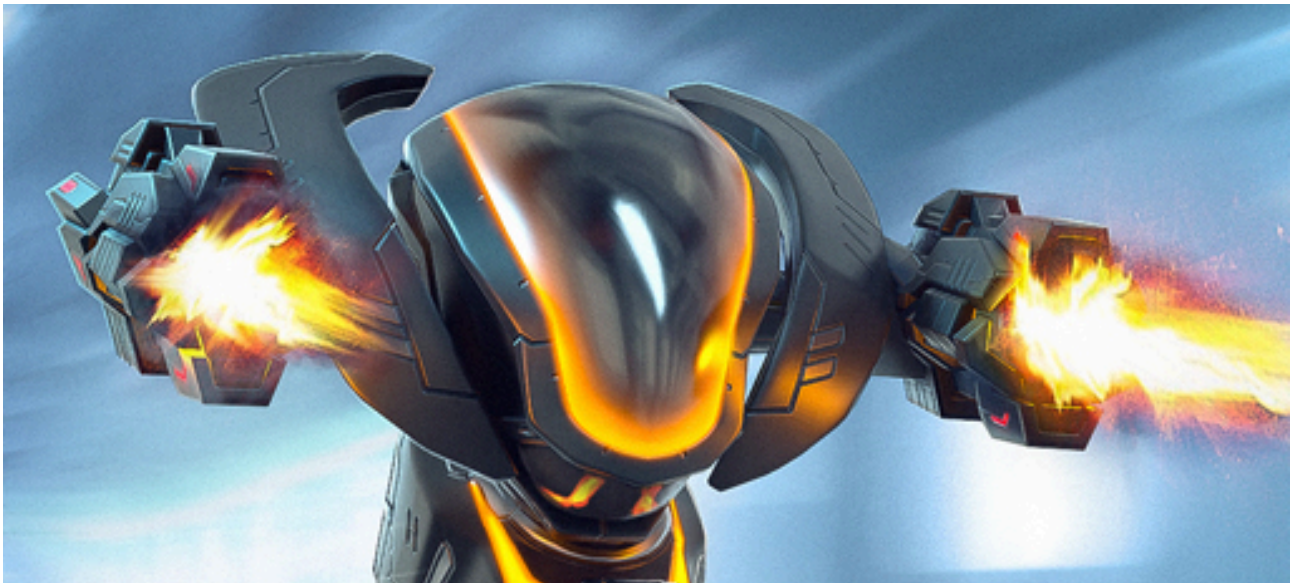
Neon Hunt: Stealth and Strike



ÍNDICE

Introducción

Este reto, que se llevará a cabo desde el **3 de febrero hasta el 27 de febrero**, tiene como objetivo el desarrollo de un videojuego de acción y sigilo en tercera persona, diseñado para ser jugado en **PC, móviles y tablets**. A lo largo de este período, los estudiantes trabajarán en la creación de un juego con un sistema de control adaptable, inteligencia artificial para enemigos, mecánicas de disparo asistido y una interfaz visual coherente con la temática futurista inspirada en *Tron* y *Mech Arena*.

El proyecto desafiará a los participantes a diseñar y programar un entorno interactivo donde el jugador deberá eliminar a todos los enemigos sin ser detectado. Para ello, se integrarán **elementos de navegación y detección mediante NavMesh**, un sistema de **disparo teledirigido con cooldown**, y una interfaz intuitiva que se ajuste a cada plataforma. Además, se fomentará el uso de **efectos visuales y sonoros** para mejorar la inmersión, utilizando recursos como **partículas, animaciones de recarga y sonido dinámico**.

Objetivo General

El objetivo de este reto es desarrollar un videojuego para un jugador que funcione en múltiples dispositivos (PC, móviles y tablets), utilizando Unity y aplicando conocimientos de programación, diseño de personajes y creación de entornos tridimensionales.

