Suono come modello "event based/data driven"

A ben vedere nell'ultima parte della sua storia, come analizzato fino ad ora, il suono nel videogioco si presenta come un modello guidato dai dati (**data driven model**).

In altri termini, è costituito da una moltitudine di file audio raccolti in un database e che vengono messi in riproduzione all'occorrenza, in seguito al verificarsi di particolari eventi (**event based**).

Ai sample che vengono riprodotti si possono certo applicare delle **modificazioni in tempo reale** come abbiamo già detto (attenuazione dovuta alla distanza, combinazioni e layering, random e granularità).

A ben vedere però questo sistema basato sui sample audio sembra in contraddizione netta con il dominio visivo, caratterizzato invece da un comportamento **continuo** e guidato da uno stream di parametri piuttosto che da eventi **discreti**.

Grafica per la ricerca del realismo

In un gioco che si basi pesantemente sull'aspetto grafico, l'immagine mostrata a schermo è fondamentale per restituire la senzazione di **realismo**.
Interessante notare comunque che l'immagine che vediamo quando giochiamo **non esiste** prima del run-time!

Consideriamo un semplice modello 3D costituito da 4 facce triangolari: occorrono 3 (OpenGL ne usa 4 in realtà) valori numerici corrispondenti ai 3 assi cartesiani per identificare la posizione di ciascuno dei suoi vertici nello spazio tridimensionale.

Un modello estrapolato da un moderno FPS tuttavia è composto da una moltitudine di poligoni, centinaia se non migliaia (_high poly_). Questi vertici e la loro configurazione nello spazio costituisce la **mesh** del modello ma da sola, non basta per creare l'illusione di realismo.

Serve una **texture** da poter mappare sulla mesh che riproduca fedelmente le caratteristiceh visive come colori e dettagli dell'oggetto ([**UV mapping**]).

Perchè il modello possa essere visto nel mondo tridimensionale occorrono **luci**: ogni oggetto dunque, colpito dai raggi irradiati da tutte le fonti di luce presenti nel mondo virtuale, verrà descritto in modo ancora più realistico, in più se il modello riporta alcuni valori per le grandezze fisice associate ai materiali di cui è composto, come coefficienti di riflessione e assorbimento, e così via, l'effetto potrà essere ancora più realistico.

Moltiplichiamo il tutto per la moltitudine di modelli simultaneamente all'interno del mondo 3D virtuale e avremo quanto computato da una speciale componente del game engine, il **rendering engine** (che traspone il tutto su una superficie flat 2D), coadiuvato da componenti hardware che aiutano ad accelerare l'immane qunatitativo di calcoli richiesti.

L'interità dei modelli e delle loro mesh non sono fondamentali soltanto per ottenere una immagine 2D in uscita ma anche per creare la cosìdetta **word geometry** per il il calcolo e dell **collisioni**, indispensabili per prevedere e computare i comportamenti fisici.

Tutto questo è appannaggio del **physics engine** che non si occupa solo di collisioni ma valuta l'interà fisicità del mondo virtuale in cui siamo immersi: masse, densità, velocità e accelerazioni, forze, torsioni.

A tutto questo si aggiunge la componente di **intelligenza artificiale** che ha il compito di simulare comportamenti "_intelligenti_" per tutti quegli attori che, nel gioco, non sono comandati da un player umano.

Ebbene tutto questo viene calcolato di continuo, sempre in funzione dei dati di movimento ottenuti dalle azioni del giocatore, 60 se non più volte al secondo.

Suono come processo

Il sample audio è una registrazione, e come tale si tratta di un qualche cosa fissato nel tempo: una registrazione cattura la perturbazione della densità dell'aria, l'effetto di un movimento nello spazio in un particolare istante ma non ci dice nulla in merito al comportamento.

In questo senso l'audio fruito attraverso i campioni resta un procedimento **statico**, come un **interruttore** che può essere solamente acceso o spento, una **fotografia** fissa ed immutabile anzichè vivida e dinamica.

Ci potremmo chiedere quindi che significato ha la parola **realismo**? Premesso che il realismo non è una condizione indispensabile, anzi, ci sono videogiochi (basti pensare a PacMan, o a diverse produzioni indipendenti come Fez, Limbo, etc...) che tutt'altro ci propongono rispetto ad una esperienza "_realistica_", ce ne occupiamo perchè la ricerca del realismo sembra essere stata forse "_IL_" motore principale per lo sviluppo tecnologico nell'abito dei videogames.

