UNIVERSITÁ DEGLI STUDI DI MILANO
Facoltà di Scienze e Tecnologie
Corso di Laurea in Informatica Musicale

COMPOSIZIONE ADATTIVA DI COLONNE SONORE PER VIDEOGIOCHI

Relatore: Prof. Luca Andrea Ludovico

Correlatore: Dott. Giorgio Presti

Autore:

Simone Varenna

Matricola: 810850

A.A. 2017/2018

Indice

1 Abstract

In questo elaborato verrà introdotto il concetto di colonna sonora adattiva e verranno analizzate le problematiche legate alla sua natura e al medium interattivo a cui è, spesso, associata: il videogioco. In secondo luogo verrà illustrata la creazione e implementazione di una colonna sonora adattiva per un videogioco originale, seguita da un test finale per dimostrarne l'efficacia.

La colonna sonora adattiva in un videogioco deve trasmettere emozioni sempre diverse evolvendosi e trasformandosi a seconda degli eventi che vengono mostrati sullo schermo. Non deve interrompersi ad ogni cambiamento e deve assomigliare il più possibile a un brano lineare che cambia la propria struttura ad ogni iterazione, in base a come il giocatore affronta il gioco.

Verranno illustrati alcuni approcci per immergere l'utente nel gioco tramite la colonna sonora e, successivamente, alcune principali tecniche per la costruzione di un sistema di musica adattiva, in grado di rispecchiare gli eventi di gioco mostrati spostando dinamicamente la sfera emotiva del giocatore.
Queste nozioni sono state applicate, poi, per la creazione di una colonna sonora di un videogioco, sviluppato con un team indipendente, utilizzando Logic Pro 9 come Digital Audio Workstation per la composizione della musica e il software middleware FMOD Studio per definirne i comportamenti real-time. Il sistema è stato poi implementato tramite Unreal Engine 4, motore grafico utilizzato per lo sviluppo del videogioco. Successivamente il gioco è stato fatto testare a due gruppi di persone in due versioni differenti, con e senza il sistema musicale sviluppato, mostrando l'efficacia del lavoro svolto.
Come conclusione sono stati analizzati i risultati dei test dimostrando l'importanza della colonna sonora adattiva nel videogioco e discutendo delle eventuali lacune rilevate nel progetto svolto.

2 Stato dell'arte

2.1 Audio lineare e audio non-lineare

La musica è diventata uno strumento importantissimo nei media digitali, in particolare nei film viene usata per enfatizzare e sottolineare l'azione su schermo o per trasportare una scena in una particolare sfera emotiva o, ancora, per caratterizzare personaggi e luoghi andando a creare un legame con lo spettatore. Ogni emozione che si vuole esprimere nella scena di un film può essere facilmente amplificata con l'aiuto della musica, giocando con l'aspettativa e le sensazioni degli spettatori. La musica per i videogiochi ha sempre tentato di assumere lo stesso ruolo che ha la colonna sonora per un film, ma la differenza tra questi media digitali ha fatto si che gli approcci alle composizioni del comparto sonoro si sviluppassero in maniera totalmente differente.
Ciò che rende la musica e l'audio nei film uno strumento così potente, è il fatto che si possa controllare con estrema precisione la sincronizzazione con ciò che viene mostrato. Di ogni singolo movimento, dialogo, azione può essere accentuato l'impatto emotivo con un cambiamento consistente nella composizione musicale, più o meno inaspettato.

Da questa riflessione nasce la consapevolezza della principale differenza tra la musica in un film e quella in un videogioco: la sincronizzazione con gli eventi a schermo.
In un videogioco sarebbe l'ideale poter sincronizzare la musica ad ogni evento con la stessa precisione che si trova nella musica in un film. Il problema è che, essendo il videogioco un medium interattivo, non sapremo mai quando un particolare evento accadrà dopo l'avvio del gioco, né se quell'evento è importante negli interessi dell'emozione che si vuole esprimere.
Al contrario, quando un compositore si cimenta nella composizione dello score lineare di un film, si troverà a lavorare su una scena editata e di durata fissa: saprà esattamente quali sono gli eventi da considerare, la durata degli stessi, l'esatto momento in cui accadranno e in che ordine si verificheranno dopo l'avvio della pellicola.

Nell'ambito dei videogiochi l'obiettivo diventa quello di creare una colonna sonora che accompagni scene non-lineari con una musica altrettanto non-lineare. Si utilizza il codice e i parametri del gioco per tener conto degli eventi e lanciare risposte appropriate nella soundtrack, cercando di considerare anche il contesto musicale corrente ed evitando transizioni inappropriate e anti-musicali. Ormai, dal momento che i videogiochi attuali hanno una profondità ricercata che ha poco da invidiare ai film hollywoodiani, nasce l'esigenza di avere un sottofondo musicale dinamico che si possa adattare ad ogni scena e ad ogni evento di gioco. Da qui il nome “musica adattiva”.

2.2 La musica adattiva

Guy Whitmore - “Adaptive music isn't important for videogames, it is mandatory” [S-3]

Comporre numerosi “loop” musicali e riprodurre di volta in volta quello più adatto alla situazione, senza curarsi della performance, non rispecchia ciò che vuole essere un sistema di musica adattiva. Questo accadeva in molti giochi, come nei primi *Super Mario Bros*, ma i tempi cambiano, i videogiochi si evolvono e con essi cambiano anche le esigenze in ambito musicale.

Considerando che nei videogiochi troviamo diverse funzioni da attribuire alla musica, che variano da piccoli cambi di stato a temi di lunga durata, è necessario un sistema musicale più complesso per coprire tutti gli stati e le situazioni del gioco[B-1]. Alcune delle funzioni della musica che troviamo nei videogiochi possono essere:

- Impostare la scena: i giocatori hanno bisogno di indizi su dove si trovano quando entrano in un ambiente virtuale, la musica può aiutare a definire luogo e periodo.
- Introdurre personaggi: dei motivi possono aiutare a caratterizzare i personaggi e aiutare a sviluppare la percezione o a creare una connessione emotiva con ogni personaggio.
- Segnalare un cambiamento nello stato di gioco: ad esempio un gioco come *Red Dead Redemption* distingue gli stati di esplorazione, cavalcata e combattimento.
- Incrementare o decrementare la tensione: diverse tecniche sono usate per aumentare la tensione, ad esempio in *Asteroids* e *Space Invaders* aumenta il tempo della composizione per aumentare la tensione alla fine di ogni livello.
- Comunicare un evento al giocatore o influenzare le sue scelte: la musica in questi casi agisce introducendo come dei punti esclamativi sonori che permettono di confermare al giocatore un particolare avvenimento, come la fine di un combattimento o l'avvistamento di un nuovo nemico.
- Connettere emozionalmente il giocatore e il gioco: sviluppare un tema iconico per il gioco può servire a stabilire un particolare tono e mood del proprio gioco permettendo di infondere eccitazione o agitazione quando il giocatore lo sente.
- Enfatizzare la narrativa e la storia: in modo molto simile ai film, solitamente i giochi hanno uno strato narrativo, che la musica può enfatizzare nei suoi punti più elevati o più bassi.

Avendo preso atto, quindi, della vastità delle funzioni che può ricoprire la musica in un videogioco e avendo compreso che non è abbastanza comporre un' ottimo brano se poi non soddisfa i requisiti degli eventi di gioco, dare una definizione più precisa di ciò che si intende con “musica adattiva” aiuta ad avere una prospettiva più fresca e chiara dell'argomento:

“La musica adattiva deve incorporare un sistema per generare delle performance della stessa composizione significativamente diverse le une dalle altre, in risposta a dei parametri di input pre-determinati, senza conoscerne l'esatta sequenza, quantità, presenza o valori, mentre produce un output che deve essere coerente e soddisfacente all'interno della tradizione musicale selezionata dal compositore.” [S-1]

Se ascoltiamo orchestre diverse suonare una brano lineare, ad esempio, di musica classica, ognuna di queste performance avrà un'interpretazione differente, ma la forma e la struttura del pezzo rimarrà la stessa, mantenendo l'intenzione iniziale del compositore. La musica adattiva, invece, è significativamente differente da performance a performance per definizione, in quanto l'intenzione della composizione è avere una forma e una struttura essenzialmente flessibile. Espressioni individuali della stessa composizione di musica adattiva possono essere molto diverse tra loro, ma ognuna di esse sarà rappresentativa dell'intenzione originale della composizione.

Il sistema di musica adattiva è guidato da parametri di input che corrispondono a specifici eventi nel gioco a cui si interfaccia la musica, generando variazioni sostanziali nella performance musicale. Il sistema considera ognuno di questi parametri e decide come comportarsi al verificarsi di questi.

Non conoscendo l'esatta sequenza, quantità, presenza o valori di questi parametri si può dire che la forma finale della performance viene generata in risposta a eventi indeterminati. Il sistema è progettato per gestire specifici parametri, quindi in risposta all'input degli stessi parametri viene generato lo stesso identico output, rendendo il sistema di musica adattiva deterministico.

Ogni performance generata dal sistema deve suonare come se fosse una composizione lineare. Uno degli obiettivi principali per la musica adattiva è quello di funzionare esteticamente come un brano musicale in ogni sua istanza. La casualità deve avere delle regole, in quanto il caos completo non è parte della tradizione musicale. Inoltre bisogna tener conto della fatica d'ascolto, provocata da una componente sonora troppo ripetitiva e invadente da risultare fastidiosa a lungo andare, e cercare di far in modo che il sistema adattivo sia in grado di evitare il fenomeno e risultare sempre piacevole e pertinente.

2.3 Immersione

Ciò che rende un videogioco differente da altre forme d'arte e d'intrattenimento è essenzialmente la sua interattività. "Immersione" è un termine che viene usato molto comunemente nel discutere di videogiochi e di esperienze di gioco. È generalmente inteso come un elemento positivo dell'esperienza interattiva videoludica, ma spesso viene usato in modo vago e senza specificare a quali esperienze o fenomeni si riferisce.

Laura Ermi e Frans Mayra [B-3] definiscono l'immersione come "diventare fisicamente o virtualmente parte dell'esperienza" e identificano ciò che secondo loro sono i tre fattori principali che portano all'immersione in un videogioco: il fattore sensoriale (*sensory*), il fattore della sfida (*challenge-based*) e il fattore dell'immaginazione (*imaginative*). Il modello SCI (*sensory*, *challenge-based*, *imaginative*) sviluppato da Ermi e Mayra, offre un'analisi multidimensionale dell'immersione ben diversa delle teorie più vaghe e generiche sul coinvolgimento del giocatore in un videogioco.
Il primo dei tre fattori (*sensory*) riguarda la qualità audiovisiva e lo stile dell'opera videoludica. È la prima dimensione dell'esperienza videoludica, in quanto, una grafica attraente rende il gioco più piacevole e anche le persone con meno esperienza nei videogiochi possono percepirlo: lentamente il giocatore perde le informazioni sensoriali provenienti dal mondo reale e diventa totalmente concentrato nel mondo di gioco e nei suoi stimoli.
Il fattore della sfida (*challenge-based*) risulta importante in quanto il piacere derivato dal giocare è fortemente influenzato dalla sensazione di successo e l'insicurezza dell'esito finale ricopre un ruolo importantissimo nella suspense di gioco. Infatti, gli sviluppatori di videogiochi curano minuziosamente il bilanciamento delle difficoltà di un gioco per infondere una sensazione di realizzazione nei giocatori, evitando di rendere il gioco troppo facile o troppo difficile.
Il fattore dell'immaginazione (*imaginative*) è ciò che porta il giocatore ad essere assorbito nella storia o nel mondo di gioco e iniziare ad indentificarsi in uno o più personaggi. Il verificarsi di questo fenomeno è frequente anche se non si tratta di giochi di ruolo o story-driven, dove la narrazione e la profondità dei personaggi è una delle componenti principali.

Gordon Calleja [B-4] propone un nuovo modello per definire l'immersione del giocatore affermando che esso si identifica nell'unione di una varietà di fenomeni empirici - più specifici rispetto al modello SCI - procurati da un gameplay coinvolgente. Calleja distingue sei dimensioni di coinvolgimento:

- La dimensione cinestetica (*kinesthetic*) che riguarda la libertà d'azione concessa al giocatore e la difficoltà della curva di apprendimento dei controlli.
- La dimensione spaziale (*spatial*) che riguarda l'interazione del giocatore con la navigazione e l'esplorazione dell'ambiente virtuale.
- La dimensione condivisa (*shared*) che riguarda la consapevolezza del giocatore di poter interagire con altri attori (umani e non) nell'ambiente di gioco, in termini di co-abitazione, cooperazione e competizione.
- La dimensione narrativa (*narrative*) che si riferisce al coinvolgimento verso gli elementi narrativi che sono stati scritti nel gioco, così come quelli che emergono dall'interazione del giocatore con il videogioco.
- La dimensione affettiva (*affective*) che comprende varie forme di coinvolgimento emozionale, dalla sensazione di serenità nell'oltrepassare un

paesaggio esteticamente piacevole alla scarica di adrenalina di un first-person-shooter competitivo online.
- La dimensione ludica (*ludic*) che esprime l'interazione del giocatore con le scelte prese nel gioco e le ripercussioni di tali scelte.

È chiaro che la musica dovrebbe ricoprire un ruolo importante in questi modelli, eppure viene data pochissima attenzione al comparto sonoro di un videogioco nei riguardi dell'immersione del giocatore: nel modello SCI rientra nel fattore sensoriale, ma la musica viene appena citata dalle autrici, mentre nel modello di Calleja rientra indirettamente nella dimensione affettiva.

Isabella van Elferen [B-2] delinea un nuovo modello analitico, l'ALI model, che integra quelli presentati precedentemente e che considera l'immersione nei videogiochi data esclusivamente dalla musica.
Questo modello consiste in tre fenomeni empirici: affetto musicale (*musical affect*), alfabetismo musicale (*musical literacy*) e interazione musicale (*musical interaction*).

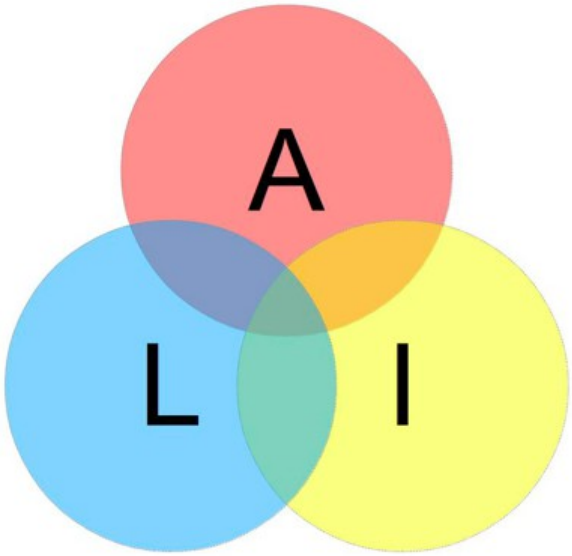

Figura 1: Rappresentazione grafica del modello ALI

- L'affetto (*affect*) è un aspetto vitale e inevitabile di ogni esperienza musicale: ascoltare musica non può che far provare emozioni. Ogni volta che sentiamo musica, anche senza prestare attenzione, il nostro stato d'animo è influenzato dalla connotazione musicale. Così avviene nei film e nei videogiochi, nei quali siamo influenzati, senza accorgercene, dalla musica che accompagna ciò che vediamo e gli eventi interattivi. Senza musica, i videogiocatori esprimono una mancanza di coinvolgimento anche in scene drammatiche che implicano morte o sofferenza. Siccome le emozioni collegate alla musica non sono assolutamente oggettive e universali, l'immersione affettiva musicale sembrerebbe altamente soggettiva e quindi imprevedibile, cosa assolutamente inaccettabile: la musica nei videogiochi deve necessariamente evocare emozioni prevedibili così che possa aiutare a definire situazioni di gioco (una scena pericolosa deve essere riconosciuta come tale, quindi deve essere sottolineata da una musica spaventosa). Per assicurarsi che l'affetto musicale porti a emozioni prevedibili il modello ALI include l'alfabetismo musicale (*literacy*).

- L'alfabetismo (*literacy*) musicale è il sentire la musica dei film, tv e pubblicità attraverso la propria esperienza (plasmata dalla continua esposizione a questi media) ed essere in grado di interpretare facilmente ciò che vuole comunicare. L'esistenza di numerosi standard e convenzioni aiuta il pubblico a interpretare correttamente gli intenti di quello che gli viene presentato: difficilmente, sentendo degli accordi bassi e dissonanti di un violoncello, uno spettatore che guarda un film horror non si aspetterebbe una catastrofe incombente. Attraverso i riferimenti alle convenzioni audio visive di altri media, le soundtrack nei videogiochi si affidano all'alfabetismo del giocatore per creare l'effetto di immersione: i giocatori riconoscono certi stili compositivi ed è per questo che riescono ad interpretare correttamente gli eventi di gioco e si sentono coinvolti negli ambienti, nella narrazione e nel gameplay. Le boss-fight sono spesso accompagnate da composizioni caratterizzate da un tempo sostenuto, orchestre dissonanti e percussioni sincopate che i giocatori riconoscono da scene esistenti di film d'azione eroici.
  Bisogna considerare che la composizione di colonne sonore per videogiochi non dipende più solo da alfabetismi provenienti da altri ambienti non videoludici, ma che ha sviluppato il proprio alfabetismo videogame-related. La musica nei videogiochi non è spesso originale o complicata e per un motivo preciso: la *literacy* nell'interpretazione della musica nei videogiochi deve essere facilmente acquisibile e immediatamente riconoscibile, in quanto la musica ha un ruolo cruciale nell'interazione col gioco.

- L'interazione (*interaction*) musicale nei videogiochi stabilisce una connessione diretta tra le azioni del giocatore e la soundtrack. Questo fenomeno corrisponde esattamente con la definizione di musica adattiva citata precedentemente, rendendo i sistemi adattivi uno dei punti principali del modello di analisi dell'immersione ALI.

Il fenomeno dell'affetto (*affect*) nel modello di Isabella van Elferen ha come risultato l'attribuzione di un particolare "mood" alla musica. La nozione di "mood", però, è sempre stata molto controversa, in quanto la musica è una struttura molto complessa di regole e suoni ed è difficile provocare un'emozione precisa nell'ascoltatore.

Oltre all'utilizzo del fenomeno di alfabetismo (*literacy*), previsto dal modello ALI, si può determinarne il mood di una musica utilizzando un piano multidimensionale, formato dai parametri di *arousal* e *valence*. [B-7]
Il primo parametro (*arousal*) riguarda uno stato soggettivo della persona che va da calmo-rilassato fino ad eccitato-stimolato. La valenza (*valence*) invece, viene identificata con uno stato emotivo di positività o negatività nell'ascoltatore.

