ELABORATO D'ESAME

GAME ENGINES AND INTERACTIVE EXPERIENCE

ROBERTO BASILE GIANNINI

N97000340

Introduzione

Con il presente elaborato si è voluto realizzare un'esperienza interattiva che prenda spunto da diversi videogiochi annoverabili nella categoria "Horror psicologico", come *Observer*, *Amensia* e molti altri. Per la sua realizzazione, oltre al game engine Unreal Engine 4, ho usato software esterni quali *Blender* per la modellazione di mesh, *Autodesk Maya* per la realizzazione di animazioni, *Audacity* per l'editing audio. Inoltre, buona parte dei modelli, superfici e decals utilizzati sono stati presi dal marketplace di Unreal o da specifici siti web, come *CGTrader*, *Turbosquid* e *Free3D*, mentre per le tracce audio i portali interessati sono stati *Freesound* e *YouTube*.

L’esperienza obiettivo

L’esperienza di gioco obiettivo consiste nel ricreare uno scenario in grado di richiamare il momento storico che attualmente stiamo vivendo (quale la pandemia di COVID-19), accompagnato da sfumature horror e con l’intenzione di trattare il fenomeno dell’antivaccinismo e delle rispettive conseguenze, sociali e non. In particolare, si vuole fornire una visione a metà tra le due fazioni che inevitabilmente si sono andate a creare: i pro e gli anti vax. Le meccaniche di gioco si limitano ad un approccio esplorativo del mondo di gioco, con la possibilità di interagire con l’ambiente circostante, raccogliere oggetti per la risoluzione di enigmi/meccanismi e soprattutto leggere documenti sparsi nei due livelli realizzati.

Paura

In generale, le emozioni possono emergere confrontando le osservazioni contestuali con quella che è la conoscenza a priori, in particolare se queste sono in grado di richiamare stati emotivi passati. Essendo l’obiettivo quello di innescare *paura*, è bene osservare come questa sia definibile come un’emozione anticipatoria dell’emozione anticipata relativa al dolore. Dolore che, come ovvio che sia, un’esperienza digitale non può scaturire: si tratta pur sempre di finzione, di un contesto palesemente inventato e che in nessun modo può configurare un reale pericolo. Questo aspetto evidenzia la difficoltà che si ha nel realizzare un videogioco che sia in grado di generare una tensione continua, tale da accompagnare il giocatore per tutto il gioco (o quasi) e senza far uso di jumpscare continui (e a volte incoerenti) che possano condizionare negativamente l’engagement. Per questa ragione, un obiettivo primario è quello di realizzare un contesto estremamente credibile, con una buona dose di realismo e che possa essere facilmente mappato con le esperienze pregresse del giocatore (ed in questo caso, purtroppo, la pandemia ha riguardato tutti). Il sottosistema dei material e quello delle animazioni giocano un ruolo centrale del punto di vista della credibilità e del realismo, ed è il motivo per il quale sono stati i sottosistemi principalmente trattati. Va comunque precisato che anche altri sistemi sono stati coinvolti, come quello dell’audio (strettamente legato a quello delle animazioni), ed anche il Particle System e la simulazione fisica (entrambi in forma molto marginale).

L’ambiente di gioco

L’ambiente di gioco è costituito sia da spazi aperti che da spazi chiusi, volutamente non dispersivi per evitare di allontanare il giocatore dai “luoghi chiave” nei quali trovare gli interagibili necessari per progredire col gioco. La planimetria (e in particolare quella degli spazi chiusi) è stata pensata con l’obiettivo di mantenere viva la tensione nel giocatore, e lo stesso vale per quanto riguarda la scelta delle soundtrack e dell’illuminazione degli ambienti.

Nell'ambiente sono inoltre presenti un buon numero di trigger, aventi lo scopo di innescare eventi al verificarsi di precise condizioni: per esempio, durante il primo livello, quando il giocatore raccoglie il cacciavite e si dirige verso la cabina elettrica numero 3, grazie ad un trigger (e ad una variabile booleana settata al momento della raccolta dell'oggetto) la Level Blueprint reagisce iniziando la riproduzione di diversi suoni e di una soundtrack. Ulteriori trigger sono stati pensati per evitare la sovrapposizione di soundtrack, aventi una funzione di "Fade-Out" della soundtrack correntemente in esecuzione, e per dare inizio a cutscene.

Leggibili e graffiti: creare contesto

In tutto il gioco sono presenti leggibili (diari, note) e graffiti che contribuiscono alla creazione del contesto narrativo. In particolare, come da introduzione, si è voluto dare voce ad entrambe le fazioni (PRO-VAX e NO-VAX) al fine di mostrare il quadro "umano" di entrambe le realtà. Lo scopo è spingere il giocatore alla conclusione che, per quanto determinate posizioni dogmatiche e prive di evidenze scientifiche siano da condannare, è altrettanto vero che queste vengano assunte in mancanza di determinati strumenti intellettuali e di una certa sensibilità verso specifiche tematiche: anziché dare origine ad una vera e propria guerriglia, ciò dovrebbe avvicinare la fazione opposta nel tentativo di sradicare determinate convinzioni antiscientifiche e deleterie, per il bene del singolo e della comunità. I graffiti (realizzati tramite decalcomanie, come verrà spiegato nel capitolo relativo al sottosistema dei material) contengono messaggi rappresentativi delle due fazioni, evidenziando l'astio reciproco e la tensione sociale che si respira. Per guidare il giocare in una più chiara identificazione dell'autore del messaggio (cioè la "fazione che scrive"), si è scelto di usare font e colori diversi, nonostante il contenuto del messaggio permetta, comunque, di fare questa distinzione. Grafie diverse, inoltre, permettono di identificare sì "persone diverse", ma in particolare "persone". I leggibili disseminati nei due livelli hanno un duplice obiettivo:

- Dare un'immagine del contesto circostante, riducendo l'information gap che di per sé non è molto alto, dal momento che, nella base, il mondo di gioco non è troppo distante dal nostro attuale (temporalmente, il gioco è collocato nel 2024, come verrà indicato in un diario);
- Guidare il giocatore alla risoluzione di enigmi e meccanismi, non indicando con esattezza cosa fare (per evitare che le soluzioni siano troppo banali), ma lasciando indizi chiari e immediati.

Con un information gap riducibile "al costo" di un leggibile ed un contesto di base non molto lontano dai giorni nostri, si vuole motivare il giocatore a progredire nel gioco con lo scopo di vederci chiaro in merito a:

- La natura della creatura presentata nel primo leggibile;
- Come si è evoluta questa faida tra pro e anti vax;
- Come si è evoluto il mondo entro il 2024.

