

Senior Project

Design and Development of Project VIVA: Virtual Interactive Voice Agent

Nadine Adnane



Submitted in Partial Fulfillment of the Requirements for the degree
of Bachelor of Arts in Computer Science.

Advisor: Jane Chandlee
Bryn Mawr College
Spring 2020

Abstract

The way we interact with technology is changing rapidly. Voice interaction systems such as Siri, Cortana, and Alexa have increased the demand for real-time, personalized, human-like interfaces capable of making the completion of everyday tasks more engaging and efficient. Combined with the rise of virtual reality applications, now more than ever users are expecting highly immersive, interactive experiences.

While vocal-only technologies are very useful, they lack the richness of experience and stimulation that can only be achieved by interfaces which incorporate visual feedback. Many companies are working to capitalize on this market. For example, Amazon has recently released a new tool, Amazon Sumerian, which allows developers to create immersive virtual or augmented reality web and mobile applications featuring virtual assistants. The applications of this technology are endless – training simulations, medical scenarios, travel & hospitality guides etc. Due to the rise of this technology, this project aims to explore the ease with which an individual can create highly interactive virtual agent applications using mostly free and open-source tools.

The first half of this thesis describes a method of creating a custom text-to-speech (TTS) voice using Microsoft Azure’s new Custom Voice cloud-based service. A major goal of this project is to customize as much of the interface as possible, so creating a personalized TTS voice is essential. The second half of this thesis focuses on the implementation details of the final product – a virtual reality application developed with Unity Game Engine. Using Microsoft’s speech recognition and synthesis services, the app will feature a virtual humanoid agent, “Viva”, which is capable of answering questions about Bryn Mawr College rendered in my own custom TTS voice. My results indicate that the agent is able to respond to in-scope questions with an accuracy rate of 70% and an average of 85% confidence in intent matching.

Acknowledgements

To my endlessly supportive and loving Mama and Papa – there are no words that could fully express my gratitude for you and everything that you do. Thank you for all of the sacrifices you have made and hardships you have endured to provide us with a better life and allow me to be who I am today. I could not have made it through all of the sleepless nights, unexpected obstacles, and moments of doubt not only during the development of this project but all throughout my time at Bryn Mawr were it not for your encouragement, advice, and affection.

A special thanks to my older sister Lydia, whose compassion, positivity, wisdom, and selflessness knows no bounds, and my sweet and whimsical younger brother Zachary, who has never failed to put a smile on my face and joy in my heart.

I would also like to thank from the bottom of my heart my project adviser Professor Jane Chandlee, and Senior Conference adviser Professor Deepak Kumar. Thank you for believing in me and my ideas, providing insightful feedback on my many drafts, and supporting and guiding me through the many challenges of this project (and year!). I would also like to thank Professor Richard Eisenberg for constantly challenging and inspiring me throughout the course of my Computer Science education at Bryn Mawr.

To my dearest friends, Anna Baroud, Tanjuma Umme Haque, Viktoria Braun, and Nisha Choudhary – it is no secret that you all mean the world to me. Thank you for listening to my problems sincerely, cheering me on at all hours of the night, and reminding me to not be so hard on myself, take breaks, and be proud of all I have achieved. I am so inexpressibly grateful that Bryn Mawr has brought us together.

A special thanks goes to ViAnna Bernard, my personal Hall Adviser and beloved friend. In addition to all of the warm hugs, constant silliness, and invaluable advice, thank you for always listening so intently to my ideas and for getting so excited about my projects! Your constant enthusiasm makes me feel like I can do anything.

I would like to dedicate this project to my dear grandparents, the roots of my existence. We may be separated by thousands of miles, but you will be with me forever and always in my heart. I hope I have made you proud.

Last but not least, I would like to thank Bryn Mawr College, my home away from home for four years and many more to come.

My gratitude is immeasurable, and my heart is entirely full. Anassa Kata!

Contents

1	Introduction	1
2	Motivation	3
3	Background	4
	3.1. Statistical Parametric Model for Speech Synthesis	4
	3.2. Virtual Reality	5
4	Related Work	7
	4.1. MMDAgent	7
	4.2. Impact of Inclusion of Human-like Interfaces	10
	4.3. The SEMAINE Project	11
5	Methodology	12
	5.1. Custom Text-to-Speech Voice Creation	13
	5.2. Speech Synthesis in Unity	16
	5.3. Speech Recognition in Unity	17
	5.4. Dialogflow AI Agent Creation	19
	5.5. Unity Project	25
	5.6. Character & UI Design	25
	5.7. Additional API Integrations	33
6	Discussion	35
	6.1. Results & Analysis	35
	6.2. Challenges	39
7	Future Work	41
	7.1. Speech Synthesis & Recognition	41
	7.2. Dialogflow Agent	42
	7.3. Character & Interface	43
	7.4. Application Capabilities	44
8	Summary	47

References

1 Introduction

The goal of this project was to create a virtual reality campus guide application featuring a 3D assistant capable of answering questions about Bryn Mawr College. In this context, virtual reality involves a computer-generated 3D environment that can be interacted with in a seemingly real or physical way, without the need for specialized equipment such as sensors or headsets. The input to the application is a question or command from a user provided as audio from a microphone. The output of the program is an animated 3D avatar which responds to the user with synthetic speech generated by a custom text-to-speech voice built from my own voice recordings.

My virtual interactive voice agent, acronymically named “Viva”, can respond to a variety of questions about student life with answers derived from my own experience and from the Bryn Mawr College website. For example, Viva can answer questions about each of the dorms, residential life, clubs, traditions, the health center, dining services, and dorm leadership team positions. Viva can also retrieve the time and date, describe the current weather in Bryn Mawr town, and retrieve a user’s daily horoscope given their zodiac sign. Additionally, the agent can tell jokes, respond to greetings and questions about itself, and remember the current user’s name. Thanks to the use of artificial intelligence and machine learning, Viva is able to recognize and provide a randomized response to a range of phrasings of the same question.

The user may also interact with Viva through a few specialized commands instead of questions. For example, the user can tell Viva to change her outfit, change or reset the background, and show or hide a map of the Bryn Mawr College campus. Last but not least, given a valid US phone number Viva can send the user a text message each day, and if the user provides a valid email address, Viva will send them an email.

Several example interactions in this context are provided on the following page.

User : “Hey Viva, what is Lantern Night?”

Viva : *“In the fall semester, the first-year students gather in the cloisters where the sophomores will present them with their lanterns in their class color or the McBride color of purple. The sentiment behind Lantern Night is that the light of knowledge is being passed from one class to the other.”*

****User :** “How old are you?”

Viva : *“I am immortal, so it doesn't matter”*

****User :** “How old are you?”

Viva : *“That's a secret!”*

User : “Tell me about Merion.”

Viva : *“Merion is known for its warm, cozy, and quiet atmosphere. To many students including my creator, Merion feels like home. Many of its rooms still contain the fireplaces and window seats used by the College's first students, while its stairwells are decorated with the original stained-glass windows. It is also home to Lillian, one of the college's beloved ghosts!”*

User : “Will you marry me?”

Viva : *“Aww sorry, I can't! I'm happily Bryn-Married to my best friend.”*

User : “Is there a freshman dorm?”

Viva : *“All dorms on campus contain a mixture of classes. There is no dorm on campus that only holds first years, sophomores, juniors, or seniors.”*

User : “You're annoying.”

Viva : *“That's not very nice.”*

User : “Can you tell me a joke?”

Viva : *“My first job was working in an orange juice factory, but I got canned because I couldn't concentrate!”*

User : “Hey Viva, change into your HA shirt.”

Viva : [*different outfit appears on Viva 3D character model*]

****** - Demonstrates agent's ability to respond differently to the same question when appropriate.

[] - Indicates that an action was performed in place of an audio response.

Roadmap

The design and development of Project Viva is explained in sections as follows. Section 2 explains the motivation behind the creation of a custom text to speech voice and development of a virtual agent app. Section 3 provides background information on two main areas of my thesis: the statistical parametric model for speech synthesis and virtual reality. Section 4 explores previous works involving virtual agents and chatbots. Section 5 details the methodology of the project including the voice creation and app development stages. Section 6 provides a discussion and analysis of the results and challenges of the project. Section 7 proposes future work and improvements that could be made to the project down the road. Section 8 summarizes the goals and overall results of the project.

2 Motivation

My thesis project was primarily inspired by “MMDAgent”, a software developed by researchers at the Nagoya Institute of Technology in Japan. MMDAgent – less widely known as “MikuMikuDance Agent” – is an open-source toolkit which incorporates recent speech recognition and synthesis technologies with a 3D CG rendering module that can manipulate expressive embodied agent characters (Lee et al., 2013). The software allows users to build voice interaction systems and was released in hopes of contributing to the popularization of speech technology (Lee et al., 2013).

In many ways, this project was an exploration of the tools and services available to the general public (i.e. students, hobbyists, etc.) for the creation of such highly interactive virtual experiences. Although my love for software development, AI, and speech synthesis was enough to motivate my pursuit of this project, I also wanted to see for myself if it would be possible to create a convincing virtual agent using mostly free or easily accessible tools and my computer science education. Finally, I wanted to create an experience tailored to the context of Bryn Mawr College, as it is my senior year and I believe that this virtual tool could be very beneficial to the admissions process and prospective Mawrtys to come.

3 Background

In this section, I will provide general information on the two underlying foundations of my thesis project: the statistical parametric model for speech synthesis which was used to create a custom text-to-speech voice, and virtual reality.

3.1 Statistical Parametric Model

The statistical parametric model is similar yet distinct from the HMM or “Hidden Markov Model” approach to speech synthesis (King, 2010). In fact, statistical parametric approaches are often referred to as HMM synthesis because they normally use HSMMs, or “Hidden Semi-Markov Models” (King, 2010). The model can largely be understood by breaking down its name. A speech synthesis model is “parametric” if it breaks speech down into parameters, instead of stored “exemplars” (King, 2010). To use King’s example, an exemplar-based speech synthesis model works similarly to the index in a book. The model stores parts or the entirety of the speech corpus, making sure to grab an instance of each type of speech sound (King, 2010). Then, as King describes the model “looks up” all of the necessary parts to form a specific sentence or “linguistic specification”. Systems that use this approach are commonly known as “unit-selection” systems. Unlike unit-selection systems, systems made with statistical parametric models do not store parts of the corpus. The second part of the name, “statistical”, comes from the fact that the model describes the parameters using statistical properties, such as the “means and variances of probability density functions” (King, 2010). These statistics provide a good sense of the distribution of parameter values found in the training data (King 2010). One important benefit of using a statistical parametric model for speech synthesis which is relevant to this project is that “the quality of the resulting voice scales up as the amount of data is increased” (King, 2010). As King explains, this is not only because a larger dataset contains more speech, but also because it will contain a greater variety of contexts, allowing the model to be more sensitive and produce better speech (King, 2010). To my surprise, King also mentions that the statistical parametric model generally can produce better results for low quality speech data than can state-of-the-art concatenative systems.

Learning about statistical parametric and related models was beneficial for my understanding of the training of my custom text-to-speech voice. At the time that this project was started, Microsoft had not specified the training method behind their Custom Voice service, but it was recently revealed that there are three methods available which are mentioned in Section 7 Future Work. Although Microsoft’s service does not require any prior knowledge of speech synthesis or linguistics, King’s paper helped me to

understand how Microsoft is able to turn a collection of audio files and text transcript into a virtual voice! This also helped me to better understand the MMDAgent project – my primary inspiration - which is described in detail in Section 4.1, and the options available in the MaryTTS software which is mentioned in Section 6.2 Challenges.

