Mixed Interactive/Procedural Content Creation in Immersive Virtual Reality Environments

João Filipe Magalhães Moreira

Dissertation

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rui Pedro Amaral Rodrigues
Co-Supervisor: Rui Pedro da Silva Nóbrega

July 27, 2018
Mixed Interactive/Procedural Content Creation in Immersive Virtual Reality Environments

João Filipe Magalhães Moreira

Mestrado Integrado em Engenharia Informática e Computação

July 27, 2018
Acknowledgments

Thank you to my mother, Gabriela Magalhães, father, Carlos Manuel and brothers Carlos & Nuno for all the support in this long path of 5 years, and all the years before.
Thank you to my coordinator Rui Rodrigues for all the guidance. Thank you to my co-coordinator Rui Nóbrega for all the patience with all my questions.
Thank you, in no particular order, to Diana Pinto, Diogo Amaral and Vasco Ribeiro as my thesis buddies. A thank you to, in no particular order, Vitor, Inês, Gil, Mota, Marisa, Pontes for all the company in the long nights working.
Thank you to everyone at GIG for they help, support and advice.
Thank you to all the teachers of the teachers of the Faculdade de Engenharia da Universidade do Porto for all that they have taught me along the this path.
Thank you to everyone who took time of their day to either give an opinion, advice and test my thesis.
Abstract

With the growing of VR it is only natural that ways to help and simplify the work of the content creators would appear. In the ideal VR editor, the user is inside the scene building, painting and modeling the surrounding area. However, the main advantage of immersion is also its main drawback: inside the virtual world it is hard to use input devices (e.g., keyboard). The goal is to create content using different interaction paradigms (e.g., voice commands) and obtain assets from several automatic sources such as: available web assets, photogrammetry and procedural content generation. Exploring a wide range of asset sources, procedural generation and AI-supported content creators will enable a rich VR content creation tool. This work will explore the combination of multimodal inputs such as gestures and voice in an immersive environment, together with pre-existing model libraries, procedural generators and AI-support to create immersive content.

The main environment is based in Unity 3D, works on Oculus Rift headset, and uses a combination of external services. These were analyzed during the state of the art research, to assess their adequacy and which one was selected to be used in the prototype. To summarize the main objective is to explore how the combination of multimodal inputs such as gestures and voice in an immersive environment, together with pre-existing model libraries, procedural generators and AI-support could support creation of 3D/VR content in an immersive way and in such a manner that enables the non technical users to do it.

Keywords: VR, VR Editor, HTC Vive, Unity3D, Procedural Generation, Voice Input
Resumo

Com o crescimento da RV é natural que começem a aparecer maneiras de ajudar e simplificar o processo e o trabalho necessário para a criação de conteúdo. No editor ideal de VR, o utilizador estaria dentro da cena que ele estaria a construir; pintar e modelar as áreas que rodeiam a cena. A principal vantagem deste nível de imersão é também a sua maior desvantagem: dentro do mundo virtual torna-se difícil utilizar os periféricos, como o teclado. O principal objetivo é criar conteúdo através da interação de diferentes paradigmas (comandos de voz) e obter conteúdo através de diferentes fontes como por exemplo: web assets, phtogrammetry e geração procedimental de conteúdo. Através da exploração de variado leque de fonte de conteúdo, a geração procedimental e criação de conteúdo suportada por IA irão permitir aos criadores ter uma rica e poderá ferramenta de criação de RV. Este trabalho irá explorar a combinação de inputs multimodais como gestos, comandos de voz num ambiente imersivo, onde se irá combinar com bibliotecas, pré-existentes, geração procedimental e criação suportada por IA para criar conteúdo 3D e de RV. O principal ambiente de desenvolvimento será baseado em Unity 3D e no headset de HTC Vive, onde será utilizada uma combinação de ferramentas externas. Estas ferramentas foram analisadas durante a pesquisa para o Estado da Arte, com o objetivo de descobrir quando adequadas seriam para o protótipo. Sendo que o principal objetivo deste projeto é explorar a combinação de inputs multimodais como gestos e voz num ambiente imersivo, juntamente com o conteúdo pré-existente, conteúdo gerado através de IA e geração procedimental.
“The art of simplicity is a puzzle of complexity.”

Douglas Horton
Contents

1 Introduction 1
  1.1 Motivation and Goals ............................................. 1
  1.2 Research Questions .............................................. 2
  1.3 Objectives ......................................................... 3
  1.4 Evaluation of Results .............................................. 3
  1.5 Thesis Structure .................................................. 3

2 State of the Art 5
  2.1 Natural Speech Recognition ......................................... 5
  2.2 VR .......................................................... 10
    2.2.1 VR Editors .................................................. 14
  2.3 Procedural & AI Generation ....................................... 16
  2.4 Combining Inputs ................................................ 18
    2.4.1 Key differences .............................................. 18
  2.5 Summary ........................................................ 19

3 Architecture 20
  3.1 Context .......................................................... 20
  3.2 Requirements ..................................................... 21
  3.3 Functionalities ................................................... 22
  3.4 Concepts ........................................................ 24
  3.5 Proposed Solution ................................................ 25
    3.5.1 Package Diagram ............................................ 27
    3.5.2 Concept Blocks .............................................. 28

4 Implementation 29
  4.1 Speech to Text .................................................... 30
  4.2 Process Text .................................................... 35
  4.3 Structure of a Request .......................................... 37
  4.4 The creation of Objects ......................................... 40
  4.5 Object Selection ................................................ 43
  4.6 Editing Actions .................................................. 44
  4.7 Final Application ................................................ 53
  4.8 Summary ........................................................ 56

5 Evaluation 57
  5.1 User Studies ..................................................... 57
  5.2 Methodology ..................................................... 58
    5.2.1 Voice Testing ................................................. 58
List of Figures