El juego será un **combate estratégico en un entorno futurista**, donde el jugador debe eliminar a todos los enemigos antes de ser detectado. Los enemigos patrullarán el área y estarán en constante búsqueda del jugador, pero no dispararán. La clave para ganar será usar el entorno a tu favor, ocultándote tras obstáculos y eliminando a los enemigos uno por uno. **El juego se inspira en la estética de la película *Tron* y en el diseño de personajes de *Mech Arena*.**

Se espera que los estudiantes integren técnicas de diseño de interfaces, optimización de aplicaciones para diferentes plataformas y el uso de inteligencia artificial para el comportamiento de los enemigos.

Requisitos Técnicos y Funcionales

Mecánica de Control

El personaje se manejará de manera intuitiva y adaptable a distintos dispositivos, ofreciendo una experiencia fluida tanto en PC como en móviles.

En PC, el jugador podrá moverse utilizando las teclas **W, A, S y D**, mientras que el apuntado se realizará con el ratón. Al mover el ratón en cualquier dirección, el robot girará en ese sentido, permitiendo un control preciso de la orientación. Para disparar, bastará con hacer clic con el botón izquierdo.

Si se juega con un mando, el control se ajustará a un esquema similar al de los juegos de acción en tercera persona. El **joystick izquierdo** permitirá moverse por el escenario, mientras que el **joystick derecho** controlará la dirección del personaje, haciendo que gire hacia donde se mueva el stick. El disparo se activará al presionar el gatillo **R2**, proporcionando una sensación más inmersiva y natural para los jugadores que prefieren este método de control.

En dispositivos móviles, el esquema se adaptará para una experiencia táctil. La pantalla mostrará **dos joysticks digitales**, con el izquierdo dedicado al movimiento y el derecho al giro del personaje. Además, habrá un **botón de disparo**, situado en una posición accesible para facilitar su uso sin comprometer la movilidad. Estos controles estarán visibles solo en dispositivos Android, asegurando que no interfieran en la experiencia de los jugadores de PC.

La cámara seguirá al personaje de manera dinámica, similar a la vista en juegos de acción en tercera persona. Mantendrá siempre una perspectiva que permita ver tanto al personaje como su entorno inmediato, asegurando una visión clara del campo de batalla. Al girar el ratón o el joystick derecho, la cámara acompañará el movimiento, facilitando la orientación y el apuntado en todo momento.

Movimiento del Personaje:

El desplazamiento del personaje estará basado en un **Rigidbody** con un **Sphere Collider**, siguiendo un enfoque similar al utilizado en el tutorial de Unity "Roll a Ball". Sin embargo, en lugar de una esfera visible, se aplicará un modelo de robot inspirado en los diseños de *Mech Arena*. Para ello, se deshabilitará el componente **MeshRenderer** de la esfera, dejando que esta actúe únicamente como un cuerpo físico que controla el movimiento.

Una característica clave del personaje será la presencia de una rueda en su parte inferior, la cual girará de manera dinámica según la velocidad alcanzada. Esto no solo aportará un efecto visual atractivo, sino que también simulará una sensación de inercia al moverse, haciendo que el desplazamiento sea más natural y fluido.

Además, el cuerpo del robot se inclinará en función de la velocidad y el ángulo de movimiento, añadiendo un efecto dinámico que refuerza la sensación de desplazamiento y dirección. Cuando el personaje gire hacia la derecha o la izquierda, su inclinación se ajustará de manera acorde para representar la fuerza centrífuga del giro. Este efecto puede

lograrse aplicando una rotación gradual basada en la **velocidad del Rigidbody** y su dirección.

Para inclinar el cuerpo del robot dependiendo del movimiento, podemos utilizar la **velocidad del Rigidbody (`rigidbody.velocity`)** y aplicarla a la rotación del personaje. Una forma de hacerlo es modificar el ángulo de inclinación en función de la dirección y la magnitud de la velocidad.

```
// Obtener la dirección de movimiento normalizada
Vector3 velocity = rigidbody.velocity;

if (velocity.magnitude > 0.1f) // Evitar inclinaciones cuando el personaje está quieto {
    // Calcular la inclinación en base a la dirección del movimiento
    float tiltAngle = Mathf.Clamp(velocity.x * tiltAmount, -tiltAmount, tiltAmount);

    // Aplicar la rotación en el eje Z (para inclinarse hacia los lados)

    elementoARotar.rotation = Quaternion.Euler(0, transform.eulerAngles.y, -tiltAngle);
}
```

