

*"Año de la recuperación y consolidación de la economía peruana"*



Facultad de Ingeniería de Sistemas E Informática

**"Guía Práctica N°12"**

**ASIGNATURA:** DESARROLLO DE VIDEOJUEGOS

**DOCENTE:** FERNANDEZ RIVERA, DIEGO ALEJANDRO

**NRC:** 62092

**ESTUDIANTE:**

Apellidos y Nombres	CODIGO
BELITO RAMIREZ MORI OCTAVIO	74902137
CORONEL BURGOS JAVIER DANIEL	71997263
QUISPE UBALDO ALFREDO	71438344

**HUANCAYO - 2025**

## INFORME DE RESULTADOS DE PRUEBAS QA:

Rol: Tester QA / Integrador

Asignado a: Coronel Burgos Javier Daniel

Proyecto: Práctica 12 – La Chispa de Vida

Fecha: 06/11/2025

### Objetivo de las pruebas

Verificar el correcto funcionamiento del sistema de Inteligencia Artificial implementado mediante el Patrón de Diseño State, garantizando que los estados de Patrulla (PatrolState) y Persecución (ChaseState) se comporten según lo especificado en la guía y que las transiciones entre ellos sean naturales, fluidas y sin errores.

### Configuración del entorno

Motor: Unity 2022.3 LTS

Render Pipeline: Universal Render Pipeline (URP)

Escena: WorldScene – Terreno con NavMesh horneado

**Agente IA:** Prefab Enemy con componentes:

NavMeshAgent

AIController (asociado a los scripts AIState, PatrolState, ChaseState)

Player: Objeto con tag "Player", controlado por FirstPersonController

Waypoints: Waypoint\_01, Waypoint\_02, Waypoint\_03 (asignados en AIController)

### Parámetros de IA (Dinámicos):

Patrol Speed: 2

Chase Speed: 5

Detection Radius: 10

Lose Sight Radius: 15

### Procedimiento de prueba

#### Inicio del juego

Se presiona Play desde el Editor.

El enemigo inicializa correctamente en el estado PatrolState.

En consola se registra:

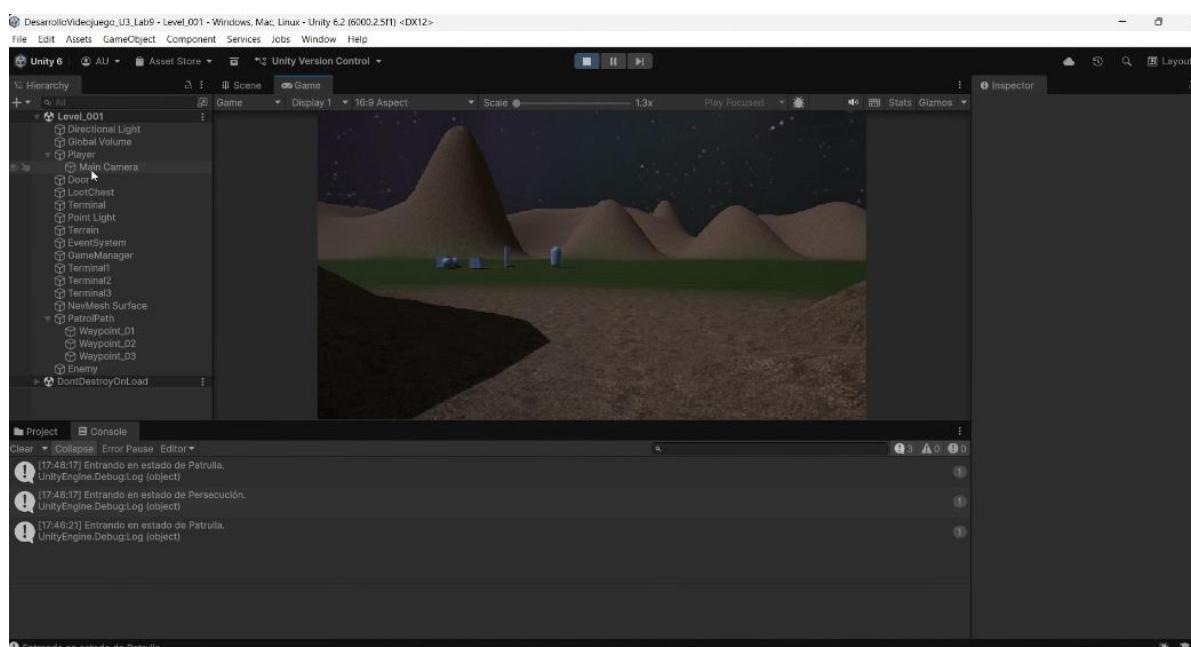
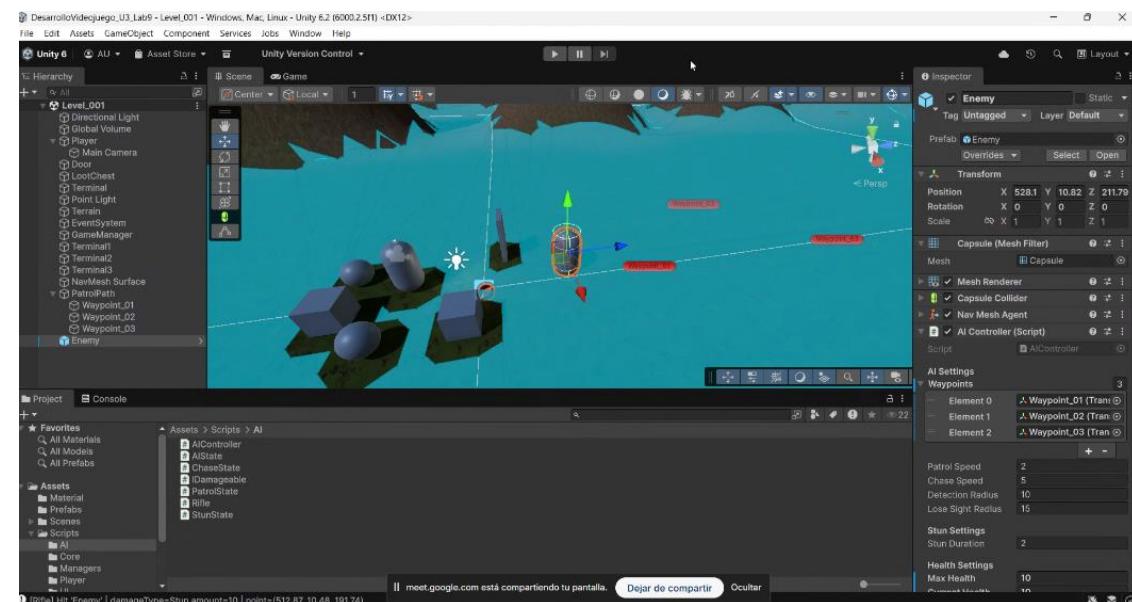
Entrando en estado de Patrulla.

El agente se desplaza hacia el primer waypoint utilizando la malla de navegación.

**Resultado:** El enemigo sigue la ruta de patrulla correctamente, rotando suavemente hacia cada punto asignado y continuando en bucle.

Comportamiento esperado: **Cumplido.**

### Anexo:



## Prueba de detección del jugador

El jugador se aproxima progresivamente al enemigo.

Al ingresar dentro del radio de detección (< 10 unidades), la IA ejecuta transición a ChaseState.

