Massively Multiplayer Operas:
Interactive Systems for Collaborative Musical Narrative

David Su
Bachelor of Arts, Columbia University, 2014

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences
at the
Massachusetts Institute of Technology
June 2019

©Massachusetts Institute of Technology, 2019. All rights reserved.
Massively Multiplayer Operas:
Interactive Systems for Collaborative Musical Narrative

David Su

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning
on May 10, 2019 in partial fulfilment of the requirements for the degree of
Master of Science in Media Arts and Sciences

Abstract

Music, narrative, and social interaction have long been intertwined. The objective of this thesis is to create a platform, designed for interactive multiplayer operas, that explores the potential for technology-enabled systems to facilitate creativity through expression, the emotional affordances of musical storytelling, and the spatiotemporal boundaries of copresence. A variety of design experiments for collaborative musical narrative are implemented and evaluated. The work also introduces a real-time lyrical conversation system, with user interfaces that allow for simultaneous musical and narrative expression with a high degree of granularity. These experiences are encapsulated by an overarching lyrical multiplayer narrative opera platform. This project seeks to provide a novel means of creating and understanding multi-user, interactive music systems in which users participate in active and collaborative music-making in conjunction with narrative engagement.

Thesis Supervisor: Tod Machover
Muriel R. Cooper Professor of Music and Media
Program in Media Arts and Sciences
Massively Multiplayer Operas:
Interactive Systems for Collaborative Musical Narrative

David Su

The following person served as advisor for this thesis:

Thesis Advisor: ____________________________________________

Tod Machover
Muriel R. Cooper Professor of Music and Media
Program in Media Arts and Sciences
Massively Multiplayer Operas:
Interactive Systems for Collaborative Musical Narrative

David Su

The following person served as a reader for this thesis:

Thesis Reader:  

Zach Lieberman  
Adjunct Associate Professor of Media Arts and Sciences  
Program in Media Arts and Sciences  
Co-Founder  
School for Poetic Computation
Massively Multiplayer Operas:
Interactive Systems for Collaborative Musical Narrative

David Su

The following person served as a reader for this thesis:

Thesis Reader: Alex Rigopulos
Co-Founder and Chairman
Harmonix Music Systems
Acknowledgments

I am deeply thankful to the following people for their support and inspiration:

My advisor, Tod Machover, for his extraordinary mentorship, direction, and encouragement over the past two years.

My readers, Zach Lieberman and Alex Rigopulos, for their invaluable guidance and feedback throughout the thesis process.

My girlfriend, Dominique Star, for all her wisdom, creativity, and patience.

My parents, Richard Su and Vicki Zhang, for their unceasing faith and support.

Joseph Bonavia, for his assistance with interface design and prototyping.

The Opera of the Future group: Ben Bloomberg, Priscilla Capistrano, Sizi Chen, Charles Holbrow, Rébecca Kleinberger, Hane Lee, Nicole L’Huillier, Alexandra Rieger, Nikhil Singh, and Akito van Troyer, for a truly amazing two years with an incredible family.

Professors Rosalind Picard, Pattie Maes, Scot Osterweil, Vijay Iyer, Mark Feldmeier, Frederick Harris, and John Harbison for everything they taught me.

Friends and collaborators at the MIT Media Lab, including Tomás Vega Galvez, Adam Haar Horowitz, Shannon Johnson, Thomas Sanchez Lengeling, Nick Ouejian, Vik Parthiban, and Emily Salvador, for providing a creative environment like no other.

Melissa Plasse, Magnus Ferguson, Kate Burrows, Alexandria Petterutti, Christopher Pegram, Josh Shpak, and Laurie Lee, for being there during tough times.


Mochi Star-Su, for the moral support.
3.2.1 User Interface Design ......................................... 29
3.2.2 Audio Implementation ...................................... 31
3.2.3 Playtesting Feedback and Discussion ....................... 33
3.3 Evergreen Blues Suite ........................................... 35
  3.3.1 All the World’s a Stage ..................................... 37
  3.3.2 Evergreen Blues ............................................. 42
  3.3.3 Look at the Audience ....................................... 49
  3.3.4 Playtesting Feedback and Discussion ..................... 55

4 Towards a Lyrical Multiplayer Narrative Opera Platform 58
  4.1 Architecture Overview ....................................... 58
  4.2 The LMNOP API ............................................... 59
    4.2.1 Music ................................................... 60
    4.2.2 Dialogue ............................................... 63
    4.2.3 Data .................................................... 63
    4.2.4 Scene .................................................. 66

5 Conclusions and Future Work ..................................... 70

Bibliography .......................................................... 74
1 Introduction

Almost every culture in the world has a tradition of multiple people creating music together, from gamelan to gagaku to Gregorian chant. Most cultures also have traditions of musical storytelling through the combination of music and words, such as that of the yuefu (乐府) in China, the griot in West Africa, and the bush ballad in Australia. Perhaps the most all-encompassing musical narrative form in Western culture is the opera. When people attend an opera, they engage in an experience involving relationships between the audience members and performers, between the performance and musical composition, and between the musical elements and underlying narrative. The joy of interacting with other humans through musical creation as well as the emotional resonance of musical storytelling both appear to be universal.

While the networking capabilities of the Internet have allowed us to participate in social interactions on a whole new scale, advances in recording and streaming technology have not achieved the same results with regards to our collective engagement with music. One of the most common means of experiencing music today involving a single person listening to static, linear sound files, mediated by a streaming service’s algorithm that caters towards disparate songs in an effort to maintain short-term attention. In essence, the predominant music experience in the 21st century is isolated, passive, and ephemeral.

This thesis aims to rectify such a situation by providing a social, interactive, and persistent musical experience through the creation of a platform for massively multiplayer operas; that is, interactive musical narrative experiences that retain statefulness between scenes, that are playable by many simultaneous participants, and that foster collaboration and expression.

The remainder of this thesis is organized as follows. Chapter 2, Background,
presents an overview of existing artwork and literature in the realms of interactive operas, playfully collaborative music-making, and choice-based narrative systems. The chapter then presents design principles of expressivity and granularity in the context of musical conversations as well as persistence and consistency across scenes on a narrative level. Chapter 3, Design Experiments, discusses the creation and playtesting of three multiplayer interactive operatic experiences: Seasons Change Together (an interactive song construction experience), a real-time lyrical conversation system, and the Evergreen Blues Suite (a suite of three interactive songs). Chapter 4, Towards a Lyrical Multiplayer Narrative Opera Platform, describes the architecture and implementation of a platform allowing for the creation of such multiplayer opera experiences. Finally, chapter 5, Conclusions and Future Work, synthesizes the conclusions of this thesis in addition to presenting potential future research directions.

It is the goal that this thesis will push the boundaries of the operatic form as well as open up new possibilities for the creation and enjoyment of music, narrative, and social interaction.
2 Background

In order to create a platform for collaborative interactive musical narrative, the choices and actions that players make should be musically expressive, narratively meaningful, and social at heart. This chapter reviews related work within the field in addition to design principles that inform the thesis project’s development.

2.1 Related Work

A wide range of artwork and literature exists along the axes of interactive operas, playfully collaborative music-making, and choice-based narrative systems. While it is impossible to cover all such works, a selective review of relevant projects is useful in contextualizing the experiences and platforms developed in this thesis.

2.1.1 Interactive Operas

An opera generally refers to a “drama set to music to be sung with instrumental accompaniment by singers usually in costume”, with music playing an integral role in the experience [34]. A variety of approaches have been taken to introduce interactivity to the operatic experience.

Tod Machover’s Brain Opera [28] involves a pre-performance audience participation component, in which listeners explore and create sounds using custom instruments. The audience-generated material, as well as material from remote participants through the Internet, is then incorporated into the opera’s performance. Importantly, the Brain Opera does not have a traditional narrative; the libretto is derived from interviews between Machover and Marvin Minsky.
Nevertheless, as a “psychological journey with voices” [42], it undoubtedly falls within the operatic form while introducing innovations on the interactive front.

Björk’s *Biophilia* [5] app suite, developed in collaboration with Second Wind Apps and RelativeWave, allows players to experience the album’s songs interactively via touch-based musical interfaces that control instruments in addition to graphically-based gameplay. For example, the interface for the song *Thunderbolt* involves a multi-touch virtual lightning bolt that controls the bassline; moving one’s fingers horizontally controls the pitch range of the bassline, while moving one’s fingers vertically controls the bassline’s rhythmic density.

Another example of the song as instrument comes in the form of David Kanaga’s *Oikospiel Book I* [20], which fuses game design and musical interaction in the form of a surreal “dog opera”. Throughout the game, Kanaga alters both the musical interaction mappings and the game mechanics themselves. For example, the opening of the *Orpheé Salt and Scab* scene presents itself much like an interactive book. Moving the mouse triggers various high-pitched squeaks while rotating various icons on the virtual page, while clicking either the left or right mouse button navigates to the next page in addition to triggering different squeak samples. The scene then immediately transitions to a 3D environment, with the player character presented as a chicken via a third-person perspective. Mouse movement now controls the camera’s rotation around the player character, while clicking the right mouse button zooms the camera out; neither action is tied to any musical gesture at all. In this way Kanaga has completely changed the feel of the game on an interactive level; in an interview with Pierre Depaz he describes game feel as a fundamentally musical concept in that “it is the game trying to become a musical instrument (whether sonic or not)” [10].