3.2 Virtual Reality

Virtual reality (VR) describes the way in which computers can create experiences for users, either to simulate things that can already be experienced in the outside world, or to envision experiences that can only be imagined. Because the “virtual world” is often used to emulate experiences similar to those that already exist, it is necessary for the virtual world created to be believable enough that users really feel that they are experiencing reality, preventing the descent into what is known as the “uncanny valley”. The term “uncanny valley” was first used by Professor Masahiro Mori of the Tokyo Institute of Technology in an essay about people’s reactions to robots that look and act almost human (Mori, 1970). In particular, Mori hypothesized that “a person’s response to a humanlike robot would abruptly shift from empathy to revulsion as it approached, but failed to attain, a lifelike appearance” (Mori, 1970). This sharp “descent into eeriness” is known as the uncanny valley (Mori, 1970). Figure 1 below shows a graphical representation of this phenomenon.

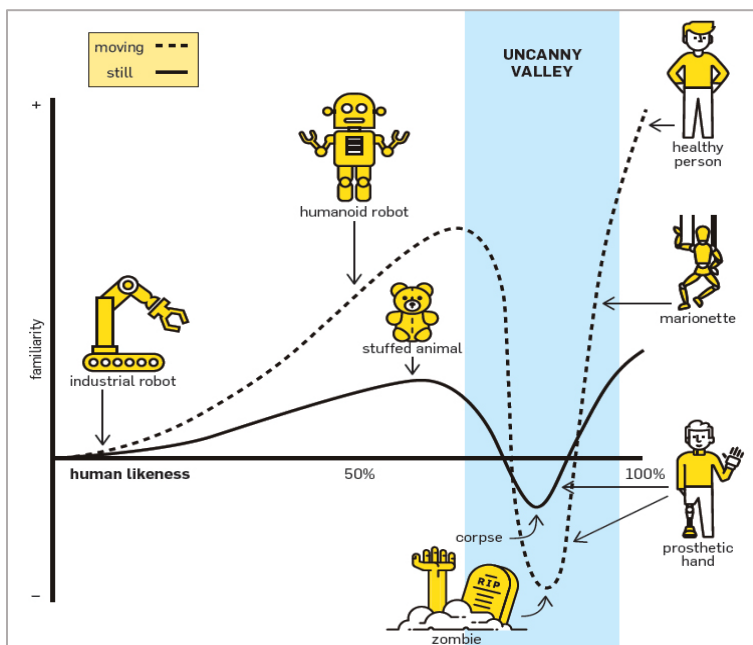


Figure 1: A graph depicting the “uncanny valley” and proposed relationship between the human likeness of a robot (x-axis) and the viewer’s reaction to the robot or “familiarity” (y-axis). Essentially, humanoid robots or characters which bear an “uncanny” or just slightly imperfect resemblance to real humans can evoke a repulsed reaction from the viewer. This figure was reproduced from Gregory Mone’s ACM journal article “The Edge of the Uncanny”, 2016.

Virtual reality includes many options for how immersive of an experience is desired. Additions that create a fully immersive experience can include the use of a headset or head-mounted display. More immersive VR generally includes the use of more advanced technologies, including binaural sound, haptic feedback, and voice input and output. Additionally, user head and hand tracking can be used to create a more “real” experience. Non-immersive virtual reality typically includes highly realistic renditions of computer-generated worlds using audio-visual media that still conveys the intended experience for the user. Generating a “rich environment for potential users” that creates the “illusion of participation in a synthetic environment” (Earnshaw, 2010) is one of the most important considerations in the creation of a virtual reality world. The use of modelling systems to generate realistic 3D objects combined with the application of effective audio are key parts of the development of a VR experience.

This technology is unique in that the user is no longer just an observer of the virtual world but is rather intended to interact with it in a meaningful way. The environment is meant to change and adapt to the input from the user, and the user is able to engage in a multi-sensory experience. VR is a concept that has been introduced to the masses relatively recently, and there is an abundance of false information regarding the current capabilities of this technology. Despite the relative novelty of virtual reality technology, the history of VR development dates back to the 1960’s. Early iterations could never have predicted the advancements made in the field of virtual reality. In recent years, VR has become a rapidly developing technology used for everything from games to medical treatments and which will continue to develop in the 20th century. While Project Viva will not require any special equipment other than computer, internet connection, and a microphone to be used, the project shares the goals of high interactivity and immersion of virtual reality described above.

4 Related Work

This section explores related projects and analyses which served as useful references during the development of my project.

4.1 MMDAgent

In this section, I will provide further details about the MMDAgent interface and software capabilities to establish a clearer picture of my goals for this project.



Figure 2: The MMDAgent interface developed by the MMDAgent Project Team in the Department of Computer Science at the Nagoya Institute of Technology (Lee et al., 2013). The default agent “Mei” is shown standing politely awaiting a recognizable question or phrase while a menu displaying possible conversation topics revolves continuously. A black shaded bar visualizes the audio waveforms.

4.1.1 Humanoid 3D Character

At startup, the app presents a 3D female humanoid virtual agent named “Mei”, who stands in the center of a simple 3D environment. Mei is dressed similarly to a flight attendant and is animated to stand politely when idle.

4.1.2 Animations

The sample comes with several animations including gestures to express a variety of emotions. Each animation file must be in the .VMD format, otherwise known as “Vocaloid Motion Data” (Higuchi, 2008). Animations in this format originate from a Japanese choreography and 3D animation software, MMD or “MikuMikuDance”, which is very closely involved in MMDAgent. MMD can be used to create choreography animations, short films, and other animated media (Higuchi, 2008). VMD files can contain animation frames for character models, accessories (ex. microphones or objects without humanoid structures), camera movements, and stage positions. Before creating a VMD animation in MMD, users must first either download an open-source 3D model from the internet or create their own. Either way, the models must be in one of two specific file formats - .PMD (Polygon Model Data) or .PMX (Polygon Model Extended). These models can be created using another software which goes hand-in-hand with both MMD and MMDAgent entitled PMXEditor (Higuchi, 2008).

4.1.3 Model-Rigging

Similar to most humanoid 3D model formats, .PMX and .PMD files are composed of a 3D mesh with textures and materials rigged to a weighted bone structure or “armature”. Models may also have facial expression sliders (“morphs”), joints, and rigid-body colliders for physics animations. Unsurprisingly given the name, MMDAgent also requires one of these 3D model file formats for each virtual agent. Mei’s character model is in the .PMD format.

4.1.4 Facial Expressions & Lip-syncing

PMXEditor allows users to manipulate 3D meshes to create facial expressions or other “morphs” which can then be controlled as sliders in VMD animations. Mei’s character model features morphs which allow her to express happiness, sadness, anger, and surprise. The Mei model also has mouth sliders for each vowel and basic sound used when speaking Japanese. These sliders are used to lip-sync Mei’s character model with her audio responses to user questions for a more natural, rich experience.

4.1.5 Speech Recognition & Synthesis

Users can speak into a microphone to ask Mei various questions about the Nagoya Institute of Technology's campus, including questions about the current time, weather, and directions to buildings on campus. Mei also responds to various greetings and questions about herself, such as inquiries about her name and age. Currently, the MMDAgent software only uses Japanese for both input and output due to the use of Julius (speech recognition system) and Open JTalk (TTS system).

The virtual agent responds to the input questions in Japanese using a speech synthesis system called HTS. The artificial voice is a Hidden Markov Model (HMM) based voice.

4.1.6 Features & Examples

The MMDAgent sample assistant "Mei" can do all of the following:

- Provide campus guidance
- Introduce academic staff
- Introduce the university President
- Provide event information
- Display various event posters
- Provide weather forecasts
- Provide horoscopes

The following examples reproduced from the MMDAgent website illustrate some of the capabilities which I intend to emulate (Lee et al., 2013).

Visitor: Toshokan wa dokodesuka? (Where is the library?)

*Mei-chan: Toshokan wa seimon kara miruto migimaeno hoko ni ari-masu.
(From the front gate, the library is on the right, at the front.)*

Visitor: Shumi wa nandesuka? (What is your hobby?)

Mei-chan: Yakyuu kansen desu. (I like watching baseball.)

Visitor: Gourmet joho wo oshiete. (Please give me some information on dining possibilities.)

Mei-chan: Meikoudai campus naino lunch map desu. (Here is an on-campus lunch map.)

Visitor: Nansai desuka? (How old are you?)

Mei-chan: Himitsu desu! (It is a secret!)

4.2 Impact of Inclusion of Human-like Interfaces

From educational software to video games to advertisements, the use of visual representations of human-like agents is becoming more and more prevalent (Yee et al. 2007). But why is this so? After all, standalone voice interaction systems are more than sufficient for a wide variety of tasks. What impact does the inclusion of visual humanoid representations - as opposed to simply providing auditory feedback - have on the effectiveness of an application? Yee et al. (2007) conducted a meta-analysis to answer this question. The study focused on a collection of empirical data from previous studies which compare interfaces with “visually embodied agents” to interfaces without agents for the meta-analysis. In this context, a visually embodied agent is a visual, human-like representation accompanying a computer interface (Yee et al. 2007). The results of the meta-analysis revealed that adding an embodied agent to an interface has a larger effect than just animating an agent to behave more realistically (Yee et al. 2007). The study also demonstrated a distinction between the analysis of subjective responses and behavioral responses. Subjective responses in this context include questionnaire ratings and interviews, while behavioral responses include task performance and memory (Yee et al. 2007). The results showed that the presence of an embodied agent had a larger effect on the results of subjective responses compared to behavioral responses (Yee et al. 2007). The study took into account three major properties that could affect a user’s experience when interacting with an interface including the presence or lack of an embodied agent, how realistic the virtual agent appears to be (through animations and behaviors), and the type of response being measured (subjective or performance-based) (Yee et al. 2007). The researchers primarily wanted to see if people react differently to interfaces with:

- No visual representation at all
- A human-like representation with low realism (ex. a cartoon character)
- A human-like representation with high realism (e.g., 3D model with animated gestures)

The results showed that the presence of a visual representation produced more positive social interactions than those lacking any visual representation (Yee et al. 2007). It was also concluded that in the context of subjective applications, human-like representations with higher realism resulted in more positive social interactions than representations with lower realism (Yee et al. 2007). These findings help explain the motivation for companies and application designers in general to move towards more immersive and interactive virtual reality experiences, especially those intended to mimic natural human interactions.

4.3 The SEMAINE Project

The SEMAINE Project created Sensitive Artificial Listeners (SAL) in order to have emotionally-colored, fluent conversations with users (McKeown et al., 2011). McKeown et al. started by creating a large audio-visual database that allowed for 4 different SALs to respond to users. The four agents, Poppy, Spike, Prudence, and Obadiah are shown in Figure 3 below. Each SAL is able to respond to the user’s comments (but not questions) in a unique way based on pre-determined personalities. Poppy is an outgoing and optimistic agent, while Spike is a rude SAL which responds in a way that is meant to offend and insult the user. Prudence is a pragmatic and practical agent, while Obadiah is gloomy and depressed (McKeown et al., 2011). In order to determine whether or not the SALs were successful in responding to user input, interactions between the SALs and the users were recorded and assessed by individual judges based on the flow of the conversations and several other factors.



Figure 3: A screenshot displaying the four SAL agents followed by a screenshot of a user interacting with an agent. In the top screenshot the agents from left to right are Poppy, Spike, Prudence, and Obadiah. The bottom screenshot depicts a user interacting with Spike, the rude SAL.

The SEMAINE project served as a very useful reference while developing Project Viva. Interacting with and learning more about this project has helped me to identify which characteristics make an agent convincing, friendly, and natural, as well as which behaviors and components detract from these qualities. For example, I found it very unnatural when the agent interrupted or spoke at the same time as me. On the other hand, the fact that the agents blink, make eye contact, possess facial expressions, and speak with natural-sounding voices made the experience more believable and pleasant. The SEMAINE project

also demonstrates the importance of non-generic responses when interacting with an AI agent. Although the SALs cannot actually answer questions, the repetitive responses (ex. “Tell me about that”, “Why?”, “Is that so?”) to user input detract from the experience. It is essential that my agent be able to respond to user input in an appropriate and interesting way so that she can fulfill her purpose as a convincing Bryn Mawr campus guide. With this project in mind, Viva was programmed to have an optimistic and friendly personality and is able to respond in a more convincingly “human” way to many statements and questions thanks to the use of Dialogflow, which is discussed Section 5.4 Dialogflow AI Agent Creation.