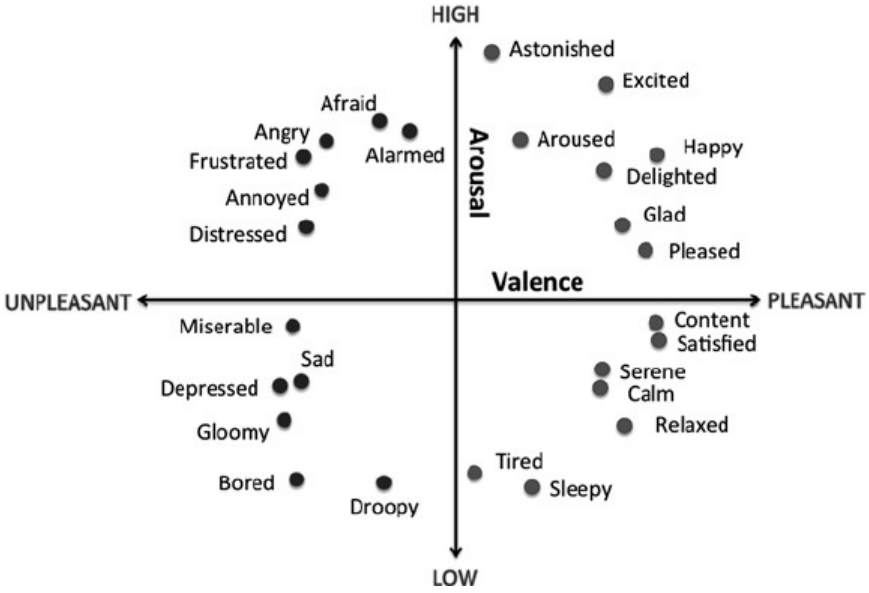

Figura 2: Il piano valenza/arousal delle emozioni

Essendo la musica una struttura complessa di regole e suoni, diversi studi [B-5][B-6] hanno tentato di identificare gli elementi musicali che, all'interno di un brano, vanno ad incidere sull'arousal e sulla valenza, permettendo di muovere le emozioni dell'ascoltatore a seconda delle situazioni.

Alcuni dei fattori musicali che vanno a incidere sull'arousal riguardano la parte ritmica di un brano, infatti il tempo della musica e l'intensità della sezione ritmica sono ciò che permette di manipolare lo stato di eccitazione nelle persone. Aumentando o diminuendo l'intensità e il tempo è possibile indurre uno stato di serenità o agitazione nell'ascoltatore.
Inoltre l'arousal può essere alterato manipolando lo spettro armonico di un brano, che spesso è legato alla strumentazione. Ad esempio, una composizione suonata al pianoforte rispetto alla stessa composizione eseguita da un'orchestra avrà sempre un impatto molto meno eccitante. Più complesso è lo spettro armonico della musica, più elevato sarà l'arousal.

La valenza, invece, può essere considerata come il riconoscimento di una melodia piacevole contro una melodia non piacevole. Un brano di musica tonale sarà sempre riconosciuto come più piacevole rispetto a un brano atonale, infondendo nell'ascoltatore, rispettivamente, emozioni di positività o negatività.

2.4 Principali tecniche di musica adattiva

I loop musicali seamless sono uno dei primi elementi costituivi nella creazione di musica per i videogiochi. Tipicamente questi loop costituiscono dei pezzi musicali dei quali non si percepisce l'inizio e la fine, quindi la musica potrebbe continuare ad essere riprodotta all'infinito senza interruzioni. È una delle prime tecniche sviluppate per la musica nei videogiochi essendo semplice ma molto efficace, in quanto, ad esempio, non conoscendo a priori in quanto tempo un giocatore finirà un livello, il loop non lascerà mai il gioco senza musica.
L'utilizzo dei loop introduce, però, uno dei principali problemi nell'audio per videogames: la fatica d'ascolto. Il cervello umano è efficientissimo nel riconoscimento di pattern regolari e a lungo andare percepirebbe la ripetitività dei loop inducendo una sensazione poco piacevole al giocatore. Creare dei loop abbastanza lunghi e con una struttura che comprende parti diversificate durante tutta la sua durata non risolverebbe del tutto questo grosso problema, ma quanto meno potrebbe renderlo molto meno percettibile.
Tipicamente, nei giochi che utilizzano esclusivamente questa tecnica, vengono creati diversi loop musicali, differenti tra loro, per caratterizzare, ad esempio, diversi luoghi o diverse aree. Un esempio potrebbe essere *Super Mario* che presenta diversi loop musicali per livelli all'aperto, livelli acquatici, sotto terra o nel castello.

I loop musicali creati per un videogioco possono ricadere in due categorie: loop non-sincronizzati e loop sincronizzati. In base alla categoria di appartenenza si può comporre la colonna sonora utilizzando le tecniche "*horizontal resequencing*" o "*vertical remixing*".

2.4.1 Horizontal resequencing

Invece di sfruttare i silenzi (come i caricamenti di un gioco tra un livello e un altro) per effettuare il passaggio tra loop non-sincronizzati, oppure lasciare che il codice cambi in modo brusco tra un loop e l'altro, si possono utilizzare tecniche adattive di *horizontal resequencing*.

Il *cross-fading* è la più semplice delle tecniche di *horizontal resequencing* e consiste nell'abbassare gradualmente il volume di un loop e contemporaneamente alzare quello di un secondo loop, creando una transizione morbida e piacevole.

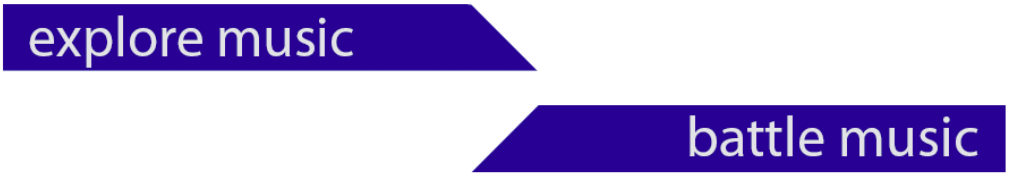

Figura 3: Rappresentazione grafica di cross-fading

Uno dei vantaggi principali è che l'implementazione di questa tecnica è molto semplice, il compositore potrà, dunque, passare più tempo a comporre un ottimo brano musicale piuttosto che pensare a come implementare la musica nel gioco. Inoltre è possibile una reazione pressoché immediata da parte del sistema di musica adattiva nel momento in cui avviene l'evento di gioco scatenante, riuscendo a sincronizzare in modo impeccabile i cambiamenti della musica con l'azione del giocatore.

Il problema principale di questa tecnica è che al nostro orecchio non risulta sempre musicale: spesso non si tiene conto di tempo e armonia quando si passa da un pezzo all'altro ed è altamente probabile che le frasi musicali all'interno di un loop vengano interrotte durante il *cross-fade*. I cambiamenti tramite *cross-fade* potrebbero essere avvertiti un po' come cambiare le stazioni di un radio, rischiando di spiazzare il giocatore.

Si possono utilizzare diverse tecniche di *horizontal resequencing* per creare delle transizioni più musicali ed evitare comportamenti imprevisti nella colonna sonora.
La prima è il *musical demarcation branching* che consiste nell'effettuare il cross-fade tra due loop in prossimità di un punto di demarcazione musicale, come una misura o un quarto all'interno della battuta. Questa tecnica permette di evitare comportamenti inaspettati dal sistema adattivo, facendo in modo che le transizioni accadano rispettando la griglia temporale della musica.

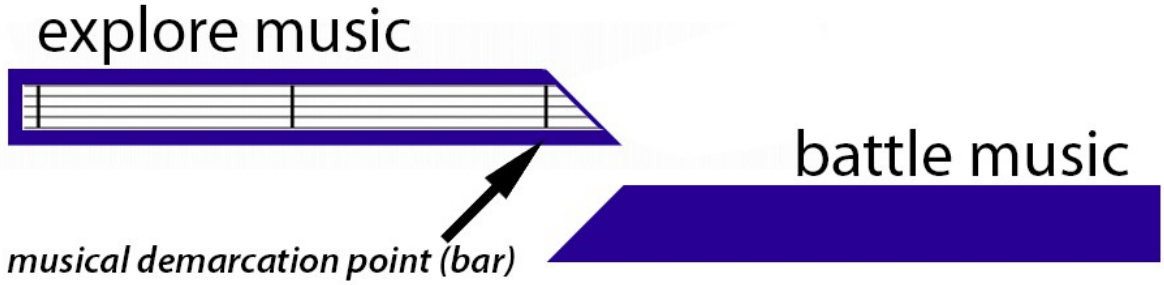

Figura 4: Rappresentazione grafica di musical demarcation branching

Anche se risolve il problema della musicalità nei cambiamenti della musica, non viene risolto il problema dell'interruzione di frasi musicali, in quanto queste non sono per forza ancorate alla mappa temporale del brano. Inoltre, introduce un ulteriore problema: i cambiamenti non potranno più essere immediati, ma accadranno solo una volta raggiunto il punto di demarcazione musicale deciso precedentemente.

Un'altra tecnica orizzontale è il *phrase branching*. Come la precedente, rende i cambiamenti molto musicali, risolvendo anche il problema dell'interruzione di frasi musicali. Infatti questa tecnica consiste nel ritardare le transizioni tra i loop alla conclusione di ogni frase, rendendo il sistema adattivo ancora più musicale.

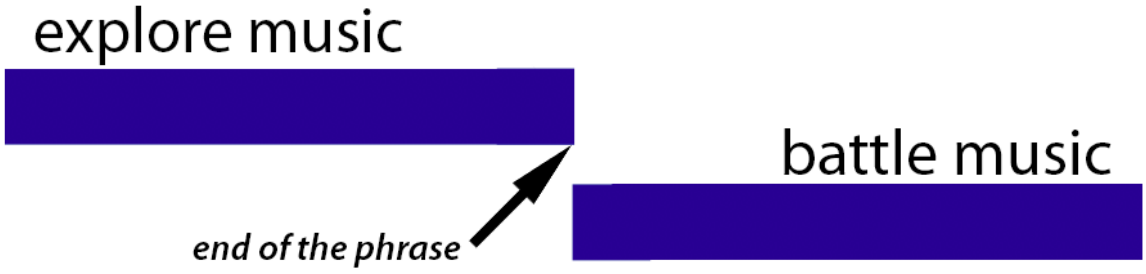

Figura 5: Rappresentazione grafica di phrase branching

Viene amplificato però il problema dell'immediatezza del cambiamento e la sincronizzazione con l'azione del giocatore: se l'evento scatenante avviene appena dopo l'inizio di una frase musicale, potrebbe volerci un tempo troppo esteso per cambiare lo stato della musica, rischiando di confondere il giocatore.

Un gioco che fa largo uso di questa tecnica è *Monkey Island 2: LeChuck's Revenge* (1991) [S-5] della LucasArts, primo ad utilizzare il sistema musicale iMuse [S-4] sviluppato dalla stessa casa videoludica. Trattandosi di una avventura grafica punta-e-clicca non ha bisogno di un sistema musicale con una reattività molto elevata, ma, al contrario, un sistema che punti alla continuità e alla coerenza musicale senza cambiamenti bruschi e troppo evidenti. L'iMuse riusciva perfettamente in questo intento creando la colonna sonora di quella che è considerata una delle più iconiche saghe di giochi d'avventura punta-e-clicca.

Infine è possibile utilizzare delle transizioni: brevi estratti musicali che hanno la funzione di collegare due loop differenti mascherando l'eventuale stacco brusco tra i due segmenti musicali e creando ciò che, verosimilmente, scriverebbe un compositore per un passaggio in un brano lineare.

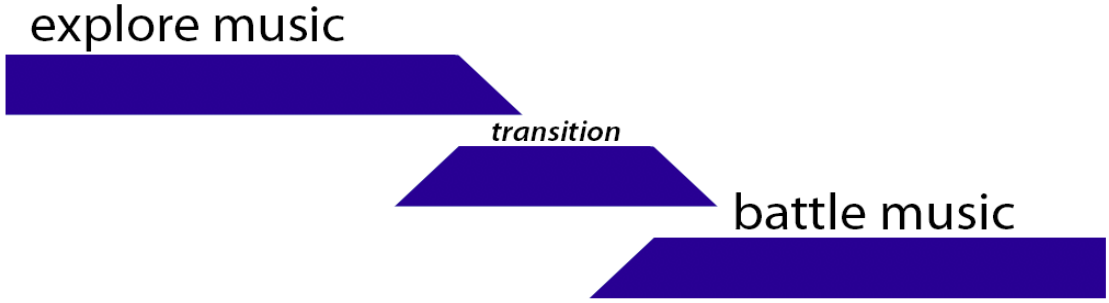

Figura 6: Rappresentazione grafica di una transizione

Con questa tecnica è possibile passare tra due loop con framework armonico e tempo completamente differente in modo molto naturale, tutto dipende da come vengono composti i loop e le transizioni.
Se queste transizioni avvengono a distanza di un tempo molto breve è bene che il compositore scriva un numero elevato di transizioni differenti, in modo da evitare ripetitività e fatica di ascolto nel giocatore. Nonostante ciò, alcuni giochi fanno leva su questa ripetitività rendendo iconiche alcune delle loro transizioni musicali: nella serie *Metal Gear Solid*, la trasizione musicale riprodotta quando il protagonista viene scoperto dai nemici è diventato un suono rappresentativo dell'intera saga.

2.4.2 Vertical Remixing

Nel caso in cui i loop musicali siano sincronizzati si può usare la tecnica adattiva di *vertical remixing*. Essa consiste nell'usare dei loop sincronizzati rappresentandoli come i layer di un brano musicale, (ad esempio, diversi strumenti o "sezioni", come percussioni, archi, bassi ecc..) e, a seconda della situazione, aggiungerli o sottrarli al mix per creare diversi livelli di intensità o emozione.
Ad esempio, avendo tre layer corrispondenti a droni ambientali, percussioni ed elementi melodici è possibile arrangiarli per creare un sottofondo musicale per accompagnare un gameplay action: mentre il giocatore esplora il mondo di gioco viene riprodotto solo il layer di droni. Nel momento in cui si avvicina ad un pericolo vengono inserite nel mix le percussioni e quando inizierà ad affrontare un nemico verrà aggiunto anche il layer della melodia. Alla fine dello scontro il giocatore uscirà dall'area di battaglia e le percussioni e la melodia verranno sottratti dal mix lasciando solo i droni ambientali, segnale che la battaglia è finita e il giocatore può tornare all'esplorazione.

Figura 7: Rappresentazione grafica di vertical remixing

Questa tecnica è utilizzata in quasi tutti i giochi più popolari, più o meno recenti, in quanto è abbastanza semplice da implementare, permette cambiamenti quasi immediati e in più mantiene una continuità musicale tra i vari stati di gioco: si tratta, infatti, di aggiungere e sottrarre elementi alla stessa composizione, mantenendo, quindi, la stessa struttura armonica e lo stesso tempo.
Uno dei principali svantaggi di questa tecnica è quello di riprodurre essenzialmente un loop musicale diviso in diversi layer sovrapposti. Quindi indurrà, a lungo andare, al riconoscimento dell'inizio e la fine della composizione, provocando fatica di ascolto nel giocatore che ne percepisce la ripetitività. Inoltre può risultare poco piacevole sia perché alcune frasi musicali possono essere interrotte, sia perché, solitamente, in una composizione lineare si utilizzano dei crescendo appropriati per aggiungere o sottrarre elementi al brano. Infine, con questa tecnica, non è possibile apportare cambiamenti nel framework armonico della colonna sonora e nemmeno nel tempo, e le variazioni tramite *vertical remixing* possono essere talmente sottili da non essere percepiti dall'ascoltatore, nel caso in cui un layer non contenga un contenuto musicale rilevante.

2.4.3 Stinger-based

Un'ulteriore tecnica di composizione adattiva è quella di basare la colonna sonora su degli stinger. Questi consistono in una serie di corte idee musicali che vengono riprodotte quando viene scatenato l'evento di gioco corrispondente e possono sottolineare la fine di una partita, la scoperta di un segreto, la sconfitta di un boss e moltissimi altri eventi.
Gli stinger possono essere riprodotti all'inizio di uno specifico beat o misura sovrapponendosi alla musica di sottofondo. In questo modo lo stinger calzerà perfettamente la struttura temporale della colonna sonora, con lo svantaggio di venir riprodotto leggermente in ritardo rispetto all'evento su schermo.
Possono essere, quindi, riprodotti anche immediatamente senza considerare il tempo musicale, rimanendo perfettamente connessi agli eventi di gioco, ma rischiando di essere poco calzanti con la musica di sottofondo.
Alcuni segmenti di giochi come *Tomb Rider (2013)* o *Uncharted* [S-6] basano l'intera colonna sonora sugli stinger, che costituiscono dei crescendo non vincolati da un tempo preciso e senza essere sovrapposti ad una musica di sfondo.

Tipicamente l'approccio migliore è quello di utilizzare una combinazione di tutte le tecniche sopracitate per creare un mix più sofisticato che può adattarsi in intensità, variazione e tematica ed è possibile utilizzare DSP (Digital Signal Processing) per creare una profondità maggiore, aggiungendo, ad esempio, filtri, equalizzatori o compressori che processano il segnale real-time. [S-7]

2.4.4 Musica Generativa

Tra la musica adattiva si può localizzare anche la musica generativa o procedurale. La musica, in fondo, è una struttura matematica con armonie e scale, ciò significa che può essere anche programmata. Strumenti, tempo, scala, struttura possono essere utilizzati come *seed* da dare in pasto ad un algoritmo di generazione randomico per creare delle complete musiche che si adattano al contesto. Questo approccio potrebbe generare musiche sempre diverse ma con risultati mai controllati, togliendo quella che potrebbe essere la parte artistica data da un compositore che scrive musiche appositamente studiate per ogni evento. Alcuni giochi, anche molto recenti, utilizzano un approccio ibrido, includendo musica registrata e musica procedurale che coabitano nella stessa composizione. Un esempio può essere quello di *Rise of the Tomb Rider (2015)* che utilizza il Dynamic Percussion System [S-2] che permette di creare dei groove di percussioni sempre diversi ma che si adattano perfettamente al gameplay.

3 Tecnologie

3.1 Game Engine: Unreal Engine 4

Unreal Engine 4 è diventato uno dei più popolari motori di gioco degli ultimi tempi. Un "game engine" è essenzialmente un ambiente di sviluppo composto da una suite di strumenti per la creazione di videogiochi, con la possibilità di portare un gioco su diverse piattaforme con minime modifiche al codice sorgente del gioco. Gli strumenti all'interno dell'engine sono tra i più disparati: dal motore di rendering grafico, al motore che gestisce la fisica, le luci e le ombre fino a un editor video in-game con rendering in real-time, utile per creare filmati e cut-scene all'interno del proprio gioco. Nella maggior parte dei motori di gioco si trova sempre anche un motore audio, capace di gestire tutte le logiche del comparto sonoro di un videogioco, dai sound effects alla colonna sonora. Unreal Engine 4 fornisce la possibilità di creare delle *Sound Cue* che possono essere mandate a delle *Sound Class*, creando un routing molto simile a quello di un mixer, raggruppando vari elementi sotto uno stesso canale. Le *Sound Cues* sono dei suoni composti, che permettono di modificare il comportamento al playback sonoro, combinare gli audio files e applicare modificatori (come modulatori di ampiezza e pitch, delay e altro) tramite un pratico sistema di visual scripting a nodi.

