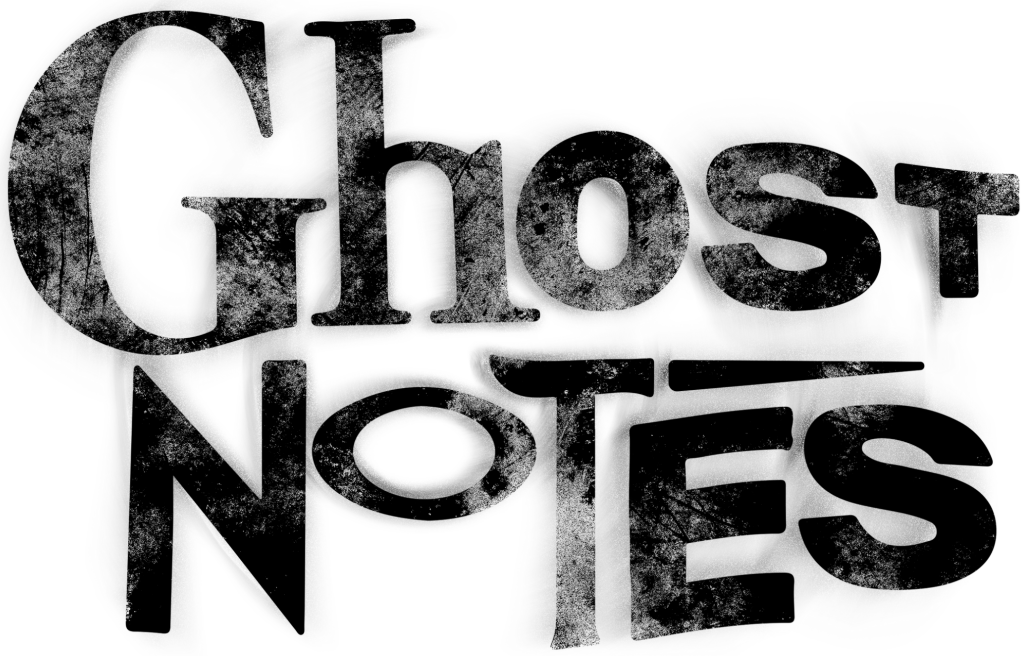


Project Documentation

Ben Johnson
M.A. in Interactive Media
Elon University
April 21, 2021

Table of Contents

Project Overview

1. Description

Ghost Notes is an interactive, web-based narrative with elements of music production, filmmaking, and game design. The primary goal of this project is to illustrate different techniques and music styles employed by rock and alternative musicians in the United States. By utilizing customizable music and elements of music production, the project also introduces users to the process of music creation in the group setting of a rock band.

The narrative of the project puts users into the role of a rising tour manager who has just been brought on to host the next tour of a mysterious, reclusive rockstar with a secret identity. Having just gone off the grid, the musician tasks the user with running the entire production by hiring new band members each step of the way.

Though the project puts the user in the role of a tour manager, their tasks might resemble that of a producer or songwriter (and at times, an engineer or stagehand). At each stop of the tour, users are presented with a song to perform and must choose a singer, guitarist, bassist, and drummer to fill out the band. During the performance, users are judged on how well they are able to mix different styles of music and keep up with technical difficulties through a series of mini games.

Figure A: musician selection screen

2. Gameplay

There is one song per set, and four members in the band. At the beginning of each show, the user is given a choice between three musicians for each role of the band. During each performance, technical issues occur that force the user to multi-task and complete quick time events while maintaining a balanced mix. Interspersed between each concert sequence are narrative scenes primarily driven by diegetic audio.

In future expansions of the game, users will get a post-show rundown of how well the band performed and how much money has been earned for the next show. The user's amount of success will then determine the size of the venue they are able to play on the next stop (a bar is the bare minimum, a rock club is moderate, and a stadium is the top tier.)

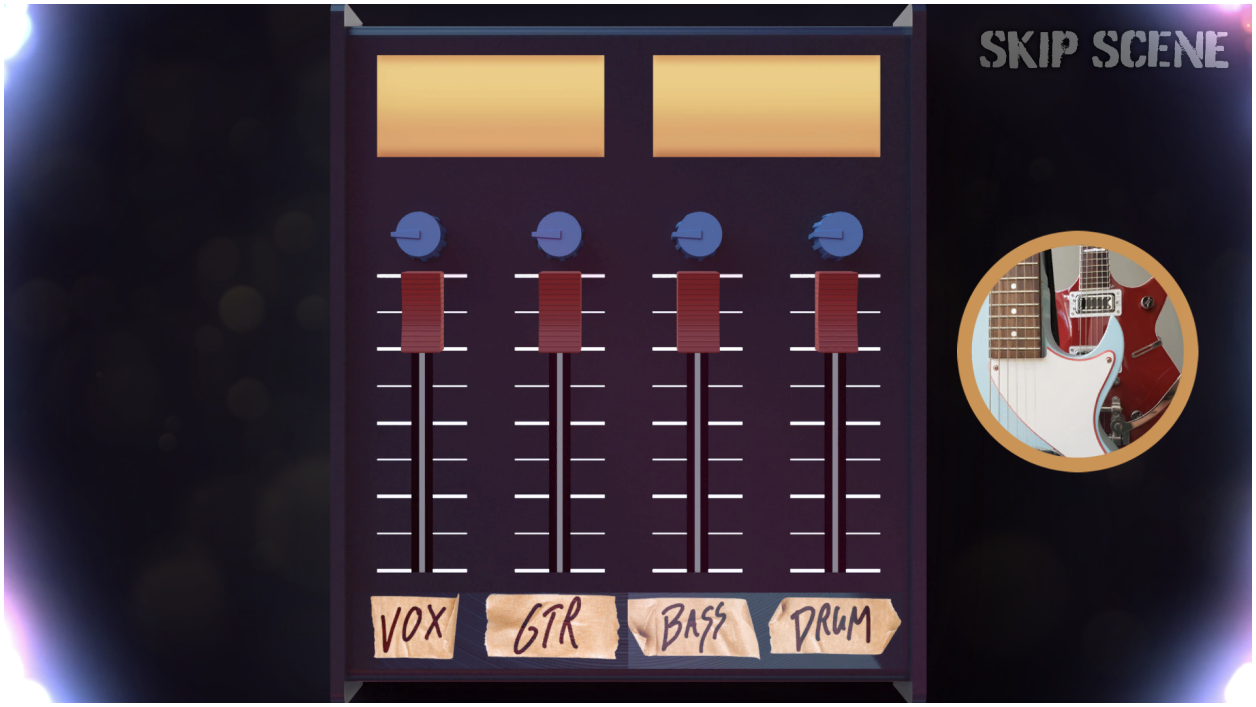


Figure B: concert gameplay screen, with a quick time event prompt to the right

3. Style

The visual identity of the graphical user interface resembles items typically found on concert tours. Menus, loading screens, and other functional elements resemble spray-painted equipment cases similar to what would be used to transport an instrument. This visual is a large part of the identity of the game, as it is the primary visual device that transitions the narrative from scene to scene.

