Amazing Challenge

Amazing Games Studio

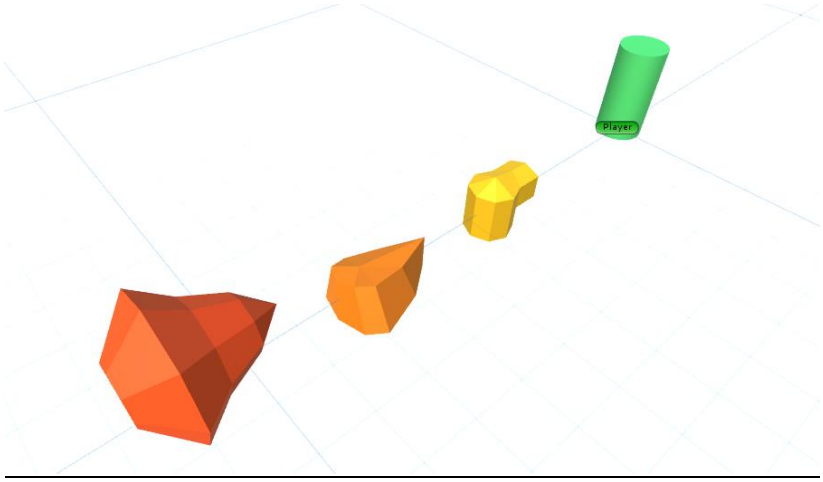
1. Shooter Survival

Este es un reto que busca identificar las aptitudes del programador al realizar un videojuego utilizando Unity3D. El programador debe ser capaz de:

1. Identificar deficiencias en el código y corregirlas.
2. Implementar todas las mecánicas expuestas en el documento de diseño.
3. Mantener un código ordenado y fácil de entender.

Desarrollar el proyecto que se encuentra en la carpeta **AmazingChallenge** utilizando Unity 2018.1.x. Los recursos son referenciales, sentirse en libertad de agregar efectos visuales o de sonido. No hay necesidad de hacer un build del juego, solo entregar un *zip* con el proyecto terminado.

El juego consiste en sobrevivir los ataques de los enemigos y derrotarlos.



Jugador

El jugador debe ser capaz de moverse a lo largo del plano de juego.
Al hacer click en la pantalla con el botón izquierdo del mouse el jugador cambiará su orientación para mirar en la dirección donde se hizo el click. Además de cambiar su orientación el jugador disparará una bala.

Enemigos

Los enemigos deben desaparecer cuando colisionan con una bala del jugador.
Hay 3 tipos de enemigos con los siguientes comportamientos.

- Enemigo A (Amarillo): No se mueve. Dispara cada 5 segundos.
- Enemigo B (Anaranjado): No se mueve. Rota aleatoriamente de forma instantánea cada 3 segundos y dispara.
- Enemigo C (Rojizo): Rota mirando al jugador cada 3 segundos y dispara.

Balas

Las balas son proyectiles que viajan en una sola dirección. La dirección en la que viajan se basa en la orientación del disparador.
Hay dos tipos de balas:

- **Balas de jugador:** Las balas solo deben colisionar con los enemigos.
- **Balas de Enemigo:** Las balas solo deben colisionar con el jugador.

Las balas NO deben colisionar con otras balas.

Condición de Victoria

El juego termina cuando el jugador elimina todos los enemigos o cuando colisiona con una bala enemiga.
En caso de ser victorioso una pantalla deberá aparecer con el mensaje: “Felicitaciones”
En caso de ser derrotado una pantalla deberá aparecer con el mensaje: “Perdiste”
*Implementar las pantallas utilizando el sistema de UI de Unity.

2. Optimización

Optimizar el siguiente código en C#. Considerar el uso de memoria, performance y legibilidad del código.

```
int main()
{
    //*Es posible modificar tipos de variables
    //*Optimizar el performance y el uso de memoria
    //Mantener un codigo organizado y facil de entender

    ///Asumir que los valores de verdad de las misiones pueden ser modificados
    bool mision1EsObligatoria = true;
    bool mision2EsObligatoria = false;
    bool mision3EsObligatoria = true;
    bool mision4EsObligatoria = true;

    bool mision5EsObligatoria = false;
    bool mision6EsObligatoria = false;
    bool mision7EsObligatoria = true;
    bool mision8EsObligatoria = true;

    int totalRecompensa = 1;
    //Por cada mision Obligatoria
        //multiplicar el totalRecompensa por 2

    print(totalRecompensa);//En este ejemplo: totalRecompensa == 32
    return 0;
}
```