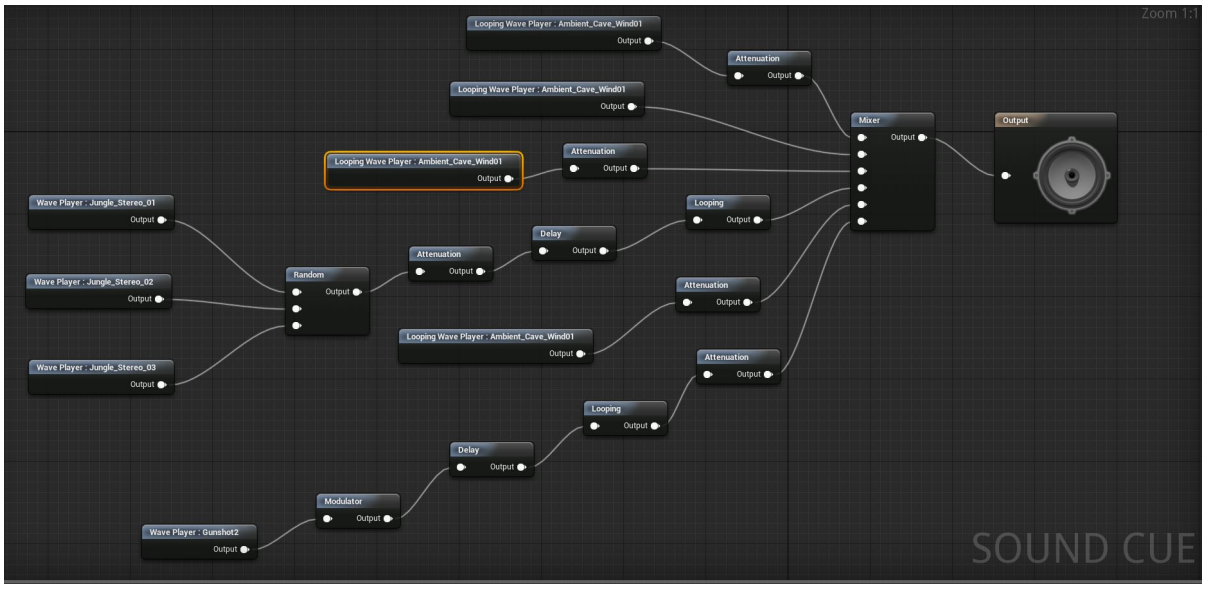

Figura 8: Una complessa Sound Cue che mischia diverse Sound Wave usando diverse proprietà quali attenuazione, randomizzazione, looping e delay.

Però, il motore audio di Unreal Engine 4, nonostante sia un tool molto potente, potrebbe risultare limitato e inadatto allo sviluppo di sistemi complessi. È molto comune, nell'industria videoludica, la scelta di utilizzare strumenti esterni più specializzati per aumentare la complessità e velocizzare il processo della creazione del sonoro in un videogioco. Software middleware come Audiokinetik Wwise o FMOD Studio sono alcuni degli strumenti più usati in campo videoludico che hanno l'obiettivo di creare complessi sistemi sonori permettendo al compositore o al sound designer di concentrarsi su ciò che vuole fare piuttosto che sul come poterlo fare.

3.2 Audio Middleware: FMOD Studio

FMOD Studio è uno dei principali audio middleware che vengono usati nell'industria videoludica nei nostri giorni. Si presenta con un'interfaccia molto familiare per chiunque abbia avuto a che fare con Digital Audio Workstation (DAW) e vari programmi di produzione musicale, rendendolo accessibile anche a sound designer o compositori che non sono necessariamente programmatori o professionisti specifici dell'ambiente videoludico.

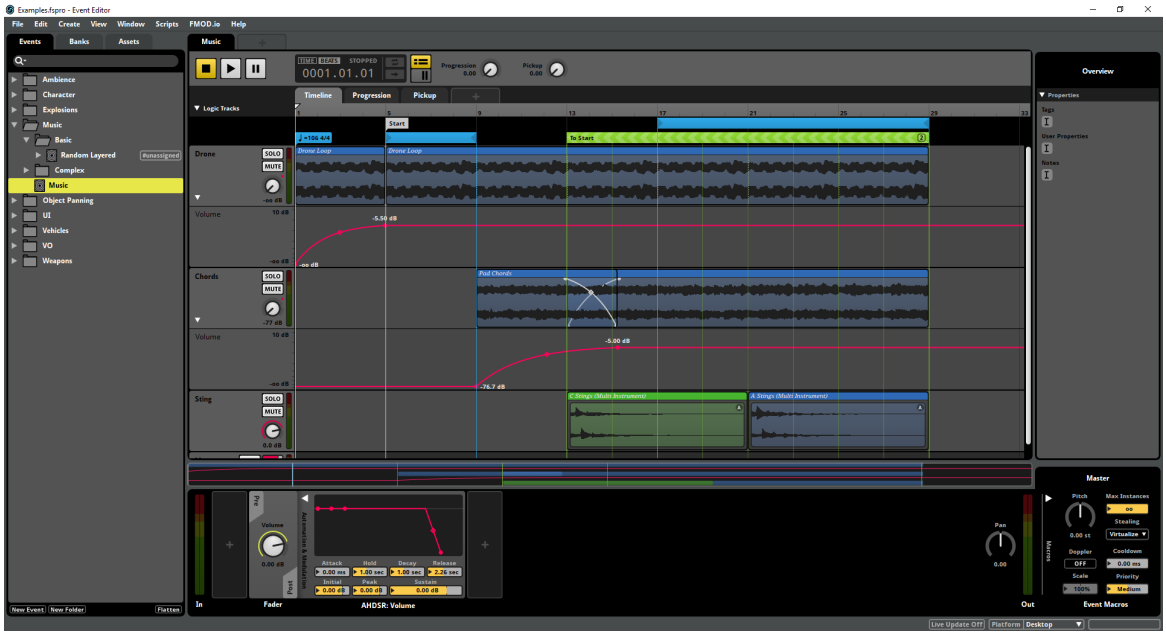

Figura 9: Interfaccia di FMOD Studio

FMOD Studio permette di creare degli *Events* che sono delle unità di contenuto sonoro istanziabili, che possono essere innescate, controllate o stoppate dal codice di gioco. All'interno degli *Events* è possibile creare delle tracce, che funzionano esattamente come le tracce in una DAW, a cui possono essere assegnati degli effetti che processeranno il segnale in real-time. Ogni segnale audio ha origine dagli *Instruments* che appaiono come delle regioni (trigger-regions), risiedenti in ogni traccia, e che possono contenere uno o più sound files.

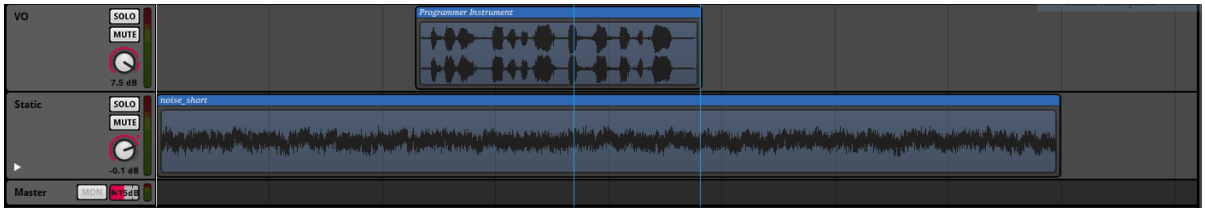

Figura 10: Tracce e *Instruments* all'interno di un *Event*

Quando, tramite il codice di gioco, si innesca un'*Event*, la playhead al suo interno inizia a percorrere la timeline. Ogni volta che la playhead della timeline passa sopra u n a trigger-region di un *Instrument*, innescherà gli audio file contenuti in quest'ultimo. Inoltre si ha la possibilità di definire la probabilità e le condizioni di riproduzione, la quantizzazione dell'innesco rispetto al tempo musicale, o le regole di polifonia di ogni *Instrument*.

I sound files rappresentano gli elementi costitutivi degli *Events* e degli *Instruments* e possono essere riprodotti in modo sincrono o asincrono rispetto alla playhead della timeline.

Una delle caratteristiche più importanti di questo software sono i parametri, grazie ai quali è possibile definire il comportamento degli *Event* mentre viene riprodotto ciò che sta al loro interno. I parametri sono delle proprietà di un *Event* che possono essere aggiornate in tempo reale tramite il codice di gioco e possono essere assegnati a praticamente qualsiasi cosa all'interno di un evento, per automatizzare i valori delle proprietà degli *Events* o per controllare la timeline attraverso dei marker logici.

Figura 11: Parametri in FMOD Studio

Ogni *Event* può essere mandato al project mixer per creare dei gruppi e di conseguenza al master bus, dando la possibilità di creare mix complessi come su una DAW.

Infine le *Sound Bank* sono dei contenitori di *Events* compressi che verrano usati dal motore di gioco. È possibile caricare *Sound Bank* dinamicamente scegliendo quali *Events* servono in determinati momenti del gioco permettendo di avere controllo sul consumo di memoria dell'audio di gioco.
Le *Sound Bank* compilate sono tutto ciò di cui avrà bisogno il game engine per fare uso degli *Events* che contengono e, quindi, interagire con le logiche sviluppate su FMOD Studio utilizzando il codice di gioco.

4 Progetto

Il videogioco per il quale è stato creato un sistema di musica adattiva è un riadattamento del classico gioco “Battleship” (Battaglia Navale) con ambientazione spaziale, sviluppato con un team indipendente.

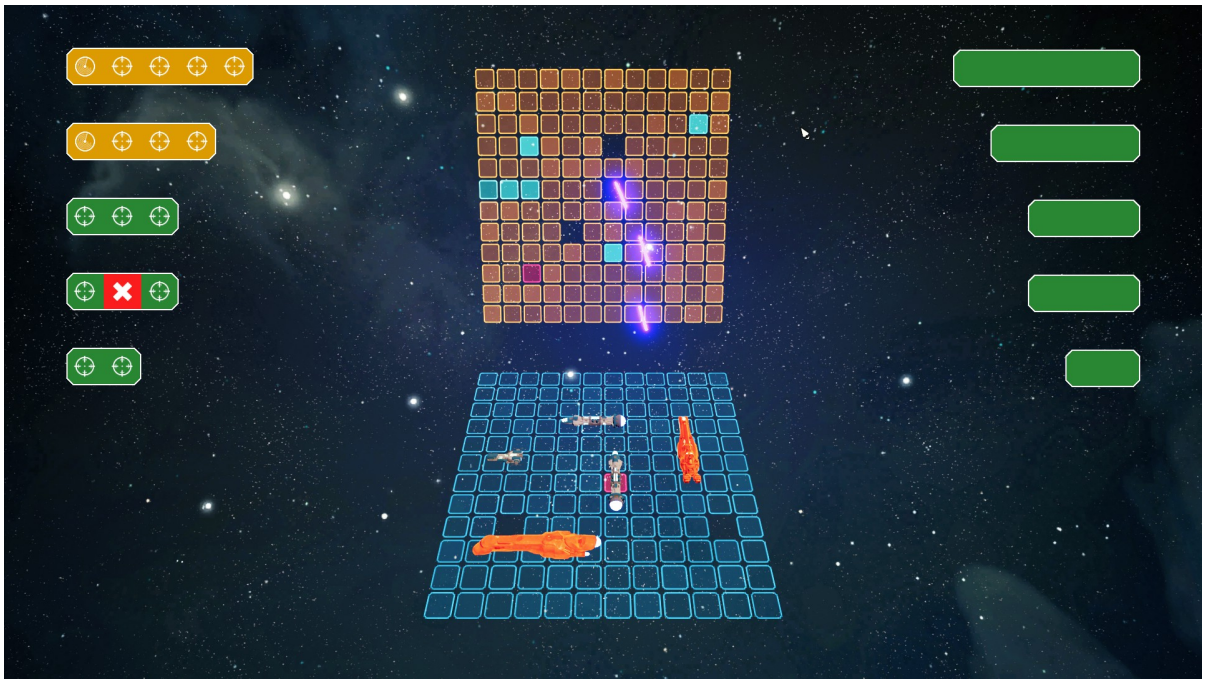

Figura 12: Schermata di gioco

Il gioco presenta solo la modalità giocatore singolo, nella quale l'utente sfiderà un'intelligenza artificiale (AI).
Una delle maggiori sfide che ha comportato questo progetto è stata quella di rendere avvincente e immersivo un gioco che, per sua natura, è molto semplice e statico. La grafica moderna e accattivante è un fattore che già aiuta molto a rendere il prodotto più appetibile, ma senza un sonoro all'altezza potrebbe, alla fine, risultare noioso.
Il gioco non differisce molto dalla battaglia navale tradizionale, consistendo in un giocatore umano e un'intelligenza artificiale che, a turno, dovranno selezionare le celle del campo di battaglia per colpire le navi avversarie nascoste. La differenza più importante rispetto al gioco tradizionale è quella di dover effettivamente utilizzare le navi della propria flotta per colpire quelle avversarie, avendo a disposizione più di un colpo per ogni turno, dipendentemente dalla nave selezionata. Inoltre, la potenza di fuoco della propria flotta diminuirà per ogni nave colpita o affondata dall'avversario.

Il flusso di gioco presenta delle animazioni della telecamera che inquadrano i campi di battaglia in cui si concentra l'azione, e, inoltre, delle animazioni che mostrano il fuoco delle navi e il percorso dei proiettili fino all'arrivo sul campo di battaglia nemico, con conseguenti effetti grafici e sonori dipendentemente dal fatto che i colpi siano andati a segno o meno. Queste animazioni sono importanti in quanto sono state prese in considerazione progettando la colonna sonora, che ha potuto usufruire dei tempi delle animazioni sia per aggiungere elementi musicali nuovi, sia per creare transizioni più piacevoli.

4.1 Progettazione soundtrack

La soundtrack creata si pone di rispettare il modello ALI di immersione [B-2], caratterizzando la musica di elementi che siano pertinenti con l'ambientazione spaziale del gioco e che trasmettano tensione o distensione reagendo allo svilupparsi della partita. L'ispirazione musicale parte da altre opere fantascientifiche, come il film *Bladerunner* o la space-opera videoludica *Mass Effect*, con l'intenzione di creare una colonna sonora ambient-elettronica con droni sintetizzati ed elementi orchestrali.
Questa musica ha il compito di immergere il giocatore in una battaglia tra due flotte di navi spaziali, muovere lo stato d'animo dell'utente, dipendentemente dallo svolgimento della partita, e, infine, sottolineare quegli eventi di gioco significativi che possono avere un impatto emotivo sul giocatore.

La composizione di questa colonna sonora si divide in due parti:

- Una base musicale riprodotta in loop che definisce il mood del gioco e che deve avere la possibilità di mutare il proprio stato (a seconda dei parametri di arousal e valenza).
- Degli elementi musicali che vanno ad aggiungersi alla base per sottolineare eventi di gioco specifici, utili a trasmettere emozioni immediate al giocatore.

4.1.1 Base musicale

I fenomeni di *affect* e *literacy* del modello ALI di Isabella van Elferen prevedono la trasmissione di emozioni riconoscibili dal giocatore e che rientrino in uno specifico mood. Nel caso di questa soundtrack, si è cercato di determinare un'atmosfera che diventasse sempre più intensa con il proseguire della partita alternando fasi di tensione con fasi di distensione, dipendentemente da quale contendente si trovi in vantaggio.

Si è deciso di dividere l'intensità emotiva del gioco in 3 livelli di arousal e di distinguere i vari stati di tensione con valenza “negativa”, “neutrale” e “positiva”.

Identifichiamo, quindi, 9 stati differenti in cui deve rientrare la musica (Tabella 1):

Arousal livello 1 Valenza negativa	Arousal livello 1 Valenza neutrale	Arousal livello 1 Valenza positiva
Arousal livello 2 Valenza negativa	Arousal livello 2 Valenza neutrale	Arousal livello 2 Valenza positiva
Arousal livello 3 Valenza negativa	Arousal livello 3 Valenza neutrale	Arousal livello 3 Valenza positiva

Tabella 1: Gli stati definiti dalla combinazione di arousal e valenza

Il fenomeno di interazione (*interaction*) del modello ALI viene rispettato sfruttando la somma delle navi operative in gioco per determinarne l'intensità emotiva: più sono le navi affondate, più sarà alto il livello di arousal. Questo serve a dare la sensazione di

una battaglia sempre più intensa con il culmine dell'eccitazione che sfocerà nella distruzione completa di una delle flotte e la vittoria di uno dei due contendenti.
Considerato che la somma massima delle navi in gioco è 10 (5 dell'utente e 5 della AI), dividendola per i 3 livelli di arousal si trova che ogni 3 navi affondate l'arousal dovrà salire di un livello (escludendo il caso in cui la somma delle navi sia 1, che rappresenta la fine della partita).

La tabella di seguito (Tabella 2) mostra i livelli di arousal per tutte le situazioni di gioco riguardanti la somma delle navi.

Navi Utente / Navi AI	5	4	3	2	1
5	10	9	8	7	6
4	9	8	7	6	5
3	8	7	6	5	4
2	7	6	5	4	3
1	6	5	4	3	2

Arousal livello 1 | Arousal livello 2 | Arousal livello 3 | Arousal livello 3*

**Secondo il ragionamento precedente queste situazioni dovrebbero essere di arousal livello 2, ma si è deciso di renderle di livello 3 in quanto uno dei due contendenti è rimasto con una sola nave operativa, di conseguenza l'eccitazione deve essere per forza molto elevata.*

Tabella 2: Livelli di arousal per ogni situazione di gioco

La valenza, invece, varia in base alla differenza fra le navi operative dell'utente e quelle dell'intelligenza artificiale. In caso di differenza nulla la valenza sarà "neutrale", mentre sarà "negativa" o "positiva" nel caso in cui, rispettivamente, l'utente sarà in svantaggio o in vantaggio (ovvero se ha meno o più navi operative rispetto all'intelligenza artificiale).

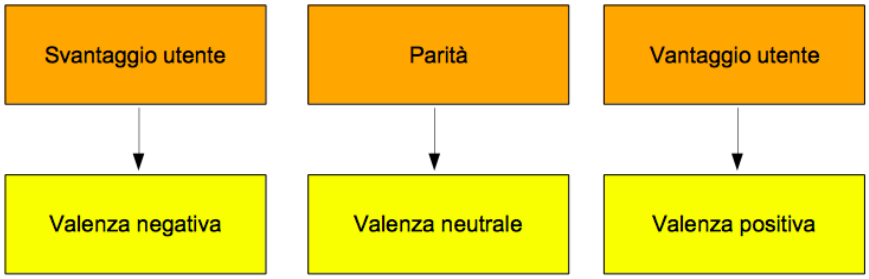

Tabella 3: Valenza calcolata in base al vantaggio o svantaggio dell'utente

Nella Tabella 2 le somme delle navi in bianco rappresentano le situazioni in cui si verifica valenza positiva. Tracciando una diagonale su queste si può trovare le situazioni di valenza negativa e positiva, rispettivamente, a destra e sinistra rispetto la diagonale. L'unica eccezione viene fatta quando entrambi i giocatori si ritrovano con una sola nave (somma rappresentata in grigio nella Tabella 2). In questo caso, secondo il ragionamento precedente, la valenza dovrebbe avere valore neutrale, ma per aumentare la tensione dell'utente si è deciso di forzare la valenza a negativa in quanto, a questo punto, la partita si sta per concludere e il risultato è completamente incerto.

