Pt2: Game Sound overview

Re-Re-Repetition

Quando siamo **fruitori di opere di intrattenimento** - libri, cinema, teatro, etc... - tra noi e gli autori dell'opera si instaura una sorta di tacito accordo.

Da una parte nasce (inconsciamente) in noi volontà di **sospendere l'incredulità** per mettere momentaneamente da parte le nostre facoltà critiche, ignorare le incongruenze secondarie e godere appieno dell'opera di fantasia. Come spettatori ci lasciamo guidare e ci abbandoniamo alla narrazione.

Dall'altra parte l'autore si impegna ad introdurci e nel guidarci attraverso un percorso comune per raccontare una storia.

La sospensione dell'incredulità nasce da un equilibrio molto sottile, tanto più difficile da creare quanto da mantenere da parte dell'autore dell'opera, soprattutto in epoca moderna dove si è bombardati da flussi continui di informazione e da moltissime forme diverse di intrattenimento.

L'illusione si può spezzare a causa di una storia poco convincente, un personaggio poco caratterizzato oppure qualche battuta di dialogo non troppo azzeccata.

Anche il suono può talvolta essere causa di questa rottura: un effetto sonoro di troppo o uno mancante, un volume inadeguato all'interno del mix oppure un **suono ripetitivo**.

Un suono ripetitivo viene riconosciuto dal nostro cervello come un **pattern**. Il nostro cervello è abilissimo ad identificare pattern, strutture ricorrenti di ogni tipo.

In natura non esiste nulla che si ripeta esattamente identico a sè stesso e, quando questo avviene, il cervello entra in stato di _allarme_ riportando la nostra coscienza sul piano del reale più concreto.

Scott Selfon di _Microsoft Corporation_ parla della ripetizione nel videogioco al GDC 2014 in un talk intitolato "_Techniques for Fighting Repetition in Game Audio_".

La ripetizione è un male

Un suono che si ripete, sempre uguale a sé stesso, anche solo due volte di fila, è quanto di più innaturale si possa percepire. la ripetizione di un suono può quindi essere un male e deve essere evitata in tutti quei casi in cui provochi la rottura dell'illusione:

* **dialoghi**: immaginiamo una situazione in cui affrontiamo uno schieramento di nemici e ognuno di essi pronunci un grido di battaglia assolutamente identico a quello pronunciato dai vicini;
* **fooley**/**sdfx**: esempio tipico è quello dei _footsteps_, in cui lo stesso identico suono si sente ripetuto più e più volte durante il gioco. L'effetto suona innaturale e contribuisce a "_buttare fuori_" lo spettatore;

Ora, prendendo in esame il **cinema** perchè quanto di più vicino di può essere al genere videogioco (non fosse altro perchè contempla la fruizione attraverso il mezzo _schermo_), sappiamo che si tratta di un **media lineare**. Esiste ciò solo un modo di fruire del prodotto di intrattenimento: dall'_inizio_ alla _fine_.

La cosa può sembrare banale ma vedremo che non è così.
Quando si lavora alla sonorizzazione di un prodotto di questo tipo è dunque presumibile che si possa avere un certo grado controllo sulla ripetizione e si possa dunque evitarla quando non necessaria o addirittura nociva.

Ben intesi, in certi casi la ripetizione in sè, proprio perchè innaturale, può essere usata come espediente comico.
In altri casi invece la ripetizione diventa una sorta di licenza poetica, una sorta di firma sonora inequivocabile come il caso del famoso **Wilhelm screem**.

Arrivando al mezzo _videogioco_ invece è quasi scontato dire che si tratti invece di un mezzo di intrattenimento per nulla lineare.

Non è infatti possibile prevedere quando il giocatore compirà determinate scelte, non è possibile prevedere quali personaggi incontrerà, quali oggetti userà e quali no, quali le battute di dialogo che pronuncerà, quali difficoltà dovrà affrontare, nè come la trama potrà snodarsi durante il tempo di gioco. Esiste certo una sorta di intelaiatura predisposta dal _game designer_ ma ben lungi da essere veramente rappresentativa dell'esperienza di gioco individuale.

