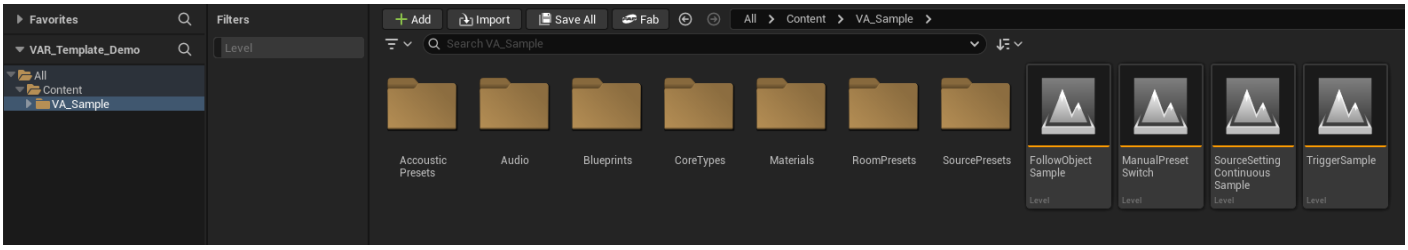
echotope met Unreal Engine

De Unreal Engine template is specifiek gemaakt in versie **5.7.4**, maar zou ook moeten werken in nieuwere 5.7 varianten, en wellicht ook wel in nieuwere 5.x varianten. In oudere versies gaat het importeren van de *Blueprints* wel goed, maar zijn de *Maps* leeg / corrupt.

Import & Check

1. Zorg dat je een bestaand Unreal project hebt, of maak een nieuw project aan in de juiste versie.
2. Pak de template ZIP file uit zodat de *VA_Sample* folder in de *Content* folder van je project zit *([PROJECT_FOLDER]/Content/VA_Sample)*

Vervolgens zou je 4 maps moeten hebben in de VA_Sample folder, en een aantal extra mappen.



Elke Map bevat een klein voorbeeldje wat gemaakt is met Blueprints die je kunt vinden in de Blueprints folder.

> **Als je de 4 Maps (hier rechts zichtbaar) niet kunt zien, of niet kan openen (of ze zijn leeg), dan is de Unreal Versie wellicht niet compatible, of is er iets fout gegaan bij het uitpakken.**

De overige mappen zijn:

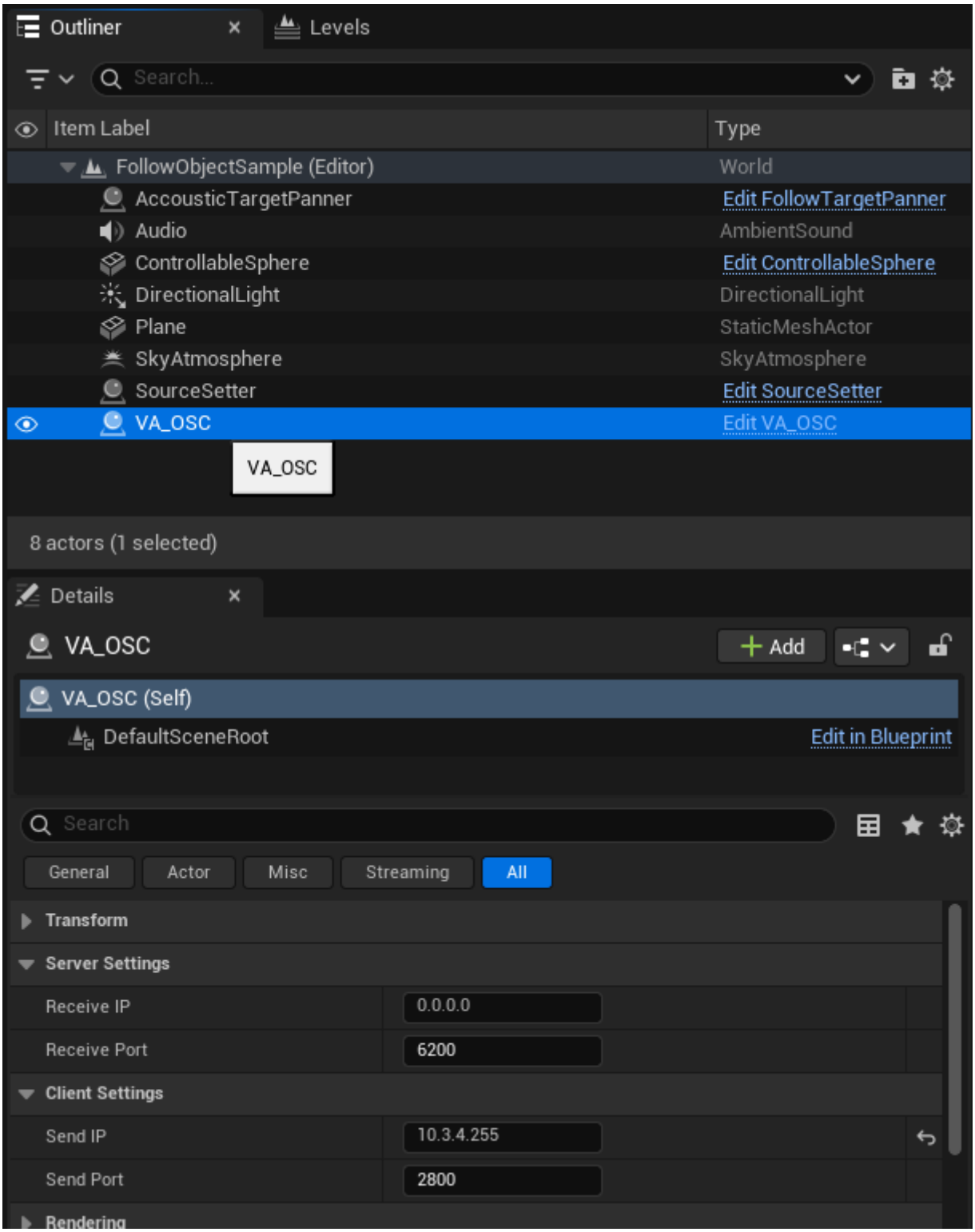
- CoreTypes
  - Deze map bevat alle basis types voor het opslaan & communiceren met het echotope Systeem.
    1. Accoustic / Room / Source **Setting**: een *enum* die het handiger maakt om een parameter naam te kunnen selecteren (met een drop-down)
    2. Accoustic / Room / Source **Settings**: de *Data Asset* class die alle parameters definieert, en ook een functie bevat om ze individueel aan te spreken (Break).
- Accoustic / Room / Source Presets

    - Deze folders bevatten *Data Assets* die zijn gebaseerd op de corresponderende class in CoreTypes, zodat het makkelijker wordt om te wisselen tussen zelf-ingestelde sets van alle waardes van een source / room / accoustic.
- Audio
    - Bevat wat voorbeeld audio (uit de standard assets van Unreal)
- Materials
    - Bevat wat voorbeeld materials die gebruikt worden in de Maps.