Similarly, Cardboard Computer’s *Kentucky Route Zero* [7] utilizes formal and spatial elements of theatrical set design in its creation of an “illusionistic
pictorial landscape” [21]. While not strictly in the operatic tradition, *Kentucky Route Zero*’s consideration of scenography in conjunction with multimedia narrative immersion, with a strong emphasis on music and sound design, serves as a powerful example of one form an interactive opera might take.

While these works present innovations in turning the operatic experience into an interactive one, they are, with the exception of the *Brain Opera*, single-player experiences. Because this thesis is concerned with *multiplayer* musical narrative, it is necessary to examine works that emphasize music in a collaborative setting, including experiences that exist outside the realm of narrative.

### 2.1.2 Playfully Collaborative Music Making

As music-making so often involves real-time collaboration through play (as in the case of an ensemble performing together), it is natural that technological approaches to playful, collaborative music creation have arisen. These differ on a variety of axes, including reliance on loops, choice of user interface, and explicitness of collaboration.

Golan Levin’s *Scrapple* [24] is an installation that turns a table surface into a sound-producing "active score”, akin to a sequencer, which loops across a three-second long musical phrase. *Scrapple* employs playful audiovisual forms such as the use of long curve shapes to create melodies as well as cloth and wind-up toys to create rhythms; by moving or reshaping objects, players can change the melodic, rhythmic, and timbral aspects of the musical phrase.

As an alternative to loop-based systems, Dinahmoe’s *Plink* [11] provides a multiplayer audiovisual interface that is designed for remote play via the Internet. Each player navigates their computer cursor to an on-screen row that corresponds to a discrete pitch; by pressing their mouse down, the player is able
to trigger continuous instances of a sound at that particular pitch. Zach Lieberman’s *Play the World* [25] maps notes on a piano keyboard to corresponding fragments of radio broadcasts; the fragments are analyzed for pitch content and updated in real-time. Like *Plink*, sounds are triggered by key presses in *Play the World*. On the other hand, while *Play the World* can function effectively as a single-player experience, *Plink* explicitly relies on multiplayer collaboration as a central tenet of its interaction scheme.

*DropMix* [15], a music game developed by Harmonix and Hasbro, involves music-making on the level of song verses rather than individual notes. Players place cards down on a board; cards either add or replace prerecorded instrumental and vocal parts, as well as affecting the key, harmonic progression, and tempo of the current mix. Like *Scrapple*, *DropMix* does employ looping, though its loops span multiple song sections so as to make musical repetition less noticeable. By the time the loop restarts, it is likely that most of the cards (and hence the musical layers) have already been replaced by new ones.

A common thread throughout these examples of playfully collaborative music-making is that participants make music *together*. Questions of presence and interplay thus arise: should one player’s actions affect another’s, and if so, how? For example, *Plink* demonstrates no direct consequence between players’ actions; one player triggering a pitch has no bearing on the possibility for another player to trigger theirs. On the other hand, in *DropMix*, a single card put down by a player can change the entire harmonic and rhythmic fabric of the music, which in turn changes the possibilities for what occurs when other players add or remove cards.
2.1.3 Choice-Based Narrative Systems

At the heart of interactive narrative, and indeed interactivity in general, is player choice and decision-making.

Narrative interaction design is often theorized in terms of a dichotomy between emergent narrative, which involves storytelling through interactions of multiple systems [1], and structured narrative, which achieves storytelling primarily through pre-authored content. In practice, many systems incorporate elements of both, and the balance between a system’s flexibility and the potential to create authored dramatic arcs depends largely on the intended experience [27]. One promising project on this front is Ceptre [30], a programming language by Chris Martens that uses linear logic to design interactions and generative narratives.

The most common paradigm for multiplayer interactive narrative is that of the multi-user dungeon (MUD); the first such experience, named MUD1, was created by Richard Bartle in 1978 [2]. Players inhabit a persistent virtual world and are able to interact with one another by typing commands into a parser. Andrew Plotkin’s Seltani [37] is a MUD-like platform for multiplayer choice-based interactive fiction that employs hypertext elements in addition to parser-based input. Authors are able to craft their own worlds for players to explore and interact in; an example of this is Emily Short’s Aspel [44]. Chris Martens and Rogerio Cardona-Rivera describe a plot-centric approach for procedural narrative generation that builds upon open-ended simulation by allowing for the authoring of partial stories that are then curated by a story mediator [31]. The MuSE storytelling engine, spearheaded by Mark Riedl, enables multiplayer interactive narratives by modeling a story world on both individual and group levels [40].
On a more localized scale, Night School Studio’s Oxenfree [32] provides an innovative example of narrative interaction in a real-time setting, albeit within a single-player rather than multiplayer paradigm. In the horror adventure game, the player is able to interrupt other characters’ dialogue with that of their own. As such, the question of when a player makes a choice, as opposed to simply which choice they make, becomes of utmost importance, bringing conversational rhythm into the decision-making process.

2.2 Design Principles

This thesis project extends the concept of conversation systems (also known as dialogue systems) to lyrical dialogue, bringing narrative decision-making into the realm of music. The concepts of expressivity and granularity can be thus explored on both musical and narrative levels simultaneously. Similarly, for long-term meaning to emerge, a musical narrative must retain some sense of consistency and persistence across individual scenes.

2.2.1 Conversations, Expressivity, and Granularity

The granularity of control available to the player has implications for musical expressivity [23, p. 46] as well as for narrative consequences. One hypothesis of this thesis work is that a conversation system focused on player control of lyrics has the potential to afford alignment of granularity in terms of action and consequence on both a musical and narrative level.

In the realm of music perception, granularity on a temporal scale is often defined in terms of the tatum, that is, “the lowest level of the metric musical hierarchy” [4], in addition to the tactus and measure level, which define the tempo of a piece and musical measures respectively [22]. In the context of multi-user
rhythmic copresence, temporal granularity affects the possibilities for musically expressive behavior along axes such as entrainment [8] and microtiming [19]. As such, musical conversations are bounded by the degree to which participants have granular control over their interactions and gestures.

Almost all interactive music experiences encompass a variety of levels of granularity. For example, the instrument interfaces in *Biophilia* allow for a high degree of musical granularity and expression due to the real-time nature of the instruments themselves, while the actual progression of the underlying song material involves a relatively low degree of musical expression on the audience's part. Increased granularity does not necessarily equate to a more compelling experience; it is up to the creator of the experience to determine the most appropriate degrees of granularity with regards to musical control.

With regards to narrative granularity, Jon Ingold highlights the difference between “binary” (not granular at all) and “analogue” (highly granular) controls in gameplay, noting that choice-based games often veer more towards the former than the latter [16]. In such cases, players are only allowed to make meaningful choices at certain specified moments, and often the action and the consequence are somewhat misaligned. For example, decisions that involve a great deal of emotional investment, such as choosing which one of two characters the player should save from drowning, are often relegated to a single button press. As with the case of musical control, a range of narrative granularity can exist within the same experience. *Oxenfree* provides an illustrative example of generally analogue gameplay, with more binary decision-making at certain key moments of the game.
2.2.2 Scenes, Consistency, and Persistence

While a lyrical conversation system can provide a narrative framework within any given scene, in order to provide a unified, meaningful experience, elements between scenes should be *consistent*. This means that any musical interactions that take place in one scene should also be able to take place within another. This is akin to the structure of many role-playing games, in which individual quests can be undertaken in a non-linear fashion, with all the quests co-existing within the same virtual world, and involving the same set of gestures and actions.

Similarly, what goes on in one scene should have consequences for players’ future actions in other scenes. In other words, the characters and environment between scenes should be *persistent*. To achieve this, we must keep track of state. A method similar to Elan Ruskin’s rule databases for contextual dialogue and game logic [41] can be implemented for such a world state, as it also allows for the granular interactions described in chapter 2.2.1.

The requirements of consistency and persistence map to Chris Martens’ formulation of the world as a state space with mechanics [30]. A persistent state space with consistent interaction mechanics should facilitate a compelling experience for players; in this case, the mechanics are the real-time lyrical conversation system itself.

By designing for consistency and persistence across scenes, the thesis project becomes not only a single experience but a platform, open for other artists to create experiences with. Just as Andrew Plotkin’s *Seltani* enabled Emily Short’s *Aspel*, so might this platform enable the authoring of a whole multitude of interactive multiplayer operas.
2.2.3 Ethical Considerations

Several potential ethical issues arise concerning the creation and adoption of a platform for multiplayer interactive operas.

With the possibility of multi-user interaction comes the danger of harmful and malicious player behavior. Because the verbal content is bounded by the available lyrical material, the potential for inappropriate verbal content is limited. Furthermore, the fundamentally musical and collaborative design of the interactions prevents the possibility for violent behavior that might otherwise exist in, say, a competitive first-person shooter. Nevertheless, even a gesture such as interrupting another player’s musical dialogue could provide a means for bullying and toxic behavior, and thus such interactions should be carefully designed, with measures in place to stop players if necessary.