Molteplici sono le possibilità, per non dire infinite come _idealmente_ infiniti dovrebbero essere i contenuti sonori da riprodurre in modo da contemplare tutte le varie possibilità. Tuttavia **le

risorse sono limitate, sia in termini di costi che di tempi di realizzazione. Inoltre occorre pensare che il videogioco deve essere fruito usando supporti fisici o connessioni internet che intrinsecamente sono caratterizzate cciascuna con le proprie limitazioni.

Ricorrere al compromesso è dunque la soluzione più ragionevole e l'unica percorribile ma questo fa sì che nel videogioco diventi più difficile mantenere il controllo sulla ripetitività.

Quando la ripetizione è un bene

Ben intesi, nemmeno nel videogioco la _ripetizione_ è sempre da considerarsi un male a presindere. La ripetizione si rende talvolta indispenzabile per trasmettere e afferamre al giocatore un messaggio chiaro.

Anche nell'ambito dei suoni associati alla **UI** ad esempio, la ripetizione è quasi necessaria: un particolare suono sta ad indicare che si sta selezionando una opzione specifica e lo stesso suono ripetuto serve a sottolineare il fatto. Se l'interfaccia utente proponesse suoni diversi mentre l'utente si muove nell'interfaccia, anche interagendo con il medesimo pulsante, la cosa causerebbe non poca confuzione.

Al giocatore va data una sorta di feedback, una **audio reward** ad indicare che si sta facendo qualcosa di giusto o qualcosa che migliora la propria condizione (es. bere la pozione rinvigorente in "_Prince of Persia_").

Grandi matrici

Come punto di partenza, la ripetizione si può evitare assegnando a ciascun **evento** del gioco un **suono** caratteristico ben preciso: una spada che collide con uno scudo in legno non suona certo come un _footstep_ su una superficie erbosa.

Che cosa si intende per evento? All'interno del game engine, un _evento_ potrebbe essere qualsiasi verificarsi di una qualche condizione. Ad esempio:

* il click del mouse;
* il raggiungimento di un determinato punteggio;
* il superamento di un traguardo in un gioco di corse autombilistiche;
* il fatto di pronunciare una determinata frase oppure no;
* oppure ancora una collisione tra due corpi rilevata e riportata dall'engine di fisica.

Concentriamoci per ora su questa parta della produzione e chiediamoci: "_Quanti diversi suoni devono essere creati per sonorizzare il nostro videogioco?_"

Supponiamo di avere una oggetto `A` ed un oggetto `B`. Supponiamo poi che l'evento `A colpisce B` "_suoni_" esattamente l'evento `B colpisce A`. E' facile vedere come un singolo suono sia per ora sufficiente.

Aggiungiamo ora un terzo suono `C` ed arriviamo a 3 diversi suoni.

Complichiamo ulteriormente le cose e contempliamo anche collisioni fra oggetti della stessa tipologia: vediamo ora che gli _assets_ sonori da produrre sono diventati rapidamente il doppio.

Il caso esaminato qui è certo un caso fin troppo semplice ma, per chi di voi sia più interessato alla matematica in gioco, ci mostra che la formula usata per il calcolo è quella delle **combianzioni con ripetizione**.

Proviamo ad applicarla ora ad un caso più complesso: quanti suoni servono per sonorizzare un videogioco che contempli 13 differenti oggetti? poco più di 90 suoni diversi sono necessari in questo caso! Sorprendente no?

A ben vedere la cosa è più complicata di così: anche ammesso di avere un suono per ciascun evento possibile, cosa succede se lo stesso evento si verifica ripetutamente (come il collidere dei passi con il suolo o il colpire ripetutamente il nemico con la stess spada)? la risposta è _ripetizione_!

Non è sufficiente infatti che ogni **varietà** di evento abbia il proprio specifico suono, ma occorre che all'evento siano associati anche altre **varianti** dello stesso suono così da garantire un certo grado di varaibilità!

Anche perchè un oggetto non ha solo un singolo modo di "_suonare_"!

Un oggetto può essere colpito in un particolare punto o subire un impatto da un corpo che poi rimane in contatto. Può essere strisciato e causare una eccitazione da frizione, soffiato per eccitarne eventuali cavità d'aria oppure scosso da una vicina sorgente ad esso accoppiata ed essere portato in risonanza.

