University of Southern California
School of Cinematic Arts
Interactive Media and Game Division

Ascension: An Analysis of Game Design for Speech Recognition System Usage and Spatialized Audio for Virtual Reality

By Crystal Chan

A Thesis submitted to the School of Cinematic Arts in partial fulfillment of the requirements for graduation with a Masters of Fine Arts Degree

Degree Awarded May 2019

Acknowledgments

Maureen McHugh, Robert Nashak, Tom Smurdon
Shane Kneip, Cameron McNair, and the rest of the Survios Audio Team
Carl Schnurr, Jeff Watson, Jane Pinckard
Dennis Wixon

Mari Kyle and Kira Kyle
Space Shuttle Ascension Team
USC IMGD MFA Cohort Class of 2019

Mom, Dad, 玲玲姑媽

Table of Contents

Abstract

Ascension: An Analysis of Game Design for Speech Recognition System Usage and Spatialized Audio for Virtual Reality details the process of design for the virtual reality experience *Space Shuttle Ascension*. This experience revolves around including a speech recognition mechanic and making it feel natural and organic. *Space Shuttle Ascension* is an investigative virtual reality experience in which the player must manipulate time and interview virtual characters in order to find out why the Space Shuttle Ascension has crashed. There are four modes of interaction: gaze-based interaction, voice interaction, object interaction, and time manipulation by gesture. The goal of this thesis is to create a satisfyingly reactive audio system using both technical audio programming skills and creative sound design. While this project has other goals regarding narrative, worldbuilding, and character development, this document will specifically be focusing on the audio and speech recognition system.

Keywords: Speech Recognition, Game Design, Game Audio, Audio Spatialization, Sound Design, Virtual Reality

Introduction

As someone who works in audio, I have been taught that good audio is essentially invisible audio-- something that blends so perfectly in the background that the less the audience notices, the better it is. With *Space Shuttle Ascension*, and the speech recognition system in particular, my goal is to bring audio to the forefront, while still maintaining a highly immersive experience, making design choices that allow the players to feel as though their voice input not only makes sense in the world but has a significant impact. The virtual characters react realistically and believably in ways that feel completely natural and organic. This system adds to *Space Shuttle Ascension* as an experience overall because, ultimately, it is a narrative piece: a story, and, at the very core of it, a story about people.

Beyond challenging what it means for media to have good audio, I hope to help pave the way for more unconventional methods of interaction in games. Voice input oftentimes feels clunky and awkward, and, certainly, there is a difficulty in teaching players that speaking is an integral part of the experience on top of creating an environment where they feel safe and comfortable enough to speak at all.

As an industry, I feel as though we have grown comfortable with the familiar. We have our *Call of Duty*'s and our *Madden*'s, our *Uncharted*'s and our *World of Warcraft*'s, our *The Sims*' and our *Mario*'s. The list goes on for genres, but we can expect our players to intuitively pick up the dual joystick controllers or the keyboard and mouse. The beginnings of virtual reality for the mass public is the perfect timing and opportunity to introduce more new concepts as players are open to innovation and exploring different ways to play.

State of the Literature

In this section, I will provide research from academic papers as well as pieces of media that have helped to set a foundation for the rest of my thesis paper and the project.

Prior Research

Simon Alexanderson's *Mimebot—Investigating the Expressibility of Non-Verbal Communication Across Agent Embodiments* outlines how the designed behaviors of robots and their gestures can create more engaging and approachable interactions with humans. Alexanderson and his team tested readability of their Mimebot (their artificial agent animated through performance motion capture of a professional mime actor). Although *Space Shuttle Ascension* relies on speech recognition, arguably the exact opposite of non-verbal communication, this study demonstrates the value of non-verbal communication in terms of facial and body animations, which weighed into our spending production time to include animations of both the face and body of our virtual characters in order to improve the player's conversational experiences with them.

Coda: An Alliance of Human and Machine revolves around the idea of machine cognition and human cognition being complementary (one is not replacing the other) and supports the use of computer technology to help free up the mind's capacity, allowing more room for creativity. This paper specifically focuses on IBM researchers and their plans for Watson, their speech artificial intelligence (AI) software which is being used in this thesis project. It puts a lot of skeptics' fears of AI taking over humans and their jobs to rest, emphasizing that their efforts are to one day make machines "expert, trusted [advisors] to humans [...] to help us reason over

human-created data." Automation of menial, repeated tasks will only help us be more innovative and imaginative creatures. That is not to say that the abuse of this powerful technology cannot occur, but fearing it would be a waste and also be detrimental to our growth.

In *Voice Input for Collaborative Systems*, Marie Meteer explains the critical problems that need to be solved in a system that is controlled solely by a user's voice (no eye contact or hand gestures), focusing on the system needing to understand context and finding a balance between system-initiated and user-initiated interactions. Some important terminology include:

- *System-initiated*: Systems that prompt or question users to elicit specific information. This is considered more simple, limiting the possible user responses, but has better performance, since the system knows to target expected responses.
- *User-initiated*: Systems that allow the user to freely ask questions or give commands. This allows for more flexibility, since the user is not restricted in terms of what they can say or when to interact, but the performance of the system relies heavily on how well it has been designed.

Ryan Wistort of the MIT Media Lab illustrates his findings from exploring the development of "highly expressive yet inexpensive robotic characters" in *Only Robots on the Inside*. He boils these findings down to several design concepts for creating more lifelike characters, some of which include:

- Deliver on expectations. While he mainly relates this concept back to the physical form of the robot communicating its affordances, we took this and applied it to virtual characters. In *Space Shuttle Ascension*, we gave each of the four crew members in the

space shuttle roles: Commander, Engineer, Doctor, and Biologist. This allows the player to understand which characters might be able to answer which types of questions.

- Eyes. Even the mere presence of eyes on a robot significantly enhances the feeling of social presence and connection. Wistort explains the importance of actuated pupils here, explaining that the more realistically the eyes operate, the stronger the connection, which informed our decision to work on having our virtual characters maintain eye contact with the player while in conversation.
- The illusion of thinking. Winstort argues that the ability to convey emotions is useless if "no one appears to be behind the wheel." A happy accident that occurred was that the latency from when player input is detected to the moment the correct character voiceline plays emulates a person who is thinking of a response within a conversation (refer to IBM Watson Application section for more information).
- Engagement. This concept primarily deals with body language of the robot. Robots seem more lifelike if they demonstrate awareness of their surroundings and a curiosity and fascination for the things around them. In *Space Shuttle Ascension*, in addition to eye contact, the character with which the player is speaking will turn to face them and convey an interest in the conversation.

Prior Art

This section will include some comparables from existing media (games, films, research studies). The comparables have influenced *Space Shuttle Ascension* in some shape or form, which could be narratively, technologically, conceptually, and/or artistically. I will also explain each comparable's similarities to the project and how *Space Shuttle Ascension* builds upon the foundation established by these similarities.