Another consideration with regards to interactive experiences is whether the underlying system can deliver on the promise of choice and interactivity itself. When commenting on player choice in Quantic Dream’s game Beyond: Two Souls [39], Lucy O’Brien notes that “it is unwise to build up our expectations of being able to choose our actions and then take that away from us where it feels most important” [33]. Whether or not one agrees with O’Brien’s assessment of Beyond: Two Souls, her statement provides a strong case for the creator’s responsibility to give players true freedom of expression, as opposed to simply manipulating the player into illusions of decision-making, if choice and interactivity are what have been promised.

On a wider level, all technology is imbued with the biases of its developers, and this thesis project is no exception. Such biases encourage certain means of interacting with the technology while discouraging others, and thus this project has the potential to limit creative expression. Player feedback, and playtesting
in general, can help minimize this problem.
3 Design Experiments

Three design experiments, in the form of interactive musical narrative experiences, were carried out for this thesis project: *Seasons Change Together* (an interactive song construction experience), a real-time lyrical conversation system, and the *Evergreen Blues Suite* (a suite of three interactive songs). Each experience implemented for this thesis was made available for playtesters within the MIT Media Lab. Subjective feedback was elicited through observation of gameplay in addition to spoken conversations both during and after gameplay.

As artistic works, an important measure of success is how compelling players find the experiences. In addition, the success of the musical interfaces can be evaluated in terms of aforementioned criteria such as intuitiveness, expressivity, and creativity. As playful experiences that draw inspiration from the medium of games, the works should be fun. Finally, given the goal of allowing people to engage with musical narrative in a collaborative, interactive way, the resulting experiences should be emotionally meaningful for players.

3.1 Seasons Change Together

*Seasons Change Together* is a first attempt at creating a slice of a multiplayer opera in the form of a single interactive song. Players are able to control the song’s content on three axes: words, textures, and patterns.

The experience is presented as a local multiplayer game, with three iPads each running one of the “words”, “textures”, and “pattern” interfaces, as shown in Figure 1. Players are able to freely move between the iPad interfaces, while a separate laptop provides synchronized audio playback based on input data from each device. Because networking between devices is all accomplished via the
Internet, it is actually possible for *Seasons Change Together* to run remotely as well as locally. However, a remote multiplayer version of the experience would necessitate design changes to the interfaces.

The *Seasons Change Together* system is built entirely using JavaScript, with Tone.js [29] powering the underlying audio engine, Node.js [9] handling multiplayer networking, and Ink [17] being used for narrative scripting.

![Figure 1: Photo of *Seasons Change Together*’s three-iPad interfaces.](image)

### 3.1.1 Words

The “words” interface is a text-based experience, with players able to choose lyrics from a set of lines, similar to the hypertext options in *Seltani*. Figure 2 shows a screenshot of the “words” interface. When a lyric is pressed, the audio engine adds the choice to a queue of choice actions, and waits until the appropriate time in the current musical phrase to play back the requested vocal sample. For *Seasons Change Together*, each phrase is two measures long, and will loop until the player makes a choice.
Figure 2: Screenshot of the “words” interface for *Seasons Change Together*.

Narratively, the lyrics utilize the branching paths design pattern as popularized by choose-your-own-adventure gamebooks such as Edward Packard’s *Sugarcane Island* [35] as well as Jorge Luis Borges’ *The Garden of Forking Paths* [6]. However, to prevent combinatorial overgrowth of paths, the lyrics will converge, or gather, at certain points, most notably in the choruses.

Figure 3 illustrates the branching paths made available given the choices “Summer warmth is slow” followed by “See the sunbeams show”, while figure 4 depicts the corresponding Ink code for the path. Each asterisk represents an additional layer of branching; for example, lines 7, 16, and 25 of figure 4 correspond to the second row of figure 3. Line 5 sets the `season` variable’s value to “summer”, which is then propagated across all the different devices. The `#SECTION` tags, such as in lines 3 and 8, inform the underlying music system of the chosen lyric to trigger as well as the corresponding musical section to transition to.

Following the second chorus, players are then presented with options of mu-
Figure 3: Example decision tree for the “Summer warmth is slow” path in *Seasons Change Together*. For clarity, the path options for unselected choices are omitted.

Musical notation instead of text. Players still click and choose one of the phrases shown, but instead of the phrase being vocalized, the melody will be played on a ukulele. This gesture is a slight variation of the mechanic that players will have gotten accustomed to during the first two verses and choruses; it represents an effort to introduce variation as well as allow for player expressivity on the level of melody in addition to narrative. Figure 5 shows the first set of melodic choices in the musical notation section of the piece.

The lyrical content of *Seasons Change Together* is derived from *Seasons* [47], an interactive song created in collaboration with singer-songwriter-engineer Dominique Star, who also provides vocals for the experience.
* Summer warmth is slow

    # DELAY: 5000.0
    # SECTION: verse1_line1_summer

- season = "summer"

** See the sunbeams show
    # SECTION: verse1_line2_sunbeams

*** It will pass in a while
    # SECTION: prechorus1_pass

*** It may last for a while
    # SECTION: prechorus1_last

** Hear the evening’s crow
    # SECTION: verse1_line2_crow

*** Not a care in the world
    # SECTION: prechorus1_care

*** Not a clue in the world
    # SECTION: prechorus1_clue

** Watch through your window
    # SECTION: verse1_line2_window

*** It will pass in a while
    # SECTION: prechorus1_pass

*** It may last for a while
    # SECTION: prechorus1_last
3.1.2 Textures

Using the “textures” interface, players are able to orchestrate the song on-the-fly by activating and deactivating individual instruments. In this case, the available instruments are ukulele, vibraphone, and cello. The adaptive adjustment of instrumentation fits in with the game music technique of vertical re-orchestration [36], in which the overall mix of audio stems is constantly changing in reaction to player input. Figure 6 shows the “textures” interface; in this example, the ukulele and cello are active, while the vibraphone is muted.

3.1.3 Patterns

The “patterns” interface, shown in Figure 7, allows players to construct the underlying percussion pattern that plays during the song. Such construction is achieved using a traditional sequencer layout, involving a 2-dimensional grid of cells (also known as steps) in which the x-axis represents time (in beats), and...
Figure 6: Screenshot of the “textures” interface for *Seasons Change Together*.

![Textures Interface](image)

Figure 7: Screenshot of the “patterns” interface for *Seasons Change Together*, along with a musical notation representation of the currently displayed pattern.

![Patterns Interface](image)

The y-axis represents instruments or voices.

In the case of *Seasons Change Together*, the grid is divided into twelve steps,
representing a 12/8 meter. Four percussive voices are available, corresponding roughly to a kick-snare-hihat-auxiliary grouping. Figure 7 illustrates an example of a rhythmic phrase as it appears in the “patterns” interface as well as in musical notation representation.

The specific sounds that each percussive voice corresponds to depends on the season that players choose in the “words” interface; for example, choosing “Springtime softly glows” results in sounds of rock and gravel, while choosing “Winter so it goes” results in sounds of snow and ice. Such interplay between words, textures, and patterns is a potentially compelling application of collaborative musical narrative; having one player’s choice of words affect another player’s choice of textures reinforces the fact that players are constructing a shared musical fabric together.

3.1.4 Playtesting Feedback and Discussion

Playtesters generally enjoyed the Seasons Change Together experience, and appreciated being able to move freely between the different iPad interfaces, which fostered a sense of collaboration between individuals. Several players described feeling a sense of accomplishment at constructing a song, noting that they felt they were able to make meaningful lyrical decisions within the context of the experience.

One limitation of the experience is the relative absence of interplay between the different interfaces; the only instance of such interplay in Seasons Change Together is the initial season declaration determining the percussive samples used for the “patterns” interface. As such, the collaborative nature of the experience is somewhat stilted; while players share control over a common musical fabric, what one player does has no effect on any other player’s actions, reducing any sense of copresent consequence. One potential solution to such a problem
is to make the interfaces themselves more interrelated; for example, instead of separate “words”, “textures”, and “patterns” interfaces, perhaps each interface could allow control over a combination of all aspects of the song.

The “patterns” interface itself also proved confusing to non-musicians, who were sometimes unable to make the conceptual connection between the sequencer pattern and the looping percussive phrase. This issue speaks to the implicit assumptions that are often made when designing for a general audience, and to the importance of playtesting and diverse perspectives in the design process. A potential solution to the lack of cognitive connection between the visual and musical aspects of the “patterns” interface could be to trigger playback of a sample upon touching any cell in the sequencer, thus providing real-time auditory feedback for the player’s actions (as opposed to having the player wait until the next iteration of the loop to hear the changes they have made). Indeed, perhaps moving away from a loop-based paradigm, as established by works such as Scrapple and DropMix, and more towards the paradigms presented in Plink and Play the World, could open up further possibilities for expressive control of musical content.

A related failure of the experience in terms of design goals is that of lyrical granularity in the “words” interface. Because lyrics are chosen and executed on a line-by-line basis, it is necessary for the music system to introduce a delay of at least one bar before the next lyric decision in order to maintain the musical form. In other words, the experience only allows interaction on the measure rather than tatum level. As a result, the player is forced to wait, unable to perform any action for the time being; such inaction shifts the entire experience from an interactive one to a passive one, however briefly. A proposed solution to Seasons Change Together’s lack of lyrical granularity is the real-time lyrical conversation system presented in the following section.
3.2 A Real-Time Lyrical Conversation System

In order to create a platform for collaborative interactive musical narrative, the choices and actions that players make should be musically expressive, narratively meaningful, and social at heart. One way to achieve such a design goal is the development of a real-time lyrical conversation system as the primary form of gameplay and interaction.

