Bachelor's thesis

Information and Communication Technology

2019

Iida Lintuaho

IBM WATSON IN A GAME PROJECT

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communication Technology / Game Technology

2019 | 45 pages

Supervisor: Principal Lecturer Mika Luimula, Adj.Prof.

Iida Lintuaho

IBM WATSON IN A GAME PROJECT

The most popular combinations of speech recognition and artificial intelligence are the virtual assistants by big corporations like Amazon's Alexa and Apple's Siri. In games and smaller applications the use of artificial intelligence features with voice control is almost nonexistent, since the development of both of these technologies usually require too many resources compared to the gained benefit. With the help of IBM Watson and other similar technologies this situation might change in the future as cognitive systems become more common. Watson offers an easy and affordable solution to every game and software developer to add artificial intelligence for understanding natural language, speech recognition and many other features to their products, something that would have been difficult to do previously.

The object for this thesis was to add new features to a virtual reality game with IBM Watson's Assistant, Speech to Text and Text to Speech services. This game, called A Day To Remember, aims to recognize dementia in an immersive way with small tasks and questions. With Watson the player is given instructions and they can control parts of the game with speech. Additionally, games and applications using speech recognition and natural language processing were researched along with IBM Watson's history, features and capabilities.

After implementing the Watson features it was clear that this technology can be beneficial in different game projects. With the Watson services it was easy to build elements that understand natural language. Based on the experience gained during this project and the research done on games utilizing speech recognition, it can be concluded that it is possible to create playable games able to understand speech and natural language with these tools.

The game was developed with Unity and C# in Turku Game Lab.

KEYWORDS:

Speech recognition, AI, artificial intelligence, IBM, IBM Watson, natural language processing, NLP

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka / Peliteknologia

2019 | 45 sivua

Ohjaaja: Yliopettaja Mika Luimula, dos.

Iida Lintuaho

IBM WATSONIN HYÖDYNTÄMINEN PELIPROJEKTISSA

Tunnetuimmat puheentunnistuksen ja tekoälyn yhdistämisen käyttötapaukset ovat suurten yritysten virtuaaliavustajat, kuten Amazonin Alexa ja Applen Siri. Peleissä ja pienemmissä sovelluksissa niiden käyttö on lähes olematonta, sillä laadukkaan tekoälyn ja toimivan puheentunnistuksen kehittäminen on yksinäänkin työlästä toteuttaa alusta asti, eikä tarve niille ole vielä ollut kovin suuri. IBM Watsonin ja muiden samanlaisten teknologioiden avulla tilanne saattaa muuttua tulevaisuudessa, sillä ne tarjoavat jokaiselle peli- ja sovelluskehittäjälle mahdollisuuden lisätä tuotteeseensa sekä puheentunnistuksen että luonnollista kieltä ymmärtävän tekoälyn.

Opinnäytetyön tavoitteena oli lisätä virtuaalisessa todellisuudessa pelattavaan peliin ominaisuuksia, joissa IBM Watsonin työkaluja hyödyntäen kehitetään puheentunnistuksella ohjattavia osia, jotka vastaavat pelaajan toimiin tekoälyn avulla. Kyseinen peli, nimeltään A Day To Remember, on suunnattu dementian tunnistamiseen immersiivisellä tavalla pienten tehtävien ja kysymysten kautta. Lisäksi tavoitteena oli tutkia sovelluksia ja pelejä jotka käyttävät puheentunnistusta ja tekoälyä, sekä oppia IBM Watsonista. Mitä ominaisuuksia se tarjoaa, ja miten hyödyllinen se voi olla pienissä peliprojekteissa?

Watson-ominaisuuksien toteuttamisen jälkeen oli selvää, että tämän teknologian palveluista voi olla hyötyä erilaisissa peliprojekteissa. Niiden avulla voitiin helposti rakentaa luonnollista kieltä ymmärtäviä osia. Ominaisuuksien kehityksen aikana opittujen asioiden ja puheentunnistusta käyttävien pelien tutkimisen perusteella voidaan päätellä, että näiden työkalujen avulla on mahdollista luoda peleihin toimivia puheentunnistusta hyödyntäviä toimintoja.

Peli kehitettiin Unity-pelimoottorilla ja C#-ohjelmointikielellä Turku Game Labissa.

ASIASANAT:

Tekoäly, puheentunnistus, IBM, IBM Watson, NLP

CONTENT

PICTURES

LIST OF ABBREVIATIONS

AI	Artificial intelligence
API	Application programming interface
NLP	Natural language processing
NLU	Natural language understanding
QA	Question answering
SDK	Software development kit

1 INTRODUCTION

Developing games has become approachable for everyone with a computer. With certain tools even programming skills are not necessary for creating playable games anymore and the internet is full of music and graphical assets to be used freely or for a reasonable price in one's projects. Many game engines for building the games are free or offer a free version with comprehensive documentations and tutorials. Open source softwares and a diverse selection of tools and software development kits (SDK) give every aspiring developer the chance to build innovative and modern applications and games utilizing recent technologies, such as speech recognition.

Speech recognition in games and applications is not a new concept, but it is still very much restricted to products by big companies and a few bigger games which either utilize it with short commands or base the whole game idea on it. Technologies like IBM Watson might change this situation in the future by making speech recognition and the understanding of natural language more available to developers.

In this thesis IBM's Watson technology is inspected closer along with the history and the current state of games and applications using speech recognition and artificial intelligence. After researching these areas, Watson's services will be used to implement speech recognition and artificial intelligence features to a game called A Day To Remember, a virtual reality game with an objective to help recognize dementia in the players. In the first version of the game Watson is used to make the player conversate with the game using their voice. For transcribing speech to written text and synthesizing text to natural sounding speech Watson's Speech to Text and Text to Speech services are used. The third service used is Assistant, which is a tool for building chatbots, which in this thesis is used to handle the dialog and understanding of natural language.

The game is developed in Turku Game Lab, a working environment by Turku University of Applied Sciences and University of Turku. The team working on this project handled all the graphical and audio assets as well as the basic mechanics that did not include Watson. All of the Watson features were developed as part of this thesis.

This project is part of the Futuristic Interactive Technologies research group and the Business Ecosystems in Effective Exergaming (BEE) project, funded by Business

Finland. The goal of the BEE project is to create new approaches to exercising and rehabilitation through games.

IBM Watson was chosen for this project and thesis mainly because IBM offers a SDK for Unity development. With the help of the SDK, developing the Watson features in the project can be started immediately without building the mechanics for utilizing the technologies in Unity first. The Watson services can also be used for free up to a certain point. This offers the possibility to test these technologies without a budget, which sometimes is the case in small projects with small teams.

This implementation is done with keeping in mind how effectively the Watson technologies can be utilized in a project of this scale, with beginner level developers with a minimal or non-existent budget. In the last part of this thesis the implementation is reviewed and the viability of current day speech recognition technologies for games, mostly small projects, is assessed.

2 SPEECH RECOGNITION AND ARTIFICIAL INTELLIGENCE IN GAMES AND APPLICATIONS

Making a game already takes up a lot of resources and has numerous aspects that need to be taken into account for it to be playable, let alone successful. Adding speech recognition on top of programming, creating graphical assets, writing a story and so forth increases the workload tremendously with new features to program, test and localize. In this chapter artificial intelligence, speech recognition and natural language processing are briefly gone over in order to gain an understanding of what are the past and current states of these technologies and how they are used in games and applications.

2.1 Artificial intelligence and games

Artificial intelligence (AI) is an area of computer science where the goal is to make machines and computers able to behave and perform tasks like humans on their own. This means the computer collects information, and learns, reasons and acts based on it. These actions are easy for humans, but usually present the hardest challenges for AI. (Brookshear 2007.) Defining what AI involves varies depending on what is considered intelligence and who is answering the question. This has led to different requirements for AI, for instance, it has to be able to learn and act accordingly or it has to have a conscience. AI technologies that meet these requirements are considered strong AI, and respectively those that do not, are called weak or narrow AI. AI in games almost without exception belong to the latter category (Bourg & Seemann 2004.)

Games have been an inspiration to AI researchers for decades. In the 1950s Alan Turing developed a chess playing computer program, A. S. Douglas created a digitalized Tic-Tac-Toe, and Arthur Samuel designed reinforcement learning to make a program able to learn how to play checkers. From there we have moved on to video games, either to use AI to play them, control elements in them or generating content for them. (Yannakakis & Togelius 2018.)

There are numerous reasons for why AI research has found its challenges from games time after time. Games offer many areas where AI can be applied, and there are many games and players creating data to process and utilize in research. For instance, games

require a lot of effort and there are usually many ways to complete them, so software has to learn how to play them. (Yannakakis & Togelius 2018.) Similarly, speech recognition and natural language processing and understanding used in games present challenges for the development of AI technologies as well. This is the case with IBM Watson and Jeopardy!, which will be explained in more detail in the next chapter.

Machine learning is a big part of the development of speech recognition and natural language processing. It is a part of AI, where systems learn and adjust their behavior to fit their intended purpose. The system performs certain tasks based on information it is given and receives feedback for those actions. Next time the system performs the same tasks it uses the feedback to adjust its performance and approach to them. Repeating this loop produces more knowledge for the system to use and act on, meaning that it is learning. (Zhongzhi 2011)

Speech recognition is not a new technology, it has existed for decades. The reason it has become increasingly popular is due to the improvements in the field. Nowadays speech recognition technologies can reach the same level of accuracy as humans, which is around 99%. Speech recognition has overcome challenges in order to be successful, such as being able to produce accurate results from different speakers in different surroundings. Language and pronunciation also change as time passes, so a speech recognition system needs to be updated to fit its intended purpose. A solution for this has been to utilize machine learning to make the technology able to learn from data and its user's behavior. (Geitgey 2016, Xuedong & Li 2010.)