Adventures of an Adolescent Trash Barrel Robot is a short video documentation of the extensive research of Wendy Ju and Leila Takayama regarding human-robot interactions. They have essentially found that it is important for robots to not only be technically efficient in practically completing their tasks, but also socially efficient. To be "socially efficient," robots need to seem engaging and approachable to people. In the video, although the trash can is actually remotely controlled and not automated, it is meant to demonstrate how people (who are unaware of it being controlled by another person) are more likely to engage with it if it behaves as if it had some sort of personality: responding to someone waving at it or going away when someone shoos it. One person went so far as to ask "are you okay?" to it after it had walked into the grass and fallen over. In terms of *Space Shuttle Ascension*, this is an important concept to keep in mind while trying to create sympathetic virtual characters to which we want the players to feel connected and eager to interact.

Fig. 1 - In-game screenshots of *Star Trek: Bridge Crew* (Photo credit: Holly, 2017)

Star Trek: Bridge Crew is a VR game released in 2017 in which the player explores an uncharted sector of space in search for a new home world. A unique selling point of the game was Ubisoft's partnership with IBM Watson, having integrated voice commands that the solo captain player can yell at their NPC crew to delegate tasks accordingly and making the most use out of the AIs. The setting (a spaceship), the platform (VR), and the modes of interaction (voice commands as seen in Fig. 1) line up very well with what we are trying to accomplish in *Space Shuttle Ascension*. While *Star Trek: Bridge Crew* is a game in the most traditional sense and has set voice commands that the player can look up and memorize, we aim for *Space Shuttle*

Ascension to be more of an experience that is heavily driven by narrative with voice interactions that instead feel like organic conversations with the virtual characters.

Fig. 2 - Image of the different types of robots in *Wall-E* (Photo Credit: The Wall-E Builders, 2008)

Wall-E is a Pixar animation released in 2008 set on an abandoned earth driven to waste by humans. The movie follows the journey of Wall-E who finds what seems to be the last plant on the planet and after encountering a slew of different robots and making it onboard the Axiom, brings humankind back to earth to restore the planet and grow life again. *Wall-E*, like most Disney Pixar films, tackle relevant subject matters (environmental waste, technology, religion) in a very accessible way to a diverse audience. In addition, the sound design in the film is exceptional, reminiscent to the birth of video game consoles, where games did not have nearly as much processing power or storage as nowadays and were very limited in resources. The sounds are all highly polished and expertly crafted, yet seem simple. Each robot has its own personality even though they never speak. Their entire characters can be expressed through their foley. As the audio lead, I aim to be very deliberate in my sound design for the virtual characters since the narrative is very character-centric.

Purpose of the Study

As virtual reality is being made more accessible to the mass public, the way that we have grown to know games will change. This paper and the project are heavily influenced by the research and works of Dr. Leila Takayama, a cognitive and social scientist studying human-robot interaction. I draw concepts primarily from her paper *Expressing Thought: Improving Robot Readability with Animation Principles*, where she collaborated with Wendy Ju, a fellow scientist, and Doug Dooley, who works at Pixar Animation Studios, in order to test how forethought and reactions (core principles in animation) in robots allow them to seem more readable and approachable to the humans with which they interact. Robots typically refer to physically embodied systems, while on the other hand, agents describe a software system (Kiesler). However, with virtual reality, we can begin blurring the hard line that distinguishes the two. While the virtual characters in *Space Shuttle Ascension* are agents in the sense that they are purely software-based and have no physical embodiment whatsoever, what happens when the player is transported into sharing the same virtual space as them? Virtual reality offers interesting uncharted territories for us to explore and see how much of the human-robot interaction research can be applied.

Methodology

In order to create a satisfyingly reactive audio system, I built a speech recognition system, set up an interactive music system, and utilized 3D audio plug-ins in *Space Shuttle Ascension*. This section will primarily outline the technical aspects of the project, how IBM Watson is used for the speech recognition system, Audiokinetic's Wwise for interactive music and general audio design, and Oculus Audio for 3D audio.

Designing for Speech Recognition

As of today, we are still designing the way that the player starts and ends a conversation with a character. Some ideas we have include: saying the character's name or title, greeting them in their native language, tapping them on the shoulder, looking at them for a set amount of time, or any combination of these. Figuring out how we should initiate a conversation with a character also leads us to thinking about how we should create a more controlled method for the player to talk and how this system handles anything that is said that does not register as an "Intent," words and/or phrases that IBM Watson will specifically look for as the player interacts within the experience.

Continuing on the idea of a controlled method for the player to talk, we made the design decision to limit the player to being able to have a conversation with one character at any given time. Once the player starts a conversation with one character, they are locked into the conversation until they leave it by saying bye to the character. In terms of controlling how the system handles player input that we have not implemented as Intents in the game, we made a big narrative decision to set the experience as a simulation. Therefore, anything the player says that

does not correspond to any of the voicelines we have written will cause the simulation to glitch a bit and Maxwell to prompt the player that what they said exists outside of the world of this simulation.

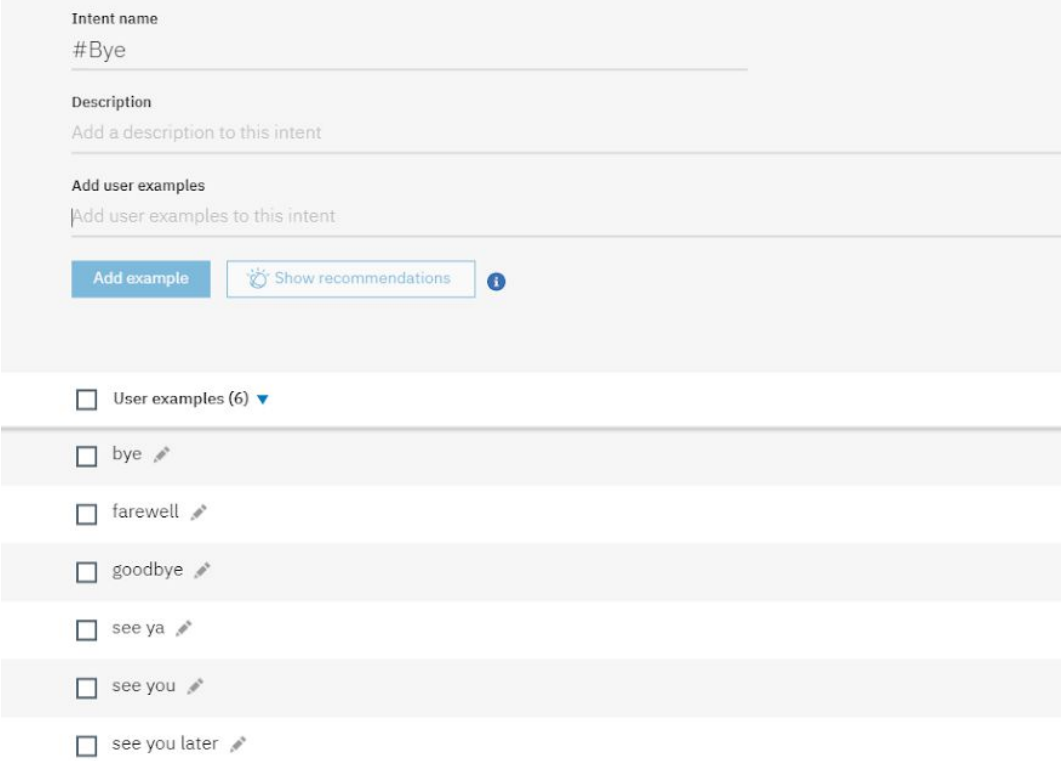


Fig. 3 - Screenshot of an Intent set up in IBM Watson Conversation. (Photo Credit: Crystal Chan)

A recurring problem we have while showing demos to people has been the environment being too loud with too many other background conversations happening, which causes the speech recognition system to behave poorly. The game runs flawlessly when it is quiet and only one person is speaking. However, once it picks up multiple voices, it starts to fall apart since it does not know which voice is the player's voice and random words from other conversations get added into the Speech to Text parsing or it finishes parsing much later than when the player stops talking because it thinks the other voices are a part of the input.