Basti pensare che si è passati al massiccio uso della grafica tridimensionale non appena le tecnologie lo hanno permesso (questo vale anche per il mondo dell'**animazione**). Si ricerca sempre una maggiore definizione dei modelli. Ci si avvale del supporto di potenti motori di fisica per simulare fedelmente i comportamenti quali collisioni, esplosioni, attrazione gravitazionale, spinta del vento, forze etc...

L'intelligenza artificiale contribuisce a rendere sempre più credibili i movimenti e le azioni di attori e creature npc e la lista potrebbe continuare a lungo.

In abito audio invece quello che si fa, in pratica, è "**_premere il tasto play_**" quando serve, il che sembra un po' **riduttivo**, soprattutto per il fatto che:

* il suono riveste un ruolo importante di guida emotiva all'interno del gioco;
* l'audio è una parte importante anche per la narrazione;
* contribuisce alla percezione di _verosimiglianza_ (vedi più avanti);

Non è sempre stato così

Un approccio _event based/data driven_ in somma rende il suono malamente accoppiato con il motore di gioco sottostante, incoesivo con l'esperienza complessiva. Eppure non è sempre stato così.

All'inizio della storia dei videogame, delle console e dei computer, era l'**audio** la motivazione principale che ha **guidato lo sviluppo tecnologico**.

L'audio veniva generato in tempo reale e rispecchiava linearmente le azioni del giocatore e le reazioni dell'engine. L'audio veniva sintetizzato in tempo reale. Questa tendenza si è interrotta indicativamente attorno alla seconda metà degli anni '90, momento storico dove si può collocare la comparsa sul mercato dei prini CD e che vede il diffondersi dell'audio campionato ad alta qualità (44100@16bit).

Case study: SID

Il **SID** (Sound Interface Device) era il chip sonoro utilizzato dal Vic20, **C64** e C128, sviluppato da _Robert Yannes_ di _MOS technology_ il quale, oltre al background tecnico, ne sapeva molto anche di musica.

Il suo intento era sviluppare un chip di sintesi sottrattiva totalemente differente dai sistemi sonori presenti nei computer dell'epoca e il risultato fu qualcosa di innovativo.

Il chip era programmabile in BASIC o in linguaggio macchina e possedeva molte caratteristiche interessanti, le principali elencate qui di seguito:

* 3 generatori di suono (voci);
* 4 diverse forme d'onda disponibili (sawtooth, triangle, rectangle w/ pulse width modulation, noise);
* 3 modulatori d'ampiezza (adsr);
* 1 controllo di Master volume (in 16 steps);

Erano possibili ****effetti**** come la _ring modulation_ o l'[_hard sync_]tra gli oscillatori.

Inoltre il SID disponeva di un **filtro programmabile** (low pass, bandpass, high pass), con frequenza di taglio e risonanza selezionabile. Il funzionamento del filtro era possibile grazie alla presenza di alcuni componenti analogici che completavano il circuito: 2 capacitori. Questa caratteristica rendeva il suono del SID unico e difficilmente replicabile fedelmente, anche al giorno d'oggi.

Ecco qui di seguito un piccolo programma d'esempio:

Da ricordare che, usando l'istruzione `poke` del linguaggio di programmazione BASIC è possibile accedere e scrivere sui singoli registri interni del SID specificando sia l'indirizzo, sia il valore numerico da memorizzarvi.

Il programma usa la prima voce impostata come _onda triangolare_ per riprodurre una nota A440 di durata pari a circa 500 millisecondi. Le istruzioni usate sono:

1. indirizzamento del SID `SI=54272`;
2. impostazione del volume master (registo 4: `L=SI+24` con valori da 0 a 15, volume basso/alto);
3. impostazione dell'envelope con 4 bit per ciascuna fase = 2 byte per la memorizzazione (registri 5 e 6: `A=SI+5` e `H=SI+6`, rispettivamente per attack/decay e sustain/release);
4. impostazione della frequenza (registri 0 e 1: `FL=SI` e `FH=SI+1`)
5. selezione dell'onda (registro 4). Alcuni valori possibili sono:
   * tringolare: 17;
   * dente di sega: 33;
   * rettangolare: 65 --> parametro addizionale "_duty cycle_" (registri 2 e 3 `TL=SI+2`, `TL=SI+3`);
   * noise: 129;
6. ciclo _for_ per la durata della nota.

