Leitmotif in a Dynamic Environment

AJ Kolenc

Committee Chair

Dr. Michael Nitsche

Committee Members

Dr. Jason Freeman, Dr. Brian Magerko

In Partial Fulfillment of the Requirements for the Degree

Master of Science in Digital Media

School of Literature, Media, and Communication

Georgia Institute of Technology
May 2015

TABLE OF CONTENTS

I. INTRODUCTION

Richard Wagner saw music as a means of expressing drama; his famous tool for doing so is the leitmotif, a small music phrase associated with a specific object or idea. Through the manipulation and superposition of these themes, Wagner recreates and enhances the story being told on stage purely in music. Although the use of leitmotif is often associated with Wagner, it has since been used in film, opera, and even video games. Unlike opera or film, video games feature interactions and events that are triggered by user input; this makes traditionally composed, leitmotif-driven music at best out of sync, and at worst completely irrelevant to the events on-screen. “Drama” in video games is not prescribed: it can happen spontaneously through user input and can involve variable combinations of objects and ideas.

One approach that has been explored to resolve this dilemma is the procedural creation of music to fit the current state of the game. Whether through manipulation of pre-composed pieces or algorithmic generation of compositions themselves, these attempts have often focused on capturing the mood or atmosphere of the moment rather than the actual events onscreen. I ask this question: is it possible to algorithmically manipulate several leitmotifs to enhance and describe the dynamic environment of a video game?

To answer this question, I created a project that explores the interaction of several agents — each with a distinct leitmotif — and dynamically creates music that combines these themes in musically and dramatically appropriate contexts. This involved designing techniques for effectively handling multiple leitmotifs, building a procedural music engine that uses those techniques, and creating a small game that generates dynamic drama as input for this engine.

II. PROBLEM AND SOLUTION

Problem

Despite their great diversity in genres and appearances, video games all share one element in common: player agency. In order for a game be effective, players must know that their actions can affect the course of the game in a meaningful sense, whether this is total exploratory control or limited reactionary events. This profoundly affects drama as understood in an operatic or filmic context; the work is no longer a static creation, but a dialogue between the player and the game.

Music has long been associated with drama, and its use in opera and film has been thoroughly investigated. Although music is constantly applied to video games, the short age of the medium has shown no precise or entirely satisfactory method to do so. Several innovative games (Otocky, Spore, Portal 2) have experimented with new uses of music, but most games simply use film or genre music out of context and try to smooth the rough edges. The problem is: how can music be uniquely suited to the medium of a video game, while still providing the dramatic function of its application in film and opera?

Solution

I believe that leitmotifs can hold the key to drama in a dynamic, interactive environment. In fact, leitmotifs are famously used in many popular video games: the Legend of Zelda's "Opening a Chest" theme, Super Mario's underwater theme, and Sonic the Hedgehog's drowning theme are just some of many instantly recognizable tunes. The problem is that these games don't go far enough: they use these themes in pre-composed packages that are either context-insensitive or scripted into the game. I propose taking leitmotifs into more complex and dramatic areas. For music to accurately describe the player-paced action onscreen, it needs to be dynamically generated.

The core of my project is an engine that provides access to the required musical transformations to use leitmotif dynamically. This engine — coupled with an algorithm that views the drama and responds to it in real-time — creates descriptive music that uses the themes of its entities in intelligent ways. Leitmotif has already been heavily developed through its application in film and opera: by analyzing the music in these media, I have discovered proven techniques to manipulate leitmotifs and used them in my engine. To demonstrate its capabilities, I have also created a small game with tightly-controlled interactions to drive the drama the algorithm interprets.

III. BACKGROUND

History of Leitmotif

A leitmotif is a theme that is associated with a particular entity within another medium. By tracing its history, I was exposed to many prominent usages of leitmotif in different media, which informed the manipulations I chose to implement in my project.

Origins

Leitmotif was a term coined by Friedrich Wilhelm Jähns in reference to the music of Carl Maria von Weber. It translates roughly to "guiding motif", and is a more specific version of the musical concept of a motif: a short, recurring sequence of notes. What distinguishes leitmotifs is their tie with a specific person, place, thing, or idea; when a leitmotif plays, the object it is connected with is recalled in some manner.

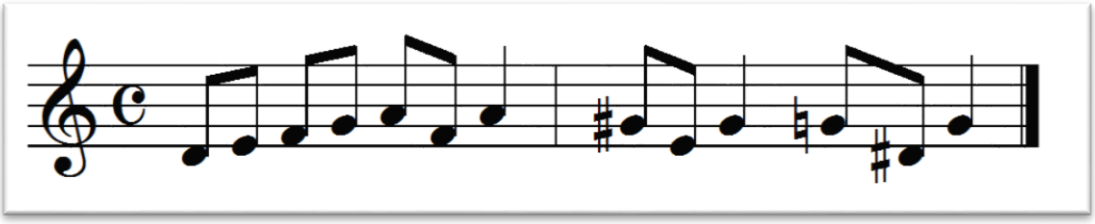
Although they were used to some degree before his time, Richard Wagner is the composer most often associated with the use of leitmotifs. His *Der Ring des Nibelungen*, a cycle of four operas, uses anywhere from dozens to hundreds of leitmotifs (depending on the analyst) which recur both within each individual opera and across the four (Constantini, 2010).

The appearances of leitmotifs in Wagner's works is a widely researched topic. Many music scholars have provided "guides" to Wagner's works that describe the individual

themes and follow them throughout the works. One of the most famous is that of Deryck Cooke, a British musicologist whose work on the subject was stopped before completion by his untimely death. His posthumously-published analysis *I Saw the World End: A Study of Wagner's Ring*, while incomplete, provides a more in-depth analysis of Wagner's use of leitmotif than previous analyses, which tend to simply note appearances rather than trace development (Cooke, 1992).

Applications in Film

Leitmotif has greatly influenced film scores. The technique was widely used during the 1930's and 40's, although it has its roots in the silent era when music would be the only sound heard by the audience. When used in film, leitmotif helps the audience identify characters and places, but it can also be used to convey extra information. For example, in *M* the murderer is identified by the tune he whistles.



Ex. 1: Theme in M, From Peer Gynt *(Grieg, 1875)*

M's use of leitmotif was groundbreaking for incorporating the theme within the diegetic framework of the film; later movies such as *Inception* would play with this concept nearly a century later (Constantini, 2010).

Many film composers continue to use leitmotif today: the famous *Star Wars* scores by John Williams use them heavily and are credited with a revival of popular interest in the technique.

Perhaps the most extensive use of the technique in recent times is by composer Howard Shore for the expansive *Lord of the Rings* trilogy. Doug Adams, a musicologist who documented the creation of the scores for those films, identified over 80 themes and variants used throughout the nearly 12 hours of music. Shore carefully crafted his themes with underlying structures and instrumentation to allow identification on multiple levels, as well as development over the course of the work (Adams, 2010).

I chose to study *Lord of the Rings* in greater detail for several reasons. The sheer quantity of material provided many opportunities to identify different uses of leitmotif, and the drama it describes is clear and varied. Doug Adams' detailed analysis of this material not only allowed me to avoid the task of examining such a large score, but is also constructed to specifically analyze leitmotifs in the work.

Other Applications

Although film has seen the most widespread adoption of the technique, other media have turned to leitmotif to provide musical descriptions of their drama. Wagner first used the technique in his operas, and later composers such as Debussy and Berg used their own versions of leitmotifs in their operas as well. Russian composer Prokofiev used

the technique quite explicitly in his *Peter and the Wolf* composition; each character has their own theme, as well as a specific instrumentation. Several famous ballets have used leitmotif, including Delibes *Coppélia* and Stravinsky's *The Firebird* (Woodstra, Brennan, & Schrott, 2005).