Come verrà chiarito nel capitolo relativo alle blueprint, i leggibili altro non sono che oggetti con i quali è possibile interagire e costituiti da un widget contenente il messaggio.

Il protagonista

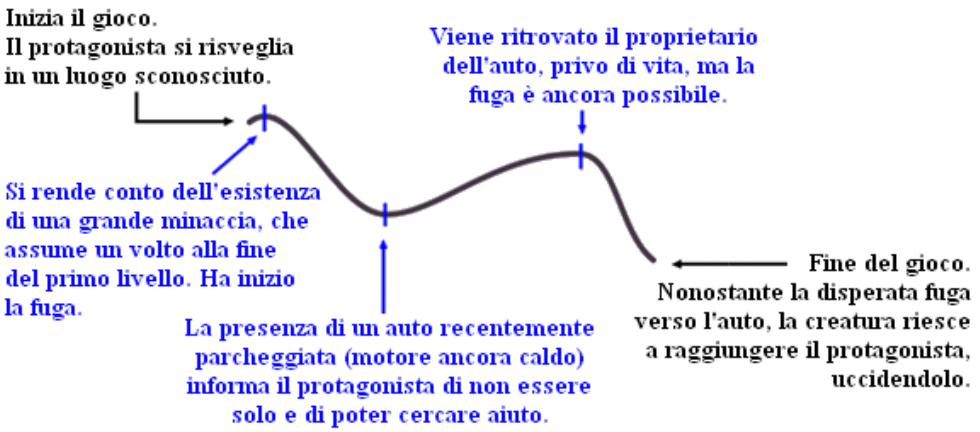
Il protagonista non viene mai presentato, né visto e né sentito, e di esso non si conosce nemmeno la sua (eventuale) collocazione tra le due fazioni nel gioco. L'obiettivo è quello di garantire la massima immedesimazione del giocatore nel protagonista e per questo motivo si è scelta come visuale di gioco quella in prima persona. L'assenza di un volto, di una voce e di una presa di posizione (che rappresenterebbe una mentalità caratterizzante il personaggio) impediscono al giocatore di fornirgli una personalità, e questo può essere un assist al giocatore stesso affinché questi ne vesta i panni durante il gioco. Anche in questo caso si evidenzia l'obiettivo di "far entrare" il più possibile il giocatore all'interno del mondo di gioco, così da semplificare la generazione delle emozioni (in questo caso la paura, come anticipato).

La creatura

La scelta del modello della creatura presente nel gioco, la quale rappresenta il mostro che semina terrore e vittime, è stata condizionata dalla volontà di richiamare dei tratti umani. L'idea non è solo quella di associare una figura mostruosa alla pandemia, ma di rappresentare un mostro generato dalla fusione del pericolo dovuto alla pandemia stessa con gli atteggiamenti umani assunti in una situazione del genere. Non precisare mai all'interno del gioco se questa creatura possa essere legata solo ad infetti no-vax o ad infetti pro-vax vuole comunicare che, a modo proprio, entrambe le fazioni possono contribuire negativamente alla lotta contro l'unico reale nemico, quale il virus.

Arco narrativo

In termini di arco narrativo, si è optato per un trend negativo, relativo dunque alla tragedia. Questo in quanto si vuole comunicare, coerentemente alla non identificazione via gioco (ma magari via giocatore) della posizione assunta dal protagonista nel contesto in cui si trova, che uno scenario del genere sarebbe drammatico per l'intera comunità, e non solo per l'una o l'altra fazione. La curva emotiva adottata è la Oedipus, di seguito riportata:



La fuga riuscita alla fine del primo livello, gli sforzi fatti per trovare (con successo) il mezzo per fuggire dall'imminente pericolo, le speranze prima infrante (a causa del ritrovamento senza

vita del proprietario dell'auto) e poi parzialmente rinnovate (grazia al fatto che questi portasse con sé le chiavi dell'auto), sono elementi che hanno l'obiettivo di creare un'aspettativa positiva nel giocatore, avendo l'obiettivo di fuggire e ritenendo probabilmente l'effettiva fuga del protagonista. Protagonista che, come noto, verrà fatalmente raggiunto dalla creatura, con lo scopo di provocare una reazione emotiva nel giocatore, sorprendendolo.

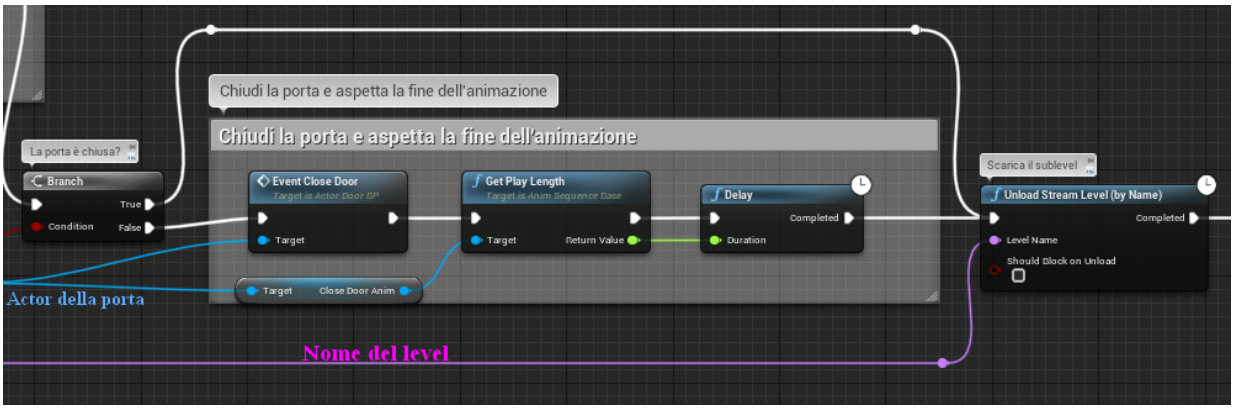
Sub-levels

La composizione del secondo livello e la logica con la quale è possibile accedere a determinate aree, permettono una strutturazione dello stesso in sottolivelli. Nella fattispecie, sono tre i sottolivelli previsti:

- *HUB*, relativamente all'hub vaccinale presente nel gioco;
- *Office*, relativamente all'ufficio presente nel gioco.
- *StorageArea*, relativamente all'area di fine gioco, ovvero un deposito.