5 Methodology

The development of my virtual interactive voice agent involves four main stages. First, I recorded hundreds of voice clips to build my own text-to-speech voice. Next, I configured my Unity application to connect to my Microsoft Azure Cognitive Services account to allow Viva to provide answers to user input queries rendered in my own voice. Then, I gathered and used data from various pages on the Bryn Mawr website and synthesized some responses of my own to create and train a Dialogflow AI chatbot. The chatbot allows Viva to detect the intent behind a question rather than listen for specific, hard-coded phrases and then respond appropriately. I then designed and implemented the UI, designed a 3D humanoid model using a character builder, and configured several animations to enhance the user’s experience and provide effective visual feedback. An overview of my methodology is provided in Figure 4 and the subsections below provide a more detailed explanation of each step.

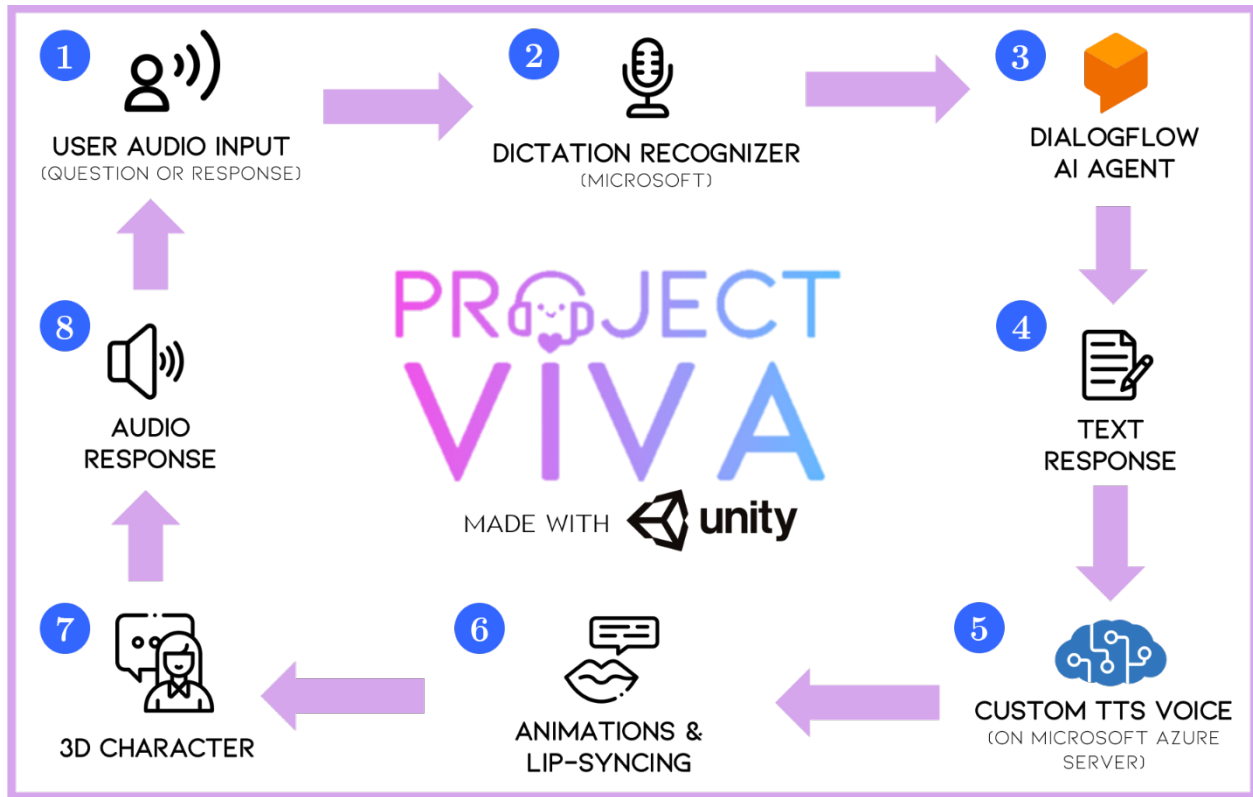


Figure 4: An overview of Project Viva. Unity, Dialogflow, and Azure icons reproduced from their respective company websites. Other icons made by Freepik from www.flaticon.com

5.1 Custom Text-to-Speech Voice Creation

Motivation

While using a pre-made TTS voice would likely provide higher quality results, the purpose of this app is to provide a highly interactive and customized experience which allows users to ask questions about Bryn Mawr College. It seemed fitting that they should receive answers voiced by an actual Bryn Mawr student! Thus, I decided to create my own text-to-speech voice as the first part of this project.

Research

The first stage of my project involved researching methods of custom text to speech voice creation. While there are several companies which offer services to accomplish this task, unfortunately there are very few which are financially accessible to students and individuals with non-enterprise level projects. Additionally, many of these services do not yet offer the option to export the customized voice model, but instead only offer the

synthesis and download of audio clips. It was my intention from the start of this project to make the experience as interactive and unrestricted as possible, so I decided to avoid the pre-generation of audio clips and instead limited my search to real-time synthesis services only.

In the following subsections, I detail my final approach to creating a custom text to speech voice. To learn more about my initial approach, see Section 6.2 Challenges.

5.1.1 Microsoft Azure Custom Voice

After the initial custom voice creation attempt (MaryTTS, described in Section 6.2 Challenges) failed, it was time to start from scratch. Upon further research into methods of custom TTS voice creation, I decided to use Microsoft Azure’s Cognitive Services to build a “Custom Voice Font”. The Custom Voice service is advertised as a way for both individuals and businesses to build unique voices for text to speech applications. Unlike MaryTTS, Azure Custom Voice does not currently allow users to export and download their custom voices, as it is a cloud-based, pay-as-you-go service. Thankfully, Microsoft offers a student subscription which includes up to \$100 of credits to use the Cognitive Speech Services amongst others.

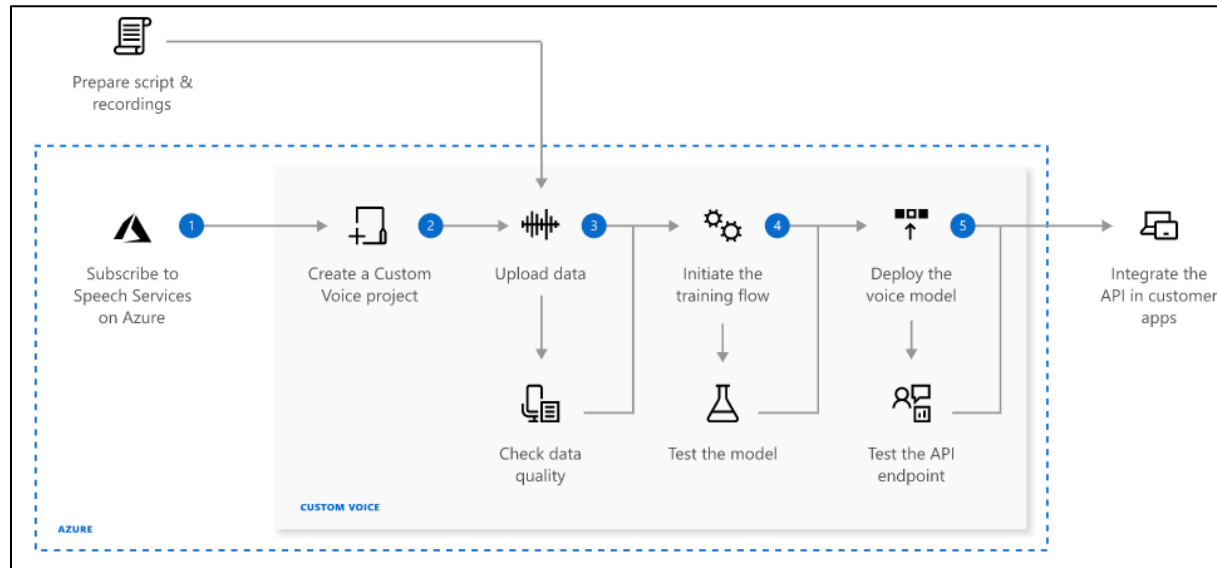


Figure 5: An overview of how the Microsoft Azure Custom Voice service operates reproduced from the Microsoft Azure Custom Voice webpage. The process is very user-friendly and only requires voice recordings and a transcript as input. The output is an “endpoint”, which is essentially a link to the custom voice which can be incorporated into projects on a wide range of platforms.

The custom voice service only requires two inputs from the user – an archive of waveform audio files (.wav), and a text transcript containing the sentences spoken in each audio file. The following subsections detail the process used to prepare the audio recordings and transcript for this project.

5.1.2 Transcript

The first step in this process was to find or create a transcript to read. Although it would have been interesting to create a transcript from scratch, I decided to search for an open-source corpus. This decision was made in the interest of time and in consideration of the expertise required to construct a substantial, well-balanced transcript file which provides a good representation of sounds in the English language. Thus, for my project I decided to use Carnegie Mellon University’s (CMU) Arctic transcript file, which is included with their set of open-source TTS voices. The transcript file includes 1132 sentences carefully selected from Project Gutenberg, an online library which collects and publishes out-of-copyright literature (Kominek et al., 2003). The sentences in the transcript were selected by Kominek et al. using a seven-stage process to ensure that the data was ideal for speech synthesis and voice creation.

Next, the transcript was formatted using regular expressions to meet Microsoft Azure’s Custom Voice data requirements. Luckily the structure of the Arctic transcript only needed minor edits involving padding filenames with zeros, removing quotations and parentheses, and adding tabs between each filename and corresponding sentence.

The table below provides a few example sentences from the formatted transcript file.

466	Captain West may be a Samurai, but he is also human.
467	And so early in the voyage, too.
468	In the matter of curry she is a sheer genius.
469	The eastern heavens were equally spectacular.
470	He spat it out like so much venom.

Figure 6: A table containing sentences from the CMU Arctic transcript file. Note that the sentences are kept short to make reading easier for the vocal talent.

5.1.3 Recording Conditions

The next step in this process was to acquire a high-quality microphone for recording the audio files. The Arctic CMU voices were recorded in a sound-proof booth using a Sennheiser MD431 near field condenser microphone (Kominek et al., 2003). I had initially hoped to replicate the recording conditions used to create the CMU Arctic databases; however, the equipment used was too expensive and inaccessible.

Instead, the recordings for this project were made with a Blue Yeti Blackout USB microphone. The microphone was used on the Cardioid pattern setting, which allows sounds made directly in front of the microphone to be picked up while sounds made towards the sides and back are ignored due to its shape. The gain dial on the microphone was set to zero to minimize noise interference, and a piece of painter's tape was placed one foot from the microphone to ensure that recordings were made from a consistent distance. Initially, I had planned to record in the same location throughout the whole process. However, this was not possible as I was unable to find a consistently quiet recording space on campus and later at home. The recording process was spread out over the duration of the semester to avoid vocal fatigue and to allow for progress in the app development portion of the project.

All recordings were created in Audacity, a free open-source, cross-platform audio software. In accordance with the Azure guidelines, each audio clip was recorded with a sampling rate of 48000 Hz, given a numeric filename, saved in the PCM 16-bit .wav format, and was no longer than 15 seconds.

With the transcript and audio files completed, the next step was to upload the data to Azure and select a training method. Due to the limited number of recordings, my project was only eligible for the basic Statistical Parametric Synthesis training method. The automated model training process took roughly a total of three hours. Finally, the custom voice model was deployed, resulting in a URL endpoint.

In the next section, I explain the setup of my Unity project and how the custom voice endpoint was integrated.

5.2 Speech Synthesis in Unity

After creating my custom text to speech voice, the next step was to make sure that my app would be able to make requests to the Microsoft servers for speech synthesis. I had initially intended to use a Unity plugin from the asset store for this purpose, but later

discovered that the plugin only supports pre-made, non-custom Azure voices and ended up writing a TTS Client script myself.

Due to my inexperience with API calls and web requests, this took a significant amount of time to debug; however, the implementation was successful, and the knowledge gained made later API integration much easier.