3.2.1 User Interface Design

Because the project aims to accommodate a range of musical experience amongst players, the interfaces by which players interact should ideally be “instantly knowable” and “indefinitely masterable” [23, p. 54]. As such, the user interface by which players engage in lyrical conversation consists solely of button-based gestures; each player has two directional buttons (up and down) as well as a “trigger” button. Figure 8 depicts an implementation of a two-player interface that demonstrates such gameplay.

A lyric’s pitch information is conveyed through its vertical position and normalized to a given key and mode. Since the system operates in real-time, a given lyric’s rhythmic information is determined by the player interaction gesture itself. Figure 9 illustrates how choice and decision-making is implemented with such an interface; both players occupy the same screen, with their virtual characters being offset horizontally. Each player controls a white orb, whose vertical position represents its pitch. In this example, the lyrics cycle over the phrase “all the world’s a stage”.

The proposed interaction scheme aligns granularity of action and consequence by allowing the player to make choices on a syllable-by-syllable basis, with individual notes being performed in real-time. Such a scheme makes possi-
Figure 8: Screenshot of the real-time lyrical conversation system.

Figure 9: A step-by-step example of lyric choices being activated by both players at different pitches; the player on the right has chosen a pitch higher than that of the left.

ble a wide range of expressive possibilities, as the player has fine control of both rhythm and melody. This in turn allows players to make more granular narrative decisions, giving them multiple opportunities to steer plot and character in one direction or the other, and ultimately increasing emotional investment [16].
3.2.2 Audio Implementation

The development of the real-time lyrical conversation system essentially entails the creation of a lyric-based vocal sampler. Vocal lines are prerecorded for purposes of quality and immersion, with additional manipulations and processing that take place in real-time as the experience is played through. Recorded phrases are on the level of individual notes as opposed to the entire phrases found in *Seasons Change Together*; the *Evergreen Blues Suite* would see a blend of single notes and two-to-three-note fragments.

![Figure 10: Mapping between individual vocal samples and requested MIDI notes for the lyric “stage” in the real-time lyrical conversation system.](image)

In the implemented example, it is necessary to record the same syllable or fragments at multiple pitches, with pitch-shifting performed to fill in the gaps; keeping the range of these gaps within three semitones tends to yield a successful balance of preventing noticeable artifacts while minimizing asset sizes, as outlined in [3]. Figure 10 shows the MIDI mapping between samples and notes; with 12 samples, each 3 semitones apart, a total range of 3 octaves can be accounted for. Such an implementation applied across the 5 words “All the world’s a stage” yields a total of 60 individual vocal samples.

Dynamic looping, using simple attack-decay-sustain-release (ADSR) envelopes
Figure 11: Envelope markers for “all” note A1 in the real-time lyrical conversation system.

Figure 12: “Loop Start” transition timeline for “all” note A1 in the real-time lyrical conversation system.

Figure 13: “Tail” transition timeline for “all” note A1 in the real-time lyrical conversation system.
[49], is also employed in order to allow for variable timing in between player note choices; a given note may need to be stretched out to account for a delay in action, or contracted to account for actions (such as interruptions) that take place before sample playback is complete, even in the case of player-contributed vocals.

Figure 11 shows the dynamic looping as implemented for “all” note A1 within the FMOD Studio environment [12]. The grey “Loop Start” and “Tail” flags indicate time positions that define when the sustain and release of the sample respectively begin; the green regions define the periods in which the audio engine can transition to either of those markers. When a note is sustained, the engine will continuously loop through the “Loop Start” marker and the green region around beat 2.4; figure 12 illustrates the crossfade that occurs within such a transition timeline. Similarly, when the note is released, the audio engine will immediately execute the transition timeline shown in figure 13, crossfading into the “Tail” portion of the sample. Individual ADSR envelopes are defined for every vocal sample in order to account for timing and performance differences.

The real-time lyrical conversation system provides a new means of musical interaction and narrative gameplay, offering a set of design possibilities and interaction mechanics by which musicians, game designers, and interactive fiction writers alike can create with and expand upon. A variant of the real-time lyrical conversation system was utilized to develop the Evergreen Blues Suite, a multi-scene interactive narrative experience.

3.2.3 Playtesting Feedback and Discussion

Because the prototype of the real-time lyrical conversation system served mostly to demonstrate a given interaction scheme, the lyric possibilities were limited to “All the world’s a stage”, making it difficult for players to draw any strong
narrative meaning from the experience. Nevertheless, some players mentioned that being able to control *when* an individual syllable was played did in fact feel meaningful, even within such a limited scope of lyrics.

The visual alignment of two player avatars on the same screen also assisted with players’ sense of copresence and collaboration. Playtesters would often attempt to either synchronize or alternate their trigger gestures, and a great deal of expressive play emerged from experimentation with different relative pitch positions between the two players.

Granular interactions also allow for unique interactions between players. For example, since conversational actions are on the scale of syllables, players can potentially interrupt one another mid-phrase or echo one another through harmony. Given a larger set of lyrical dialogue possibilities, such interruptions can be implemented through means similar to those used by Naughty Dog for their in-game conversation systems [13, p. 997], with additional music-specific features such as stingers [36, p. 339] for coherence and interest.

The most commonly described limitation of the experience was that while the pitch control felt musically expressive, it was not demonstrably tied to any musical *meaning*, and in fact that more pre-determined melodies could have more of an emotional effect. While the difference in perceived emotional impact with regards to lyrical choice versus musical choice is beyond the scope of this thesis, it may well provide the basis for more compelling interactive musical narrative experiences as a whole.

Playtesters mentioned an uncanny valley effect when hearing back the vocal samples; the voices sounded *almost*, but not quite, as if they were realistically being sung, resulting in an unsettling feeling on the part of the player. This effect could be due to a combination of factors, from the recorded performances to the audio implementation itself. Additional recordings per note could mitigate
the effect, as could the addition of audio effects such as delay. With regards to actual note expansion, spectral “freezing” using a phase vocoder [38] as well as time-stretching may yield more perceptually smooth sonic results than the sole use of dynamic looping as currently implemented.

3.3 Evergreen Blues Suite

The *Evergreen Blues Suite* is a collection of three interactive songs that together provide a collaborative musical narrative experience in the form of a multiplayer operatic game. Despite somewhat minimal development of plot and characterization, like all operas the game uses music (and in particular, the voice) as an integral component in order to bring the audience on an emotional, dramatic journey across multiple scenes. Of the experiences created as part of this thesis work, this suite of interconnected songs represents the most significant step towards the intended thesis goals.

Two players simultaneously control the construction and direction of a piece of music through the use of a real-time lyrical conversation system, allowing for granular control of musical expression. Choices made in one song influence the outcomes of the next, paving the way for multi-scene interactive experiences grounded in narrative principles of persistence and emotional consequence. Table 1 shows the set of variables that persist throughout the entire three-song experience.

Figure 14 illustrates the user interface employed for the *Evergreen Blues Suite*. Each player uses a set of four keys to make lyrical choices. The player on the left uses [Q,W,E,R], while the player on the right uses [U,I,O,P]. For example, in Figure 14 the player on the right can choose between “unseen” and “serene”, in order to construct the phrase “Sights go unseen” or “Sights go
One immediate addition to the previous version of the real-time lyrical conversation system is that a lyric’s linguistic information is now represented directly through its text, as opposed to relying on a pre-determined cycle. On the other hand, the *Evergreen Blues Suite* eschews the fine-tuned pitch control that the previous lyrical conversation system provides; this was done partially for the purposes of prototyping and emphasis on bigger-picture narrative development, as well as based on the aforementioned playtesting feedback from the real-time lyrical conversation system.
<table>
<thead>
<tr>
<th>Variable name</th>
<th>Data type</th>
<th>Possible values/range</th>
</tr>
</thead>
<tbody>
<tr>
<td>redemption_choices</td>
<td>int</td>
<td>0 to 255</td>
</tr>
<tr>
<td>revenge_choices</td>
<td>int</td>
<td>0 to 255</td>
</tr>
<tr>
<td>rescue_choices</td>
<td>int</td>
<td>0 to 255</td>
</tr>
<tr>
<td>optimism</td>
<td>int</td>
<td>-255 to 255</td>
</tr>
<tr>
<td>atw_page_cave_save</td>
<td>string</td>
<td>“page”, “cage”, “save”</td>
</tr>
<tr>
<td>atw_away_chains_grade</td>
<td>string</td>
<td>“away”, “chains”, “grade”</td>
</tr>
<tr>
<td>atw_should_will</td>
<td>string</td>
<td>“should”, “will”</td>
</tr>
<tr>
<td>atw_before_beneath_beneath_between</td>
<td>string</td>
<td>“before”, “beneath”, “between”</td>
</tr>
<tr>
<td>evergreen_first_choice</td>
<td>string</td>
<td>“you”, “we”, “they”</td>
</tr>
<tr>
<td>evergreen_verse3_sights_sounds</td>
<td>string</td>
<td>“sights”, “sounds”</td>
</tr>
<tr>
<td>evergreen_verse4_line1_occurred</td>
<td>bool</td>
<td>true, false</td>
</tr>
<tr>
<td>evergreen_verse4_line3_occurred</td>
<td>bool</td>
<td>true, false</td>
</tr>
<tr>
<td>lata_least_choice_category</td>
<td>string</td>
<td>“redemption”, “revenge”, “rescue”</td>
</tr>
<tr>
<td>lata_final_choice_category</td>
<td>string</td>
<td>“redemption”, “revenge”, “rescue”</td>
</tr>
</tbody>
</table>

Table 1: Table of persistent variables throughout the entire three-song suite.