Partiamo dall'osservazione che l'hub e l'ufficio sono entrambe aree accessibili da subito, a differenza del deposito che è accessibile solo al verificarsi di un determinato evento (l'inserimento di un codice). Dal punto di vista narrativo, non avrebbe alcun senso prevedere il caricamento del sottolivello StorageArea prima del verificarsi di questo evento ed è per questa ragione che lo streaming del sottolivello avviene solo dopo questo evento. In particolare, per il caricamento della StorageArea è stato utilizzato il nodo *Load Stream Level*, operando tramite Level Blueprint: il nodo viene eseguito subito dopo l'apertura della serranda del deposito, evento che si verifica ad una distanza sufficiente dal player, tale da nascondergli il caricamento degli asset. Per quanto riguarda i sottolivelli HUB e Office, anche in questo caso è stato utilizzato Load Stream Level accompagnato dalla controparte Unload Stream Level e dalla presenza di un trigger.

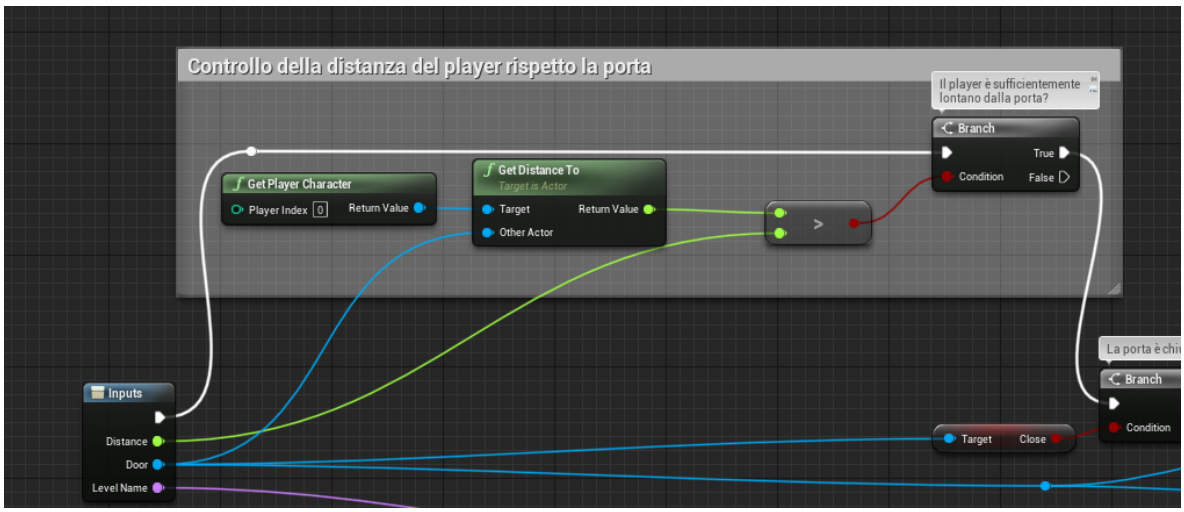
Si è scelto di operare in tal senso e di non utilizzare dei Level Streaming Volume al fine di poter personalizzare la logica di "unload" dei sottolivelli, come mostrato dalla macro *Try to Unload SubLevel,* eseguita solo quando il giocatore esce dal trigger di quella determinata area:



Prima di parlare della condizione che precede l'effettivo unload del sottolivello (non presente nell'immagine), analizziamo i due possibili scenari relativi allo scaricamento, partendo dallo scopo di nascondere al giocatore lo streaming:

- Il giocatore esce dall'area interessata e chiude la porta. Una volta uscito dal trigger, viene eseguito il nodo Unload Stream Level (ramo True) ed il sottolivello viene scaricato;
- Il giocatore esce dall'area interessata, ma non chiude la porta. Uscito dal trigger, viene prima chiusa la porta e, terminata l'animazione, viene eseguito il nodo Unload Stream Level (ramo False) ed il sottolivello viene scaricato.

Con questo approccio, viene assicurato lo scaricamento del livello in maniera invisibile all'utente, sia quando effettivamente il giocatore entra ed esce dall'area interessata, sia qualora dovesse solo attraversare il trigger. Per quanto riguarda la precondizione:

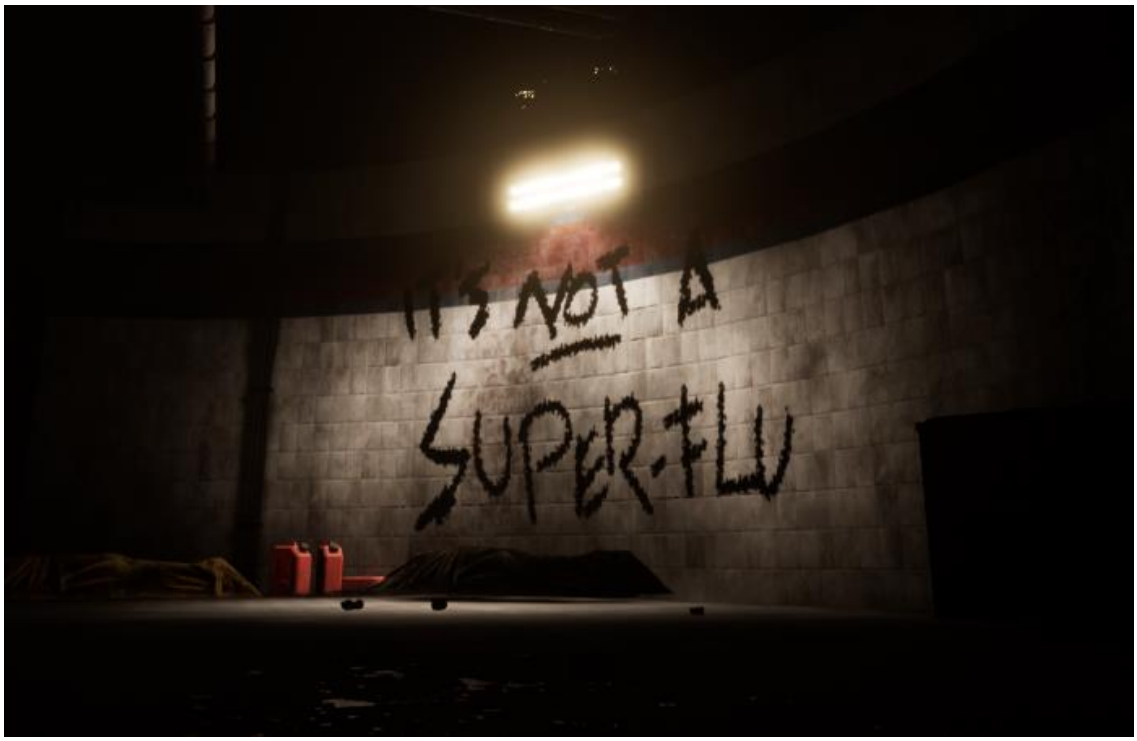


Infatti, l'uscita dal trigger si verifica anche quando il giocatore entra effettivamente nell'area caricata: in assenza di questo controllo, l'area verrebbe immediata scaricata. Sostanzialmente, si vuole distinguere il caso in cui il giocatore sta uscendo dal trigger perché si sta allontanando dall'area, dal caso in cui il giocatore sta uscendo dal trigger perché sta entrando nell'area.