In the main scene of Project Viva, there is a Voice Manager GameObject which runs my TTS client script. The TTS script was written based on examples in the Microsoft Azure Custom Voice endpoint documentation. The main method requires a Microsoft Azure Cognitive Services API key, a JSON access token generated specifically for my account by the service online, the URL endpoint of my custom voice, and the speech service region for my voice which is EastUS. The script also requires an Audio Source GameObject to play the resulting clips from.

The following section describes how speech recognition was integrated into Project Viva.

5.3 Speech Recognition in Unity

After speech synthesis was configured and tested successfully, I began to research methods of speech recognition in Unity. During the brainstorming phase of this project, I had initially planned to preprogram a set of possible questions for users to ask Viva. For example, the phrase “Tell me about Merion Hall” was programmed into the recognizer but the user instead says “I’m curious about Merion”, the phrase would not be recognized despite having the same intent. This restrictive initial approach was easy to implement using Microsoft’s KeywordRecognizer class and is detailed below.

5.3.1 Initial Approach – KeywordRecognizer

The KeywordRecognizer class is part of Microsoft’s Mixed Reality Toolkit and is primarily used for HoloLens development. The KeywordRecognizer takes as input an array of string commands to listen for, and recognizes phrases using the Microsoft speech SDK.

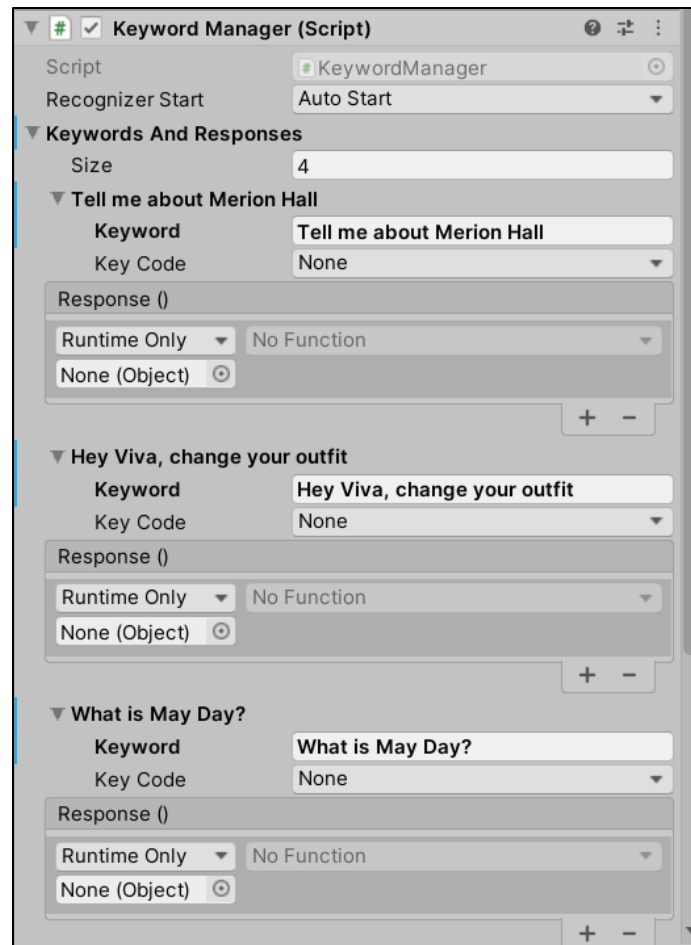


Figure 7: A screenshot of the Keyword Manager script which uses the KeywordRecognizer class. The programmer can specify the number of recognizable keywords and their responses. In this example, there are four possible keywords (though only three are shown). Note that although entries are called “keywords”, each input can actually be an entire phrase.

It is clear from the figure above that programming every possible wording of every possible question within the project scope would be impossible, or at the very least a significant task. Of course, the app could provide users with a menu of possible phrases to ask, but this approach was too limited for my goals.

5.3.2 Secondary Approach – Streaming

Upon further research, I realized that the speech recognition could be done by the Google Cloud API by streaming audio data to Dialogflow. The microphone would continuously listen to the user, and the streamingDetectIntent function would send the stream of audio data to the cloud. Both speech recognition and intent detection - which is discussed in detail in Section 5.4 – would be performed online and a text response returned to my app. Unfortunately, I was unable to successfully implement the function Dialogflow offers for

this purpose due to time constraints and the limited number of examples in the documentation and on Stack Overflow.

However, I was able to send individual audio queries to Dialogflow. This method involved a recording stop and start button. Although this suited my needs, it somewhat defeated the purpose of using speech recognition in the first place. To create a completely hands-free experience, I continued to research and upon revisiting the Microsoft documentation eventually came across the DictationRecognizer, a related class to the KeywordRecognizer which perfectly suited my needs with slight modification.

5.3.3 Final Approach – DictationRecognizer

The DictationRecognizer class continuously listens to audio from a microphone and converts the speech to text. In this project, the DictationRecognizer results are displayed in the Debug panel of the UI, discussed in a later section.

To prevent the app from recognizing and attempting to respond to the audio playing from the computer’s speakers, a condition was added. If Viva is speaking, then recognition is turned off using a stop function. Otherwise, if the TTS audio source has stopped playing, then the DictationRecognizer is re-enabled and Viva is listening again. This feature is also visible in the UI Debug panel.

Slight corrections were made to the resulting recognized text before deployment to Dialogflow. For example, the DictationRecognizer often picks up the word “Merion” as “Marion” or “Marian”. I encountered and corrected several misrecognized words during testing; however, there are likely to be more. Each recognized word is added separately to the resulting string, which made programming the corrections easier.

In the next section I explain the purpose of Dialogflow, motivation behind its use, and implementation details.

5.4 Dialogflow AI Agent Creation

Dialogflow is an online end-to-end, build-once deploy-everywhere development suite for creating conversational interfaces for websites, mobile applications, messaging platforms, and IoT devices (Dialogflow, Google Cloud). The service is used commercially by many well-known companies, including Comcast, Domino’s Pizza, and Mercedes-Benz, but also offers a free plan with generous data limits for individuals to create chatbots of their own (Dialogflow, Google Cloud). The tool is well-documented and relatively easy to use without extensive programming knowledge.

Motivation

I wanted Viva to be able to recognize a variety of phrasings of each question without having to hard-code each one by hand. Dialogflow allows developers to create lists of intents, entities, and knowledgebases to simulate artificial intelligence in their apps and achieve this goal. Using machine learning, Dialogflow is able to use training phrases and the aforementioned lists to detect the intent behind a user's input, whether it be a question, command, or statement, and match that intent to the correct response. The following subsections further explain the setup of Project Viva's Dialogflow agent.

5.4.1 Setup

Before training a Dialogflow agent, I wanted to be sure that the API calls would work in Unity. Thus, I started by creating a free Dialogflow account, blank agent, and an API key. Figuring out how to connect to the API took slightly longer than expected due to issues with the library files described in Section 5.5 Unity Project, and due to the fact that Dialogflow was recently upgraded to version 2.0 so many tutorials are now outdated. Connecting to Dialogflow in Unity requires a project ID, session ID, and API key. Two types of queries can be submitted to Dialogflow, either text or audio. Although audio queries were experimented with, this project submits solely text queries to the Dialogflow service. The text is produced from the DictationRecognizer described in Section 5.3.3.

5.4.2 Intents & Entities

A Dialogflow intent categorizes a user's intention for one conversation turn (Dialogflow, Google Cloud). When a user says something to Viva, Dialogflow matches the expression to the best fitting intent in the online agent. In order for Dialogflow to match an intent correctly, the developer must provide a set of training phrases. Training phrases are examples of what a user might say. When an input phrase resembles one of the training phrases for an intent, the input is matched to the intent and the response for that intent is given. It is recommended that developers provide around 10 training phrases for each intent, as Dialogflow uses machine learning to expand the list with other similar phrases behind the scenes. From each training phrase, Dialogflow can extract variables called entities. The list of possible entity values for an entity type is defined by the developer but can also be added to automatically during agent training if desired. Figure 8 below illustrates an example of how training phrases, intents, and entities work together.

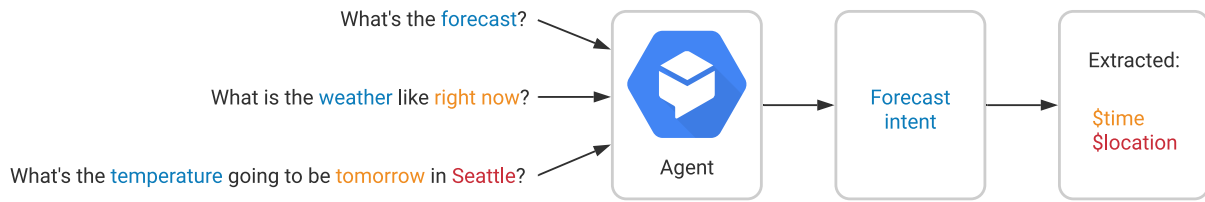


Figure 8: A diagram reproduced from the Google Cloud Dialogflow V2 documentation. The sentences on the left are training phrases, and each highlighted word corresponds to an entity. The different entities are color coded. For example, the words “forecast”, “weather”, and “temperature” highlighted in blue could correspond to a “weather” entity. The words “right now” and “tomorrow” highlighted in yellow correspond to the “time” entity. Finally, the word “Seattle” highlighted in red corresponds to the “location” entity.

Dialogflow also offers several pre-built agents which contain training phrases, intents, and entities for tasks useful for many developers. For this project, the “Small Talk”, “Jokes”, and “Easter Eggs” pre-built agents were imported. The Small Talk agent provides default responses to a variety of fun questions and comments about the agent such as “How old are you?”, “Will you marry me?”, and “You’re annoying”. A few of these responses were later modified to give Viva more personality. For example, if asked “Will you marry me?”, one of the pre-built responses is “In the virtual sense that I can, sure”. If asked, Viva will also respond with “Aww I’m sorry, but I’m already Bryn-Married to my best friend”, to reflect one of Bryn Mawr’s many traditions and add a personal touch.

Intent and entity creation took a significant amount of time due to the limitations of my laptop. In total, 5 custom entities were created: dorm, library, outfit, sign, and tradition.

tradition

SAVE

☒ Define synonyms ⓘ
☐ Regexp entity ⓘ
☐ Allow automated expansion
☐ Fuzzy matching ⓘ

parade night	parade night
lantern night	lantern night
wtf week	wtf week, welcome-the-first-years week, hell week
may day	may day, mayday
class colors	class colors
college cheer	anass, anassa, anassa kata, the anass, the anassa, college chant
senior bell ringing	senior bell ringing, senior bell, taylor bell, taylor bell ringing, ring the bell, bell-ringing, taylor bell-ringing, senior bell-ringing, the bell, the bell tradition
owl in rock arch	owl in rock arch, stone owl, rock arch owl, rock owl, arch owl, protector of the college
athena	athena, patron goddess, offerings, offerings to athena, gift to athena, greek goddess athena
senior row	senior row
senior steps	senior steps
garden party	garden party, garden party girl
moon bench	moon bench
lantern night teas	lantern night teas
may day gifts	may day gifts, mayday gifts
customs week	customs week, orientation, customs group
general	general

Figure 9: A screenshot of the custom “tradition” entity created for this project in Dialogflow. Note that possibly entity values are defined on the right, while possible synonyms for each value are on the right.

Additionally, a total of 46 custom intents were created to cover:

- Questions related to each dorm (ex. “Tell me about Merion”, “I’m curious about Rock”, “Denbigh” etc.)
- Questions related to each tradition (ex. “What is May Day?”, “What’s the deal with Lantern Night?”, “What are the class colors?” etc.)
- Questions about the current weather (ex. “Tell me the weather”, “What is the weather today”, “Weather forecast please” etc.)
- Questions about the libraries (ex. “Tell me about Collier Library”, “I wanna learn about Canaday”, “Talk about Carp” etc.)
- Questions or commands asking Viva to change outfits
- Questions or commands asking Viva to change or reset the background
- Questions or commands asking Viva to send a text
- Questions or commands asking Viva to send an email
- Questions or commands asking Viva for the daily horoscope
- Questions or commands asking Viva to show or hide the campus map

With several intents and entities programmed, it was time to substantially expand Viva’s capabilities by adding a Knowledgebase.