Despite its relatively recent introduction into music, leitmotif has had a profound impact on the medium and specifically its usage within other media. I chose some of the more prominent historical uses of the technique to review in detail – Wagner's *Der Ring Des Nibelungen* and Shore's *The Lord of the Rings* – and coupled them with samples from other composers to provide the basis of my engine's techniques. This review, however, only encompasses one part of the solution. In order to create a procedural music engine, I needed to research approaches to creating music dynamically.

History of Procedural Music

Algorithmic composition is defined as the use of explicit procedures to generate music, typically without direct human intervention. Additionally, the term is tied to the use of computers to produce this music, although in theory the algorithms could be executed without the use of a computer. One particular example of this is an 18th century game that created minuets and trios by rolling dice and using the result to append a measure of music from a table (Phillips, 2014).

Origins

One of the first forays into computer-composed algorithmic music was the Illiac Suite (1955), a piece composed for string quartet by a specialized computer (programmed by Lejaren Hiller and Leonard Isaacson). Surprisingly advanced for the time, the piece showcases the computer's ability to follow counterpoint rules, improvise, and even create music based on non-musical algorithms: Markov chains. Instead of playing the score itself, the computer output a score for string quartet to be played by humans.

Iannis Xenakis, on the other hand, used the computer as a compositional aide rather than as the composer itself. Using "stochastic" methods of composition, he let the computer decide notes based on random number generators, weighted with probabilities he decided. He then took this material and created his own compositions out of it. Xenakis's weighted probabilities served as inspiration for the addition of an "elaboration" feature later in the project (see V: Additional Features).

Two more recent projects are also worth mentioning: CHORAL and EMI. CHORAL takes the rules Bach followed in his part-writing to create chorale harmonizations in the same style; the sheer thoroughness of the rules it follows (over 270) is worth mentioning. EMI (Experiments in Musical Intelligence) is a program that, instead of following programmed rules, attempts to learn the rules itself by synthesizing music from various

styles. It then attempts to mimic music in a given style using musical material it has already heard (Collins N. , 2010).

Methods

As is evident from the various projects in the history of algorithmic composition, there are multiple ways to approach the problem.

Rule-based systems. By using musical rules already deduced by music theorists, these algorithms create pieces that follows these rules as well as they can. CHORAL uses his method extensively.

Stochastic systems. These systems use a high degree of randomness to create compositions, often weighted through the use of Markov chains or user input. Although most dynamic music uses some amount of stochastic processes, Xenakis' compositional aide exemplifies the method.

Artificial intelligence systems. Using the power of AI algorithms already invented for the computer, these systems attempt to deduce a style from the examination of large sets of musical data. EMI's machine learning imitation is based on this method.

These categories are far from exclusive: many attempts at algorithmic composition use techniques from multiple systems (Fernández & Vico, 2013).

Real-time Applications

Pioneers in procedural music have developed real-time systems that generate music for other purposes. OMax, a "virtual improviser Max agent," synthesizes sound data from other musical performers and improvises along with them, matching their patterns and play style (Assayag, Bloch, Chemillier, Cont, & Dubnov, 2006). Another computer improviser is George Lewis' Voyager that plays along with his trombone solos to create interesting jazz music (Lewis, 2000).

A related space involves the generation of music through dance. These projects use the motions of a dancer to create dramatically appropriate music, using theories of dance movement and coupling them with music theory (Dean, 2009). Although the movements of a dancer qualify as a dynamic environment, the focus on a single actor and thus the lack of musical motives limit their application to the realm of video games.

Procedural music has been created in many unique ways; randomness, steadfast rules, and even processing large amounts of data. The requirements of my engine along with the practical matter of implementation eliminated AI methods as viable options, but the remaining two methods I identified would both factor into the final design of the engine.

Procedural Music in Games

History of Game Music

Music started appearing in video games soon after these games became popular. Early music tended to be simple, digitally generated sounds, which needed to be manually coded into the game. As technology progressed, sound quality improved as well as the ability to experiment with music in games; for example, *Dig Dug* featured music that only plays when the player moves (Collins K. , 2008).

Gradually, with the increase in sound quality and storage capabilities for games, game music began to more closely resemble film music. Several game series have become well-known for their music, including the *Mario* games and *Final Fantasy*. For the most part, these games featured pre-composed tracks that would change when the game changed level or state: when Mario picks up a star power-up, for example, the music abruptly shifts to an up-beat tune to indicate invincibility. Early games often featured musical sound effects, and *The Legend of Zelda* series even incorporated musical performance as a game mechanic in *Ocarina of Time* (Collins K. , 2008).

Procedural music in games developed in tandem with typical game music, but has never caught on as a dominant means of composition in the medium. Broadly, the types of games which ventured into generative music can be broken into **commercial** and **research** games.

Commercial Games

Procedurally creating music for a video game is not a new idea; games as early as the 1980s have attempted to incorporate dynamic music. Otocky, a Famicom game from 1987, had players generate their soundtrack on top of a bass line by mapping the direction in which they could attack to different notes. Creatures 2 (1998) influenced the feeling and texture of the background music based on the present characters and their moods (Collins K. , 2009).

The iMuse software engine, created by LucasArts in the 1990s, was an early step towards procedural music in games that allowed varying instrumentation based on gameplay intensity and smooth transitions between pre-composed materials. iMuse was a large technical achievement for procedural game music, as it surmounted the challenges of dynamically manipulating MIDI tracks. The engine was used and continually developed by LucasArts until 2013, when the studio was closed down (Phillips, 2014).

More recently, Spore (2008) brought press attention when ambient musician Brian Eno was hired to create a procedural score to the game. The music is influenced by numerous factors, and garnered significant reactions from the editor segments of the game which respond directly to player actions; overall, however, the game maintains an ambient electronic atmosphere absent of motivic development. Portal 2 (2011)

incorporates similar elements, with the different parts of the puzzle each having a specific sound that play when you interact with it (Lasser, 2013).

Despite the innovations of these games, commercial games have generally been reluctant to adopt procedural music techniques. Many games incorporate small elements of procedurality through transitory material or mixing, but only a few pioneering games have embraced generating their music in real-time.

Research Games

Commercial games are not the only test-bed for algorithmic compositions: many researchers and academics have explored the concept, and some have even applied it to games themselves.

Several researchers have attempted to capture the drama on-screen through the emotion of the scene, rather than specific agents. Chih-Fang Huang includes a numerical representation of emotion, Hevner's concept of valence and arousal on a two-axis plane, and a mapping of this representation onto specific musical concepts (Huang, 2011). This application of algorithmic music to a changing dramatic landscape is related to my own, and integrating the mappings Huang created into my system would be a potential area of expansion for the project.

The team of GhostWriter uses generated music to enhance their virtual reality game, focusing primarily on suspense and its changes across gameplay (Robertson, de Quincy, Stapleford, & Wiggins, 1998). The emphasis on player actions directing the musical score resonates with my own approach, but the limited dramatic contexts of the game make much of my research irrelevant.

A missing element from both of these games is any sort of motivic development. The research in generative music often focuses on capturing the emotional experience of the gameplay, at the expense of the descriptive power of the music and player identification with themes.

Leitmotif is used in many media to enhance the dramatic experience, from opera to film and even in games. Because of the dynamic nature of games, however, traditional reproduction of pre-composed leitmotif falls short of accurately describing the drama. Procedural musicians have explored many techniques to generate music programmatically, but few games have incorporated their work; the ones that do focus on atmosphere rather than thematic development. My project bridges this gap between the proven utility of leitmotif and the procedural nature of games.

IV. PROJECT DESIGN

Overview

The project consists of two major parts: a music generation engine, and a game that will serve as the context. These two sections are integrated with an algorithm that translates the input from the game into commands to the music engine.

Music Engine

The music engine is both the most novel aspect of the project and the most difficult to implement. The explicit use of leitmotif in a procedural engine is unprecedented, so it is largely informed by techniques derived from classical music theory.