1.1 Gartner hype cycle 2017 [16] ................................. 2
2.1 Google Milestone .................................................. 9
2.2 The Sword of Damocles .......................................... 11
2.3 Aspen Movie Map .................................................. 12
2.4 Virtuality ............................................................ 12
2.5 Google Cardboard .................................................. 13
2.6 Virtual Reality Headsets ......................................... 13
2.7 Unity 3D Editor model to Sketchfab ......................... 15
3.1 External Assets .................................................... 22
3.2 Pointer .............................................................. 25
3.3 Proposed Workflow ............................................... 26
3.4 Package Diagram .................................................. 27
3.5 Block Diagram ...................................................... 28
4.1 Overall System Diagram ......................................... 29
4.2 System Diagram .................................................... 30
4.3 Wit Ai Request Header ............................................ 32
4.4 Text Processing Sequence Diagram ......................... 36
4.5 Object .............................................................. 38
4.6 Colour of Objects .................................................. 38
4.7 Number of Objects ............................................... 38
4.8 Circle Layout ....................................................... 39
4.9 Square Layout ..................................................... 40
4.10 Grid Layout ....................................................... 40
4.11 Unity Primitives ................................................... 41
4.12 Object Selection .................................................. 44
4.13 Create Action ...................................................... 44
4.14 Delete Action ....................................................... 45
4.15 Duplicate Action .................................................. 46
4.16 Group Action ....................................................... 46
4.17 Rotate Action ....................................................... 47
4.18 Move Action ....................................................... 49
4.19 Scale Action ....................................................... 49
4.20 Website, where the users can feed commands to train the bots ........................................... 53
4.21 A scene with assets from multiple sources .................. 54
4.22 VR Menu .......................................................... 55
4.23 Select .............................................................. 55
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>User studies instructions.</td>
<td>58</td>
</tr>
<tr>
<td>5.2</td>
<td>Example of Commands.</td>
<td>59</td>
</tr>
<tr>
<td>5.3</td>
<td>User study case 2</td>
<td>60</td>
</tr>
<tr>
<td>5.4</td>
<td>User study case 2</td>
<td>60</td>
</tr>
<tr>
<td>5.5</td>
<td>Age of Participants.</td>
<td>61</td>
</tr>
<tr>
<td>5.6</td>
<td>Percentage of users who think the commands are a must have.</td>
<td>63</td>
</tr>
<tr>
<td>5.7</td>
<td>Asset Reutilization.</td>
<td>64</td>
</tr>
<tr>
<td>5.8</td>
<td>App Questions</td>
<td>65</td>
</tr>
<tr>
<td>5.9</td>
<td>User Experience.</td>
<td>66</td>
</tr>
<tr>
<td>5.10</td>
<td>User Experience.</td>
<td>66</td>
</tr>
<tr>
<td>5.11</td>
<td>User experience with the commands.</td>
<td>67</td>
</tr>
<tr>
<td>5.12</td>
<td>Application Testing.</td>
<td>68</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Overview of Speech Recognition</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Siri Commands</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>Some Variables in Speech Recognition</td>
<td>8</td>
</tr>
<tr>
<td>2.4</td>
<td>VR throughout History</td>
<td>10</td>
</tr>
<tr>
<td>2.5</td>
<td>Oculus Minimum Specs.</td>
<td>13</td>
</tr>
<tr>
<td>2.6</td>
<td>HTC Vive Recommended Specs.</td>
<td>14</td>
</tr>
<tr>
<td>2.7</td>
<td>Technical specifications of playstation VR.</td>
<td>14</td>
</tr>
<tr>
<td>2.8</td>
<td>Procedural content generation approaches</td>
<td>16</td>
</tr>
<tr>
<td>2.9</td>
<td>State of Connectivity [47]</td>
<td>17</td>
</tr>
<tr>
<td>2.10</td>
<td>User Interaction with the Generator [47]</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>Requirements</td>
<td>21</td>
</tr>
<tr>
<td>3.2</td>
<td>Functionalities</td>
<td>23</td>
</tr>
<tr>
<td>3.3</td>
<td>Concepts</td>
<td>24</td>
</tr>
<tr>
<td>4.1</td>
<td>Wit Ai Characteristics [55]</td>
<td>30</td>
</tr>
<tr>
<td>4.2</td>
<td>Speech To Text Characteristics [3]</td>
<td>33</td>
</tr>
<tr>
<td>4.3</td>
<td>Detailed Information</td>
<td>34</td>
</tr>
<tr>
<td>4.4</td>
<td>Editing Actions</td>
<td>37</td>
</tr>
<tr>
<td>4.5</td>
<td>Layouts</td>
<td>39</td>
</tr>
<tr>
<td>4.6</td>
<td>Unity Primitives</td>
<td>41</td>
</tr>
<tr>
<td>4.7</td>
<td>Poly Request</td>
<td>41</td>
</tr>
<tr>
<td>4.8</td>
<td>Assets Formats</td>
<td>42</td>
</tr>
<tr>
<td>4.9</td>
<td>Sketchfab Request</td>
<td>42</td>
</tr>
<tr>
<td>5.1</td>
<td>Tests Description</td>
<td>58</td>
</tr>
<tr>
<td>5.2</td>
<td>Characterization of the Participants.</td>
<td>62</td>
</tr>
</tbody>
</table>
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR</td>
<td>Augmented Reality</td>
</tr>
<tr>
<td>DARPA</td>
<td>Defense Advanced Research Projects Agency</td>
</tr>
<tr>
<td>DOD</td>
<td>Department of Defense</td>
</tr>
<tr>
<td>HMD</td>
<td>Head-Mounted Display</td>
</tr>
<tr>
<td>SUR</td>
<td>Speech Understanding Research</td>
</tr>
<tr>
<td>PGC</td>
<td>Procedural Generation of Content</td>
</tr>
<tr>
<td>VR</td>
<td>Virtual Reality</td>
</tr>
<tr>
<td>RV</td>
<td>Realidade Virtual</td>
</tr>
<tr>
<td>XR</td>
<td>External Reality</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In 2016 Goldman Sachs issued a report on the status of the VR industry, they found that so far $3.5 billion dollars had been done by capital investments. Furthermore, by 2014 Google announced that 10 million Google Cardboard had been distributed \[17\]. When Samsung launched its VR head-set it took 48 hours to sell out at the price of $99 \[5\]. The VR market has been in constant growth for the past few years, leading to a state where innovation is striving, a place that is maturing, with still a great room for growth.

As VR grows (Figure 1.1) so does the demand for content. However, for a long time, the creation of 3D Scenes has been deterred by a myriad of obstacles from different sides. It is a highly technical job in the creation of 3D assets. Even after you have the assets you are still required to assemble the scene, to do so you will most likely require an editor, which in turn will probably have its own language or way to piece together the scene.