Some possible solutions we are looking into are: setting up a threshold for the volume of the microphone for voice input so that nothing below a certain volume level (like background conversations) get parsed, requiring the player to hold down a button or move the controller to their ear as if using a comm radio (or both) whenever they are talking in order for the game to recognize that they are speaking so that all other speech or noises are automatically ignored at any other time, creating a sort of calibration system where the player requests a radio check to test microphone clarity and the game responds with a result, or training the player to adhere to a system such as always saying 'Over' when ending their voice input.

For the thesis show, we will be utilizing the tall wooden structures that serve as fake walls in film productions that are already in the soundstage to construct a makeshift booth, in addition to situating ourselves in a corner, in order to mask as much of the noise as possible. The walls will be covered in huge pieces of heavy cloth to further dampen sound. We are also hoping that this booth will provide enough privacy for people demoing to feel more comfortable and less self-conscious about using their voice and talking, since it is such an unconventional mode of interaction within games.

Because voice input is not a very mainstream type of interaction and yet such a crucial part of *Space Shuttle Ascension*, our team has been keeping the mechanic at the front of our minds whenever we are designing. We spent a long time drafting up our opening sequence and making iterations on the script to create an extremely simple scene that will introduce only the most important elements that the player needs to know right away: Maxwell and voice input interaction. One of our thesis advisors, Alex Schwartz, who was the Founder/CEO of Owlchemy Labs, explained the importance of introducing the main mechanic of a game within the first ten

seconds. For example, in *Job Simulator*, the player is prompted to take a business card that is handed to them, teaching them to pick up any sort of objects they see within the world. In our case, we have the player verbally countdown "3, 2, 1" and then say "Liftoff" to launch the shuttle. Although there are no conversation-based interactions, we indicate to the player that this game will require them to speak.

Fig. 4 - Screenshot of the opening sequence of *Space Shuttle Ascension*. (Photo Credit: Crystal Chan)

We do not introduce the player to a more formalized tutorial until they finish that opening sequence and are put into the cockpit of the shuttle. Here, we solidify the fact that Maxwell is the player's guide by having him explain different mechanics like gaze-based interactions, grabbing of objects, and time manipulation. We are allowing Maxwell to demonstrate how these mechanics work, so that players can learn by copying exactly what he does. Another stepping stone towards having a conversation with the ship's crew is simple voice commands. We are

continuing the very basic mechanic that we taught the player in the opener, but this time in the main environment and in a way that is more in tune with being a part of the world. We will introduce Houston, the ship's computer, which will operate very similarly to smart devices such as Alexa, Siri, and Google Assistant. Through Houston, the player can do very simple tasks such as turning the lights on and off or things that are a little more complex like pulling up audio logs or a diagram of the ship. As they become more acquainted with triggering events through their voice input, the conversation mechanic should feel more organic.

Something that surprised us while developing the opening sequence and transitioning to setting up the basics of the conversation system was that the conversations actually feel a lot more natural than the voice commands. There is a bit of latency that occurs as IBM Watson takes a pause to register that the player has stopped talking and then triggering the Intent and the events that are associated with it. It is noticeable when the player says the countdown and liftoff, and it can be a bit confusing, since the half second of silence can cause people to think that the game did not recognize their input. However, the latency is virtually undetectable when the player reaches the conversation mechanic because it is completely natural for pauses to occur in conversation as they wait for the character to "collect their thoughts and form an answer" in response.

We have been brainstorming with our team to hopefully alleviate this problem by providing some sort of visual representation or feedback for the player to indicate that the game is registering voice input. Smart devices such as Amazon Echo and Google Home experience the same problem with latency, but the lighting up of the device or appearance of a few dots communicates to the user that their input has been received and a response will be provided

shortly. We do not want something extremely distracting since it would be occurring quite often, every time the player speaks. An idea is to have the player see their breath when they are talking on the launchpad in the opener. Another solution might be showing waveforms, either as a UI element or coming from the player.

Alternative Speech Recognition Systems Review

Before delving into the technicalities of IBM Watson, I will detail our process of scoping out the current market of speech recognition systems and how they compared with one another, and, ultimately, why we decided to use IBM Watson. While looking for a speech recognition system to integrate into the Unity game engine, I referred to Dioselin Gonzalez's *Speech Recognition and VR* article on Unity Blog. As a VR Principal Engineer at Unity Labs, she mentions a few speech-to-text tools to explore: Wit.ai, Windows dictation recognition, Google Cloud Speech, and IBM Watson.

Wit.ai exists as open source code, so the system itself changes and grows based on the activity from the community. While it is completely free, including for commercial use, and does not have a strict rate limit, Wit.ai seemed to require a lot of time spent reading the documentation and tweaking the existing code in order for it to work well within our project.

Windows dictation recognition utilizes the Microsoft Windows speech recognition system that powers Cortana. Unity integration with this system is extremely seamless, with the only step being to import the `UnityEngine.Windows.Speech` namespace. That immediately grants access to different types of speech application programming interfaces (APIs) such as `DictationRecognizer`, which is essentially a speech-to-text system, and

`KeywordRecognizer`, which is used specifically for triggering events through voice commands. Pricing also did not seem to be an issue, considering it is a system that is so well integrated into Unity. However, Windows dictation recognition seemed to revolve mostly around voice commands rather than sustaining conversations with virtual characters and seemed to fall apart once more complex conversational input was tested.

Google's Cloud Speech-to-Text is part of the system behind Google Assistant, the speech recognition AI in Google Homes and Google Pixel phones. This system is a much better fit for the things we are trying to accomplish in *Space Shuttle Ascension*. It is widely tested and used, with extensive documentation, and extremely powerful and robust. Google Cloud uses machine learning technology to train its system and improve accuracy. A unique feature is their pre-built models, which can be useful if we needed performance optimization for specifically voice commands (command_and_search model) or video transcription (video model). At this tier of robustness, price starts becoming a more significant factor. Google Cloud Speech-to-Text API comes with 60 free minutes of usage every month. After 60 minutes of processed audio, it costs $0.006 for every 15 seconds of processed audio. Google Cloud was a strong contender in our decision process.

Before outlining the features of IBM Watson, I would like to talk a bit about Amazon Lex, which was also an option we considered (and is not mentioned in the Unity Blog article). Amazon Lex powers Alexa in the Amazon Echo products. We met with representatives from Amazon who became interested in *Space Shuttle Ascension*. They offered us a deal: free support from Amazon engineers in the form of being added to the Slack channel for their development team, and, in exchange, we develop our game using Amazon Sumerian (their game engine for

developing virtual reality experiences) and Amazon Lex. While this was a highly enticing offer, we ended up going in another direction since developing with Amazon Sumerian and integrating Amazon Lex would essentially mean that we were assisting them in creating the documentation for those services. We felt as though we needed all the time we could possibly have dedicated to production instead.