Workflow

De manier waarop je dit in je eigen Maps kan integreren, is door de **VA_OSC** Blueprint in je scene te plaatsen, en dan in je eigen Blueprints (of door gebruik te maken van een van de voorbeelden) functies van *VA_OSC* aanroept om informatie naar echotope te sturen.

Zorg ervoor dat de juiste **Send IP** & **Send Port** van het echotope systeem zijn ingevuld. Je kan hier ook een **Receive Port** instellen (de Receive IP moet meestal 0.0.0.0 blijven) om ook dingen te kunnen ontvangen, en die berichten worden dan geprint op het scherm (bijv. om te testen of je wel de juiste informatie stuurt)

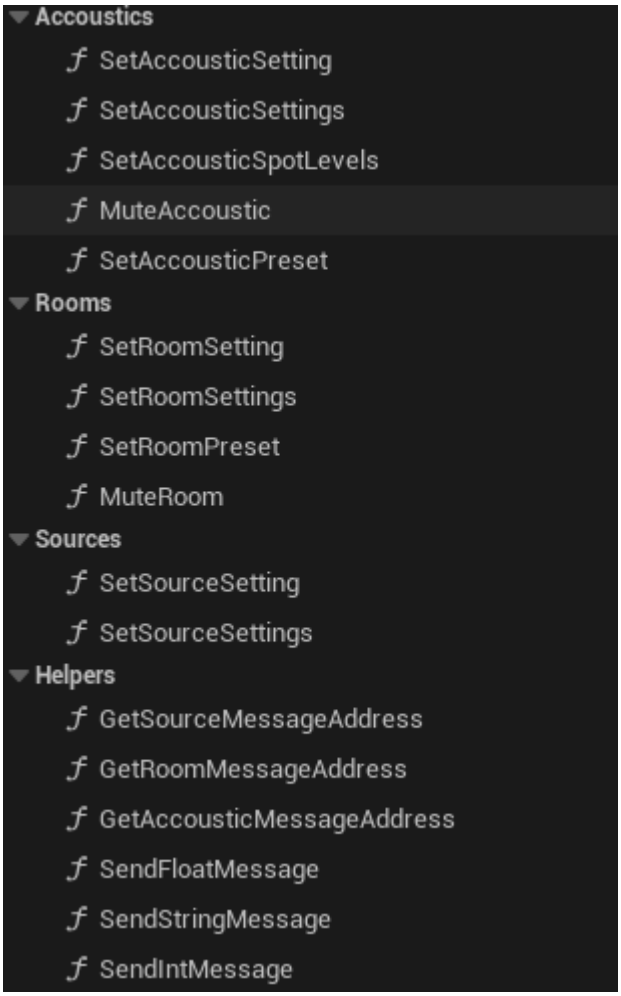


Om echt effectief te kunnen werken met echotope zal je wel wat kennis moeten hebben van *Blueprints*, dus overweeg daar ook wat starter-tutorials voor te bekijken. Hier zijn enorm veel bronnen vool, die ook constant veranderen, dus voegen we hier niet allemaal in. Hier is een voorbeeld van een preview van een uitgewerkte Udemy course (Blueprints for Beginners in Unreal Engine 5.6 - Learn Blueprints in 30 Minutes).

VA_OSC Functies

De VA_OSC Blueprint bevat functies die met het echotope systeem praten. Deze zijn opgedeeld zodat het zo makkelijk mogelijk wordt om specifieke dingen te doen: *enkele waardes aanpassen* (Set...Setting), *alle waardes aanpassen* (Set...Settings), *mute/unmute* (Mute...), & *preset switching* (Set...Preset). Zie het *echotope protocol* voor meer informatie over wat elke parameter precies doet met het geluid / systeem.

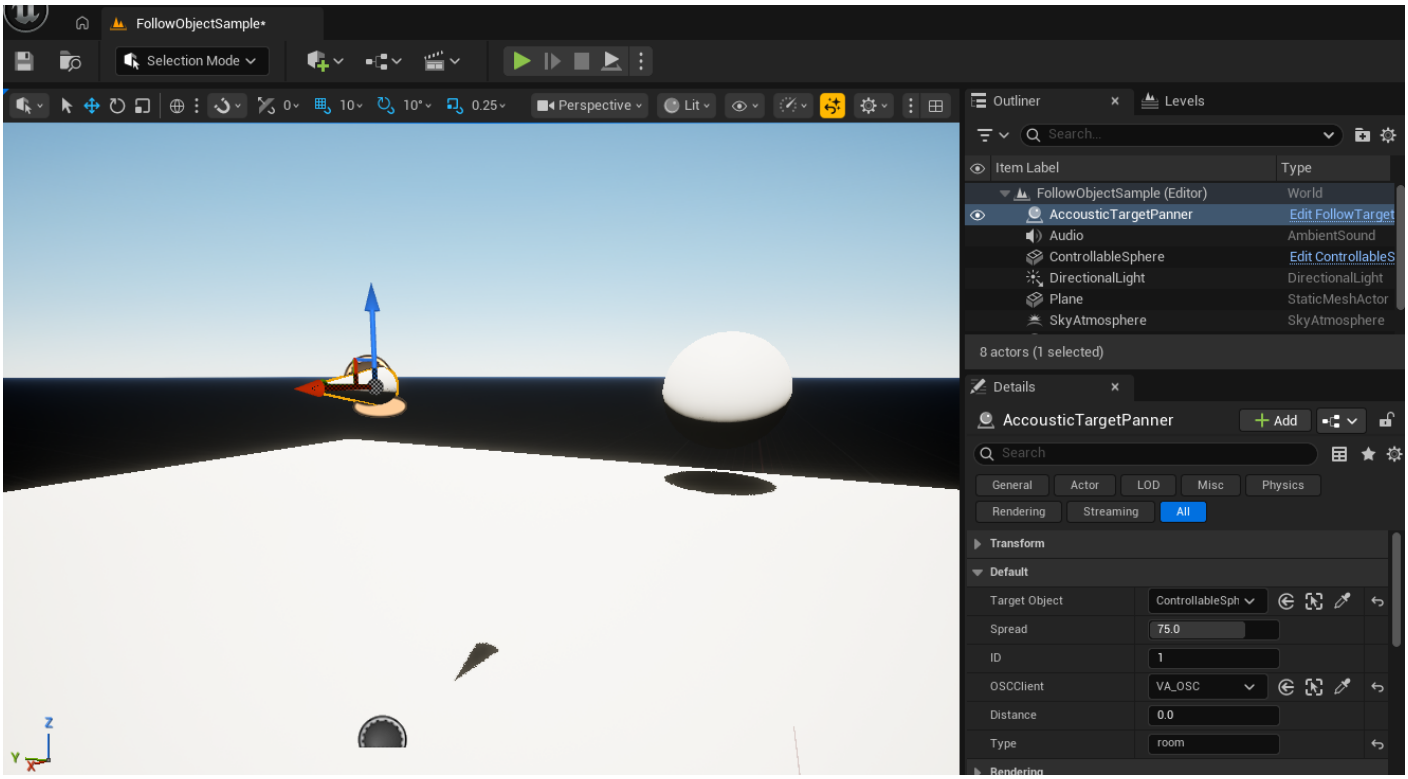
De *Helpers* worden de andere categorieën gebruikt. Als je precies wil weten wat er gebeurd kan je elke functie openen en kijken hoe deze is uitgewerkt.