1.1 Motivation and Goals

One of the goals of this dissertation would be to see how far the process of creating 3D scenes can be simplified, so that non-technical professionals are able to create them on the fly. In the ideal scenario, for instance, a psychologist trying to create a virtual scenario for e.g. phobia treatment should be able to utter the command "Give me a Conference Room" or "Give me one table six chairs and place 1 chair on one side a five on the other" and the program should be able generate such a scene.

Another reason is the innovative aspect of the concept. Even though VR editors, speech processing, procedural creation are already available and well established, the combination of these elements is not. Being able to issue voice commands while editing and using the VR, will further allow the content creator to get a better picture of how the scene will be perceived by the end user.

Ideally the system would also be able to query external databases for easy asset re-utilization. This focus on asset re-utilization could possibly lead to a fall of development costs, for instance some educated guesses place the budgets, of contemporary games, such as World of Warcraft (WoW) at $20,000,000–$150,000,000 \[20\]. Furthermore, anecdotal evidence seems to imply that there is an increase in the fraction of game production costs spent on generating content, which is already pointing towards
30–40% of the budget. This means, supposedly, WoW spent in between $8,000,000-
$60,000,000 [20] just in content generation. While big projects such as WoW may
not directly benefit from this asset re-utilization it could certainly lead to decrease in
costs for smaller games. Following through the cost reduction will lead to a reduced
cost in development, which in turn might make VR content more affordable to create,
buy and may even lead to an increase in the content creation.

The chance to make it even easier to follow the adage "Do not reinvent the wheel"
through the re-utilization of assets and content creators will not have to spend time
creating already done work.

Figure 1.1: Gartner hype cycle 2017 [16]

1.2 Research Questions

This section is going to go through the research questions and what they entail to
and how they guide this dissertation.

Can a multimodal combination of inputs enable non-technical users to cre-
ate tailored 3D scenes?

This question is understand if by using multi-modals inputs, like speech commands
combined with touch commands would allow for a big enough simplification so that
non-technical users are able to edit and create 3D content. Furthermore, it would also
be evaluated if a more immersive user is a more productive and efficient user, will a
user who finds himself/herself more "in touch" with a scene do a faster and better job.

Can the re-utilization of assets help speed up development time and im-
prove quality of a scene?

Will the usage of external asset database have an impact in development time, is
it possible that by using already done assets to improve the overall quality of scene,
how much time and money the users are able to save by re-using this assets, these questions focus on how a scene can be improved and reduce development time, instead of having in mind if the process could be simplified for non-technical users. It is important to denote that the financial impact could prove hard to measure.

This two questions and the sub-questions, that possibly will originate from them, will be the main focus of this dissertation.

1.3 Objectives

One of the main objective is to asses if through the usage of multiple input paradigms it will be possible to simplify the process of creating content, to such an extent, that even non-technical users are able to perform it with relative ease. Furthermore to aid in this quest, procedural generation will be used to allow the user to place objects according to certain layouts.

The application should also provide, to the user, feedback of its actions, as well as clues to what happened or what could have gone wrong, so that the user is able to have a better grasp of the outcome. With this we hope the user will be able to better use and understand the system.

1.4 Evaluation of Results

To evaluate our answers to the a group of participants will be asked to perform 2 Study Cases. The cases will be followed by some questions, that will attempt, to evaluate the user’s thoughts about the prototype and it’s usability. Certain metrics, besides the questionnaire will also be used like time taken and observations made by the facilitator.

1.5 Thesis Structure

This works contains 5 more chapters:

State of The art This chapter will cover the different takes and solutions that were already done in the different fields of the applications. For instance, what kind of applications use multi-modal inputs and how. The various engines used for 3D, VR content creation, how VR itself has evolved. Finally the different ways procedural generation has been used.

Architecture The different requirements and behaviors are explained, which will lead to high level architecture solution being proposed.

Implementation This is chapter, where the details of how the application was implemented are given.

Results This chapter will cover how the application is going to be evaluated, the results of said evaluation. As well as an attempt to understand and discuss the results and the usability of the application.
Introduction

Conclusions  This chapter besides the authors conclusions will also include the future work, that could be done regarding the application.
Chapter 2

State of the Art

This chapter is concerned with what has been done in the different fields related with the topics of this dissertation. It will cover some of VR editors that are available, what are their similarities and differences. The different Speech recognition is being used, how it can and was combined with other inputs. Finally, it will end with a look at Procedural Generation of Content.

2.1 Natural Speech Recognition

Speech Recognition, Table 2.1 is an inter-disciplinary field, that is part of computational linguistics. Speech Recognition concerns itself with devolving technologies and methodologies that can recognize speech and transcribe into text through the usage of computers. It is also known as "computer speech recognition" (CSS), "automatic speech recognition" (ASR) or just "speech to text" (STT).

Computational linguistics are also an inter-disciplinary, however this one is concerned with finding a rule-based or a statistical model of the natural language, from a computational perspective. The Association for Computational Linguistics defines it as: "the scientific study of language from a computational perspective. Computational linguists are interested in providing computational models of various kinds of linguistic phenomena" [50].
The first sounds recorder was made by Thomas Edison in 1877 the Phonograph, despite not being able to process what was being said, this device could record and reproduce sounds. 75 years later in 1952 “Audrey” was launched by Bell Laboratories, it could only recognise digits being told by a user, it has a success rate of 97% - 99% accuracy rate, if it was adapted to the user [37]. Fast forward a decade and in 1962 Shoebox is released by IBM and it was able to do math through voice commands. Shoebox could understand 10 digits and 6 control words, such as “plus,” “minus” and “total” [22]. Just 5 years later IBM releases Automatic Call Identification system, which enabled engineers to receive and send spoken words to a computer.