Rob Hubbard

Il SID è uno chip che dispone di sole 3 voci ma non è detto che nelle mani di capaci musicisti programmatori non possa ricreare la polifonia e la ricchezza timbrica di un ensamble molto più numeroso

Si tratta della in game music del gioco **"Commando"** uscito per **C64** nel **1985.** Ecco le tracce separate per apprezzare meglio la ricchezza di variazioni, il timing e intuire l'ingegnosità dei programmi scritti da Hubbard:

Anche dalle immagini che mostrano la forma d'onda della parte iniziale della prima voce si può comprendere la complessità della lavorazione:

> Il lavoro di Hubbard inoltre è un mirabile esempio di come spesso si riescano ad ottenere risultati ammirevoli partendo da risolrse limitate o necessità stringenti.
> {: class="note"}

Attenzione, questo ****non significa che in passato i sample non venissero utilizzati**;** Nintendo, SEGA e ancora prima nelle coin-op, dove possibile, si inserivano sistemi DAC in grado di riprodurre, seppure non con la stessa qualità CD, campioni e sample pre-registrati, soprattutto per la voce. La ricerca del "_realismo_" tuttavia ha infine soppiantato i sistemi di sintesi tradizionale.

Niente trucchi da quattro soldi!

Sotto questa luce, i "_trucchi_" descritti poco fa, usati normalmente per ridurre o mascherare la ripetitività, sono in realtà dei rimedi temporanei.

Un modo alternativo, mutuato dalla storia del videogioco, è quello di pensare il sonoro come ****sintetico**:** in questo modo due suoni associati allo stesso evento non suonerebbero mai esattamente allo stesso modo.

Si tratta in fondo di una caratteristica peculiare della sintesi sottrattica, la quale fa uso del rumore come forma d'onda base.

Anche se le differenze potrebbero essere estremamente sottili, il nostro ****cervello**** ha la capacità particolare di riconoscere queste sorgenti come **"_vive_"**.

Inoltre, un approccio _data driven_ come questo ha i suoi svantaggi vediamo un paio di esempi:

La porta

Supponiamo di avvicinarci ad una abitazione sconosciuta. L'intento è quello di entravi senza destare l'attenzione di chi sta all'interno. Nel gioco guideremo il nostro avatar fin sulla soglia e, a questo punto, a seconda dell'ispirazione del momento, potremo aprire la porta di colpo e preciparci all'interno oppure socchiuderla lentamente e magari interromperci al primo sospetto di non essere soli.

Secondo l'approccio tradizionale _sample based_ un campione preregistrato di scricchiolio viene riprodoto non appena si verifica l'evento "_collisione_" tra il nostro avatar e l'oggetto _porta_. Specie se l'audio file è lungo si nota che la riproduzione dell'audio file prosegue, anche se l'azione sulla porta si interrompe dopo breve.

Anche nel caso il game sound engine utilizzasse meccanismi di dissolvenza in uscita per temporizzare correttamente il livello, comunque risulterebbe chiara la disomogeineità tra i domini grafico e uditivo.

Immaginiamo cosa accadrebbe se poi scostassimo la porta ripetutamente, avanti ed indietro...

Lastra percossa

Alcuni oggetti come le campane o i tubi creano differenti suoni in base al proprio stato. un tubo che è già in vibrazione e che venga colpito in un punto diverso, incorporerà le nuove eccitazioni nel pattern di vibrazione a creare un suono diverso rispetto a quello che avrebbe fatto se colpito in stato di quiete. Niente di simile è ottenibile usando i campioni.

Immaginiamoci con un arma di fronte alla fortificazione di un piccolo bunker dal quale dobbiamo stanare i nostri nemici. Cominciamo a sparare e colpiamo ripetutamente una delle lastre di metallo che rivestono l'esterno della costruzione.