Voorbeeld Maps

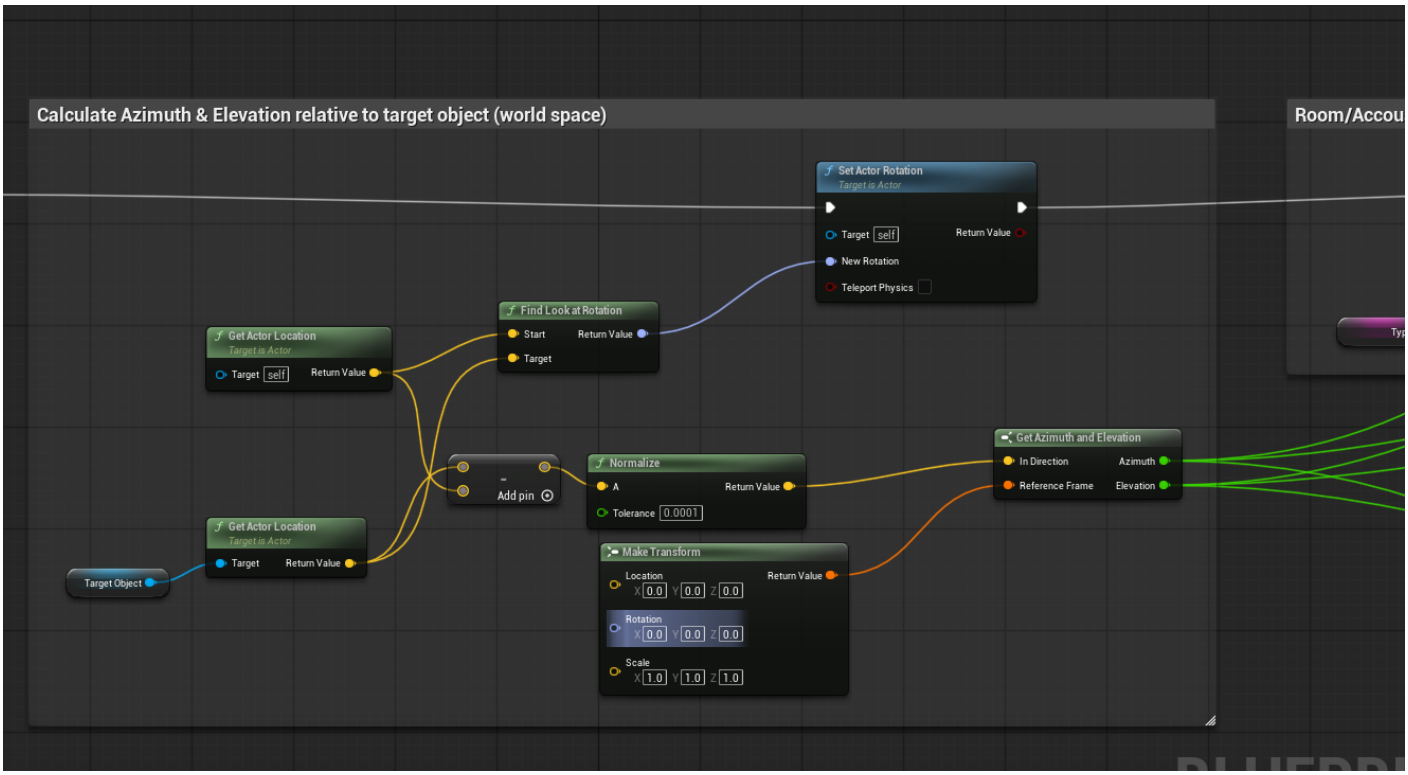
FollowObjectSample

Deze Map is een voorbeeld van hoe je een object de richting waar een Source / Room / Accoustic vandaan komt kan bepalen. Dit bestaat dus uit een **AccousticTargetPanner**, die in het midden van de Map staat, en een **ControllableSphere** die je met instelbare toetsen kunt besturen om het te testen. Dit kan natuurlijk ook een object zijn dat op basis van andere informatie beweegt (zoals gameplay!)