In the 70s, many meaningful advances were made, mostly due to the Department of Defense (DOD) Defense Advanced Research Projects Agency (DARPA), which was funding the Speech Understanding Research (SUR) program, one of the largest in the history of speech recognition. One of the programs to be developed from it was Carnegie Mellon’s “Harpy’ system [28], which had the capacity of understanding at

<table>
<thead>
<tr>
<th>Inventor</th>
<th>System</th>
<th>Year/Decade</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thomas Edison</td>
<td>Phonograph</td>
<td>1877</td>
<td>First Device to record and reproduce sound [33].</td>
</tr>
<tr>
<td>Bell Labs</td>
<td>Audrey</td>
<td>1952</td>
<td>Understands Spoken digits [37].</td>
</tr>
<tr>
<td>IBM</td>
<td>Shoebox</td>
<td>1962</td>
<td>Capable of understanding up to 16 English Words [12].</td>
</tr>
<tr>
<td>IBM</td>
<td>Automatic Call</td>
<td>1971</td>
<td>Allows people to talk and receive spoken answers from a device [33].</td>
</tr>
<tr>
<td>Carnegie Mellon</td>
<td>Harpy</td>
<td>1970s</td>
<td>Understands 1011 words [28].</td>
</tr>
<tr>
<td>IBM</td>
<td>Tagora</td>
<td>1980s</td>
<td>Capable of understanding 20 000 spoken words [22].</td>
</tr>
<tr>
<td>Xuedong Huang</td>
<td>Sphinx-II</td>
<td>1993</td>
<td>First large vocabulary continuous speech recognition system [26].</td>
</tr>
<tr>
<td>IDM</td>
<td>MedSpeak</td>
<td>1996</td>
<td>First commercial system that can process continuous speech [22].</td>
</tr>
<tr>
<td>Siri</td>
<td>Apple</td>
<td>2011</td>
<td>Digital personal assistant, it can recognize speech, it’s also able to understand the meaning of what it was told and take appropriate action [35].</td>
</tr>
<tr>
<td>Cortana</td>
<td>Microsoft</td>
<td>2014</td>
<td>Similar to Siri [31]</td>
</tr>
<tr>
<td>Echo</td>
<td>Amazon</td>
<td>2014</td>
<td>Voice-controlled speaker, it is powered by Alexa, a digital personal assistant [54].</td>
</tr>
</tbody>
</table>
State of the Art

least 1000 words, which is more or less the vocabulary expected from the average three-year-old.

In the 80s IBM, once again, releases the transcription system Tagora, which had the ability to process 20 000 words. For a user to use Tagora they had to train the system and do brief pauses between each word. Despite the leap forward achieved with Tagora it was still enough for it to be commercially viable. To do so it would be necessary to decrease the cost of computing and increase the processing power. This is where Lai and Vergo ground-breaking work, [25], lead to the invention of MedSpeak, in 1996, which would become the first commercial system to be capable of processing continuous speech. Although, MedSpeak was the first commercial available system in 1993 Xuedong Huang released SPhinx-II, which was the first system with large vocabulary, that was able to process continuous speech.

In 00s and the 2010s most of the work done would culminate in the release of Digital Personal Assistants which are able to recognize speech, understand the meaning and take due action, some of the most known digital assistants are Siri, Figure 2.2, Cortana and Alexa.

![Image](image.png)

Table 2.2: Siri Commands

At the current moment speech recognition could be divided into multiple sections, they can represent the different variables in the recognition, Table 2.3
State of the Art

Table 2.3: Some Variables in Speech Recognition

<table>
<thead>
<tr>
<th>Variable</th>
<th>Values</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speaker</td>
<td>Speaker-dependent or Speaker-independent</td>
<td>If it needs training for a user or not.</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>Specialized or Nonspecific</td>
<td>If a dictionary provided will include words related to the task or if will be for general purpose.</td>
</tr>
<tr>
<td>Flow of Speech</td>
<td>Discrete or Continuous</td>
<td>How speech is interpreted, does the user have freedom or has to follow a specific formula.</td>
</tr>
</tbody>
</table>

For instance, it can be discrete in nature [13], which means the application needs to adapt, learn. To achieve this, they require the user to speak slowly, insert pauses between words, without this the system will not be able understand. The other type, the more current one, is continuous, which allows its users to speak in a more natural way. Another section would be speaker-dependent or speaker-independent [13]. Speaker-dependent systems need the user to train the recognizer, so it can learn the speech patterns. Whereas Speaker-independent attempts to achieve the highest level of accuracy possible for all users, without the need for a user to train it. Finally, a system can also be divided by the size of the vocabulary, dictionary.

In 2008 an application to help nurses in triage was developed, in which they observed that without a specialized vocabulary, the system yielded an accuracy level of 75%. In comparison, utilizing a specialized vocabulary yielded a 95% accuracy rate [10].

Dragon Naturally Speaking Medical Edition, which was used for the purpose of writing clinical letters through speech input. Found that, at first, it was necessary to establish a control process, which utilized an audio recorder and human transcription. Since the system is speaker-dependent, it required the physicians to train. The training required them to complete a two-hour speech recognition engine session. However, after doing so, only one of the four physicians decided to continue the study, this was caused, in part, due to the frustration created by the training. The remaining participant recorded his time spent dictating the letter, where he alternated between speech recognition and transcription services. They found that it took the physician almost 10 minutes to complete one letter using speech recognition and only 3.3 minutes using the control process, transcription [23]. Additionally, the accuracy of speech recognition was only 90% whereas the transcriptions had an accuracy of 99% [23]. However, in recent times there has been an increase in reliability. Such as Google’s new milestone, Figure 2.1, their voice recognition software managed to obtain the same level of accuracy as humans.

There can also be systems that are discrete in speech, where the user must follow a certain group of rules like "Speak slowly", "leave pauses between words", a system can also force the user to say the commands while following a strict formula, for example "Create ... there" [7]. The opposite is also true with systems like Siri or Google’s Assistant where the user is free to say what he pleases, and the system will have try to figure it out.
State of the Art

In this case, the voice inputs can be used to handle the CSUD, create, select, update, destroy. The user would follow a hybrid syntax. On one hand it would be similar to "Put that There" [7], where the commands would have to be according to a fixed structure like "Create/Destroy ....". But on the other hand, the user should be free to say commands like "All blue chairs move to the right 2 meters", which don’t follow a pre-created syntax but still hold valid meaning, within the context of the system. Therefore system should still attempt to understand what is that the user has commanded.

To handle the processing of Natural speech two APIs - DialogFlow and Wit.ai - were analyzed. Both APIs follow the same pattern when dealing with speech, they have Actions, Intents, Entities, which make it simpler to take what is learned from one to another. However, this makes it so, that the shortcomings of one might also be present in the other. For example, both API are oriented to emulate a conversation:
the user ask a question and the bot will answer. Although some Speech-to-Text despite being deprecated it was used to handle voice command, it could handle both the text and the semantic.