5.4.3 Knowledgebase

This feature is only available through Dialogflow version 2 beta 1, which is the Dialogflow version Project Viva uses. Knowledgebases represent a collection of “knowledge documents” which a developer provides to Dialogflow. The collection of documents contains information, usually in the form of questions and answers, that may be useful to the user. Dialogflow offers two methods of creating knowledgebase documents.

Information can either be extracted from a general document, or an FAQ page. For a general document, Dialogflow will use extractive answering to try and provide the user with the information they are seeking.

In this project, only FAQ documents were used. The knowledgebase feature is still in development; thus, I found the extractive answering method to be too inaccurate for my purposes. Question and answer data was manually extracted from the Bryn Mawr College FAQ pages. Figure 10 below shows all of the documents and sources used to create the Viva Dialogflow knowledgebase.

Document Name	Knowledge Type	Mime Type	Source/Path
Chemistry	FAQ	text/html	https://www.brynmawr.edu/chemistry/faqs
Clubs	FAQ	text/csv	File uploaded
Community Day of Learning	FAQ	text/html	https://www.brynmawr.edu/community-learning/community-day-learning-2020/answers-frequently-asked-questions
Convocation and Commencement	FAQ	text/html	https://www.brynmawr.edu/conferences/frequently-asked-questions
Credit/No-Credit	FAQ	text/html	https://www.brynmawr.edu/registrar/creditno-credit-faq
Dance	FAQ	text/html	https://www.brynmawr.edu/dance/faqs-0
Dining Services	FAQ	text/html	https://www.brynmawr.edu/dining/faqs
Dorm	FAQ	text/csv	File uploaded
Dorm Leaders	FAQ	text/csv	File uploaded
Economics	FAQ	text/html	https://www.brynmawr.edu/economics/faqs-students
First Year	FAQ	text/csv	File uploaded
Health Center	FAQ	text/html	https://www.brynmawr.edu/healthcenter/frequently-asked-questions
International	FAQ	text/html	https://www.brynmawr.edu/admissions/international-faqs-frequently-asked-questions
Math Department	FAQ	text/html	https://www.brynmawr.edu/math/frequently-asked-questions
One Card	FAQ	text/html	https://www.brynmawr.edu/onecard/faq
Q-Project	FAQ	text/html	https://www.brynmawr.edu/qproject/frequently-asked-questions
Reserving Space	FAQ	text/csv	File uploaded
Residential Life	FAQ	text/csv	File uploaded
Writing Center	FAQ	text/html	https://www.brynmawr.edu/writingcenter/students/faq

Figure 10: A table of information about my Bryn Mawr Knowledgebase showing all of the documents Viva can pull questions and answers from. Note that some webpages, such as the Clubs, Dorm, Dorm Leaders, Reserving Space, and Residential Life pages were not properly formatted for Dialogflow and had to be manually transferred into two-column CSV files and uploaded instead.

The Knowledge base allows Viva to cover questions such as:

- “What is Engineering Club?”
- “Do we need tickets for Commencement?”
- “Can I dine at Haverford or Swarthmore?”
- “What is a Customs person?”

While the responses are not always perfect, the Knowledge base made expanding the agent’s capabilities much more efficient.

In the next section, I describe the essential configuration settings of my Unity project.

5.5 Unity Project

General Settings

Project Viva was created using Unity version 2019.3.0f6 Personal edition with all scripts written in #C using Microsoft Visual Studio Community 2017, version 15.9.7. In order for my project to be compatible with multiple APIs, it was essential to change a few settings. First, in the Build Settings the project was set to PC, Mac & Linux Standalone, with Windows selected as the target platform. Although the project may run on Mac and Linux, this has yet to be tested. Next, in the Project Settings under the “Player” tab, the API Compatibility level was set to .NET 4.x. This was essential for the Google API library files to work properly.

NuGet Packages

The majority of the library (.DLL) files necessary for this project needed to be imported from the NuGet Package Manager in Microsoft Visual Studio. Unfortunately, although NuGet Packages can be imported to Unity projects through the auto-generated Visual Studio project solution, these files are not actually recognized by Unity without some reorganization. In this project, I needed to download several library files in order to use the Dialogflow and Microsoft Speech APIs. By default, NuGet Packages are downloaded to the “Packages” folder in a Unity project.

Unfortunately downloading these library files triggered roughly 99 errors in my project which prevented compilation. After a few days of head scratching and panic, I discovered that library files from NuGet must be placed inside the Assets folder of the project in order to be recognized. Thankfully this resolved the issue and allowed me to move on to the next stage of development.

5.6 Character & UI Design

This section describes the reasoning and process behind the design of Viva’s character model and the project UI.

5.6.1 Character Design & Animations

I had originally intended to use a character model in the same format (.PMX or .PMD) and in a similar cartoonish style as Mei from the MMDAgent sample; however, after developing the prototype shown in Figure 11, I decided that this style did not match my

vision for Project Viva. My prototype used a plugin called MMD4Mecanim to import and convert .PMX and .PMD files to .FBX, but there were some compatibility issues with the model shaders when the app was built and tested.



Figure 11: The prototype of my app featuring a cartoonish .PMX model similar to Mei-chan from the MMDAgent sample. This character model was created by user “Jjinomu” on DeviantArt and modified slightly for the prototype. As shown, there were some slight issues with the model’s hair physics causing strands to clip through the character’s body.

Instead, the character builder included with the popular video game “Sims 4” was used to create the Viva character model. The Sims 4 builder allows players to customize virtually everything about their character or “Sim”, including its eye color, skin tone, hair, clothes, makeup, and accessories. Players can also customize essentially every part of the Sim’s body and face by dragging parts to the desired length and size. Viva was created to slightly resemble me in my last semester at Bryn Mawr, with long and wavy dark pink hair, brown eyes, and a casual outfit. Although the model could have been designed to be much more realistic, I decided to go with a semi-realistic style to avoid the uncanny valley phenomenon. Viva’s default t-shirt was originally blank, but the texture was modified to reflect an actual t-shirt available for sale in the Bryn Mawr bookstore! The logo belongs to and was designed by Uscape Apparel. Once Viva was designed, the final model and several alternative outfits were exported to Blender, a free and open-source 3D creation suite. All of Viva’s alternative outfits are shown in Figure 12 below.



Figure 12: A collage of all of Viva’s possible outfits. Outfit changes are triggered when the `changeOutfit` intent is recognized. The outfit names from top left to bottom right are as follows: pirate costume, witch costume, HA shirt (“Hall Adviser” shirt), blouse, punk outfit, mayday outfit.

Animations

Several animations were also imported from Sims and converted to work with the .FBX format using Blender. This includes idle, speaking, bored, happy, sad, confused, and many other animations. Unfortunately, due to time constraints, many of the animations are never played and are left to be implemented as described in Section 7 Future work. Viva currently greets the user with a handshake upon entry, plays an idle animation when speaking, and displays a proud animation after changing outfits.

Later in the project, I realized that a lot of the time Viva speaks while looking off into the distance instead of at the user. Thus, an `EyeController` class was written to ensure that Viva makes eye contact with the user when appropriate. This was achieved by updating the position of Viva’s eyes to always face the main camera. Eye movements are updated every frame to override any currently playing animations which already contain eye movements.

Lip-syncing

I had originally intended to use a pre-made lip-syncing script from the Unity Asset Store; however, this required the use of shape keys which Sims models do not have. Shape keys are used to deform objects into new shapes for animation (Blender Manual). Figure 13 below displays an example use of shape keys in Blender.

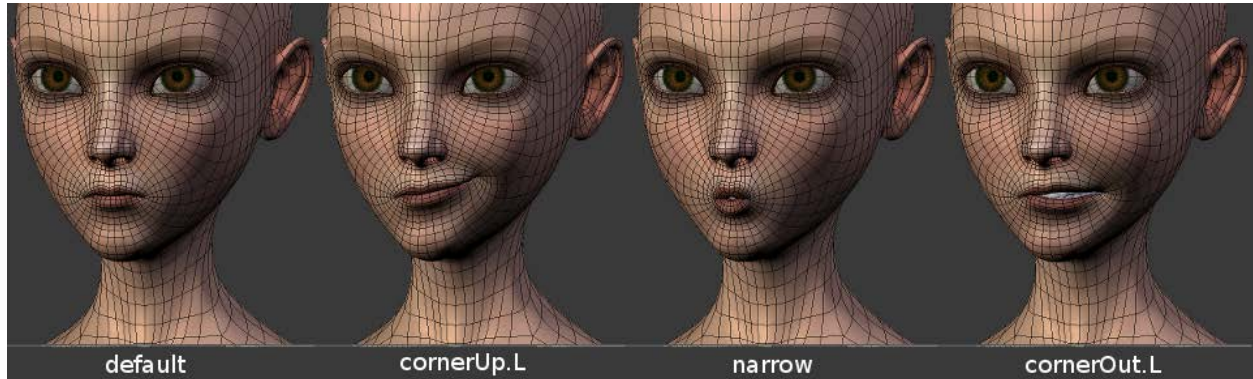


Figure 13: An example of a mesh with shape key facial animations reproduced from the Blender website. The leftmost face represents the base mesh, while the following three faces each represent a different shape key; in this case each is a facial expression. Once a shape key is created, a slider can be used to morph the base into the selected shape.

Instead, in this project facial expressions were achieved by armature or “bone” manipulation. Due to time and knowledge constraints, creating shape keys for each vowel and consonant was not possible. Thus, I opted for a simple, ventriloquist-like approach. My lip-script works by manipulating the jaw and lower-lip bones of the character model.

Copyright

As Viva was created with a Sims 4 tool, all rights to the character model belong to EA Games. This project will not be redistributed or used for commercial purposes as long as a Sims character mesh or animation is in use to avoid copyright infringement.

5.6.2 UI Design

Splash Screen

Upon entry, users are greeted by a splash screen which displays the Project Viva logo, author, and current version as shown in Figure 14. The splash screen appears very briefly and was included to give the final application a polished, somewhat professional feel.



Figure 14: The opening splash screen of Project Viva. The logo was created using Microsoft PowerPoint. The font used is “Kiona” which is free for personal use and which was created by Ellen Luff. The microphone headset icon used for the “O” in “Project” was created by Freepik and modified slightly to include a heart and smiley face.

Main, Settings, & Debug Menus

Project Viva is intended to be primarily a hands-free experience. Thus, the UI was intentionally designed to be minimalistic. The UI consists of three simple right-side menus. The main menu is shown expanded in Figure 16 below. The menus were designed using resources from the free Modern UI pack which is available on Github by user Michsky.

The purpose of the main menu is to remind users of the possible conversation topics available for interaction with Viva. At the moment the items in the list do not function as buttons; however, this possibility is discussed as Future Work in Section 7. The default state of the app at runtime is shown in Figure 15 below. The three side menus are shown in Figure 17.



Figure 15: A screenshot of Project Viva showing in the default state, with all three side menus closed. The top tab toggles the main menu, the middle tab toggles the settings menu, and the bottom tab toggles the debug menu.