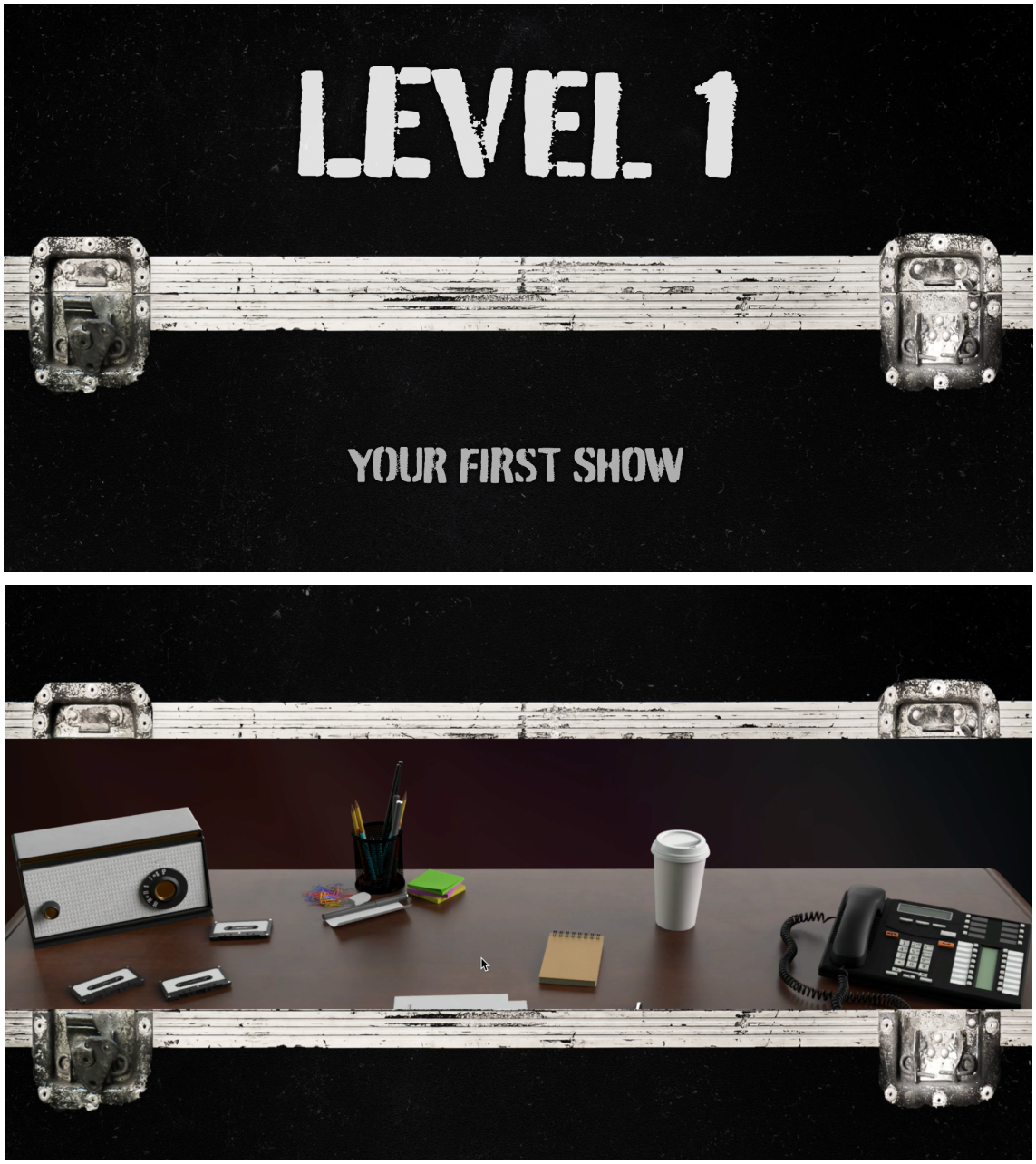
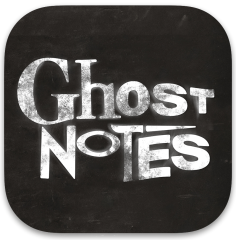


Figure C and D: stage crate transition sequence

4. Technology

Ghost Notes was created in Unity 2019 and is scripted in C#. The final product is available as a downloadable application for both Mac and Windows, and a demo version of the game utilizes the Unity WebGL framework to be playable in most modern browsers. All audio was recorded and mixed in Ableton Live, mastered in iZotope Ozone, and compressed for lower latency in Adobe Audition. 2D visuals and UI elements were created using Adobe Photoshop and Illustrator. The 3D mixer asset was created in Blender and all 3D scenes were arranged and rendered in Adobe Dimension. See credits for specific assets and libraries used.