Interfaz de Usuario (UI):

Al iniciar el juego, el jugador será recibido por un **menú principal** que servirá como punto de partida antes de sumergirse en la acción. Este menú contará con un diseño limpio y accesible, incluyendo dos botones principales: **"Jugar"**, que llevará directamente a la partida, y **"Salir"**, que permitirá cerrar la aplicación. Opcionalmente, podrá incluirse un botón de **"Opciones"**, donde se ofrecerán ajustes personalizables, como el volumen del sonido o la sensibilidad de los controles.

Una vez dentro del juego, la interfaz de usuario proporcionará información clave en todo momento. En pantalla se mostrará el **número de enemigos restantes**, permitiendo al jugador conocer su progreso en la partida. Además, se reflejará el **tiempo restante** antes de que la misión termine y la cantidad de **puntos acumulados**, ofreciendo una referencia clara del desempeño del jugador.

El diseño de la interfaz deberá adaptarse de manera fluida a diferentes dispositivos, asegurando que los elementos sean fácilmente accesibles tanto en PC como en pantallas táctiles. En dispositivos móviles, los controles y la información en pantalla estarán distribuidos de manera estratégica para evitar que interfieran con la jugabilidad, mientras que en PC se aprovechará la mayor disponibilidad de espacio en pantalla para ofrecer una experiencia cómoda y bien organizada.

Enemigos y Comportamiento:

Los enemigos del juego patrullarán constantemente el área, utilizando **NavMesh** para moverse de manera fluida y realista por el escenario. Su comportamiento estará diseñado para detectar al jugador si este entra en su campo de visión, tal como se explica en los tutoriales de Unity Learn: Movimiento de Enemigos con NavMesh y Tutorial de enemigos y detección - Jhon Lemon. Estos enemigos no dispararán, pero estarán en una búsqueda constante del jugador, activando el estado de "Game Over" si logran detectarlo.

Para evitar ser visto, el jugador podrá esconderse detrás de obstáculos, como cajas o muros, aprovechando la topografía del escenario para mantenerse fuera del alcance visual de los enemigos. Esta mecánica se inspira en el comportamiento de los fantasmas en el tutorial mencionado. Finalmente, el juego concluye cuando el jugador elimina a todos los enemigos presentes en el escenario, completando así su misión con éxito.

Disparo y Mecánica de Eliminación de Enemigos:

El personaje tendrá la capacidad de disparar, y el sistema de disparo será asistido. Al realizar un disparo, se utilizará un área de detección basada en la funcionalidad **Overlap Sphere** de Unity (Documentación de Overlap Sphere) para identificar si hay enemigos presentes en su rango. En caso de que varios enemigos estén dentro del área, el disparo se dirigirá automáticamente hacia el enemigo más cercano al centro de la pantalla, asegurando precisión y efectividad en el ataque.

Los disparos se representarán mediante proyectiles modelados, que estarán diseñados para emitir un efecto visual de humo mientras avanzan hacia el objetivo. Estos proyectiles serán **teledirigidos**, utilizando un script que les permitirá seguir al enemigo seleccionado. Si el enemigo logra esconderse detrás de un obstáculo, como una caja o parte del escenario, el proyectil impactará contra el objeto y explotará, produciendo un efecto visual y reforzando la inmersión del jugador.

Entre cada disparo habrá un tiempo de **cooldown**, durante el cual no será posible disparar nuevamente. Si el jugador intenta disparar durante este periodo, recibirá una notificación clara de que el arma está en recarga. Esto se podría implementar mediante un sonido breve, un icono sobre el arma, o un cambio temporal en el color del botón de disparo.

El cooldown estará representado de manera visual y sonora, incluyendo una animación de recarga y un efecto auditivo. Durante el disparo, el ataque será reforzado con movimientos de cámara ligeros (*screen shake*) y efectos de partículas llamativos, que utilizarán el paquete War FX. Dado que los enemigos serán eliminados con un solo disparo, los efectos visuales y sonoros enfatizarán la potencia del ataque, asegurando una experiencia inmersiva y satisfactoria para el jugador.

