**Documento de Diseño**

**Project RUN**

# Concepto inicial

El concepto con el que se ha iniciado el proyecto consiste en el desarrollo un videojuego de realidad virtual basado en la idea de un *infinite runner*.

Para ello, el jugador, en una posición estática, deberá de esquivar diversos obstáculos y recoger tesoros a medida que estos se acercan a su posición. Entrando en detalles se ha definido:

* + **El jugador:**
    - Se mantendrá estático para evitar mareos en jugadores no experimentados.
    - Hará uso de los movimientos de todo su cuerpo para superar los obstáculos (cabeza y ambas manos).
    - Se le darán 3 vidas
      * Cada vez que colisione o falle uno de los obstáculos perderá una de estas vidas.
      * Se mostrará el número de vidas restantes en todo momento.
    - Una vez el jugador se quede sin vidas, se le dará una puntuación dependiente del tiempo que ha estado con vida y del número de tesoros recogidos.
  + **Los obstáculos:**
    - Se generarán aleatoriamente mientras el jugador siga con vida.
    - Cada uno de ellos obligará al usuario a moverse en una dirección o forma diferente (agacharse, moverse a un lado, golpear un objeto, …).
    - Su velocidad y la distancia entre ellos dependerá del tiempo transcurrido, de manera que resulte más complicado a medida que el jugador pase más tiempo en el nivel.
  + **Los tesoros:**
    - La generación de los tesoros será dependiente de los obstáculos, de manera que cada uno de ellos tendrán predefinidos varios puntos en los que podrán aparecer o no estos tesoros.
    - La posición de los tesoros deberá de suponer un reto extra para la habilidad del jugador.
    - El jugador podrá coleccionar estos tesoros entrando en contacto con ellos.

# Desarrollo primera fase

Hemos decidido hacer uso de esta primera fase para desarrollar una versión mínima del ciclo jugable con un acabado lo más cercano posible a lo que se esperaría de la versión final. En ella el jugador podrá esquivar dos tipos de objetos que se acercarán a su posición de forma lineal, y recibirá daño por cada colisión con estos objetos.

Entre los sistemas desarrollados, podríamos destacar el sistema encargado de gestionar el **ciclo de juego**, que va desde le generación de obstáculos hasta el sistema de reinicio del juego.