Ad ogni impatto (di un proiettile sulla lastra) la lastra viene eccitata e di tutti i possibili modi strutturali saranno quelli primari ad assorbire il maggior quantitativo di energia e, come risultato, percepiremo sempre più chiaramente le frequenze di risonanza della lastra mano a mano che i proiettili continuano ad incidere su essa.

Lo stesso effetto non è ottenibile riproducendo lo stesso audiofile ripetuttamente per ogni nuovo impatto.

La sintesi in musica

Se, i videogiochi moderni hanno via via preferito i campioni, i paradigmi di audio sintetizzato, più tradizionali se vogliamo, sono sopravissuti in una parte del mercato legato all'audio: la **musica** e gli **strumenti musicali**. In questi campi ci si accorge subito delle limitazion intrinseche dell'audio per campioni (il sample è un _tradimento_ della realtà), per questo la ricerca e lo sviluppo di nuovi sistemi di sintesi continua a progredire.

Nascono i primi sistemi di sintesi basati su **modelli**: i fenomeni della fisica del suono vengono studiati e sintetizzati in modelli matematici e poi innestati all'interno dei circuiti integrati o dei software per computer.

Quando si vuole simulare i suoni dei vecchi sintetizzatori analogici, si sviluppano tecnologie volte a riprodurre in digitale tutte le idiosincrasie dei componenti elettrici discreti che li costituivano e nascono i modelli [**virtual analog**].

ad esempio:
* la tecnologia [**True Analog Emulation (TAE)**] di **Arturia**, usata in plugin come il [mini V] o l'[arp-2600].* Il [**Pod**] di **Line6** implementa un suond engine in grado di simulare molti preamplificatori per chitarra e basso, cabinets e acustia degli ambienti. Anche l'italiana [**Acustica-Audio**] usa molta [tecnologia] interessante all'interno dei loro plugin come il [Nebula3] ad esempio, che "_is a multi-effect plug-in that is able to emulate and replicate several types of audio equipment and uses libraries which are created using a sophisticated “sampling approach” making it possible to “record” aspects of the sound of audio devices and play them back_". Interessantissimi anche i lavori dell'italiana [**AudioThing**]; I virtual instruments della svizzera [**Togu Audio Line (Tal)**];

Quando invece si vuole simulare strumenti musicali acustici o elettroacustici nascono modelli volti a riprodurre i fenomeni fisici delle sollecitazioni, vibrazioni, attenuazioni dei materiali: nasce il [**physical modelling**].

* [**Arturia Stage-73 V**]; i pianoforti virtuali di [**PianoTeq**]; **Antares** [**Auto-tune**] oppure [Throath]; Celemony [**Melodyne**] e [**Capstan**], [**Izotope RX**]. Altri esempi

potrebbero essere l'[**Aerophone AE-10**] di **Roland**, che fa uso dell'engine di dintesi per modelli [**SuperNATURAL**]in parallelo con la tradizionale dintesi per campioni; [**Native Instruments B4**], etc...

> Solo un appunto interessante a proposito di [**Pianoteq**]: il peso del software è di soli **40MB**. In confronto al [**Ravenscroft** Virtual Piano 275] - che fa uso di **17.000** samples - che richiede **6GB** per funzionare (perchè i dati sono compressi) e un HD a stato solido per garantire una performance ottimale!
> {: class="note"}

Esistono da diversi anni scuole di pensiero volte a traslare questi metodi di sintesi, da tempo diffusisi sul mercato e molto apprezzati per le loro qualità, nel mondo videoludico e dell'interazione più in generale.

Pionieri di questa filosofia di pensiero sono persone come [**Perry Cook**] e [**Andy Farnell**], solo per citare i più noti, i quali ritengono possibile derivare dagli studi fatti fino ad ora, modelli finalizzati non tanto a simulare suoni appartenenti al dominio musicale ma piuttosto volti a sintetizzare una moltitudine di classi sonore associate ad oggetti e fenomeni più generali: così da rendere possibile la sintesi, potenziale, di ogni tipo di suono possibile.

L'audio procedurale

Da questo punto di vista il suono nel videogioco diventa un modello inteso come processo (**sound as a process**), in contrasto con il paradigma precedente di _event driven/data driven model_.