Capabilities

In order to serve the needs of a dynamic game environment, the music engine operates in real-time; this allows it to respond to continuous shifts in the gameplay appropriately. In order to achieve the high-level thematic components of the engine, a framework for the representation of notes and sounds was created. This framework supports playback from multiple instruments simultaneously, and exposes an interface for the creation of themes.

Leitmotif Manipulations

The core functionality of the engine is its ability to manipulate leitmotifs and associate them with different in-game objects. This is done both at the levels of single motifs and multiple simultaneously, which provides a sufficient descriptive basis for the events unfolding on-screen.

Single Motif Manipulations

Common practice music theory has yielded a large variety of ways to alter a singular theme. These can vary from slight adjustments to significant distortions; because of the requirement that a leitmotif recall its associated object regardless of its transformation, some of the more extreme techniques were culled. The techniques I chose are:

Augmentation. Doubles the duration of each of the notes in the theme.

Diminution. Halves the duration of each of the notes in the theme.

Chromatic transposition. Adds a number of semitones to each note in a theme, collectively lowering or raising the pitch while maintaining the modality.

Parallel key modulation. Modifies the notes of a theme in a particular mode (Major, Minor, Dorian) to another mode while maintaining the fundamental pitch of the key.

Fragmentation. Reduces a theme to a single segment. This segment may then be used independently in other manipulations.

These techniques can be used independently as well as layered. Particularly, I chose *chromatic transposition* and *parallel key modulation* out of the various types of transpositions and modulations because, in combination, the two techniques can replicate the effects of the others. I describe some particular ways I do this in the *Additional Features* section of Chapter V.

Single-motif manipulations are simple, direct procedures that operate on themes. They serve as the basis on which the multi-motif manipulations can be built. I chose a subset of the techniques that have been developed over music history that allow for both the implementation of the multi-motif techniques and interesting combinations among themselves.

Multi-Motif Manipulations

In comparison to single motif manipulations, the realm of manipulating multiple motifs is largely undefined. Despite the wealth of music using leitmotif since its popularization by Wagner, there appear to have no been formal attempts to distill the use of multiple motifs into a set of discrete techniques. This is not to say that composers do not use similar approaches when using multiple motifs, but that they rely on intuition and logical connections with the source material.

A procedural music engine cannot rely on intuition; thus, I identified three techniques myself through the study of scores using leitmotif and reading the analyses of musicologists on specific works. The techniques I chose are by no means the only ones commonly used. Rather, these three techniques provide a breadth of expression that allow the procedural engine to use multiple motifs effectively while retaining their identity. Additionally, they are defined such that their purposes are clear.

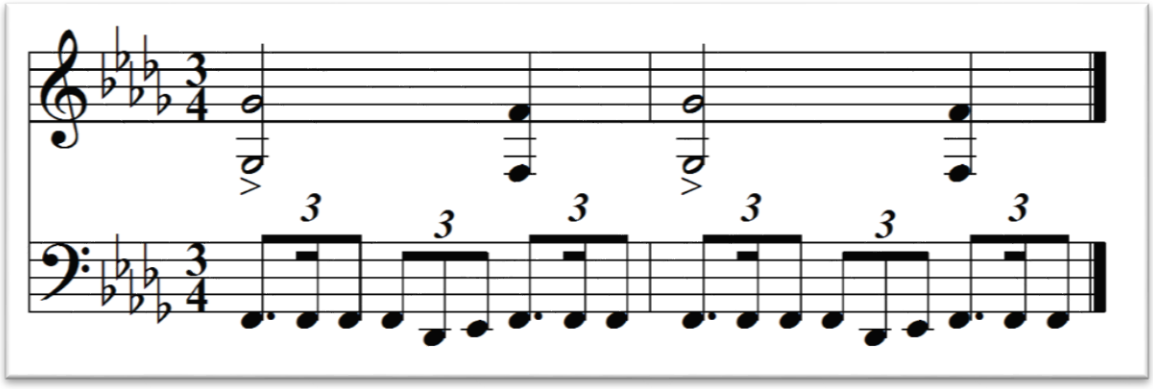
Melodic Overlap

Melodic Overlap is a technique that allows two leitmotifs to be playing simultaneously, while still being recognized for their individual identities. One theme is transposed down below the other, and typically the two themes are modulated to the same key.

Melodic Overlap has a number of qualities that make it suitable for use in my engine. It allows two themes to interact directly without modifying one another, which lets it describe situations of conflict musically; the simultaneous playback of two themes implies that both are actively participating. But the technique can also convey cooperation if the themes are harmonized consonantly. The two themes used in the manipulation retain their distinctive characteristics (intervals and instrumentation), allowing them to be recognized clearly during the technique.

Cooke analyzes an example from Wagner's *Das Rheingold*, where two themes relating to Alberich's ring are overlaid during the transition to the Dwarven realm of Nibelheim.

The motifs, one relating to servitude and the other to forging, grate against one another to represent the plight of the enslaved Dwarves forced to forge by Alberich (Cooke, 1992).

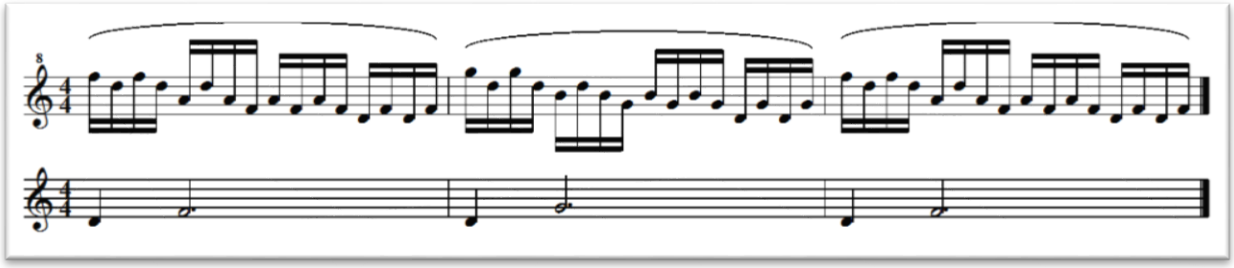


Ex. 2: "Servitude" (top line) overlaid on "Forging" (bottom line), from Wagner's Das Rheingold *(Wagner, 1873)*

This example illustrates *dissonant overlap*: when the two motifs being used are in conflict. This variation uses harmonies that contain dissonant intervals (minor 2nd, tritone, major 7th) which in turn produces an unsettling feeling. The alternative to this type of overlap is *consonant overlap*, in which the themes are modified to avoid conflicting harmonies and emphasize consonant intervals.

In order to allow for both types of harmonies without requiring an engine that follows the extensive classical rules of counterpoint (an entire project itself), I designed the technique such that one theme is in a more subdued role in comparison the other. This means that it is both lower in pitch (through octave transposition) and slower in note

values (through augmentation or diminution). Film composers have used this particular procedure before: it is heard in James Newton Howard's score for *The Last Airbender*.