Inoltre tutto questo può accadere in aria oppure sott'acqua il che modificherebbe la velocità del suono e la sua propagazione. Oppure può accadere mentre l'oggetto, l'ascoltatore o il mezzo di propagazione sono in movimento causando l'effetto Doppler, etc...

Per fare un altro esempio si potrebbe considerare ancora quello dei _footsteps_: senza troppo eccedere, potremmo supporre che il gioco contenga 20 diverse superfici calpestabili come legno, metallo, ghiaia, sabbia, cemento e così via.

Per ottenere il numero di file audio da produrre, dobbiamo moltiplicare questo numero per quello dei personaggi più importanti per il gioco, i quali hanno bisogno d'essere caratterizzati particolarmente nei loro movimenti.

Questi personaggi possono poi indossare diversi tipi di calzari e possono camminare a diverse velocità: anche questo incide sul suono (il _correre_ non suona come il _camminare_).

All'interno dell'equazione potremmo poi inserire anche il paramtro _peso_, dovuto ad esempio dal numero di oggetti trasportati nell'inventario personale, oppure ancora considerare la pendenza del terreno, e così via...

In una produzione videoludica, il normale lavoro del sound designer, dove è richiesta tutta la sua abilità, è quello di preparare un grand numero di assets sonori "_riempiendo gli speadsheed_" (derivati in fase di preproduzione dall'_audio design document_) e registrare centinaia se non migliaia di suoni diversi. Una matrice ad incroci enorme che richiede un sacco di tempo e risorse per essere prodotta.

Come si vede la matrice degli assets sonori in un caso come quelli illustrati diventerebbe davvero gigantesca e multidimensionale.

Un caso di studio interessante potrebbe essere quello illustrato da _Alastair MacGregor_ della _Rockstar games_ al GDC 2014 rigaurdo agli assets del gioco GTA V

Realtime

Sempre maggiori sono gli assets sonori richiesti dai moderni videogames e la tendenza non cambierà radicalmente nel prossimo futuro.

Tuttavia, una possibile soluzione che aiuta anche nel ridurre la ripetitività, è quella di generare variazioni in ****realtime****, usando, ove possibile, le capacità offerte dai dispositivi ****hardware**** della piattaforma oppure programmando via ****software**** le features necessarie.

Ecco le principali accortezze che si potrebbero avere:

Volume / attenuation

Si tratta di un sistema facile ed economico da mettere in atto perchè comporta operazioni CPU base. Fornisce già una discreta illusione della profondità e della spazialità, tuttavia non è uno dei sistemi più incisivi per vincere la ripetitività;

Pitch changes

L'orecchio umano è meno abile nel riuscire a percepire differenze di intonazione. Inoltre questo sistema non è sempre applicabile, specie nelle linee di dialogo, per le quali invece una variazione di pitch risulterebbe particolarmente fastidiosa.

La causa di questo è da ricercare nel fatto che tutti questi cambiamenti nel pitch avvengono attraverso cambiamenti di sample rate, nel dominio del tempo quindi, con conseguente variazione nella durata.

Così facendo la voce diventa rapidamente innaturale (quando una voce è "_pitchata_" verso l'alto ad esempio si incorre nel problema legato alle armoniche superiori che, superata la frequenza di nyquist, finiscono per ripresentarsi nella parte bassa dello spettro, causando distorsioni e artifatti percepibili).

Inoltre da considerare la banda di trasferimento dati da disco a memorie/processore: con un pitch shift di un ottava sopra ci ritorviamo un audio dalla velocità raddoppiata il che necessità un più veloce accesso ai dati.

Pitch shifting più avanzati possono essee usati: si tratta dei pitch shifting che agiscono invece nel dominio della frequenza e si basano quindi sulla FFT. Sono pitch shifting con la preservazione delle formanti che garantiscono alle voci di mantenere la loro credibilità. Con questi tipi di pitch shifting inoltre lo streaming dei contenuti audio da disco resta coerente.

Filtering

Spesso l'hardware dei dispositivi su cui il gioco verrà giocato permette di effettuare operazioni di filtraggio in modo diretto senza costi di alcun tipo. In caso questo non sia possibile invece è comunque semplice implementare lo stesso tipo di filtri base come un LPF o un banco di BPF, via codice DSP.