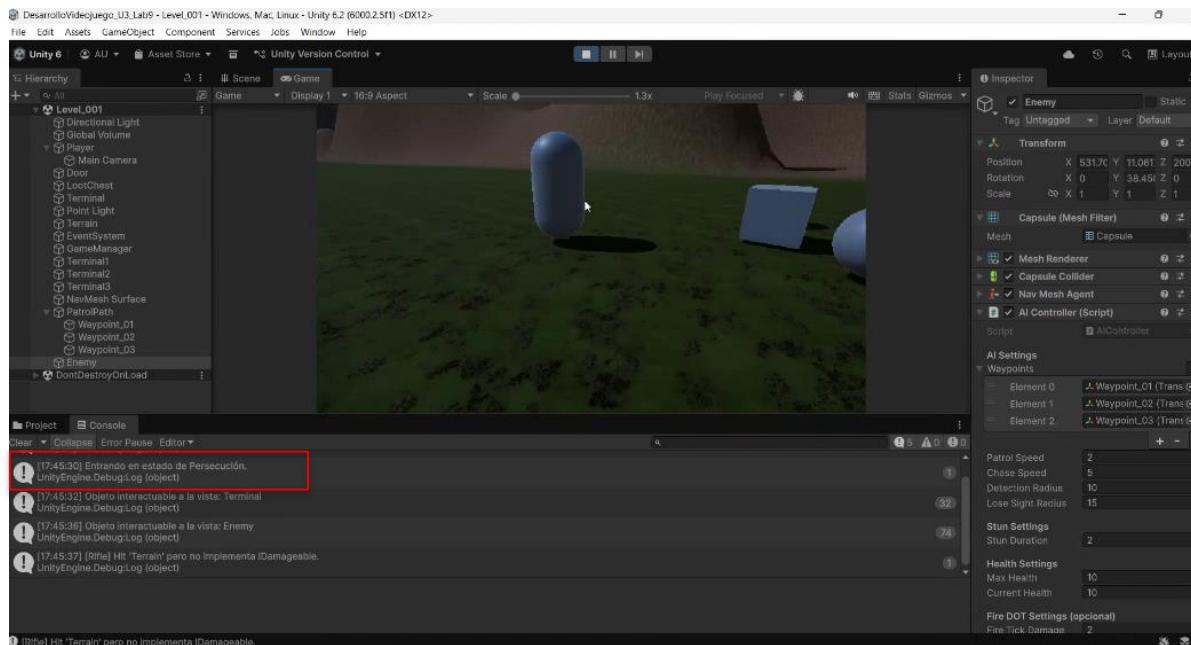
En consola aparece:

Entrando en estado de Persecución.

El agente incrementa su velocidad y ajusta su destino dinámicamente hacia la posición del jugador (`m_agent.destination = m_playerTransform.position`).

**Resultado:** El enemigo detecta correctamente la presencia del jugador y cambia al estado de persecución.

Comportamiento esperado: **Cumplido.**



## Prueba de persecución activa

Durante el estado ChaseState, el enemigo mantiene la persecución continua, recalculando trayectorias al moverse el jugador.

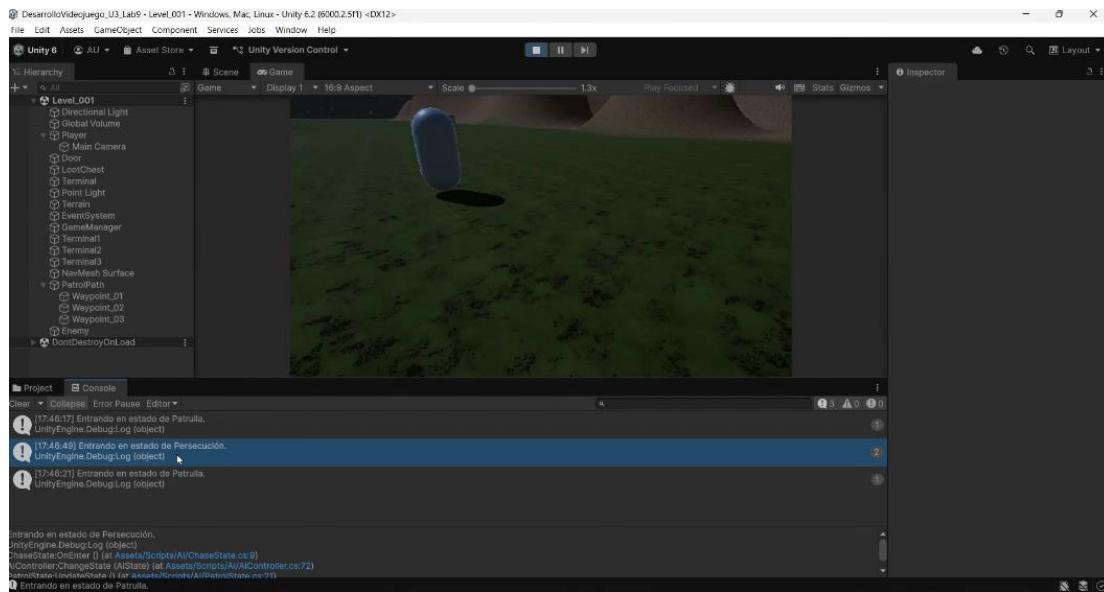
No se observan errores de trayectoria ni colisiones con el terreno.