Finally, IBM Watson seems to be on par with Google Cloud in terms of performance and documentation and also uses machine learning to train its system. A particularly interesting feature of IBM Watson's Speech to Text service, though, is its ability to register stutters like "uh" or "um" as `%HESITATION`. This allows for an entirely new facet in the design of the game to make the virtual characters more lifelike. Although we did not have enough time to implement reactions from the characters to this type of input, it serves to be an option when we continue development. In addition to Speech to Text, IBM Watson also has a Conversation service that is targeted for experiences like *Space Shuttle Ascension*. The interface is incredibly easy to learn and use, which saved a lot of time and frustration. Pricing is also rather similar to Google Cloud. IBM Watson's Speech to Text service allows for the first 100 minutes for free every month. After that, it costs $0.02 every minute.

In the end, we went with IBM Watson over Google Cloud because of its features catering so well towards the game we are trying to make. IBM Watson has also been integrated in a well-known VR game (the aforementioned *Star Trek: Bridge Crew*), so we had the comfort of knowing that the system can handle similar types of game input and integration into a game engine.

IBM Watson Application

Space Shuttle Ascension's speech recognition system relies on IBM Watson's Unity Software Development Kit (SDK). We are using IBM Watson's Speech to Text and Conversation services. Speech to Text parses the player's voice input into text that is fed into the Conversation service, which then figures out the player's Intent in order for the Unity game engine to play back the correct voiceline in response.

The Speech to Text function is relatively simple in terms of setup. After getting the auto-generated credentials off of IBM Watson's website and including them into the C# script in the Unity SDK, Speech to Text was up and running. On the other hand, the Conversation service takes a bit more setup and requires more maintenance as the project grows in scope. Workspaces are created and hold all the conversation logic needed for the game. We mainly work with Intents, which are continually being added and updated as more voicelines are written.

IBM Watson is smart and does a fantastic job in figuring out what the player says and matching that to an Intent. This is a bit of a double-edged sword. On the one hand, it takes the pressure off our shoulders in having to manually type in every single way a player might say something along with all of its countless possible variations. However, it sometimes does too good of a job. A prime example is the countdown mechanic we use to introduce voice input interaction to the player in the opening sequence. The Countdown Intent is "321" (three two one). It works perfectly fine and flawlessly detects it whenever the player says "three two one." Unfortunately, IBM Watson also matches player input to the Countdown Intent when the player says only "three." Although the prompt is very exact in that it specifically asks the player to say "three two one," edge cases can occur where players end up saying something that is not the

countdown but is close enough to still trigger it, breaking immersion. We have noticed that there is a feature to include "conflicts," which we suspect is specifically for cases like these where we want only the exact phrase to trigger the Intent, but we have not had a chance to test that theory since it is a feature only available for premium versions of IBM Watson.

As a student project, we have been operating on the free/lite plans for the IBM Watson services, meaning we are limited to 10,000 API calls to Conversation and 100 minutes of Speech to Text usage. The usage resets at the start of every month. We never even came close to hitting the limits of the Conversation service, but had quite a bit of trouble with Speech to Text. Because Speech to Text is much more of a background task and is active at the slightest bit of voice input, something as simple as forgetting to disable it when it is not needed while testing in Unity can quickly eat up the minutes at a ridiculously fast rate. We were scrambling to find a solution to that, whether that be to use the 6-month academic license (which did not look too promising when we tried it out) or to bite the bullet and upgrade to a standard plan and pay $.02 every minute of use. Through this process, we found that IBM Watson's customer service is unfortunately slow and, after many weeks since opening the ticket (and then multiple tickets), not very helpful in providing effective resolutions to what seemed like a rather simple problem. We were fortunate enough to win the Kanter Award through the USC Thornton School of Music from one of our team members who is a Music Licensing student and will be using some of the funds to cover for paying for the Speech to Text service.

Furthermore, because we are using the free plan for the Conversation service, we are also limited to one Workspace. Luckily, this limitation has not been an issue thus far. A Workspace is where we set up all the Intents that our game needs to scan for, as well as dialogue branches and

"Entities," (specific types of input such as numbers, locations, or dates), which we have not seen a use for yet. The way we have the game set up works with just a single workspace holding all the information.

An integral part in building this speech recognition system is planning how IBM Watson and Unity communicate with one another. The reason that a single Workspace is sufficient for our use is that we are essentially using all the same Intents for every character. Only in code do we have our game react differently based on the character the player is speaking to. Within the C# script handling the speech recognition system, we have a State Machine setup so that the player can only be having a conversation with one character at any given time; when the player is talking to that character, their voicelines will play when Intents are triggered.

Wwise Application

Next, I will be talking about Wwise and its application in *Space Shuttle Ascension*, detailing how it handled the interactive music system and all of the sounds and voicelines we created.

Audiokinetic's Wwise is an audio middleware. It provides the foundation on which we build the interactive music system. A combination of both vertical and horizontal project structures allows our composer's musical ideas the dynamics they need.

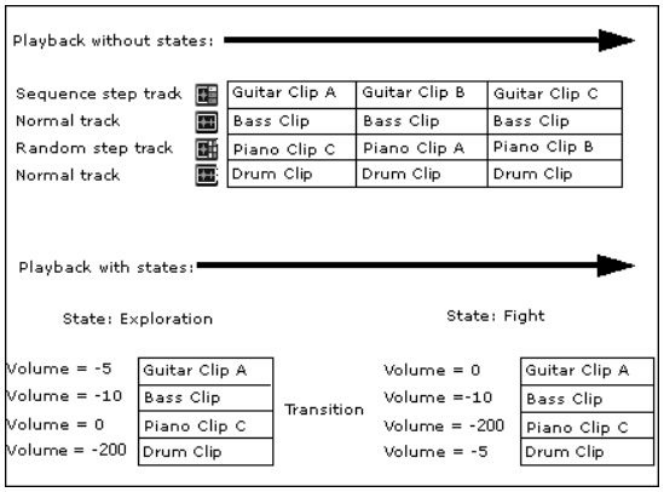


Fig. 5 - Illustration of a vertical project structure in Wwise. (Photo Credit: Audiokinetic)

With a vertical project structure, we have a more complex musical score separated into individual stems that can be triggered on and off. This is useful in situations such as emphasizing important object interactions, causing certain audio tracks to toggle on when an interaction has been completed.

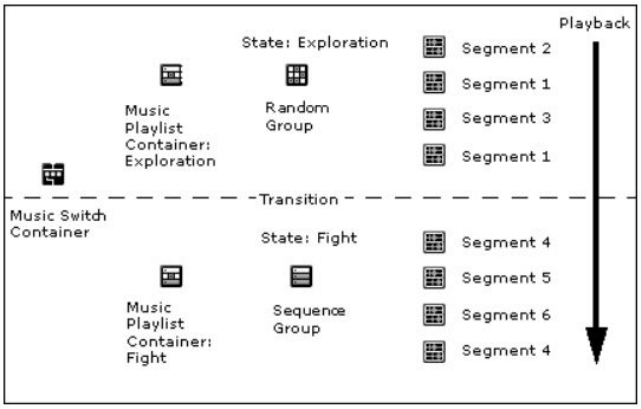


Fig. 6 - Illustration of a horizontal project structure in Wwise. (Photo Credit: Audiokinetic)

The horizontal project structure allows us to have several loopable sections that make up the musical score. In this project structure, we implement real-time parameters that are determined during gameplay to trigger transitions to and from different sections of the score.

