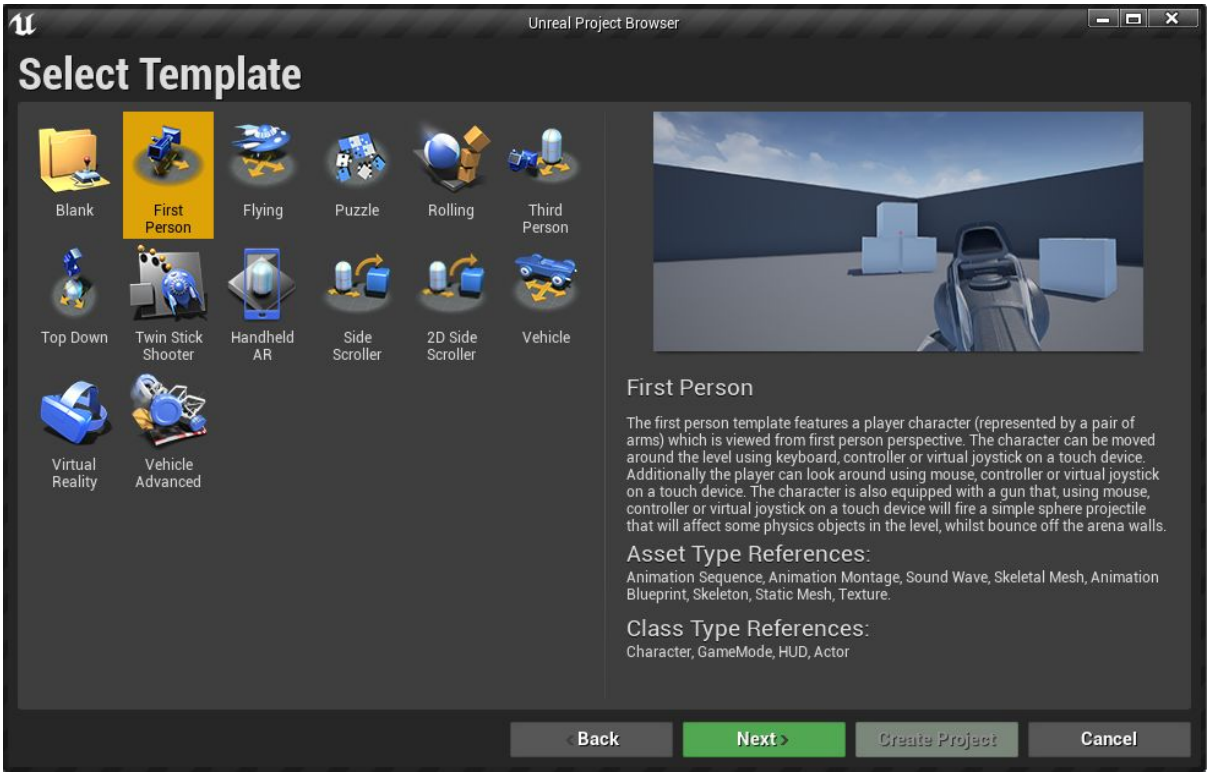
Blackmouth Games

Prueba de gameplay - Servidor dedicado

Para esta prueba es recomendable empezar desde el proyecto base “**FirstPerson**”.



Las secciones a completar tienen un requerimiento de lenguaje (**C++/BP**) . En aquellas donde se indique que debe usarse **C++** , lógicamente se pueden asignar referencias a assets, exponer valores para poder modificar desde Blueprint y modificar valores existentes (colisión, física, etc…).

En la prueba se valorarán varios aspectos, tales como la limpieza y legibilidad del código o de los nodos de blueprint, la configurabilidad de los valores expuestos. También se valorará el uso adecuado de comentarios, donde fueran necesarios.

La prueba debe ir acompañada de un archivo **README** que explique la entrega, para facilitar su corrección y entender las motivaciones de las soluciones dadas.

En el caso de que no se pueda completar una sección, se valorarán también soluciones parciales, y se tendrá en cuenta el contenido del README incluido con la entrega. No conseguir completar todas las secciones de la prueba no significa por defecto que se falle la prueba completa.

Sección 1 - [Blueprint/C++]

Configurar **FirstPersonCharacter** y **FirstPersonProjectile** para replicación con **Servidor Dedicado**.

Como objetivo de esta sección, ambos personajes debe poder:
- Ver en su pantalla la animación de disparo del otro jugador.
- Ver el proyectil resultante del disparo del otro jugador.
- Añadir algún elemento visual como “cuerpo” del jugador, para poder discernir mejor su posición. Esto puede ser una simple forma geométrica.

Sección 2 - [C++]

Crea un **componente de Vida** y añadirlo al personaje principal. Este componente debe ser diseñado para multijugador, con la vida replicada desde el servidor. Las funcionalidades mínimas serán:
- Dañar al jugador una cantidad específica.
- Curar al jugador una cantidad específica.
- Una función que devuelva la vida normalizada de 0 a 1.
- Una función que devuelva la vida restante, sin normalizar.
Estas funcionalidades deberán ser accesibles desde blueprints también.

Se podrán implementar funciones / variables / eventos adicionales a discreción del programador si se considera necesario, o como complemento a futuras partes de la prueba.

Sección 3 - [C++]

Implementar que el **disparo dañe al jugador enemigo.**

Cuando la vida del jugador llegue a 0, este debe ser eliminado. Una vez pasados 10 segundos desde la eliminación debe respawnear en un punto aleatorio del mapa (sin aparecer dentro de una pared). Para facilitar esta parte de la tarea (respawn sin colisionar) se pueden usar blueprints adicionales colocados en el nivel si fuera necesario.

Sección 4 - [Widget Blueprint]

Crear y añadir un **widget de barra de vida**. El widget deberá tener eventos para ocuparse de mostrar visualmente los daños y curas que reciba el jugador.

Sección 5 - [C++]

Implementar un componente que contenga una **habilidad ofensiva**, una esfera centrada en el dueño que crece hasta un máximo mientras él mantenga pulsada una tecla. Tras soltar la tecla, hace daño a las entidades enemigas que se encuentren dentro. Una vez usada, esa habilidad entra en cooldown durante X segundos.

Sección 6 - [Widget Blueprint]

Crear un widget que muestra una **barra de progreso de esa habilidad**, estando vacía si no se ha empezado a cargar, y llena si ha llegado a su carga máxima. Así como también un número indicando el cooldown restante.