# SAVEGAME

Nos quedamos pq no sabiamos como diferenciar si un actor se spawneaba dinámicamente o en la carga del nivel.

En los Pawns hay un campo autoposeess player. Queremos esto, ver cuando esta placed o spawneado.

Buscamos entonces Pawn.h autopossessAI

Text

Description automatically generated

bStartup lo pone a true cuando esta inicilizando actores.

Nos podemos ir al savegame component

Para los actores del escenario

Text

Description automatically generated

Si no, estamos spawneando

(…)

Y ya esta 18:55 podriamos guardar el estado de cuantos enemigos spawneados hay

Podemos gestionar los componentes, que queremos guardar sus datos de los actors records por si queremos guardar cosas en el inventario o algo así

Enseña sobre las 19:27 – 19:33 como crear un BREAKPOINT cuando se modifica una variable

# NIVELES

20:35 -> queremos ver niveles

Window – levels

Text

Description automatically generated

Tenemos un nivel Persistente “Levels” y podemos añadir subniveles que podemos cargar.

**World Partition** lo podemos separar en capas (para artistas y programadores).

Un mismo nivel lo podemos separar en capas (audio, render, logica) para poder trabajar en paralelo entre artistas y programadores

Vamos a crear un nivel nuevo, y copiamos lo de la plantilla para que no esté vacio.

MCV > LEVELS > Y crea uno ahí (LV\_StreamingTest)

Ahora podemos ir a window levels y

A picture containing text, screenshot, monitor, screen

Description automatically generated

Nos sale el persisten level.

Graphical user interface, text, application

Description automatically generated

Creamos uno vacío

LV\_StreamingTestLogic

Y se pone como subnivel, podemos modificarlos con los ojos para ver o ocultar cosas

Podemos **mover actores de un nivel a otro**:

* Seleccionamos actores
* Click derecho en el nivel destino
* Move Selected Actors to Level

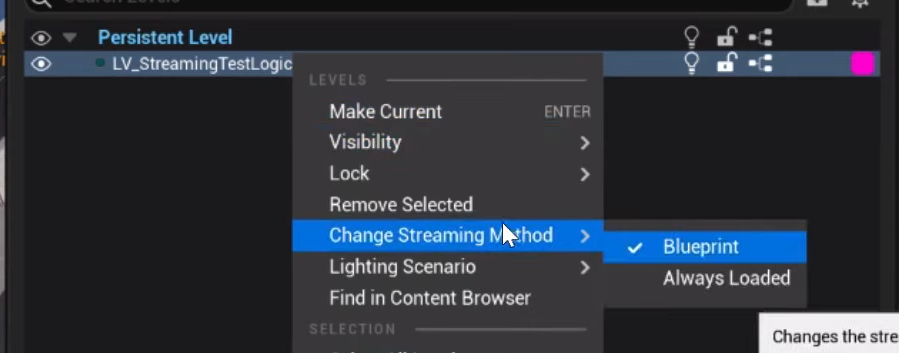
Doble click para **activar un nivel** u otro.

Graphical user interface, text, application, chat or text message

Description automatically generated

## Cargar niveles

Los niveles pueden cargarse de dos formas

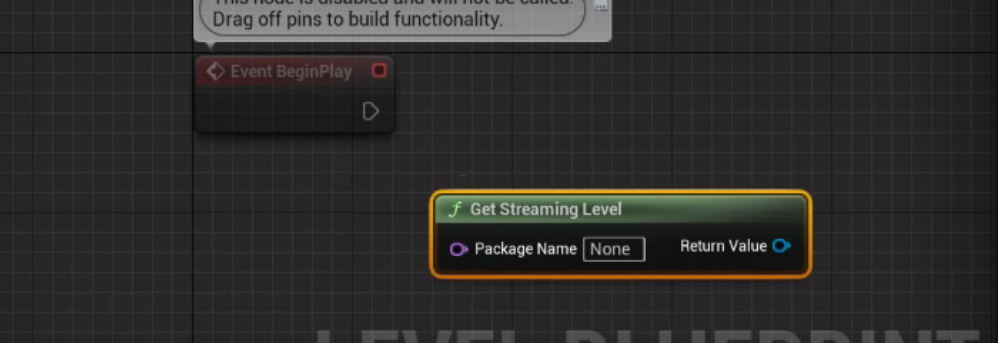


**Always Loaded**: normal, siempre como dependencia del persistent. Pero no nos sirve si queremos cargar los niveles sobre la marcha