This helps the player to understand when they are being moved to a new part of the narrative and communicate to them that the story is progressing forward.

Both vertical and horizontal project structures include the ability to add in cues (entry and exit cues). Cues allow us to set up parts in the music where transitions can happen, making sure every transition occurs on beat and in time.

Space Shuttle Ascension primarily uses the horizontal project structure since it is a narrative driven experience and this type of project structure lends well to communicating a sense of progression. Our composer created musical themes for each character, which trigger when talking to the respective virtual character. As the player asks more questions that are relevant to moving the narrative forward and figuring out what went wrong with the space shuttle, each musical theme also gradually transforms to reflect changes in the game's state.

Fig. 7 - Screenshot of Switches for Conversation Partner in Wwise. (Photo Credit: Crystal Chan)

An integral part of our core gameplay involves having conversations with each of the different virtual characters. We set up switches in our Wwise project to handle all of the voicelines spoken by the characters. We have five switches: Biologist, Commander, Doctor, Engineer, and Maxwell. Even though the player can also speak to Houston, the ship AI, it is not

included in the switch set due to the different types of Intents for which that conversational interaction listens. Houston interactions deal mainly with voice commands; the player tells Houston to do something (turn on the lights or play audio logs). Meanwhile, with the five characters labeled in the switch set, the player can have longer, more complex conversations with them. More importantly, the player can ask each of them the same questions or say the same things. For example, if the player specifically is talking to Commander Liu Wei and asks "What is your name?," we want the game to play Commander Liu Wei's line "Wei. Commander Liu Wei." Likewise, if the player is conversing with the Engineer, he should respond with "I'm Stephen Hobb."

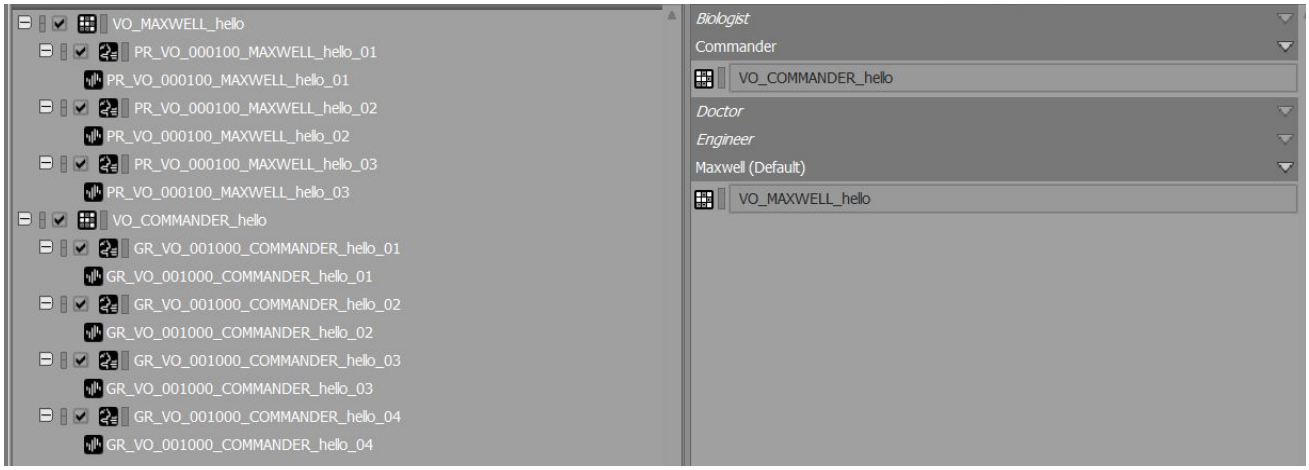


Fig. 8 - Screenshot of character-specific VO lines set up in Wwise. (Photo Credit: Crystal Chan)

While all of the conversation partner logic is setup in Wwise, it does not actually play the correct voicelines until we script it into Unity. Fig. 7 shows the code we are using to set the proper switch to play the respective voicelines. The example currently shows setting the different switches based on the player pressing the number keys on the keyboard, which is a temporary solution. We are working on implementing a system that uses the player's gaze to see which

character they are looking at along with some of the other design thresholds mentioned in the Designing for Speech Recognition section and setting it to the correct switch.

```
void Update () {

    if (Input.GetKeyDown(KeyCode.Alpha1))
    {
        conversationState = State.Maxwell;
        conversationPartner = Maxwell;
        AkSoundEngine.SetSwitch("VO_Conversation_Partner", "Maxwell", Maxwell);
    }
    else if (Input.GetKeyDown(KeyCode.Alpha2))
    {
        conversationState = State.Commander;
        conversationPartner = Commander;
        AkSoundEngine.SetSwitch("VO_Conversation_Partner", "Commander", Commander);
    }
    else if (Input.GetKeyDown(KeyCode.Alpha3))
    {
        conversationState = State.Engineer;
        conversationPartner = Engineer;
        AkSoundEngine.SetSwitch("VO_Conversation_Partner", "Engineer", Engineer);
    }
    else if (Input.GetKeyDown(KeyCode.Alpha4))
    {
        conversationState = State.Doctor;
        conversationPartner = Doctor;
        AkSoundEngine.SetSwitch("VO_Conversation_Partner", "Doctor", Doctor);
    }
    else if (Input.GetKeyDown(KeyCode.Alpha5))
    {
        conversationState = State.Biologist;
        conversationPartner = Biologist;
        AkSoundEngine.SetSwitch("VO_Conversation_Partner", "Biologist", Biologist);
    }
```

Fig. 9 - Screenshot of code setting Wwise switches in Unity. (Photo Credit: Crystal Chan)

Wwise also has occlusion and obstruction capabilities, if we so choose to include these features into the game in the future. Occluded sounds are indirect sounds that are heard from another room. Obstructed sounds are the reflections of the direct source because there is a wall or beam between the sound source and the player. However, because occlusion and obstruction can be quite expensive in computational runtime, we will not be incorporating these settings unless we find them to be crucial to gameplay, which is why, for the meantime, we have not applied these effects to our sounds.