Sulla base di queste considerazioni è stata composta la base musicale, che deve adattarsi in modo pertinente ai valori di arousal e valenza che, a loro volta, saranno aggiornati dal codice di gioco ogni volta che una nave verrà affondata.

4.1.2 Elementi musicali connessi a eventi specifici

Per quanto riguarda, invece, gli elementi che vanno a sottolineare gli eventi di gioco, essi dovranno essere coerenti con la base musicale a cui si sommano e sembrare parte della composizione stessa. Tra gli eventi importanti identifichiamo:

- il turno del giocatore umano: l'enfatizzazione di questo evento ha sia la funzione di indicare all'utente che può fare la propria mossa, sia di incitarlo nel compiere l'attacco
- l'animazione di attacco delle navi: sottolineare questo evento permetterà di creare anticipazione, aumentando la suspense provocata dall'incertezza dell'efficacia dell'attacco
- l'evento lanciato nel caso in cui viene colpita o affondata una nave: acclamare il giocatore nel caso in cui riesca a colpire o affondare una nave nemica, oppure sottolineare il proprio insuccesso nel caso sia lui a subire una perdita aumenterà il coinvolgimento dell'utente in quello che sta succedendo nel gioco
- il passaggio da “hunt mode” a “target mode”: i giocatori (sia quello umano che l'intelligenza artificiale, che è stata costruita proprio su questo principio), alternano il proprio comportamento tra due modalità: la prima viene chiamata “hunt mode”, che consiste nella ricerca della posizione delle navi nemiche; la “target mode”, invece, è la modalità in cui entra un giocatore quando colpisce una nave avversaria, che consiste nella ricerca dell'orientamento della nave e il conseguente tentativo di affondarla. Enfatizzare il passaggio di uno dei due giocatori alla “target mode” crea una tensione ancora maggiore mettendo in allarme il giocatore che sta subendo l'attacco, facendogli capire che nel giro di pochi turni potrebbe perdere una nave.
- la fine della partita: è bene premiare il giocatore con un jingle che celebri la sua vittoria o che marchi il suo fallimento in caso di sconfitta

4.2 Pianificazione della composizione per esprimere le giuste sensazioni

Per dare la sensazione di una battaglia sempre più intensa con il passaggio dai livelli di arousal più bassi a quelli più alti, sono stati sfruttati principalmente gli elementi ritmici e timbrici della musica [B-5]:
Il primo livello di arousal è costituito solo da elementi che coprono la parte bassa dello spettro di frequenze e senza nessuna indicazione ritmica.
Per creare contrasto con il livello più basso, nel secondo livello sono state inserite delle percussioni suonate in modo più leggero e, nel terzo livello, con una intensità più elevata, aumentando man mano l'impeto della composizione. Parallelamente vengono aggiunti elementi che piano piano riempiono lo spettro di frequenze nei livelli di arousal più alti, andando a espandere il timbro della composizione e aiutando a dare la sensazione di intensità crescente.

Per avere il cambio di valenza e mantenere il contrasto tra i vari livelli di arousal, la composizione è rimasta pressoché invariata a livello spettrale fra le varie sfere emotive, ma per ogni tipo di valenza viene essenzialmente cambiato il framework armonico della musica e vengono aggiunti motivi più o meno melodici a seconda della valenza:
Per la valenza negativa viene costruita l'armonia del pezzo intorno all'intervallo musicale di seconda minore, che, trattandosi di un intervallo formato da note molto vicine, viene percepito dall'orecchio umano come dissonante e sgradevole [B-5]. Inoltre, per aumentare lo spettro ricoperto dalla musica nei livelli più alti di arousal, è stato fatto del sound design introducendo degli effetti che ricordano dei "glitch" ritmici tipici di malfunzionamenti elettronici.
Per la valenza positiva, invece, sono stati utilizzati degli intervalli più piacevoli all'orecchio umano, come l'unisono o la quinta perfetta, oltre ad introdurre delle melodie cristalline che producono un effetto lenitivo.
Per la valenza neutra è stata fatta essenzialmente una via di mezzo dei due estremi, utilizzando delle melodie non troppo dissonanti ma che mantenessero comunque un'atmosfera di tensione.

Il turno del giocatore umano viene sottolineato semplicemente sommando alla base musicale un layer di *shaker* che aumenta il groove della musica, con l'effetto di incitare l'utente a compiere il proprio attacco. Soluzione analoga è stata adottata per enfatizzare il passaggio di un giocatore a "target mode", aggiungendo un layer di un sintetizzatore che scandisce gli ottavi, che allerta il giocatore o lo esorta ad attaccare.

Aggiungendo degli strumenti che creano tensione e anticipazione vengono accompagnate anche le animazioni degli attacchi delle navi, in particolare, per quanto riguarda l'attacco dell'utente, viene gradualmente aggiunto un sintetizzatore che segue il percorso dei proiettili, mentre per l'intelligenza artificiale viene utilizzato un crescendo dissonante suonato da degli archi per creare tensione.
I proiettili, poi, possono mancare, colpire o affondare una nave. Se è il giocatore a colpire o affondare verranno riprodotti delle brevissime frasi musicali con note crescenti per celebrarne il successo, altrimenti, se è l'intelligenza artificiale a colpire, verranno riprodotte delle frasi musicali con note decrescenti che danno un senso di frustrazione.

Melodie crescenti e decrescenti vengono utilizzate anche nei jingle di fine partita, rispettivamente per celebrare o marcare la sconfitta del giocatore.

4.3 Tecniche musica adattiva da usare

Analizzando e testando il prototipo di gioco, si è potuto notare che i cambi di stato della musica sarebbero avvenuti con un breve intervallo di tempo tra essi, questo ha portato alla decisione di utilizzare al minimo la tecnica di horizontal resequencing, che è più indicata per cambi drastici e più distanti temporalmente l'uno dall'altro. Infatti, cambiare completamente il tema musicale molte volte in un breve lasso di tempo rischia di disorientare il giocatore e non riuscire ad immergerlo in una particolare sfera emotiva.
Principalmente, quindi, è stata utilizzata la tecnica di vertical remixing, mantenendo lo stesso tema musicale e cambiandone solo la connotazione. Con strumenti tutti sincronizzati tra loro, è stato molto semplice sottrarre, aggiungere o sostituire layer al mix in corrispondenza degli eventi corretti.

Gli stinger, inoltre, sono stati molto importanti per accentuare alcune azioni e si è fatto in modo che questi venissero "quantizzati" e riprodotti a tempo con la musica di base.

L'unico caso in cui è stata usata la tecnica di horizontal resequencing con musical demarcation branching è stato nell'evento di fine partita, per il quale è stata creata una transizione che porta al jingle di vittoria o sconfitta, anche esso quantizzato sul quarto della musica di base, in modo che sembri una naturale evoluzione del brano.

4.4 Preparazione audio files

Rispetto a un brano di musica lineare, per il quale si può arrangiare, bilanciare e mixare il brano direttamente nella Digital Audio Workstation, nel caso della colonna sonora adattiva per videogiochi vanno pianificati i layer di strumenti e i sample che si vogliono, poi, importare sul software middleware, per creare le logiche che saranno controllate dal codice di gioco.

Il lavoro di composizione per questo progetto è stato svolto su Logic Pro 9, una workstation di casa Apple, ma avrebbe potuto essere svolto su qualsiasi tipo di DAW sul mercato.

4.4.1 Preparazione base musicale

Per creare la musica di base, che interagisce coi parametri di arousal e valenza tramite vertical remixing, si è pensato di dividerla in layer di strumenti, o "stems", che vanno ad aggiungersi in base ai parametri. Nella tabella seguente (Tabella 4) viene illustrata la struttura dei layer per ogni stato musicale relativo ai parametri di arousal e valenza:

	Valenza negativa	Valenza neutrale	Valenza positiva
Arousal livello 1	Droni negativi	Droni neutrali	Droni positivi
Arousal livello 2	Droni negativi + Stem negativo livello 2	Droni neutrali + Stem neutrale livello 2	Droni positivi + Stem positivo livello 2
Arousal livello 3	Droni negativi + Stem negativo livello 3	Droni neutrali + Stem neutrale livello 3	Droni positivi + Stem positivo livello 3

Tabella 4: Layer che compongono la musica di base per ogni stato

Il layer di droni, riprodotti ad arousal di livello 1, costituisce il tappeto musicale sopra cui verranno riprodotti tutti gli altri layer sincronizzati. Eccetto il primo, gli altri livelli di arousal hanno uno stem corrispondente che si aggiunge al tappeto principale.

L'idea alla base del primo livello di arousal, è quella di creare un sottofondo ambient (che richiama molto il genere fantascientifico) che non sia troppo invadente.
Le tracce che compongono questo layer sono solo due:

- Drones: droni di frequenze più basse, che sono le fondamenta del pezzo
- Hi Drones: droni più alti che vengono riprodotti ogni 8 battute per aggiungere varietà.

I droni più bassi sono semplicemente una lunga nota con un filtro passa basso la cui frequenza di taglio è stata modulata con un controller MIDI. Sono state esportate 8 take lunghe 6 battute a 90 bpm, ognuna con una modulazione del filtro differente, creando 8 diverse performance. La prima e ultima battuta sono quelle di attacco e rilascio che verranno sfruttate, poi, su FMOD Studio creando dei crossfade naturali.

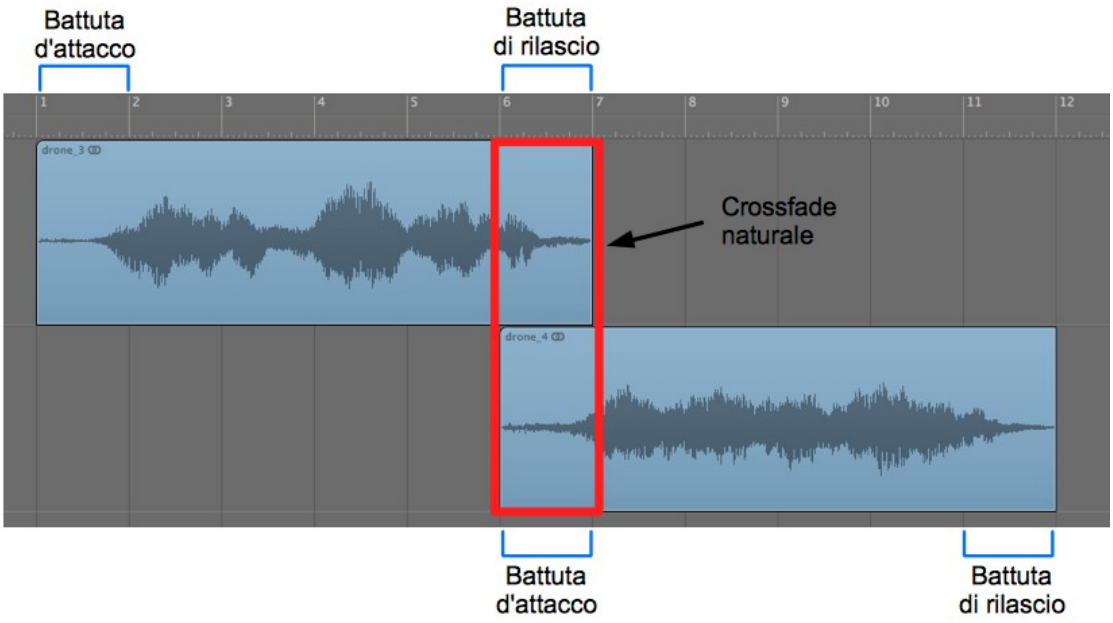

Figura 13: Illustrazione del comportamento dei droni bassi pianificato

Del drone più alto (Hi drone), invece, è stato esportato solo un audio file per ogni valenza, non avendo bisogno di performance differenti in quanto questo verrà riprodotto una volta ogni 8 battute, quanto basta per non renderlo ripetitivo.

Per creare la valenza positiva e negativa sono stati processati in modo differente questi droni. La valenza positiva condivide i droni bassi con quella neutrale, mentre per quella negativa sono state esportate le stesse 8 take processate con un pitch shifter, che ha permesso di creare un intervallo di seconda minore rendendo i droni più dissonanti.
Per i droni alti è stata effettuata la stessa operazione per la valenza negativa, mentre per la valenza positiva è stata creata una nuova take suonando un intervallo di quinta perfetta con un suono più morbido, rendendolo più piacevole all'orecchio.

Per creare il loop di base verranno sfruttate solo queste tracce arrangiandole su FMOD Studio in una maniera strategica.

Per gli altri due livelli di arousal sono state aggiunte delle percussioni che suonano sempre più intensamente un ritmo che rende più energico il sottofondo ambient, creando un contrasto con il livello più basso di arousal, nel quale non c'era nessun

tipo di indicazione ritmica. In secondo luogo, sono stati aggiunti altri strumenti e melodie che aumentano lo spettro armonico coperto dalla musica, rimanendo pertinenti con la dissonanza della valenza negativa e con la tonalità della valenza positiva.
Per ognuno degli stati di valenza e livelli di arousal (eccetto il primo) è stato, quindi, esportato lo stem corrispondente che verrà sommato ai droni su FMOD Studio.

4.4.2 Preparazione elementi musicali connessi a eventi specifici

Per ognuno degli eventi specifici del gioco da enfattizare è stato composto lo stem o il sample corrispondente ed esportato dalla DAW in modo che possa essere aggiunto successivamente sopra alla base musicale.

Per quanto riguarda gli stem che vengono messi in loop (come gli shaker che indicano il turno del giocatore o il layer di sintetizzatore che segnala il passaggio di un giocatore a "target mode") sono state esportate un numero pari di misure che potranno essere messe in loop all'interno di FMOD rimanendo sempre sincronizzate con la base musicale. Essendo il layer di shaker uno stem prettamente ritmico, non aveva bisogno di particolari variazioni, mentre del layer che segnala la "target mode" è stato esportato sia l'originale, con la melodia in unisono con i droni, sia una versione più alta di mezzo tono che verrà attivata nel caso in cui la valenza nella partita risulterà negativa.

Per tutte gli altri eventi sono stati esportati i singoli "sample" one-shot che verranno lanciati da FMOD.

Particolare attenzione è stata dedicata al sample che enfatizza l'animazione d'attacco dell'intelligenza artificiale. In questo caso si è deciso di creare un accordo dissonante suonato da degli archi con la tecnica del tremolo, emulando un suono tipico che si ritrova in molte colonne sonore di film thriller o dell'orrore.
Il sample consiste in un crescendo (attacco) che aumenta la dinamica della performance fino ad arrivare ad un massimo di espressione e di tensione (loop) fino ad un rilascio dell'accordo che riproduce la coda creata dai riverberi.

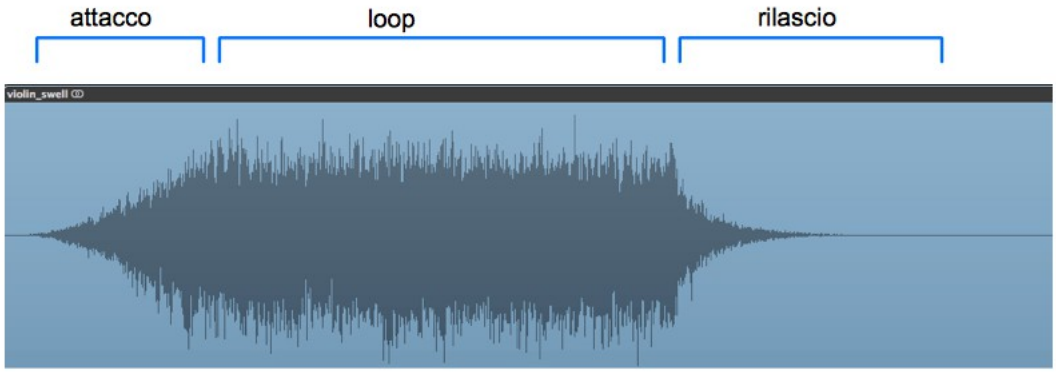

Figura 14: Struttura del sample di archi

Non sapendo precisamente quanto sarebbe stato lungo il tempo dell'animazione di un attacco nel gioco, che dipende anche dal numero di colpi sparati e dalla randomicità dei tempi tra essi, la parte centrale di massima intensità del file audio è stata editata in modo che possa essere messa in loop e allo stesso tempo rimanesse collegata in modo seamless all'attacco e al rilascio della performance.

4.5 Creazione logica con FMOD Studio

Nel creare la logica su FMOD Studio per la colonna sonora, sono stati sfruttati principalmente i *Nested Events:*
si presentano sulla timeline dell'*Event* principale come degli *Event Instruments,* rappresentati da trigger-regions allo stesso modo dei più semplici *Instrument*. Questi *Nested Events* non sono altro che degli *Events* referenziabili che possiedono una propria timeline. Quando la playhead dell'*Event* principale viene a contatto con la trigger-region dell'*Event Instrument*, viene innescata anche la playhead all'interno del *Nested Event* corrispondente.