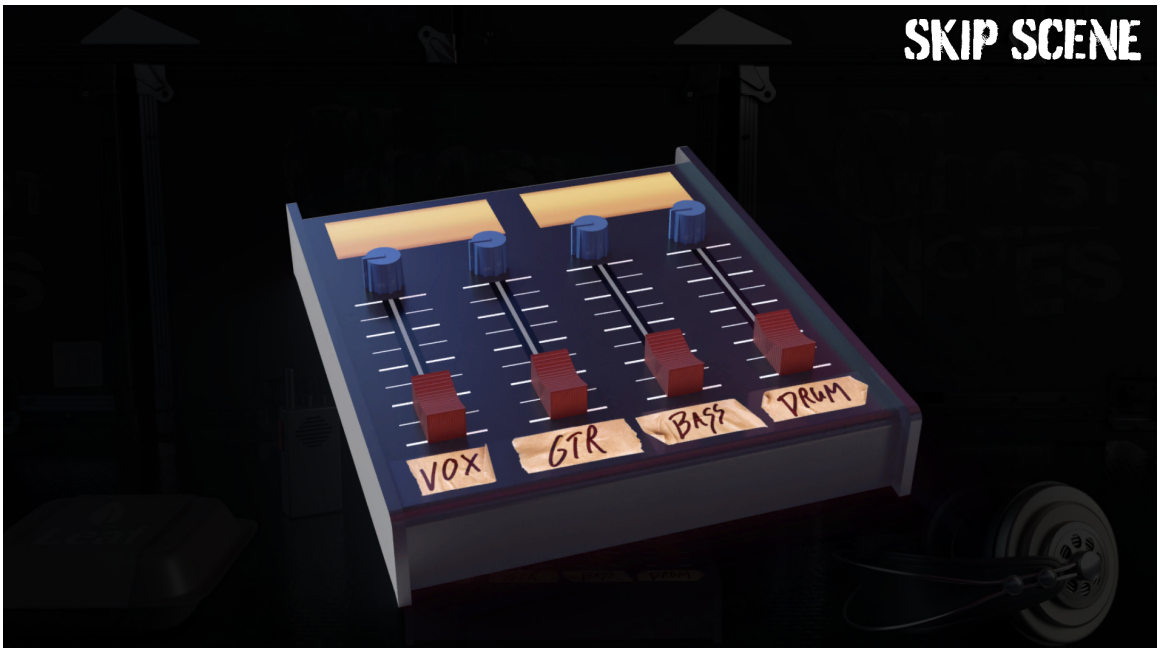


Figure E: application icon

Figure F: fully custom 3D mixer asset, as shown in tutorial

5. Target Audience

The primary audience for this project is millennial-aged users who may have enjoyed games like Guitar Hero and Rock Band in the past and who are interested in the behind-the-scenes aspect of the music industry. They are creative, curious, task-oriented, and may be described as an auditory learner.

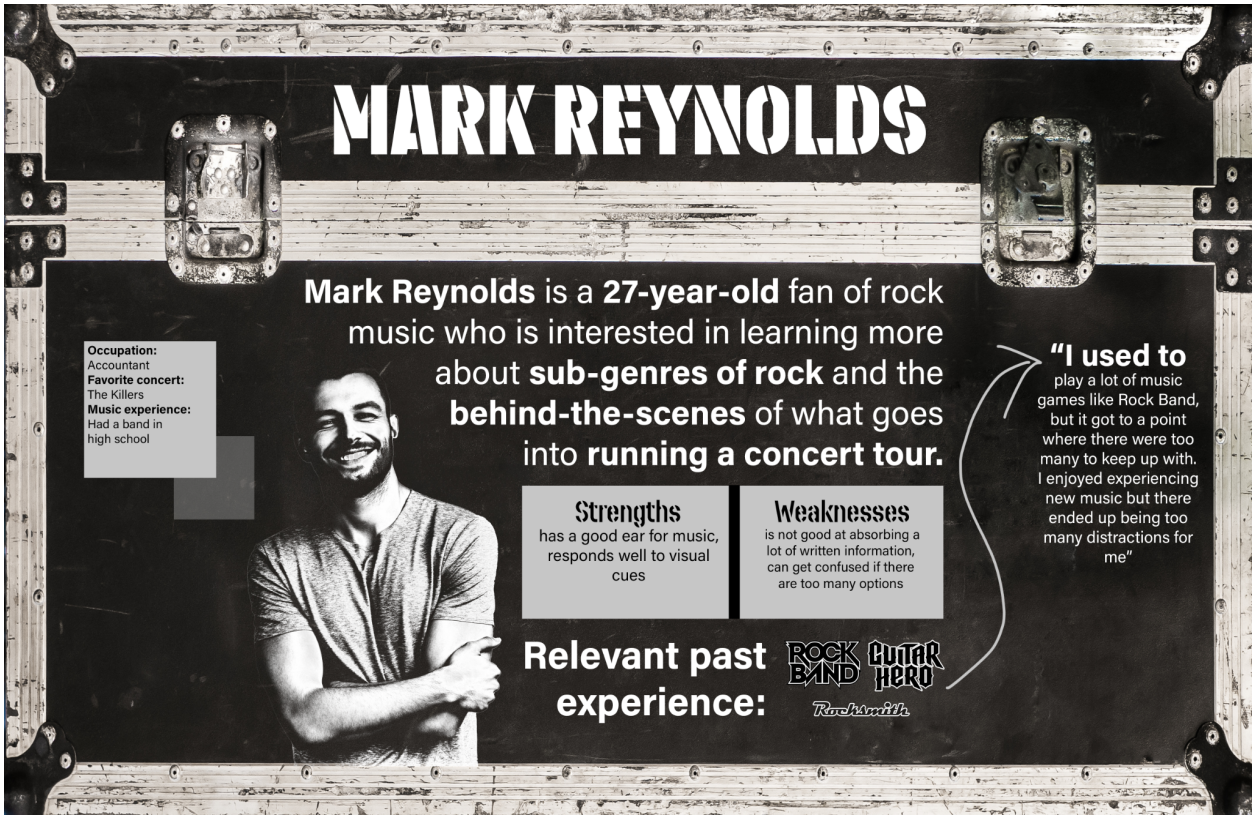


Figure G: primary audience user persona

6. Project Timeline

This project officially began in January of 2021, but the premise was conceived as early as 2017. Pre-production took place from January to February, with story elements, art style exploration, and music coordination being the primary focus. The production phase was completed in March and saw the creation of one fully custom 3D asset (the audio mixer), all scene backgrounds, and most voiceover work. The development phase spanned the remainder of March into April and included all technical building of the app, as well as the user test process and subsequent refinements. Below is a rough breakdown of weekly accomplishments:

Week of January 18	● Proposal submitted and approved
Week of January 25	● Artifact created: user persona
Week of February 1	● Artifact created: content inventory ● Research journal started
Week of February 8	● Artifact created: narrative map ● Artifact created: narrative script
Week of February 15	● Artifact created: storyboard ● Completed list of musicians, began contacting process ● First voiceover session, rough draft of scene one complete
Week of February 22	● Coordinated musicians
Week of March 1	● Artifact created: mood board and style guide ● Artifact created: logo animation
Week of March 8	● Began production of mixer asset ● Created first prototype, a simple set of UI sliders controlling audio
Week of March 15	● Music demos finalized, sent to artists
Week of March 22	● Completion of mixer asset
Week of March 29	● All assets complete
Week of April 5	● Playable prototype developed ● First round of user tests completed
Week of April 12	● Second round of user tests completed ● Refinements implemented
Week of April 19	● Artifact created: user test report ● Final build of game delivered

7. Modifications

Ghost Notes began as a strictly web-based experience but was shifted to a downloadable title when it became clear that limitations of the Unity WebGL framework would prevent the end user from experiencing the project as intended. Three concert sequences were originally planned, but scheduling conflicts with involved musicians prevented additional recordings to be collected. Of the six remaining songs from these two scrapped concerts, two songs were repurposed as background music for the main menu screen and the credits sequence.

