está empezando a cobrar vida, pero no podemos olvidar que los enemigos también forman parte de esta experiencia inmersiva."*

En una nueva reunión, el cliente destaca la importancia de crear enemigos que no solo desafíen al jugador, sino que también proporcionen información visual clara sobre su estado durante los enfrentamientos. Para ello, solicitan implementar una **barra de vida** en los enemigos que permita a los jugadores medir su progreso durante un combate.

Además, el equipo de TalentoLab recibirá un encargo relacionado con la construcción del mundo: es momento de empezar a crear estructuras más complejas e interactivas utilizando **ProBuilder**, una herramienta que permitirá modelar y personalizar los entornos directamente dentro de Unity.

Ejercicios prácticos:

Con el panel de datos dinámico implementado, el equipo de *Nexus* se prepara para el siguiente gran paso: **darle vida al mundo y a los enemigos**. Durante la última reunión, el cliente dejó en claro dos nuevas prioridades:

1. **Incorporar una barra de vida visible en los enemigos**, que ayude al jugador a interpretar visualmente el estado del combate.
2. **Diseñar una maqueta funcional del entorno**, utilizando **ProBuilder** para comenzar a construir las estructuras principales del mundo Nexus.




Elizabeth te envió un mensaje:

"Nuestro mundo está tomando forma, pero los enemigos también deben comunicar información clara y útil al jugador. Y es momento de que empecemos a visualizar cómo será realmente Nexus." — TalentoLab

1. Vida visible en enemigos

Implementá un sistema que muestre la **barra de vida sobre la cabeza de cada enemigo**, cumpliendo los siguientes requisitos:

- Usá un **Canvas en World Space** como hijo del enemigo.
- La barra debe orientarse **siempre hacia el jugador**.
- Solo debe activarse si el jugador está a cierta distancia (para no sobrecargar la pantalla).
- La vida debe poder disminuir visualmente (por ejemplo, reduciendo el ancho del `Image.fillAmount` o `scale.x`).

 Podés utilizar el código base compartido en clase o adaptarlo según tu implementación actual.

2. Maqueta de nivel con ProBuilder

Construí una **maqueta funcional** del entorno utilizando exclusivamente herramientas de **ProBuilder**.

El objetivo no es que sea visualmente definitiva, sino que represente:

- Las principales áreas del nivel.
- Conectividad entre zonas.
- Elementos jugables como:
 - Plataformas.
 - Rampas y escaleras.
 - Pasillos y accesos.
 - Zonas elevadas o secretas.

 **Sugerencias:**

- Usá New Shape y New Poly Shape para experimentar con distintas formas.
- Organizá la jerarquía con nombres claros.
- Si te ayuda, podés incluir un plano o boceto previo de cómo pensás organizar la escena.

Objetivo de esta entrega

Este ejercicio busca que puedas:

- Integrar elementos de UI al gameplay (feedback visual sobre enemigos).
- Explorar tus ideas de diseño de niveles con herramientas rápidas y flexibles.
- Tener una primera versión del espacio jugable que servirá como base para futuras iteraciones.

Esperable:

- Escena funcional con:
 - Al menos **un enemigo con barra de vida visible**.
 - Una **maqueta jugable** creada con ProBuilder.
- Capturas o video corto (opcional, pero recomendable).
- Archivo .unitypackage o proyecto comprimido.



Buenos Aires
aprende
Agencia de Habilidades para el Futuro

BA Buenos
Aires
Ciudad