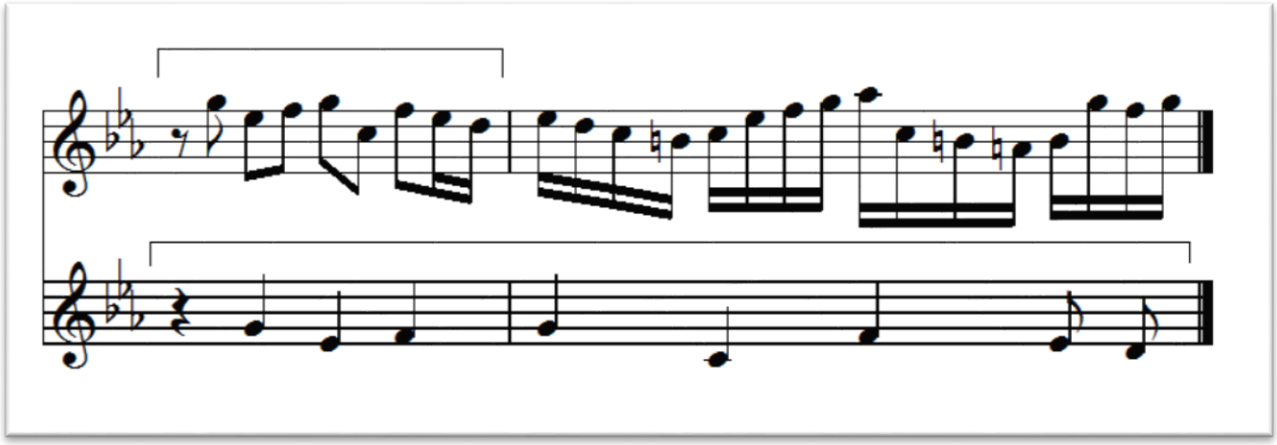
Ex. 3: "Water" (top) diminished and overlaid on "Avatar" (bottom) from The Last Airbender *(Howard, 2010)*

This excerpt shows the "Avatar" theme underlying the "Water" theme, describing musically the main character's fulfillment of his role as Avatar as waves of water destroy the enemies' forces. The arpeggiated figures of the Water theme are harmonically modified in the second measure to produce consonances with the Avatar theme's changing progression.

I designed Melodic Overlap with some connotations regardless of which harmonies are used. The technique retains the entirety of the individual leitmotifs, which is meant to suggest that the music would describe an action rather than a change in either theme. The asymmetry inherent in my design of the technique should also affects its perception. The subdued theme is often the more passive participant in the action, or alternatively may be already in a position of power. The second theme, then, fills a more active role, driving the action or attempting to usurp power from the first theme.

The harmonic context of the technique is the most apparent element, however. In dissonant overlap the two themes are in conflict, each vying for the prevalence of their own key. This could mean a direct confrontation or a more abstract interpretation, such as if the ideologies of the objects represented by the leitmotifs are in opposition. Consonant overlap implies that the two themes are united but distinct. Two characters working together towards a common goal would warrant the technique, but the technique can also be applied to different emotional contexts such as shared grief over a loss.

While the technique is found in scores that use leitmotif, it has its roots in classical counterpoint. J.S. Bach, a respected master of counterpoint, used a variant in some of his fugues that involved augmenting a theme and playing it on top of itself.



Ex. 4: A theme (top bracket) overlaid on an augmented version (bottom bracket) in BWV 871 *(Bach, 1742)*

In this example, Bach transposes the augmented theme down an octave in a similar fashion to my design of the technique. Instead of repeating the top theme, however, he begins new elaborating material while the bottom theme finishes.

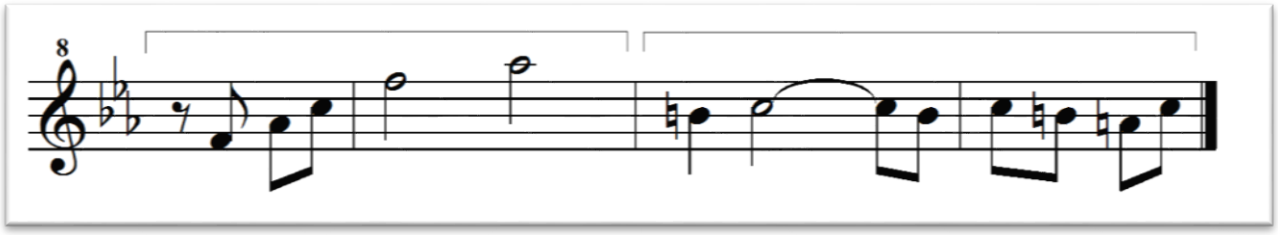
Melodic Overlap provides the engine with a multi-faceted tool to describe interactions in the game. By preserving the entirety of the melodies of the leitmotifs, the technique reminds listeners of the identities in play while allowing its harmonic context to dictate the emotional and logical connections.

Combination

Combination takes two motifs and melds them into one musical phrase. It suggests a stronger connection than *Melodic Overlap*, as the individual themes retain less of their original identities.

While Melodic Overlap is best used to accompany specific actions or events, it doesn't support transformative or permanent development. Combination, however, results in an entirely new theme that retains its influences, even in different contexts. This new theme can be carried over for broad sections of gameplay and itself manipulated. This gives the music engine a means of long-term development as well as a way to describe dynamic changes to objects.

One example of Combination is in Howard Shore's score for the Two Towers: a motif representing Gollum is combined with the "History of the Ring" theme to suggest their connection in the past (Adams, 2010).



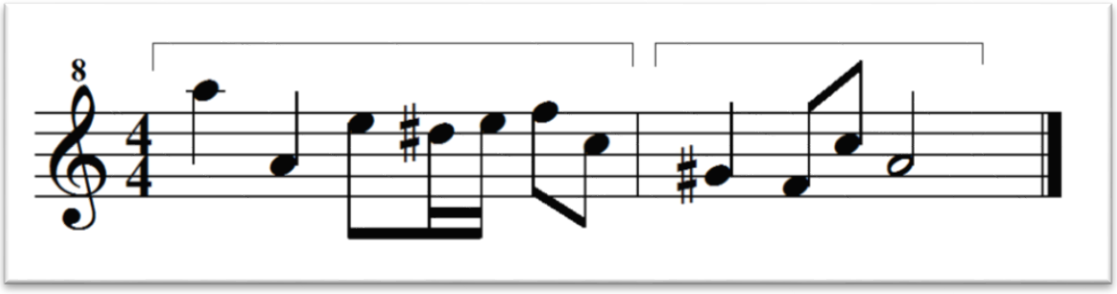
Ex. 5: "Pity of Gollum" (first) combined with "History of the Ring" (second) in The Two Towers *(Shore, 2002)*

The order in which the themes are combined matters; the effect of the combination is to suggest that the first theme is transforming into the second, which establishes the first as the primary identity. In this example, since Gollum is the primary actor in the scene where this combination takes place his theme is the first in the combination. As Gollum is having an internal struggle, the Ring tempts him to abandon and betray Frodo, reverting back to his old ways when he was controlled by the Ring (Adams, 2010).

Combination produces a theme that is markedly different from either of its origins, making it at once the most subtle technique and the most altering. There is a sense of finality and irreversibility in this resultant theme; Gollum's encounters with the Ring have left him permanently transformed (Adams, 2010).

Another use of the technique is to musically foreshadow a transformation. In *The Phantom Menace*, John Williams uses Combination to meld Anakin's theme with the

Imperial March, hinting at his later conversion into Darth Vader. In this case, the former theme will completely succumb to the later over the course of the story.



Ex. 6: "Anakin's Theme" (first) combined with "Imperial March" (second) in The Phantom Menace *(Williams, 1999)*

If the two leitmotifs are played on different instruments, the *Instrument Translation* technique will need to be used as well. In this example, the instrumentation of Anakin's theme prevails due to the early stage of his transformation.

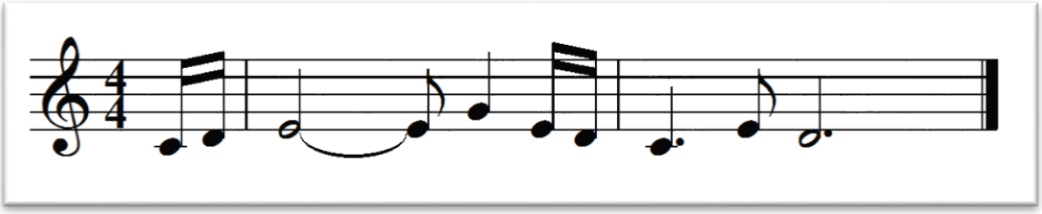
Combination is a less prevalent technique than Melodic Overlap; because it results in an entirely new theme, the influences of the original themes can be difficult to hear. When used effectively, however, it adds a satisfying level of complexity and coherence to the score. In my engine, it functions as both a means of varying thematic material and drawing connections between the leitmotifs involved.