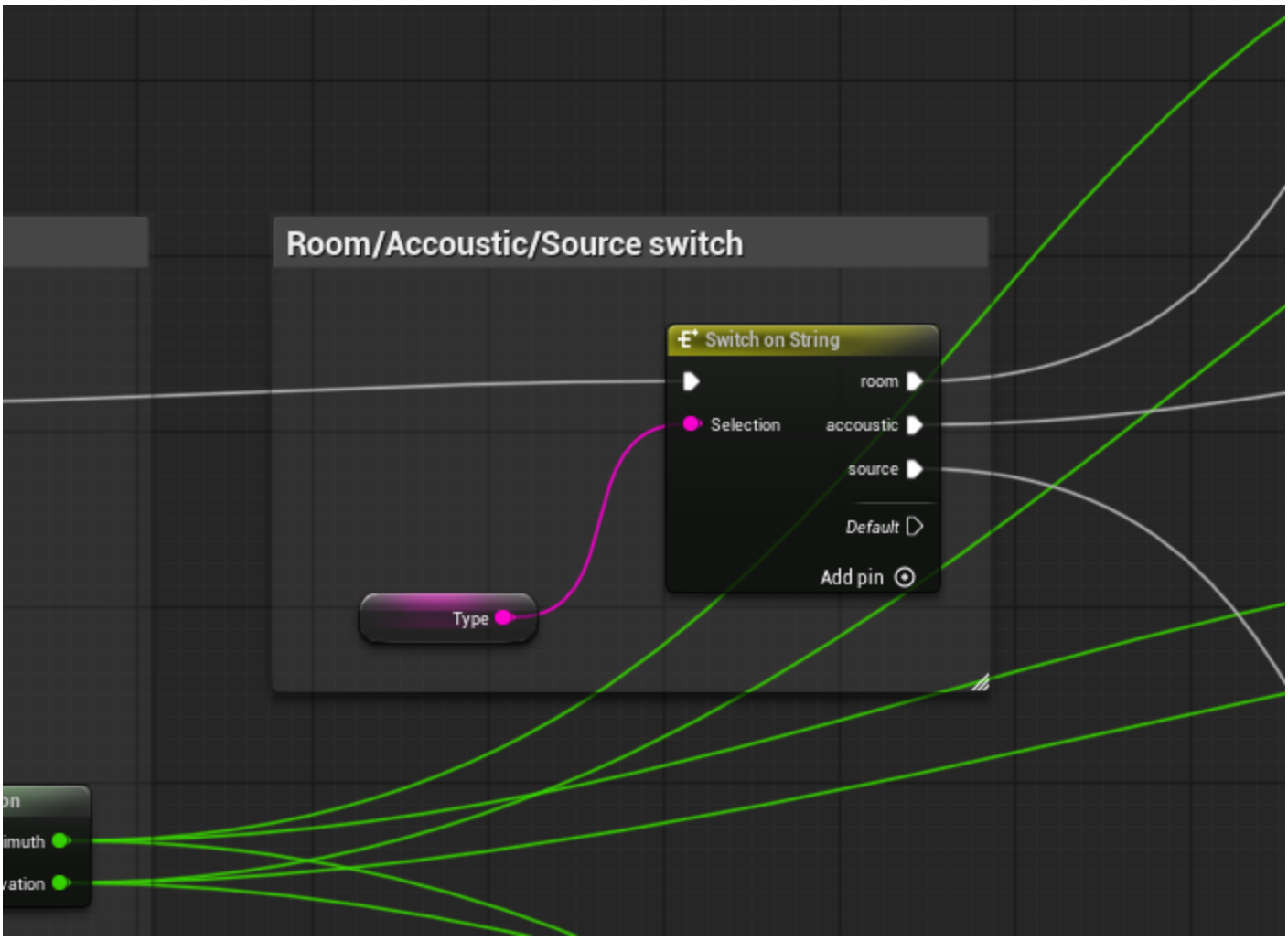


Het *TargetObject* is waar naar gekeken wordt. De *Spread* waarde bepaald hoe breed het geluid moet zijn (dit is een specifieke echotope parameter), de *ID* is de identifier van de source / room / accoustic. De *OSCClient* is de *VA_OSC* blueprint in de scene (zodat de communicatie kan plaatsvinden). *Distance* bepaald hoe ver weg het geluid moet klinken (dat is momenteel in echotope 1.6 nog niet actief, en staat dus nog op 0). En de *Type* kan "room", "accoustic" of "source" zijn zodat het Blueprint Script (*FollowTargetPanner*) weet wat voor soort bericht het moet sturen.

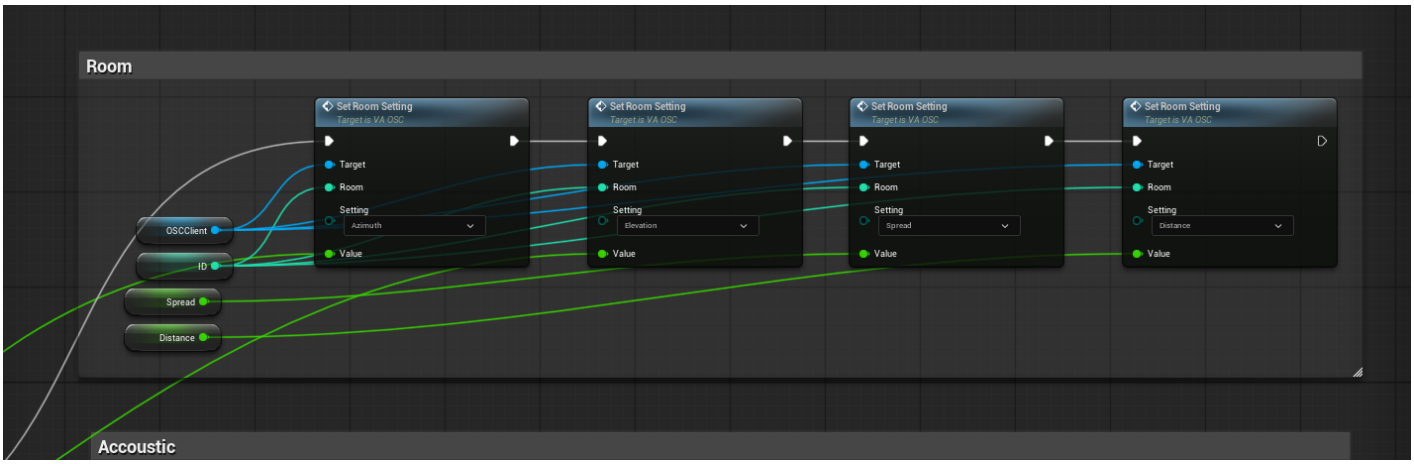
Je kan de *FollowTargetPanner* blueprint openen, en van links naar rechts de *Event Tick* volgen om dit te bevestigen.



*Rekent de richting uit waar het TargetObject is in **polar coordinates** waar echotope mee werkt.*



Voert een ander pad uit op basis van het geluidstype dat we aansturen (source / accoustic / room).