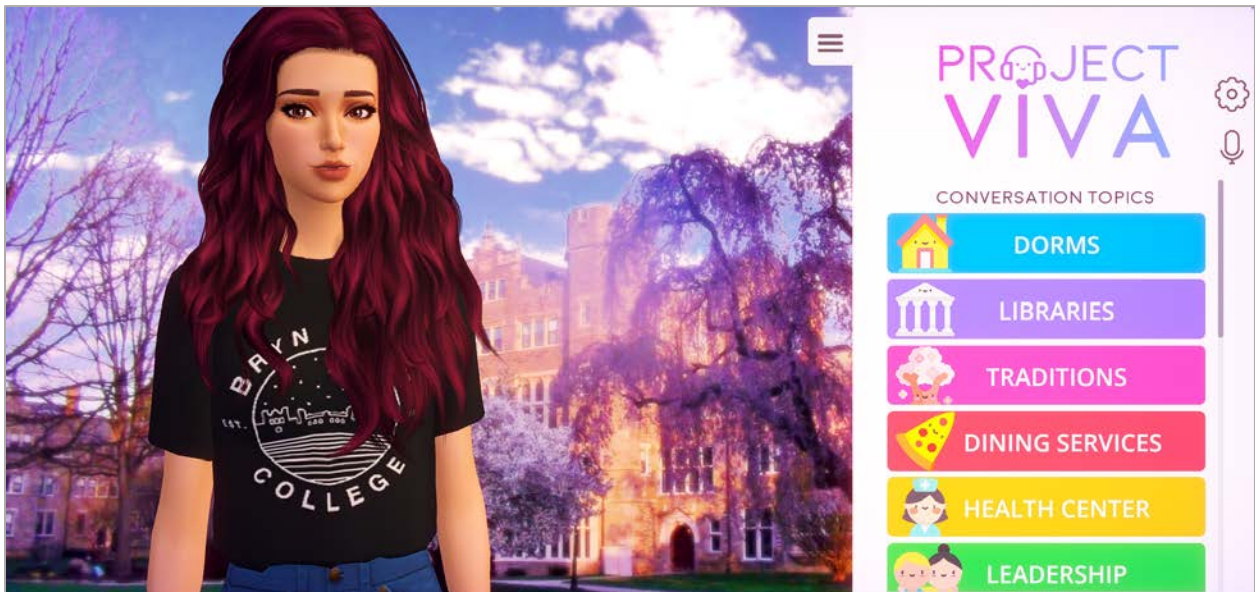


Figure 16: A screenshot of Project Viva showing the main menu extended.

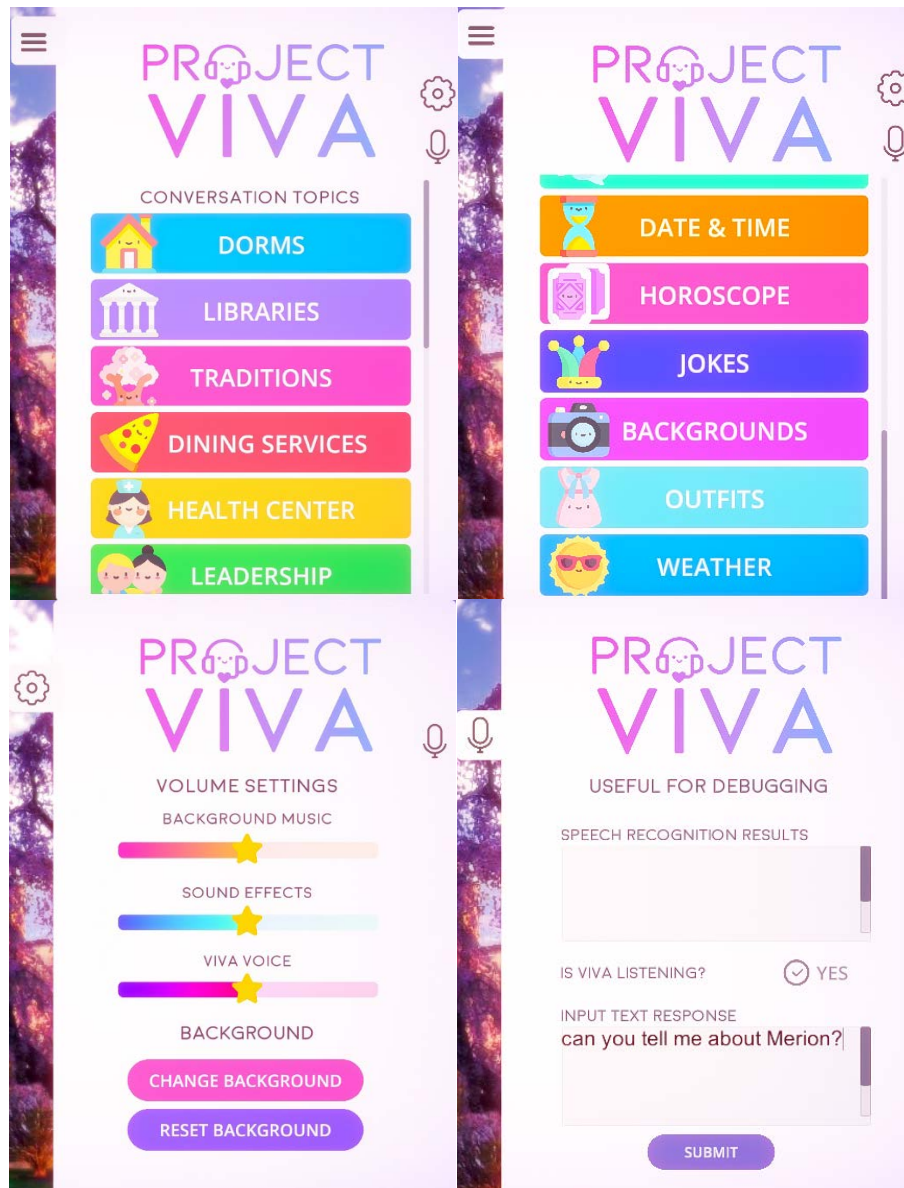


Figure 17: A series of four screenshots showing the main (top left and right), settings (bottom left), and debug (bottom right) menus of the Project Viva UI. In the main menu, each of the colored boxes represents a different conversation topic which can be brought up to interact with Viva. There are 16 topics total, including Dorms, Libraries, Traditions, Dining Services, Health Center, Dorm Leadership, Clubs, Email, Text Message, Date & Time, Horoscope, Jokes, Backgrounds, Outfits, Weather, and Campus Map. Additional questions may be asked about topics from the Knowledgebase, but these are the core options. The settings menu shows options to control the volume of the background music, sound effects, and Viva’s voice. There are also two buttons for changing and resetting the background (which can also be done through voice commands). The bottom right image shows the debug panel. Words recognized by the DictationRecognizer appear in real-time in the top scrolling window. A checkbox below indicates whether or not the DictationRecognizer is running or in other words if Viva is “listening”. The bottom scrolling window allows users to make text queries in the case that a microphone is unavailable or inconvenient.

Post-Processing

In order to achieve a vibrant and welcoming aesthetic, post-processing effects were added to the scene in several layers. The main effects used were Bloom, Vignette, Depth of Field, and Color Grading. The Bloom effect was used to soften the overall appearance of the scene and provide some luminance to the character model's skin. The Vignette effect was used to subtly darken the edges of the scene for a more realistic view. Depth of field effects were applied to slightly blur the background and focus the viewer's attention on Viva and the UI menus. Finally, Color Grading effects were used to add warmth and increase both the saturation and contrast of the scene to provide a more vibrant, welcoming result. Effects were applied in layers, such that Viva, the background, and UI elements all possess different settings for post-processing. Additionally, a yellow-toned directional light was added to the scene to shed light on the character model. Figure 18 below shows the main project scene before any effects were applied. Figure 19 shows the scene after the effects were added and tweaked thoroughly. It is evident by comparison that post-processing effects make a significant difference to the appearance of the final product.



Figure 18: A screenshot of Project Viva before any lighting or post-processing effects were applied.



Figure 19: A screenshot of Project Viva after lighting and several post-processing effects were applied in different layers.

In the next section I briefly discuss the additional APIs integrated into the project.

5.7 Additional API Integrations

OpenWeatherMap

The OpenWeatherMap API was used to allow Viva to check the current weather in Bryn Mawr. Users can ask Viva to check the weather in several ways; however, only the current weather in Bryn Mawr can be returned at this time. The API does allow for future and past forecasts, as well as any city in the US and abroad, but these features are out of scope for this project which is specific to Bryn Mawr.

Aztro Astrology

The Aztro API developed by Kumar et al. was used to allow Viva to check a user's daily horoscope. The API uses the KudosMedia Astrology website as the source of horoscope descriptions. Given a user's zodiac sign, Viva will describe the zodiac sign the user is especially compatible with for the day, their lucky time, lucky number, and lucky color. Viva will also provide the mood of the day (ex. "Peaceful"), and a description of the user's horoscope, which may include encouragements, warnings, or other advice.

Textbelt

The Textbelt API allows developers to send text messages from their applications. In this project, the free version of Textbelt was used to allow Viva to send users a text message

each day. Due to the limitations of the free API, Viva can only send one free text message a day in total, not per user. This feature was included to further enhance the interactivity of the app and can be expanded upon as described in Section 7 Future Work. This feature uses the “textMe” custom intent in Dialogflow which can be triggered either if the user asks Viva to text them, or if they just say a valid US phone number. Currently, the text message simply greets the user by name as follows: “Hi [username], it’s your friend Viva!”

SendGrid

The SendGrid API allows users to send email messages from their applications. Developers can send 100 free emails a day using the free plan. This feature was similarly included to enhance interactivity with Viva, but also to provide a way for Viva to disseminate Bryn Mawr related info to the user as described in Section 7 Future Work. This feature uses the “emailMe” custom intent in Dialogflow which can be triggered either if the user asks Viva to email them, or if they just say a valid email address. The user receives emails from the following address: your.friend.viva@gmail.com. The message is the same simple greeting used for the text messages. Ideas for the improvement of this feature are also discussed in Section 7 Future Work.

Mapbox

The Mapbox API provides developers with precise location data and powerful mapping tools to create accurate and highly customizable maps for a variety of applications. For this project, I had initially planned to use the Google Maps API to allow Viva to pull up a map of Bryn Mawr College upon a user’s request, however this option was not free. Thankfully, Mapbox offers a very generous free tier which allows up to 50,000 maps to be loaded on a webpage or in a webapp each month. For this project, I created a custom map style to match the simple and friendly UI design of the menus. As a base, I used the “Picture Book” style created by Saman Bemel Benrud. Originally, the map did not display POI (point of interest) labels, lots of trees, and buildings. These features were added in my custom style to show the trees of Senior Row, and to ensure that each Bryn Mawr building was shown and labeled properly. Figure 20 below shows the finalized map.



Figure 20: A screenshot of my custom campus map made with Mapbox. Each POI (point of interest) is labeled with blue text. Each building is shaded in tan. It was possible to replace the green background with the satellite view of Bryn Mawr, however the map was intentionally created in a minimalistic style to match the app’s UI design. Possible improvements are discussed in Section 7 Future work.

6 Discussion

6.1 Results & Analysis

The evaluation of my project is broken into two sections to reflect the custom voice building and application development stages of this project respectively.

Custom TTS Voice

As expected, while my custom TTS voice bears likeness to my actual voice, the final result was somewhat robotic. This could be the result of many factors. During recording, the transcript sentences may have been read too monotonically, or there may not have been enough recordings to train the model properly. Although there is no strict requirement for the number of recordings required to build an English TTS voice, the recommendation is 2000 or more clips for other language locales. My voice model was training using the statistical parametric method described in Section 3 Background. Because this method involves parameterizing the input speech, a large context is essential for a high-quality product (King, 2010). Although the voice does a good job of

pronouncing most words, some words such as “Bryn Mawr” are pronounced strangely. In this example, the word “Mawr” is drawn out slightly too long. In my opinion, the voice does not sound very friendly and sounds very serious. This again may be due to the fact that during recording, I had tried to remove any inflections from my voice and as a result some of the recordings may sound strained or too forceful.