Instrument Translation

Instrument Translation requires that each of the two leitmotifs involved have a specific instrumentation associated with that theme. The technique takes the instrumentation of one of the leitmotifs and uses it to play the other.

Although simple, Instrument Translation is vital in making interactions with multiple motifs intelligible. Combination requires the use of Instrument Translation, but it can also be used independently. I chose this technique to allow the engine to have multiple instruments; this allows listeners to be able to identify thematic development apart from melodic content.

In *The Fellowship of the Ring,* this technique appears clearly during a scene in Moria in which Gandalf imparts some wise words to Frodo; an alto flute, the instrument associated with Gandalf's theme, plays the "Hobbit's Understanding" theme, implying musically the source of this newfound knowledge (Adams, 2010).



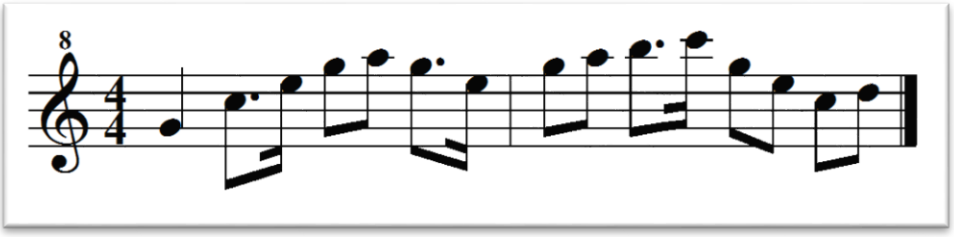
Ex. 7: Alto flute playing "Hobbit's Understanding" in The Fellowship of the Ring *(Shore, 2001)*

While Gandalf is present in this scene, Shore uses the same manipulation later in the trilogy when Frodo recalls the words alone, emphasizing the connection with Gandalf and establishing a unique variant of a leitmotif that serves its own purpose, while solely differing in instrumentation from a different theme (Adams, 2010).

The leitmotif whose instrument is used during the technique is implied to be affecting the other theme, subtly influencing it without changing its fundamental identity. In cases

where the instrumentation is more strongly associated with an object than a particular theme, however, this relationship can be reversed.

An interesting example comes from Prokofiev's *Peter and the Wolf*, one of the more famous examples of instrumentation and leitmotif. After Peter climbs the tree, he instructs the bird (identified by the flute) to distract the wolf.



Ex. 8: "Peter's Theme" played on the flute (Prokofiev, 1940)

In this case, Peter's influence on the bird manifests itself by having Peter's theme played on the flute, indicating that the bird is doing what Peter wants. Prokofiev uses this type of relationship again later in the piece, this time to indicate Peter's triumph over the wolf by having the French horns (the wolf's instrumentation) play Peter's theme.

Instrument Translation has been commonly used in orchestrated pieces to vary the texture of a theme, even before leitmotif was invented. Leitmotif with accompanying distinct orchestration adds an element of descriptive power to the technique, while leaving its proven utility intact.

Besides being a necessary technique for Combination, Instrument Translation provides a simple way for casual listeners to identify development. Additionally, allowing themes to be played on multiple instruments varies the texture of the music and maintains interest.

The music engine consists of a framework for western music theory, single-motif manipulations, and multi-motif manipulations. These allow leitmotifs to be represented, played back, and ultimately transformed in ways that are appropriate for the context of the game they operate within.

Game

The game part of the project, titled *Vikare*, serves to demonstrate the capabilities of the music engine. The key principle that drove the design of the game was to provide a limited selection of themes a variety of scenarios to interact with each other. In order to emphasize the impact of user input on the medium, I chose to make the game multiplayer; this increases the combinations of inputs drastically and allows players to be the primary force that causes drama. The players needed to be transformed in some way over the course of gameplay to allow

Ex. 9: Title screen of Vikare

the Combination technique to be used, and each of the themes needed their own unique instrument.

In the game, four players compete for the possession of a set of wings; once a player obtains them, the other players must attempt to remove them by harnessing the sun's energy and melting them, restarting the cycle. The game is inspired in broad strokes by the tale of Icarus (Vikare translates to Icarus in Etruscan), although it deviates significantly to support competitive multiplayer gameplay.

Gameplay

Gameplay is broken into three phases. The ways that players must cooperate and compete differs in each phase, as does the presence of the two external forces in the game (the wings and the sun).

Wing Token Collection

To be granted the wings, a player must collect two wing tokens. The players are all equally competing with each other at this phase. Once a player collects two tokens, they are given wings which allow them to fly around the stage.

Ex. 10: A wing token

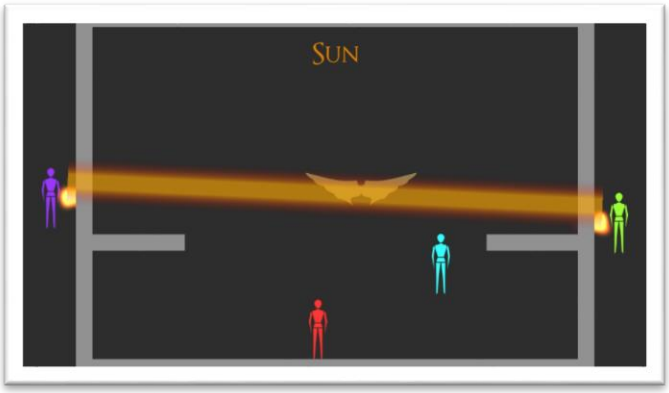
Sun Token Collection

Ex. 11: A sun token

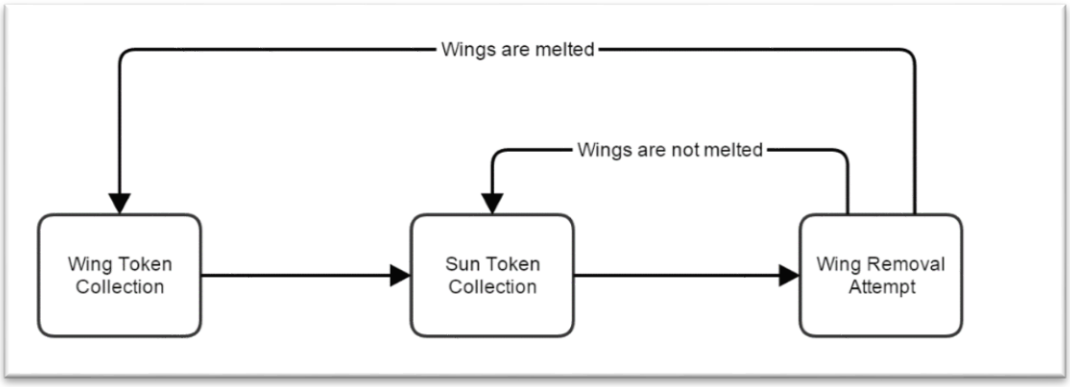
After a player has obtained the wings, the other players must collect sun tokens that will allow them to melt the wings. When a player collects two of these tokens, they are empowered and moved to the side of the screen; the phase ends when two players have done this. The player with the wings, however, may attempt to thwart the others by collecting the tokens themselves, and their enhanced mobility gives them an advantage.

Wing Removal Attempt

When two players become empowered by the sun tokens, they cooperate and use their power to attempt to melt the wings. When the players both activate their ability, a ray of heat shoots across the screen; if the winged player is caught in the ray, the wings melt and the game restarts from the first phase. If the ray misses, however, the players must collect more sun tokens and attempt again.