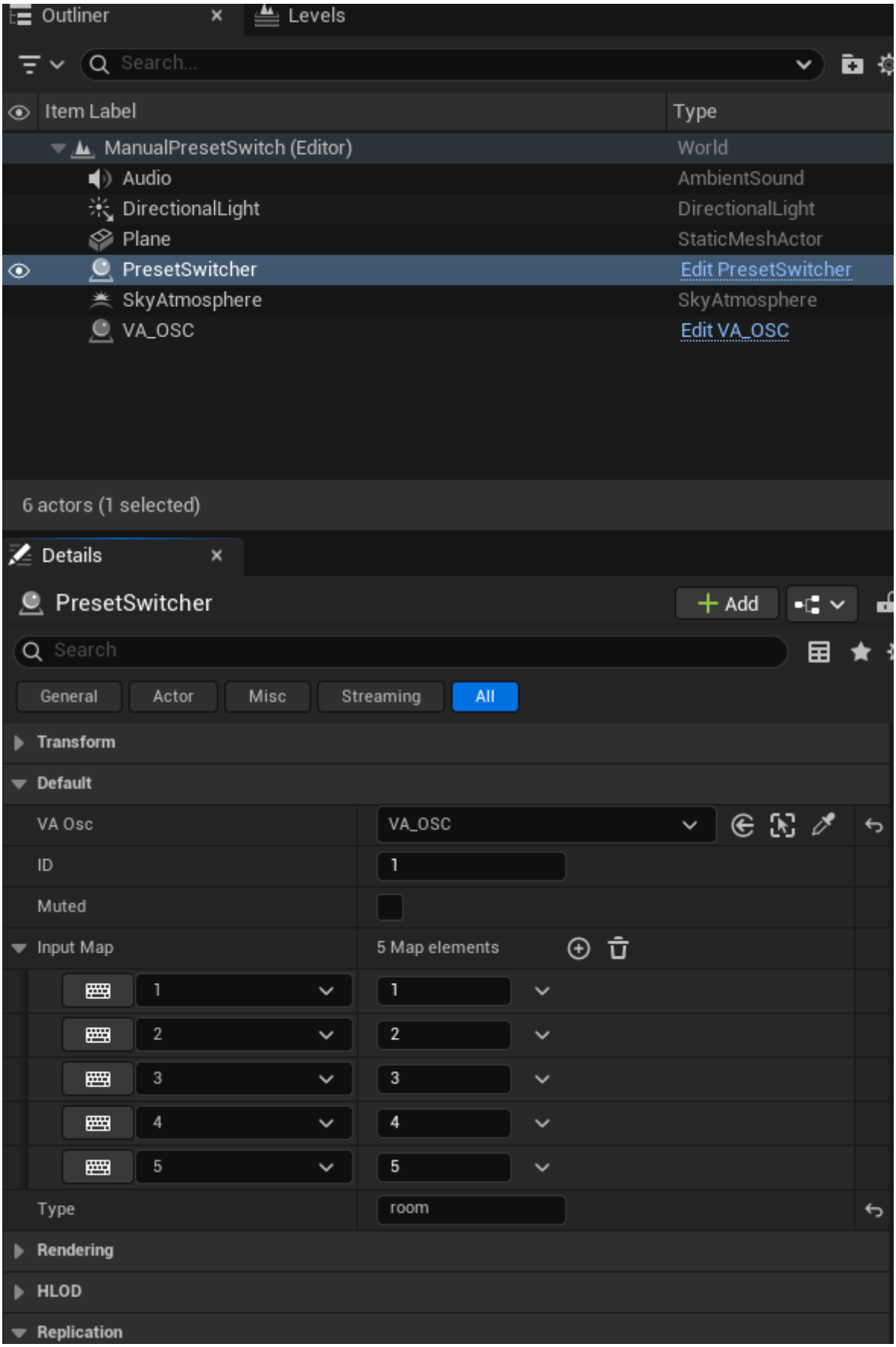


Stuurt de relevante waardes voor het Room-pad door, door de functies van de VA_OSC Blueprint aan te roepen. De overige paden zijn zeer vergelijkbaar.

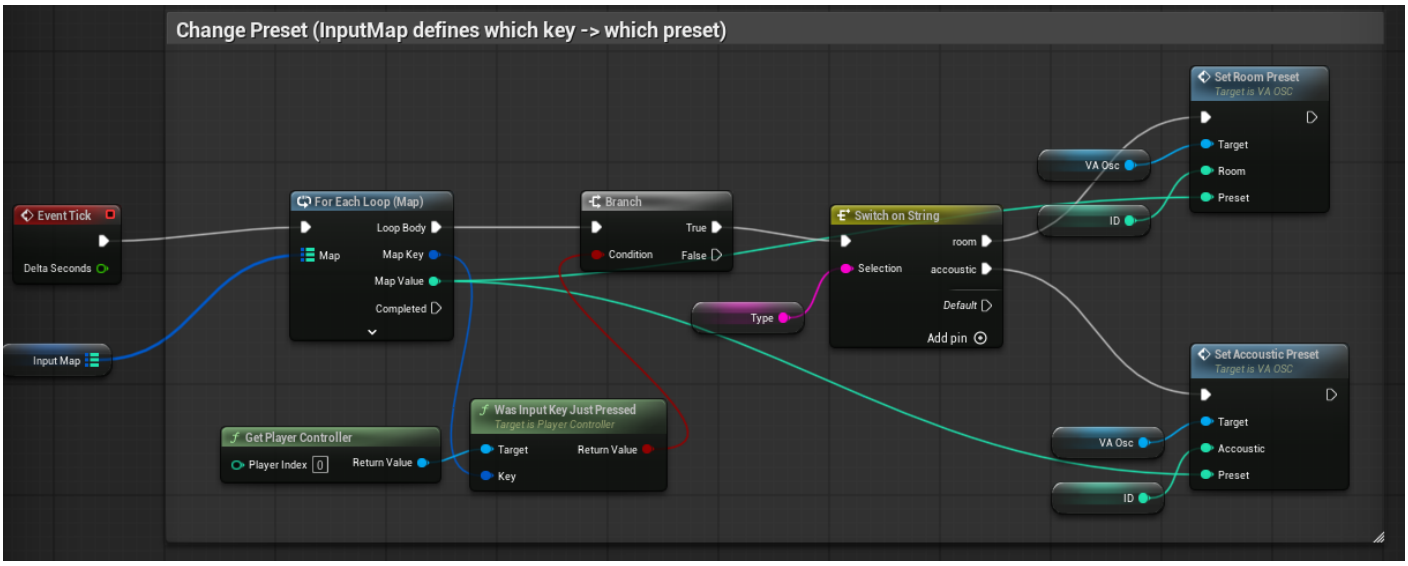
ManualPresetSwitch

Deze Map is een heel simpele opzet, waarmee via de *PresetSwitcher* Blueprint specifieke knoppen kunnen worden gekoppeld aan een preset waarde.