Estilo Visual y Efectos:

- **Inspiración en *Tron*:** El estilo visual del juego debe recordar a la estética de la película *Tron*, con luces de neón y efectos de estela cuando los personajes se mueven.
- **Detalles de implementación:** Al avanzar los personajes y enemigos, se dejará un rastro luminoso en el suelo.
- **Enlaces de inspiración:**
  - Estilo visual de Tron.
  - Estilo visual de Tron 2.

Sonido

Sonidos y Música

El diseño sonoro del juego será una parte clave para la inmersión del jugador, ayudando a reforzar la atmósfera futurista y la jugabilidad. A continuación, se definen las categorías de sonidos necesarios y recursos para obtenerlos:

Categorías de Sonidos

1. **Sonido de Disparo:**
   - Representará la potencia del ataque, con un efecto de eco y energía futurista.
   - Complementado con un sonido de explosión al impactar contra enemigos u obstáculos.
2. **Sonido de Movimiento:**
   - Sonido leve asociado al giro de la rueda del personaje y al movimiento de los enemigos (por ejemplo, un motor o mecanismo futurista).
3. **Música de Fondo:**
   - Una pista dinámica que acompañe al jugador durante el gameplay. La música debe reflejar tensión, incrementando su intensidad cuando el jugador esté en peligro o queden pocos enemigos.
4. **Sonido de Detección:**
   - Un sonido de alarma claro que indique que un enemigo ha detectado al jugador, acompañado por una breve pausa en la música para enfatizar el impacto.
5. **Sonido de Cooldown:**
   - Un sonido breve y sutil que refuerce la notificación visual cuando el jugador intente disparar durante el tiempo de recarga.
6. **Otros Sonidos Ambientales:**

- Efectos como explosiones al fallar un disparo, impactos de proyectiles contra el escenario, y sonidos de interacción con objetos como cajas o paredes.

Recursos para Sonidos y Música

Los sonidos y la música se pueden obtener de los siguientes recursos:

- **Unity Asset Store:**
  1. Free Sound FX Pack
  2. War FX (efectos sonoros y partículas)
- **Webs con Sonidos Libres:**
  1. Freesound
     - Gran biblioteca de efectos de sonido libres, ideal para disparos, explosiones y sonidos ambientales.
  2. Zapsplat
     - Recursos gratuitos y premium para efectos de sonido y música.
  3. Free Music Archive
     - Música libre para videojuegos, perfecta para seleccionar una pista de fondo.
  4. SoundBible
     - Efectos sonoros gratuitos bajo licencias Creative Commons.
  5. Pixabay Sounds
     - Banco de sonidos gratuitos para efectos y ambientaciones.

Escenario:

- El escenario se construirá utilizando **ProBuilder** y formas básicas de Unity, con texturas que se alineen con el estilo visual del juego.
- **Enlaces de tutorial:**
  - Introducción a ProBuilder.
- **Detalles:**
  - El escenario tendrá diferentes alturas y rampas, lo que permitirá al jugador usar la topografía a su favor para evitar ser detectado.

Objetivos y sistema de puntuación

El objetivo principal del juego es eliminar a todos los enemigos del escenario antes de que el tiempo límite de **7 minutos** se agote. El jugador deberá moverse estratégicamente, aprovechar los obstáculos para ocultarse y disparar con precisión para completar la misión. Cada enemigo eliminado otorgará **2 puntos**, pero cada disparo realizado restará **2 puntos**, incentivando el uso inteligente de los recursos.

Además, el tiempo restante al finalizar la partida se convertirá en puntos adicionales, sumando **10 puntos por cada segundo que sobre**. Esto añade un componente competitivo al juego, recompensando tanto la eficiencia como la precisión en la ejecución.

La partida terminará de inmediato si un enemigo detecta al jugador, lo que obliga a tomar decisiones rápidas y cuidadosas.