Ex. 12: Empowered players melt the wings



Ex. 13: Flow of gameplay phases

The gameplay phases were designed to allow a continuous flow while still providing ample time for the music to develop within each phase. The allegiances between players shift in each phase, and each cycle the groups of players involved may be different; this ensures the involvement of each of the actors in multiple ways over the course of a game.

Integration

The two independent parts of the project are integrated with some specific design choices. Each gameplay phase emphasizes one particular multi-motif manipulation.

Wing token collection. In this phase, all the players are competing equally. The engine alternates between the themes of the players and uses the single motif manipulations to create musical interest. When a player collects a token, however, their theme occurs more frequently and Instrument Translation takes the instrument of the wings' theme and applies it to that player.

Sun token collection. When the player is granted the wings, Combination blends their theme with that of the wings, creating a new identity that persists until the wings are melted. The winged player is dominant in this phase, so the combination is the primary musical material. As the other players collect sun tokens, however, their themes begin to occur, translated to the instrument of the sun theme. When a player collects two tokens and becomes empowered, their theme is combined with the sun's theme until the end of the next phase.

Wing removal attempt. The last phase pits the winged player's combination against the combinations of the sun-empowered players. Melodic Overlap is the predominant technique, both in its consonant variation (using the two empowered players' themes) and its dissonant one (using the winged player's theme and either of the empowered players).

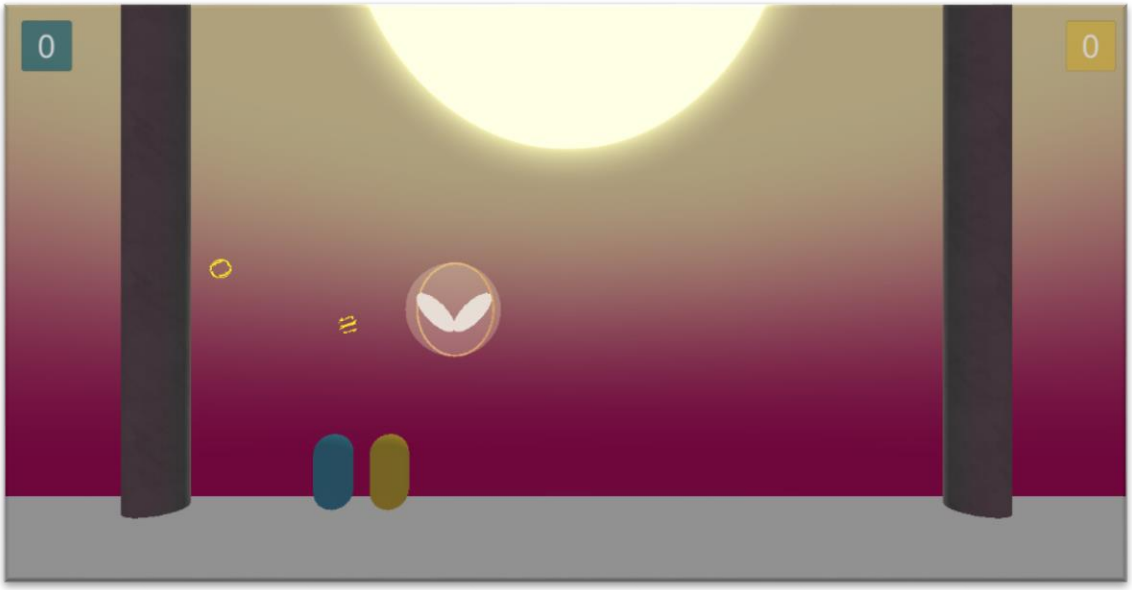
While the emphasized manipulation is the most prevalent element in each phase, the effects of the techniques layer over time to produce a significantly different score by the last phase. This provides an arc to the music of the game, which increases in complexity and sonic density through the phases. Thus, the intensity of the music corresponds with the intensity of the drama in the game, but arises naturally from the logical evolution of the leitmotifs.

This tight integration with the gameplay is impossible to achieve through traditional composition methods. Because the engine is procedural, it also has the advantage of being different on each playthrough, which avoid the repetitive nature of pre-composed music. The procedural music methods currently used in games, while well-integrated, do not provide the same identification of theme nor development over the course of a game.

V. IMPLEMENTATION

Early Prototype

The original design of the game was much simpler, but ultimately provided less situations for the themes to interact. The goals of the game remained similar — to obtain wings, or remove them from the player who did — but the wings were removed by throwing other players at the winged player.



Ex. 14: The visual style of an early prototype

This design was modified for several reasons. Firstly, the changing circumstances of player cooperation transitioned too quickly, without allowing time for the engine to develop the leitmotifs adequately. Secondly, the design was too simple to allow for enough combinations to showcase the engine. Ultimately the addition of the sun and segmentation into explicit phases would solve these problems.

This original prototype helped solidify the design of both the future game and the music engine. It illuminated the types of gameplay that the engine would not work well for, as well as the required complexity to provide sufficient musical development of the involved themes.

Technical Design

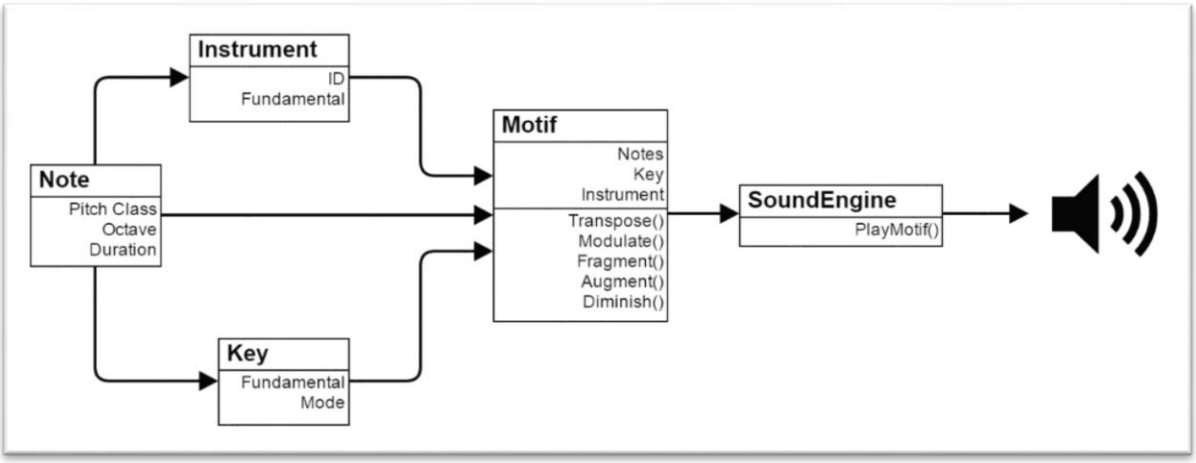
Both the game and the music engine were developed in the Unity3D game engine, using C# as the programming language. This allowed the music engine to communicate easily with the inputs in the game without significant transfer difficulties.

Music Engine

The music engine was split into two parts: an underlying musical framework and high-level motif manipulations.

Framework

The framework required that I create an object-oriented abstraction of the western classical music system. The fundamental unit of this system is the Note, which has a pitch class, duration, and octave. Keys have a fundamental Note along with a mode that defines the set of pitches that can be played while in the Key. Instruments also have a fundamental Note that allows them to sound in their natural range, as well as a number that uniquely identifies them.



Ex. 15: Classes and relations in the music framework

The highest-level and subsequently most functional element of the framework is the Motif. A Motif contains a sequence of Notes in a Key, to be played with a particular Instrument. The final piece of the framework, the SoundEngine, takes in Motifs and outputs them to the speakers. The SoundEngine uses the ID of the Motif's Instrument to determine the texture of the sound.

The framework was designed to be compatible with, but not dependent on, MIDI. An early version of the SoundEngine used recorded samples of a note and manipulated the frequency to the correct pitch, but the final version uses MIDI output. This was due to both increased audio fidelity and a broader range of instruments.