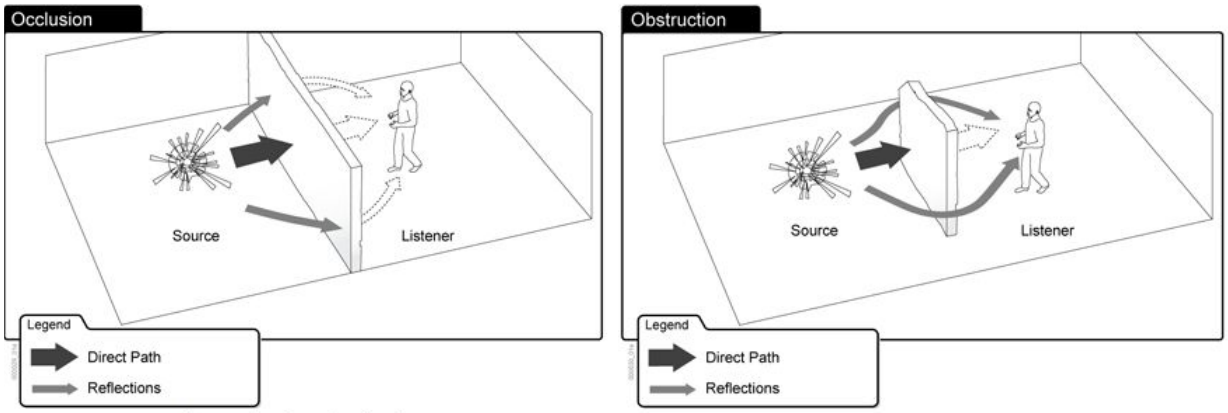


Fig. 10 - Examples of Occlusion (left) and Obstruction (right). (Photo Credit: Audiokinetic)

Oculus Audio Usage

In this section, we will see how Oculus Audio brought to life the sounds we play through Wwise, why it is important to have it integrated into our VR project, and the comparisons of it against other 3D audio plug-ins.

	Steam Audio	Oculus Audio	Google Resonance
Unity Support	x	x	x
Wwise Support	(Planned)	x	x
FMOD Support	x	x	x
Spatialization	x	x	x
Ambisonics	x	x	x
Occlusion	x		x
Environmental Modeling	x	x	x

Fig. 11 - Comparison of 3D audio plug-ins and their features. (Photo Credit: Crystal Chan)

3D Audio is especially crucial in this experience because it takes place in virtual reality. The primary use of a 3D Audio plug-in in this project is for Head-Related Transfer Function

(HRTF) spatialization. HRTF helps to simulate how people hear in reality in a VR environment utilizing complex algorithms that involve interaural level difference, initial time delay, direct/indirect ratio, high frequency dampening, etc. to calculate directional localization for sounds (Bosun).

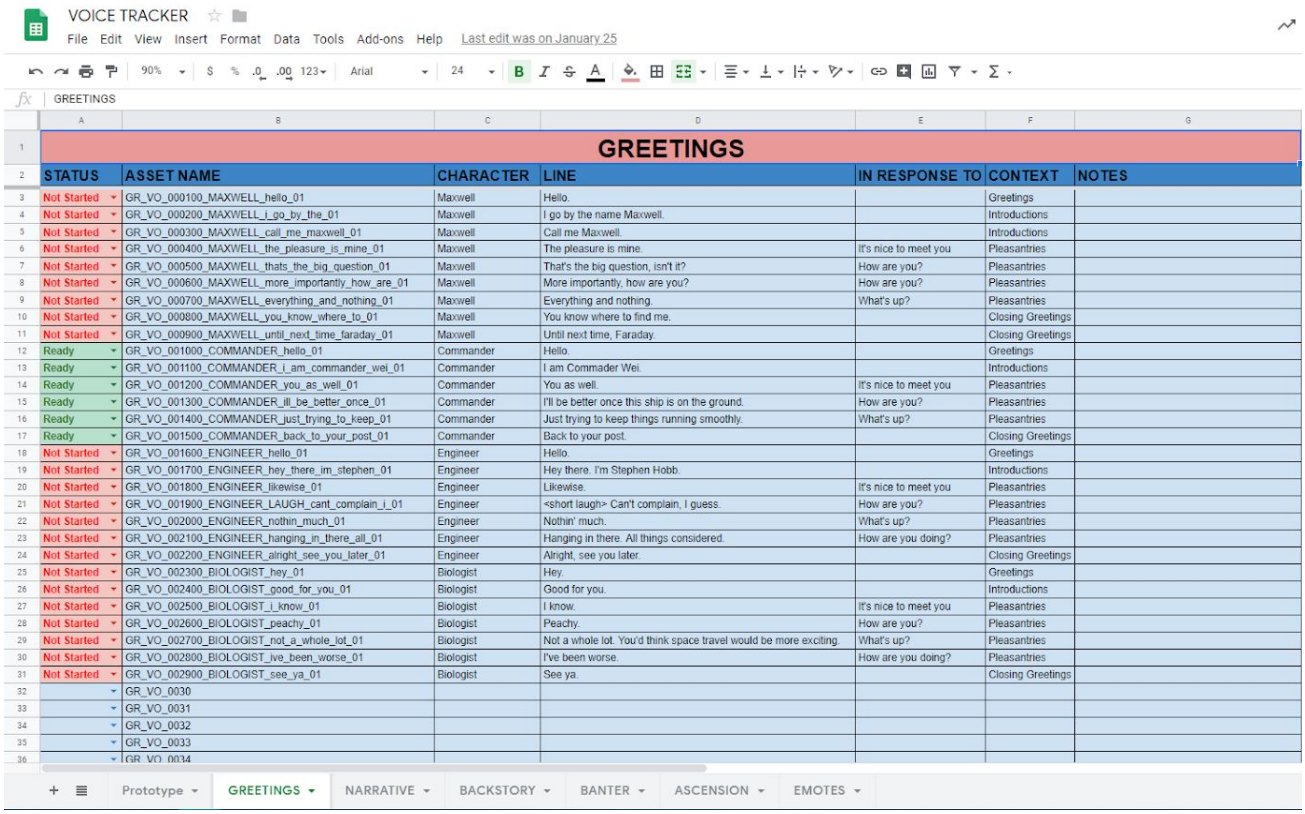
Considering that we will depend on Wwise to build our interactive music system, the best option is to go with Oculus Audio or Google Resonance, since Steam Audio currently does not have Wwise support (refer to Fig. 9 for comparisons of the three 3D Audio plug-ins). While Google Resonance has all the features that this project could possibly need, one of its biggest selling points is its optimization for mobile VR. Therefore, despite the lack of occlusion settings, Oculus Audio seems to be the best fit for this project, a VR game that utilizes a standalone head-mounted display. In addition, Wwise has an intuitive system for handling occlusion effects, solving any issues for if we need to include it in the game (refer to the previous section Wwise Application for more details).

Voiceline Organization and Pipeline

This section will cover the workflow starting from when the writers populate their spreadsheets with new voicelines and ending with those voicelines actually ending up in the game after recording, post-processing, and implementing.

Each writer is responsible for their own spreadsheet of voicelines for their respective characters for which they are writing and developing. Formatting is completely up to them and their preferences, with the exception of always notating newly written lines by filling those cells

with bright highlighter yellow to indicate that those lines need to be added to the Voice Tracker, our master spreadsheet of all the voicelines in the game.