L'uso del filtro è fondamentale per restituire senzazioni come la **distanza** (un filtro passa basso è ottimo per ricalcare il naturale _roll-off_ delle alte frequenze) oppure per sottolineare la presenza di **elementi attenuanti** come l'umidità ad esempio (ne parleremo meglio fra poco).

Applicare variazioni randomiche sul filtto di una voce inoltre, aiuta soprattutto a fornire la **sensazione di direzionalità**: nell'esperienza quotidiana, come ascoltatori ci accorgiamo che **la nostra testa** non è mai perfettamente ferma nello stesso punto e, anche piccole variazioni nella posizione, possono modificare lo spettro dell'audio percepito.

Usare i filtri sul sonoro e anche sulle voci permette di raggiungere un alto grado di variabilità. **Non importa se il filtraggio non rispetta** con accuratezza la fisica del **fenomeno**, il semplice fatto che ci sia un filtro fa suonare il tutto più naturale.

Anche questa è una tecnica relativamente economica da applicare.

Timing variations

Di nuovo in questo caso si può citare l'esempio dei _footsteps_. Il suono dei passi è infatti un suono molto articolato, per questo si può pensare di **spezzarlo nelle sue componenti individuali** e applicare variazioni casuali a ciascuna di queste per ottenere un risultato molto più ricco.

Un altro espediente è quello di **variare il tempo** che intercorre tra i vari elementi. In un suono ambientale che debba generare un tappeto costante di sottofondo, un particolare altamente riconoscibile come un **ululato** o il bubolare di un **gufo** può destare l'interesse dell'ascoltatore (il cervello si aggrappa a cose di questo tipo), pertanto una ripetizione a loop dell'audiofile, verrebbe immediatemente classificata come finta.

In questi casi meglio separe le diverse parti e lavorare con tempi diversi e casuali frapposti tra i suoni che possono essere più problematici.

Silence

Talvolta il silenzio può risultare un grande alleato quando si cerchi di sconfiggere la ripetitività.

Semplicemente contemplare il silenzio come una delle possibili opzioni nel ventaglio di delle possibilità, può aggiungere un po' di variazione in più!

Envelope

Applicare curve di inviluppo su volume o frequenza di taglio dei filtri aggiungono ancora ulteriore variabilità.

Positional variation

La variazioni applicata al posizionamento dei suoni nell'ambiente è importante; in particolare per tutti quei suoni che non hanno un proprio corrispettivo visivo.

Il _bubolare di un gufo_ nello scenario del bosco visto poco fa deve essere posizionato con cognizione e mantenere la propria posizione coerentemente con i movimenti del giocatore (la cosa potrebbe non essere scontata: pensiamo ad un audio ambientale prerenderizzato; mono, stereo o multicanale).

Inoltre, considerando che i suoni non sono quasi **mai omnidirezionali**, occorre considerare anche quale la **direzione** verso cui la sorgente sonora è orientata.

Un "_epic fail_" in questo caso sarebbe il percepire un gufo sul proprio lato destro quando, sopraggiungendo dal bosco, si sia ormai arrivati a costeggiare sulla destra un alto edificio.

Environmental variation

L'ambiente deve interagire con il suono prodotto dagli emitter: il mondo 3D in cui siamo immersi può essere sfruttato per creare variazione.

Per continuare con l'esempio del muro, il suono dei nostri passi dovrebbe in questo caso essere arricchito da una serie di early reflection. Ascoltando il suono così realizzato non soltanto si avrebbe una percezione di variazione ma anche una maggiore sensazione di immedesimazione nell'ambiente.

Un esempio potrebbe essere il plug-in **Reflect** di Wwise il cui compito è proprio quello di realizzare le early reflection dinamicamente su base della struttura 3D del mondo.

Ad oggi la cosa può sembrare scontata ma, forse qualcuno se ne ricorderà, prima che il calcolo dinamico delle occlusioni diventasse possibile e venisse implementato massicciamente, poteva capitare che si sentisse il suono di un nemico sopraggiungere dal lato ma che, voltandosi verso la direzione di provenienza del suono, non ci fosse nulla se non un muro. Il nemico c'era ma aldilà della parete.

Quando applicare tutti questi effetti?

Dialoghi

Aggiungere variazione non è sempre facile: il dialogo è la cosa più difficile da manipolare dinamicamente su cui è più difficile applicare quanto visto fino ad ora.