Natural language processing (NLP) is an area of AI and computer science, where the objective is to make computers capable of understanding and generating natural language the same way humans do. Natural language understanding (NLU) is a part of NLP which covers the ability of understanding the meaning of words and phrases. For humans, speaking in natural language is the easiest way of communication, so making it possible to communicate with machines in the same manner would increase effectiveness. This has already been achieved in tasks like language translation, where machine learning has been used to improve the results. (Seif 2018, Donges 2018) This is a complicated field of technology and to understand how computers can be made capable to communicate with humans without using structured data would require the understanding of a vast area of technologies. In the next chapter IBM's take on this is presented to explain how Watson works, but it also works as an example of a computer able to communicate with natural language.

The relationship between AI and games is mutualistic, while AI research benefits from games, games also benefit. With AI, the player experience can be enhanced by tailoring the game to each player based on their behavior, making the game more interesting, for example by having more complex elements behaving in different ways. A game can also be optimized with the help of AI, for instance, generating elements for the game at runtime, which saves memory and can add replay value. (Yannakakis & Togelius 2018.)

2.2 Games with speech recognition

While implementing speech recognition is an effortful task, it does have its benefits. A game can be a lot more immersive when the players can execute tasks by voice, just like they would be done in real life. Speech recognition can also make games more accessible for players with certain disabilities, and possibly more approachable when commands can be given orally instead of a pressing buttons and keys after learning what the sometimes complicated key bindings are. Speech recognition is also used in games for learning a language and to help with pronunciation. (Van der Velde 2018, Joseph 2019)

The complexity of speech recognition in games varies from making any sound with a certain volume to control the game, like in Chicken Scream where talking in normal voice makes the chicken walk and screaming makes it jump, to Bot Colony where the players can ask questions from in-game characters by using natural language (Perfect Tap Games 2017, North Side Inc., 2013). In this chapter a few different games using speech recognition are used to show examples on how games can be controlled by speaking to them.

Game developers using speech recognition in games is still quite rare, although not that new. As early as in 1999 Vivarium released Seaman, a video game where the player takes care of a fish with a face, with whom they can have a conversation with. This was made possible by using and modifying the ELIZA program to fit the game. (Steinberg & al. 2016) ELIZA is program from 1966 developed by Joseph Weizenbaum, where users can write to ELIZA using natural language, and the program identifies context, keywords and phrases in the input and responds with an appropriate message making it seem like there is another person answering to them (Weizenbaum 1966).

Rest of the examples are over ten years newer than Seaman, but most of them do not utilize speech recognition for having conversations, but rather by giving short commands that trigger actions in the games. The first example of a game like this is Mass Effect 3 by BioWare and Electronic Arts, which utilizes Xbox Kinect's voice command abilities in the game. Players can give commands to AI teammates, switch weapons, interact with the environment and perform other functions with voice commands that the Kinect recognizes, that would normally require them to go through an ability wheel to access them. (Bioware 2012.) The Kinect's voice commands' main purpose is to navigate and use Xbox's menus and operating system. For instance, Xbox can be turned on and signed into, games and applications can be launched and volume can be changed (Microsoft 2019).

A similar approach to voice commands is used in Star Trek Bridge Crew. It is a game from Red Storm Entertainment and Ubisoft published in 2017 for PC and PlayStation 4, which can be played with or without a virtual reality headset (Ubisoft 2017). The goal of the game is to explore uncharted parts of space with your ship and crew, that either consists of other players or AI crewbots. These crewbots are partly built with IBM Watson's speech recognition and Assistant in order to make them capable of reacting to the commands given by the player with speech. These bots can fill any position and the game can be played and completed with them as well.

After the crewbot mechanics were implemented to the game, the development team started working with IBM Watson to bring the immersiveness to a new level. Instead of finding the commands from the in-game menu and issuing them to the crewbots, the player can just give the commands by speech, just like they would be given on a real vessel. Since Watson learns and can be taught to understand certain vocabularies, the development team trained Watson to understand words characteristic to the Star Trek universe so that the players can experience the game as it should. This also allows the players to give the commands in a more natural way using synonyms, meaning they do not need to use the exact word every time when issuing a certain command. (Ubisoft 2018.) The Watson services will be explained in more detail later in this thesis.

While Star Trek Bridge Crew is a good example of how to use speech recognition in a game, it does not fully showcase how understanding natural language can be used in playing and controlling games. Bot Colony by North Side Inc is one of the few games that is utilizing speech recognition and natural language technologies in an attempt to develop a game where the player can freely speak to it. Bot Colony is an adventure game

about a robot manufacturer where the objective is to investigate a case of missing robot technology. The main elements of the missions are executed with natural language by having conversations with robots and giving them orders. (Joseph 2019.)

Bot Colony uses mostly North Side's own technology for speech recognition, speech generation and dialog management. The interactions start from getting a message from the player, if it is a spoken message, it is first transcribed to written text. After that begins the process of understanding what the player is saying, and reacting to whatever it is, as long as the input is viable. To ensure that the game can understand the player as accurately as possible, it lets the player train their own acoustic model. With this model the speech recognition can transcribe the input with a higher accuracy, which is needed for the missions to be effortless to execute and the game to be engaging. (Joseph 2019.)

A rather new form of voice-controlled games and applications are called Skills, games that you play with Amazon's Alexa, which are controlled by speaking to the Alexa device. There are no other controls, user interface or graphics, only sound effects and a story voiced by actors. The player gives a command and the game acts accordingly. For example, in The Wayne Investigation where the player is a detective solving the murders of Bruce Wayne's parents, they can choose what evidence to inspect closer or decide what to do next by uttering the name of the item or action. These Skills are exclusive to Amazon's Alexa and are developed with the Alexa Skills Kit that gives developers the tools to create games and applications that benefit from the technology behind Alexa (Amazon 2019, Warner Bros 2016).

With the development of speech recognition and NLP it is probable that many more applications using these technologies will be released in the following years. Depending on the accuracy and effectiveness more games might start utilizing them as well. One of the challenges for this is that using keyboards, mouses and controllers is the most familiar way of playing games for many players, so adopting new ways of controlling might be a big change for many. Voice controls will have to be as efficient as traditional controllers for them to be successful and not frustrating to use.

2.3 Virtual assistants

Virtual assistants, sometimes called AI assistants or voice assistants, are their own distinctive group of applications using AI. They are applications that understand natural

language and assist their users to accomplish certain tasks, such as making a web search, writing a text message or putting on music on command. The best-known virtual assistants are Google Assistant, Apple's Siri, Amazon's Alexa, and Microsoft's Cortana. (Van der Velde 2018.) These softwares are the most widely used applications that utilize speech recognition and AI, for example, in 2018 Apple was approaching its two billionth shipped iOS device, most of which have had Siri installed on them since 2011 (Apple 2018, SRI 2019).

The beginning of virtual assistants dates back to 2003 when Defense Advanced Research Project Agency's (DARPA) launched the Personalized Assistant that Learns (PAL) program where the goal was to "develop software that could significantly advance the science of cognitive systems and revolutionize how computers interact with humans" (DARPA 2003). A part of this program was Cognitive Assistant that Learns and Organizes (CALO), which aim was to develop virtual assistants capable of decision-making and reasoning based on past experiences. This project would later in 2007 lead to the establishment of Siri Inc, which brought these virtual personal assistants available to the public. In 2010 Apple bought Siri and integrated it to iPhone 4S in 2011. (SRI 2019.)

Virtual assistants are becoming more and more capable of performing more complex chores, like making a phone call for the user in the background. In May 2018 Google showcased Google Assistant making reservations on behalf of the user for a haircut and a restaurant. A human-like bot, like IBM's Watson Assistant, made phone calls to the establishments trying to book these services with certain variables given by the user, like date and time of the day. If it was successful at its task, it would give a confirmation for the user about their appointment. All of this and more was done by utilizing AI to achieve the understanding and generating natural sounding speech by a computer. (Google Developers 2018.)

Speech recognition and AI software are not limited to phones anymore. For example, with Amazon's Alexa and Echo device other smart home devices can be controlled with them, like lights, speakers and robot vacuums. Similarly, other big companies like Apple, Nuance and Google have each their own versions of virtual driving assistants, which work like virtual assistants, but are specified to be used while driving and are meant to be accessed via car's own user interface rather than a phone. (Van der Velde 2018.)

3 IBM WATSON

After the success of the chess playing computer Deep Blue IBM's research department was moving on to a new challenge. That challenge came to be Watson, named after IBM's first CEO Thomas J. Watson, a computer that could compete against real people in the game show Jeopardy!. (Ferrucci & al. 2010.) In Jeopardy! the players are asked questions from a vast collection of categories, a correct answer will award the players money, and incorrect answers give them a penalty. The participants are given clues in the form of an answer and the answer has to be given as a question. (IBM 2019) For example, an answer could be "What is Watson?". This means it is more complicated for a computer to understand what the question, usually presented in an ambiguous way, is actually about.