Il termine spesso associato a questo paradigma è **audio procedurale**, eccone una definizione:

> > “Procedural audio is non-linear, often synthetic sound, created in real time according to a set of programmatic rules and live input.” - "[An introduction to procedural audio and its application in computer games].” by Andy Farnell

Processo guidato da uno stream continuo di dati provenienti dall'interazione dell'utente, manipolati e usati come parametri per controllare in tempo reale una serie di algoritmi che sintetizzano audio.

A ben vedere il concetto di audio procedurale non ci è del tutto estraneo; un esempio a cui siamo abituati è il _riverbero digitale_ che usiamo tutti i giorni sottoforma di plug-in o di outboard fisico in studio.

TODO: classi di modelli (tassonomia)
TODO: perchè scegliere il procedurale e quando sceglierlo.

Vantaggi e svantaggi del paradigma procedurale

Tra i **vantaggi** possiamo considerare:

Differimento

metre l'atto di registrare un suono è un azione che fissa nel tempo senza lasciare alcune possibilità di intervento successivo, l'audio procedurale è dinamico e lascia che molte delle decisioni, anche strutturali, vengano rimandate al real-time;

Variabilità

Caratteristica fondamentale del suono procedurale che garantisce la possibilità pressochè completa di modificazione del suono in tempo reale e di produrre in questo modo risultati sonori anche molto diversi tra loro pur facendo capo ad uno stesso modello.

Default forms

Dal momento che la **crescita** dei sounds assets è **combinatoria**, diventa difficile provvedere alla creazione di tutti i suond assets necessari mano a mano che il mondo virtuale cresce. Il vantaggio di un modello procedurale è che il suono può essere generato in modo automatico derivando le proprietà dagli oggetti presenti nel gioco.

Questo **non elimina** la figura del **sound designer**, il quale interviene laddove alcuni suoni necessitino di particolari caratteristiche perchè più importanti per la narrazione, ma **garantisce** che ogni oggetto abbia sempre un **suono di default** associato, senza incorrere così nel rischio che qualche evento sonoro non possa essere triggerato.

Costo variabile e dinamico (Dynamic Level of Details)

Il suono sintetico si dimostra vincente rispetto all'approccio _data driven_ quando si debbano descrive ampie scene con innumerevoli sorgenti sonore (tra 100 e 1000 sorgenti concorrenti).

I dati sottoforma di campione hanno un costo fisso (lo streaming dati da disco non cambia se il suono deve essere riprodotto completamente dry oppure deve subire real-time processings).
Inoltre, la densità di suoni che possono essere riprodotti in una scena ha un limite:
* sia in termini di accuratezza del mix dei vari suoni (più suoni = più difficile sommarli tra loro, problemi di dinamica e saturazione);
* sia dal punto di vista psicoacustico (alcuni dei contibuti sonori sono inutili all'atto pratico perchè non posso essere uditi per via dei diversi mascheramenti, in tempo o in frequenza).

Già quando si arriva a sommare 100 suoni simultanei, si può vedere che il contributo di ogni nuovo suono aggiuntivo, in rapporto al costo computazionale associato, diminuisce.

D'altro canto, una approccio sintetico può offrire un costo variabile ad ogni sorgente: un esempio potrebbe essere il suono di un bicchiere che si infrange cadendo a terra.

Un metodo sintetico potrebbe produrre un suono molto "_realistico_" e mantenere ancora un alto grado di correlazione audio-video rimpiazzando i frammenti perimetrali con singoli segnali sinusoidali o granuli di rumore, fornendo un alto grado di dettaglio per quie frammenti che cadono vicino al player.

Il paradigma procedurale permette il LOAD - Level Of Audio Details - (che sfrutta le conoscenze in ambito di acustica e psico-acustica) caricamento dinamico sulla CPU esattamente come già fa la parte grafica cone il **mipmapping**.

Consideriamo una esempio quale potrebbe essere un suono in grande lontananza:

nel caso di un motore audio tradizionale, al suono verrebbe applicato notevole processing DSP in tempo reale da far sì che il sample - attenuato e filtrato - dia l'impressione di provenire da lontano. Tuttavia, nonostante il suono ne risulti pesantemente modificato rispetto all'originale, comunque il costo in termini di risorse sarebbe invariato rispetto alla normale riproduzione di un audio file.