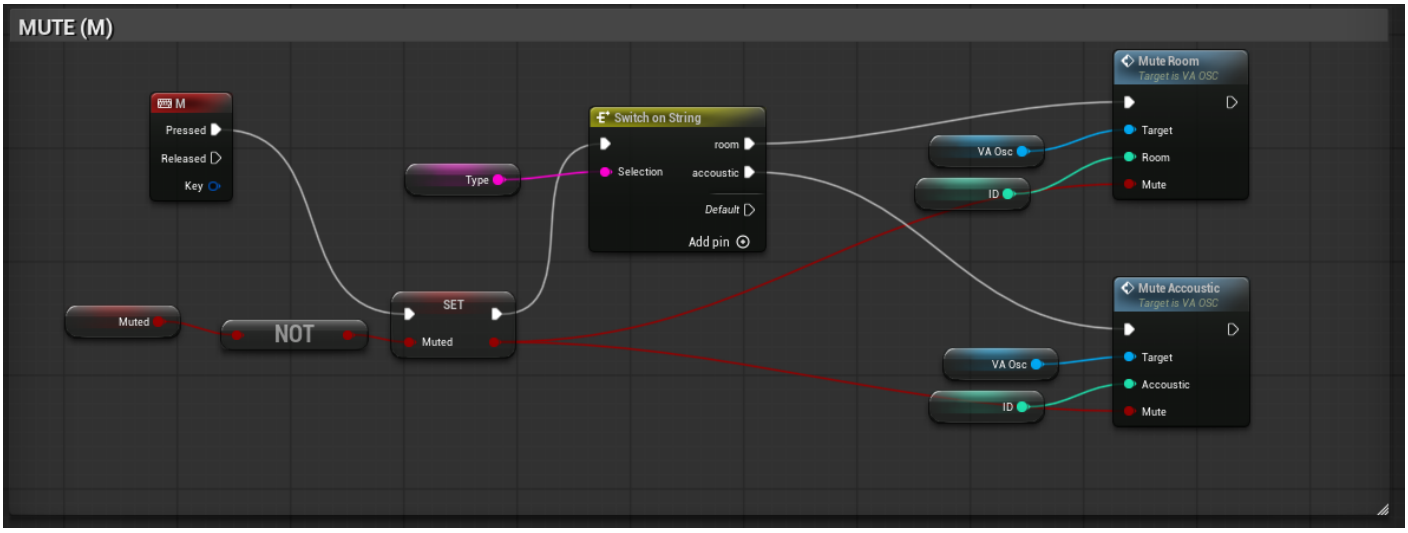
Hieronder zie je de PresetSwitcher in de Outliner geselecteerd, en daaronder zichtbaar de *VA_OSC* instance die is gelinked. Overige informatie is de *ID* (welke source / accoustic / room spreken we aan?), *Muted* (moet je het horen ja/nee?), en de *InputMap* die een knop koppeld aan een preset nummer (1-15). Dit is nu een directe mapping van de knoppen 1-5, naar de presets 1-5. Als laatste is er nog de *Type* waarde die bepaald wat er wordt aangesproken (source / accoustic / room).



Als je de *PresetSwitcher* Blueprint opent kan je dit ook bevestigen:



Gaat door elke knop/preset koppeling in de InputMap heen, en kijkt of deze knoppen nu net zijn ingedrukt door de speler (index: 0). Zo ja: stuurt dit via VA_OSC functies door naar echotope voor de relevante types (room / accoustic)



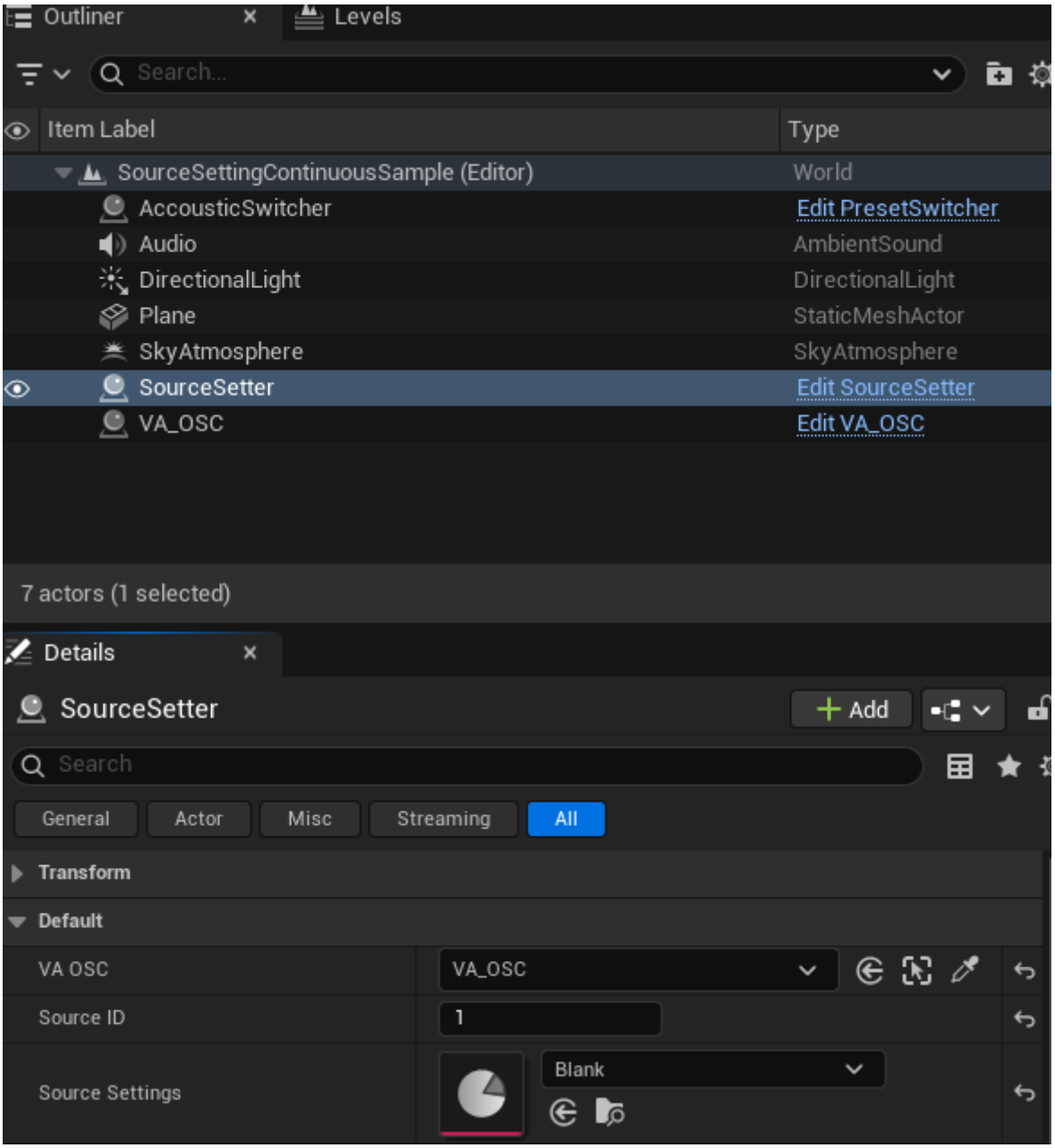
Kijkt of de M toets is ingedrukt, en zo ja zet het de Mute functie aan/uit voor de relevante Type (room/accoustic). Hiervoor gebruikt het wederom VA_OSC functies.