### 2.2 VR

This section will be focused on VR and how it has evolved through the years and it will be complemented with a look at a few editors for VR, such as Unity VR, Sketchfab VR, Unreal, CryEngine and Lumberyard.

Table 2.4: VR throughout History

<table>
<thead>
<tr>
<th>Year</th>
<th>Name</th>
<th>Inventor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1962</td>
<td>Sensorama</td>
<td>Morton Heilig</td>
<td>Was a mechanical device that encompassed all the senses while viewing films.</td>
</tr>
<tr>
<td>1968</td>
<td>The Sword of Damocles</td>
<td>Ivan Sutherland, Bob Sproull</td>
<td>First Head Mounted display system.</td>
</tr>
<tr>
<td>1978</td>
<td>Aspen Movie Map</td>
<td>MIT</td>
<td>Simulation of Aspen, where the users could walk the streets in either summer, winter, or polygons</td>
</tr>
<tr>
<td>1991</td>
<td>The Cave</td>
<td>Carolina Cruz-Neira</td>
<td>First cubic immersive room.</td>
</tr>
<tr>
<td>1991</td>
<td>Sega VR headset</td>
<td>Sega</td>
<td>Headset for arcade games</td>
</tr>
<tr>
<td>1991</td>
<td>Virtuality system</td>
<td>Virtuality</td>
<td>First mass-produced, networked, multiplayer VR entertainment system.</td>
</tr>
<tr>
<td>1992</td>
<td>Virtual Fixtures</td>
<td>Louis Rosenberg</td>
<td>First true augmented reality experience</td>
</tr>
<tr>
<td>2001</td>
<td>SAS Cube</td>
<td>Z-A Production</td>
<td>First PC based cubic room</td>
</tr>
<tr>
<td>2014</td>
<td>Google</td>
<td>Google Cardboard</td>
<td>Virtual Reality Platform focused on mobile devices.</td>
</tr>
<tr>
<td>2016</td>
<td>Oculus</td>
<td>Oculus VR</td>
<td>Virtual Reality Headset for PC.</td>
</tr>
<tr>
<td>2016</td>
<td>HTC Vive</td>
<td>HTC Valve Corporation</td>
<td>Virtual Reality Headset for PC.</td>
</tr>
<tr>
<td>2016</td>
<td>Playstation VR</td>
<td>Sony</td>
<td>Virtual Reality Headset for the console PlayStation 4.</td>
</tr>
</tbody>
</table>

10
State of the Art

In 1950 Morton Heilig wrote about an "Experience of Theatre", where the viewer would get an experience that would involve all of the senses in such a manner that it would create an immersive viewing. In 1962 Heilig built his prototype and named it Sensorama. With it he released five short films, where the machine would engage the multiple senses, such as sight, sound, smell, and touch. This device was mechanical and predates digital computing [9, 19]. Fast forward 6 years and in 1968 Ivan Sutherland with the help of Bob Sproull created what may be considered the first head-mounted display (HMD) system, The Sword of Damocles, Figure 2.2. The HMD was so heavy that it was necessary to suspend it from the ceiling and its graphics were simply wire-frame model rooms.

In the 70s MIT released Aspen Movie Map, Figure 2.3, which allowed the user to roam through the streets of Aspen wither in Winter, Summer or polygon mode. For the first two modes MIT photographed every possible movement throughout the city streets in both seasons, the last mode is just a 3D representation of the city [32]. Originally the project was commissioned by the Cybernetics Division of the Defense Advanced Research Projects Agency (DARPA) of the US military. The US military was impressed with how the Israeli army handled the hostage situation at Entebbe, Uganda, in 1976, they built a crude replica of the airport and practised in it before the mission of saving hijacked Israeli citizens. The Department of Defense (DOD) had hoped the project would show a future where the computers would be able to create three dimensional simulations of hostile territory at a much lower cost and higher speed [32].

In the 90s a lot of development happened in Virtual Reality, The Cave (Cave Automatic Virtual Environment) [11]. In its essence it is a movie theatre, where the walls are made up of rear-projection screens. In the CAVE a user can see objects floating in the air; it is also possible to walk around and can get a full 360-degree view. The projectors inside the CAVE are used to display lifelike visuals. The Sega VR headset, was also released during the 90s, it was used for arcade games and the Mega Drive console. It made use of LCD screens in the visor, stereo headphones, and inertial sensors that allowed the system to track and react to the movements of the user’s head. Furthermore, the 90s also saw the release of the first mass-produced multiplayer VR system Virtuality, Figure 2.4. The system would end up costing up to 73,000$ per multi-pod system, they feature headsets and exoskeleton gloves.
Louis Rosenberg invented the first true Augmented Experience, Virtual Fixtures. It was developed at the U.S. Air Force’s Armstrong Labs, it used a full upper-body exoskeleton, that enabled a more realistic physical experience in virtual reality. The system allowed for the overlay of physically real 3D virtual objects that registered with the user’s direct field of view in the real world, producing an augmented reality experience that enabled sight, touch, and sound[41].

In the 00s there was the first PC-based cubic Room SAS Cube, which was developed by Z-A Productions. However in the 2010s was experiencing a new breakthrough with the release of many platforms and many headsets.

In 2014 Google cardboard, Figure 2.5, was released as a head mount for smartphones, its name derived from the fold-out cardboard viewer. As of 2014 10 million units of it have been shipped. Due to its success Google has released in 2016 the VR platform Daydream, with the current price of 99$.

During 2016 there was the release of Playstation VR, Table 2.7, Oculus Rift and HTC Vive, Table 2.6 and Figure 2.6, opening an era of easier access to privately-owned VR systems.
Oculus Rift is the first of the three headsets that are going to be covered, released in 26 March of 2016 after a successful Kickstarster campaign, where it raised 2.5 million$ [27] and after being purchased by Facebook for 2 billion $. To be able to run you are required to have to have a system with the minimum specifications detailed in Table 2.5 [34]:

| **Graphics card** | NVIDIA GTX 1050 Ti / AMD Radeon RX 470 or greater. |
| **Alternative graphics card** | NVIDIA GTX 960 4GB / AMD Radeon R9 290 or greater. |
| **CPU** | Intel i3-6100/AMD Ryzen 3 1200, FX4350 or greater. |
| **Memory** | 8 GB+ RAM |
| **Video output** | Compatible HDMI 1.3 video output. |
| **USB ports** | 1x USB 3.0 port, plus 2x USB 2.0 ports. |
| **OS** | Windows 8.1 or newer. |
State of the Art

Table 2.6: HTC Vive Recommended Specs.