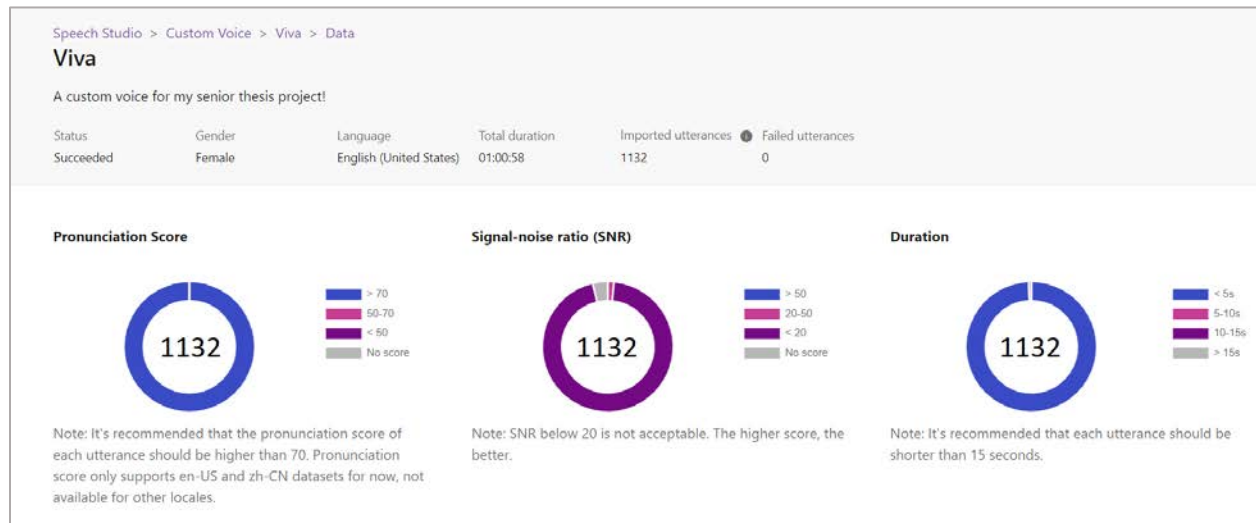


Figure 21: Analysis of my audio data generated by Microsoft Azure’s Custom Voice service. For the majority of my audio recordings, the pronunciation score for each utterance was over 70, the signal-noise ratio (SNR) score was less than 20, and the duration was less than 5 seconds.

As shown in Figure 21 above, the Microsoft Azure service provides three diagrams to help developers evaluate the quality of their custom voices and determine where improvements could be made. The leftmost diagram in the figure represents the pronunciation score. As indicated by the blue coloring, most of my recordings had a pronunciation score of 70 or more. This means that the recordings were clear enough that the words in each transcript sentence could be matched accurately, so pronunciation likely did not detract from the voice quality. The middle diagram provides an overview of the signal-noise ratio (SNR) for all of the clips. As indicated by the mainly purple coloring, the majority of my voice clips had an SNR of 20 or less, which is “not acceptable” according to Azure. This is likely due to the fact that compared to the CMU Arctic voice clips, my voice clips were quieter. I had considered manually boosting the volume of each clip but did not due to time constraints and in order to avoid making the background noises louder as well. To solve this problem, I would need a designated sound-proof recording space where I could speak as loudly as desired. The rightmost diagram represents the duration of all of the clips. As indicated by the blue coloring, the majority of audio clips were 5 seconds or less in length.

Although the voice is certainly imperfect, I consider this portion of the project to be a success considering all of the factors involved and the fact that the voice is not “jumpy” in the sense that sound transitions are not audible.

Final Application

To evaluate the success of my final interactive voice agent, I asked Viva a set of 20 questions and collected data on the intent that was detected and confidence of the match for each query. Questions were chosen randomly from both my custom intents and custom knowledgebase.

As shown in Figure 22, some questions were asked multiple times. This was done either because the first attempt failed due to a DictationRecognizer mistake in recognizing the sentence spoken, or to give the agent another chance at recognizing the correct intent.

Along with the agent response and various other parameters, Dialogflow allows the confidence of each intent matching attempt to be returned, as reported in the last column of Figure 22. The average confidence percentage was 85% (the three questions that resulted in a “null” confidence were excluded from the calculation). Out of the 20 total questions, 14 were matched correctly, thus the agent’s accuracy for this dataset was 14/20 or 70%. Taking a closer look at the data, we can see that the agent performs very well on the manually created intents. Out of the 20 questions asked, 6 questions in total (30%) were matched to the wrong intent.

Input Phrase	Recognized Phrase	Actual Intent	Recognized Intent	Confidence
My name is Nadine	my name is nadine	name.user.save	name.user.save	100%
Tell me about the dorms	tell me about the dorms	dorms.description	dorms.description	100%
Merion	merion	dorms.merion	dorms.merion	100%
What's the deal with Lantern Night?	what's the deal with lantern night	traditions.lanternnight	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg (Night Owls)	69%
What is engineering club?	what is engineering club	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	97%
Can I bring a microwave to campus?	can I bring a microwave to campus	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	96%
What's the weather like?	what's the weather like	weather	weather	100%
Can you tell me about Brecon?	can you tell me about brecken	dorms.brecon	Default Fallback	null
B R E C O N (spelled out)	BRECON	dorms.brecon	dorms.brecon	100%
Can I dine at Haverford?	can I die at haverford	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	Default Fallback	null
Can I dine at Haverford?	dinat haverford	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	Default Fallback	null
Can I dine at Haverford?	can I dine at haverford	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	96%
Email me	email me	emailMe	emailMe	70%
What is Lantern Night?	what is lentor night	traditions.lanternnight	smalltalk.greetings.goodnight	85%
What is Lantern Night?	what is lintern night	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg (Night Owls)	69%
What is a Peer Mentor?	what is a peer mentor	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	97%
What is the Community Day of Learning?	what is the community day of learning	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	98%
What is Bev Club?	what is bev club	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	97%
What happens if my OneCard is lost?	what happens if my one card is lost	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	Knowledge.KnowledgeBase.ODM3MTQ0NjIzODM5ODExOTkzNg	90%

Figure 22: A table of 20 random questions asked to the Viva agent. Questions were asked to include information which triggers both intents and knowledgebase responses. The rows highlighted in green signify correctly matched questions. Rows highlighted in yellow signify questions mismatched due to speech recognition errors, and rows highlighted in red signify questions mismatched due to agent errors. Note that both yellow and red highlighted rows represent incorrect results, thus the accuracy for this dataset is 14/20, or 70%.

Three of these intent errors were due to issues with the agent itself, while the other three were due to flawed speech recognition by the DictationRecognizer. It is also worthy to mention that 3 (50%) of the mismatched inputs were supposed to be matched to a knowledgebase entry. This is important to note because the knowledgebase feature is still in development and thus is expected and known to have various bugs.

The remaining mismatched inputs were expected to be matched to manually programmed intents. This could be for several reasons; however, after some research I found that many developers encounter this issue. Although the intent and any necessary entities are

programmed correctly, sometimes Dialogflow fails to recognize the entities and thus also the intent. For example, although the phrase “What’s the deal with Lantern Night?” was recognized correctly by the DictationRecognizer, but the agent matched the question to the knowledgebase entry for the club “Night Owls”. Upon investigation, the training record shows that the question was marked as the Default Fallback Intent (which often returns the phrase “I’m sorry, I didn’t get that. Can you say that again?”) because Dialogflow did not auto recognize the traditions entity “traditions.lanternnight” as expected. Figure 23 below shows the incorrect behavior of the Dialogflow service on the left while on the right my manual corrections are shown.

USER SAYS	what's the deal with lantern night
INTENT	Default Fallback Intent
USER SAYS	what is the deal with lantern night
INTENT	Default Fallback Intent
USER SAYS	what's the deal with lantern night
INTENT	traditions.lanternnight
USER SAYS	what is the deal with lantern night
INTENT	traditions.lanternnight

Figure 23: Two screenshots showing the incorrect (pictured on the left) and manually corrected (pictured on the right) behaviors of the Dialogflow service. The lantern night intent was not properly triggered because the “traditions.lanternnight” entity was not auto-recognized by the system.

While the above sample size is small, the patterns described were evident throughout the testing and training phases. Considering that the knowledgebase feature is still in beta testing and the fact that my model has not been extensively trained on user input yet, I consider a 70% accuracy with an average of 85% confidence in matching to be successful.

6.2 Challenges

In this section I will describe one of the major challenges faced during the development of this project.

Initial Approach to Custom Voice Creation – MaryTTS

One of the most popular and accessible methods of custom text to speech voice creation involves the use of MaryTTS. MARY, which stands for Modular Architecture for Research on speech Synthesis, is an open-source, multilingual text-to-speech platform written in Java (Schröder et al., 2011). After conducting extensive research into the tools available for custom voice creation, I had initially decided to use MaryTTS for my project. In particular, I was primarily interested in using the VoiceImportTools (VIT)

system. VIT is an open-source voice creation toolkit which supports the creation of both unit selection and HMM-based voices to be used in the MaryTTS platform (Schröder et al., 2011). The toolkit provides users with the tools and modules to create new voices and add new languages to the platform, and was created primarily for individuals such as myself and research groups who do not yet have their own speech technology to experiment with (Schröder et al., 2011).

While this tool seemed very promising, unfortunately I experienced many difficulties while attempting to build a test voice. In order to determine if the tool would be a viable option for my project, I attempted to re-build one of Carnegie Mellon’s CMU Arctic voices by extracting the source voice recordings and transcript file. The CMU Arctic voice was created from 1200 utterances and was designed for use in speech synthesis (Black & Lenzo, 2007). Although only 1200 utterances were actually used to create the voice, the voice source files contained 1132 .wav files and a matching transcript with 1132 sentences. I later used this transcript file to build my final custom voice.

As I followed the VoiceImportTools tutorial in the MaryTTS documentation, I encountered countless errors. At first, I had tried to run the tool on my Windows 10 computer, but many of the dependencies either were easier to use with or strictly required Linux. Thus, I switched to working on a Linux virtual machine. Although I was able to get an older version of the tool to at least run on Linux, I was unfortunately unable to successfully build a voice. The tool validates and confirms that it is able to find all dependencies, but then provides an error when trying to run the baum-welch algorithm for labeling the audio files, stating that one of the scripts could not be found. I have verified that the script is in the correct folder and have attempted to modify the source code of the tool, but to no avail. Additionally, the tool is now deprecated. VoiceImportTools will soon be replaced by a new Gradle-based plugin; however, this new software is still under development.

I sought help on the official MaryTTS Github repository and received a response many weeks later, but by then I had already found an alternative tool as described previously in Section 5.1.1 Microsoft Azure Custom Voice.

7 Future Work

With additional time and resources, there are many improvements and expansions that would enhance this project, some of which are described in the sections below.

7.1 Speech Synthesis & Recognition

The quality and naturalness of my custom text-to-speech voice could be improved in the following ways.

Increase the number of recordings

There are several ways that Viva's custom voice could be improved. The first option would be to substantially increase the number of voice recordings used to train the model. For this project I was only able to record 1132 voice clips, the same amount used for Carnegie Melon's CMU Arctic voice. Although the voice is effective enough, more recordings would improve the quality and naturalness of the resulting voice model. The two most basic demo voices available on the Microsoft Azure Custom Voice webpage, "Jessa" and "Guy", were created with two hours of speech data. The 1132-line transcript provided by Carnegie Melon resulted in about one hour of audio, so perhaps recording another thousand clips or creating and reading a longer transcript would improve the results.

Improve the quality of recordings

As previously mentioned, it was difficult for me to find a quiet space for recording. Often times towards the end of the recording process, I had to wait until 2 or 3 am when either my dormmates or family members were asleep to record. This likely affected the consistency of the recordings as I often felt obligated to whisper, even if I knew no one could actually hear me. My custom voice would be greatly improved if recordings could be made in a sound-proof room similar to the booth used by the CMU Arctic team. This would allow for greater consistency and noise reduction.

Use a different training method

Additionally, Azure offers three different types of voice model training as illustrated in Figure 24. Due to the low number of audio recordings available, my voice model could only be created through the Statistical Parametric training option. However, with 6001 or more utterances, a voice can be created with the Concatenative training method, which results in a higher-quality, more natural-sounding voice.