### 3.3.1 All the World’s a Stage

In *All the World’s a Stage*, the first song, players alternate between verses, each making choices individually. Figure 15 demonstrates the very opening sequence of lyrics; with each keystroke trigger, the corresponding lyric propagates visually from the player character to the stage. A vocal sample of the lyric is also played back immediately.

Narratively, the song sets the stage for a short scenario in which characters seek to escape their predestined roles in life; the title and central lyric of the song is derived from Jaques’ monologue in William Shakespeare’s *As You Like It*, Act II, Scene VII [43].
The first set of branching decisions is given to the player on the left, who is presented with three choices “Can we turn the page?” / “Can we burn the cage?” / “Can we learn to save?”, corresponding to character motivations of redemption, revenge, and rescue respectively. Figure 16 shows a screenshot of that first set of decisions. If, for example, the player chooses “Can we burn the cage?”, the \texttt{revenge\_choices} variable will be incremented by 1 in the game’s global state. While this individual choice goes by quickly such that it may not feel significant on its own, subsequent choices throughout the \textit{Evergreen Blues Suite} will also have an effect on the \texttt{redemption\_choices}, \texttt{revenge\_choices}, and \texttt{rescue\_choices} variables, allowing for more persistent narrative significance.

![Figure 15: Sequence of lyrics being controlled by the player on the left in \textit{All the World’s a Stage}.](image)

The next set of decisions is given to the player on the right, with the available options determined by the choice that the player on the left has made. Figure 17 illustrates the logic flow between the two decisions. For example, if the left
player chooses “Can we turn the page?”, the right player will be presented with choices resulting in either “Go beyond our worst behaviors” or “Go beyond our best behaviors”, whereas if the left player chooses “Can we burn the cage?”, the right player will be presented with choices that result in either “Go beyond our broken favors” or “Go beyond our bold-faced favors”.

Taken together, this first pair of decisions amongst the two players demonstrates a common interactive narrative technique of diverts and gathers [18], in which paths branch out (as in “turn the page” / “burn the cage” / “learn to save”), coalesce back together (“Go beyond our”), and branch back out (“our worst”, “our best”). Such fluidity of branching reinforces Ingold’s principles of narrative granularity and analogue choice-making [16].

Beyond the redemption/revenge/rescue choice paths, *All the World’s a Stage* also keeps track of a rudimentary optimism variable for the overall story state as shown in Table 1. This variable, represented as an integer, is incremented
Figure 17: Flowchart of branching lyric possibilities between the two players’ first choices in *All the World’s a Stage*. 
or decremented by 1 in accordance with certain lyric choices, such as “best” (which increments optimism by 1) vs. “worst” (which decrements optimism by 1) while constructing the phrase “Go beyond our best/worst behaviors”.

Figure 18: Screenshot of on-stage piano, violin, and cello players in *All the World’s a Stage*.

A variety of visual cues are also given to further tie together the music, lyrics, and stage imagery. For example, when the piano and cello enter upon the lyric “stage”, a piano and cello show up onscreen, as seen in the last frame of Figure 15. Similarly, upon the second iteration of “all the world’s a stage”, the musical introduction of the violin is heralded by the visual presence of a violin, as shown in Figure 18; at this point, on-stage characters are depicted playing the active instruments. Throughout the entire song, a theatrical stage is built both sonically and visually.

To further emphasize the core mechanic of players alternating choices, the order of player control for the second half of the song is reversed from that of
the first half. The player on the left begins the first verse with “All the world’s a stage”, followed by the player on the right iterating through “And all of us merely players”. On the other hand, the third verse (which starts the second half of the song) begins with the player on the right re-initiating the line “all the world’s a stage”, prepending it with either “If” or “Since”. The player on the left then prepends “we forever be players” with either “Should” or “Will”.

### 3.3.2 Evergreen Blues

The second song of the Evergreen Blues Suite, simply titled Evergreen Blues, builds upon the interaction scheme established in All the World’s a Stage by introducing musical polyphony and narrative simultaneity between players.

Figure 19 illustrates an example playthrough of the beginning of Evergreen Blues. By the end of this excerpt, the player on the left has constructed the phrase “You feel evergreen”, while the player on the right has constructed the phrase “Call us something”. As in All the World’s a Stage, each lyric choice triggers a prerecorded vocal sample, but these samples are now able to overlap. In doing so, players can engage in a degree of rhythmic interplay with one another on a musical level, in addition to “interrupting” each other in a manner similar to that in Oxenfree.

*Evergreen Blues* also employs a more traditional verse-chorus song form in comparison to *All the World’s a Stage*. In order to coordinate local phrase-based polyphony while still maintaining overall musical form, the song makes use of a checkpoint system, in which the first player to reach the end of a phrase must wait for the second player to finish their corresponding phrase. However, both melody and harmony within a phrase are asynchronous. Figure 20 demonstrates an example of how the harmony during the verse can change in such a manner. The first notated excerpt in the figure depicts a possibility for when the player on
Figure 19: Example flow of polyphonic lyrical decision-making in *Evergreen Blues*.
the left chooses “evergreen” before the player on the right chooses “something”; in this case, the Ab chord is transitioned to in measure 3. On the other hand, the second excerpt on the bottom depicts the harmonic progression for when, all else being equal, the player on the left does not choose “evergreen” at that precise moment. In that case, the Ab chord is only transitioned to when the player on the right chooses “something” in measure 4.

While the checkpoints help align players on a temporal level, it can be argued that they also limit the expressive freedom of players as well as the experience as a whole. Nevertheless, even with the checkpoint system in place, the possibility for local polyphony in *Evergreen Blues* allows for a higher degree of expressivity than the strict alternating mechanic seen in *All the World’s a Stage*.

In the second verse of *Evergreen Blues*, the current value optimism variable plays a role in determining the right player’s choices. The dialogue option “bad dreams” will only appear if optimism is below 0, while the dialogue option “sweet dreams” will only appear if optimism is above 0. As Ingold notes, structuring the choices in this narrowing manner creates an escalation of stakes, however slight; players’ choices are gradually funneled towards more meaningful actions, with more consequence given to later actions [16]. Of course, the positive-feedback structure shown in this example from *Evergreen Blues* is only one possibility for narrowing decision-making; one could easily imagine the choices being reversed, which would have just as palpable a narrative effect.

On a musical level, different lyric choices also correspond to different melodic variations in a manner similar to that of the notation section in *Seasons Change Together*. For example, in the final verse, if the left player chooses “Leave”, the vocal sample will begin on an Ab, whereas if the player chooses “Keep” the sample will begin on a Bb. The melodic variations between “in between”, “in the seams”, and “in a dream” are distinguished even further through different
Figure 20: Example of the asynchronous harmonic progression during the verse of *Evergreen Blues*. 
Figure 21: Melodic variations for different lyric options in *Evergreen Blues*, verse 4, left player.

Melisma patterns. Figure 21 illustrates the different melodic possibilities that arise from lyric choices in that final verse of *Evergreen Blues* for the player on the left; throughout the whole song, both players have access to such melodic variations.
Figure 22: Verse piano variations based on time since player action in *Evergreen Blues*.

Figure 23: Flute phrases based on player synchrony in *Evergreen Blues*. 
*Evergreen Blues* also introduces musical behavior based on calculation of player temporality along the axes of *player synchrony* and *time since player action*. Player synchrony is defined as the inter-onset interval between two players’ actions, with lower intervals corresponding to higher synchrony. In *Evergreen Blues*, player synchrony is mapped to note density of a musical phrase performed by the flute; the higher the synchrony, the more notes are performed. Figure 23 shows these flute variations in order of ascending synchrony; the phrase at A is mapped the lowest synchrony level, while the phrases in D are mapped to the highest synchrony level. Such a melodic gesture provides immediate musical feedback to players regarding their temporal relationship to one another, yet is unobtrusive so as to not obfuscate the rest of the musical narrative.

Similarly, the time since player action determines the rhythmic density as well as dynamics of the underlying instrumentation; both elements are commonly associated with musical energy [26]. Figure 22 illustrates the piano variations for the Eb chord in the verse of *Evergreen Blues* as they correspond to time since player action. Phrase A plays when the time is at its greatest, while phrase D plays when it is at its lowest; phrases B through C cover the temporal range in between. As such, the energy level of the song itself becomes a reflection of player action, deepening the musical interactivity of the experience.