Nel caso il problema venga affrontato con il paradigma procedurale invece, soprattutto facendosi forti del fatto che il modello è stratificato e costituito da più parti indipendenti tra loro, il carico di lavoro potrebbe essere ridotto: certe componenti del suono, proprio per via del fatto che la sorgente sonora è ascoltata da grande distanza, potrebbero essere del tutto tralasciate rispetto ad altre. Il lavoro del processore sarebbe avvantaggato.

Lo stesso dicasi se il suono riprodotto da un modello si trova riprodotto in associazione con altri suoni che causerebbero il _mascheramento_ nel tempo o in frequenza di alcune sue componenti.

Real Time

Talvolta è semplicemente impossibile conoscere a priori quale sarà il suono da registrare (o il tipo di processing cui sottoporlo).

Esempio di audio procedurale in **"_No Man Sky_"** (raycasting per procedural occlusion, tool interno per sintesi di vocalizzazione di creature aliene: _VocAlien_).

Tra gli **svantaggi**:

Industrial inertia: you gotta ship titles

Al momento attuale non sembra ci sia interesse nell'implementare quanto necessario per inserire questo paradigma nel workflow per la produzione di giochi. Spesso in questi casi ci si scontra con metodi di lavoro consolidati che sono difficili da modificare, soprattutto per via dei costi e dei tempi necessari per la transizione.

New workflows, new skills

Il paradigma procedurale richiede personale che sappia operare in campo audio in nuovi modi. Per studiare modelli derivati o ispirati dal mondo reale e implementarli poi un una forma hardware o software richiede abilità e competenze che normalmente non fanno parte del bagaglio culturale di un sound designer tradizionale.

La sintesi è brutta (si fa per dire)

Permane la falsa concezione che la sintesi audio sia in qualche modo sinonimo di finzione (sintesi = suono "_di plastica_") e, come tale, sia qualcosa di insoddisfacende, di deludente.

In realtà non è così e, se anche lo fosse, il ragionamento non sta in piedi in quanto il ****realismo**** non serve!
Lo sanno bene i sound designer e tutti coloro che, in generale, hanno già qualche esperienza nel mondo dell'intrattenimento, il realismo spesso delude.

>"Il tutto è più grande della somma delle sue parti." (**Aristotele**, Metafisica)

Quello di cui c'è bisogno è il ****verosimile**** (come dice molto bene [**Chion**]) o addirittura dell'****hyperrealism**** ("_more than reality_").

The Future

In un futuro presumibilmente non troppo lontano, il paradigma procedurale avrà preso ancora più piede e il mondo del lavoro nel settore dell'audio per videogiochi si arricchirà di tutta una serie di nuove figure professionali.

Proprio come negli ultimi 20 anni sono nate specializzazioni di ogni tipo nel mondo della **computer grafica** (professionisti che si occupano esclusivamente di **_rigging_**, **_textures_**, **_animazione_**, **_modellazione_**, **_light_**, **_visual fxs_**, **_compositing_** etc...), così anche nel mondo del sound design nasceranno nuove figure speciallizzate nella modellazione di suoni e fenomeni fisici differenti (**acqua** e [**bolle**], **fuoco**, [**fracture sound**], **impatti**, [**frizioni e sfregamenti**], [**accartocciamenti**], [**acustica delle stanze**], etc...).

Animation driven by audio

C'è addirittura chi si specializza nel processo inverso, ovvero nel ricreare animazioni basandosi sul suono in una sorta di [**inverse fooley**], il che può portare a risultati davvero sorprendenti:

Se, come spesso accade, l'audio è considerato come accessorio e secondario rispetto alla parte grafica/visiva, ci sono tuttavia alcuni casi in cui questo andamento è ribaltato: `event --> sound` diviene qui `sound --> event`.

Automatic lip sync: un sistema studiato da moltissimo tempo per animare grafiche e modelli 3D in modo automatico basandosi sul segnale audio (vedi [patent **Sierra**].

Procedural animation: [**Tom Clancy's Ghost Recon Advanced Warfighter 2**] (Ubisoft 2007) case study: un aeroplano precipitato nel deserto esplode e continua a bruciare a terra. Le **fiamme** sono sferzate a destra e a sinistra da un vento irregolare. Polvere e fumo sono generati proceduralmetne in base al livello dell'audio pre-prodotto (vedi [questo talk] di Scott Selfon al minuto 23:14)!