Cutscene

Sono presenti tre cutscene all'interno del gioco, precisamente all'inizio e alla fine del primo livello, e alla fine del secondo. Come anticipato, la loro riproduzione comincia quando il giocatore si ritrova in una determinata condizione ed entra in un trigger. Dal punto di vista del camera movement e dello shot size, si è scelto di mantenere la visuale in prima persona riproducendo i movimenti esatti del giocatore, con lo scopo di rendere continua l'esperienza di gioco anche durante la transizione tra livelli. Le due cutscene finali hanno lo scopo di comunicare delle informazioni ben precise, in particolare:

- La prima cutscene mostra, per la prima volta, le fattezze della creatura e la sua intenzione di catturare il protagonista. A questo punto, il giocatore sa di essere inseguito da qualcosa e questa informazione lo accompagnerà per tutto il secondo livello;
- La seconda cutscene mostra il tentativo di fuga fallito da parte del protagonista, durante la seconda ed ultima apparizione della creature. L'inquadratura finale vuole comunicare *chi* avrebbe dovuto essere il solo ed unico nemico per entrambe le fazioni, evitando la sua esplosione tramite determinati comportamenti umani e di renderlo ancor più letale.



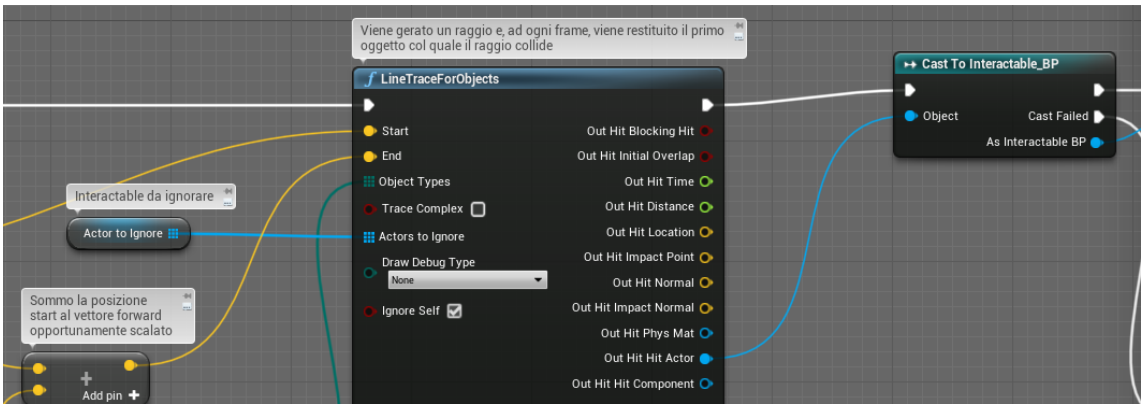
All'interno del gioco sono stare realizzare un buon numero di blueprint, coerentemente alle esigenza manifestate. In particolare, per realizzare la principale meccanica di gioco, quale l'interazione del giocatore con il mondo circostante, è stata realizzata una vera e propria gerarchia di blueprint, con radice *Interactable_BP*.

Interactable

Un actor con il quale è possibile interagire (indicato come *interactable*) appartiene a questa gerarchia. La blueprint Interactable_BP è una abstract class alla quale è aggiunta l'interfaccia *Interactable Interface* contenente a sua vuole la funzione *interact*. Questa funzione non è direttamente implementata da Interactable_BP, ma sarà compito delle blueprint derivate farlo, così da poter implementare lo specifico comportamento al cospetto dell'interazione del giocatore con quella determinata blueprint della gerarchia. Interactable_BP mette a disposizione degli elementi base che sono in comune con qualsiasi tipo di interactable:

- Un *message interact*, ovvero il widget che viene mostrato quando il mirino del giocatore punta l'interactable ed è sufficientemente vicino (settabile tramite editor);
- Una variabile booleana di inizializzazione, quale *init non interactable*, per dare la possibilità ad un interactable di essere inizializzato come non interactable (settabile tramite editor);
- Eventi e funzioni di servizio.