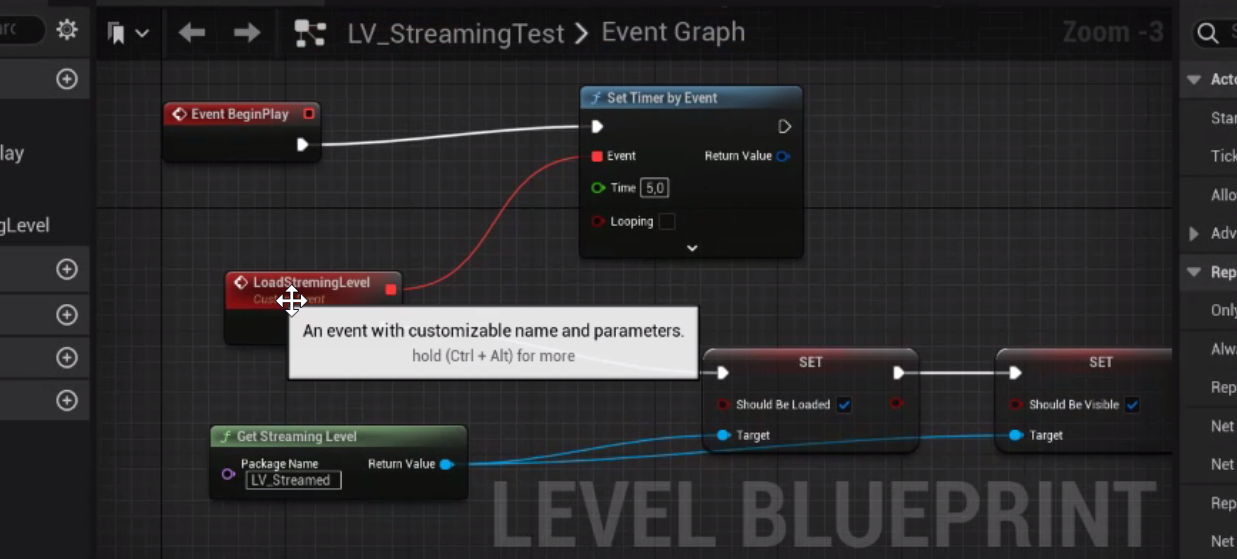
**BP**: aparece solo lo del persistent (lo que hemos movido al nivel de Lógica no).

* Cargarlo por C++
* Cargarlo por BP:

Cogemos el nivel por un string

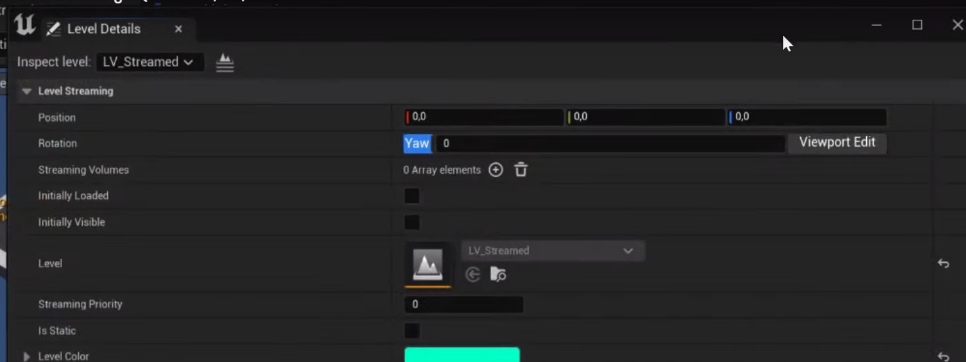


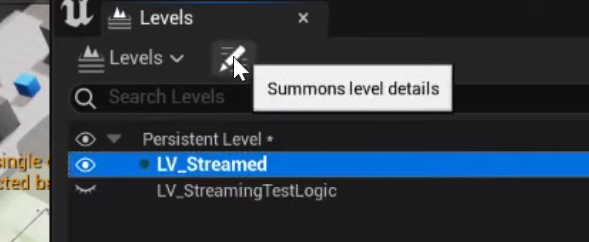
### Cargar niveles a mano



Después de 5 segundos, se cargará el nivel. **ShouldBeLoaded** and **ShouldBeVisible**.