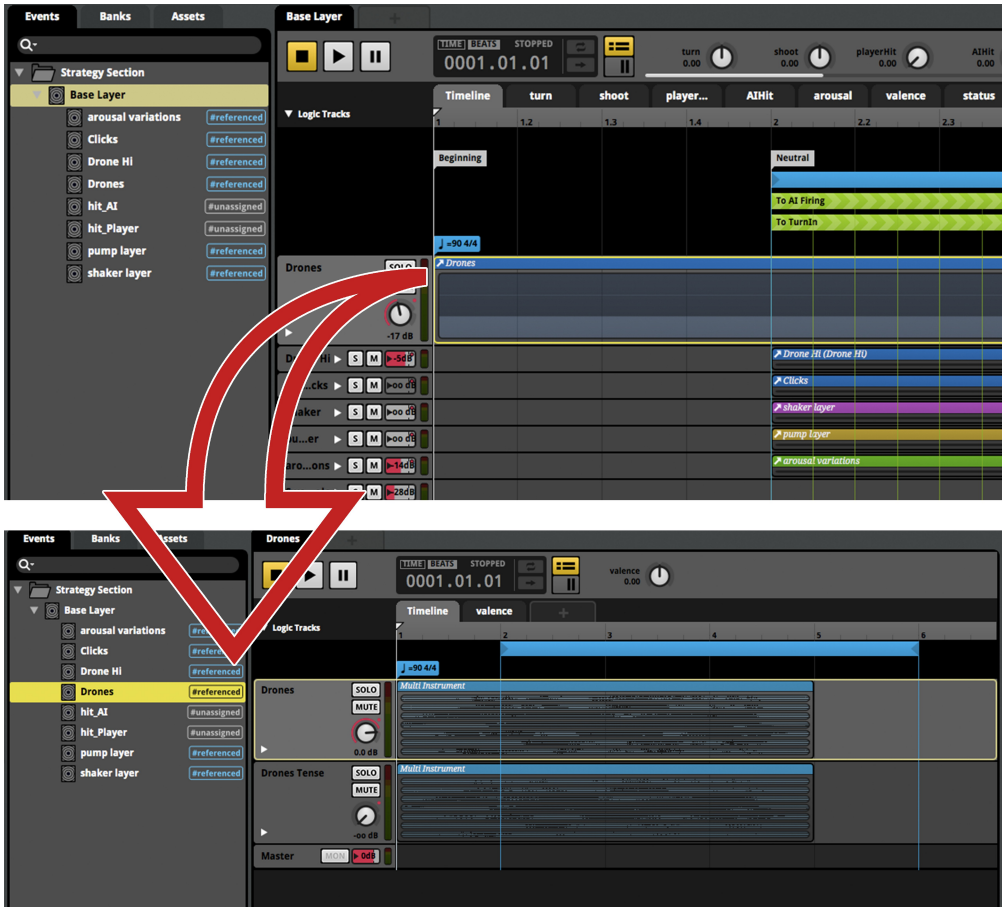

Figura 15: *Event Instrument* e *Nested Event* corrispondente

L'utilizzo di questi oggetti ha fatto si che si potessero creare dei comportamenti più complessi sincronizzando i vari layer e mantenendo pulita e semplice la timeline principale.

I layer sincronizzati con i rispettivi *Nested events* sono 5:

- Drones: il layer di droni bassi
- Drones Hi: il layer di droni alti
- Variations: il layer con gli stem delle variazioni di arousal e valenza della base musicale
- Shaker: il layer che indica il turno dell'utente
- Static Lead: il layer che indica il passaggio a “target mode” di uno dei giocatori

Eccetto il layer “Drones”, che ha un comportamento differente che verrà spiegato in seguito, i file audio all'interno di questi *Nested Events* vengono riprodotti a ripetizione grazie all'inserimento di una loop-region della durata appropriata, sfruttando il fatto che sono stati esportati in modo “seamless” da Logic Pro.
Questi layer saranno sempre riprodotti insieme e, per come sono stati composti e gestiti su FMOD Studio, il loro contenuto audio sarà sempre sincronizzato.
Ognuno di questi *Event Instrument*, al proprio interno, avrà delle tracce con file audio che rappresentano variazioni di valenza e/o arousal dello stesso layer (eccetto il layer “Shaker”).
Prendendo come esempio il layer “Variations”, al suo interno sono state inserite 6 tracce (Figura 16): 3 che contengono gli stem per ogni valenza ad arousal livello 2 e altre tre per ogni valenza ad arousal livello 3 (si ricorda che ad arousal livello 1 l'unico contenuto sonoro della base musicale sono i droni, quindi non ci sono stem che si aggiungono).

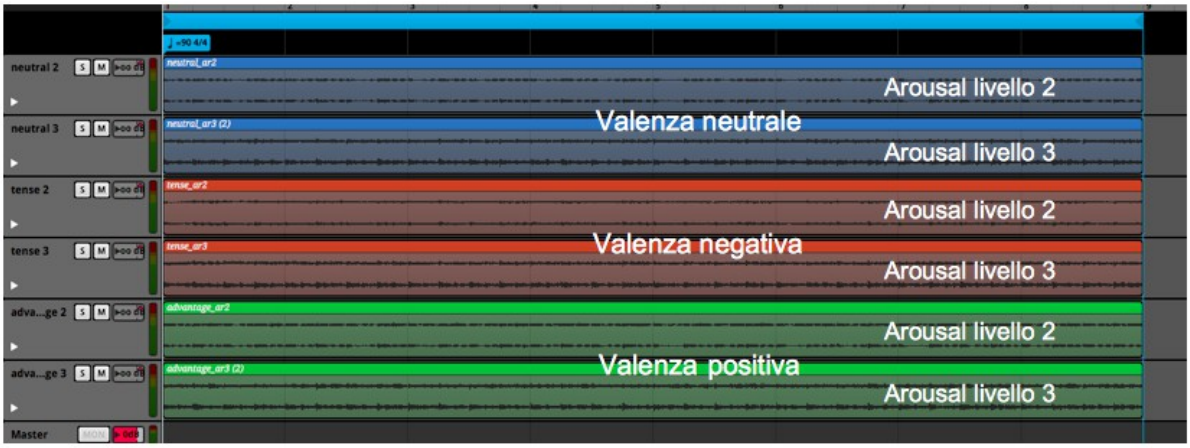

Figura 16: Gli stems contenuti nel layer “Variations”

Dopo aver creato dei parametri “arousal” e “valence” su FMOD è stato possibile usare delle automazioni di volume per inserire o sottrarre dal mix i vari stem (Figura 17) in base, appunto, alla valenza e all'arousal della partita. Quando l'arousal di gioco si trova a livello uno, tutte le tracce in questo *Nested Event* saranno a volume minimo e, ad ogni cambio di valenza e arousal, verrà alzato il volume solo dello stem corrispondente alla combinazione dei valori dei parametri.

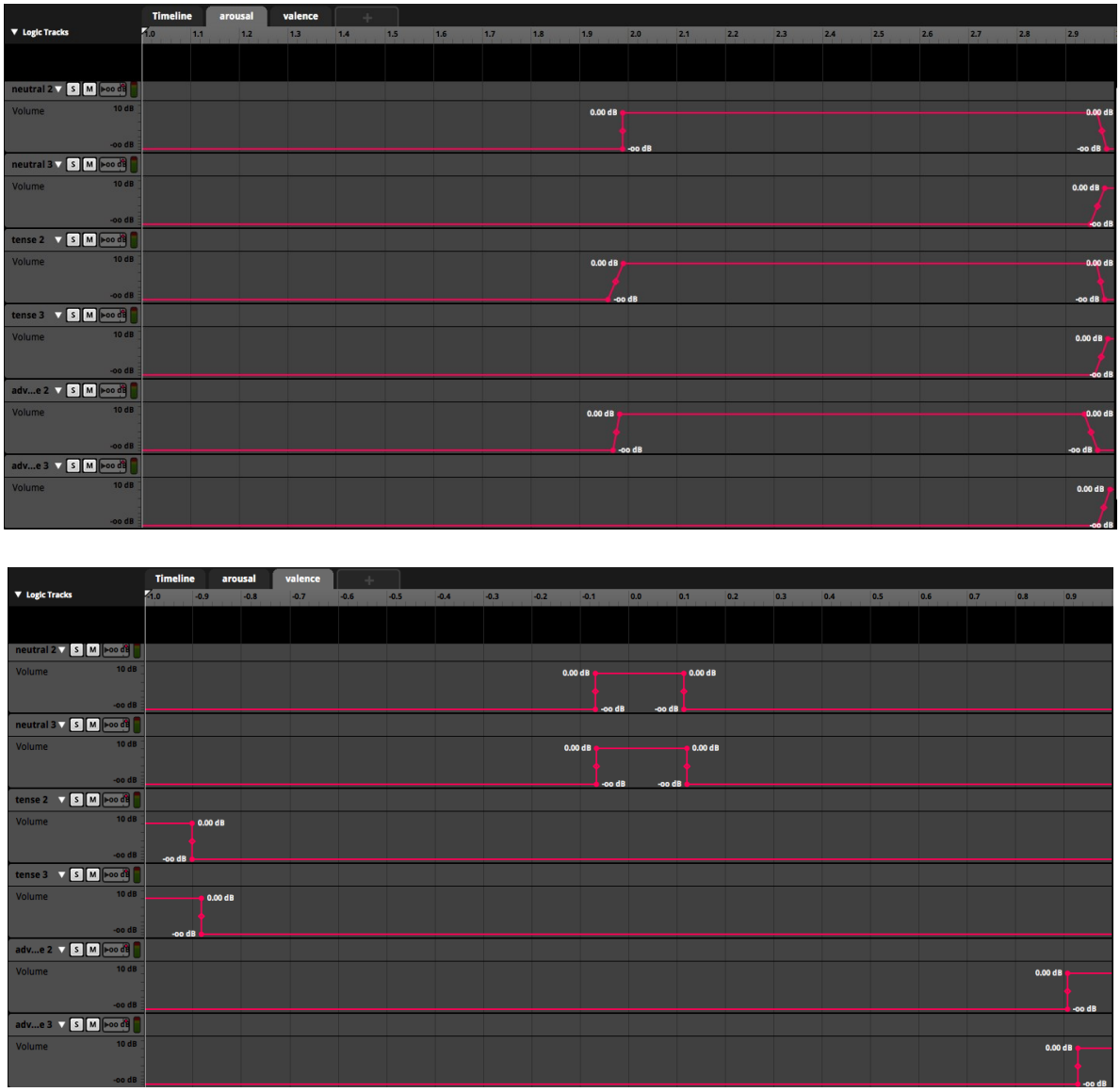

Figura 17: Automazioni di volume del layer “Variations” per valori di arousal (sopra) e valenza (sotto).

I numeri in alto delle timeline in Figura 17 indicano i valori che può assumere ciascun parametro (da 1 a 3 per l'arousal e da -1 a +1 per la valenza). FMOD non fa distinzione tra parametri che possono assumere valori decimali e parametri che assumono valori interi, sarà il codice di gioco che deciderà, come nel caso di questo progetto, se assegnare solo valori interi. Considerando, quindi, i valori interi dei parametri sono state scritte le automazioni per ciascuna traccia contenente uno stem.

In tutti gli altri *Nested Event* citati sopra, sono presenti variazioni di valenza che vengono affrontati, analogamente, con delle automazioni guidate dal parametro “valence”.
Aggiornando i parametri di “arousal” e “valence”, quindi, automaticamente si aggiorna la riproduzione di tutti questi elementi che compongono la colonna sonora rispettando i comportamenti pianificati precedentemente.

Il layer “Static Lead”, oltre ad avere delle automazioni all'interno del proprio *Nested Event* per decidere il file audio da inserire in base alla valenza, presenta

un'automazione anche nell'*Event* principale che "attiva" l'intero layer in base al valore del parametro "status". Quest'ultimo è un parametro a cui verrà assegnato il valore 1 dal codice di gioco nel caso uno dei due contendenti sia passato a "target mode". Il layer "Static Lead", quindi, verrà aggiunto al mix quando il parametro "status" assume valore 1 e, viceversa, verrà sottratto quando avrà valore 0.
Un'ulteriore modifica è stata fatta per rendere questo layer meno invasivo con valori di arousal più bassi: è stato introdotto un filtro passa-basso che processa il segnale in real-time la cui frequenza di taglio viene alzata in corrispondenza del valore del parametro "arousal" (Figura 18). Il filtro, quindi, viene aperto in base all'intensità della partita permettendo di avere un suono più ovattato e meno invadente con arousal minore e un suono più tagliente con arousal più alto.

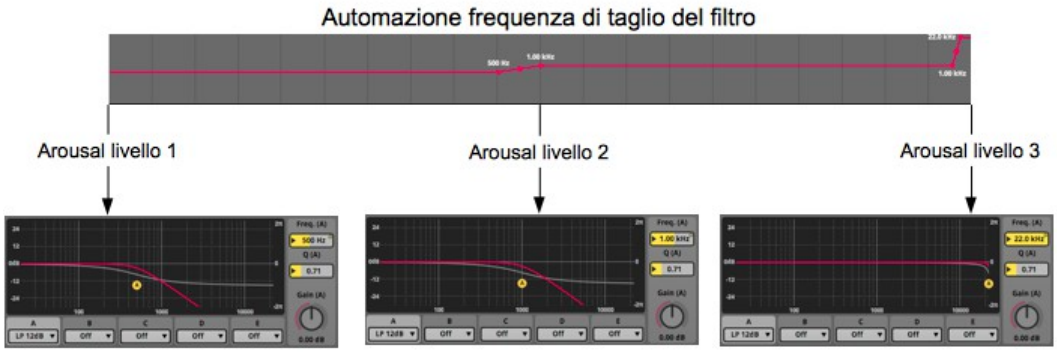

Figura 18: Comportamento del filtro passa basso con i vari livelli di arousal

Il *Nested Event* dei droni bassi presenta una struttura più complicata al suo interno rispetto agli altri *Nested Event*, che è necessaria in quanto questo layer fungerà da fondamenta principali della musica. Le sue tracce non sono composte da singoli stem seamless che vengono messi in loop, ma vengono utilizzate le 8 performance differenti del drone (esportate in precedenza da Logic pro) in un *Multi-instrument* con riproduzione asincrona. Questo significa che, quando la playhead passa sulla trigger-region del *Multi-instrument*, viene riprodotto uno dei file audio al suo interno, selezionato randomicamente, senza che la sua riproduzione rimanga legata alla playhead. Quest'ultima, infatti, serve solo per innescare la riproduzione di uno dei file audio casuali all'interno del *Multi-instrument.*

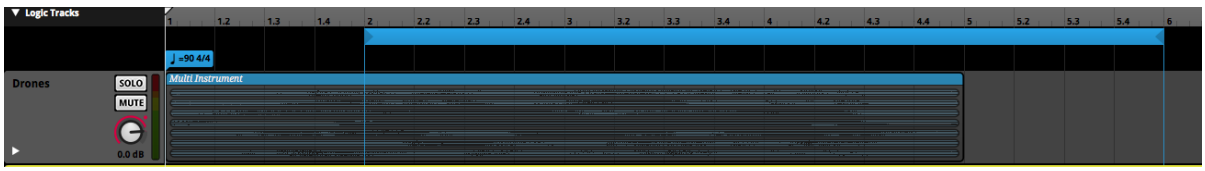

Figura 19: Disposizione di *multi-instrument* e loop-region nel layer "Drones"

La loop-region, all'interno del *Nested Event* si presenta più corta rispetto alla lunghezza dei singoli file audio. Questo fa si che la playhead, quando raggiunge la fine della loop-region, torni all'inizio innescando un secondo file audio dal *multi-instrument* sovrapponendolo al primo. Il secondo file innescato verrà riprodotto a partire dalla sua battuta di attacco, mentre il primo riprodurrà l'ultima parte che corrisponde al "rilascio", creando un cross-fade naturale tra i due file.
Questo sistema genera un continuo loop sempre diverso in quanto ogni file audio, pescato randomicamente dal *multi-instrument* ad ogni passaggio della playhead, presenterà una performance differente.

Un ulteriore metodo per creare vertical remixing su FMOD, usato per gli elementi musicali connessi a eventi specifici, è quello di sfruttare le transition-region: quando la playhead si trova all'interno di queste regioni, si può decidere di farla saltare a un *Destination Marker* cambiando il valore dei parametri coinvolti. È inoltre possibile innescare le transizioni ritardandole ad un punto di demarcazione musicale predeterminato.

Utilizzando questi strumenti, quindi, è stata sfruttata la timeline dell'*Event* principale dividendola in diverse "zone" che coincidono con diverse situazioni ed eventi che si verificheranno nel gioco (Figura 20):

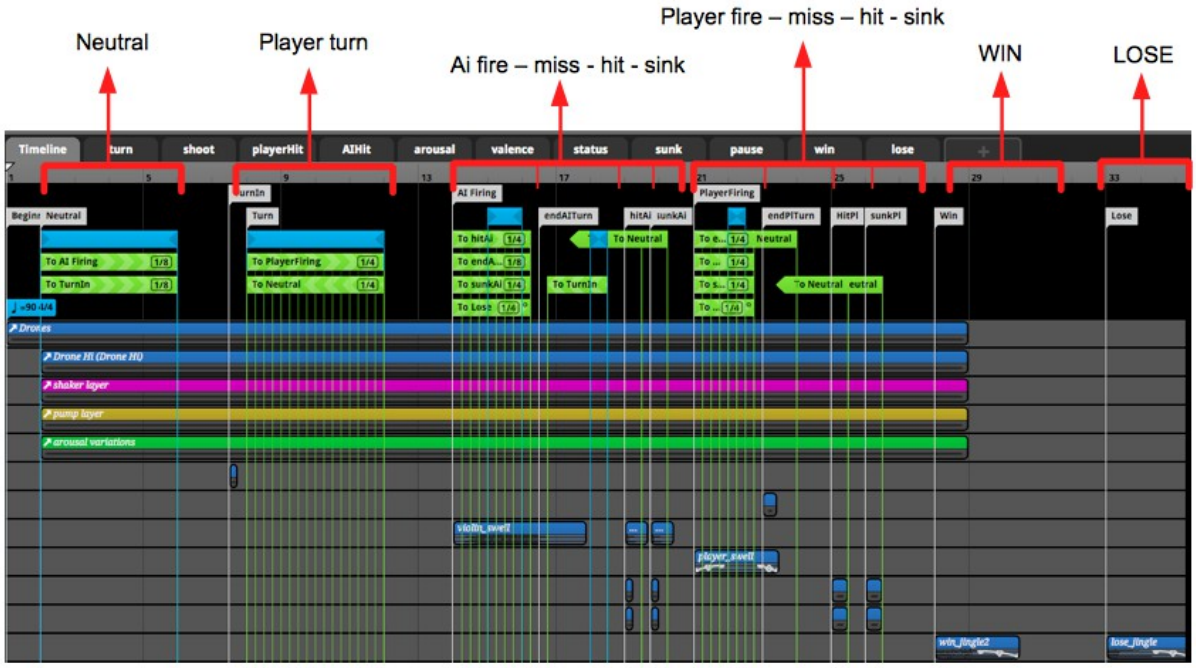

Figura 20: Zone identificate sulla timeline

Dislocando la playhead sulla timeline nelle zone stabilite è stato possibile attivare i comportamenti programmati inserendo sample e automazioni in prossimità di questi punti.
Come si vede dall'immagine la playhead, ovunque si muova, rimarrà sempre a contatto con le trigger-regions dei *Nested Events* e non verrà interrotto o de-sincronizzato l'audio riprodotto in esse. Si può muovere, quindi, la playhead a piacimento innescando i sample a tempo con la musica di base.