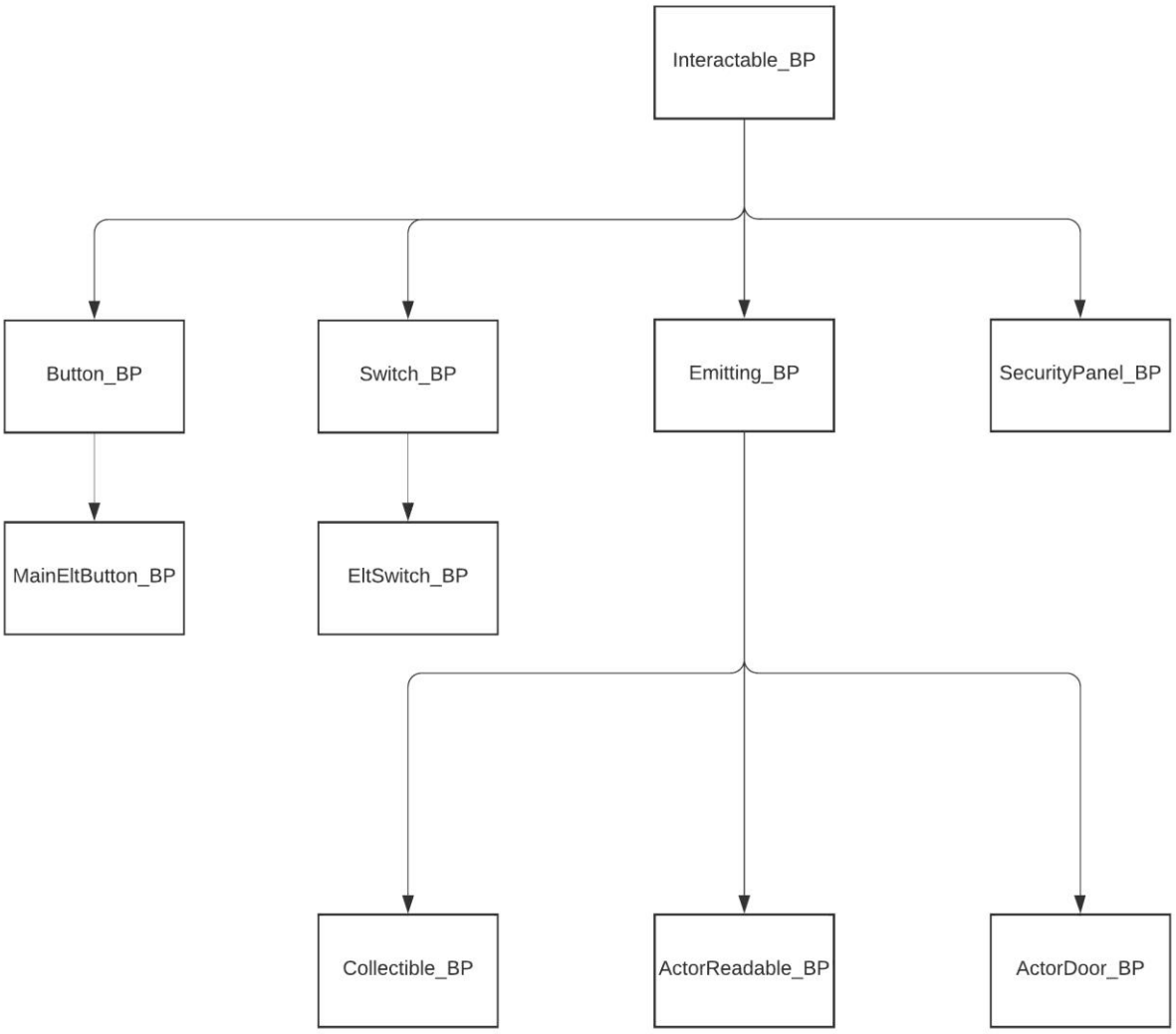
Per capire come è possibile l'interazione del giocatore con gli interactable, è necessario accedere alla blueprint relativa al *FirstPersonCharacter*. In essa, infatti, si ha modo di constatare che per il rilevamento degli oggetti di tipo interactable è stato utilizzato il raycasting, sfruttando il nodo *LineTraceForObjects*: ad ogni frame, viene restituito il primo oggetto col quale il raggio collide e viene tentato un casting a Interactable_BP, che in caso di successo ci permette di concludere che ci si trova al cospetto di un interactable. La dimensione del raggio, chiaramente, è finita ed è settabile tramite editor.



Il risultato visivo di un cast che ha avuto successo è dato dal cambiamento di colore del mirino del giocatore e la comparsa di un widget, così da segnalare al giocatore la possibilità di interagire e qual è l’azione che può essere eseguita.

Albero degli interactable

Di seguito l'albero gerarchico che vede come radice la blueprint Interactable_BP:



Le blueprint Collectible_BP, ActorReadable_BP e ActorDoor_BP sono blueprint derivanti dalla blueprint Emitting_BP, dal momento che tutte e tre le tipologie di actor brillano all'interno del gioco (maggiori dettagli verranno forniti nel capitolo relativo al sottosistema dei material). In particolare:

- Collectible_BP riguarda gli oggetti che possono essere raccolti nel mondo di gioco. L'inventario del player è costituto da una mappa Stringa-Booleano, dove la stringa corrisponde al nome dell'oggetto che può essere raccolto e il booleano allo stato raccolto-non raccolto: è stato adottato questo approccio dal momento che gli unici oggetti che possono essere raccolti sono oggetti chiave. La blueprint implementa la funzione Interact aggiornando l'inventario del giocatore e segnala, tramite dispatcher, l'avvenuta raccolta di quel determinato oggetto (utile alla Level Blueprint per innescare determinati eventi);
- ActorReadable_BP riguarda gli oggetti che possono essere letti ed implementa la funzione Interact gestendo l'apertura e la chiusura del widget contenente il testo;
- ActorDoor_BP riguarda l'interazione con le porte e a tal proposito si rimanda al sottosistema delle animation.

Le blueprint Button_BP e Switch_BP implementano, rispettivamente, il concetto di azionatore monostabile e bistabile. Ne derivano delle blueprint specializzate in azionatori elettrici. In particolare, la blueprint EltSwitch_BP presenta due riferimenti a MainEltButton_BP, dal momento che quest'ultima implementa il concetto di "azionatore monostabile principale", relativo cioè ad una cabina elettrica principale. Infatti, sulla base dello stato della cabina principale (dipendente dagli azionatori principali), un azionatore EltSwitch_BP:

- Può cambiare il suo stato in "acceso" e dunque chiamare il relativo event dispatcher che verrà catturato da tutti quei dispositivi elettrici azionabili a distanza (in questo caso le blueprint WallLamp_BP, WallLantern_BP, WallLed_BP);
- Può mostrare un messaggio di assenza di corrente nel caso in cui si dovesse tentare di cambiare il suo stato in "acceso", ma non c'è corrente.

EltSwitch_BP inoltre fornisce degli eventi relativi ad inizio e fine di un sovraccarico, ed è anche presente un componente di tipo Particle System per simulare la presenza di scintille. Per finire, SecurityPanel_BP gestisce la logica relativa ad un pannello di sicurezza nel quale è possibile inserire un codice. Il pannello è implementato dal widget *SecurityPannel_UI* e consiste in un tastierino numerico: specifici eventi della blueprint verranno chiamati quando il codice inserito risulterà essere corretto o sbagliato, e nel primo caso sarà presente anche un dispatcher che segnali l'accaduto.

Material

Il material stabilisce lo stile visivo di un determinato asset presente nel gioco. In un videogioco del genere horror psicologico l'impatto visivo (così come le animazioni e i suoni) svolge un ruolo centrale ai fini dell'immersività nelle scene di gioco, ma anche per manifestare il suo stato interno. Nel caso specifico, oltre ai consueti Master Material, sono stati utilizzati Masked Material, Material Instance (Constant e Dynamic), e Decals. Sono stati utilizzati (interamente o parzialmente) materials messi a disposizione sul marketplace o dai modelli gratuitamente scaricati dalla rete. Nella cartella "MyMaterials" sono presenti materials da me realizzati da 0 (a partire da texture preesistenti) oppure da me parzialmente modificati, così come in "MyDecals" per le decalcomanie.