Figure H: main menu screen

Project Deliverable

A digital download of the game is available at 3enjohnson.com/ghostnotes.html for both Mac and Windows platforms. A web-based “lite” demo version of the game will also be available in the near future and will be hosted on the same page. This project has been tested on a number of computers and should work on most modern machines. It is recommended that the game be played without any programs running in the background to cut down on possible lag. It is also suggested that the game be played with headphones if possible.

Research

1. Influence Journal

At the beginning of the pre-production process, a research journal was created to keep track of design inspiration and possible influences. A total of eight relevant and unfamiliar video games were experienced and analyzed. The list is as follows:

1. DJ Hero (FreeStyleGames, 2009)
2. Quest For Fame (Virtual Music Entertainment, 1995)
3. Guitar Hero: On Tour (Vicarious Visions, 2009)
4. Lego Rock Band (Harmonix, 2009)
5. Rock Revolution (HB Studios, 2008)
6. Guitar Hero: Warriors of Rock (Neversoft, 2010)
7. Amplitude (Harmonix, 2003)
8. Parappa The Rapper (NanaOn-Sha, 1996)

Note: Fuser (Harmonix, 2020) was also analyzed but is excluded from this list as it was not fully experienced

2. Initial data collection

Throughout the early stages of the pre-production process, survey forms and informal interviews were conducted to ensure aesthetics such as song production and UI imagery successfully conveyed the appropriate time period and genre.

3. User test

In the final stages of development, four user test sessions were conducted based on a simplified version of the game. The purpose of these tests was to gauge initial reaction to the user interface and collect opinions on the tutorial scene. Users were able to play a web-hosted version of the game and participate in a remote session via Zoom as this process took place during the Covid-19 pandemic.

Feedback from these sessions influenced a rewrite of the tutorial sequence, with the current version now utilizing data chunking methods and a hands-on trial stage to better establish game mechanics and enhance clarity.