VOICE TRACKER

File Edit View Insert Format Data Tools Add-ons Help Last edit was on January 25

GREETINGS

	A	B	C	D	E	F	G
1	GREETINGS						
2	STATUS	ASSET NAME	CHARACTER	LINE	IN RESPONSE TO	CONTEXT	NOTES
3	Not Started	GR_VO_000100_MAXWELL_hello_01	Maxwell	Hello.		Greetings	
4	Not Started	GR_VO_000200_MAXWELL_i_go_by_the_01	Maxwell	I go by the name Maxwell.		Introductions	
5	Not Started	GR_VO_000300_MAXWELL_call_me_maxwell_01	Maxwell	Call me Maxwell.		Introductions	
6	Not Started	GR_VO_000400_MAXWELL_the_pleasure_is_mine_01	Maxwell	The pleasure is mine.	It's nice to meet you	Pleasantries	
7	Not Started	GR_VO_000500_MAXWELL_thats_the_big_question_01	Maxwell	That's the big question, isn't it?	How are you?	Pleasantries	
8	Not Started	GR_VO_000600_MAXWELL_more_importantly_how_are_01	Maxwell	More importantly, how are you?	How are you?	Pleasantries	
9	Not Started	GR_VO_000700_MAXWELL_everything_and_nothing_01	Maxwell	Everything and nothing.	What's up?	Pleasantries	
10	Not Started	GR_VO_000800_MAXWELL_you_know_where_to_01	Maxwell	You know where to find me.		Closing Greetings	
11	Not Started	GR_VO_000900_MAXWELL_until_next_time_faraday_01	Maxwell	Until next time, Faraday.		Closing Greetings	
12	Ready	GR_VO_001000_COMMANDER_hello_01	Commander	Hello.		Greetings	
13	Ready	GR_VO_001100_COMMANDER_i_am_commander_wei_01	Commander	I am Commader Wei.		Introductions	
14	Ready	GR_VO_001200_COMMANDER_you_as_well_01	Commander	You as well.	It's nice to meet you	Pleasantries	
15	Ready	GR_VO_001300_COMMANDER_ill_be_better_once_01	Commander	I'll be better once this ship is on the ground.	How are you?	Pleasantries	
16	Ready	GR_VO_001400_COMMANDER_just_trying_to_keep_01	Commander	Just trying to keep things running smoothly.	What's up?	Pleasantries	
17	Ready	GR_VO_001500_COMMANDER_back_to_your_post_01	Commander	Back to your post.		Closing Greetings	
18	Not Started	GR_VO_001600_ENGINEER_hello_01	Engineer	Hello.		Greetings	
19	Not Started	GR_VO_001700_ENGINEER_hey_there_im_stephen_01	Engineer	Hey there. I'm Stephen Hobb.		Introductions	
20	Not Started	GR_VO_001800_ENGINEER_likewise_01	Engineer	Likewise.	It's nice to meet you	Pleasantries	
21	Not Started	GR_VO_001900_ENGINEER_LAUGH_cant_complain_i_01	Engineer	<short laugh> Can't complain, I guess.	How are you?	Pleasantries	
22	Not Started	GR_VO_002000_ENGINEER_nothin_much_01	Engineer	Nothin' much.	What's up?	Pleasantries	
23	Not Started	GR_VO_002100_ENGINEER_hanging_in_there_all_01	Engineer	Hanging in there. All things considered.	How are you doing?	Pleasantries	
24	Not Started	GR_VO_002200_ENGINEER_alright_see_you_later_01	Engineer	Alright, see you later.		Closing Greetings	
25	Not Started	GR_VO_002300_BIOLOGIST_hey_01	Biologist	Hey.		Greetings	
26	Not Started	GR_VO_002400_BIOLOGIST_good_for_you_01	Biologist	Good for you.		Introductions	
27	Not Started	GR_VO_002500_BIOLOGIST_i_know_01	Biologist	I know.	It's nice to meet you	Pleasantries	
28	Not Started	GR_VO_002600_BIOLOGIST_peachy_01	Biologist	Peachy.	How are you?	Pleasantries	
29	Not Started	GR_VO_002700_BIOLOGIST_not_a_whole_lot_01	Biologist	Not a whole lot. You'd think space travel would be more exciting.	What's up?	Pleasantries	
30	Not Started	GR_VO_002800_BIOLOGIST_ive_been_worse_01	Biologist	I've been worse.	How are you doing?	Pleasantries	
31	Not Started	GR_VO_002900_BIOLOGIST_see_ya_01	Biologist	See ya.		Closing Greetings	
32		GR_VO_0030					
33		GR_VO_0031					
34		GR_VO_0032					
35		GR_VO_0033					
36		GR_VO_0034					

Prototype | GREETINGS | NARRATIVE | BACKSTORY | BANTER | ASCENSION | EMOTES

Fig. 12 - Screenshot of Voice Tracker document. (Photo Credit: Crystal Chan)

The Voice Tracker is based off of the practices I learned while working as an Associate Sound Designer at Survios. The tabs on the bottom usually indicate different levels within the game, but its main purpose is to categorize the voicelines into intuitive groups, so while *Space Shuttle Ascension* does not have levels, we broke them up based on interaction type. Greetings include all the hi, bye, nice-to-meet-you, and how-are-you-doing type of interactions. Narrative lines provide narrative exposition and move the story forward. Backstory lines reveal more information about the respective characters. Banters are fun ways that the characters can show a bit more of their personality while answering the player's questions. The Ascension tab holds all

voicelines that will play no matter what the player says, such as the prompts in the opening sequence that tells the player to count down or say “liftoff.”

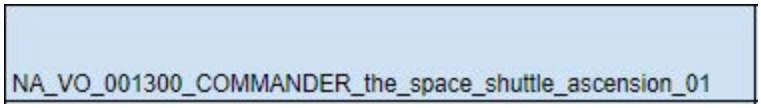


Fig. 13 - Example of the naming convention for VO lines. (Photo Credit: Crystal Chan)

Organizing voicelines is a crucial part of any game development process and becomes exponentially more important the greater number of them there are in a project. Asset names should be extremely readable and communicate vital bits of information, so that specific voicelines are easy to find and reference, even as we continue to add more. This naming convention contains (in order): an abbreviated level code, “VO” to indicate that it is a voiceline, line number, speaking character name, the first four words of the line, and the variation number.

I explained the purpose of levels in an earlier paragraph in this section. The abbreviated level code is an extension of that purpose. Line numbers are important in parts of the script where lines have to be in order. For example, in our opening sequence, the JPL Engineer prompts the player to count down and then Maxwell prompts them to say “Liftoff.” Those two lines have to be in that order, so the line number reflects where the lines happen in relation to everything in that level. Speaking character name allows us to filter for lines said by a particular character, which is particularly useful in *Space Shuttle Ascension* since all the characters can respond to the same player input and we sometimes need to isolate a single character’s responses. The first four words allow us to easily recognize the line being said simply by looking at the line name, without us having to play the file and listening to decipher what line #001300 might be. Lastly, the variation number is typically more relevant in terms of emotes/efforts,

where we need ten different attack grunts to randomize, allowing us to essentially keep the same asset name, but just increment the variation number to illustrate how many variations of that specific sound we have.

The Voice Tracker also includes a separate Character and Line column to help filter for different characters' lines and reference which line goes with which asset name. Context and Notes columns tend to be more flexible and vary depending on each project's needs. In our case, we use context to further define when the character would say a certain line. We will most likely be dedicating the Notes column for acting direction. For *Space Shuttle Ascension* in particular, we have also added an In Response To column in order to match our voicelines to predetermined player input, as well as a Status column to keep track of where we are in the process for each line (Not Started: new line; Recorded: recording done; Ready: processed and mastered, ready for implementation).