Per questo la variazione la si ottiene in fase di registrazione, con l'attore e il direttore del ADR che registrano battute con differenti intenzioni ed intonazioni.

Basti pensare che, già in questo modo, il risultato può sorprendere negativamente: è l'esempio di "_Beyond, two souls_". Nel gioco infatti capita talvolta che, seppure il protagonista risulti calmo nell'atteggiamento, ad esso venga associata una battuta recitata in modo concitato e aggresivo. Forse questo il risultato di una produzione dei files di localicazzazione del tutto decontestualizzata dalla reale situazione che si vevrebbe nel gameplay.

Interessanti saranno i futuri sviluppi dell'**intelligenza artificiale** per la sintesi vocale la quale potrebbe presto soppiantare la necessità di disporre di _voice talents_ in carne ed ossa per la registrazione di linee di dialogo (vedi il progetto **WaveNet** ad esempio).

Nel caso dei dialoghi eventualmente si possono applicare modifiche in real-time come il **voice-masking** (usato tra l'altro per ridurre il gender gap (flip gender) nell _interviste di lavoro on-line_, inducendo un artificiale "_gender flip_") oppure ancora per una "_radio-lizzazione_" (voice coming from the radio).

In tale caso è possibile aggiungere più o meno rumore, crakcles o distorsione in modo dinamico e in termpo reale.

Altro esempio è il filtering real time che si applica per simulare la voce proveniente da una comunicazione telefonica. in genere utilizzando un filtro passabanda da 300Hz a 3000Hz.

Resta comunque il fatto che la stessa frase non può essere mai detta identicamente 2 volte di fila! E' inaccettabile all'ascolto.

Un altro aspetto interessante che dovrebbe essere gestito è nell'**interazione di più characters** durante il dialogo: come individui nella quotidianità del mondo reale, siamo naturalmente in grado di ascoltarci l'un l'latro e aspettare il nostro turno per esprimere la nostra opinione. In caso di sovrapposzione è possibile interrompere l'interlocutore alzando la voce oppure restare in silenzio fino a che colui che parla non concluda.

Un gioco non è in grado di fare questo. Interessante sarebbe capire come ed in che modo introdurre sistemi intelligenti per simulare questo tipo di interazioni sociali per restituire una suono più verosimile.

Una nota va spesa per citare i sistemi di **speech recognition** come shortcut sostitutivi all'uso dei controller fisici per agire all'interno del gioco (implementati ad esempio in titoli come "_Mass Effect 3_", "_The Elder Scrolls V: skyrim_").

Importante comunque generare variazioni come la già citata _positional variation_ considerando che il suono proveniente dalla bocca non è direzionale e pertanto va opportunamente filtrato a seconda dei movimenti.

Sound fxs

Come già detto, le variazioni hanno ampia applicazione sugli effetti sonori.

Music (case study: iMuse)

Per la musica si potrebbe aprire un intero capitolo a parte parlando di composizioni interattive, musica generativa/algoritmica, passando per iMuse, Farnell, etc...). Forse in una lezione futura :)

Per ora ci basti dire che anche la musica è una elemento che può certo avvantaggiarsi delle tecniche di variazione fino ad ora discusse.

TODO: caso d'esempio "minigame del combattimento"

Un sistema davvero interessante è quello ideato dai compositori Michael Land e Peter McConnel nel 1991, quando all'epoca lavoravano ai videogiochi di avventura della _LucasArts_: **iMuse**.

Nasce nel 1991 (brevettato nel 1994); è un sistema che premette l'introduzione di componenti di audio dinamico in un linguaggio di scripting. Fondamentalmente iMuse è un database di sequenze musicali che possono contenere **punti di decisione** o **markers** all'interno delle tracce.

Il sistema, utilizzando eventi SysEx nei file MIDI, permette l'interazione tra le azioni del giocatore e il sonoro del gioco.
Gli eventi in questione sono di due tipi: **markers** e **hooks**.