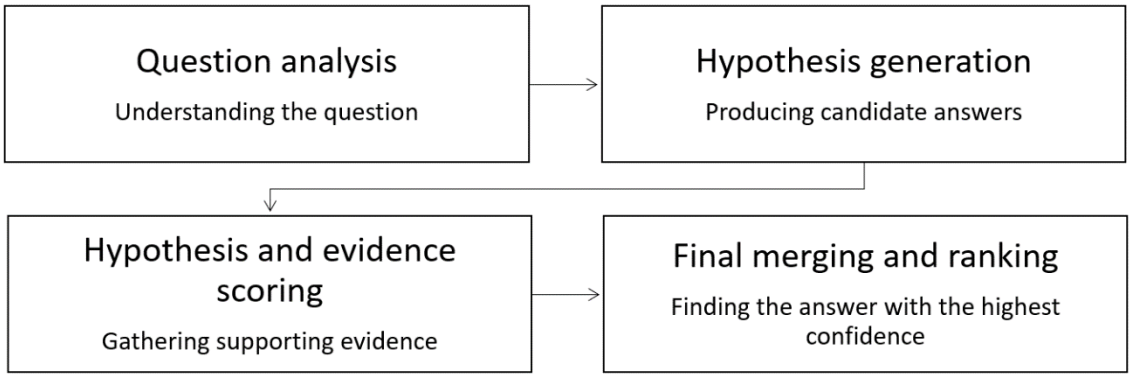
The development team had to take all these features into account when creating a computer that could be up for the Jeopardy! challenge. Watson would have to be able to take the question apart and understand natural language, and sometimes word plays and puzzles, to comprehend what it is being asked specifically. It would have to have access to knowledge and data from as many as possible topics, like science, history, literature and popular culture. In order to avoid receiving penalties Watson would have to be able to determine how confident it is with the answer it ends up with. Ultimately, for Watson to be able to compete against human contestants, all of this must happen in a matter of seconds. Some features of the game show were overlooked, since sometimes being successful in Jeopardy! is solely based on luck, so developing Watson to be able to respond to such situations would not be beneficial for the QA technology in question, which is about correctness, confidence and speed. (Ferrucci & al. 2010) To tackle this challenge, IBM Research utilized question-answering technology (QA), and begun the development of the Deep Question-Answering technology, DeepQA.

While partaking in Jeopardy! might have triggered the development of this new technology in the first place, and successfully competing in it helped to set the requirements for it, the real goal of developing Watson was to create new technology able to understand and process 80% of the data in the world, which is unstructured and unregulated, hard for computers to comprehend. This means finding connections, answers and trends in the data more efficiently than the current technologies, that don't really find answers to questions, but material that might contain the answer based on

found keywords. Being able to do this means Watson can help people make decisions in fields that use a lot of data and information, such as healthcare and education. (IBM 2019, IBM 2011, High 2012.)

3.1 DeepQA

Watson is a deep NLP system, running a software called DeepQA, Deep Question-Answering. While its first task was to answer questions in a game show, it can do a lot more. DeepQA finds connections and relations in information and can produce detailed solutions based on it. (IBM 2019.) In this part the process Watson goes through when finding an answer to question is presented in a simplified way.



Picture 1. DeepQA process for finding an answer (IBM 2011)

The process Watson goes through when it is asked a question can be divided into four steps, where numerous algorithms look at the question from different perspectives, trying to break it into understandable pieces in order to deduce the answer, which requires some level of confidence for Watson to present that answer as the correct one. These steps, shown in picture 1, are question analysis, hypothesis generation, hypothesis and evidence scoring and final merging and ranking. Before this process Watson is given resources it will use to deduct the answer since Watson is meant to be operated without an access to the internet. Those resources are encyclopedias, databases, books and any other sources of information a Jeopardy! question could be about. (IBM 2011, IBM 2019)

What happens first when Watson is presented with a question is question analysis. Here the system tries to understand what the question is specifically asking and plans what

the rest of the process is going to do in order to find the answer with the highest confidence possible. For example, the question is checked for a lexical answer type, which is a word or a phrase indicating what type of answer is suitable for the question. This means whether the answer is a person, country and so forth. Many different technologies are used to ensure the question is analyzed as correctly as possible. (Ferrucci & al. 2010)

In hypothesis generation candidate answers are formed based on the results acquired in question analysis. The object is to find as much content as possible that might contain the correct answer. Like elsewhere in DeepQA, many different algorithms and approaches are used in this search. After this, Watson will score, filter and evaluate the candidate answers generated to make sure the candidates are correct, since if the correct answers are not among these candidates, Watson will not be able to answer the question successfully. (Ferrucci & al. 2010)

In hypothesis and evidence scoring the candidates and their evidence that passed the previous steps are processed with many deep scoring analytics to determine what is the most likely answer. To make this evaluation more effective, the system gathers more evidence to support the candidate answers. With different scoring methods the evidence is evaluated for how confident Watson is that it will support the candidate answer properly. DeepQA supports many types of scoring techniques that evaluate different aspects, like lexical relations of words and phrases, and they can be adjusted to work in a desired way. For clarity, all of the results are added together to acquire one score for determining if the confidence is high enough. (Ferrucci & al. 2010)

In the last step, final merging and ranking, all gathered hypotheses and candidate answers are assessed for their probability of being the correct choice. After gathering all of the answers and evidence and scoring them, answers that hold the same information, for example person's last name and the same person's full name, are merged together so that they will not be compared to each other. These merged hypotheses and their scores are then ranked for their estimated confidence. The highest scoring one will be given as an answer if its confidence passes the threshold for buzzing in. (IBM 2011, Swiezinski 2013.)

3.2 Services

Currently IBM Watson offers 15 different services. All but one of these services can be implemented and used for free with free lite versions, up to a certain point. (IBM 2019) In the project A Day To Remember three of these services, Assistant, Speech to Text and Text to Speech, will be used and gone over in more detail to explain what they do. For this thesis only the free versions of the services will be needed.

The other services are Discovery, Visual Recognition, Natural Language Understanding, Natural Language Classifier, Personality Insights, Tone Analyzer, Language Translator, Watson Studio, Knowledge Studio, Machine Learning and Knowledge Catalog. These services use AI to help the user to use their data more effectively and build applications for different purposes. Many of these services and their features have been available for developers for a few years, in 2015 IBM released new services and features for already existing services, as well as older services for general availability for them to be more accessible to a larger audience (IBM 2019, Goyal 2015).

Text to Speech is an application programming interface (API) that converts written text to natural sounding speech using IBM's speech synthetization. At the moment the service can be used to synthesize speech in 10 different languages and dialects, and it offers features that allow the modification of the generated audio clip to fit the needs of the developer. For instance, the expressiveness of the tone of the speech can be adjusted with Speech Synthesis Markup Language to sound suspicious or joyful. (IBM 2018)

With Speech to Text service speech can be converted to written text in real time, either from microphone input or a sound file. Currently the service supports 10 languages and dialects and offers features for gaining more information and modifying the output to a suitable format, such as differentiating between speakers in the voice input. It uses machine learning and combines the knowledge of grammar, language structure and formations of audio to transcribe human voice to text. (IBM 2018)

Watson Assistant is fundamentally a bot that can be adapted to different use cases depending on the needs of the developer and is mainly marketed towards businesses for using machine learning to interact with their customers on digital platforms. It combines the technologies of NLU, machine learning and dialog tools to make it possible for humans to conversate with applications in an easy manner. (IBM 2019.) An Assistant

consists of one or more skills that contain the elements of the dialog it can have with the users. Depending on what the user is trying to do, the Assistant steers them to the right skill. These skills are navigated with intents and entities, which are tracked in the input received from the user. Assistants can be integrated to different platforms, for example Facebook Messenger, where it can be used to interact with customers. (IBM 2018)

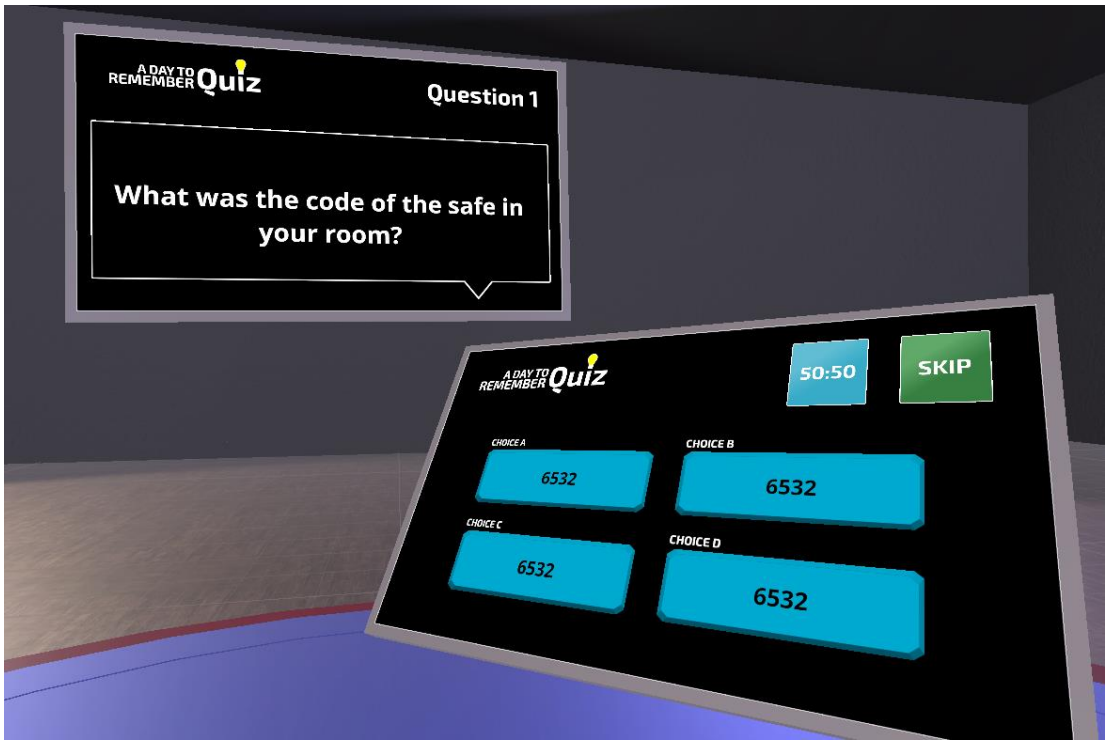
Intents are collections of words and sentences that mark what the user wants to do, such as booking a service or inquiring information on a certain subject. Assistant recognizes the intents and moves to a corresponding dialog node and continues the conversation with the user. Based on the words and sentences given to the Assistant, it can also learn to identify similar statements that mean the same thing. This gives the users the opportunity to communicate freely without having to use certain words in order to have the conversations. (IBM 2018.)