Process Documentation

1. Audio

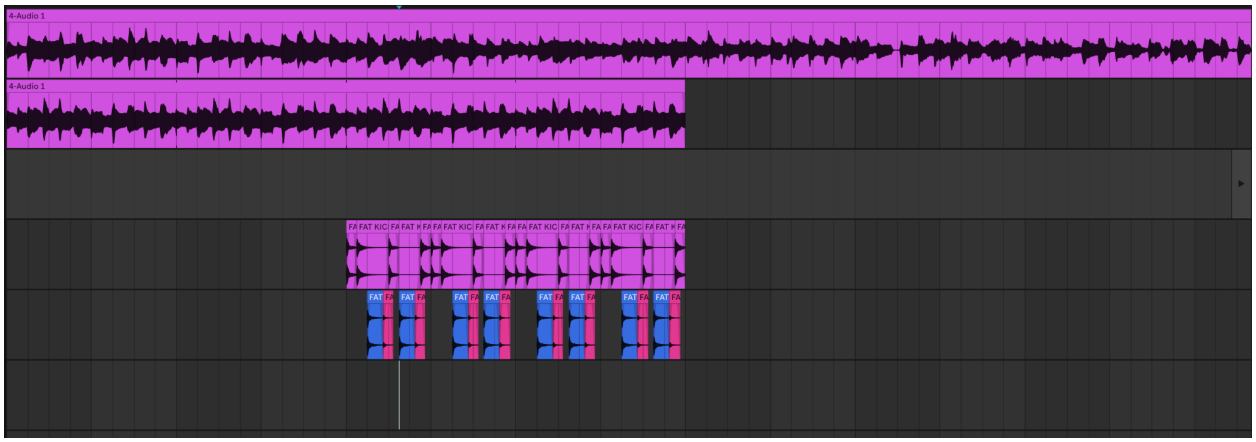


Figure I: Ableton Live session for first song demo

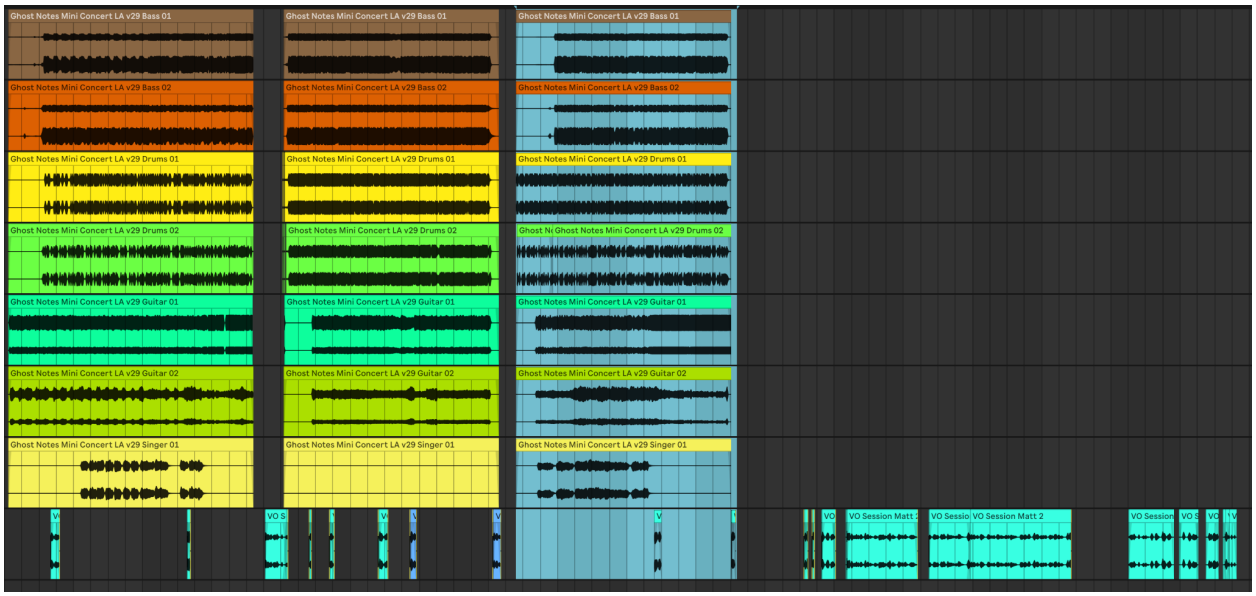


Figure J: final Ableton Live session, including all voiceover cues

2. Visuals

Figure K: mood board and style guide, created in Photoshop

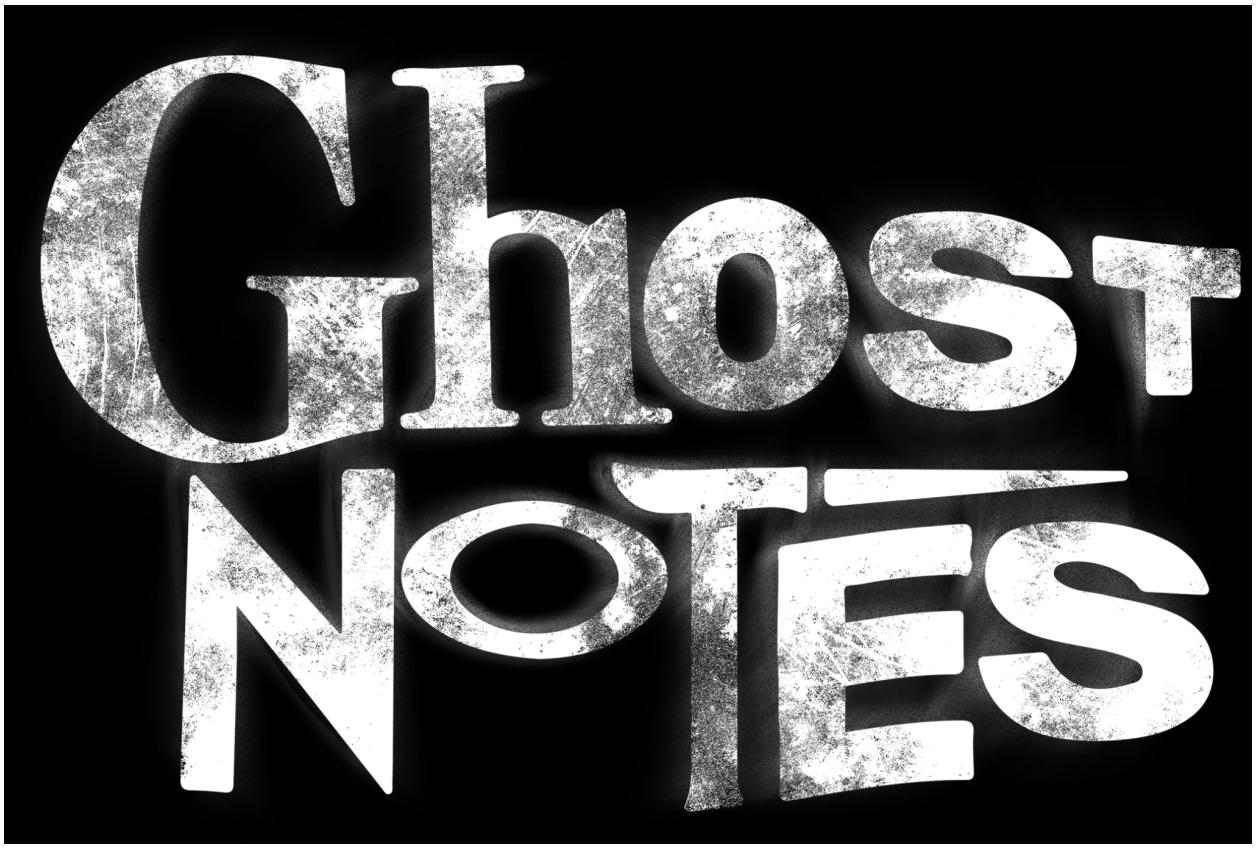


Figure L, M, N: logo iterations. Styled from pasting magazine clippings together, to represent the coming together of different genres and styles. Created in Photoshop and Illustrator