* Un _marker_ viene inserito nel file MIDI nel punto che, una volta raggiunto dal lettore MIDI, deve triggerare l'esecuzione di un particolare comando da parte dello script del gioco. Il comando in questione è inserito in una lista (_coda fifo_) è ne viene attivata l'esecuzione non appena il MIDI player raggiunge un marker con un determinato ID. I comandi possono essere qualsiasi cosa, dal fade in/out alle pause;
* Un _hook_ contiene un ID e l'azione da eseguire una volta che l'hook viene raggiunto. Lo script lancia un comando che si occupa di aspettare che un certo hook venga incontrato (_callback_), e quindi di mettere in esecuzione il comando contenuto in quest'ultimo.
Gli hook si distinguono in vari tipi, quali ad esempio salti, trasposizioni, abilitazione/disabilitazione di strumenti.

Vediamone un paio di esempi sfruttando il motore _ScummVM_ e giocando a _Monkey Island 2: LeChuck revenge_ (nota: nella particolare dimostrazione usiamo una emulazione software della scheda _Roland MT-32_, all'epoca lo stato dell'arte dell'audio nel mondo videoludico);

Game audio engine tradizionale

Il _game audio engine_ è una componente del game engine, oppure un modulo _middleware_ da affiancare ad esso, che si occupa della gestione di tutto ciò che è suono all'interno di un videogioco.

Quali sono i compiti e le caratteristiche principali di un game audio engine tradizionale?

Random

Come abbiamo ampiamente detto poco fa, la randomicità può essere una grande risorsa per aggiungere variabilità.

Il game audio engine dispone di una serie di strumenti integrati per aggiungere variabilità pressochè a ciascun parametro disponibile.

Switching

Ricordiamo che abbiamo sempre a che fare con **risorse limitate**. Quando parliamo di _switching_ ci riferiamo a tutti quelle logiche e meccanismi di **allocazione dinamica delle voci**, mirate a dare priorità ai suoni e discriminare la loro riproduzione.

Nei sintetizzatori polifonici esiste lo stesso tipo di concetto e si parla di **voice stealing**.

Un esempio di voice stealing nel videogioco lo si ha in _Super Mario Bros_ (1985), dove il sound engine agisce sulla voce assegnata alla melodia principale (il suono più acuto), deallocandola e riassegnandola per la sintesi degli effetti sonori delle monete, mentre lo stesso avviene per gli effetti sonori del salto per la seconda voce.

Possiamo venderne un esempio live utilizzando il bell'emulatore **FCEUX** il quale ci consente anche di esaminare le sinagole voci nel dettaglio. Il sistema sonoro del NES era costituito da una chip PSG (Programmable Sound Generator) proprietario in grado di suonare fino a 5 voci simultaneamente: 2 _quadre_, 1 _trinagolare_, 1 _noise_ e una per la riproduzione dei campioni _pcm_.

Lo switching può essere guidato anche dal ruolo che il particolare suono riveste all'interno della **narrazione**: in una sitauzione in cui sono presenti molti suoni, sono quelli meno importanti ai fini di quanto si deve raccontare ad essere sacrificati per primi.

Ai suoni in riproduzione in un videogioco sono quasi sempre associati **valori di priorità** o **rating** in modo automatico o semi-automatico in base alle scelte del giocatore. Questi valori numerici sono fondamentali nella scelta di quali voci deallocare in un scenario di _voice stealing_/switching.

Blending

Il _leitmotif_ è: le **risorse sono limitate**.

Crossfade parametrico tra campioni diversi, quello che nella sintesi prende il nome di **multisampling**. Questa tecnica è implementata in gran misura nei campionatori i quali infatti rispondono a diverse velocity di tocco con un mix tra campioni corrispondenti.

In un gioco, immaginiamo una caduta di un oggetto da diverse altezze; questo comporta intensità diversa ma non solo, anche variazione timbrica.

Mixer, Grouping and Buses

Molti game audio system incorporano un mixer del tutto analogo a quallo in uso nelle grandi produzioni: un banco large frame con gruppi, mandate e ritorni, etc...

La differenza è che, mentre per una produzione tradizionale la configurazione del banco rimane statica, praticamente invariata lungo tutta la durato di Un medesimo brano o album, nel caso di un videogioco il mixer deve spesso riconfigurarsi del tutto in pochi istanti.

Un esempio potrebbe essere il passaggio da una sitauzione _in-game_ ad un _menù di interfaccia_ (pausa o salvataggio).