Utilizzando trigger-regions e i relativi parametri in modo strategico è stato possibile creare una logica che obbliga la playhead ad attraversare le zone della timeline in un ordine preciso che rispecchia il flusso degli eventi di gioco.
Il diagramma successivo (Figura 21) illustrerà tutti i possibili movimenti della playhead:

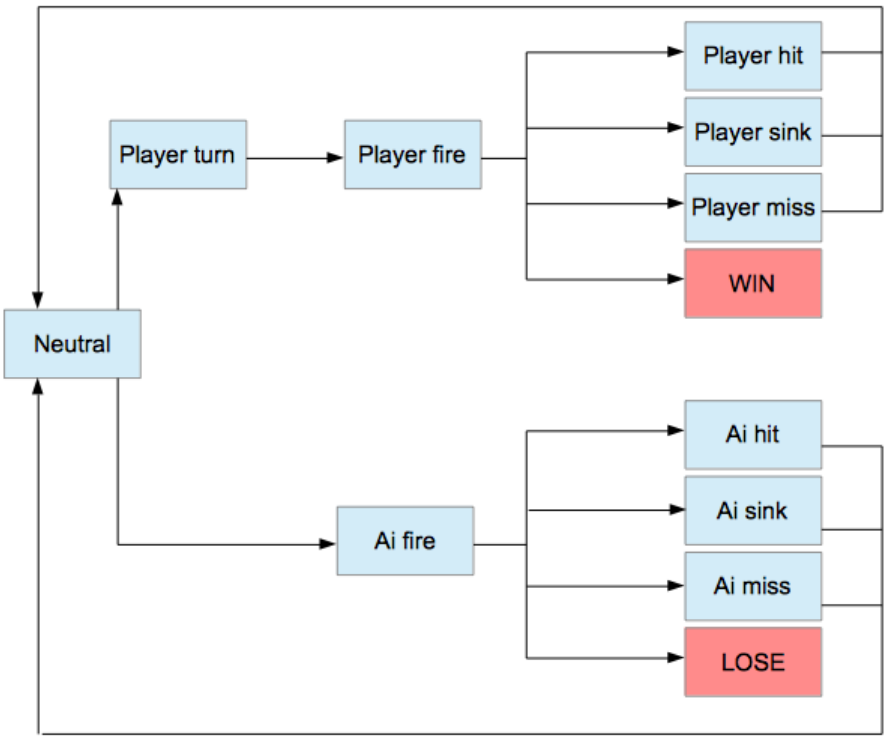

Figura 21: Diagramma rappresentante i movimenti nelle zone possibili dalla playhead

All'avvio del gioco la playhead entrerà nella zona "Neutral", accompagnando lo schieramento della flotta dell'utente sul campo di battaglia. Finito lo schieramento, sarà il turno dell'utente per attaccare e verrà assegnato il valore 1 alla variabile "turn", la playhead salterà, quindi, alla destinazione "Player turn".
A questo punto la playhead innescherà la riproduzione del sample di uno "swell", che accompagna l'animazione della telecamera e introduce il layer "Shaker", sincronizzato con tutti gli altri layer principali. Il layer "Shaker" viene inserito nel mix con una automazione di volume e, finchè la playhead rimarrà in questa zona delimitata dalla loop-region, continuerà ad essere presente nel mix.

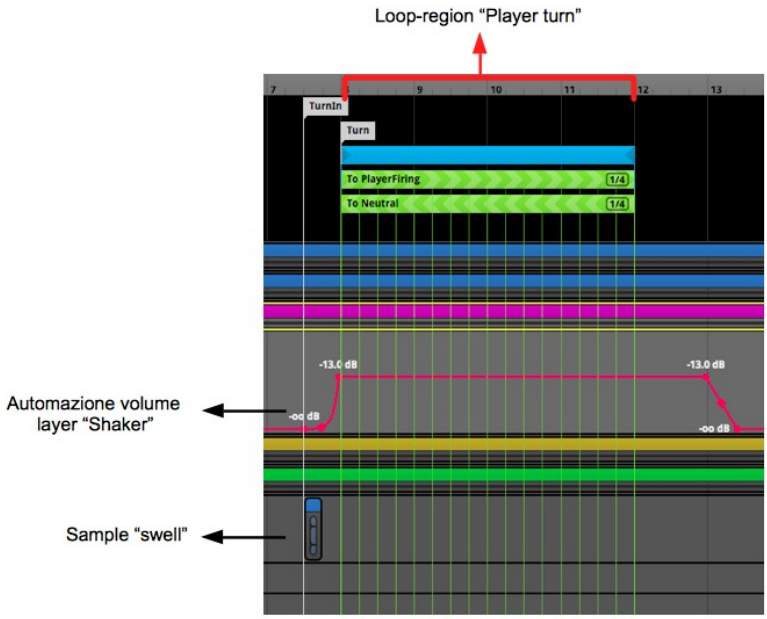

Figura 22: Struttura della zona "Player turn"

A questo punto la playhead potrà solo andare alla destinazione “Player firing” quando viene impostato il parametro “shoot” a 1. Qui la playhead riprodurrà il sample che accompagna l'animazione dei proiettili arrivando, poi, alla loop-region, dentro la quale rimarrà fino a che l'animazione non sarà conclusa. La conclusione dell'animazione verrà comunicata a FMOD Studio dal codice di gioco impostando i valori dei parametri “playerHit”, “sunk” o “shoot” a seconda se, rispettivamente, l'utente ha colpito, affondato o mancato le navi avversarie. In ognuno di questi casi la playhead salterà alla destinazione corrispondente (“Player hit”,”Player sink”, “Player miss”) riproducendo il sample audio relativo al risultato dell'attacco.

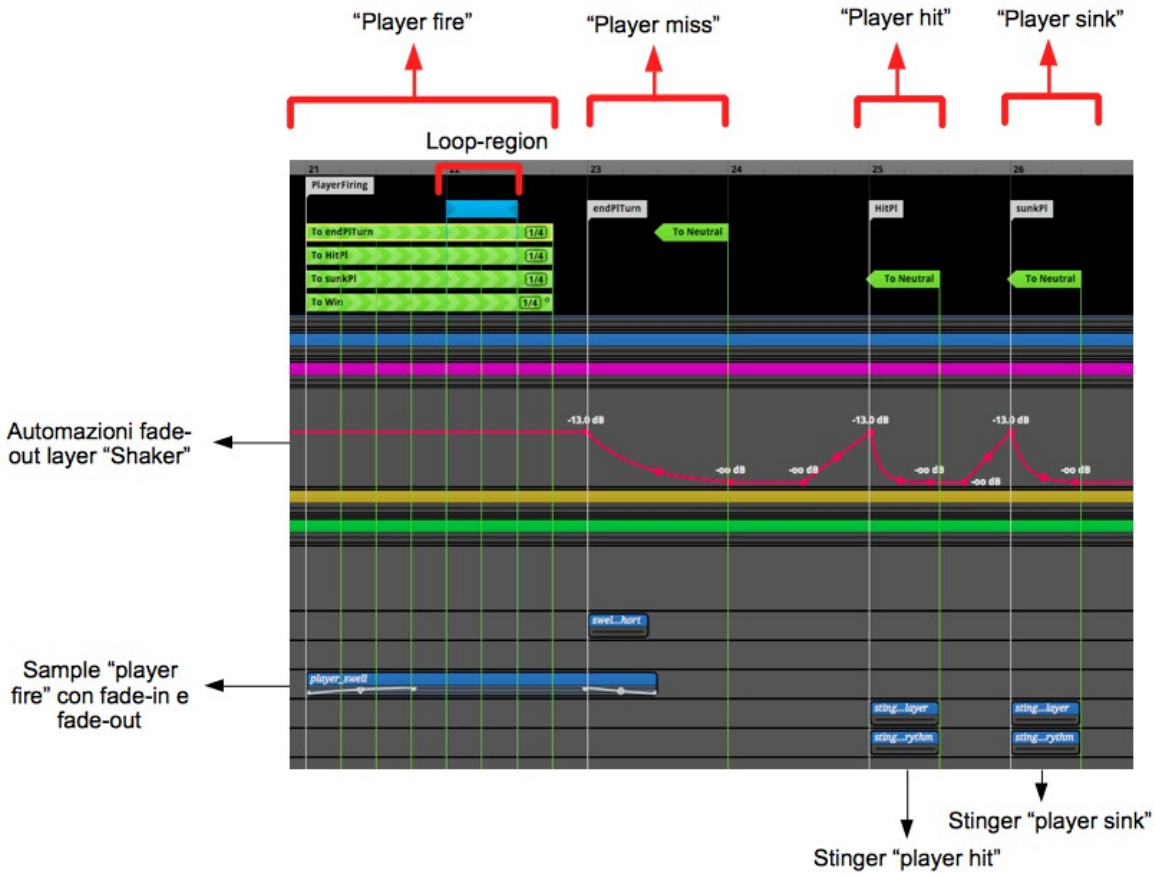

Figura 23: Struttura delle zone “Player fire”, “Player miss”, “Player hit” e “Player sink”

La playhead tornerà, poi, automaticamente alla zona “Neutral” grazie ai *Transition Point* presenti alla fine di ognuna delle zone, dando spazio all'intelligenza artificiale per compiere il proprio attacco.

La playhead salterà, quindi, nella zona “Ai fire” al cambiamento del parametro “shoot”, iniziando a riprodurre il sample del crescendo di archi. Come già spiegato precedentemente questo sample è stato editato in modo che la zona centrale possa essere messa in loop, ed è proprio qui che la playhead rimarrà fino a che l'attacco della AI non sarà terminato. Analogamente alla logica utilizzata per l'attacco dell'utente, la playhead salterà nelle zone di “colpito”, “affondato” o “mancato” in base al valore dei parametri “AIHit”, “sunk”, e “shoot” riproducendo i sample corrispondenti sovrapposti alla coda del sample di archi che accompagnava l'attacco.

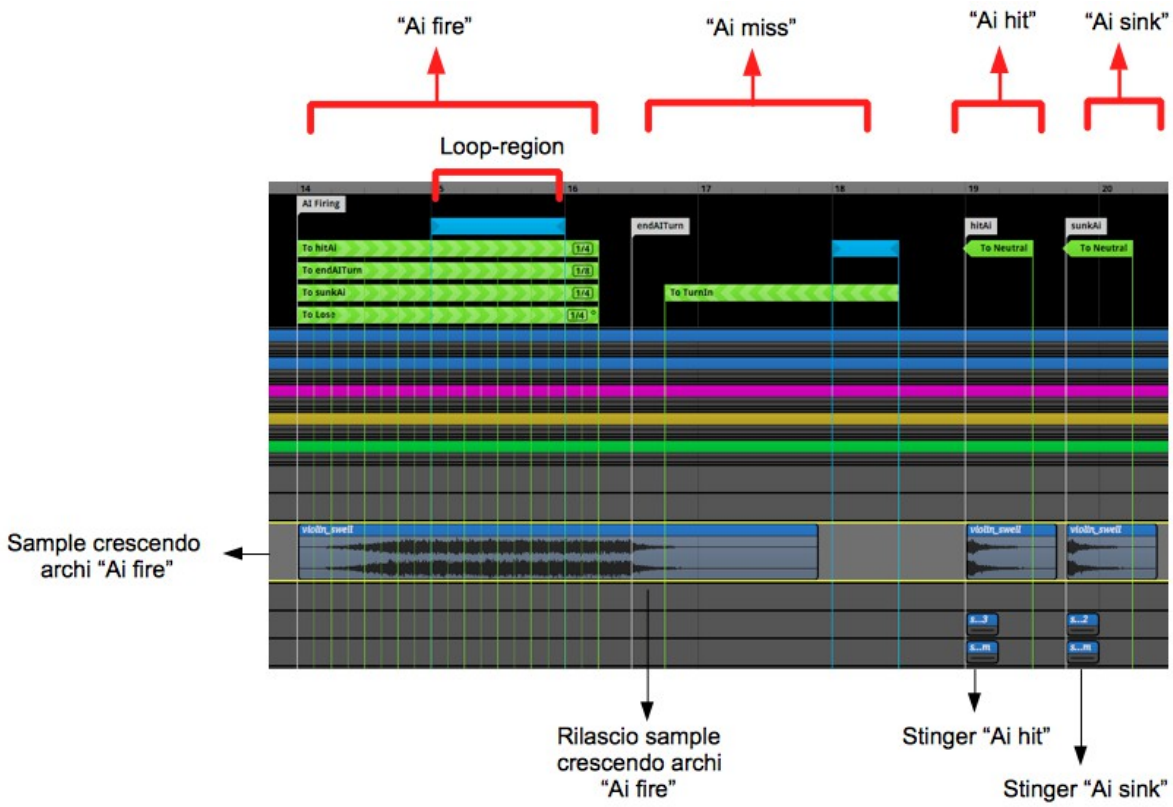

Figura 24: Struttura delle zone "Ai fire", "Ai miss", "Ai hit" e "Ai sink"

La playehead, durante l'azione di attacco dell'utente o dell'intelligenza artificiale, avrà inoltre la possibilità di saltare alle zone, rispettivamente, di "WIN" o "LOSE" (la vittoria si può verificare solo dopo l'attacco dell'utente, mentre la sconfitta solo dopo l'attacco dell'intelligenza artificiale) successivamente all'assegnazione del valore 1 ai parametri "win" o "lose". Questo è l'unico caso in cui è stata usata la tecnica di horizontal resequencing. La logica prevede il passaggio "seamless" tra due segmenti musicali in prossimità di un punto di demarcazione, effettuato sfruttando le *Transition timeline*, uno strumento di FMOD dedicato a questo tipo di comportamenti.

4.6 Implementazione in C++ della colonna sonora su Unreal Engine 4

La Firelight Technologies mette a disposizione dei plugin per integrare le librerie native di FMOD Studio nei più famosi engine, tra cui anche Unreal Engine 4. È necessario solo scaricare il plugin e inserirlo nel percorso appropriato dell'engine per poter usufruire delle funzionalità di FMOD Studio.
È necessario, poi, impostare le preferenze di FMOD Studio in modo che le S*ound bank* vengano compilate all'interno del progetto di Unreal Engine 4 corrispondente.
Una volta importato il plugin e compilate le *Sound bank* su FMOD, sarà possibile usare le funzionalità dell'audio middleware nel game engine.

Per gestire i comportamenti della colonna sonora è stata creata una classe `AudioManager` in cui vengono effettuate tutte le operazioni necessarie. Invece, la classe `FMODAudioComponent` fornita dal plugin, darà tutte le funzioni necessarie per modificare i valori dei parametri della colonna sonora creati su FMOD Studio.

All'avvio della partita la classe GameMode caricherà tutti gli elementi necessari per avviare il gioco, tra questi anche la classe `AudioManager` chiamando la funzione `Init()`. Questa funzione inizializza le variabili necessarie alla classe e assegna l'*Event*, generato da FMOD Studio durante la compilazione delle *Sound banks*, all'`FMODAudioComponent`, mettendolo in play e avviando, così, la colonna sonora.

```
void AAudioManager::Init(ACFG_AIController * aiController, ACFG_PlayerController * humanControl
ler, UBN_GameInstance * gameInstance)
{
    aiHeatmapController = static_cast<ACFG_AIController_HeatmapBased *> (aiController);
    aiPlayer = static_cast<ACFG_AIPlayer *>(aiController->GetPawn());
    humanPlayer = static_cast<ACFG_HumanPlayer *>(humanController->GetPawn());

    this->gameInstance = gameInstance;

    fmodAC->SetEvent(musicEv); //assign event to FMODAudioComponent
    fmodAC->Play(); //play event assigned to FMODAudioComponent
}
```

L'`FMODAudioComponent` mette a disposizione la funzione `SetParameter()` per modificare il valore dei parametri specificati su FMOD in real-time. Per come è stata gestita la logica su FMOD Studio questa funzione è tutto ciò di cui si ha bisogno per far interagire la musica con il gameplay.

La funzione `UpdateOnGameEvent()` permette di sfruttare la state-machine che gestisce gli stati di gioco, in modo da decidere quali eventi di gameplay sfruttare per aggiornare i parametri.

`SelectTurn()` è una funzione che, ogni volta che viene chiamata, aggiorna la variabile di controllo `currentTurn` (quando `true` è il turno dell'utente, quando `false` è della AI) e in base al suo valore aggiorna il parametro "turn" di FMOD.

```
void AAudioManager::SelectTurn()
{
    //update variable currentTurn
    currentTurn = !currentTurn;

    //if it's AI's turn
    if (currentTurn == 0)
    {
        //update FMOD parameter
        fmodAC->SetParameter("turn", -1);
    }
    //if it's user's turn
    else
    {
        //update FMOD parameter
        fmodAC->SetParameter("turn", 1);
    }
}
```

La funzione `Shooting()` controlla di quale giocatore è il turno e in base a questa informazione aggiorna il parametro "shoot" di FMOD.

```
void AAudioManager::Shooting()
{
    //if it's user's turn
    if (currentTurn)
    {
        //update FMOD parameter
        fmodAC->SetParameter("shoot", 1);
    }
    //if it's AI's turn
    else if (!currentTurn)
    {
        //update FMOD parameter
        fmodAC->SetParameter("shoot", -1);
    }
}
```

`ResetTriggers()` resetta al valore di default i parametri di FMOD specificati. Viene usata strategicamente per non creare comportamenti inaspettati nella soundtrack.

```
void AAudioManager::ResetTriggers()
{
    //update FMOD parameters
    fmodAC->SetParameter("sunk", 0);
    fmodAC->SetParameter("playerHit", 0);
    fmodAC->SetParameter("AIHit", 0);
    fmodAC->SetParameter("shoot", 0);
}
```

La funzione `UpdateArousalValence()` controlla il numero di navi sul battlefield e la differenza del numero di navi nemiche rispetto a quelle dell'utente e, di conseguenza, aggiorna i parametri di "arousal" e "valence" secondo la Tabella 2

```
void AaudioManager::UpdateArousalValence() {
    //AROUSAL
    //if the total amount of ships in game is > 7
    if (aiPlayer->GetRemainingShipsNum() + humanPlayer->GetRemainingShipsNum() > 7)
    {
        //update FMOD parameter (arousal level 1)
        fmodAC->SetParameter("arousal", 1);
    }
    //if the total amount of ships in game is > 4 and < 8
    else if (aiPlayer->GetRemainingShipsNum() + humanPlayer->GetRemainingShipsNum() > 4 &&
             aiPlayer->GetRemainingShipsNum() + humanPlayer->GetRemainingShipsNum() < 8 )
    {
        //update FMOD parameter (arousal level 2)
        fmodAC->SetParameter("arousal", 2);
    }
    //if the total amount of ships in game is < 5 or one of the players has only one ship
    else if (aiPlayer->GetRemainingShipsNum() + humanPlayer->GetRemainingShipsNum() < 5 ||
             humanPlayer->GetRemainingShipsNum() == 1 || aiPlayer->GetRemainingShipsNum() == 1)
    {
        //update FMOD parameter (arousal level 3)
        fmodAC->SetParameter("arousal", 3);
    }

    //VALENCE
    //if user has more active ships than AI
    if (aiPlayer->GetRemainingShipsNum() < humanPlayer->GetRemainingShipsNum())
    {
        //update FMOD parameter (positive valence)
        fmodAC->SetParameter("valence", 1);
    }
    //if user has less active ships than AI or user has only one ship
    else if (aiPlayer->GetRemainingShipsNum() > humanPlayer->GetRemainingShipsNum() ||
             humanPlayer->GetRemainingShipsNum() == 1)
    {
        //update FMOD parameter (negative valence)
        fmodAC->SetParameter("valence", -1);
    }
    //if user and AI have an equal amount of ships and user has more then 1 ships
    else if (aiPlayer->GetRemainingShipsNum() == humanPlayer->GetRemainingShipsNum() &&
             humanPlayer->GetRemainingShipsNum() > 1)
    {
        //update FMOD parameter (neutral valence)
        fmodAC->SetParameter("valence", 0);
    }
}
```