3. Game Development

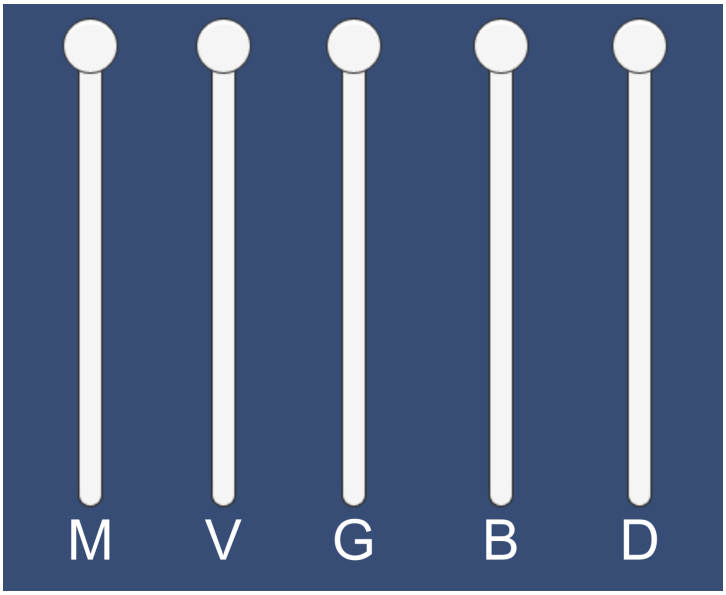


Figure O: early gameplay prototype in Unity

Figure P: early narrative prototype in Unity, with footage edited in Premiere

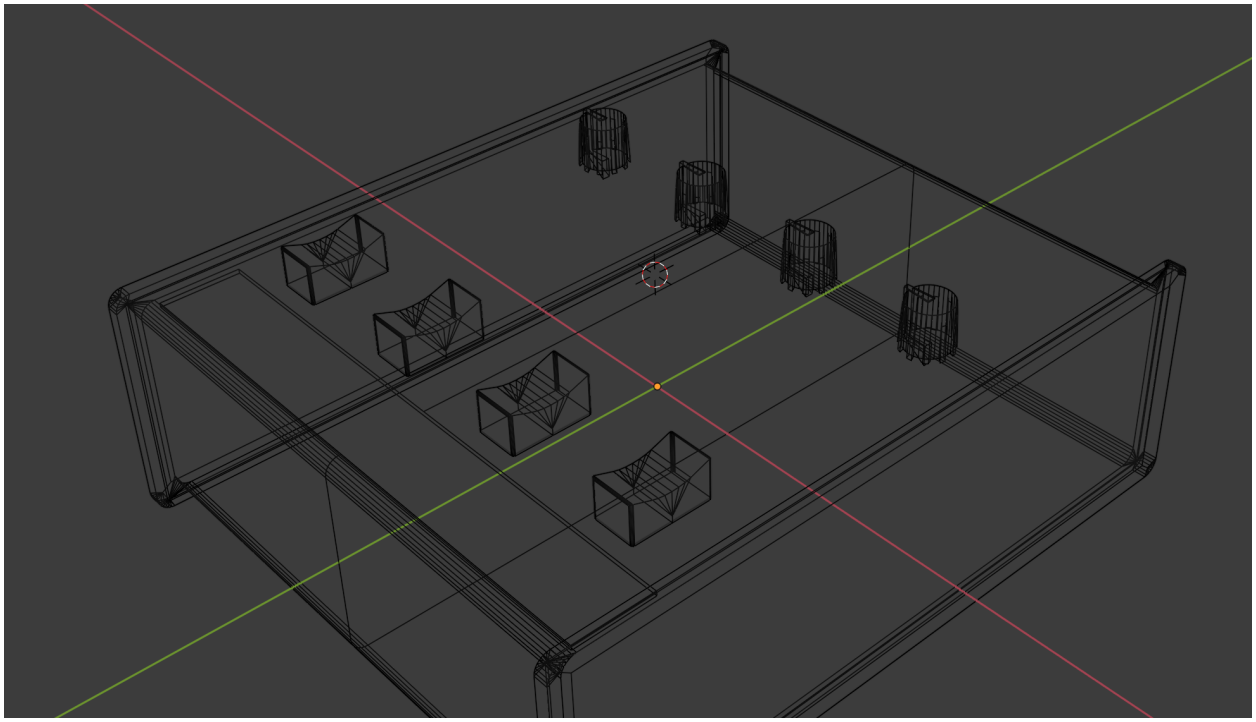


Figure Q: structural wireframe for audio mixer created in Blender

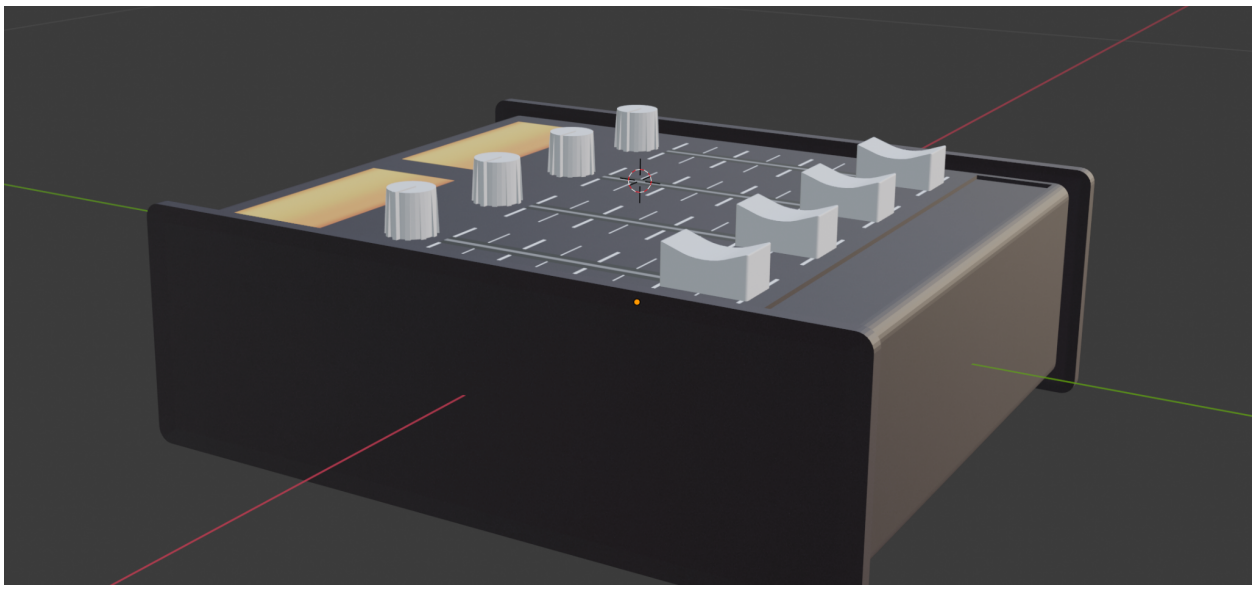


Figure R: texture map, first draft in Blender

Figure S and T: first iterations of narrative scenes, rendered in Adobe Dimension

```
if (song == "Bliss") {

  // AudioSource silentAudioSource = gameObject.AddComponent<AudioSource>();
  // silentAudioSource.clip = Resources.Load("PunkTrack_Master") as AudioClip;
  // silentAudioSource.outputAudioMixerGroup = silentMixer;
  // silentAudioSource.Play();


    AudioSource voiceoverAudioSource = gameObject.AddComponent<AudioSource>();
    voiceoverAudioSource.clip = Resources.Load("Bliss_Voiceover") as AudioClip;
    voiceoverAudioSource.outputAudioMixerGroup = voiceoverMixer;
    voiceoverAudioSource.Play();



        if (currentSinger == 1) {
          AudioSource singerAudioSource = gameObject.AddComponent<AudioSource>();
          singerAudioSource.clip = Resources.Load("dummyAudio") as AudioClip;
          singerAudioSource.outputAudioMixerGroup = singerMixer;
          singerAudioSource.timeSamples = voiceoverAudioSource.timeSamples;
          singerAudioSource.Play();
        }

        if (currentGuitar == 4) {
          AudioSource guitarAudioSource = gameObject.AddComponent<AudioSource>();
          guitarAudioSource.clip = Resources.Load("Bliss_Guitar01") as AudioClip;
          guitarAudioSource.outputAudioMixerGroup = guitarMixer;
          guitarAudioSource.timeSamples = voiceoverAudioSource.timeSamples;
          guitarAudioSource.Play();
        } else if (currentGuitar == 5) {
          AudioSource guitarAudioSource = gameObject.AddComponent<AudioSource>();
          guitarAudioSource.clip = Resources.Load("Bliss_Guitar02") as AudioClip;
          guitarAudioSource.outputAudioMixerGroup = guitarMixer;
          guitarAudioSource.timeSamples = voiceoverAudioSource.timeSamples;
          guitarAudioSource.Play();
        }


        // bassAudioSource = gameObject.GetComponent<AudioSource>();
        if (currentBass == 7) {
          // bassAudioSource.clip = bass01;
          // Debug.Log("Works!");
          AudioSource bassAudioSource = gameObject.AddComponent<AudioSource>();
          bassAudioSource.clip = Resources.Load("Bliss_Bass01") as AudioClip;
          bassAudioSource.outputAudioMixerGroup = bassMixer;
          bassAudioSource.timeSamples = voiceoverAudioSource.timeSamples;
          bassAudioSource.Play();

        } else if (currentBass == 8) {
          // bassAudioSource.clip = bass01;
          // Debug.Log("Works!");
          AudioSource bassAudioSource = gameObject.AddComponent<AudioSource>();
          bassAudioSource.clip = Resources.Load("Bliss_Bass02") as AudioClip;
          bassAudioSource.outputAudioMixerGroup = bassMixer;
          bassAudioSource.timeSamples = voiceoverAudioSource.timeSamples;
          bassAudioSource.Play();

        }

        if (currentDrums == 10) {
          AudioSource drumsAudioSource = gameObject.AddComponent<AudioSource>();
          drumsAudioSource.clip = Resources.Load("Bliss_Drums02") as AudioClip;
          drumsAudioSource.outputAudioMixerGroup = drumsMixer;
          drumsAudioSource.timeSamples = voiceoverAudioSource.timeSamples;
          drumsAudioSource.Play();
        } else if (currentDrums == 11) {
          AudioSource drumsAudioSource = gameObject.AddComponent<AudioSource>();
          drumsAudioSource.clip = Resources.Load("Bliss_Drums01") as AudioClip;
          drumsAudioSource.outputAudioMixerGroup = drumsMixer;
          drumsAudioSource.timeSamples = voiceoverAudioSource.timeSamples;
          drumsAudioSource.Play();
        }



      }
```