Whereas intents are like verbs, statements that mean the desired action, entities are nouns. Like intents, entities are recognized from the messages from the user, but with them the desired action can be specified, for example booking a service on a certain date at a certain time for a certain number of people, at a certain location. Assistant provides a set of ready-made generic collections of intents, related to for example banking and insurance, and entities like numbers, dates and times of the day that can be easily added to all dialogs and Assistants. (IBM 2018.)

4 A DAY TO REMEMBER

A Day To Remember is a virtual reality game that provides a gamified and immersive way to identify and track signs of dementia with the player as well as help them with their everyday life. The game is built for PC and is meant to be played with the HTC Vive virtual reality system, but since virtual reality is not very accessible for the masses yet, a desktop version might be a possibility in the future.

The game consists of three stages and each of them contains different tasks that require the player to memorize something. In the first stage the player wakes up in their home and they have to complete simple puzzles in order to move to the next stage. For instance, the player has to find a code in the house and enter it to a safe that holds an invitation that is needed to get to a game show in the last stage. After completing the first stage, the player moves on to the second stage, in which they go along a path in the woods and play a memory game. At the end of the path player gets into a bus that takes them to the third and final stage, where they take part in the game show. There they are asked questions about everything they have seen and done earlier in the game, for example about the code for the safe, as shown in picture 2.



Picture 2. Quiz stage setting

During the game two types of points are gained. The points that the player sees indicate how well they did in the game, similarly to other games. They only exist for the sake of entertainment. The second set of points, hidden from the player, indicate the level of dementia. If the player did well, their score is low. If it took too long or took many tries to complete the first stage, they did not manage to finish the memory game or they did poorly on the questions, their score is higher.

Since many of the potential players have likely never used a virtual reality headset or controllers, or even played that many games, the control mechanics are kept as simple as possible. All interactions in the game are executed with the controller's trigger button or with simple gestures by moving the controller, such as waving a hand. In order to prevent nausea and motion sickness when the player moves around, they will be teleported between predefined movement points.

To achieve a certain level of immersiveness, the visual style will be fairly realistic but simple, so that it will appeal to all players of different tastes. Also, this is where IBM Watson comes to play. For example, the question stage will be more immersive and feel more like a real game show where the questions are asked out loud and answers spoken by the player are received by a bot built with the Assistant service.

5 EXECUTION

This project was developed with the Unity game engine due to its familiarity to the development team and support for HTC Vive virtual reality system. The addition of Watson technologies to the project was decided after the development had started, but using Unity made the decision easier, because IBM offers an SDK for using Watson in Unity. Its pre-made methods and codes for using the services in a Unity project made it possible to start developing the features easily and fast from the start.

The free versions of the Watson services were sufficient for the project at first. It did not feel necessary to move up to the paid versions later since work in the project was not done full-time. In case the Assistant's API calls, Text to Speech characters or Speech to Text minutes would run out, they would reset by the end of the month and work could be continued.

For this project the best solution for having Watson powered features was to create the mechanics and necessary game objects in a separate scene and then adding them as prefabs to the actual scenes where Watson is meant to be used. Many other elements were built this way as well. The reason for this was that multiple people were not working in a same scene and causing problems in version control.

5.1 Planning the Watson features

The first thing the player does in the game is turning off an alarm which is shortly after followed by a phone call from Watson. The caller introduces themselves, currently as Watson, and asks the player for their name. After getting an answer Watson asks how the player is doing. In the first version player is expected to have either a good or a bad mood and the goal was to let them express that quite freely. After the pleasantries Watson starts to give the player instructions about the game. This information tells them about the game show, the invitation needed for it and so forth. The first set of information is always given and after that Watson asks if the player wants to hear more, to which they can accept or refuse. If the player does not want to hear more or they have heard all the information available, Watson wishes them good luck on the upcoming competition and ends the call. After this the player can start exploring their surroundings.

In the quiz stage, the goal was to let the player play it through only using their voice if they wanted to. This part was a lot more straightforward than the conversation part since it did not require the possibility of letting the player answer in various ways meaning different things. The answers are limited to four options along with “yes” and “no” answers and they can be expressed in numerous ways like “my answer is B”, “yes I want to answer this”, “no I would like to change my answer” or just “B” and “yes”. Despite the player using different phrases to answer, Assistant will be able to recognize the meaning of them as long as the answer option or some synonym for “yes” or “no” is present in the input.

The question stage mechanics could be created by only using Speech to Text and Text to Speech. Watson has very few lines to say and variation within those lines could be implemented by having few different options from which one would get randomly chosen and sent to the methods handling the communication with the Text to Speech service. Answer options could be recognized by instructing the player to say only “A”, “B”, “C” or “D”, and checking that with an if-statement. This could work as long as the player speaks clearly enough and the answers would get transcribed correctly every time.

After some testing with the Speech to Text service it turned out that sometimes the answer would get transcribed as other words that sound similar. For instance, saying “A” might be transcribed as “hey”. These words could be checked in the same if-statement, but still some words would be left out and the statement would grow unreasonably large. Using Assistant makes checking these words easier by adding them as synonyms for the answer options in their entity group. Similarly, the “yes” and “no” answers could be checked with an if-statement along with their synonyms like “yeah” and “nope”, but with Assistant this can be handled without an if-statement that might as well grow out of bounds.

These two parts make up the first versions of the Watson features in the game. After testing and making sure that this version works, more lines for Watson can be added. For example, synthesizing the questions with Text to Speech, as well as giving feedback for the answer.

Letting the player ask questions could add a lot more immersion to the game but limiting the area of topics to which Watson would be able to answer presents some challenges. The length of the conversation should not blow out of proportion, but at the same time it should not be too short in order to be realistic and fun to play. Implementing this requires

playtesting the first version with actual players in order to gain knowledge what features work and how easy it is to play with speech recognition without prior knowledge of this type of games.

Leaving these out of the first version is also partly due to saving the API calls, characters and minutes, so the development can be carried out with the free versions, since one of the questions of this thesis is to examine whether or not the IBM Watson technologies are beneficial for low-budget game development projects.

5.2 In general

To start using IBM Watson with Unity, the only two things needed are the IBM Watson software development kit, which at the time of this thesis can be downloaded from Watson's developer GitHub page for free, and an IBM Cloud account which is used to create and access the services and the workspace for building the Assistant's dialogs.

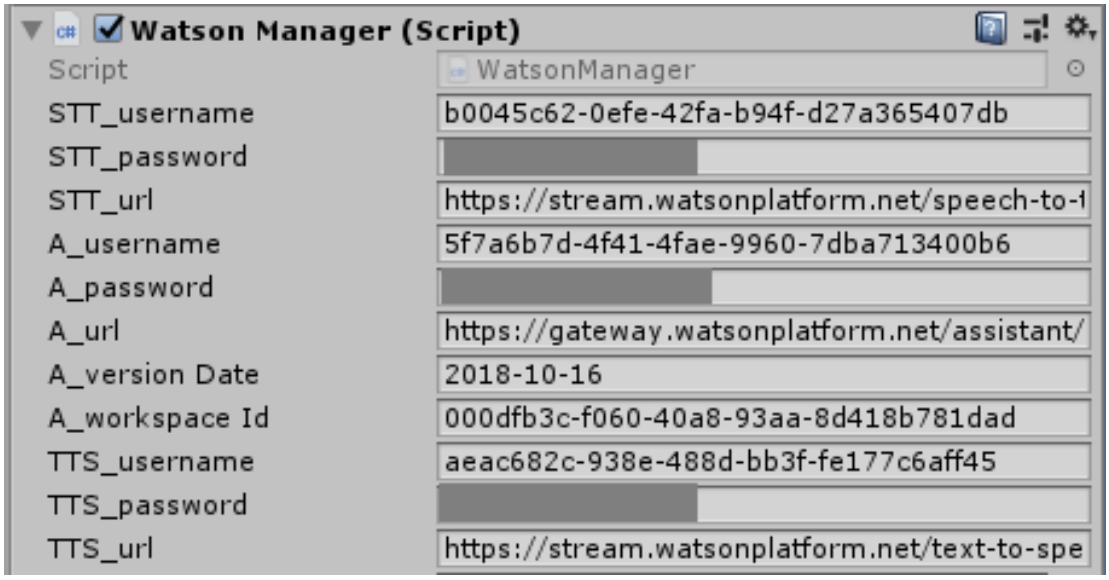
Each service has its own credentials to access them. Speech to Text and Text to Speech services only require these credentials and an address where to connect in order to access the service. For Assistant there are two additional fields of information for specifying on what skill from the workspace is being used and the version date. The dialog skills from the Assistant can be implemented to a Unity project individually, so creating an Assistant bot to access them is not required for this project.

In case the IBM Cloud account that is used to access the services needs to be changed, the Assistant dialogs can be imported to another account with JSON files. The dialogs will work just the same as long as the new credentials are used as well.

5.3 Preparing to use the Watson services in Unity

The first thing needed when starting to create the mechanics for Watson in the project was to create a script that manages every action related to Watson. Through the manager the game accesses the scripts for interacting with the services on IBM's servers. In this section each part of the manager script is reviewed individually and explained briefly what happens in them. For this project there is only one version of this script in which there are references to the stage-specific managers. Those managers handle most of the actions that happen after a Watson service has been used, for

example sending the player's chosen answer to a scoring script. This way the Watson manager stays as clear as possible.