La velocidad de desplazamiento es coherente con el valor configurado (5 unidades).

**Resultado:** El enemigo persigue de forma fluida y estable.

Comportamiento esperado: **Cumplido.**

## Anexo



## Prueba de pérdida de vista

El jugador se aleja progresivamente del enemigo, saliendo del rango de visión (> 15 unidades).

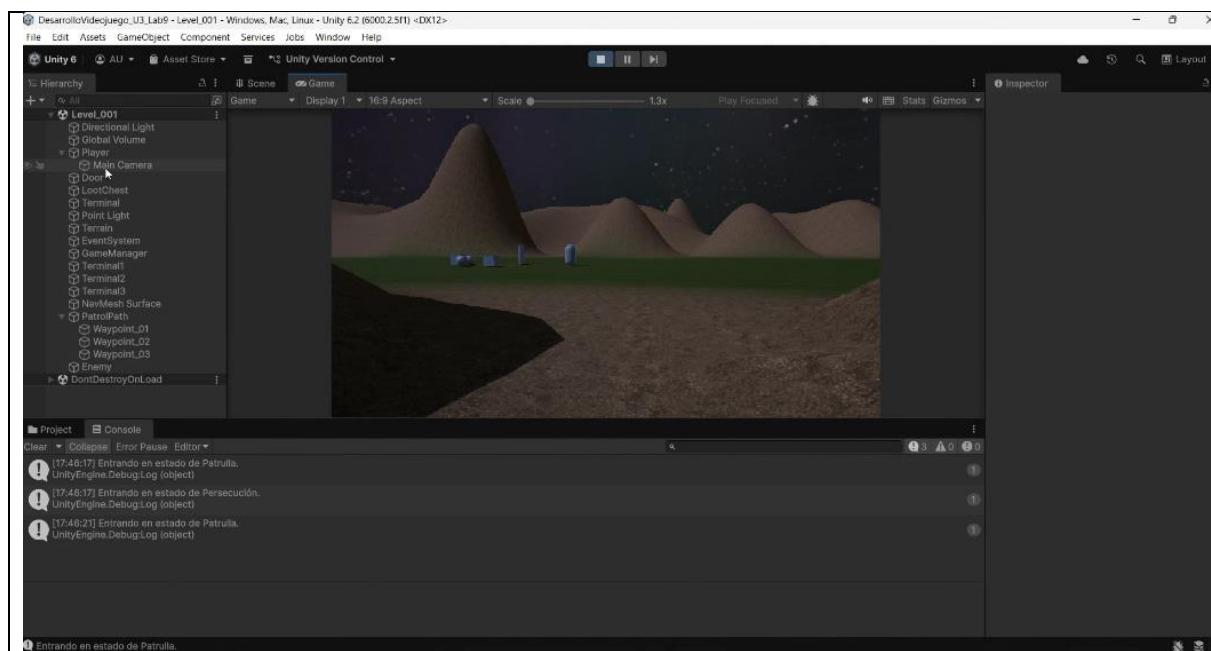
La IA detecta la pérdida del jugador y realiza transición automática al estado PatrolState.

En consola se observa nuevamente:  
Entrando en estado de Patrulla.

El agente reanuda su recorrido entre los waypoints desde el punto más cercano.