<table>
<thead>
<tr>
<th>Processor:</th>
<th>Intel™ Core™ i5-4590 or AMD FX™ 8350.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics</td>
<td>NVIDIA GeForce™ GTX 1060 or AMD Radeon™ RX 480.</td>
</tr>
<tr>
<td>Memory</td>
<td>4 GB+ RAM.</td>
</tr>
<tr>
<td>Video output</td>
<td>1x HDMI 1.4 port, DisplayPort 1.2 or newer.</td>
</tr>
<tr>
<td>USB</td>
<td>1x USB 2.0 port or newer.</td>
</tr>
<tr>
<td>Operating system</td>
<td>Windows™ 7 SP1, Windows™ 8.1 or later or Windows™ 10.</td>
</tr>
</tbody>
</table>

Table 2.7: Technical specifications of playstation VR.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>1920 x RGB x 1080 (960 x RGB x 1080).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh Rate</td>
<td>120hz, 90hz.</td>
</tr>
<tr>
<td>Field of View</td>
<td>Approx. 100 degrees.</td>
</tr>
<tr>
<td>Microphone</td>
<td>Integrated.</td>
</tr>
<tr>
<td>Sensors</td>
<td>Accelerometer, gyroscope.</td>
</tr>
<tr>
<td>Connection</td>
<td>HDMI, USB.</td>
</tr>
</tbody>
</table>

2.2.1 VR Editors

Some of the current editors, game engines, allow the creation of VR content such as Unity VR, Sketchfab VR, Unreal, CryEngine and Lumberyard.

Unity is a game engine, which supports 2D and 3D, drag-and-drop, scripting with C#. It used to support 2 other languages, UnityScript, Javascript for Unity, and Boo, but they have since been deprecated [1, 39].

Sketchfab, on the other hand is more of a platform to publish, share, buy and sell 3D, VR and AR content. It does provide a WebGL and WebVR viewer, allowing the display of 3D models on the web and they can also be viewed on any mobile browser, desktop browser or Virtual Reality headset [43].

In summary, Sketchfab allows the users to move freely around or inside the 3D scene using mouse, touch manipulation or in Virtual Reality. Further than only being able to use static 3D models, the viewer also allows to play and control 3D animations. Viewers can enable the Virtual Reality mode to make the model viewable in Virtual Reality headsets. The viewer is available through the website and mobile apps. Technologically speaking Sketchfab relies on the WebGL JavaScript API to display 3D on web pages in all major modern web browsers and it implements the WebVR JavaScript API [44] as to provide a Virtual Reality mode of its viewer on compatible VR headsets.
It is important to reinforce the idea that Sketchfab has been identified for the majority of its lifetime as a place to share, view, buy and sell 3D models, not as an editor [45], however as of 27 July 2016 they released the scene editor.

All the editors were compared in the following terms:

- **VR Support**
- **Asset Store**
- **3rd Party Speech To Text**
- **Asset Store**
- **Manipulation of Objects**
  - **Scale**
  - **Rotate**
  - **Move**
  - **Select**

The editors that were tested Unity, Sketchfab, Unreal, CryEngine, Lumberyard had very similar performances. The only expectations, being Sketchfab, that didn’t offer 3rd Party Support for Speech to Text and Lumberyard that so far doesn’t have an Asset Store.
2.3 Procedural & AI Generation

Procedural generation has been used mostly in the game industry in a way most
lot of work has been done in Procedural Generation of Content (PGC) throughout the
years. For instance, video game industry has been using such methods to simplify
the creation of levels and reduce the storage space needed. In the 80s Elite used
Pseudo-Random Number Generation (PRNG) to generate a very large universe [51].
In 2004 the game kkrieger, developed by the German group Farbrausch, used proce-
dural techniques to generate textures, meshes, and sounds that are used to create a
complex, immersive game, which required under 100 KB of storage, which in turn is
four orders of magnitude less than a similar games [20]. In theory, you could have
a completely procedurally-generated generated game, that goes through all the dif-
f erent components of a game. For instance bits could represent the textures, sounds,
environments, levels and even the story. PGC can be divided into different approaches
such as those described in Table 2.8:

Table 2.8: Procedural content generation approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization</td>
<td>Combines elements so that they will best fit some criteria.</td>
</tr>
<tr>
<td>Constraint Satisfaction</td>
<td>Creates elements must follow specified constraints.</td>
</tr>
<tr>
<td>Grammars</td>
<td>Defines a grammar that the algorithm should follow and expand to create content.</td>
</tr>
<tr>
<td>Content Selection</td>
<td>Selects the content from a library</td>
</tr>
<tr>
<td>Constructive</td>
<td>Builds content in an ad-hoc manner.</td>
</tr>
</tbody>
</table>

The optimization attempts a search for a combination of elements that best fit
some criteria [47], which can be specified mathematically by the system creator, or
curated by a human. A lot of the research done for PCG uses evolutionary algorithms,
which can be referred to as search-based PCG, which is an optimization approach.
Optimization approaches to PCG are usually computationally expensive, moreover
they are often used with a human-in-the-loop for the evaluation function [18, 40] and
to create personalized content [42] offline.

Constraint Satisfaction involves the declaration of specific properties and the con-
straints that the content will be subjected to. For instance, levels in the Math edu-
cation game Refraction are generated by searching for solutions for a set of design
constraints [46]. Declarative representations have the advantage that a designer is
able to specify how the content should appear without the need to specify how the un-
derlying algorithm should behave. The challenge with constraint-based approach has
its roots in determining an appropriate representation for facts about the generated
content.

Grammars are used by an algorithm, so that it can create content. These will not
simply be representational grammars that would have been used by an optimization-
based generator. Grammars can serve as production rules for the generation, without
any regard for the quality of the created content. Grammar-based methods strive to achieve a balance between designer-specified rules that the content should fit and computer exploration of the design space. They have been used for offline content generation [30].

Content Selection is a special case, since there is some discussion whether or not it classifies as content generation [52]. However in this work the approach will be similar to the one done by Smith [46–49] and take the position that, despite being simple it still is a form of PCG, in particular when it is used to procedurally create an environment for the user to explore or experience within different mechanics. One the advantages is its speed, it is one of the fastest methods for generating content.