Visually, now that the stage has been set, *Evergreen Blues* is able to utilize both theatrical design and onstage “actors” to further bring the operatic experience to life. For example, the actors will walk to different locations (or stay in place) depending on what lyrics players choose. Figure 24 illustrates a particularly noticeable visual result; when the player chooses “Sights”, the entire stage backdrop changes to an image of eye, whereas when the player chooses “Sounds”, the backdrop changes to an image of an ear. By having the stage and actors respond visually to players’ choices through animation and movement, albeit in rather simple and minimal ways, *Evergreen Blues* is able to take
first steps towards applying the principles of dynamic spaces outlined by Tamas Kemenczy when describing *Kentucky Route Zero* [21].

### 3.3.3 Look at the Audience

The final song, *Look at the Audience*, requires that the two players directly share control over the narrative. While in the previous two songs each player could make their own decisions without immediately affecting the other player, in this song both players control the exact same narrative thread. Figure 25 depicts the new shared control interface; whether the player on the left presses Q or the player on the right presses U, the same “want is” lyric decision will be made.

*Look at the Audience* opens with a recapitulation of the *All the World’s a Stage* lyrics and melodies. Indeed, the first three lyric choices are simply
Figure 25: Screenshot of the shared control interface in Look at the Audience.

Figure 26: Example of a choice made in All the World’s a Stage affecting available options in Look at the Audience.

repetitions of decisions that players already made back in the first song, serving both as a refresher for the player and a means of further indicating narrative
consequence. Figure 26 illustrates an example of such narrowing of choices; because the player chose the path corresponding to “Can we learn to save” in *All the World’s a Stage* (as seen on the left), both players are presented with that lyric choice in *Look at the Audience* (as seen on the right).

```python
### function calculate_least_choice_category ###
{
    - revenge_choices < rescue_choices && revenge_choices < redemption_choices:
        - return "revenge"
    - rescue_choices < revenge_choices && rescue_choices < redemption_choices:
        - return "rescue"
    - redemption_choices < revenge_choices && redemption_choices < rescue_choices:
        - return "redemption"
    - else:
        - return ""
}
```

Figure 27: Function to calculate least_choice_category in *Look at the Audience*.

The song also makes use of a least_choice_category calculation procedure, as shown in figure 27, to further hone in on a narrative conclusion. Based on the players’ choices throughout the suite, the character motivations of “revenge”, “rescue”, and “redemption” each have a current choice count; calculate_least_choice_category simply returns the character motivation with the lowest count. With this knowledge, the narrative backend can include or omit choices. For example, in one verse, players are potentially presented with “generate”, “liberate”, and “celebrate” as lyric options corresponding to the “revenge”, “rescue”, and “redemption” motivations respectively; the narrative system will then filter out the motivation that corresponds to the
least_choice_category. Figure 28 depicts two possible scenarios, one in which “generate” is omitted as a result of “revenge” being the least_choice_category, and one in which “celebrate” is omitted as a result of “redemption” being the least_choice_category.

The escalating narrative options in Look at the Audience are further emphasized by an increase in musical tension. Figure 29 shows the initial piano ostinato for the first verse of the song, in which a one-bar vamp is repeated. Upon entering the second verse, a pizzicato violin is introduced, resulting in an instrumental ostinato resembling figure 30; each bar of the ostinato is repeated indefinitely until its corresponding lyric choice is made, functioning similarly to the local harmonic progression management in Evergreen Blues. The second and third verse are broken up by a restatement of the “All the world’s a stage” phrase, which itself entails additional chord tones atop the drone introduced at the very beginning of the song. The third verse sees increased rhythmic activity
in both the piano and violin lines, as shown in figure 31. Last but not least, the fourth and final verse, notated in figure 32, introduces an upper piano line which involves an extended 3:5 polyrhythm that stochastically alternates between two Bbs an octave apart; the alternation between octaves is handled internally by
the music system through the use of multi-instrument containers in FMOD [12].

By the end of Look at the Audience, players will have made a set of musical and narrative decisions that culminate in the conclusion of the scenario set out
in *All the World’s a Stage*: the characters, motivated by some combination of revenge, redemption, and rescue, escape their predefined roles, and the audience is left to draw their own conclusions as to what occur next. In this way, the *Evergreen Blues Suite* can effectively be seen as both a standalone short story and a single act in the context of an even larger overarching narrative.

### 3.3.4 Playtesting Feedback and Discussion

Playtesters generally highlighted the lyric choice mechanic in the *Evergreen Blues Suite* as a unique and fun method of musical gameplay. As was the case with *Seasons Change Together*, players appreciated having expressive control over the music and narrative simultaneously.

The user interface proved to be somewhat non-intuitive to some players. In particular, several playtesters mentioned that the separation between the words representing players’ options (on the bottom left and right of the screen) and the actual resulting text (depicted as part of the on-stage action) necessitated a constant shift in visual attention, which ultimately felt distracting. A solution to this could be to consolidate the player choice text with the on-stage action through additional animation and movement; for example, instead of having all lyric options disappear upon a player making a choice, perhaps only the unselected options disappear. The selected option is then given visual emphasis, and can further be highlighted through increased size or brightness in addition to animated movement towards the stage. Alternatively, the appearance of the chosen text in the stage-centered speech bubble can be synchronized and given the same animation pattern as the chosen text in the actual user interface.

*Evergreen Blues* was noted as being the most musically expressive song of the three, in part due to its real-time polyphonic affordances. On the other hand, some playtesters found the real-time nature of the musical interactions some-
what disruptive and chaotic. Unlike rhythm games such as *Guitar Hero* [14], the *Evergreen Blues Suite* does not rely on skill for musical decision-making; there are no right or wrong decisions in the three-song suite, and the experience is designed such that all paths are musically valid. Nevertheless, design decisions can perhaps be made to guide players towards more consonant and less disruptive behavior. For example, the music system can quantize user input to align vocal sample playback with the underlying instrumental track. While it is true that any expressive interactive system necessarily affords the possibility of music sounding unpleasant, care should be taken to ensure that players do not get discouraged from engaging with the experience out of frustration.

Playtesters had varying perceptions on the presence of different voices in the *Evergreen Blues Suite*. All the vocal samples in the experience were sung by the same individual, which several players found detracted from the collaborative nature of the experience. On the other hand, perhaps due to the varying ranges in melodies, some players actually perceived two different voices where there was only one. Recording separate voices for each player’s part would easily remove such ambiguity. However, the false perception of additional voices does provide a testament to the power of collaborative experiences on our musical perception.

A common thread of feedback for the *Evergreen Blues Suite* involved its overall narrative. While players are able to make a variety of musically expressive choices, it can be argued that few of the decisions they make are imbued with sufficient narrative meaning or consequence, especially given the arc of the entire three songs that comprise the suite. In an attempt to maintain granularity of narrative action, it appears that the *Evergreen Blues Suite* shies slightly too far away from significant narrative consequence. On the level of individual choices, more attention can be paid to differentiate each lyric option, such that each choice is dramatically distinct from the other. For example, the first choice
given to the player on the right in *Evergreen Blues* is between “Call” and “Tell”, which is relatively insignificant on a dramatic level; a more significant pair of choices might be “Tell” versus “Hide”. Even some of the more differentiated choices, such as “Build” versus “Burn” (in the third verse of *Evergreen Blues*), can be given more long-term narrative meaning through more incorporation of persistent state as well as musical and visual consequence beyond one-off events or animations. On a larger scale, a more substantial narrative framing device could help contextualize the entire experience; even a simple addition such as an introductory cutscene at the very beginning of the suite would provide more concrete characterization and setting, leading to more emotional investment on the part of players.
4 Towards a Lyrical Multiplayer Narrative Opera Platform

In addition to the development of *Seasons Change Together*, the real-time lyrical conversation system, and the *Evergreen Blues Suite*, this thesis has led to the creation of an overarching lyrical multiplayer narrative opera platform, or LMNOP for short.

As a platform and tool, the adoption of the project by other creators to develop new scenes and experiences can be considered a measure of success. To this end, it is one eventual goal to find artists who would be interested and willing to try creating original experiences for the platform, though such an endeavor has proved to be beyond the scope of this thesis.

4.1 Architecture Overview

The platform is built using the Unity [48] game engine, which supports both 2D and 3D graphics and physics, provides multiplayer networking capabilities, and allows for convenient cross-platform building to a range of mobile (including iOS, Android, and Windows Phone), desktop (including Mac, Windows, and Linux), and console systems (including PlayStation 4, Xbox One, and Nintendo Switch).

Inkle Studios’ Ink [17] narrative scripting language is used to implement choices and state; it is powerful, flexible, and provides easy integration into Unity, although for the purposes of the thesis it has been extended and reconfigured. Ink was originally designed for single-player interactive narrative, so introducing multiplayer functionality requires the ability to sync multiple Ink scripts within Unity. The expansion of the Ink language for multiplayer support,
as well as more musically-oriented capabilities, will provide a boon to creators of interactive fiction.

On the audio front, vocal and instrumental parts follow the principles of the real-time lyrical conversation system described in chapter 3.2.2. Firelight Technologies’ FMOD [12] provides the basis of the audio engine, for reasons similar to those regarding Ink. A logic layer has been implemented to allow dynamic transitions on the note-by-note level in addition to accommodating different player pitch choices for the same lines. The platform also implements calculations for player temporality as described in chapter 3.3.2; such temporal measures can be used to enhance the operatic experience along the lines of multiplayer collaboration based on rhythmic copresence.