La funzione `CheckIfStatusRises()` controlla, sia nel turno dell'utente sia nel turno della IA, se è stata colpita, affondata una nave o se è stata del tutto eliminata la flotta avversaria aggiornando, di conseguenza, i parametri “win”, “lose”, “status”, ”shoot”, “playerHit” o “aiHit”. Infine la funzione aggiorna il turno chiamando `SelectTurn()`.

```
void AAudioManager::CheckIfStatusRaises()
{
    //user's turn
    if (currentTurn == 1)
    {
        //if a ship has been sunk
        if (aiPlayer->GetRemainingShipsNum() < currentAiShips)
        {
            //update control variables
            currentAiShips = aiPlayer->GetRemainingShipsNum();

            currentPlayerHits = gameInstance->Get_HumanStats()->cannonShotsOnTarget;

            //if AI doesn't have any ship left
            if (aiPlayer->GetRemainingShipsNum() == 0)
            {
                //user won, update FMOD parameter
                fmodAC->SetParameter("win", 1);
            }

            //if AI has ships left
            else
```

```
            {
                //user hit and sunk a ship, update FMOD parameter
                fmodAC->SetParameter("sunk", 1);

                //if there are no cells marked "HIT" on the field
                if (!GetHitsOnAiBattlefield() && !GetHitsOnHumanBattlefield())
                {
                    //no player is in "target mode", update FMOD paramter
                    fmodAC->SetParameter("status", 0);
                }
            }
        }
        //if a ship has been hit
        else if (gameInstance->Get_HumanStats()->cannonShotsOnTarget > currentPlayerHits)
        {
            //update control variables and FMOD parameters
            currentPlayerHits = gameInstance->Get_HumanStats()->cannonShotsOnTarget;

            fmodAC->SetParameter("status", 1);
            fmodAC->SetParameter("playerHit", 1);
        }
        //if users missed
        else
        {
            //update FMOD parameter
            fmodAC->SetParameter("shoot", 0);
        }
    }
    //AI's turn
    else
    {
        //if a ship has been sunk
        if (humanPlayer->GetRemainingShipsNum() < currentPlayerShips)
        {
            //update control variables
            currentPlayerShips = humanPlayer->GetRemainingShipsNum();

            currentAiHits = gameInstance->Get_AIStats()->cannonShotsOnTarget;

            //if user doesn't have any ship left
            if (humanPlayer->GetRemainingShipsNum() == 0)
            {
                    //user lost, update FMOD parameter
                    fmodAC->SetParameter("lose", 1);
            }
            //if user has ships left
            else
            {
                //AI hit and sunk a ship, update FMOD parameter
                fmodAC->SetParameter("sunk", -1);

                //if there are no cells marked "HIT" on the field
                if (!GetHitsOnAiBattlefield() && !GetHitsOnHumanBattlefield())
                {
                    //no player is in "target mode", update FMOD paramter
                    fmodAC->SetParameter("status", 0);
                }
            }
        }
        //if a ship has been hit
        else if (gameInstance->Get_AIStats()->cannonShotsOnTarget > currentAiHits)
        {
            //update control variables and FMOD parameters
            currentAiHits = gameInstance->Get_AIStats()->cannonShotsOnTarget;

            fmodAC->SetParameter("status", 1);
            fmodAC->SetParameter("AiHit", 1);
        }
        //if AI missed
        else
        {
            //update FMOD parameter
            fmodAC->SetParameter("shoot", 0);
        }
    }
    SelectTurn();
}
```

Le funzioni `GetHitsOnPlayerBattlefield()` e `GetHitsOnAiBattlefield()` scorrono ogni cella dei rispettivi campi dell'utente e della IA restituendo `true` se esistono celle marcate come "HIT" e `false` in caso contrario. Queste funzioni vengono usate da `CheckIfStatusRises()` per capire se nel turno corrente il giocatore ha colpito o affondato delle navi.

```
bool AAudioManager::GetHitsOnHumanBattlefield() {

    bool ret = false;
    APLG_Battlefield *battlefield = humanPlayer->GetBattlefield();

    //cycle through all the cells on user's battlefield
    for (int y = 0; y < battlefield->height; y++)
    {
        for (int x = 0; x < battlefield->width; x++)
        {
            //if a cell with status HIT is found
            if (battlefield->GetCellAtLocation(FIntPoint(x, y))->GetCellStatus() == CellStatus_
t::CELL_STS_HIT)
            {
                ret = true;
                break;
            }
        }
    }
    return ret;
}

bool AAudioManager::GetHitsOnAiBattlefield() {

    bool ret = false;
    APLG_Battlefield *battlefield = aiPlayer->GetBattlefield();

    //cycle through all the cells on AI's battlefield
    for (int y = 0; y < battlefield->height; y++)
    {
        for (int x = 0; x < battlefield->width; x++)
        {
            //if a cell with status HIT is found
            if (battlefield->GetCellAtLocation(FIntPoint(x, y))->GetCellStatus() == CellStatus_
t::CELL_STS_HIT)
            {
                ret = true;
                break;
            }
        }
    }
    return ret;
}
```

Di seguito (Figura 25) vengono illustrati gli stati di gioco presenti nella state-machine che vengono sfruttati dalla classe AudioManager per aggiornare i parametri di FMOD Studio e far interagire la colonna sonora con il gameplay.

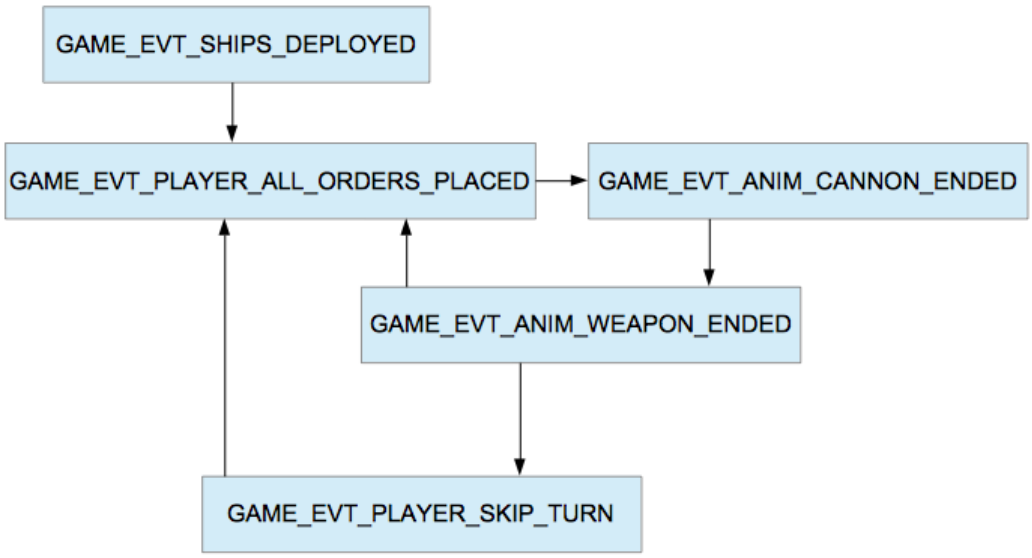

Figura 25: Diagramma che rappresenta l'ordine e il flusso degli stati della state-machine sfuttati

L'evento `GAME_EVT_SHIPS_DEPLOYED` viene scatenato quando l'utente finisce di piazzare la propria flotta sul battlefield e inizia la vera e propria partita. Durante questo evento viene chiamata la funzione `SelectTurn()` che porta la variabile `currentTurn` e il parametro di FMOD "turn" a `1` segnalando il turno dell'utente. Inoltre vengono inizializzate le variabili di controllo `currentPlayerShips` e `currentAiShips`, usate per tener conto delle navi mancanti a entrambi i giocatori.

L'evento `GAME_EVT_PLAYER_ALL_ORDERS_PLACED` viene scatenato quando il giocatore nel turno corrente ha piazzato tutti gli ordini d'attacco e viene inizializzata l'animazione di fuoco delle navi. L'unica funzione usata in questo evento viene chiamata solo se un giocatore ha saltato il proprio turno per particolari condizioni relative alle regole di gioco. Verrà illustrata successivamente.

L'evento `GAME_EVT_ANIM_CANNONS_ENDED` viene scatenato quando viene mostrato il risultato dell'attacco dopo che si è conclusa l'animazione dei proiettili sparati dalle navi. Viene sfruttato per chiamare la funzione `CheckIfStatusRises()` che segnala se il giocatore in questo turno ha colpito o affondato delle navi o se ha vinto la partita. Prima che venga chiamata questa funzione viene chiamata `ResetTriggers()` che porta a `0` i parametri di FMOD che potrebbero essere stati cambiati nel turno precedente e potrebbero influire sulla performance musicale in modo inaspettato.

L'evento `GAME_EVT_ANIM_WEAPONS_ENDED` viene scatenato quando tutte le animazioni di tutti i tipi di armi sono concluse (nel caso della AI viene lanciato nello stesso momento di `GAME_EVT_ANIM_CANNONS_ENDED`) e viene sfruttato per chiamare la funzione `UpdateArousalValence()` e apportare così, modifiche alla musica di base dipendentemente dall'esito dell'attacco.

Un giocatore può saltare il turno se ha usato tutte le navi a disposizione e l'unica nave che aveva ancora attiva è stata affondata nel turno dell'avversario corrente. In questo caso viene scatenato l'evento `GAME_EVT_PLAYER_SKIP_TURN` in cui viene dato il valore `true` alla variabile di controllo `turnWasSkipped` e viene chiamata `SelectTurn()` che cambia il turno immediatamente.
Dopodichè durante l'evento `GAME_EVT_PLAYER_ALL_ORDERS_PLACED` viene controllato se `turnWasSkipped` ha il valore `true`, in questo caso significa che l'avversario ha saltato il turno e non è stato lanciato l'evento `GAME_EVT_ANIM_WEAPONS_ENDED` e quindi non è stata chiamata la funzione `ResetTriggers()` rischiando comportamenti inaspettati. Così viene riportata la variabile di controllo `turnWasSkipped` a `false` e chiamata `ResetTriggers()`.

Con queste semplici funzioni e lo sfruttamento degli stati di gioco è stato implementato correttamente il sistema di musica adattiva pianificato, integrando il software audio middle-ware FMOD Studio con successo.

4.7 Test

Per testare l'efficacia della colonna sonora adattiva nel contesto del videogioco si è deciso di preparare due versioni diverse del gioco:

- Versione 1: presenta la colonna sonora adattiva sviluppata e illustrata nei capitoli precedenti
- Versione 2: fornita di una colonna sonora formata da un semplice loop di sottofondo riprodotto in continuazione, composto arrangiando gli elementi utilizzati per la colonna sonora adattiva e creando un brano essenzialmente lineare, ma con la caratteristica di rimanere "seamless"

Queste due versioni sono state fatte provare a due gruppi formati ciascuno da 10 persone differenti, la maggior parte di queste fra i 25 e 39 anni (60% nel primo gruppo e 70% nel secondo gruppo). I rimanenti si dividono fra individui di età compresa tra i 18 e i 25 anni e di età superiore ai 39 anni. I soggetti non sono stati selezionati in base alle loro conoscenze musicali né in base alla loro abitudine videoludica con il risultato di aver ottenuto dei gruppi molto eterogenei che comprendono sia giocatori abituali che non-videogiocatori, sia esperti di musica che persone senza nessuna preparazione musicale.
Quando è stato presentato il test, non è stato accennato né il lavoro svolto sulla colonna sonora né l'importanza della musica nel progetto, in modo che i soggetti potessero valutare l'esperienza di gioco senza essere influenzati in alcun modo.
Successivamente, è stato chiesto ai soggetti di compilare un questionario per analizzare il loro grado di coinvolgimento e gradimento del gioco. Il questionario era diviso in due parti: la prima riguardava l'esperienza del gioco in generale, la seconda presentava domande più mirate sulla colonna sonora. Si è cercato di evitare domande con risposta si/no e si è deciso di utilizzare per ogni domanda una scala di Likert (da 1 a 5) in modo che i soggetti potessero esprimersi al meglio.

Ci si aspettava che la versione 1 venisse valutata più positivamente anche dal punto di vista dell'esperienza di gioco generale, in quanto la musica è un elemento in funzione del videogioco e, anche se non viene notata e ascoltata dal giocatore, dovrebbe essere percepita positivamente nell'insieme degli elementi.

Alla domanda "Quanto ti è piaciuto giocare?" molti dei soggetti che hanno provato la versione 1 non si sono trattenuti dal valutare il loro gradimento con il massimo punteggio, mentre per quanto riguarda i soggetti che hanno provato la versione 2, seppur abbiano gradito il gioco, la maggioranza di essi ha espresso una valutazione più modesta (Figura 26).

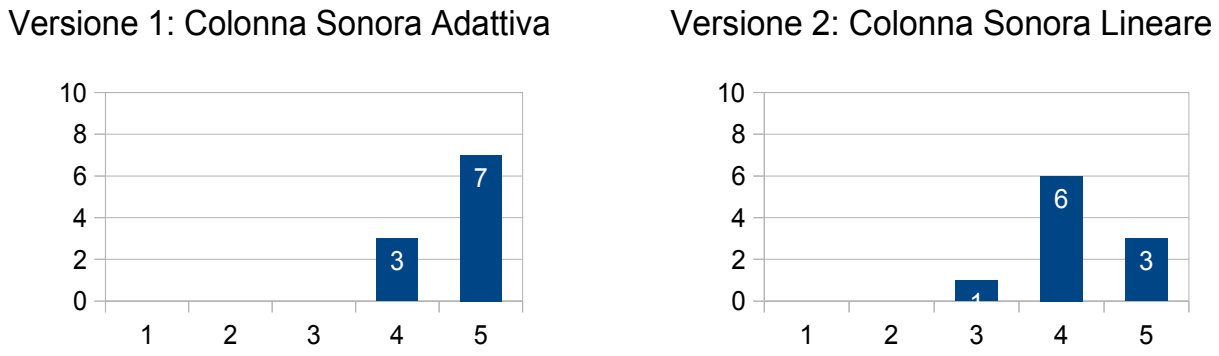

Figura 26: risposte alla domanda "Quanto ti è piaciuto giocare?"

Il coinvolgimento (Figura 27) e lo stimolo (Figura 28) suscitato dal gioco sono stati valutati pressochè analogamente da entrambi i gruppi con una leggera tendenza dei soggetti che hanno provato la versione 1 a dare una valutazione più alta.

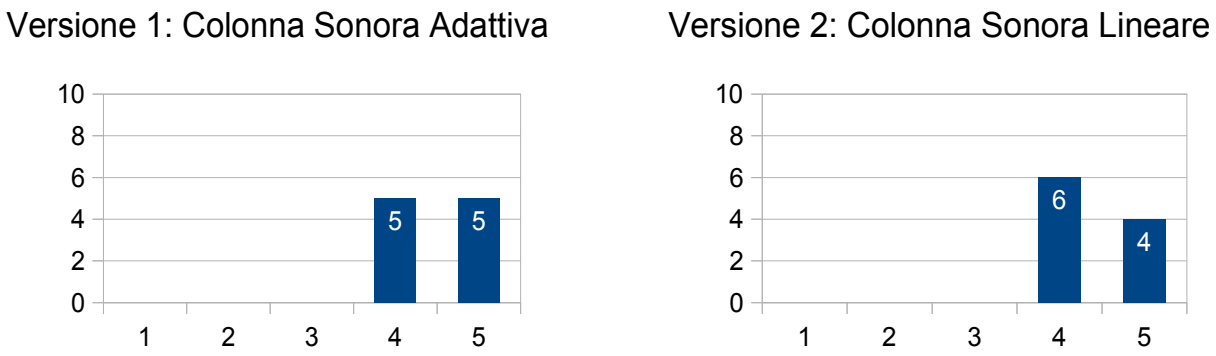

Figura 27: risposte alla domanda "Quanto ti sei sentito coinvolto nel gioco?"

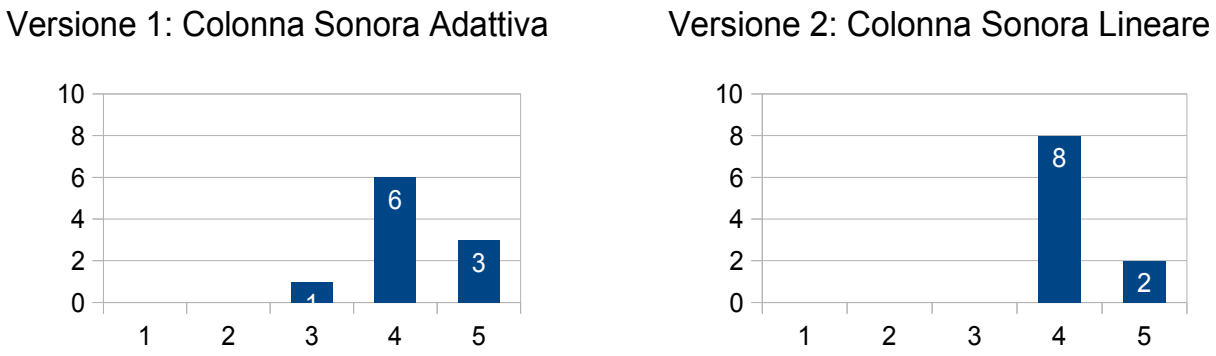

Figura 28: risposte alla domanda "Quanto hai trovato stimolante il gioco?"

Il gioco con colonna sonora adattiva, però, è stato reputato generalmente meno noioso, nonostante anche la versione con musica lineare abbia ricevuto valutazioni positive (Figura 29).

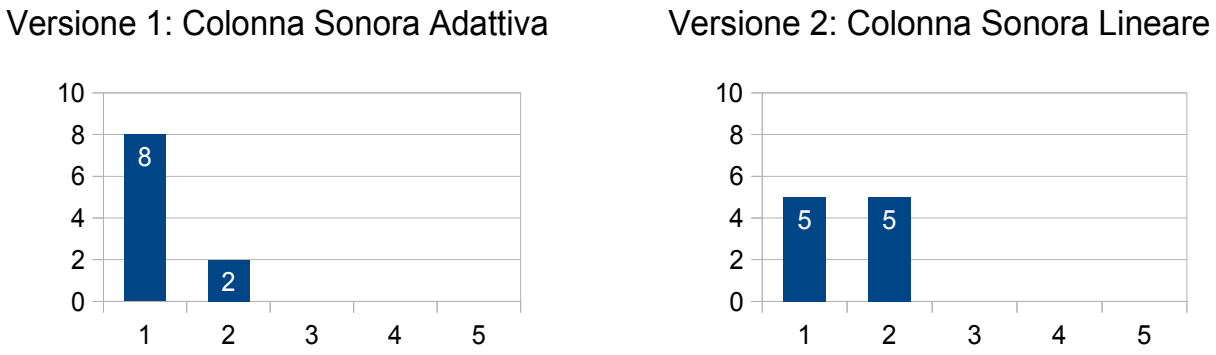

Figura 29: risposte alla domanda "Quanto hai trovato noioso il gioco?"