We are using SoundForge as a batch converter and mastering tool. This allows us to process a large amount of sounds that need the same post-processing done (like all of our voicelines) very quickly. We have a batch job, which is a saved setting with all of the processing preset, that is applied to all of the voicelines. First, we remove the sub-bass frequencies (about 100Hz and below) to take away the muddiness of the recordings and keep the low end of the frequency spectrum for sounds like bass rumbles and low drones to be more effective. Then, we use a noise gate removal to clean them up, removing any noise (unwanted sounds) from possible dirty signals. After that, SoundForge normalizes Root Mean Square (RMS) to -10dB. RMS can be thought of as the average sound level of a file and used to represent an estimation of loudness.

This process allows us to get the maximum volume while also matching volumes of all the sound files. The last step is auto-trim, where any silences at the beginning or end of the file are cut out.

Once the recorded files are properly named, processed, and mastered, they are imported into the Wwise project (please refer to the Wwise Application section for more information regarding how we set up Switches to correctly play the voicelines). After the corresponding lines of code are added into the Unity game engine, the voicelines will have been implemented into the game and are triggered when the respective player input has been detected.

Playtest

We have yet to playtest this to people who do not know that it is a game with a speech recognition system, but the feedback that we have received thus far has been positive and encourage us that we are heading down the right direction. Many people have used words like "powerful" and "satisfying" to describe the feelings they experienced when playing through the opening sequence. We want players to feel powerful because Faraday, the player character, has been put into a position of power, being chosen by Maxwell to travel through time to solve one of history's greatest mysteries and his greatest regret. Satisfaction, on the other hand, comes with the power that they feel that they have because this world responds readily to their command.

We based our subsequent design decisions on this feedback, building out more moments that will elicit the same feelings of satisfaction and power. After the player is transported to the cockpit of the shuttle after the opening sequence, they will experience the shuttle crash immediately. The shuttle is hurtling towards Mars' surface and the four crew members (Commander Liu Wei, Engineer Stephen Hobb, Doctor Thomas Sani, and Biologist Dani

Killian) are in a state of enormous panic, yelling and blaming each other. Then the crash happens as the scene cuts to black, with only the sounds of heavy impact to paint the tragedy. Once the catastrophe settles, Maxwell appears and prompts the player to say “Take me back” to go back in time. Time begins to reverse, taking the player back to many moments before the crash occurs, giving them the chance to unravel this mystery.

Conclusion

Throughout the development of *Space Shuttle Ascension*, the team and I kept the speech recognition system at the forefront. We knew that this was a unique feature of the game and the primary means for our players to receive the narrative. The mechanic is introduced almost immediately after the experience begins, but also woven into it in a way that feels authentic to the world. We also discovered that the non-verbal aspects of the game were just as important, making sure that the characters feel believable and alive, engaging and approachable. They make eye contact with the player, use body language to express emotions, and have voicelines specifically tailored to them to bring out their individual personalities.

Beyond the design process, spending time and focusing effort on production and the technical aspects help to bring these concepts to life. We made very intentional choices when deciding to use IBM Watson, Wwise, and Oculus Audio, doing research to make sure that each of these would better our project in a way that stayed true to our goals.

Next Steps

As production continues, I have come to realize the value of alternative inputs in digital experiences. Speech, along with all of the non-verbal cues that are associated with conversation, is the most direct way to communicate. After designing a game that revolves around the use of a speech recognition system, my future plans include breaking down the way we communicate even more. How do we communicate if we cannot speak? What constitutes speaking? What if we can mouth the words we want to say, but cannot produce the sounds for those words to be heard? Does that still count?

I hope to break into the field of Augmentative and Alternative Communication (AAC) to find these answers firsthand. AAC helps people with severe speech and language problems to communicate, which could include using eye tracking hardware, speech-generating devices, and, perhaps in the near future, an app that utilizes LipNet, Google's extremely accurate lip-reading AI. Limiting communication to only eye movement or the pressing of a single button can not only help us learn more about effective communication, but also contribute to a field that can provide immense help for people who can benefit from strides in this type of technology.

Bibliography

Alexanderson, Simon, et al. “Mimebot—Investigating the Expressibility of Non-Verbal Communication Across Agent Embodiments.” ACM Transactions on Applied Perception, vol. 14, no. 4, 2017, pp. 1–13., doi:10.1145/3127590.

Allison, B., Illah, N., and Y. R. Simmons. The role of expressiveness and attention in human-robot interaction. In *ICRA*, pages 4138–4142, 2002.

Xie, Bosun. *Head-Related Transfer Function and Virtual Auditory Display (2nd Edition)*. 2nd ed., J. Ross Publishing, 2013.

Call of Duty. Infinity Ward, 2003-2018.

Choi, M., Kornfield, R., Mutlu, B., and Takayama, L. (2017). Movement Matters: Effects of motion and mimicry on perception of similarity and closeness in robot-mediated communication. Proceedings of Human Factors in Computing Systems: CHI 2017, Denver, CO, 325-335.

Gonzalez, Dioselin. “Speech Recognition and VR – Unity Blog.” Unity Technologies Blog, 2 Aug. 2016. <blogs.unity3d.com/2016/08/02/speech-recognition-and-vr/>.

“Coda: An Alliance of Human and Machine.” Smart Machines, 2013, doi:10.7312/kell16856-008.

Holly, Russell. “Star Trek: Bridge Crew Review — This right here is why VR exists.” 29 May 2017. <https://www.vrheads.com/star-trek-bridge-crew-review-right-here-why-vr-exists>.

Job Simulator. Owlchemy Labs, 2016.

Ju, Wendy. “Adventures of an Adolescent Trash Barrel Robot.” *Vimeo*, research by Stephen

Yang, David Sirkin, and Brian Mok, 10 Dec 2014. <https://vimeo.com/114106601>.

Kiesler, S., Powers, A., Fussell, S.R., and C. Torrey. Anthropomorphic interactions with a robot and robot-like agent. *Social Cognition*, 2:169–181, 26.

Madden NFL. EA Tiburon, 1988-2018.

Meteer, M. "Voice Input for Collaborative Systems." 1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings, 1997, pp. 19–25., doi:10.1109/asru.1997.658971.

Rae, I., Takayama, L. & Mutlu, B. (2013). The influence of height in robot-mediated communication. Proceedings of Human-Robot Interaction: HRI 2013, Tokyo, Japan, 1-8.

The Sims. Maxis, 2000-2019.

Star Trek: Bridge Crew. Red Storm Entertainment, 2017.

Super Mario Bros. Nintendo Creative Department, 1985-2005.

Takayama, L., Dooley, D., & Ju, W. (2011). Expressing thought: Improving robot readability with animation principles. Proceedings of Human-Robot Interaction: HRI 2011, Lausanne, CH, 69-76.

Takayama, L., Groom, V., & Nass, C. (2009). I'm sorry, Dave: I'm afraid I won't do that: Social aspects of human-agent conflict. Proceedings of Human Factors in Computing Systems: CHI 2009, Boston, MA, USA, 2099-2108.

Uncharted. Naughty Dog, 2007-2017.

Wall-E Builders, The. "Caution: Rogue Robots!." 22 Jun 2008. Online image. Flickr. 17 Jan 2019. <https://www.flickr.com/photos/wall-ebuilders/2601662367>.

Wall-E. Directed by Andrew Stanton. Walt Disney Pictures and Pixar Animation Studios, 2008.

Wistort, Ryan. Only robots on the inside. *Interactions*, 17(2):72–74, 2010.

World of Warcraft. Blizzard Entertainment, 2004.