The platform provides an application programming interface (API). Each method in the LMNOP API is implemented in Unity using the C# language, and exposed to users via Ink; all the Unity methods are encapsulated within a StoryHandler class.

4.2 The LMNOP API

The LMNOP API functions primarily through parameterized tags that allow the various components of the system to communicate with each other. Tags can be defined freely in an Ink script, and will be passed to Unity upon a player choice. Figure 33 shows an example of parameterized tags in Ink; the # symbol denotes a tag, which is syntactically structured as `tagContext: tagName, tagParamValue0, tagParamValue1, ..., tagParamValueN`. The `tagContext` informs the system of the overall context, or category, that the tag is relevant to. The `tagName` refers to the specific function that should be called, while `tagParamValue0` to `tagParamValueN` refer to the arguments to
be passed into said function. For example, the tag in line 5 of figure 33 roughly equates to “within the scene context, move the PianoProp object to the location specified by PianoV1Target across a duration of 0.0 seconds; do not apply running animations, do not fix movement to the original y-position, and do not rotate the object to face the target location”.

```
1   * stage
2       # music: setEmitterParameter, AllTheWorlds_ToVerse1I, 1.0
3       # music: playEvent, V1L1_Stage
4
5       # scene: moveObjectToTarget, PianoProp, PianoV1Target, 0.0, false, false, false
6       # scene: moveObjectToTarget, CelloProp, CelloV1Target, 0.0, false, false, false
7       # scene: moveObjectToTarget, Floor, FloorV1Target, 0.0, false, false, false
8       # scene: moveObjectToTarget, PianoSeatProp, PianoSeatV1Target, 0.0, false, false, false
```

Figure 33: Example of parameterized tags in Ink for the first iteration of “stage” in *All the World’s a Stage*.

The following sections in this chapter examine the various actions within the music, dialogue, data, and scene tag contexts in further detail.

### 4.2.1 Music

The music context, shown in table 2, provides a wrapper around FMOD’s audio capabilities. The setEmitterParameter and setEmitterParameterWithDelay are generally responsible for coordinating the background music, instrumental accompaniment, and overall song form, while the playMusicEvent method generally deals with triggering vocal samples in real-time.
<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setEmitterParameter</td>
<td>paramName (string), paramValue (float)</td>
<td>Set parameter value for music emitter</td>
</tr>
<tr>
<td>setEmitterParameterWithDelay</td>
<td>paramName (string), paramValue (float), delayAmount (float)</td>
<td>Set parameter value for music emitter, with delay</td>
</tr>
<tr>
<td>playMusicEvent</td>
<td>eventName (string)</td>
<td>Play a given event and register it as the current event</td>
</tr>
</tbody>
</table>

Table 2: Table of music context methods and parameters within the LMNOP API.

Figure 34: FMOD Studio result of the `setEmitterParameter` method being called with `paramName = “AllTheWorlds_ToVerse1I”` and `paramValue = 1.0`.

Figure 34 demonstrates the result of the `setEmitterParameter` method call in line 2 of figure 33. The left screenshot depicts the internal FMOD timeline state; the engine is currently looping the “Opening” section (measure 1). Upon invoking the tag “# music: setEmitterParameter, AllTheWorlds_ToVerse1I,”
1.0", the state transitions to the screenshot on the right; the value of `AllTheWorlds_ToVerse1I` has been set to 1.0, which allows the timeline to transition to the “Verse1_I” section (measure 3).

```csharp
void PlayMusicEvent(string eventName)
{
    // Play a music event and keep track of it as current event

    string eventPath = string.Format("{0}{1}",
        _fmodEventPrefix, eventName);

    if (_currFmodEventInstance.isValid())
    {
        _currFmodEventInstance.stop(FMOD.Studio.STOP_MODE.
            ALLOWFADEOUT);
        _currFmodEventInstance.release();
    }

    _currFmodEventInstance = FMODUnity.RuntimeManager.
        CreateInstance(eventPath);
    _currFmodEventInstance.start();
}
```

Figure 35: Implementation of `PlayMusicEvent()` on the Unity/C# end, corresponding to the `playMusicEvent` method in the LMNOP API.

The implementation of the `playMusicEvent` method on the Unity/C# side, integrating directly with FMOD, is shown in figure 35. The method takes a given event name, usually referring to a vocal sample for a lyric choice, and constructs a full event path (line 5). It then optionally stops and releases the currently playing event (_currFmodEventInstance), if one such event instance exists (lines 7-11). Finally, the method constructs a new event using the constructed event path, registers it as the current music event for the given story handler (line 13), and begins playback (line 14).
4.2.2 Dialogue

The dialogue context deals primarily with components that involve lyrics, text, and user interface. Table 3 describes the available methods within the dialogue context.

Many of the actions in this context are used as convenience functions to set global interface parameters or update text elements. For example, `isUIEnabled` takes in a boolean, `shouldSetEnabled`, which determines whether the user interface should currently be enabled. This method is used in order to allow for differing player mechanics such as the alternating choice gestures in *All the World’s a Stage*.

`enterUICheckpoint` and `enterUICheckpointWithChoice` provide implementation for the checkpoint system described in chapter 3.3.2. The two methods vary in that `enterUICheckpointWithChoice` adds a choice to the stack upon clearing the current checkpoint; this enables synchrony of form, such as when arriving at the chorus of *Evergreen Blues*.

The `setObjectForUIToFollow` method allows for attachment of dialogue options to arbitrary game objects in a scene. Figure 36 shows a screenshot of dialogue options attached to an in-game character in a first-person game setting; despite visual differences between the game and the *Evergreen Blues Suite*, both experiences utilize the LMNOP API as their technological backend.

4.2.3 Data

The data context deals primarily with data across multiple story handlers, including the persistent game state, in addition to various utility and helper functions. Table 4 shows the context’s available methods.
<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setUIActive</td>
<td>shouldSetActive (boolean)</td>
<td>Activate/deactivate UI elements</td>
</tr>
<tr>
<td>setUIEnabled</td>
<td>shouldSetEnabled (boolean)</td>
<td>Enable/disable UI elements</td>
</tr>
<tr>
<td>setUIEnabledGlobal</td>
<td>shouldSetEnabled (boolean)</td>
<td>Enable/disable UI elements for all story handlers</td>
</tr>
<tr>
<td>enterUICheckpoint</td>
<td>N/A</td>
<td>Enter a UI checkpoint; disables self’s UI until all story handlers have reached checkpoint</td>
</tr>
<tr>
<td>enterUICheckpointWithChoice</td>
<td>N/A</td>
<td>Enter a UI checkpoint, and make a choice on clearing</td>
</tr>
<tr>
<td>clearCurrVerseText</td>
<td>N/A</td>
<td>Set whether speech bubble text should be cleared for next choice</td>
</tr>
<tr>
<td>clearSpeechBubbleText</td>
<td>N/A</td>
<td>Clear speech bubble text</td>
</tr>
<tr>
<td>ignoreTextForNextChoice</td>
<td>N/A</td>
<td>Make speech bubble ignore text for next choice</td>
</tr>
<tr>
<td>setObjectForUIToFollow</td>
<td>objectName (string)</td>
<td>Set name of game object for UI to follow</td>
</tr>
<tr>
<td>hideUIForDuration</td>
<td>durationInSeconds (float)</td>
<td>Hide UI for a period of time</td>
</tr>
<tr>
<td>hideSpeechBubble</td>
<td>N/A</td>
<td>Hide the story handler’s speech bubble object</td>
</tr>
<tr>
<td>showSpeechBubble</td>
<td>N/A</td>
<td>Show the story handler’s speech bubble object</td>
</tr>
<tr>
<td>setVariable</td>
<td>storyName (string), varName (string), varValue (string)</td>
<td>Set the value of a variable within an individual story handler</td>
</tr>
</tbody>
</table>

Table 3: Table of dialogue context parameters within the LMNOP API.
Figure 36: Example of the `setObjectForUIToFollow` method as applied in a first-person setting using the LMNOP API.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>broadcastVariablesState</td>
<td>N/A</td>
<td>Broadcast this story handler’s variable state to all handlers</td>
</tr>
<tr>
<td>loadVariablesState</td>
<td>N/A</td>
<td>Load this story’s handler variable state from the global state</td>
</tr>
<tr>
<td>setCurrentPath</td>
<td>pathName (string)</td>
<td>Set the current Ink script path</td>
</tr>
<tr>
<td>invokeFunction</td>
<td>functionName (string), delayAmount (float)</td>
<td>Invoke a specific function by name, with delay</td>
</tr>
<tr>
<td>debugLog</td>
<td>message (string)</td>
<td>Log a message to the debug console</td>
</tr>
</tbody>
</table>

Table 4: Table of data context methods and parameters within the LMNOP API.

`broadcastVariablesState` and `loadVariablesState` serve as a means to provide a persistent game state amongst multiple songs and story handlers. Because the LMNOP API utilizes Ink scripts in an unconventional manner in order
to accommodate real-time multiplayer capabilities, Ink’s built-in save/load functions are not used. Rather, each script’s variable state is synchronized using calls to broadcastVariablesState; these calls conceptually correspond to “save” events within the save/load paradigm, while calls to loadVariablesState correspond to “load” events. In the Evergreen Blues Suite, the global variable state consists of the variables shown in table 1.