Nella parte del questionario che riguarda la colonna sonora, entrambi i gruppi hanno espresso un certo gradimento per il comparto musicale a cui sono stati sottoposti giocando, ma sono stati ottenuti risultati migliori con la versione 1 (Figura 30).

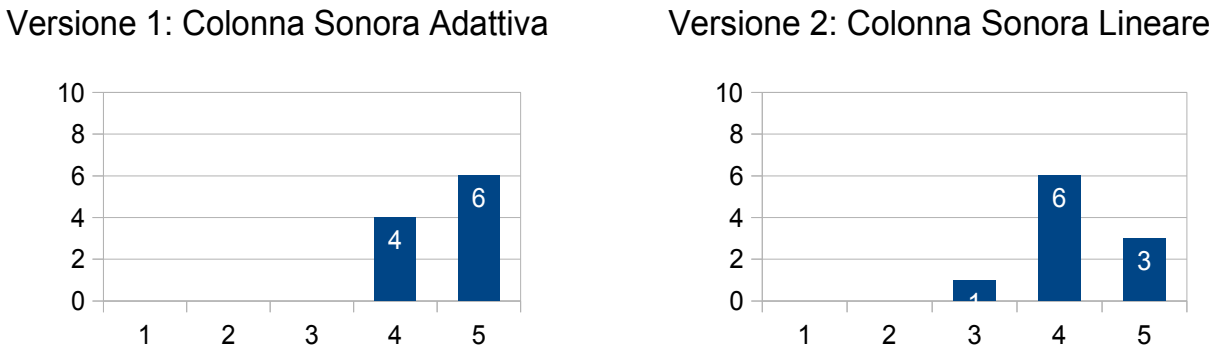

Figura 30: risposte alla domanda "Quanto ti è piaciuta la musica nel gioco?"

Entrambi i gruppi hanno reputato che la musica fosse pertinente con il gioco presentato (Figura 31), risultato abbastanza prevedibile, in quanto la musica nella versione 2 non è altro che la stessa composizione della versione 1 solo arrangiata in modo differente.

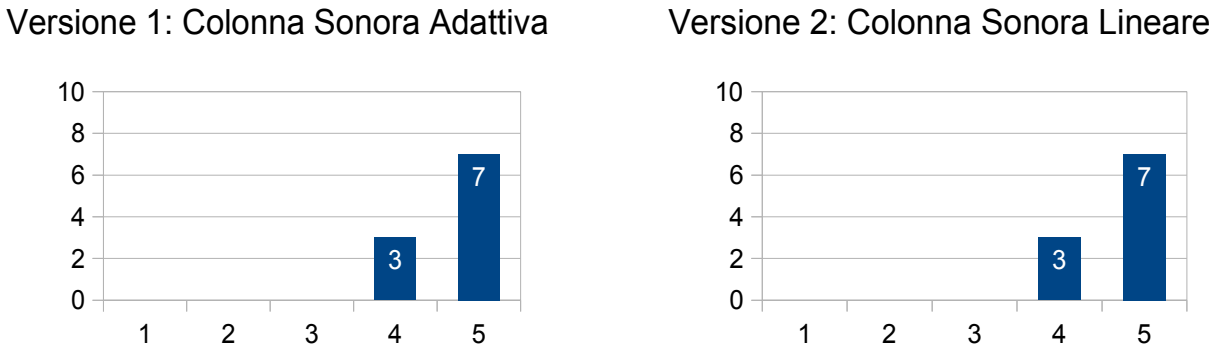

Figura 31: risposte alla domanda "Quanto era pertinente la musica con il gioco che hai giocato?"

Un' ottima valutazione è stata ottenuta dai soggetti che hanno testato la versione 1 alla richiesta di valutare quanto la musica abbia coinvolto l'utente. Per quanto riguarda le valutazione dei soggetti che han provato la versione 2, invece, seppur risulti mediamente positiva, essa si mostra chiaramente peggiore rispetto alla valutazione dell'altro gruppo (Figura 32).

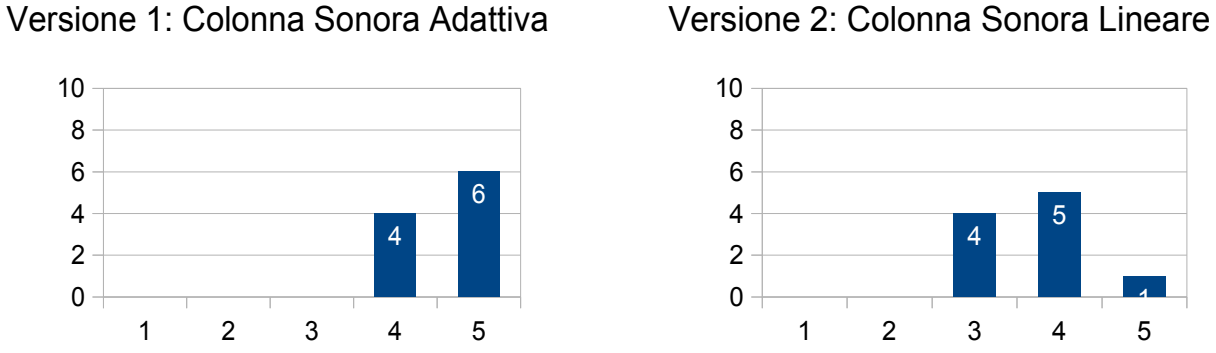

Figura 32: risposte alla domanda "Quanto ti ha coinvolto la musica in quello che facevi?"

Risultato molto soddisfacente è stato ottenuto anche alla richiesta di valutare quanto la musica fosse varia, con pochissimi risultati negativi per quanto riguarda la versione 1 e con decisamente poche valutazioni tendenti al positivo per la versione 2 (Figura 33).

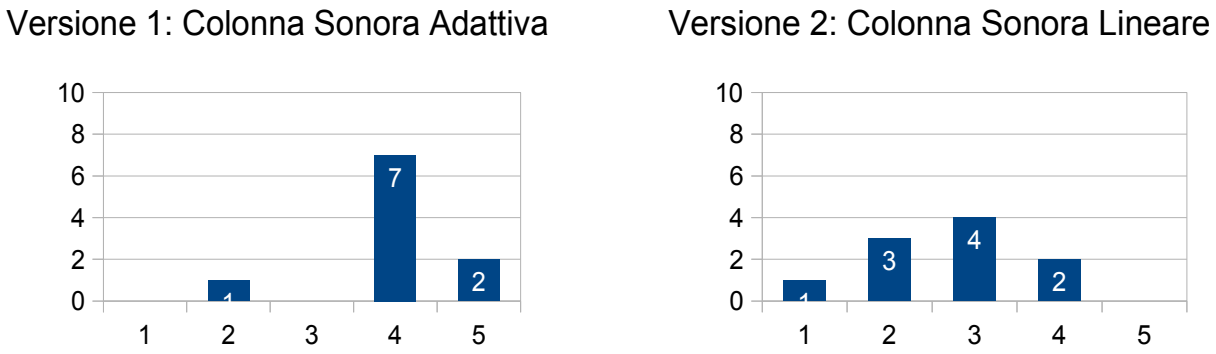

Figura 33: risposte alla domanda "Valuta quanto pensi che la musica fosse varia"

Mentre per quanto riguarda la richiesta di valutare quanto fosse, invece, noiosa la musica i risultati sono stati molto simili per entrambe le versioni (Figura 34).

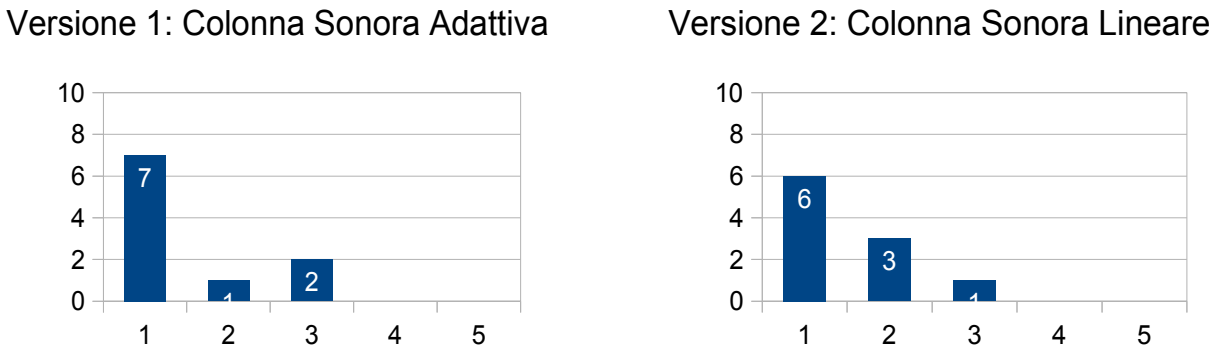

Figura 34: risposte alla domanda "Valuta quanto pensi che la musica fosse noiosa"

Il test ha mostrato, quindi, risultati non sempre drasticamente differenti fra le due versioni, ma generalmente soddisfacenti, essendo per la maggior parte migliori quando si trattava della versione con colonna sonora adattiva. In molti casi, i soggetti che han provato il gioco con colonna sonora adattiva, non si sono trattenuti dal dare valutazioni con il punteggio massimo, mentre i soggetti che hanno provato la versione 2 hanno dato risultati positivi, ma non sempre sono stati del tutto convinti.

Durante i test sono state notate anche delle reazioni involontarie nei soggetti che han provato la versione 1 che non si sono verificate nel gruppo di test della versione 2: alcune di queste reazioni vedevano l'utente canticchiare in modo soddisfatto la melodia quando riusciva a colpire una nave avversaria, oppure scandire il tempo con il piede sentendo gli shaker incitarlo a fare la propria mossa durante il proprio turno. Altri soggetti, ancora, esprimevano un elevato senso di tensione nelle fasi più tese della partita, atteggiamento notato in grado molto minore per i soggetti che hanno testato la versione con musica lineare.
Si può, quindi, implicare che il coinvolgimento effettivo fosse più elevato rispetto a quello riportato nell'autovalutazione.
Nonostante i risultati dei test appaiano soddisfacenti, si è potuto notare che alcune

valutazioni sulla versione del gioco con musica adattiva fossero comunque abbastanza negative. In particolare, alle richieste di valutare quanto la musica fosse varia e noiosa. Si è potuto evincere dai dati anagrafici raccolti, che i soggetti che hanno dato queste valutazioni non positive fossero dei non-videogiocatori. Infatti, osservandoli giocare, si è potuto notare come non fossero in confidenza nell'utilizzare i controlli e come non riuscissero a comprendere nell'immediato le regole del gioco. Questo ha fatto si che impiegassero molto tempo a svolgere certe operazioni che, altrimenti, richiederebbero una manciata di secondi. In questi casi, il tempo trascorso tra due eventi che avrebbero innescato un cambiamento nella colonna sonora risultava troppo lungo, rendendo la musica, in questi momenti, monotona e ripetitiva.

Un'ulteriore lacuna nella colonna sonora è stata rilevata, invece, osservando giocatori più esperti: alcuni soggetti hanno preso confidenza molto velocemente con le regole del gioco, tanto che sono riusciti a mettere in atto nuove strategie che non erano state previste né in fase di design del gioco, né in fase di progettazione della soundtrack. In particolare si è notata la tendenza di alcuni di essi a non affondare subito una nave dopo averne scoperto posizione e orientamento, risparmiando colpi da poter destinare alla ricerca delle navi mancanti. Siccome la colonna sonora reagisce, aggiornando la valenza e l'arousal, in base alle navi che vengono affondate, questa strategia ha fatto si che il sistema di musica adattiva si rivelasse debole e non si comportasse come programmato, rischiando di indurre fatica d'ascolto.

5 Conclusioni

La colonna sonora è solo uno dei moltissimi elementi che compongono un videogioco. È importante che tutte le discipline coinvolte nella creazione dell'opera videoludica cooperino per plasmarne l'esperienza, compresa la musica, che deve far parte di questa sinergia per conferire al prodotto un livello di prestigio molto più alto. Senza una colonna sonora adattiva il videogioco presenterà solo una playlist musicale senza nessun attaccamento al gameplay [S-3]. Come si è potuto analizzare, tramite i test effettuati, con una colonna sonora adattiva ben pianificata e ben composta è possibile coinvolgere il giocatore e intensificare la sensazione che il videogame vuole trasmettere. Senza un sistema di musica adattiva, invece, si perde un'importante strumento per immergere il giocatore e, anzi, si rischia di distaccarlo dal mondo di gioco. Se la musica è composta solo da lunghi file lineari, più questi sono lunghi più la musica sarà disconnessa dalla componente visuale del gioco [S-3], così come dall'esperienza stessa.

Essendo il videogioco un prodotto interattivo e di intrattenimento, il giocatore tende a testare la sua struttura esplorandone le meccaniche e "giocando controcorrente" ("play against the grain" [B-2]). Anche in un gioco semplice, come quello presentato, è accaduto che i giocatori più esperti potessero uscire dal modo di giocare previsto e inventare nuove strategie utilizzando le regole fornite, rischiando di rendere il sistema di musica adattiva inefficace.
Questo dimostra che pianificare una colonna sonora adattiva, non solo significa adattare una musica alle meccaniche e agli eventi, ma anche adattarla al design stesso del gioco. Bisogna tener conto del fatto che i giocatori potrebbero affrontare il videogame in maniera sempre differente da persona a persona, generando esiti totalmente eterogenei per i quali la colonna sonora dovrà sempre comportarsi come pianificato. È bene, prima di progettare un sistema musicale adattivo, capire il tipo di gioco su cui si sta lavorando e studiare il comportamento dei giocatori che ne fruiranno.

Il sistema implementato nel corso di questo elaborato è riuscito a donare al giocatore un'esperienza immersiva più coinvolgente rispetto a una colonna sonora più lineare. È, quindi, riuscito nel proprio intento, nonostante i pochi risultati negativi riportati nei test, illustrati nel capitolo precedente. In relazione a questi ultimi si sarebbe potuto aggiungere degli stinger per catturare l'attenzione di quei giocatori che impiegano più tempo a compiere un'azione e, inoltre, introdurre una divisione di livelli di arousal più fine, che tenesse conto dei colpi andati a segno oltre alle navi affondate. In questo modo, le strategie impreviste adottate dai giocatori più esperti non avrebbero interferito con il comportamento della colonna sonora e la musica si sarebbe dimostrata abbastanza accattivante da non annoiare il giocatore che impiega più tempo a stilare la propria strategia.
Sebbene per ogni opera videoludica andrebbe concepito un sistema che si adatti perfettamente alle meccaniche e al design specifico del gioco, il sistema creato può essere un esempio da utilizzare come ispirazione e base per altri progetti di natura simile, anche integrandolo con altre tecniche tra quelle illustrate.

La musica è uno strumento molto potente, ma nell'ambito dello sviluppo di videogiochi, o altri media interattivi e non, viene spesso considerata come un elemento secondario, cedendo la priorità all'aspetto visuale. Molti sviluppatori non conoscono il potenziale di quello che la colonna sonora adattiva può compiere [S-3],

seppure sia una pratica che viene sviluppata da moltissimi anni. L'industria videoludica, però, si sta sempre più evolvendo e più passa il tempo più gli sguardi si concentrano sulle colonne sonore adattive che, se progettate correttamente, possono donare all'utente un'esperienza totalmente diversa rispetto a ciò che una musica lineare più fare. La colonna sonora può essere sfruttata anche per colmare delle lacune introdotte dal design del gioco, ricatturando l'attenzione dell'utente. Il giocatore si ritroverà ad essere incitato dalla musica stessa e coinvolto più profondamente nel gameplay e, nonostante la sua attenzione non sarà rivolta direttamente alla colonna sonora, egli la percepirà nel contesto del videogioco rimanendo immerso nell'esperienza virtuale.

Bibliografia

[B-1] Michael Sweet. “Writing Interactive Music for Videogames: A Composer's Guide”. Editore Pearson, 2014

[B-2] Michiel Kamp, Tim Summers, Mark Sweeny. “Ludomusicology: Approaches to Video Game Music”. Equinox Publishing, 2016.

[B-3] Laura Ermi, Frans Mäyrä. “Fundamental Components of the Gameplay Experience: Analysing Immersion”. In DiGRA Conference: “Changing Views - Worlds in Play”, Vancouver, British Columbia, Canada. Digital Games Research Association DiGRA, 2005.

[B-4] Gordon Calleja. “In-Game: From Immersion to Incorporation”. MIT Press 2011.

[B-5] Marco Scirea, Yun-Gyung Cheong, Byung Chull Bae. “Mood Expression in Real-Time Computer Generated Music using Pure Data”. In International Conference on Music Perception and Cognition (ICMPC) – Asia Pacific Society for the Cognitive Sciences of Music(APSCOM) joint conference, 2014.

[B-6] Dan Liu, Lie Lu, Hong-Jiang Zhang. “Automatic Mood Detection from Acoustic Music Data”. In 4th International Society of Music Information Retrival (ISMIR) Conference, Baltimore, Maryland, USA, 2003.

[B-7] James A. Russell. “A Circumplex Model of Affect”. Journal of Personality and Social Psychology, vol 39, No 6, 1161 – 1178, 1980.

Sitografia

[S-1] Andrew Clark. “Defining Adaptive Music”. `https://www.gamasutra.com/view/feature/129990/defining_adaptive_music.php`, visitato a gennaio 2018.

[S-2] Jack Grillo, Philip Lemperski. “Dev blog: Scoring Dynamic Combat with Procedural Music”. `http://tombraider.tumblr.com/post/131986215580/dev-blog-scoring-dynamic-combat-with-procedural`, visitato a gennaio 2018.

[S-3] “Interview with Game Composer Guy Whitmore”. `http://melodrive.com/blog/interview-game-composer-guy-whitmore-adaptive-music-isnt-important-games-mandatory/`, visitato a gennaio 2018.

[S-4] IMUSE su wikipedia. `https://it.wikipedia.org/wiki/IMUSE`, visitato a gennaio 2018.

[S-5] “Imuse Demonstration – Seamless Transitions”. `https://www.youtube.com/watch?v=7N41TEcjcvM`, visitato a gennaio 2018.

[S-6] “Uncharted 4 Puzzle Stingers”. `https://www.youtube.com/watch?v=cqqxdyPHjkM`, visitato a gennaio 2018.

[S-7] “Mass Effect 2 Best Interactive Score”. `https://www.youtube.com/watch?v=yvgbxv18PDM` visitato a gennaio 2018.

Ringraziamenti

Vorrei ringraziare il professor Luca Andrea Ludovico e il dottor Giorgio Presti per avermi assistito durante lo svolgimento di questa tesi.
Un grazie va anche a tutti coloro che, in un modo o nell'altro, hanno partecipato alla realizzazione di questo progetto, in particolare al team Fenici, senza il quale non sarebbe stato possibile.
Vorrei inoltre ringraziare la mia famiglia, alla quale sarò sempre grato per il continuo supporto e incoraggiamento, e, infine, Ine che mi sopporta e mi sprona ogni volta a migliorare.

Grazie di cuore a tutti.