# Organizzazione della cartella Content

Tutti gli asset creati da me si trovano nella cartella **Content/MyContent**, organizzati come se fosse una cartella radice. Eventuali modifiche agli asset già presenti verranno indicate.

# Sfera Interattiva

Implementazione in **BP\_InteractableSphere**.

## Galleggiamento

Mentre la sfera si trova sopra un buco, fluttua su e giù con movimento sinusoidale che segue la formula

I valori sono impostati in modo da ottenere un effetto di galleggiamento piacevole.

## Ombra

Il materiale parent del terreno era già programmato in modo da mostrare un’ombra finta in corrispondenza della posizione della sfera interattiva sul piano XY. Ho aggiornato quel materiale in modo che il raggio dell’ombra si aggiusti (entro un range coerente) in base all’altitudine della sfera, per aumentare la sensazione di immersività quando la sfera fluttua o quando il giocatore la sposta in alto o in basso.

Siccome molti altri materiali condividono lo stesso codice, dunque anche la visualizzazione dell’ombra, ho impostato nel parent uno switch parameter che disattiva questo comportamento in tutte le istanze diverse da quella usata dal terreno.

M\_PBR



## Tocco

La sfera si ferma se il giocatore la tocca, cioè se vi sovrappone la mano. Perché fosse possibile, è stato necessario creare dei Physic Assets da associare alle skeletal mesh delle mani.

## Grab Component

Il codice del Grab Component già presente nel VR Template è stato esteso per aggiungere la presa a due mani.

## Risposta aptica della presa a due mani

Afferrare la sfera a due mani deve causare una risposta aptica continua su entrambi i controller. Tuttavia, Unreal Engine è in grado di inviare solo una risposta aptica a ciascun controller per frame. Questo va in contrasto con la risposta aptica inviata dall’implementazione default del VR Template, sovrascrivendo il segnale e facendo sì che solo il controller che stava già afferrando la sfera inizi a vibrare continuamente dopo la presa a due mani.

Per risolvere il problema, ho introdotto un nodo “Delay Until Next Tick” prima della riproduzione della risposta aptica.

Tuttavia, nell’implementare l’incremento d’intensità con lo spostamento della sfera, ho potuto rimuovere il codice precedente, non più necessario.

# Buco

Implementazione in **BP\_Hole**. I buchi da 2, 3 e 4 sono implementati come classi child. Una volta iniziata un’interazione, il giocatore può sempre afferrare nuovamente la sfera e spostarla in un altro buco.

## Buco 1

Dove si trova la sfera all’avvio. È un’istanza della classe parent, dunque non implementa nessuna logica specifica oltre al trattenere la sfera se vi si trova sopra e attivare il Niagara Effect.

## Buco 2

Riproduce il video nel portale.

## Buco 3

Fa apparire l’erba.

## Buco 4

Riproduce la musica a partire dal punto designato, con attenuazione spaziale logaritmica proporzionale alla distanza. L’asset audio è stato impostato in modo da non smettere di riprodurre se il volume si azzera, cioè se il giocatore si trova fuori dall’area di falloff.

# Game Mode

Implementazione in **BP\_UltraGameMode** come child di VRGameMode. Tiene traccia delle interazioni svolte dal giocatore: quando il giocatore ha attivato tutte e tre le interazioni possibili, fa sì che la sfera si distrugga in pezzi.

# Cursore del Teletrasporto

Sono stati modificati gli asset già presenti. La curva non modifica più il colore secondo un gradiente, rimanendo bianca. Il cursore è stato sostituito con un cerchio bianco.

# Ulteriori Sviluppi

Meta fornisce una distribuzione di Unreal Engine personalizzata e ottimizzata per lo sviluppo su Quest. Nella mia precedente esperienza, l’uso di questa versione ha migliorato notevolmente le prestazioni delle app che realizzavo rispetto alla versione ufficiale di Epic, consentendomi di spingere un po’ di più sulla resa grafica e sulle interazioni tra oggetti. Tuttavia, la sua installazione e configurazione richiede del tempo in più.