Creating a music framework was a necessary step to integrate the high-level manipulations of leitmotifs. By basing the system on traditional western music notation,

I was able to implement these higher-level techniques using the terminology they were designed with, and easily add MIDI capabilities to the project.

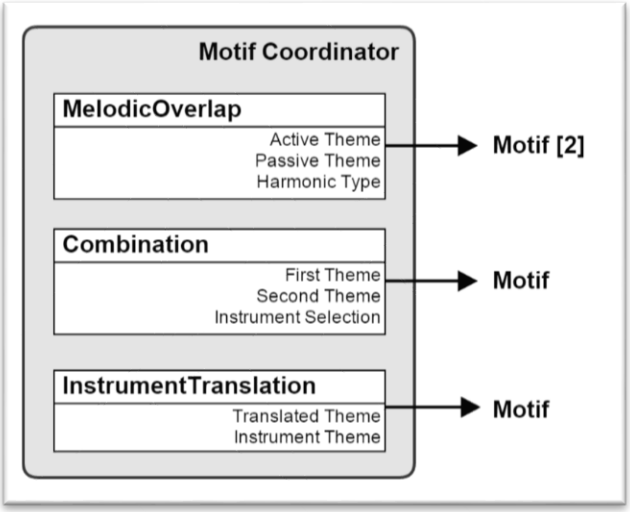
Motif Manipulation

Single Motif Manipulations

Since these types of manipulations operate on a single motif, I added them as self-modifying functions on the Motif class. The technical details of the implementations of these techniques don't merit explanation, but they lay the groundwork for the multi-motif manipulations.

Multi-Motif Manipulations

To handle these types of manipulations, an additional class was necessary. MotifCoordinator holds the three techniques, which take in two Motifs and return a new Motif (or in the case of Melodic Overlap, two) that represent the transformation.



Ex. 16: MotifCoordinator and manipulation methods

Melodic Overlap. Melodic Overlap takes in two Motifs and a Boolean that determines if the overlap is dissonant or consonant. If consonant, the two motifs are modulated to the same key; otherwise, the motifs are modulated to conflicting keys (a dissonant interval

apart). The first motif is considered the active motif and is duplicated, while the second motif is augmented and transposed down an octave. In order to prevent harmonic dissonances during consonant overlap, before the motifs are returned they are scanned. If a conflicting note is found, the first motif is modified to resolve the conflict by moving the note up or down by one scale degree.

Melodic Overlap is unique in that it returns an array with two Motifs, because the two must be playing simultaneously.

Combination. Combination takes in two motifs, and assumes that the order in which they are given is the order they are to be combined it. It also takes a Boolean parameter that determines which instrument is to be used for the combined theme. The two themes are modulated to the same key, determined at random between the original keys of the two input Motifs. It then takes the first fragment of the first theme and the last fragment of the second theme and creates a new Motif with those notes, with the appropriate Instrument based on the last parameter.

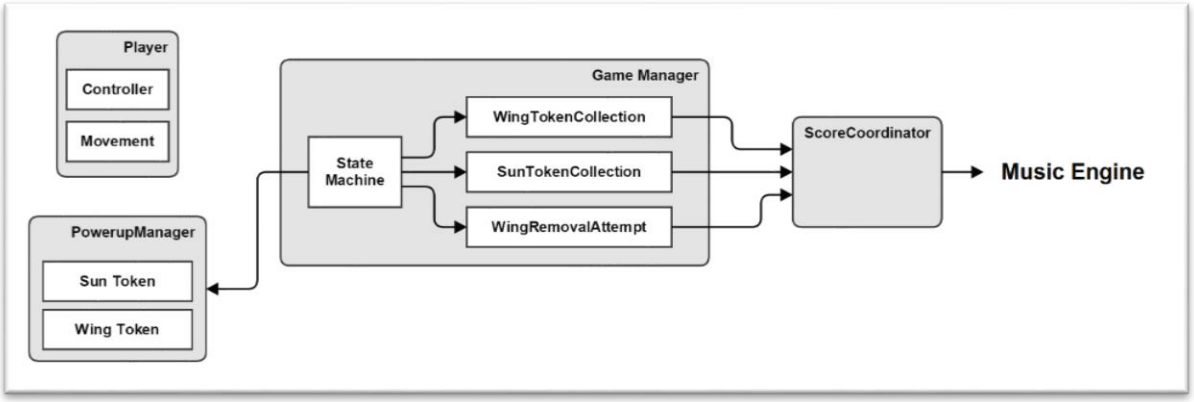
Instrument Translation. Instrument Translation is the most straight-forward of the techniques and was the easiest to implement. The function takes in two Motifs: one to be translated, and one to use the Instrument from. The first Motif is then simply assigned the Instrument from the second; the SoundEngine class in the framework

supports switching instruments dynamically, so the change in instrument is heard immediately.

The various manipulations that comprise the music engine were designed to be completely separate from any particular game implementation, such that the code could be packaged separately from any particular game and used in others.

Game

The design of the game is simple by intention, which allowed me to focus my development efforts on the music engine. Nevertheless, there are several non-trivial features of the game that warrant mention.



Ex. 17: Game class structure

Structurally, the phases of the game are handled by a state machine in GameManager, which evaluates the criteria for completion of each state and transitions between them. The individual states in this state machine inform the music engine of the current situation in the game, which is how the two parts are interconnected.

The players each have a controller and a movement script, which has different profiles depending on the state of the player (flying, empowered, etc.). The powerups are spawned in reachable areas by a PowerupManager, which is informed by the current state of the GameManager when and what it should create.

Supporting Libraries

Although the vast majority of the code in both parts projects is original, there were some outside libraries that were used to simplify some of the aspects that were unimportant to the contributions this project intends to make.

MIDI.NET. This library exposed an interface to the MIDI devices on the current computer without accessing the drivers directly. This allowed my SoundEngine to quickly transition to MIDI output.

XInput.NET. Provides a means of gathering input from Xbox controllers connected to a computer. This allowed multiple controllers to be read easily.

Unity Editor. To create the themes heard the in game, I created a simple interface for inputting notes in Unity's Editor system. This allowed me to quickly create and modify themes without directly coding them into the game.

Features from the C# programming language and Unity game engine were also helpful during development, but did not significantly influence the development process.

Through the use of a few libraries and an intentionally simple design, I was able to quickly implement a game that produces the situations that drive the music engine. The light-weight nature of the game made optimizations unnecessary, and any additional undescribed coding was for purely aesthetic purposes.

Additional Features

Through the implementation of the features I designed for the project, I discovered a few capabilities my engine required that I hadn't considered. These didn't substantially change the purpose or contributions of the engine itself, but had a considerable impact on their perception within the context of the game.

Scripted Events

In order to more cleanly transition between phases of gameplay, the music engine needed the ability to interrupt the current musical phrase and play a predetermined melody. Specifically, this feature gets used when the wings are awarded at the end of the *wing token collection* phase and when the heat ray is fired at the end of the *wing removal attempt*.

In film, moments when the music directly mirrors the action onscreen are called "synchronization points", and while having too many can make action seem cartoonish, having too few makes the music feel disconnected from the action. Adding in the ability

to have scripted events allows the game to drive when these synchronization points should occur, thus deepening the connection to the game when used in moderation.

Elaboration Algorithm

The engine described up until this point uses motifs as thematic material, but doesn't describe how this material is played. In its original form, the engine simply played back the motif in full in its original key. Even despite the various adjustments to the themes through the multi-motif manipulations, this made the music eventually become repetitive and uninteresting.

In order to alleviate this issue, I created an algorithm that *elaborates* on a given leitmotif instead of simply playing it back. There are several ways a theme can be elaborated on:

Change of mode. The theme changes mode, but not key.

Transposition. The theme moves up or down uniformly in pitch, while maintaining its original mode.

Modulation. The theme is both transposed and changes mode.