Picture 3. Credentials and URLs to connect to the services in Unity inspector

When starting up a service, the first step is to connect to it on IBM's servers. This is done by using the credentials acquired earlier when creating the service. Each of the credentials are entered to the Watson manager script in Unity's inspector, as seen in picture 3**.** They do not need to be changed later, except for the Assistant's version date in case it has been updated. Both parts using Watson, phone call and quiz stage, get their own skill in the Assistant service, so two sets of Assistant credentials are needed.

The Watson SDK contains the necessary scripts needed to use the services and the manager script utilizes them in its own methods. All the methods and code snippets that do this already can be found from the example scripts, so following them will make the development of the project and learning to use this technology faster.

5.3.1 Speech to Text

For implementing Speech to Text service to the manager script and the game, not a lot of work is needed. The example scripts provided in the SDK do exactly what is needed here and have all the methods and functions for using the service. For the sake of keeping the code as easy to understand as possible, some of the function names are

changed to more distinctive ones. Some comments will also be added to explain what each part does.

There are three conditions being checked continuously. Should the manager script be playing a synthesized audio clip from Text to Speech, is an audio clip currently being played and has the clip finished playing. If there is no audio playing, the manager starts to record anything that is spoken to the microphone.

Input from the microphone is queued to be sent to the service in the Speech to Text's own script which handles the communication with the service. This will stop once the script is told to stop recording, either when the player has been silent or talking long enough. The input is continuously sent to the server, only short WAV files can be transcribed by sending the input in a single message to the service.

The Speech to Text's script takes a method from the manager script as a callback function and once it receives the transcribed text from the service it sends it to the manager to be handled in that callback function. This is done continuously as well. In the manager this text is added to a string storing the message and once all of the speech has been transcribed the string is put into a format that can be sent to Assistant, and then sent if that is what is needed. If there are words or commands that do not need to be sent to Assistant, the input can be checked in the manager and an according action will be triggered. For example, if "yes" and "no" answers were to be checked without Assistant's help.

5.3.2 Assistant

In this project a message is sent to Assistant in two different ways, either after receiving the transcribed text from Text to Speech or as written text from other scripts. In the first version of the game player's input is always sent to the Assistant and written text is for short commands for triggering dialog nodes in order to start or continue the conversations on behalf of the player. For example, sending the command "StartConversation" to the Assistant starts the phone call conversation in the first stage. Both of these ways have their own methods, the code snippet below is showing the mechanics for sending a command to the Assistant. The same mechanics can be found in the method for sending the transcribed speech to Assistant as well.

The message is sent to Assistant by passing the method for handling the message and what is to be done with it as a callback function to the Assistant's own script, along with the method for handling errors, the Assistant's skills ID and a message request containing the message for Assistant, as seen on the last line of the code snippet below. The earlier lines are for compiling the message into a form that can be sent forward.

```
public void SendMessageToWatson(string message)
{
    Dictionary<string, object> input = new Dictionary<string,
    object>();
    input.Add("text", message);
    MessageRequest messageRequest = new MessageRequest(){Input =
    input};
    _service.Message(OnAMessage, OnAFail, A_workspaceId,
    messageRequest);
}
```

From this point on everything is executed in the Assistant's script. Putting the message request in a suitable format to be sent to the server to interact with the dialog nodes and then finally sending it. Once the Assistant sends back a response, that response is passed back to the callback function in the manager script, in this case to a "OnAMessage" method. There context, intent and the message are accessed from it and saved to variables and dictionaries for further assessing.

This next part is not from the SDK's example scripts and is written for this particular project. After getting the context and the message the function checks which part of the game is being played currently. This is done by having the references to the stage-specific manager scripts and having only one of them enabled at a time. For this project this is enough for avoiding errors and conflicts since it is easy to manage the only two instances of the manager script.

Context of the message is used to trigger the necessary actions depending on the stage. For instance, in the quiz stage the answer given by the player is returned to the manager as context after it has been recognized from the transcribed input by the Assistant. If the context is not an answer option, the player has given an invalid answer and is asked to repeat what they said. The "yes" and "no" answers and interactions in the phone call are handled the same way. The actual messages from the dialog node are passed to the

Text to Speech related functions, so the player can hear what Watson is responding to their input.

5.3.3 Text to Speech

Text to Speech is used in two ways in this project, to let the player hear the Assistant's response and synthesizing a string of text coming from some other source than the player's input.

In the three checked conditions that were mentioned in the Speech to Text part, the first one was for checking if there is a synthesized audio clip that should be playing. When a message arrives from Assistant this is triggered, and the synthesizing process starts. When the string of text is not coming from Assistant but from somewhere else in the scripts, a similar function that gets the text as a parameter is called. The reason for having two functions that essentially execute the same action is because the message from Assistant is stored in a variable in the manager script. Rather than having a public variable that is changed from outside of the manager, another slightly different public function does the same action, but this time with a parameter. Both of these functions consist of two lines of code so maintaining clean code is not an issue. Example of the other function is shown in the code snippet below. Before starting the synthetization the different features of the synthesized voice can be adjusted. In this project only changing and specifying the voice used in synthesis is necessary in case using different voices becomes necessary at some point.

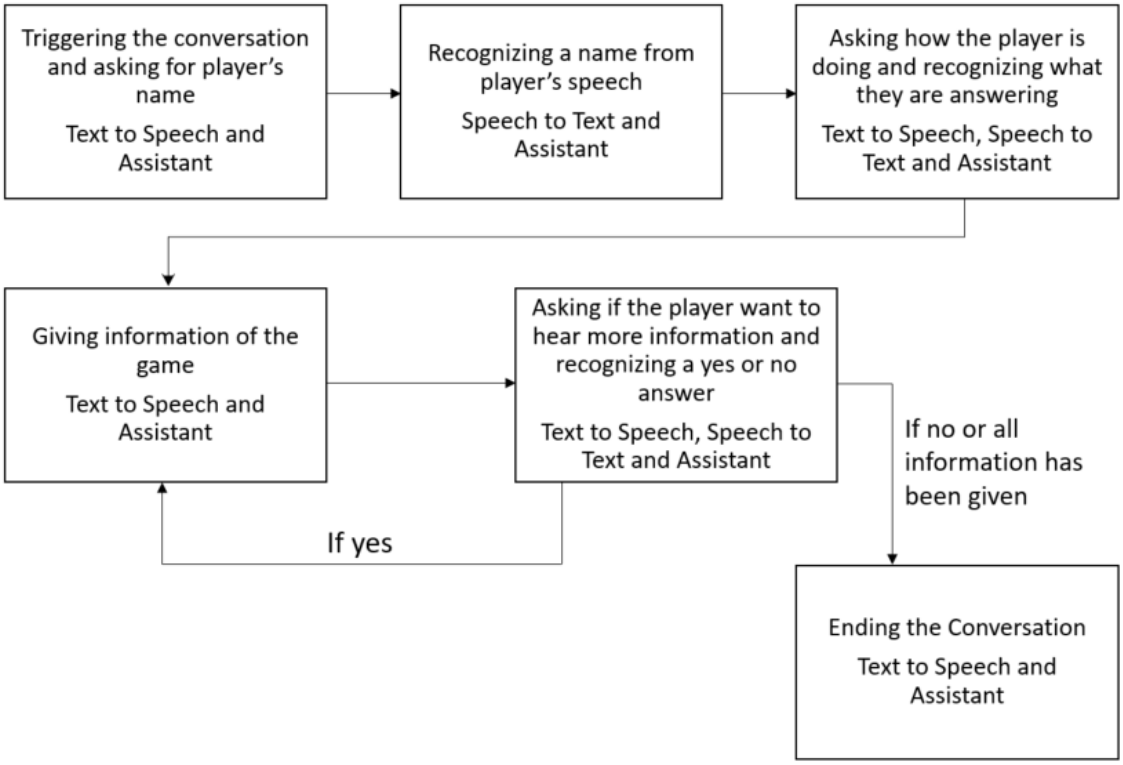
```
private void GetTTS()
{
   _textToSpeech.Voice = VoiceType.en_US_Michael;
   _textToSpeech.ToSpeech(HandleToSpeechCallback, OnTTSFail,
   TTS_content, true);
}
```

Like in the earlier parts the text that needs to be synthesized is passed to the Text to Speech's own script for synthesis along with the callback functions for successful and unsuccessful synthesis attempts, as seen in the function shown above in the code snippet. In a successful synthesis an audio clip is passed back to the manager where it

is played in the game. Once the clip has been played the manager script continues to listen to the microphone for input, repeating the process described in these parts.

5.4 Building the introduction phone call

The flow of the phone call, shown in picture 4**,** is fairly straightforward in the first version of the game. It only branches when giving information about the game to the player depending on how much information they want to hear.



Picture 4. Flow of the introduction phone call

The interaction starts with the game sending a message to Assistant triggering the first node of the dialog. In this case this message is not natural language but a string of words working as a command, such as “StartConversation” and “TellAboutGame”. For the mechanics suitable for this project it was fitting to use these so-called commands in order to easily trigger certain dialog nodes from code rather than from players input. These commands are sent to the manager script from the phone call’s own manager script. In the Assistant’s workspace these commands are stored in one entity to keep track on them.