Material Instance

A partire da Master Materials definiti in maniera tale da esporre dei parametri, una Material Instance è un'istanza di Master Material che presenta la possibilità di regolare i parametri esposti senza la necessità di ricompilare il material. Bisogna distinguere le due tipologie di Material Instance che possiamo avere:

- *Material Instance Constant (MIC),* una Material Instance che presenta la facoltà di regolare in editor i parametri esposti, osservandone gli effetti (sull'asset a cui è assegnata) in tempo reale senza la necessità di ricompilare il Material. Questo approccio risulta di grande aiuto in fase di sviluppo, ma una volta avviato il gioco i parametri vengono cristallizzati e quindi non è possibile modificarli run-time.
- *Material Instance Dynamic (MID),* una Material Instance calcolata a run-time e che dunque permette di regolare i parametri durante il gameplay (per esempio al verificarsi di determinati eventi).

In questo progetto si è fatto largo uso di Material Instance costanti e dinamiche, usufruendo dei vantaggi messi a disposizione. In particolare, tramite le MID è stato possibile sia manifestare lo stato interno del gioco, sia guidare il player nell'individuazione di determinate tipologie di oggetti. Nel primo caso, per esempio nel primo livello, al verificarsi della condizione di sovraccarico di una cabina elettrica, il material relativo al display che ne segnala lo stato cambia colore, coerentemente con la logica interna del gioco. Altro esempio riguarda la variazione del parametro di emissione di material assegnati alle static mesh di lampade e simili, compatibilmente all'accensione o allo spegnimento della rispettiva luce (e dunque, tutto guidato a livello di blueprint).


Per quanto concerne l'uso delle MID per guidare il giocatore all'interno del gioco, si è scelto di far brillare gli oggetti che possono essere raccolti o esaminati, e che hanno un ruolo chiave all'interno del gioco (da un punto di vista delle dinamiche di gioco, ma anche in termini narrativi). Questa scelta è stata presa per guidare il giocatore nella distinzione tra quelli che sono oggetti chiave e oggetti scenici con i quali non è prevista alcuna interazione (bottiglie, fogli di giornale, detriti vari).


Volendo realizzare un gioco che abbia un'impronta esplorativa, è importante lasciare dei riferimenti utili al giocatore al fine di individuare gli oggetti chiave, senza necessariamente guidarlo verso la soluzione. Si è deciso di non far brillare tutti gli Interactable principalmente per ragioni visive. Inoltre, un approccio del genere potrebbe semplificare fortemente la sfida, dando l'impressione di voler fornire immediatamente la soluzione al giocatore, condizionando negativamente l'esperienza di gioco. Tuttavia, qualora fosse stata data all'utente la possibilità di scegliere il livello di difficoltà, un criterio di scelta sarebbe potuto ricadere proprio su questo aspetto, ovvero far brillare tutti gli oggetti (difficoltà bassa), farne brillare alcuni (difficoltà media) oppure nessuno (difficoltà alta).

Masked Material

I Masked Material sono materials a cui vengono applicate delle maschere binarie: opaco (1), trasparente (0). Non si ha la possibilità di definire dei gradi di trasparenza, come per esempio avviene per i material Translucent. Attributo caratteristico di questa tipologia di material è l'Opacity Mask, una maschera che permette di specificare quali poligoni saranno trasparenti e quali opachi. All'interno del progetto, Masked Materials sono stati utilizzati per simulare oggetti aventi determinati dettagli, che se fossero stati modellati esclusivamente come mesh, avrebbero avuto un costo in termini di vertici maggiore. Un esempio concreto è rappresentato dalla realizzazione della rete metallica presente nel primo livello:

In fase prototipale, la soluzione temporaneamente adottata prevedere l'uso di una static mesh esattamente modellata come una rete metallica, contando complessivamente un numero di vertici superiore ad 800 000. In una fase successiva, usando un Masked Material, dei semplici cubi e dei cilindri, è stato possibile ottenere un risultato più che accettabile al costo di circa 4800 vertici, oltre 150 volte più basso rispetto la soluzione iniziale. È bene evidenziare che, per quanto si tratti di un risultato molto simile a quello di partenza, ogni forma di compromesso va sempre contestualizzata, dal momento che una differenza rilevabile esiste. In questo caso specifico, trattandosi di un oggetto di scena e non svolgendo alcun ruolo chiave all'interno del livello, il compromesso è più che accettabile, ma il discorso sarebbe stato diverso altrimenti. Per quanto le prestazioni vadano sempre tutelate (onde evitare di condizionare negativamente l'engagement), è altrettanto importante valutare se è il caso di adottare soluzioni accettabilmente più costose, ma qualitativamente migliori. Tuttavia, essendo l'obiettivo quello di fornire credibilità e realismo, un risultato migliore non avrebbe giustificato il degrado prestazionale.

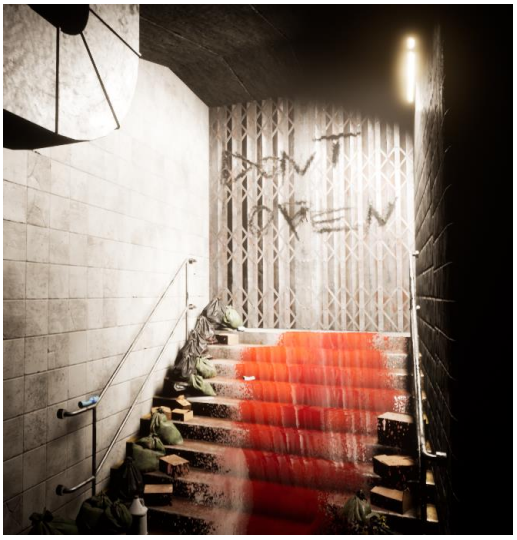
Decals

Le decals sono material proiettati su mesh e permettono, come necessario nei livelli trattati, di applicare dettagli come graffiti, scritte su oggetti presenti nel livello, imperfezioni della strada e così via. In questo caso, sono state utilizzate delle decals da me realizzate (le varie scritte presenti nei livelli) ed altre trovate sul marketplace. In dettaglio:

- Diverse scritte da me realizzate, contenenti dei messaggi ben precisi relativamente alla condizione sociale nella quale si trova il contesto trattato. L'uso delle decals, in questo caso, svolge un ruolo centrale dal punto di vista narrativo, dal momento che è possibile individuare quali sono le due fazioni coinvolte e capirne le rispettive posizioni:

In questa immagine è possibile osservare le grafie diverse dei due messaggi. Come già anticipato, ciò non è casuale.