Fragmentation. A piece of the theme is elaborated on instead of the entire theme.

Diatonic sequence. The theme is modulated up by one scale degree and played back multiple times.

The elaboration algorithm chooses one of these techniques based on a probability assigned to each. The leitmotifs at each phase of the game have a set of probabilities that govern how likely each elaboration technique appears.

Once the design of the project was researched and finalized, the implementation was able to proceed smoothly. I created a prototype before I had completely cemented my design, and that allowed me to adjust certain aspects of the game that would have hindered its effectiveness as a showcase for the music engine. The music engine framework and leitmotif manipulations were the most challenging to implement, but once the game was created and able to generate input data integrating the two parts was a smooth process.

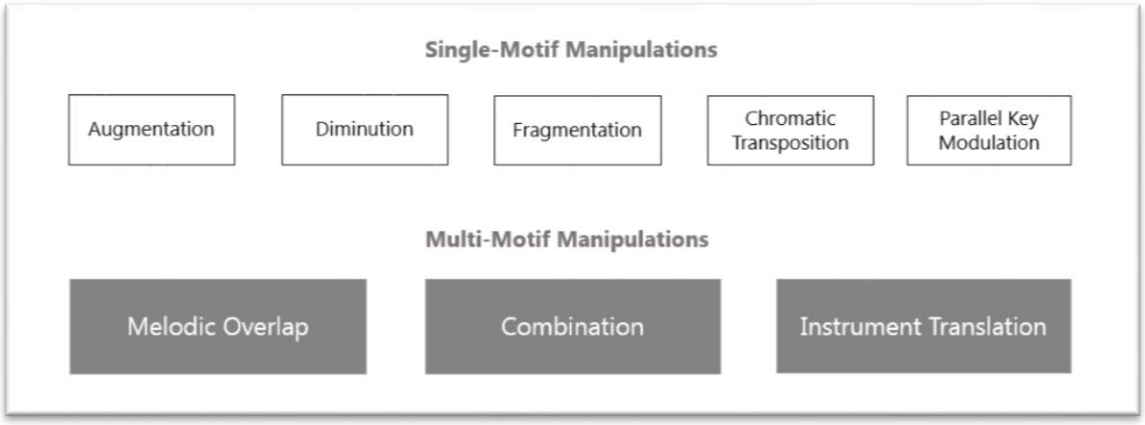
VI. CONCLUSION

Leitmotif is a powerful technique to describe and enhance the drama in other media. Film, opera, and other works have already determined effective means of using this musical tool, but its use in video games has been inconsistent. To properly use the affordances of the medium, pre-composed music must be supplanted by procedurally created, "generative" music.

Games have used generative music before, but in an attempt to capture the atmosphere of the gameplay those attempts have ignored the possibilities of classically-developed leitmotifs. By providing a direct musical connection to the objects the player interacts with on-screen, leitmotifs enhance the actions occurring and allow the music to grow dynamically.

My purpose with this project is to demonstrate that the use of leitmotifs in the context of a changing and unpredictable medium is an achievable goal. I researched the techniques that composers have used in other media to manipulate leitmotifs and distilled them into five single-motif manipulations and three multi-motif manipulations. I also researched the methods that procedural musicians have used to generate music

and defined an approach that would allow for real-time creation of music based on themes.



Ex. 18: The leitmotif manipulations implemented

I created a music engine capable of performing the leitmotifs manipulations I researched on arbitrary themes. To demonstrate this engine's capabilities, I designed and implemented a multiplayer game in which the players are frequently placed in different gameplay configurations; since each player has their own theme, this provides varied circumstances for the themes to be manipulated by the engine while maintaining a logical description of the drama in the game.

Areas for Future Work

As my goal is to demonstrate the viability of a mode of composition, I believe there are many more explorations into the area that can build upon my work to eventually reach mainstream games.

One of the most promising areas is integrating the atmospheric and textural qualities of current procedural music engines with the leitmotif-driven capabilities of my engine. This would greatly increase the audio quality as well as better maintain interest over the course of a game.

I deliberately chose a small subset of potential leitmotif manipulation techniques for implementation to effectively demonstrate them within a short game. For larger-scale games, this subset could be augmented with additional techniques that provide new ways of describing the game's drama. This would also have the impact of increasing variation within the generated score, which further alleviates listener fatigue from repetition.

One area that was out of the scope of my project was a more formal concentration on the harmonic devices of the engine. Expanding the elaboration algorithm to include chromatic harmonies, exploring different ways to make Melodic Overlap more rigorously contrapuntal, and implementing Markov models of chord progression would all add to the interest of the music, as well as bringing it closer to the traditions of film scoring (while still respecting the unique requirements of games).

Finally, a feature I had originally planned to attempt was to generate the themes for the characters themselves. This would require additional research into melody-writing rules and the creation of an algorithm to detect fragments within a theme.

REFERENCES

Adams, D. (2010). *The Music of The Lord of the Rings Films.* Van Nuys, CA: Carpentier.

Alvira, J. R. (2005). *Analysis of Bach's fugue BWV 871 in C minor (WTC, Book II).* Retrieved from Teoria: http://www.teoria.com/articulos/analysis/BWV871/

Assayag, G., Bloch, G., Chemillier, M., Cont, A., & Dubnov, S. (2006). OMax brothers: a dynamic topology of agents for improvisation learning. *ACM workshop on audio and music computing media*, (pp. 125-132).

Bach, J. S. (1742). *The Well Tempered Clavier Book II, Prelude and Fugue in C♯ minor.* Vienna: Hoffmeister & Comp.

Collins, K. (2009). An introduction to procedural music in video games. *Contemporary Music Review*, 5-15.

Collins, N. (2010). *Introduction to computer music.* Hoboken, NJ: John Wiley & Sons.

Constantini, G. (2010). *Leitmotif revisited.* Retrieved from FilmSound: http://filmsound.org/gustavo/leitmotif-revisted.htm

Cooke, D. (1992). *I Saw the World End: A Study of Wagner's Ring.* Oxford University Press.

Dean, R. T. (2009). *The Oxford Handbook of Computer Music.* Oxford University Press.

Edwards, M. (2011). Algorithmic composition: computational thinking in music. *Communications of ACM*, (pp. 58-67).

Fernández, J. D., & Vico, F. (2013). AI Methods in Algorithmic Composition: A Comprehensive Survey. *Journal of Artificial Intelligence Research*, 513-582.

Grieg, E. (1875). *Peer Gynt.* Leipzig: C.F. Peters.

Howard, J. N. (2010). *The Last Airbender (score).* Los Angelos, CA: Lakeshore Records.

Huang, C. F. (2011). A Novel Automated Way to Generate Content-based Background Music Using Algorithmic Composition. *International Journal of Sound, Music and Technology*.

Lasser, A. (2013, April 10). *4-Player: "Playing" Music In Generative Video Games.* Retrieved from The Quad: http://buquad.com/2013/04/10/4-player-playing-music-in-generative-video-games/

Lewis, G. E. (2000). Too many notes: Computers, complexity and culture in voyager. *Leonardo Music Journal*, 33-39.

Phillips, W. (2014). *A Composer's Guide to Game Music.* Cambridge, MA: MIT Press.

Prokofiev, S. (1940). *Peter and the Wolf (score).* Moscow: Muzgiz.

Robertson, J., de Quincy, A., Stapleford, T., & Wiggins, G. (1998). Real-time music generation for a virtual environment. *Workshop on AI/Alife and Entertainment.*

Shore, H. (2001). *Lord of the Rings: The Fellowship of the Ring.* Reprise.

Shore, H. (2002). *Lord of the Rings: The Two Towers.* Reprise.

Wagner, R. (1873). *Das Rheingold.* Mainz: B. Schott's Söhne.

Williams, J. (1999). *Star Wars Episode I: The Phantom Menace (score).* Sony Classical Records.