Constructive algorithms use an approach to build, create, content in an ad-hoc manner by joining together previously customized blocks. Usually it has all of its building knowledge stored inside the algorithm. Although it may do some amount of search internally. The results are not tested against any kind of external heuristic, which normally helps to guide the search process. A downside of such approach is that the generators tend to be quite project-specific. They can be seen as content creators based on the selection of smaller pieces of content.

It is possible to further divide PGC by the state of connectivity when the generation happens [53] and by how the generators interact with the user:

Table 2.9: State of Connectivity [47]

<table>
<thead>
<tr>
<th>Connectivity State</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline</td>
<td>The Content generation is done before the user begins the experience.</td>
</tr>
<tr>
<td>Online</td>
<td>The Content generation happens during the experience, as the user is using the application.</td>
</tr>
</tbody>
</table>

**Offline** content generation is done before the user begins the experience, for example the generator utilized in the game Civilization IV [15] can be placed within the group of offline generators, as the entire map is generated before the game begins.

**Online** content generation happens while the user is using the application, preferably it will act as a response to user interactions. For example Left4Dead generates scenarios by creating enemy encounters dynamically, based on the calculated stress level of the players [8].

Table 2.10: User Interaction with the Generator [47]

<table>
<thead>
<tr>
<th>User Interaction</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>The user simply interacts with the created content, has no control whatsoever.</td>
</tr>
<tr>
<td>Parameterize</td>
<td>The user is able to influence how the content will be created by sending some parameters.</td>
</tr>
<tr>
<td>Preference</td>
<td>The user can influence how the content will be generated, by evaluating previously generated content.</td>
</tr>
<tr>
<td>Direct manipulation</td>
<td>The user takes an active role in the creation of the content.</td>
</tr>
</tbody>
</table>
None, as the name implies in this type the user has no interaction whatsoever with the generator, it will simply interacts with the content created. You can find examples of this in the game Minecraft where the biomes are created before the player is allowed to start [36].

Parameterize, in this level of interaction or control, the user is able to influence the outcome of the generator by manipulating it’s parameters, for instance in a map generator the user would be able to do things like define that there would be a map without ocean or a single large continent surrounded by water.

Preference, is a type of interaction where the user provides feedback to already content created and by doing it will help guide the generator in the creation of content in the future.

Direct manipulation, the user takes, as the name implies, an active role in the generation of content, where he/she will be able to alter how the content looks. An example of this would be the creation of Sims in any sims game where the user is able to completely alter the way a character looks after it was created by the generator [2].

Other aforementioned aspects have already been touched upon by other programs such as using AI generation with the purpose of creating full stories with Versu [14]. In Versu they go even further and simulate social interaction in game, although the social mechanics are done in a simple way, they give the user the freedom to divert from any pre-made story done by an author [14]. Music is another field where there have been some successful experiments with AI generation. Darwin Music [29] is one of the many current examples of computer generated music.

2.4 Combining Inputs

When concerning combining inputs there was work done in from 2003 to 2004 and published in 2008, where an experiment was carried out to find out the potential of using multimodal inputs with VR as a support for mobile nursing [10]. Some of studies made about VR, especially those concerning healthcare have limited themselves to a single modal use, [21]. It has been reported that VR could be more effective when combining different input models [4]. It was also reported that touchscreens can perform better if used with a voice interface [6].

Taking the results of the aforementioned works into account, the application, should at least, be able to speed up the process to technical professionals. Another resemblance between the works mentioned is that both of them use keywords to know what should actually be "heard" by the application or just ignored.

2.4.1 Key differences

While Changs’ work [10], was used, it is important to denote that there are some relevant differences between their work and this dissertation. The main one being Changs’ work attempted to improve the work of professionals, all of them had experience in the field they were working in the Emergency Department (ED), 75% had a bachelor’s or higher degree and about 50% had worked in the ED for more than 8
years [10]. In the end the main goal was to make the job easier, however it did not simplify the core "mechanic", which is our goal, to simplify a technical task, so that non-technical people are able to have better grasp of it.

2.5 Summary

While most areas mentioned in this section have a lot of work done upon and are in constant evolution, their combination in the context of creation of 3D scenes for virtual reality however, has not been widely explored, which leaves room for contributions to be made. The speech processing will be discrete, in the sense, that not everything the users utters is of interest, therefore not everything is going to be processed. The application should have an engine, that is able to create and support external assets and integration for multiple asset repositories, which led the editor to be picked be Unity 3D. Furthermore, the procedural generation will be a mix of Preference and Direct Manipulation, the user will be able to alter the created scene as he sees fit and he will also have the possibility to classify the content give to him either by the search of the external databases or procedurally generated.

The application will allow "traditional" editing using the input of the HTC Vive hands, it will also allow voice commands to be issued. Through said voice commands it will also be possible, besides editing the scene, to request new objects to be added, said objects will either be taken from external databases such as ShapeNet [38] or procedurally generated. However due to time restrictions of this dissertation, to make the project more feasible it would be necessary to restrict the scope of this proof of concept. The focus would be more on integrating the voice commands with the traditional editing offered in Unity 3D and use a more naive approach to the search in external databases and the procedural generation, meaning that this project would focus more on the multimodal aspect of the dissertation, this decision is supported by reiterating that VR with multimodal inputs can be a more efficient tool [4].
Chapter 3

Architecture

This chapter presents a view of the proposed application architecture in abstract view of the application, how it works, most notably the different interactions with the external services, the APIs. The more detailed approach to the implementation will be presented in chapter 4.

3.1 Context

This thesis started out with two key questions "is it possible to simplify the content creation for non technical people" and "will the usage of already made assets speed up development". To attempt to answer these questions a proof of concept was conceived.

From the Chapter 2 it can be seen that the combination of inputs has long been used as a way to simplify or speed up tasks, however it has also been barred with some problems due the specificity of the requirements. Namely, several exercises were conducted in an attempt to transcribe what was being said on a specific situation in order to reduce the amount of paper work required to be filled by the users, so that they could reallocate their time for more important tasks.

Natural Speech recognition has been steadily improving across the years, with Google’s software achieving the same level of speech detection as a human, Figure 2.1.