SourceSettingContinuesSample

Deze Map doet eigenlijk wat er in de ietwat lange naam al wordt geïnsinueerd: *stuurt continu de settings van een source aan.*

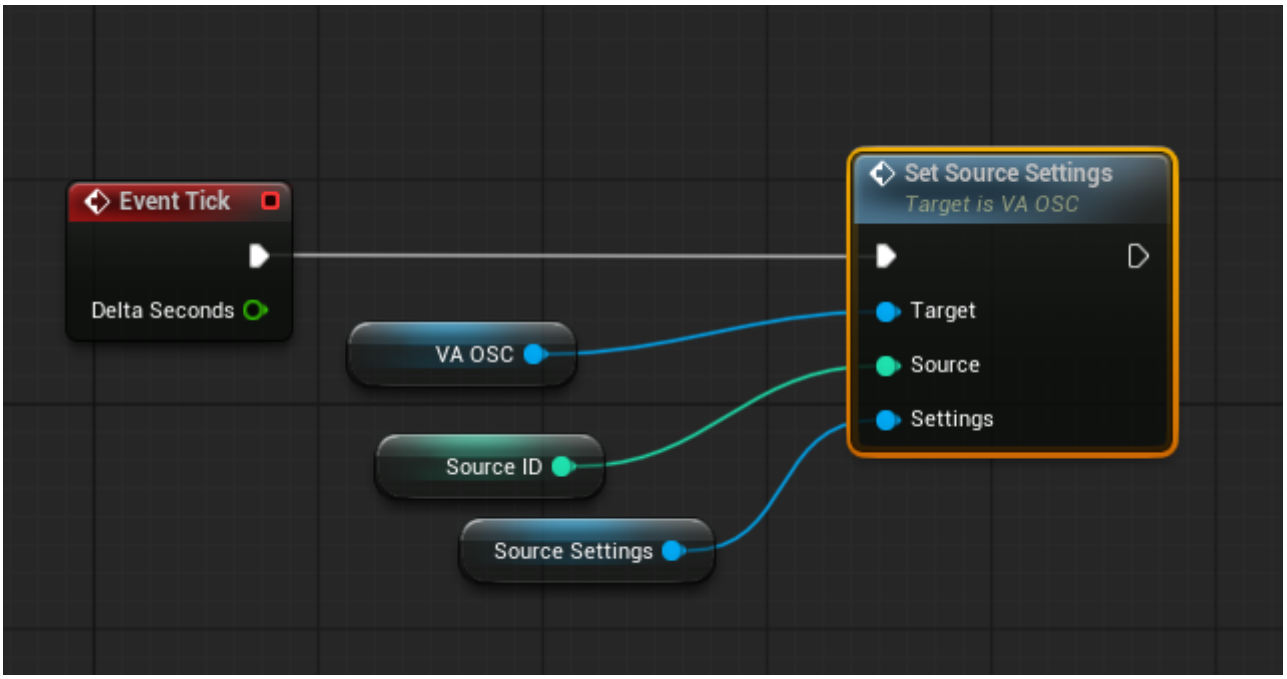
Dit gebeurd via de *SourceSetter* blueprint, die maar 3 variabelen bevat:

1. VA OSC: de VA_OSC instance om de informatie te kunnen sturen
2. Source ID: de ID van de source die we aanpassen in het echotype systeem
3. Source Settings: een verwijzing naar een *Data Asset* waar de op te sturen settings in staan.



> De situatie waar dit voor bedacht is, is als je vanuit de *Data Asset* live wilt kunnen editen, en deze waardes wil opslaan tussen speelsessies. Op die manier kan je bijv. makkelijk eigen presets maken, door die Data Asset vervolgens te dupliceren en een herkenbare naam te geven.

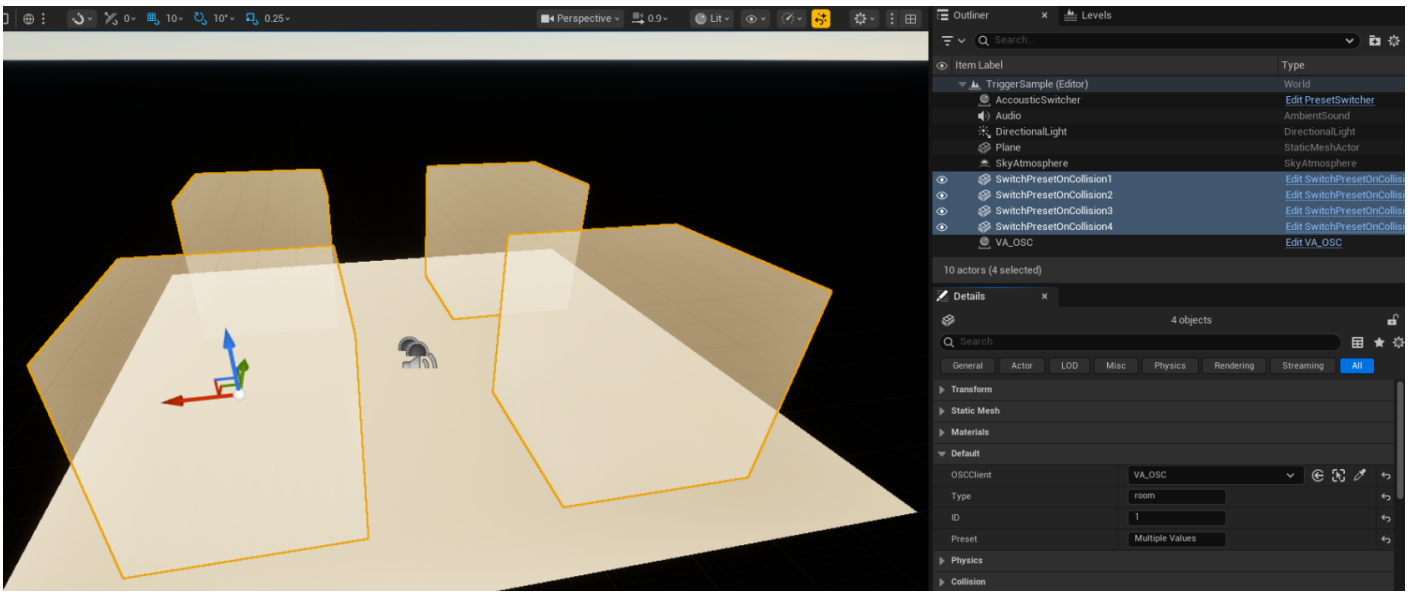
Als je de Blueprint van de *SourceSetter* opent kan je dit gedrag bevestigen:



Stelt elke Tick de Source Settings in op de Source met ID "Source ID", via een functie van VA_OSC

TriggerSample

Deze Map bevat een vier-tal doorzichte kubussen die een Preset instellen op een room, accoustic of source wanneer je met een object (in dit geval de speler) binnen het volume van de kubus terecht komt. Dit kan je bevestigen door er in Play-mode doorheen te vliegen. De kubussen printen dan "BUMP" op het scherm.

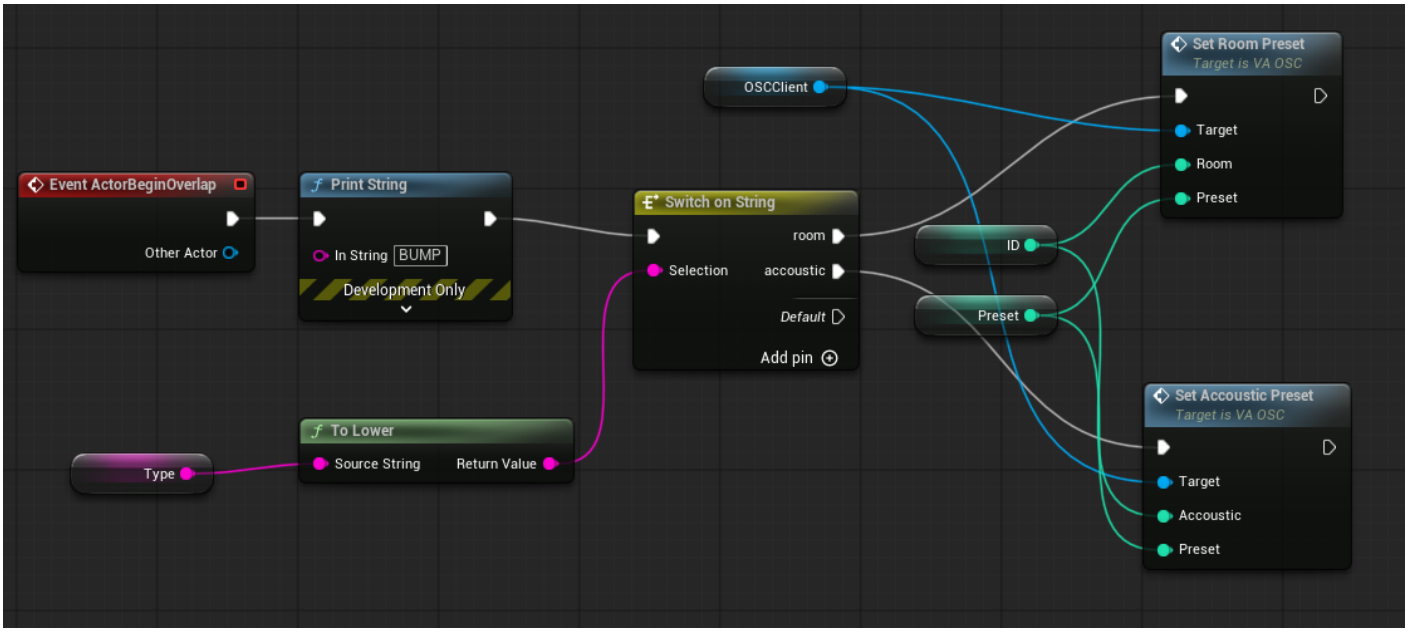


> Dit voorbeeld is gericht op een veel voorkomende situatie in games: dat je op het moment van verplaatsing naar een specifiek gebied iets wilt laten gebeuren. In het geval van dit systeem: de akoestiek veranderen (bijvoorbeeld

> binnen/buiten, verschillende soorten omgevingen, etc.)

De Blueprint bevat variabelen voor de VA_OSC instance, het Type (room, accoustic, source), de ID van de room/accoustic/source, en naar welke Preset (1-15) er moet worden gewisseld (zie het echotope protocol voor meer informatie over wat elke preset voorstelt).

Als je de Blueprint opent kan je dit gedrag ook bevestigen:



Als er een andere Actor overlapt met dit object: voor het relevante type stelt het de ingestelde Preset in via een functie van VA_OSC.

Om te zorgen dat Actors elkaar kunnen overlappen moeten er wel twee zaken samenkomen:

1. Het object moet een component hebben dat collision toestaat (bijv. de StaticMeshComponent van de kubussen)
2. De "Collision Preset" instellingen moeten zo ingesteld staan dat er een OverlapEvent wordt uitgevoerd. In dit geval:
   1. Generate Overlap Events **AAN**
   2. Collision Presets: **OverlapAll**

> Het object dat naar binnen vliegt moet mogelijk ook Collision hebben. Dat is in dit geval de "SimplePawn" Blueprint in de VA_Sample/Blueprints map, die via de "MyGameMode" wordt ingesteld als "Default Pawn Class" (waar je mee begint te spelen als je op Play drukt)

Revision #20
Created 2026-07-10 10:22:16 UTC by Aaron
Updated 2026-07-10 12:56:58 UTC by Aaron