Figure U: C# code snippet from "ChangeMusicians.cs," which loads in a different audio track depending on what musician a user chooses.

4. Story Production

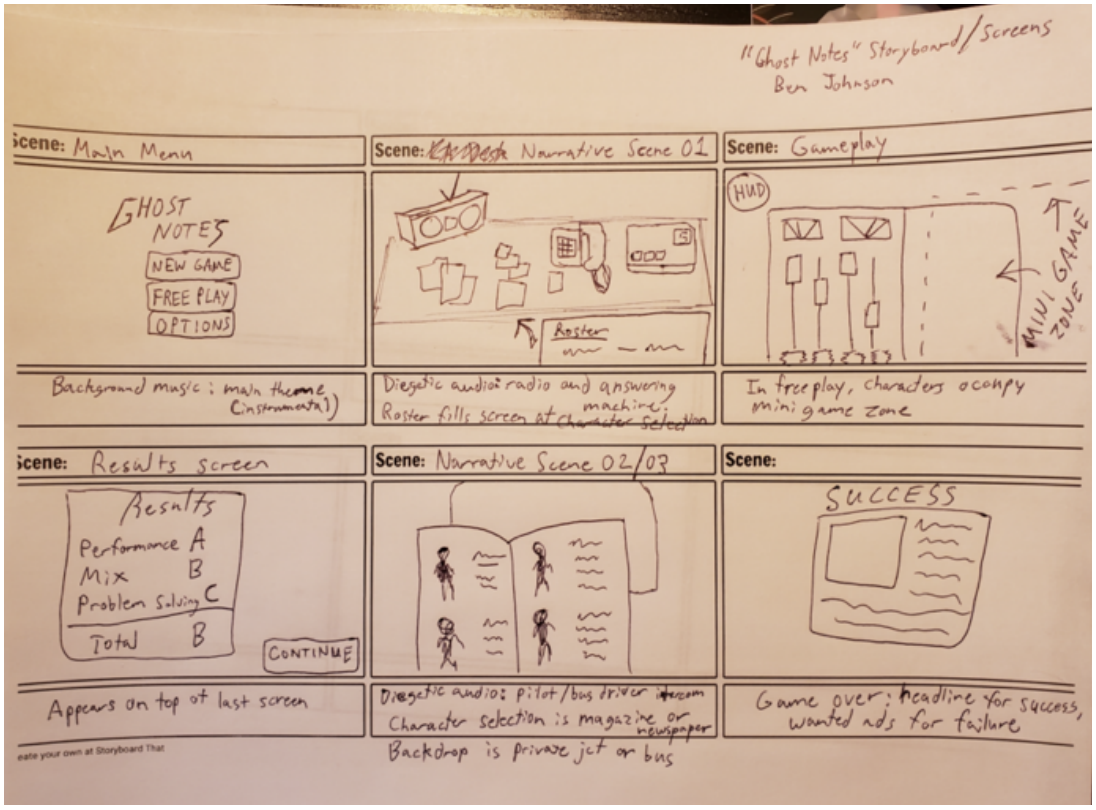


Figure V: initial storyboard

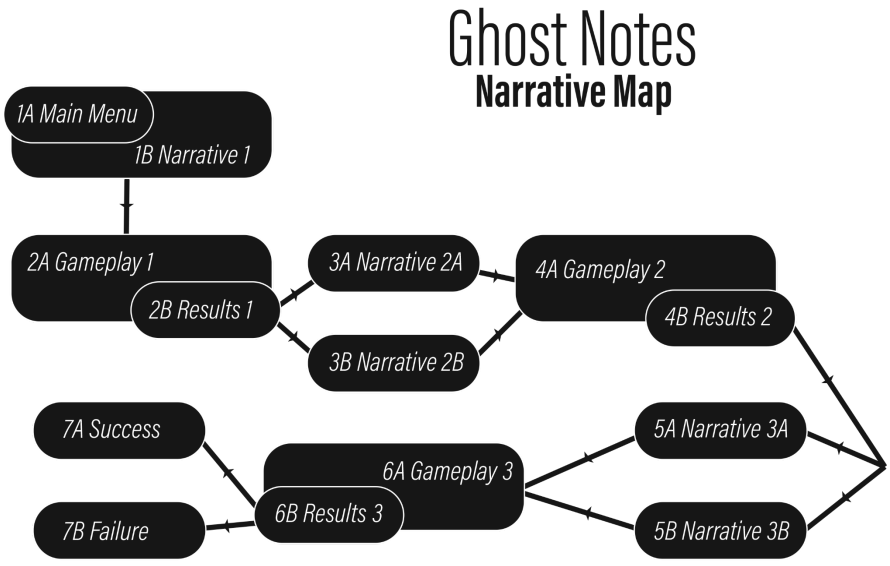


Figure W: narrative map

INT. CONCERT VENUE - MIXER VIEW

WILL MILLER

Here's the mixer, each instrument track is labelled at the bottom. At the top of the board you have your meters, basically you'll want to adjust your mix with the red faders to make sure that the meter on the right matches the one on the left.

(pause)

You can also change where each instrument sits in the mix by adjusting what's called the "stereo spread." Just turn the blue knobs to point towards where you want each instrument to go.

(pause)

There will be times in the show where a musician will play a solo, and it might be a good idea to turn down all other instruments to emphasize that particular section. A flashing light will appear over the label of the track that is about to play a solo, and will blink five times to indicate the start of the solo.

(pause)

There might be some technical issues that go wrong with the show that you might have to deal with. Keep your eye on the space around each side of the mixer and be ready to deal with problems as they occur. If you're too busy with the mix, hit the walkie-talkie button to get another crew member to deal with an issue for you.

(pause)

Alright, I think that's it! Break a leg!