Real time controllers

Il game sound engine deve stabilire con il game engine un'interfaccia per ricevere (e inviare) parametri real time da utilizzarsi per controllare interattivamente controlli quali volume, pitch o frequenza di taglio.

Alcuni esempi di parametri utilizzabile per modificare i suoni i nriproduzione potrebbero essere:
* il livello di _salute_ per il player;
* il _carico_ per un automobile in corsa;
* il _punteggio_ della partita a regolare il livello di eccitamento del pubblico sugli spalti dello stadio.

Positioning, Attenuation & Dampening

Come funziona l'audio in un gioco: **emitters** sono oggetti nello spazio tridimensionale che emettono suono e uno o più **listeners** (mono, stereo o multicanale; va pensato come un array di microfoni), in genere solidale col player.

Ogni emitter nel 3D world è caratterizzato dalla presenza due sfere ad esso concentriche che dividono lo spazio in 3 volumi.

Attenuazione e smorzamento: un discorso legato alla **distanza** tra emitter e listener, grandezza geometrica ricavata dal modello tridimensionale, in base alla quale viene modificato in tempo reale l'amplificatore di livello e/o la frequenza di taglio di un filtro passa basso.

Lo stesso si applica in casi in cui ci sia un **ostacolo** tra emittere e listener: **occlusione** ottenuta con filtri opportunamente settati. Materiali diversi

In altre parole si tratta di un sistema che si occupa di calcolare in run-time le funzioni di trasferimento e i filtri da applicare al suono riprodotto per dare all'ascoltatore la percezione che gli emettitori si trovino immersi nello spazio virtuale circostante.

Il panning "_reale_" è più complcato rispetto ad una semplice variazione di volume tra i canali. panning, multicanale (stereo, 5.1, 7.1, ambisonic (non usato), binaurale);

Un esempio interessante per quanto riguarda il posizionamento del suono nello spazio lo si ha ad esempio nel gioco "_Hellblade: Senua's sacrifice_", sviluppato da _Ninja Theory_ e pubblicato pochi giorni fa.

In questo gioco l'**audio binuaurale** diventa importantissimo per rappresentare con efficacia le voci interiori della protagonista che soffre di un disturbo psichico.

A proposito di binuaurale, mai sentito parlare del fenomeno denominato **ASMR**?

Nel gioco l'audio è talvolta utilizzato come mezzo (quasi) esclusivo per riuscire ad orientarsi nel mondo tridimensionale.

Ambience

Parlando di sample, quando si registra si predilige il suono diretto e si fa di tutto per escludere quello riverberato
Questo perchè il suono d'ambienza viene calcolato in tempo reale da processori dedicati (reverb, delay, doppler effect, filtering, fast realtime convolution).

Dialogues

Il game audio engine deve essere in grado di interfacciarsi e gestire complessi **database** di informazioni. Uno di questi è rappresentato dall'insieme degli audio file associati a tutte le varie linee di dialogo (in una o più lingue) presenti nel gioco.

Music

Il game audio engine deve essere in grado di gestire l'eventuale colonna sonora musicale interattiva (vedi ad esempio il sistema _iMuse_).

Alignment

Uno scenario in cui più giocatori prendono parte ad un partita multiplayer.
Un server preposto al controllo e al master clock per la ricezione e ridistribuzione dei pacchetti.
A seconda della contingeza ci possono essere latenze che si sommano e si accumulano, e possono essere diverse da caso a caso, e da giocatore a giocatore e cambiare nel tempo.

Il game engine, e più nello specifico l'audio engine per quanto concerne il suono, deve essere in grado di gestire situazioni come questa e di riordinare opportunamente i pacchetti in arrivo per dare un audio sempre corerente.

Decoding, data streams

L'audio engine deve interfacciarsi con quella parte di software in carico di gestire gli accessi al disco e che si occupa del memory management. Da ricordare che l'audio è immagazzinato sul supporto in forma compressa in modo da risparmiare spazio di archiviazione, il che, nel momento della riproduzione, impone un pre fetching e un buffering per la decompressione prima dell'effettiva riproduzione.

alcuni esempi di codifice dei file audio potrebbero essere: PCM e ADPCM, MP3, Ogg Vorbis, Speex, Opus, flac, musepack, per Xbox: XMA e WXMA, per Playstation: Atrac AT9