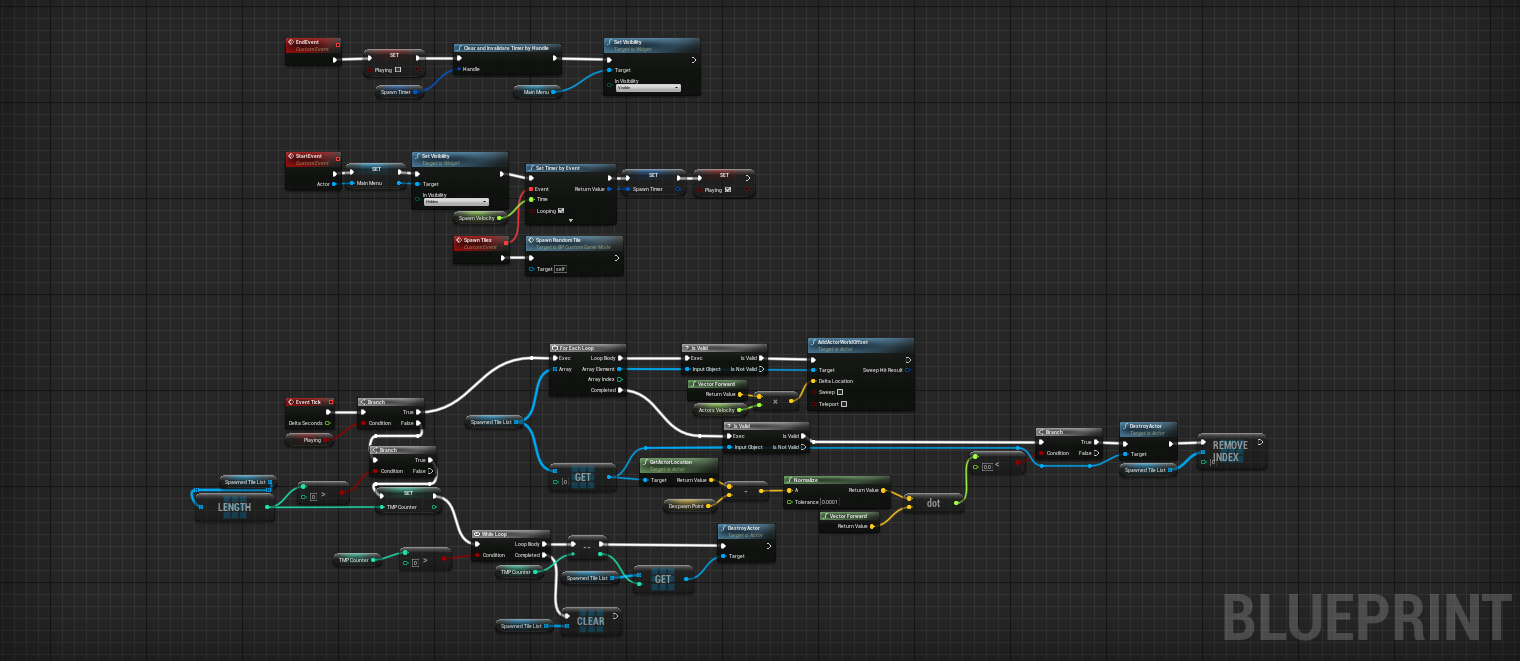


Ilustración 1. Ciclo jugable

Para la **generación de obstáculos**, como podemos observar, se aprovechan los temporizadores de eventos para ejecutar la función encargada de instanciar aleatoriamente uno de los objetos que componen la lista de obstáculos.

A la hora de **reiniciar el juego**, han de ejecutarse una serie de instrucciones para vaciar las listas de objetos generados, así como reiniciar los temporizadores ejecutados y activar la interfaz que hace de menú de inicio.

Otro de los sistemas claves del desarrollo es el **sistema de salud**. Este sistema se encarga de ejecutar la función de recepción de daño con cada colisión, restando la vida definida por el objeto colisionado y dando feedback al jugador, tanto de manera háptica, como actualizado la interfaz del jugador con la reducción de vida.

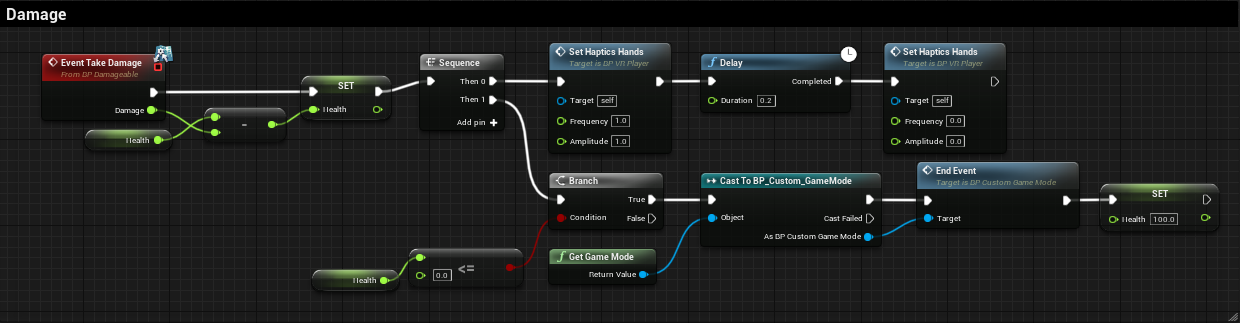


Ilustración 2. Gestión del daño

# Desarrollo segunda fase

Para esta segunda fase de desarrollo hemos decidido centrar los esfuerzos en completar los sistemas que quedaron pendientes de implementación en la fase anterior.

Entre los sistemas desarrollados, podríamos destacar el sistema encargado de gestionar las puntuaciones, además de la gestión de los tesoros.

En cuanto al sistema de **cálculo de puntuaciones**, hemos tenido que definir dos variables, una de ellas dedicada a registrar la puntuación del último intento, y otra que almacenará la puntuación máxima local. Como punto extra, hemos aprovechado las funciones de guardado del motor para almacenar de manera local la puntuación máxima del usuario y cargarla al inicio del juego.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración . Cálculo de puntuaciones

Para la **gestión de los tesoros** hemos decidido aprovechar la función de definición de clases con C++, de este modo hemos conseguido redefinir la manera en la que gestionábamos la creación de obstáculos, y crear un sistema más robusto y sencillo de gestionar.

Algunos de los cambios más importantes realizados con el paso al nuevo sistema:

* El uso de las funciones de recepción de daño propias del jugador, en lugar de usar nuestras funciones anteriormente definidas en *blueprints*. Este cambio ha venido impulsado por la imposibilidad de acceder a las funciones definidas en *blueprints* dentro del código C++.
* La gestión de los eventos de colisiones también ha pasado a gestionarse dentro de nuestra nueva clase, de este modo garantizamos que cada uno de los elementos de colisión que contiene el objeto “Base” de nuestros obstáculos ejecutará la función encargada de dañar al jugador.
* Otro punto clave de este nuevo sistema es la gestión de los tesoros, para ello se han definido una serie de propiedades editables desde los *blueprints*, que definen tanto las posibles posiciones que podrían ocupar, como el propio objeto que se generará en la posición deseada. De este modo conseguimos la flexibilidad deseada a la hora de crear estos elementos.Texto

  Descripción generada automáticamenteTexto

  Descripción generada automáticamente

Ilustración 4. Código C++