### 4.2.4 Scene

The scene context encompasses methods that pertain primarily to visual and gameplay aspects of the platform. The majority of the methods deal with game objects, with a handful of methods dedicated to navigation between songs and menus.

```plaintext
1   # scene: moveObjectToTarget, PlayerComm3Stage1, 
    PlayerComm3Stage1Verse1Target, 4.0, false, true, false
```

Figure 37: LMNOP API call to the moveObjectToTarget method used in Evergreen Blues.

Figure 38 illustrates the result of the method call shown in figure 37, which also corresponds to the in-game actions taken during Evergreen Blues as shown in figure 19 (section 3.3.2). PlayerComm3Stage1 refers to the stage actor on the left, while the PlayerComm3Stage1Verse1Target refers to the target object. In each image of figure 38, the positions of both objects are highlighted in orange. The PlayerComm3Stage1 object has an outline of a humanoid figure, while the PlayerComm3Stage1Verse1Target object is spherical in shape (the sphere is for debugging and illustration purposes only within the editor view; the target object is not rendered in the final game view).

The top image of figure 38 shows the objects prior to the moveObjectToTarget
Figure 38: Unity editor view for the result of LMNOP API call shown in figure 37. The action also corresponds to the in-game screenshots seen in figure 19.
method call from figure 37. Once the call is made, the PlayerComm3Stage1 object begins moving towards the PlayerComm3Stage1Verse1Target object, as seen in the middle image. The entire movement occurs over a period of 4 seconds (as specified by duration = 4.0), the PlayerComm3Stage1 object’s animator will walk rather than run (as specified by shouldRun = false), the PlayerComm3Stage1 object will retain its original y-position (as specified by shouldUseOriginalPosY = true), and the PlayerComm3Stage1 object will not rotate to face the PlayerComm3Stage1Verse1Target object (as specified by shouldLookAtTarget = false). By the end of the 4 seconds, the scene will resemble the bottom image of figure 38.

The LMNOP API provides a means for creators to flexibly define their own behaviors in the context of a multiplayer interactive opera while abstracting the technical implementation of audio, visual, and gameplay elements. In doing so, the API allows creators to focus on actual authoring of content.
<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>moveObjectToTarget</td>
<td>objectName (string), targetName (string), duration (float), shouldRun (boolean), shouldUseOriginalPosY (boolean), shouldLookAtTarget (boolean)</td>
<td>Move an object to a given target location</td>
</tr>
<tr>
<td>moveObjectToPosition</td>
<td>objectName (string), posX (float), posY (float), posZ (float)</td>
<td>Move an object to a specified position</td>
</tr>
<tr>
<td>instantiateObject</td>
<td>objectName (string), prefabName (string), posX (float), posY (float), posZ (float), rotX (float), rotY (float), rotZ (float)</td>
<td>Instantiate an object instance using a given prefab</td>
</tr>
<tr>
<td>destroyObject</td>
<td>objectName (string)</td>
<td>Destroy an object</td>
</tr>
<tr>
<td>rotateObject</td>
<td>objectName (string), rotX (float), rotY (float), rotZ (float)</td>
<td>Rotate an object</td>
</tr>
<tr>
<td>activateObject</td>
<td>objectName (string)</td>
<td>Activate an object</td>
</tr>
<tr>
<td>deactivateObject</td>
<td>objectName (string)</td>
<td>Deactivate an object</td>
</tr>
<tr>
<td>scaleObject</td>
<td>objectName (string), scaleX (float), scaleY (float), scaleZ (float)</td>
<td>Scale an object</td>
</tr>
<tr>
<td>hideObject</td>
<td>objectName (string)</td>
<td>Shorthand for scaleObject, 0, 0, 0</td>
</tr>
<tr>
<td>setObjectAnimBool</td>
<td>objectName (string), animBoolName (string), animBoolValue (boolean)</td>
<td>Set the value of a boolean for an object’s animator</td>
</tr>
<tr>
<td>goToSong</td>
<td>sceneIdx (int), delayAmount (float)</td>
<td>Navigate to a song after a given delay</td>
</tr>
<tr>
<td>goToMainMenu</td>
<td>delayAmount (float)</td>
<td>Navigate to main menu after a given delay</td>
</tr>
<tr>
<td>exitGame</td>
<td>delayAmount (float)</td>
<td>Exit game after a given delay</td>
</tr>
</tbody>
</table>

Table 5: Table of scene context methods and parameters within the LMNOP API.
5 Conclusions and Future Work

This thesis work presents a variety of experiences as well as an overarching platform for the creation of multiplayer interactive operas. Several conclusions and insights can be made from the carrying out of this project, setting the groundwork for potential future developments in the field of collaborative musical narrative.

Based on playtesting and feedback, it appears to be the case that players generally do appreciate control over both musical and narrative elements, and that compelling multiplayer collaboration in such a context is feasible. Indeed, allowing players to interact with each other tends to deepen the experience on musical, narrative, and social levels simultaneously.

A distinction between control and creation can be made with regards to the experiences and platform implemented in this work. As the platform currently stands, more creative affordances are currently given to authors of content, that is, people who use the platform to create individual experiences. Players themselves are afforded musical and narrative expression and control through the interactivity of the experiences, but ultimately are unable to create any new material beyond what is already provided by authors. The expressivity of the platform is hence limited by design. Further explorations of the song as instrument may help blur the boundary between player and author.

Playtesting the experiences proved to be a challenge in and of itself. The nature of play, especially in a multiplayer context, necessitates a high degree of subjectivity, and it was sometimes difficult to draw conclusions based on individual players feedback due to contradictory opinions. Similarly, distinctions between implicit and explicit player feedback should be made when attempting to base work off of playtesting. A more rigorous approach in a controlled envi-
environment, perhaps involving formal surveys as well as data collection of player actions, could yield more nuanced and generalizable playtesting results.

As mentioned in chapter 3.3.4, the overall narrative of the *Evergreen Blues Suite* tends to lack significant emotional consequence when experienced by players. In order to instill participants with more of a sense of the long-term impact of their choices, it is likely that the lyrical choices themselves need to be imbued with more drastic musical, visual, and narrative consequences. Experimenting with increasing the level of drama will hopefully help strengthen and elucidate the relationships between choice, consequence, and emotional impact in the context of a collaborative operatic game such as the *Evergreen Blues Suite*.

Another direct next step for this thesis project is the development of the lyrical multiplayer narrative opera platform. To this end, correspondence has been made with a range of artists, musicians, and technologists, with the intention of collaborating on new works using the platform. Such works, in addition to contributing to the field of interactive operatic experiences, will no doubt lead to further insights as to strengths, limitations, and future directions of the platform itself.

This work can also be extended into the realm of live performance. The experiences created as part of this thesis all rely on pre-recorded material, and all audiovisual material exists in the virtual realm. It would be a worthwhile experiment to build a system to allow for live musicians and actors to perform the material based on player actions. The system could translate actions into symbolic instructions on the level of musical notation and stage directions, which musicians and actors can then render in real-time. Improvisation on the performers’ parts can then further influence interplay between players, creating new dynamics of interaction. Such an endeavor would no doubt present unique possibilities and challenges of its own.
Another potentially fruitful direction to explore is that of user interfaces beyond screens and keyboards. For example, given the importance of vocals and singing within the operatic form, it seems natural that players should be allowed to make choices by actually vocalizing. The audio engine could then extract features pertaining to pitch, rhythm, timbre, and dynamics and map those features to a particular choice (the most obvious being that of player input pitch corresponding to output pitch, scaled along the aforementioned vertical axis). This input method is directly musical, and allows more creativity in player expression; players’ vocal input can also be used to drive background layers of music or even provide additional melodic overlays.

Furthermore, this thesis work can hopefully be used to explore the potential for interactive musical narrative experiences to improve health and wellbeing. Prior research has been conducted by the author with regards to using techniques from game music for the purposes of affect improvement [46], in addition to combining the power of familiar songs with neural entrainment on multiple temporal scales in the context of a rhythm-based game [45]. Incorporating some of the concepts introduced in this thesis may present interesting new directions for investigating the effects of musical interactivity, and interactive musical narrative in particular, on wellbeing.

The experiences implemented in this project are all relatively short, with typical playthroughs lasting under ten minutes each. As such, it becomes difficult to realistically evaluate long-term narrative implications. Creating a multiplayer interactive opera that is even one act long would provide invaluable insight into how compelling such an experience can be on the scale of long form drama; one can only imagine the narrative possibilities for an interactive experience on the scale of Wagner’s Der Ring des Nibelungen [50].

Similarly, future experiences can also expand on the massive aspect of inter-
active operas through the implementation of multiplayer mechanics that allow for more meaningful interplay between a large number of simultaneous participants. More emphasis on emergent narrative approaches may prove fruitful on that front. Parser-based interactions, in which players directly enter text input, can also pave the way for direct player authorship of content on a lyrical level; in combination with voice synthesis systems, it could be possible for all of the musical content of an interactive opera to be entirely player-generated.

Just as operas themselves take a multitude of forms, so it is the case with multiplayer interactive musical narratives. This work represents one potential direction for such experiences, but is by no means the only path. It is the hope that this thesis will be of value to those continuing work in the tradition of music, narrative, and interactivity.
Bibliography