**Resultado:** El enemigo retorna correctamente a su comportamiento de patrulla al perder al jugador.

Comportamiento esperado: **Cumplido.**



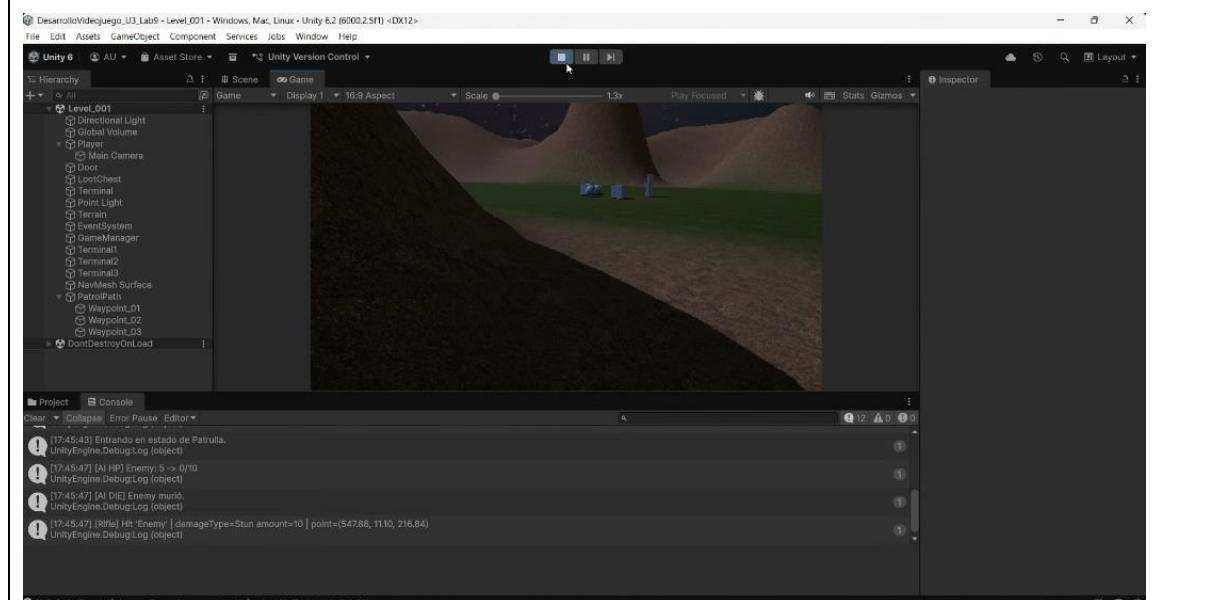
## Prueba de pérdida de vida

El jugador dispara su “rifle” al enemigo y muere a los 2 disparos (Ajustando dinámicamente los parámetros de daño).

Si se dispara al enemigo con stun pierde vida, y cuando llega a 0 muere.

**Resultado:** El enemigo muere.

**Comportamiento esperado: Cumplido.**



## Resultados técnicos

Caso de prueba	Estado esperado	Resultado	Observaciones
Inicio del juego	Patrulla activa	<input checked="" type="checkbox"/>	Ruta establecida correctamente
Detección del jugador	Cambio a ChaseState	<input checked="" type="checkbox"/>	Radio de detección sensible y fluido
Persecución	Movimiento continuo hacia el jugador	<input checked="" type="checkbox"/>	Sin saltos ni bloqueos
Pérdida de jugador	Retorno a patrulla	<input checked="" type="checkbox"/>	Transición suave sin errores
Estabilidad del NavMeshAgent	Movimiento natural	<input checked="" type="checkbox"/>	Ajuste dinámico correcto
Pérdida de vida de enemigo	Enemigo desaparece	<input checked="" type="checkbox"/>	Desaparece correctamente

### Bugs o incidencias detectadas

- No se detectaron errores de ejecución (NullReferenceException, IndexOutOfRangeException, etc.).
- No hubo bloqueos del NavMesh ni comportamientos erráticos.
- En casos de pendientes muy pronunciadas, el agente tiende a recalcular rutas, pero dentro de la tolerancia esperada.
- Se recomienda ajustar obstáculos y alturas del NavMesh si se desea evitar trayectorias complejas.

### Análisis del comportamiento

- El sistema de IA demostró ser estable, coherente y extensible.
- El uso del Patrón State permitió:
- Mantener el código desacoplado y modular.
- Añadir fácilmente nuevos estados (StunState, AttackState, etc.) sin modificar los existentes.
- Lograr un flujo claro de transiciones (Patrol → Chase → Patrol).

## Conclusión

Las pruebas confirman que la máquina de estados implementada para la IA cumple con todos los criterios funcionales y de diseño establecidos en la Guía Práctica #12.

El sistema es robusto, predecible y flexible, apto para escalar a comportamientos más complejos (ataques, detección por sonido, estados de alerta, etc.).