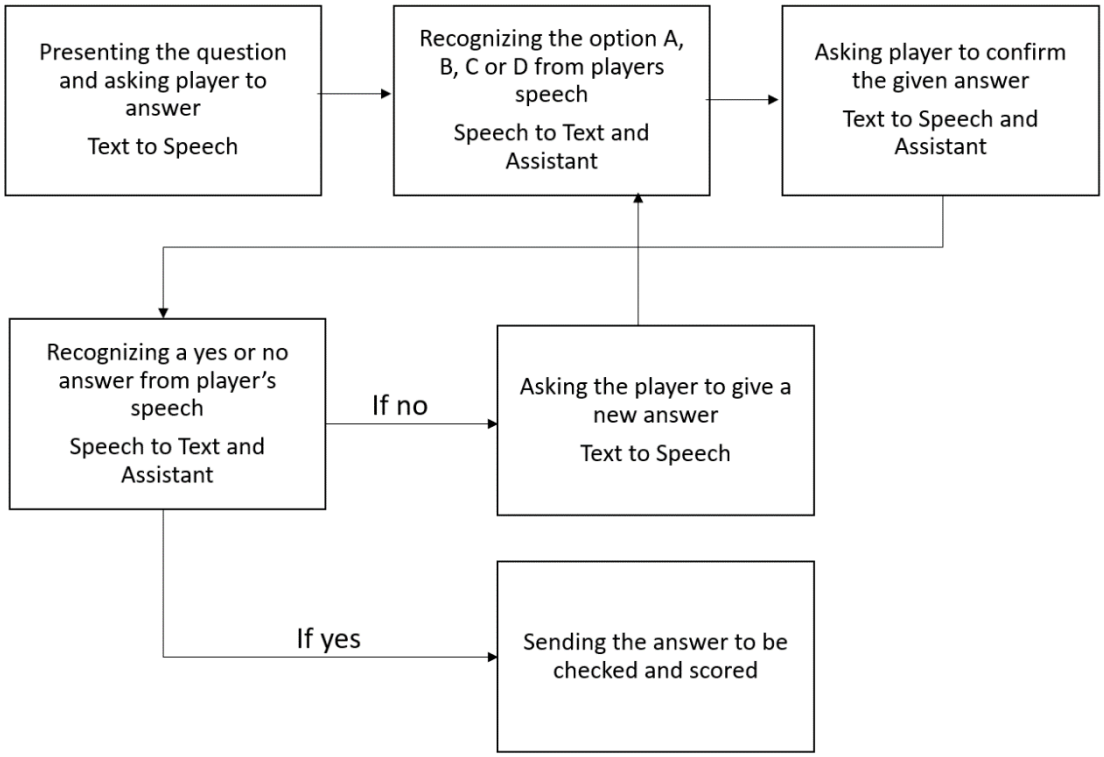
For recognizing the player's name, a prefilled entity group of names is enabled in the Assistant, so it is not necessary to create a new entity group of thousands of possible names. Once the name is recognized form player's speech and its been saved on Assistant's side, Assistant moves to the next node and uses that name by repeating it back to the player by synthesizing the message from that node with Text to Speech, saying "Hello {name}, how are you doing?". Currently it is not necessary to save the name on Unity's side since it is not used later in the game, but this could easily be done by assigning it to a string variable from the Assistant's message's context.

The question about player's mood is merely a filler to enrich the conversation and test and showcase what can be done with Watson. Once the Assistant recognizes an answer meaning either a good or a bad mood it briefly acknowledges it and moves on to giving information about the game. Similarly to earlier mechanics, Assistant's messages are synthesized with Text to Speech and every time the player is asked if they want to know more their answer is transcribed to written text and sent to Assistant. Depending on the answer Assistant will either move on to a node that contains more information or end the conversation. Conversation is ended and the phone call related game objects are deactivated in the scene from the phone call's own manager script after the manager receives the command "QuitConv" from the Assistant. The player can now continue exploring the rest of the game.

5.5 Building the quiz stage mechanics

The quiz follows a similar pattern as other similar games. A question is asked, an answer is given, and this is repeated multiple times. The interaction does not branch out and only lets the player change the answer if necessary. This is shown in picture 5 which illustrates the flow of the questioning and answering process.

The quiz can also be played without speaking out loud the answers by using the Watson features. In this case speech recognition works as another controller and the same actions can be triggered either way in each question.



Picture 5. Flow of the quiz part

There are three main steps in asking the player a question and them answering it.

i) Watson utters the question and asks the player to give an answer.
ii) Player says the answer out loud and is asked to confirm it.
iii) The confirmed answer is checked and scored.

After uttering the question and asking the player for an answer Watson waits for the player to say something. If the answer is not recognized as a valid answer, the player is prompted to give the answer again. There is an entity defined as answer options in the quiz skill and whenever the Assistant recognizes that the player said one of those options, were it "A", "answer B" or "option C" it will trigger the node that stores the given answer and sends it back to Unity's side to be saved there as well.

The player is then asked to confirm if they really want to answer this way. The confirmation can be given as "yes", "no" or as any widely used synonym for these words. If the player wants to change the answer, they are prompted for a new answer and the Assistant's dialog flow returns to the node that is tracking the input for a possible answer option. If the player confirms the answer a confirmation command is sent to the script

handling Watson's part of the quiz stage. This command will trigger a function that will in turn send the saved answer to the script that handles the basic mechanics, like checking the answer and scoring, and finally moving onto the next question.

At this version of the game in order to avoid misunderstandings, for example if the player does not speak clearly enough and their answer is understood incorrectly, the player can choose the answer with a controller if using voice proves to be too difficult, which might happen with players whose native language is not English.

5.6 Possible advancements

Increasing the level of immersion could be achieved by giving players the opportunity to answer the questions by saying the answer option out loud rather than the letter representing it, but considering the simplicity of checking four different answer options in total it seemed too unnecessary of a task to do. For the first version of the game this approach was sufficient, especially when the accuracy of the recognized words was satisfactory, and the gameplay felt natural with giving the answer as a single letter.

Asking the player for their name during the phone call is a feature that might present problems once the game gets played by more people. While the Assistant uses the prefilled entity list for names and can handle many possible names, it is bound to have challenges with certain names, for example from the Finnish target group. In order to avoid situations where the player gets stuck with trying to get the game to understand their name, this part should either be removed completely or skipped after a couple of unsuccessful tries.

It is possible to expand the pool of words an application is able to recognize. In the code snippet below, from the Unity SDK example scripts, it is shown how a word is added to a collection of custom words, in order to be understood in a suitable way when transcribing with Speech to Text. For example, if there were a situation where the name "Mikey" is not being transcribed correctly or it is recognized as "mikey", the name could be added as a custom word to fix this issue. The word "mikey" is given words that sound similar to it if necessary, but most importantly it is given the string "Mikey" as a word it should be displayed as. Now the service would present the word correctly as "Mikey" whenever it recognizes other words that have been set in this example.

```
Words words = new Words();
Word w0 = new Word();
List<Word> wordList = new List<Word>();
w0.word = "mikey";
w0.sounds_like = new string[1];
w0.sounds_like[0] = "my key";
w0.display_as = "Mikey";
wordList.Add(w0);
words.words = wordList.ToArray();

_service.AddCustomWords(HandleAddCustomWordsFromObject, OnFail,
_createdCustomizationID, words);
```

Asking the player for their name is harder with speech recognition than it would be with written text. More uncommon names can be recognized as names when they are written with a capital letter, but with speech recognition this is not always possible, so only some names are usually transcribed correctly, which was sometimes the case while working with this part.

Considering the greatest benefits for projects like this come from the Assistant, experiments were made in order to see if the quiz part could possibly be implemented by relying more on the Assistant. This would mean writing less code in the project and building more mechanics in the Assistant's workspace.

After some planning on how to execute this it was clear that recreating this exact quiz in this manner would present some problems, although it was possible to mimic it to a certain extent by only using the Assistant. By only tracking "yes", "no" and the answer option letters on the Assistant's side the dialog and entity views stay much clearer, adding the confirmation and questions there would increase the number of nodes and collections of entities significantly. Additionally, handling the questions and their answers on Unity's side also makes it easier to update and randomize them.

The biggest challenge would be tracking and saving the score. Doing this with Assistant would be very difficult, but this was expected since this service has not been built for this purpose. Even though Assistant can store variables to be used later, having a value that is not presented in the conversation and only needs to be changed when the answer is correct, is just not possible. Going through the questions disregarding scoring would make the task easier but would not fulfil the requirements for these experiments. Additionally, receiving the input from the player already requires the process to return to

the game's mechanics, so no matter how much of the quiz is built in Assistant, the exchanges between IBM's servers and the game would stay the same as they are now.

Even though Assistant is not the perfect fit for handling all or most of the mechanics in this specific quiz, different types of features can still be created utilizing Assistant in a larger scale, like in the phone call part in the first stage. Assistant was also a great help for using the Text to Speech and Speech to Text services accurately and efficiently. For example, if the quiz should be created without Assistant, checking the player's input validity would require a lot of additional mechanics and workhours.

6 RESULTS AND CONCLUSIONS

Developing the Watson features on A Day To Remember and getting to know how some of the IBM Watson services work was overall straightforward. Getting started with the example codes from the Unity SDK and IBM's own tutorials was effortless and provided enough support for the development process. Naturally some issues arose when learning a new technology, but they were relatively easy to solve with the help of either documentation and tutorials or trial and error. What contributed to the simplicity of the development was that the game did not require the use of the more complicated features of the services, such as adding custom words for the Speech to Text service to understand because the game used very simple vocabulary not specific to any area.

If there were a project of a similar scope that required the use of the more complex features, it could prove to be discouraging for a beginner developer, but not impossible. At the time of the development of A Day To Remember, the Watson services have not been accessible to the developers for too many years and the Unity SDK has only been out for a couple of years. A lot of the documentation, tutorials and other materials are limited to official publications from IBM, basic level input from independent developers and developer communities like the Slack workspace dedicated for Watson services. Comments and explanations for the mechanics in the examples were scarce, but luckily the codes handling the services were adequately commented, helping with using the SDK and understanding the processes behind them.