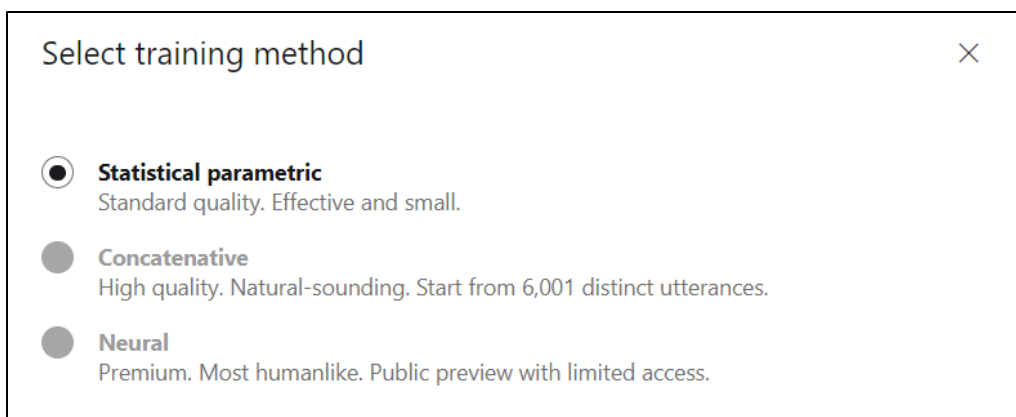


Figure 24: A screenshot of the training method selection popup in the Microsoft Azure Custom Voice creation portal. My custom voice was only eligible for the first option, statistical parametric model training which is described in Section 3 Background.

Alternative services

However, in the future I would ideally like to find an alternative to Microsoft Azure’s custom voice which allows for the export of the final voice model. This would allow for speech synthesis without an internet connection, with unlimited use, and the possibility of redistribution. I intend to continue exploring MaryTTS which offers all of these features at no cost.

7.2 Dialogflow Agent

With more time, the Dialogflow AI chatbot could be greatly enhanced with additional entities, intents, and knowledgebases. In particular, it would have been interesting to work with different departments at Bryn Mawr and LITS to format the FAQ webpages to be more compatible with Dialogflow agent knowledgebases, and to generate new pages covering more information. This would allow Viva to be easily expanded and answer many more questions about campus without the need to program more intents and entities, which can be very time consuming on a slow computer.

Open-source & Community Testing

Another way to easily improve my Dialogflow agent would be to publish this project online to enable additional users from the Bryn Mawr community and around the world to contribute to the training process, submit corrections, and program new intents. This is not currently possible due to copyright limitations described in the next section.

7.3 Character & Interface

Model

As mentioned previously, all rights to the Viva character model belong to EA Games as the model was created with the Sims 4-character builder. In the future, it would be better to build a character from scratch or from solely unrestricted resources. This would enable the app to be redistributed without the possibility of copyright infringement, and would add another personal touch to the app. Creating a 3D humanoid character model from scratch requires extensive knowledge of meshing, rigging, and physics, depending on how realistic the character is to be. Additionally, with enough time and knowledge, creating a character from scratch would allow for the creation of shape keys for more accurate and natural lip-syncing.

A fun and complex future extension could also be to build a character creator within the app to allow users to completely customize Viva's appearance at any point.

Lip-syncing

While the current simple lip-syncing method works for my purposes, the app would benefit from a more complex approach. Instead of taking the waveform data and manipulating the mesh based off of the volume of the audio, a Unity plugin such as SALSA could be used to recognize different vowel and consonant sounds to make the animation more realistic. Although I explored this possibility during development, I realized it would have taken too long to setup. Currently, the Viva character model uses armature or "bone" manipulation to move not only the body but also the eyes, mouth, and other parts of the face. The character model mesh would need to be modified to include shape keys, which are essentially facial expressions animations, for each vowel and consonant sound.

Animations

As mentioned in Section 5.6 Character and UI Design, many character animations were converted and imported but never implemented in the end due to time constraints. In the future, ideally Viva would display several different idle animations depending on how long the user has left the app running without any input. In addition to the imported animations, future work could involve creating new custom animations. These new animations could be very useful to demonstrate some of the Bryn Mawr traditions, similar to the way Customs people from every dorm perform skits to explain each tradition during Customs Week in Goodheart Theatre for the freshman class. For prospective

students, this would be more entertaining and informative than the text descriptions provided on the Bryn Mawr Traditions webpage.

Interface

As mentioned previously, the settings menu of the app has yet to be implemented. In the future, the settings menu should allow the user to manually change the backgrounds or select Viva's outfit, in the case that the voice commands are not working. Additionally, the settings menu should allow the user to select different background music and manipulate the volume of the different audio sources in the scene. Ideally, the main menu should also be updated to allow the colored blocks to function as buttons. The buttons could potentially allow the user to view example questions to ask, or more information about each conversation topic. The debug menu could also be improved to show more details about warning and error messages being logged to the console.

7.4 Application Capabilities

Singing Synthesis

My virtual agent can speak just fine... but can Viva sing? Similar to the text-to-speech technology used in the MMDAgent project, there are tools available to allow for singing synthesis, such as VOCALOID. Many of these tools appear to be for Japanese singing synthesis only, but if an English equivalent exists it would be fun and interesting to either use my custom TTS voice to allow song synthesis, or to create a new voice specifically for this purpose. This would mostly be just for fun, but in the Bryn Mawr context it would allow Viva to sing the traditions-related songs, such as the Sophias, to the user, or even help them practice the words to songs for Step Sings!

Navigation, Maps, & GPS

Although Viva can currently show the user a map of campus, in the future the user should be able to click on a building and learn more about it, or even take a virtual tour similar to the one provided by the interactive campus map provided on the Bryn Mawr College website. The map could also be upgraded to either show more detailed 2D art for each building or show 3D models of each building similar to the style of buildings which can be seen on Google Earth.

I would also love for Viva to have navigational or GPS capabilities. For example, the user would be able to ask Viva how to get to a specific location on campus, or maybe where the nearest café is off campus. I looked into this possibility during development, and it seemed like the Google Maps API would be useful, but of course comes at a cost.

Alternatively, I recently discovered and hope to look into OpenRouteService API, which is a free, crowd-sourced navigation API. Of course, this functionality would make more sense to be used on a smartphone or other mobile device.

Text messages

There is currently a limitation on the number of texts Viva can send each day. In a future version, I hope to either find an alternative service which can send more than one free text each day, or to figure out how to set up a free, self-hosted SMS server to use with Textbelt! This option seems very promising; however, the process doesn't seem to be well-documented, at least for someone with little web experience. If this feature could be implemented, Viva could send users information about Bryn Mawr directly to their smartphones. This could replace or enhance the request information process for the admissions office.

Email messages

Email messages could be improved in a similar way to text messages. While the number of emails Viva can send using the SendGrid API is not restrictive (Viva can send 100 emails a day using the free plan), at the moment Viva only sends a short message to the user which greets them by name. It would be interesting and beneficial to work with the admissions, residential life, and other offices to use this app to send students useful information to answer FAQs.

Platform

This project was tested and built for PC, Mac and Linux only. Ideally, the app would be built for mobile devices in the future, or even mixed reality devices such as the HoloLens. Unity makes porting apps to various platforms relatively easy. However, there would likely be a few compatibility issues with the various APIs used and their dependencies, the UI configuration, and the 3D models and shaders used. The Viva character model is likely too high poly for a mobile device, so the mesh would need to be simplified. If these issues could be resolved, I believe Viva would be a very useful mobile app for curious prospective and incoming students, as well as their parents. The app would be especially useful for students who are unable to visit campus in person before deciding to attend. Porting the app to a mixed reality device such as the Microsoft HoloLens would enhance the overall experience of the app by allowing users to interact with Viva using gaze and hand gestures in addition to verbal commands and questions.

Optimizations and Compression

At the moment, the app sometimes lags due to a number of factors. The character model has a high vertex count and requires a number of different shaders to be rendered properly, which can take a toll on the frame rate of the application. Although the post-processing effects greatly enhance the aesthetic of the interface, they could also be removed or optimized to reduce frame rate drop. The outfit changing system could also be improved. At the moment, all of Viva's possible outfits are loaded into the scene from the start. Inactive outfits are hidden from view and only re-enabled when the `changeOutfit` intent is triggered. Though slightly more complex, in the future outfits should be loaded into and removed from the scene at runtime to avoid unnecessary resource usage. Additionally, the project itself is around 4GB. In the future, unused and nonessential assets should be identified and removed to reduce the amount of space needed.

Moodle and Bionic Integration

Moodle and BIONIC are essential tools for every Bryn Mawr and Haverford student. It would be interesting and beneficial to users if Viva could connect to a user's Moodle or Bionic account to provide information ranging from grades to remaining tuition balances to course offerings. This could allow students to avoid the frustrations of the Bionic interface and have some fun in the process. Additionally, the Dialogflow AI agent could be trained to remember important deadlines and remind the user about upcoming or newly posted Moodle assignments.

8 Summary

In this project, I have explained the motivation behind creating a custom text-to-speech voice and virtual interactive agent application, provided background on statistical parametric speech synthesis and virtual reality, created a custom text-to-speech voice using my own recordings, created and trained a Dialogflow AI agent, integrated astrology, text, email, weather, map, Dialogflow, and Azure APIs into a Unity project, designed and lip-synced a 3D character model, and connected all of these components together to create a virtual interactive voice agent which can answer questions about Bryn Mawr College. The success of my results as described in Section 6 warrant further development of this project with several ideas suggested in Section 7. This project demonstrates the relative ease with which regular individuals, students, and hobbyists can create highly interactive and customized virtual experiences with the help of mostly free and open-source tools, and suggests that virtual agents and speech technology will continue to become increasingly involved in modern life.

References

- Yee, Nick, Jeremy N. Bailenson, and Kathryn Rickertsen. "A meta-analysis of the impact of the inclusion and realism of human-like faces on user experiences in interfaces." *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2007.
- King, Simon. "A beginners' guide to statistical parametric speech synthesis." *The Centre for Speech Technology Research, University of Edinburgh, UK* (2010).
- Mori, Masahiro. "The uncanny valley." *Energy* 7.4 (1970): 33-35.
- Mone, Gregory. "The Edge of the -Uncanny." *ACM*, 1 Sept. 2016, cacm.acm.org/magazines/2016/9/206247-the-edge-of-the-uncanny/fulltext.
- Earnshaw, R.. A., and Henry Fuchs. *Virtual Reality Systems*. Academic Press, 1993.
- Lee, Akinobu, Keiichiro Oura, and Keiichi Tokuda. "MMDAgent—A fully open-source toolkit for voice interaction systems." *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013.
- Higuchi, Yu. "VPVP - Tools for making Promotion Video of Vocaloid Series". *Vocaloid Promotion Video Project*, 24 Feb. 2008, http://www.geocities.jp/higuchuu4/index_e.htm
- McKeown, Gary, et al. "The SEMAINE Database: Annotated Multimodal Records of Emotionally Colored Conversations between a Person and a Limited Agent." The SEMAINE Database: Annotated Multimodal Records of Emotionally Colored Conversations between a Person and a Limited Agent - IEEE Journals & Magazine, IEEE, 22 July 2011, ieeexplore.ieee.org/abstract/document/5959155.
- Schröder, Marc, et al. "Open source voice creation toolkit for the MARY TTS Platform." *Twelfth annual conference of the international speech communication association*. 2011.
- A. W. Black and K. Lenzo, "Festvox: Building synthetic voices, Version 2.1," <http://www.festvox.org/bsv/>, 2007, (accessed March 2020).
- Kominek, John, Alan W. Black, and Ver Ver. "CMU Arctic databases for speech synthesis." (2003).

“Dialogflow.” *Google Cloud*, Google, cloud.google.com/dialogflow.

“Shape Keys - Introduction” *Blender Manual*, Blender,
docs.blender.org/manual/en/latest/animation/shape_keys/introduction.html.

Kumar, Sameer, Harshit Sahni, Aditya Dhawan, and Srijit S Madhavan. “Aztro – The Astrology API.” <https://aztro.readthedocs.io/en/latest>.

“KudosMedia Astrology.” *Astrology Daily Horoscope*, astrology.kudosmedia.net/.