INT. CONCERT VENUE - MIXER VIEW

STAGE MANAGER

Here's the mixer, each instrument track is labelled at the bottom. At the top of the board you have your meters, basically you'll want to adjust your mix with the red faders to make sure that the meter on the right matches the one on the left.

(pause)

You can also use the blue knobs to change where each instrument sits in the mix, but we won't worry about that right now

(pause)

There will be times in the show where a musician will play a solo, and it might be a good idea to turn down all other instruments to emphasize that particular section. I'll cue you in when these sections are about to happen

(pause)

There might be some technical issues that go wrong with the show that you might have to deal with. Keep your eye on the space around each side of the mixer and be ready to deal with problems as they occur. If you're too busy with the mix, hit the walkie-talkie button to get another crew member to deal with an issue for you.

(pause)

Alright, let's try it out

INT. CONCERT VENUE - SOUNDCHECK

STAGE MANAGER

Ok this song starts with bass and drums, and I'll cue you in when the guitar is about to start. Ready? and three two one

(pause)

Ready guitar

(pause)

Level off those faders a bit

(pause)

Drum solo incoming, bring down the guitar and bass

(pause)

Oh no the bass is unplugged, plug it back in!

(pause)

Great job! I think you're ready. Let's get to work

Figure X: tutorial script, draft four vs. Figure Y: draft six rewrite

Reflection

I have wanted to create a project that incorporates music into an interactive narrative for quite some time, and I feel that *Ghost Notes* has allowed me to do that in a meaningful way. There was a bit of a learning curve at first due to my unfamiliarity with the Unity engine, but once I got into the creative process I believe I was able to use the tool effectively. Overall I feel I have created a successful project that I am very proud of.

One issue I dealt with was that a lot of my project was contingent on the schedules of other musicians. I had several artists and managers express interest in the project, but only a small number were actually able to submit materials for the final project. Though I had somewhat planned for this and overcompensated in the number of artists I initially contacted, I now know that this was not enough and it would be important to contact even more artists if I were to recruit for any future projects.

In terms of the coding architecture, I believe my computer science background prepared me to fully think through how each scene and feature would work before I wrote a single script. However, at times I felt this was also a hindrance as I began overthinking simple solutions. For example, in a later build of the game I encountered a bug where the individual instrument tracks became out of sync causing playback to be jarring and borderline unlistenable. I began sketching out ideas on how to overcome this issue, and at one point I had thought through a script that would work as a “stem manager” and constantly load sections of a song on playback.

This was all unnecessarily complicated though, as a few web searches revealed a better strategy: time sample management. The solution that I implemented (and what is still a large part of the audio engine in the final build) is a function that initializes one of the instrument tracks and immediately sets all other tracks to the same sample rate and begins playing all audio at the same time. This allows for the audio to start and remain in sync, as the play function is not reliant on when the asset loads into the scene. This cut down on load times as well, as the scene compiler is now able to load all non-audio assets before start and pull in all audio files once the scene begins.

I also learned a lot about game design during this project. When conducting user tests with the first iteration of the tutorial screen, I was able to gain insight into how a typical user viewed a sequence that I coordinated. Though certain aspects made sense to me, they didn’t always translate. In the implementation of the second version of this scene, I tried to base my decisions more on what I observed in these user tests and I believe that has made it more successful.

Throughout this entire process, I believe I have become more user-centered as a designer. Though the tutorial sequence is a clear-cut case of influence coming from a specific source of feedback, I also integrated some changes that I felt were more user-friendly despite not hearing that specific preference in any written or verbal feedback.

A good example of this is my removal of certain game mechanics that would penalize the player for mixing in an unconventional way (i.e. drums much louder than vocals, bass overpowering guitar etc.) My reasoning for this change is actually quite simple: people mix audio differently. Though I have known this fact since I was a music production student, the user test sessions solidified this fact to me.

Though the “correct” way to mix involves everything being somewhat equal, I decided not to prioritize this because I see the value in allowing users to experiment and decide what sounds good to them. I believe this is the core of what makes Ghost Notes special; it’s the opportunity to customize the source material into something the user wants to hear, whether it’s a bass-only mix or a heavy metal guitarist playing with a soul musician.

Credits

Software Used
Unity, Blender, Ableton Live, iZotope Ozone, Photoshop, Premiere Pro, Illustrator, Adobe Dimension, Atom, Visual Studio, Zoom

Written/Directed/Produced by
Music and Lyrics by
Ben Johnson

Cast
Will Miller Daveed Shannon
Ghost Notes Ben Johnson
Stage Manager Matt Harrell

"Enrique" guitar track performed by Enrique Juaregui
"Chidi" bass track performed by Charles Chidi
"Itunu" drum track performed by Itunu Joe
All additional tracks performed by Ben Johnson

Audio Mixer 3D Asset
modeled with John Valle

Beta Testers
Reed Pake
Wen Guo
Brianna Boucher

Usability and Design Feedback
Qian Xu
T'Keya Davy
Carol Ann Friday
Alynda Pratt
Chandler Colclough
Michael Boyd
Jasmine Simmons
Meagan Chalmers
William Moner
Jeffrey Cullen-Dean
Juan Ho Hernandez
Olivia James

Interpolates Concepts and
Code Architecture from
John French
Brackeys
Unity Learn
Comp-3 Interactive

Jason Weimann
Simmer.io

Unity Effects and Extensions
“Light Rays 2D” by Andrii
“3D Music Visualizer” by Bad Raccoon
“Unity UI Extensions” by The Unity UI Extensions Project

2D Assets (most available on Adobe Stock)
"heaven, sunset over the clouds 3D rendering” by ustas
"Dust and scratches design, black abstract background” by golubovy
“Aircraft windows” by Tartila
“Paper Texture 03” by BLKMARKET
“Textured plastic background” by Unkas Photo
“Adhesive tape” by picsfive
“Lock” by Adrien Coquet
"INNATE Logo” by Natalie Oldani
“Don’t Forget to Subscribe” by Meg Boericke
“Leaf Logo” by Madeleine Horrell
“Herd Logo” by Matt Harrell

3D Assets (available on Adobe Stock and Turbosquid)
“Walkie Talkie” by Billy Reiter
“Headphones” by Wolfgang Biebach
“Bare metal waved plates material” by Substance Source
"Mesh penholder with office accessories” by HQ3DMOD
“Classic pen" by Dmitrii Ispolatov
“Stationery papers” by Adobe
“Compact audio cassette” by Francesco Milanese
“Telephone Office” by Adobe
“Vintage radio” by Steve Gonopolsky
“Office desk” by Marcus
“Drink can” by Adobe
“Notepad” by Adobe
“3D Flightcase” by resl