As always, improvements could be made. Based on the testing and learning that was done on separate practice projects, as well as the tutorials and examples for Assistant, it was clear that it could easily be used for letting the players ask the game some question. Ultimately adding this feature to the game was cut off because the area of subjects the player could ask about would have to be limited in a way the game would still be engaging, yet simple and straightforward enough. Many of the potential players from the target group most likely do not have a lot of experience with games, let alone with virtual reality or speech recognition, so the game cannot be too complicated. After the game has been played by the target group, depending on the success and feedback, this feature could be added later as a part of the phone call or as a new feature entirely. It could also inspire a different game that could put this technology to a good use as well.

If one would plan to create a game with rich dialog using natural language, some third-party tools become essential. Creating the mechanics for that from scratch is not very realistic nor reasonable for many game developers considering the workload. With Watson and other similar technologies, it is possible to diversify the conversation the players can have with the game. The dialog does not need to be limited to a couple of options of what the player can say to the game, following a certain pattern every time, which is the case in many games. In a paper on the game Bot Colony, Eugene Joseph points out that in a game containing dialog, understanding the semantics of the language and reacting to it accordingly adds a lot more immersion if the dialog options are not predefined and the player does not have to utter the exact line in order to engage in the conversation. (Joseph, 2019) While giving the player the opportunities to talk as freely and as often as in Bot Colony using Assistant is an insuperable task, it can certainly give inspiration for similar games.

When a dialog is very unrestricted and the player can talk freely about what they want to, the flow of the conversation might get led off-topic. With Assistant wandering to other nodes can be adjusted to fit the needs of the conversation with its digression feature. In case of digression, if the current topic is important and needs to be handled, Assistant will return to it after the player has gone astray. If the topic is not important, the conversation can be adjusted so that the flow does not return to it and the user can continue the interaction the way they want to. (IBM 2018.)

Starting to work with Watson was affordable since all the services offered the free versions, which in this project were sufficient. Only on one occasion the Text to Speech service characters ran out, apart from that the services had either plenty of use left or were just about to run out at the end of the month. This of course is connected to the fact of how much work was being done and was the game showcased or tested during the month. Depending on how much a game's Watson features would be used and tested during development, how widely and frequently the finished product would be used and how much would it utilize the technologies, the cost of the services would vary. In a small-scale project like A Day To Remember where the Watson features were limited to two instances per playthrough, the monthly costs would have stayed in tens of euros if the development would have required upgrading to the paid versions.

At the time of this thesis the estimate calculations for these services on IBM's sites gave a total of 20,80 euros for the use provided in the free versions. Assistant is the most expensive since both Text to Speech and Speech to Text count up to two euros

combined, but it is the most plentiful service out of the three. For example, in the free version of Assistant the game can make 10 000 calls to the server per month. In the phone call part, if the conversation flows without a problem and the player wants to hear all the information available, total of six calls are made, meaning that the conversation could be played through over 1600 times during a month. The other two services are not as lasting but are a lot more affordable.

If a requirement for a game is to have multiple conversations with a bot similar to the phone call, using the free version might be too restrictive. Assistant's free version can only contain five dialog skills and each skill can hold 100 dialog nodes, so depending on how long or versatile conversations are needed the Assistant might run out of space (IBM 2018). Implementing more than five conversations are possible with one IBM Cloud account and the free version since one dialog node can easily hold multiple conversations if planned carefully.

Based on the work done for this thesis, Assistant is a good solution for creating dialog using natural language, if it is the right choice for the purpose. Its ability to recognize variables from the input offers opportunities to control games and enter information to them in a new way, with or without speech recognition. With speech recognition or written natural language alone this would require a lot of restrictions on how the player can communicate to the game, as well as more mechanics on checking the input's validity. Although Assistant can be a big help in creating dialog using NLU, if one is planning to develop a big game with extensive, long conversations, the costs of using the service needs to be considered, as well as keeping in mind that Assistant is mainly aimed at customer service purposes for businesses. Maybe the most important aspect is that even though the dialog skills can be adapted to some degree, if a game's dialog would need some specific feature not supported in Assistant, implementing that would fall on to the developers to create. Depending on the requirements for a game's dialog, each team has to make the decision by themselves whether or not Assistant is the right tool for them. For less ambitious plans, like giving commands with natural language, Assistant can be a great help.

Both Text to Speech and Speech to Text services are as good as any other solution for implementing speech synthesis and speech recognition to games. Implementing them to a Unity project and using them was easy and the outcomes were useful and accurate, apart from some words occasionally being recognized incorrectly. Even the possible

costs for using them would not easily rise to unbearable amounts preventing developers from using them.

Without the Watson services it would have not been possible to implement neither of the features developed for this thesis. While speech recognition could have been achieved with other tools, understanding the player's input would have required a lot more work and not letting the player speak as freely as they can now. It is crucial that the input is understood correctly in order to make the game playable. This is the situation in any application or game that uses speech recognition, even the slightest inaccuracies can make the user frustrated and not use the product again.

The biggest problems during the development were caused by issues on IBM's side. During the development of this game there were two instances when using Watson was put on hold for this reason. First, certain Watson servers were down for a couple of days and accessing the necessary services was not possible. On another occasion an error occurred in Speech to Text service, but a fix was shortly found and delivered to users by the Watson developers. Although this fix allowed the development to continue, it did not restore already built and shipped applications using this service. Considering these were the only occasions when problems were not solvable by the developer during the six months when the development happened, it is safe to say Watson is providing a reliable technology to implement speech recognition and other features powered with AI to games and applications.

6.1 Other approaches and similar technologies

The Watson services are not the only way to create the features implemented in this thesis. Numerous other companies and even open source softwares offer similar services and technologies. The most similar ones to Watson are the services by Google on Google Cloud. Google offers services for synthesizing text into speech, speech recognition and Dialogflow, a tool to create chatbots and other applications using AI (Google 2019). Dialogflow uses machine learning to recognize what the user is saying using intents and entities and is capable of having humanlike conversations with the users, just like Watson's Assistant. At the time of the writing of this thesis the pricing for Google Cloud is similar to Watson's, having a free but limited standard edition, as well as enterprise editions which offer unlimited usage. (Google 2019)

For implementing all the technologies covered in this thesis, transcribing speech to written text, speech synthesis and a tool for building chatbots, with only using one service provider or software that is easily accessible to all developers, IBM and Google are the only options. On top of the mentioned services, both offer other AI powered services that could be useful for some projects, such as visual recognition.

If only one or two of these services are needed there are numerous other technologies and tools for implementing them. For example, for speech synthesis and speech recognition a company called iSpeech offers their own API's for implementing these services, along with documentation and SDKs for various platforms (iSpeech 2019). Many other developers offer SDKs for implementing their technologies on different platforms, but many do not support game engines like Unity straight away. Some can be used with tools made by other developers that offer a quick access to the technologies. This type of tools can be found on Unity's Asset Store, where many utilize Google's speech recognition services.

Downloading and signing up for third-party technologies is not always necessary. When using Windows, it is possible to employ speech recognition and even speech synthesis used in the operating system. Implementing speech recognition does not require multiple additional steps since its use is supported in Unity. Just by accessing the namespace for it the different ways to use this technology can be utilized in a game project. It offers different ways of handling the voice input as well, for example, it can listen to the microphone input and try to recognize keywords in it with a keyword recognizer. (Unity 2019)

Choosing the approach for creating dialog or chatbots depends on how complex they need to be. The simplest dialogs can be programmed from the start by the developer, but as the size grows and the possible need for more sophisticated solutions arises, third party software becomes essential. If the project requires the chatbot to recognize intents, entities and context from natural language input, options become scarce. Besides Watson's Assistant and Google's Dialogflow, a lot of the technologies for building a chatbot, for example Zendesk's Answer Bot (Zendesk 2019), are aimed at automated customer services and marketed towards businesses with clients reaching out to them via chats or other messaging channels on their websites. For the most common type of dialog in games, which is hard coded interactions in a branching story, there are both free and paid solutions for achieving it. Unity's Asset Store alone has numerous different

options for building and managing dialog between the player and game, and even creating text-based games without coding skills, such as Fungus (Snozbot 2019).

6.2 Localization for target group

The initial target group for A Day To Remember was Finnish speaking players, but since Watson did not support Finnish language at the time of this project, English was the natural choice.

The selection of available services for synthesizing and recognizing speech in Finnish is limited but offers enough alternatives for implementing them to applications. For using both in Finnish while only utilizing products provided by one developer, the options are the Google Cloud services and the speech technologies by a Finnish company Lingsoft. Among other products, Lingsoft has developed technologies for recognizing Finnish language with machine learning and deep neural networks, as well as synthesizing text into Finnish speech that sounds natural. (Lingsoft 2019) Keeping in mind that one of the objectives of this thesis was to partly concentrate on low budget games, Lingsoft's services are more suitable for projects of a larger scale rather than small indie games, prototypes and hobby projects. Therefore, for many game developers using the services by Google is the easier solution for implementing speech synthesis and recognition in Finnish.

For using a service like the Assistant, there are no alternatives for using it in Finnish right now, or at least any that are easily accessible. If using a service of this type in Finnish is absolutely necessary, a workaround for this could be using a language translation service and translating the input to English before sending it to the service. For many other languages localization is a lot easier, even when the applications need to be used in multiple languages. For example, Watson Assistant supports 13 and Google's Dialogflow 20 languages and dialects, although it must be noted that not all of the parts used in these services support each language. (IBM 2019, Google 2019.)

7 SUMMARY

The goals for this thesis were to utilize the services of IBM Watson in a game project, as well as to learn about Watson and AI with speech recognition in games in general. It is obvious that AI is widely used in games, but using it to recognize speech, let alone to comprehend natural language, is rare. In order to get an understanding of the IBM Watson services and to find out if they could be put to good use in projects similar to A Day To Remember, they were used to develop two features to the game. Both features required the player to conversate with the game by simply talking to it, and the aim was to have these conversations be as unrestricted as possible.