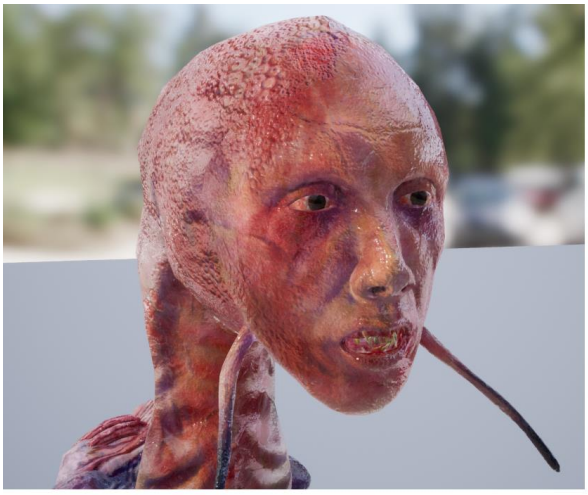
- Tracce di sangue, utili a comunicare al giocatore una situazione di pericolo, passato e potenzialmente imminente.



- Imperfezioni dell’asfalto, col chiaro scopo di donare credibilità alle scene di gioco.

Subsurface Scattering

Per Subsurface Scattering ci si riferisce a quell'effetto in cui si hanno oggetti costituiti da diversi livelli, in cui un livello superiore riflette parte della luce, mentre la restante parte viene fatta passare e riflessa dai livelli inferiori. Tramite l'editor dei material è possibile riprodurre questo effetto, selezionando Subsurface Profile come shading model e applicandone uno specifico. Sebbene il material dell'unico modello per il quale ho utilizzato questo tipo di shader fosse di per sé già estremamente dettagliato e credibile, ho comunque voluto aggiungere (senza scendere in particolare dettaglio) un effetto "livor mortis" alla creatura, essendo questa frutto di mutazioni avvenute in un corpo privo di vita.



Con SSS

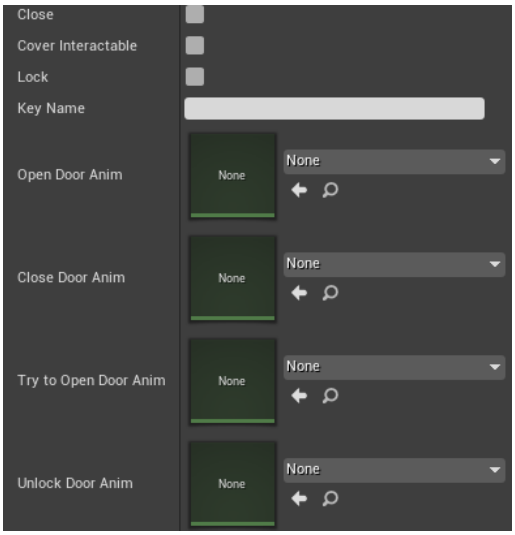
Senza SSS

Animation

Le animazioni svolgono un ruolo importante dal punto di vista dell'informazione riguardante gli attori animati, l'ambiente che li contiene e le azioni che precedono le animazioni stesse. Infatti, le animazioni possono essere usate per rendere dinamico un mondo che altrimenti parrebbe statico, oppure per manifestare gli effetti di determinate azioni o ancora (come spesso accade) per simulare dinamiche come il movimento, il parlato e così via. Tutti elementi che contribuiscono a rendere "vivo" il mondo che ospita l'esperienza interattiva e a favorire, dunque, l'immersione del giocatore in esso. Come anticipato nell'introduzione, un obiettivo chiave di questo elaborato è proprio l'immersività del giocatore, spinta dal realismo che può essere dato (anche) dalle animazioni. Questo obiettivo è funzionale all'esperienza che si vuole creare, dal momento che volendo realizzare un horror psicologico è importante che il giocatore si senta fortemente coinvolto e che percepisca gli effetti delle sue azioni sul mondo di gioco (e non solo). Nella cartella "MyAnimations" sono presenti tutte le animazioni da me realizzate (tramite Autodesk Maya), sfruttando mesh messe a disposizioni dal marketplace o trovate gratuitamente in rete (dopo essere state opportunamente trattate con Blender). Oltre a queste, anche il pawn relativo alla creatura del gioco (battezzata come "Cry") presenta delle animazioni relative al movimento e a diversi stati di riposo (il modello e le animazioni non sono state realizzate da me).

La blueprint ActorDoor

Relativamente agli effetti legati all'interazione del giocatore con l'ambiente di gioco, le animazioni realizzate hanno riguardato ActorDoor_BP (ovvero tutti quegli oggetti che possono essere definiti come porte), Button_BP e Switch_BP: essendo soluzioni molto simili tra loro, verrà riportato solo il caso di ActorDoor_BP. Seppur la soluzione suggerita dalla documentazione in merito all'animazione delle porte preveda l'uso di timeline, ho trovato più flessibile, riusabile e leggibile l'utilizzo di animazioni. Questa scelta è frutto dell'osservazione che porte a singole ante, doppie ante, porte scorrevoli o anche strutturalmente fantasiose (si pensi ad una porta sci-fi, per esempio) rimangono, concettualmente, delle porte. Inoltre, anche porte sbloccate per mezzo di una chiave o per mezzo di un cacciavite (o di un qualsiasi altro utensile che comporterebbe, inevitabilmente, delle animazioni diverse) sono concettualmente equivalenti: ho preferito, dunque, separare il sottosistema della logica da quello delle animazioni, garantendo la possibilità di realizzare animazioni anche particolarmente complesse e molto diverse tra loro (a patto che siano coerenti con il concetto di "porta") lasciando inalterata la blueprint coinvolta. ActorDoor_BP è una blueprint derivata da Interactable_BP e si presenta in editor come segue (tralasciando le voci già viste in occasione della blueprint Interactable_BP):