### Cargar nivel 2 cuando pasa por collider:

Podemos crear otor nivel LV\_Streamed (duplicar el que teniamos) y añadirle un Offset, para moverlo todo X metros.



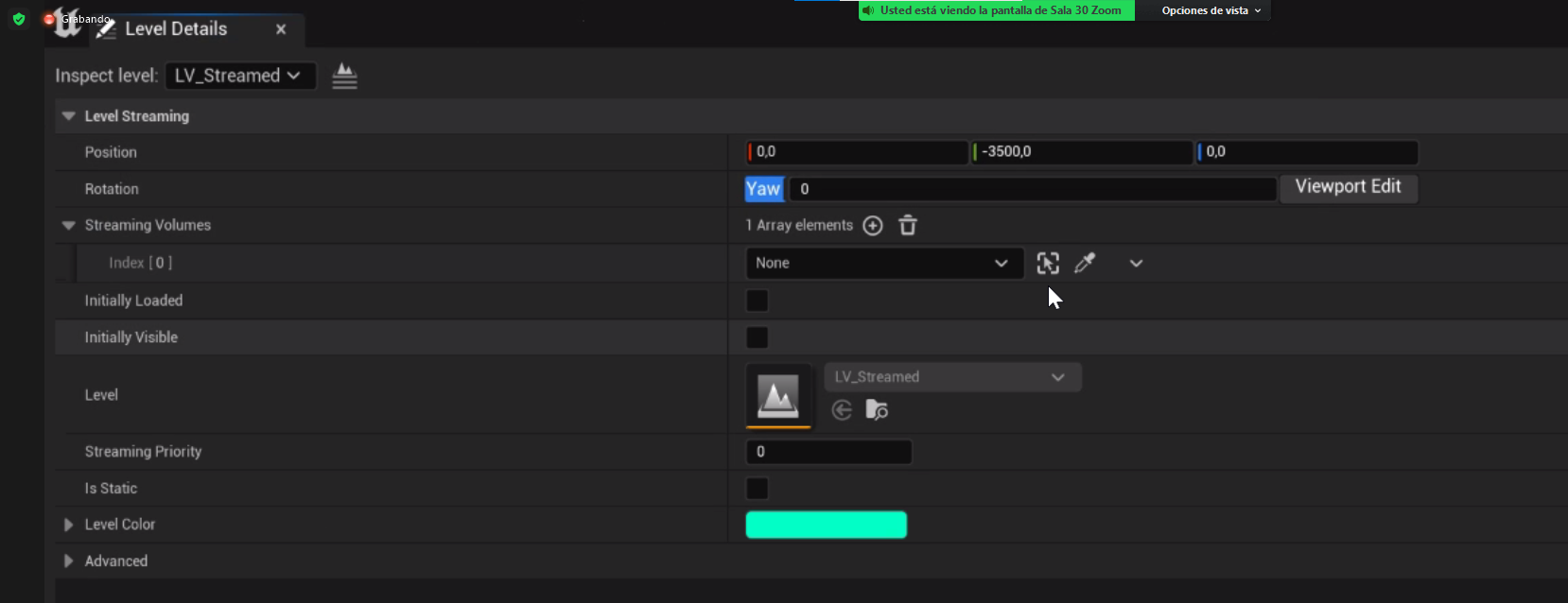
En el Persistent le puedo añadir un actor de tipo **LevelStreamingVolume**: mientras estés dentro del volumen hay un nivel que tiene que estar cargado. Nos ahorra la lógica. La idea es que ocupe el nivel 2 y el final del 1

A screenshot of a computer

Description automatically generated with low confidence

Así cuando llego a esa parte se tiene que cargar el nivel 2.

Streaming volumes



Y aquí añadimos el nivel

Window > Levels > lápiz (ajustes)

A screenshot of a computer

Description automatically generated

### Cargar niveles desde C++

Para cargar un nivel por ejemplo desde el Init de la GameInstance:

A screen shot of a computer

Description automatically generated

Una vez esté cargado debemos buscarlo y mostrarlo

A computer screen with text

Description automatically generated

Desde Blueprint es mucho mas sencillo:

A screenshot of a computer program

Description automatically generated