After implementing the features covered in this thesis it was apparent that IBM Watson offers useful tools for game developers. While there were some difficulties with the programming tasks, as there are with any game project, resolving those problems was never impossible. Especially now after gaining experience on these services and using them in Unity it is safe to assume that continuing working with them would be more straightforward and would offer possibilities to create more eloquent games and applications. Additionally, the research on games and applications that use speech recognition and can understand natural language gave inspiration for utilizing these technologies in different ways in other projects. Having many long conversations with a game using natural language might not be the easiest task, but implementing short commands with Watson, especially Assistant, is very effective.

In order to ensure the best results with projects of the same scale as A Day To Remember, where the developers are beginners and do not have knowledge about Watson or similar services, the game design should center around those technologies if possible. This way the project can benefit from their advantages more and implementing them will be easier. After gaining basic skills on these technologies it will be easier to implement them in more complicated projects, as one feature among many others.

As stated earlier in this chapter, speech recognition and natural language understanding are still very uncommon in games but considering that the technologies needed to utilize them are fairly new, it is understandable. Especially natural language processing technologies are not widely available to developers, other than through IBM, Google and other companies specializing in this technology. It is very likely that the popularity of speech technologies will rise in the future. According to MarketsandMarkets, the market

of speech and voice recognition will grow by almost 20% in the next few years. This growth will be result of the development of the technologies and the growing demand for them. (MarketsandMarkets, 2019)

Additionally, people becoming more accustomed to using voice controls, more languages being supported with higher accuracy and maybe even big game studios utilizing speech technologies in their products at some point might contribute to the rise of games benefiting from speech recognition and understanding natural language. In any case, familiarizing oneself with these technologies and their use in games could be worthwhile for a game developer. For this task IBM Watson offers viable tools to start experimenting.

REFERENCES

Amazon, 2019. Alexa Skills Kit. Accessed 10.3.2019 developer.amazon.com/alexa-skills-kit

Apple, 2018. September Event 2018 — Apple. Accessed 11.3.2019 www.youtube.com/watch?v=wFTmQ27S7OQ

BioWare, 2012. Better With Kinect. Accessed 27.2.2019 masseffect.bioware.com/about/kinect/

North Side Inc., 2013. Bot Colony

Bourg, M. D. & Seemann, G. 2004. AI for Game Developers. United States: O'Reilly Media. pages 2-3

Brookshear G. J. 2007. Computer Science: An Overview. 9th edition, United States: Pearson/Addison Wesley. Available infocat.ucpel.tche.br/disc/icc/docs/CSAO.pdf 26.2.2019, pages 452-453

DARPA, 2003. Darpa awards contracts for pioneering r&d in cognitive systems Accessed 13.3.2019 www.adam.cheyer.com/pal.pdf

Donges, N. 2018. Introduction to NLP. towardsdatascience.com/introduction-to-nlp-5bff2b2a7170 Accessed 23.5.2019

Ferrucci, D.; Brown, E.; Chu-Carroll, J.; Fan, J.; Gondek, D.; Kalyanpur, A. A.; Lally, A; Murdock, J. M.; Nyberg, E.; Prager, J.; Schlaefer, N.; & Welty, C. 2010. Building Watson: An Overview of the DeepQA Project. Accessed 17.2.2019 researcher.watson.ibm.com/researcher/files/us-mike.barborak/WatsonInAIMagReprint.pdf

Geitgey, A., 2016. Machine Learning is Fun Part 6: How to do Speech Recognition with Deep Learning. Accessed 4.5.2019 medium.com/@ageitgey/machine-learning-is-fun-part-6-how-to-do-speech-recognition-with-deep-learning-28293c162f7a

Google Developers, 2018. Keynote (Google I/O '18). Accessed 5.5.2019 www.youtube.com/watch?v=ogfYd705cRs

Googe, 2019. Cloud AI Building blocks. Accessed 6.4.2019 cloud.google.com/products/ai/building-blocks/

Google, 2019. Dialogflow. Accessed 6.4.2019 dialogflow.com/

Google, 2019. Dialog Enterprise Edition / Documentation / Languages. Accessed 13.4.2019 cloud.google.com/dialogflow-enterprise/docs/reference/language

Goyal, M. 2015. Announcing our largest release of Watson Developer Cloud services. Accessed 17.4.2019 www.ibm.com/blogs/bluemix/2015/09/announcing-watson-developer-cloud-services-release/

High, R. 2012. The Era of Cognitive Systems: An Inside Look at IBM Watson and How it Works. IBM Redbooks. Available redbooks.ibm.com/abstracts/redp4955.html? P. 1

IBM, 2011. IBM Watson: The Science Behind an Answer. Accessed 20.4.2019 www.youtube.com/watch?time_continue=13&v=DywO4zksfXw

IBM, 2018. IBM Cloud Docs / Watson Assistant. Accessed 22.12.2018 console.bluemix.net/docs/services/assistant

IBM, 2018. IBM Cloud Docs / Text to Speech. Accessed 22.12.2018 console.bluemix.net/docs/services/text-to-speech

IBM, 2018. IBM Cloud Docs / Speech To Text. Accessed 22.12.2018 console.bluemix.net/docs/services/speech-to-text

IBM, 2019. A Computer Called Watson. Accessed 15.1.2019 www.ibm.com/ibm/history/ibm100/us/en/icons/watson/

IBM, 2019. The DeepQA Research Team. Accessed 14.1.2019 researcher.watson.ibm.com/researcher/view_group.php?id=2099

IBM, 2018. Watson Assistant v2. Accessed 29.12.2018 console.bluemix.net/apidocs/assistant-v2

IBM, 2019. Watson Services. Accessed 16.4.2019 console.bluemix.net/developer/watson/services

iSpeech, 2019. Powerful Text to Speech API. Accessed 9.4.2019 /www.ispeech.org/developers

Joseph, E. 2019. Bot Colony – a Video Game Featuring Intelligent Language-Based Interaction with the Characters. Accessed 9.3.2019 www.botcolony.com/doc/BotColony_paper.pdf P. 1, 4-10

Lingsoft, 2019. Käyttöliittymänä puhe. Accessed 13.4.2019 lingsoft.fi/lingsoft-lab/puheteknologia

MarketsandMarkets, 2019. Speech and Voice Recognition Market worth $21.5 billion by 2024. Accessed 28.4.2019 www.marketsandmarkets.com/PressReleases/speech-voice-recognition.asp

Microsoft, 2019. Original voice commands on Xbox One. Accessed 13.1.2019 support.xbox.com/en-US/xbox-one/accessories/voice-commands

North Side Inc., 2013. Bot Colony

Perfect Tap Games, 2017. Chicken Scream.

Seif, G. 2018. An easy introduction to Natural Language Processing. Accessed 4.5.2019 towardsdatascience.com/an-easy-introduction-to-natural-language-processing-b1e2801291c1

Snozbot, 2019. Accessed 7.4.2019 fungusgames.com

SRI, 2019. Artificial Intelligence: CALO. Accessed 13.3.2019 www.sri.com/work/timeline-innovation/timeline.php?timeline=computing-digital#!&innovation=artificial-intelligence-calo

Steinberg, R. S.; Parmar, P. & Richard, B. 2006. Contemporary Youth Culture: An International Encyclopedia Volume 1. United States: Greenwood Press. P. 178 – 179

Swiezinski, L. 2013 Lifecycle of a Jeopardy Question Answered by Watson DeepQA. Accessed 28.5.2019 Available www.ias.informatik.tu-darmstadt.de/uploads/Teaching/AutonomousLearningSystems/Swiezinski_ALS_2013.pdf

Ubisoft, 2017. Star Trek: Bridge Crew puts you and your friends in the heart of a starship. Accessed 29.1.2019 www.ubisoft.com/en-us/game/star-trek-bridge-crew/

Ubisoft, 2018. How Start Trek: Bridge Crew uses AI to crew your ship on solo missions. Accessed 29.1.2019 news.ubisoft.com/en-us/article/331961/How-Star-Trek-Bridge-Crew-Uses-AI-to-Crew-Your-Ship-on-Solo-Missions

Unity, 2019. KeywordRecognizer. Accessed 20.5.2019 docs.Unity.com/ScriptReference/Windows.Speech.KeywordRecognizer.html

Van der Velde, N. 2018. Voice Controlled Games: The Rise of Speech Technology in Gaming. Accessed 6.3.2019 www.globalme.net/blog/voice-controlled-games

Van der Velde, N. 2018. Speech Recognition Technology Overview. Accessed 6.5.2019 www.globalme.net/blog/the-present-future-of-speech-recognition

Warner Bros, 2016. The Wayne Investigation

Weizenbaum, J. 1966. ELIZA--A Computer Program For the Study of Natural Language Communication Between Man and Machine. Communications of the ACM Volume 9, Number 1. Available www.universelle-automation.de/1966_Boston.pdf

Yannakakis, G. N. & Togelius, J. 2018. Artificial Intelligence and Games. Switzerland: Springer. Available gameaibook.org/book.pdf P. 8, 15-25

Xuedong, H. & Li, D. 2010. An Overview of Modern Speech Recognition. From Handbook of Natural Language Processing, Chapman & Hall/CRC, second edition. Available www.microsoft.com/en-us/research/wp-content/uploads/2016/02/Book-Chap-HuangDeng2010.pdf P. 351, 359

Zendesk, 2019. Answer Bot. Accessed 7.4.2019 www.zendesk.com/answer-bot/

Zhongzhi, S. 2011. Advanced Artificial Intelligence. Singapore: World Scientific Publishing Co. Pte. Ltd. P. 18-19