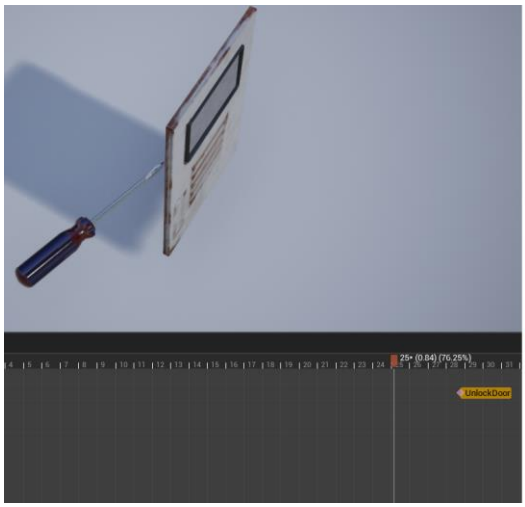


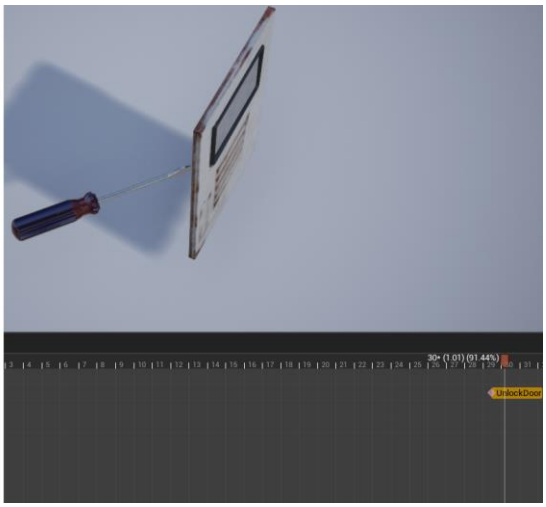
Close, Cover Interactable e Lock sono parametri booleani che indicano, rispettivamente, lo stato della porta ad inizio livello, la presenza o meno di oggetti Interactable coperti dalla porta (oggetti che è bene inizializzare come non-interactable, coerentemente con quanto visto in occasione della trattazione di Interactable_BP, se ad inizio livello la relativa porta è chiusa) e lo stato "bloccato" della porta ad inizio livello. Key Name indica il nome della (eventuale) chiave necessaria per sbloccare la porta, mentre le quattro variabili sottostanti sono di tipo Animation Sequence e riguardano, come anticipato, le animazioni specifiche che verranno realizzate esternamente. In particolare:

- *Open/Close Door Anim*, per le animazioni dell'apertura e della chiusura della porta;
- *Try to Open Door Anim*, per l'animazione relativa al tentativo di aprire una porta bloccata;
- *Unlock Door Anim*, per l'animazione relativa allo sblocco della porta mediante uno specifico oggetto chiave che può essere di qualsiasi natura (nei livelli trattati abbiamo una chiave vera e propria ed un cacciavite).

Comunicazione con il sottosistema dell'audio

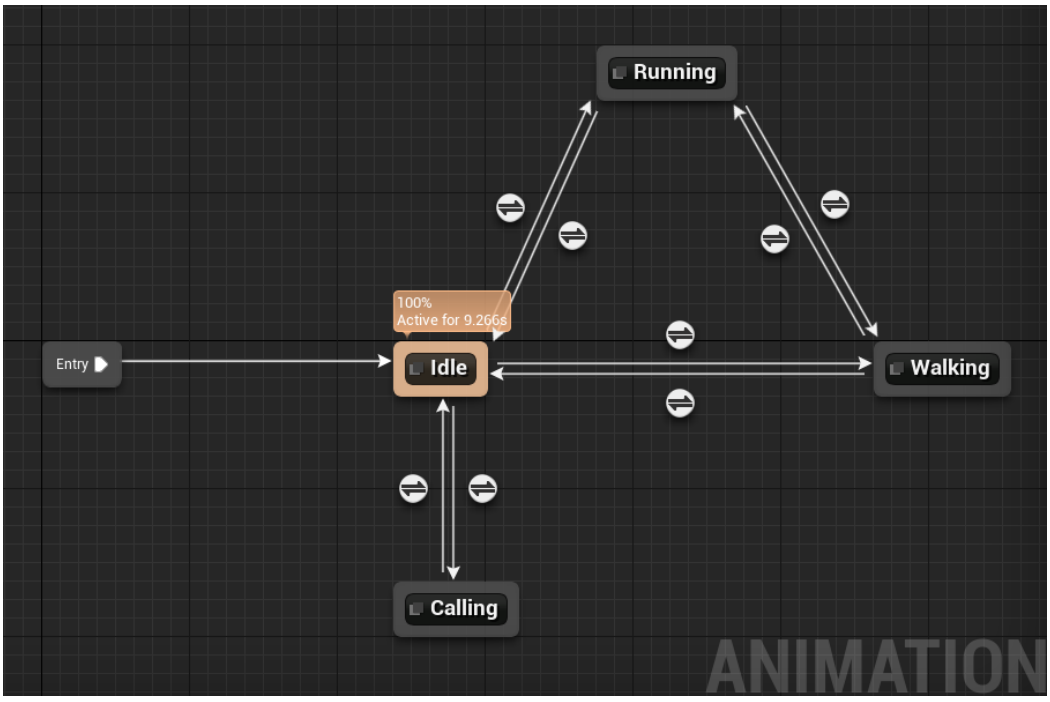
I suoni riprodotti coerentemente con le animazioni vengono gestiti a partire dal sottosistema delle animazioni, senza passare per la blueprint relativa all'oggetto e sfruttando il sistema delle notifiche. Questo approccio permette di sincronizzare determinati frame dell'animazione con il rispettivo suono, assicurando che la riproduzione avvenga in maniera sincronizzata e non influenzando la realizzazione del suono stesso (per esempio, dal punto di vista di "ritardi calcolati" e tutta una serie di modifiche delle tracce audio che ne altererebbero la riusabilità).





Cry

Come anticipato, il Pawn "Cry" presenta delle animazioni non da me realizzate, "limitandomi" alla cura della rispettiva Animation Blueprint, costruendo una specifica macchina a stati:



Nonostante la creatura compaia poche volte e per brevi istanti durante tutto il gioco, si è deciso di fornire ugualmente delle animazioni al pawn. L'obiettivo, infatti, è quello di mostrare una creatura non solo dalle fattezze inquietanti, ma anche "viva", vigile e (apparentemente) pronta a cacciare il giocatore: un autentico pericolo. Non essendoci dinamiche di combattimento ed avendo Cry solo un ruolo scenico, non sono stati aggiunti dei nodi relativi le animazioni di attacco, danno e morte (nonostante il modello le prevedesse).

Altre animazioni

Altre animazioni sono state realizzate sia per fornire dinamicità (dal punto di vista visivo, non ludico) alle scene di gioco, sia per contribuire alla realizzazione di alcuni eventi: è il caso della porta di ingresso dell'Hotel che, una volta raccolta la chiave e attraversato un trigger, viene avviata l'animazione (e il rispettivo suono) con il chiaro intento di sorprendere (spaventandolo) il giocatore. Animazioni dal peso prettamente scenico (ma significative dal punto di vista narrativo, dal momento che rappresentano un contesto di degrado urbano e sociale) sono prevalentemente presenti nel secondo livello.