Virtual Reality has entered a new golden age, in which the creation of content and availability of different headsets has never been seen before. Long gone are the days of Sensorama, Table 2.4. However, despite all of this growth and availability, the learning curve to create 3D and VR content is still a steep one. This leads to products with sub par quality.

AI & Procedural Generation has been used to generated content for a very long time, for example Elite Dangerous used it in the 1980s as seen in the section 2.3. For this proof of concept PCG will be used to allow the user to place objects in certain layouts.
3.2 Requirements

The requirements of this dissertation will be discussed in this section. Before detailing them, it should nevertheless, be established what is the meaning of requirements, in this work.

Requirements, in software engineering, are described as the multiple functions that a system, as a whole, should be able to fulfill according to the existing stakeholders, that is the users. As such the requirements of this dissertation are described in the Table 3.1

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow user to do Voice Commands</td>
</tr>
<tr>
<td>The user should be able to dictates the command.</td>
</tr>
<tr>
<td>Parse Users Voice Commands</td>
</tr>
<tr>
<td>The application should be able to split the Commands into different components of interest.</td>
</tr>
<tr>
<td>Support External Assets</td>
</tr>
<tr>
<td>The application should be able to import and handle Assets from sources outside the chosen engine.</td>
</tr>
<tr>
<td>Allow For Editing of Objects</td>
</tr>
<tr>
<td>The application, system should be able to allow the user to edit the objects that have already been placed in a scene.</td>
</tr>
<tr>
<td>Immersive Editing</td>
</tr>
<tr>
<td>The application should allow the user to feel immersed in the scene while editing. it.</td>
</tr>
<tr>
<td>Procedural Generation of Content</td>
</tr>
<tr>
<td>The application should offer support for procedural generation of Content. In this case, placing objects in Layouts</td>
</tr>
<tr>
<td>Reutilization of Assets</td>
</tr>
<tr>
<td>A user should be able to re-utilize assets that have already been either imported or placed on the scene, namely from outside sources.</td>
</tr>
</tbody>
</table>

The user should have a way to **dictate a command**. Afterwards the application should be able to parse what has been said. To do so, the voice file, that contains the command, will be sent to through a number of API calls. It will be these calls that will handle the Speech to Text and Parse of the command.

**Support for External assets**, refers to the process of making queries to an external repository and handling the import of content from said repositories. In this proof of concept there is, only, support assets from Sketchfab and Poly.
Architecture

![Image](image.png)

(a) Poly.

(b) Sketchfab.

Figure 3.1: External Assets.

Immersive editing is achieved when a user feels completely focused on the scene and tasks at hand. To achieve this, the application utilizes VR to add an abstraction layer, that makes the user oblivious to the outside world. Therefore is it able to focus only on the edition of the scene.

Editing Objects is the possibility of modifying the assets in scene, through multiple actions, such as rotate, move, scale, the basic transformations.

Procedural Generation, in this context, is the ability to place objects in a more complex placements other than simply placing an object on the scene. In this proof of concept is limited to 3 layouts.

Reutilization of Assets is achieved through the ability to re-use assets already imported or download, but also through being able to use assets that are in the external databases, which were created by other people.

3.3 Functionalities

In this section the different functionalities of application are going to be exposed and explored, as described in Table 3.2, are going to be exposed and explored. However it is important to denote that our description of a functionality is done on a high level of abstraction, leaving the details to be explained further on into the dissertation.
Make Voice commands, Speech To Text and Parse Text, can be grouped as the Voice Command Module. Create, Edit and Select Objects as part of the Application Module, while Import from external Sources is in of itself a module.

By comparing the requirements in Table 3.1 with the functionalities Table 3.2, one would be able to notice an almost one-to-one match. This could be explained, because most of the requirements were well established before the development had begun. That allowed for most requirements to be well defined and specific on what they represent.

The ability to make a Voice Command, just like its name indicates, should allow a user to dictate a command that the application would then attempt to execute.

Speech To Text is the ability to translate what a user has spoken to text. Parsing, is the ability to breakdown what has been said and extract key information from it, while at the same time ignoring the irrelevant text.

Certain functionalities like "Allow for Editing of Objects", are vague, this means that there is a lot of work that could be done. Furthermore, it is important to take into account the exploratory nature of the concept proof. The idea was to allow and develop as many Editing Actions as possible, having only a minimum of actions that it should be able to do, like a CRUD but adapted to 3D/VR content editing, as such, the core functionalities were deemed as follows: Create, Delete, Move.

Other functionalities as Select Objects, are deemed necessary, cause they are a part of other functionalities, in this case Select is a part of Edit Objects, since it offers a way for the user to let the Application know, which object it has to modify.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make Voice Command</td>
<td>Allow the user to dictate a command.</td>
</tr>
<tr>
<td>Speech To Text</td>
<td>Convert the command recorded by the user to text.</td>
</tr>
<tr>
<td>Parse Text</td>
<td>Understand what action the user wants to do and act on it.</td>
</tr>
<tr>
<td>Select Objects</td>
<td>Allow the user to select objects without having to use a Speech Command.</td>
</tr>
<tr>
<td>Create Objects</td>
<td>Allow the user to create objects.</td>
</tr>
<tr>
<td>Import Objects from External Sources</td>
<td>Allow the user to import objects from external sources.</td>
</tr>
<tr>
<td>Edit Objects</td>
<td>Allow the user to edit the objects placed in the scene.</td>
</tr>
</tbody>
</table>

Make Voice commands, Speech To Text and Parse Text, can be grouped as the Voice Command Module. Create, Edit and Select Objects as part of the Application Module, while Import from external Sources is in of itself a module.
3.4 Concepts

Table 3.3: Concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset</td>
<td>Any kind of object that the user can interact with.</td>
</tr>
<tr>
<td>Poly Asset</td>
<td>Asset imported from Poly.</td>
</tr>
<tr>
<td>Sketchfab Asset</td>
<td>Asset imported from Sketchfab.</td>
</tr>
<tr>
<td>Editing Action</td>
<td>The actions the user can do.</td>
</tr>
<tr>
<td>Layout</td>
<td>The different ways the user can automatically place objects.</td>
</tr>
<tr>
<td>Microsoft Request</td>
<td>A call to Bing Speech API to convert the voice command to text.</td>
</tr>
<tr>
<td>Wit Ai Request</td>
<td>A request to Wit Ai API that can convert voice to text and/or parse the text given.</td>
</tr>
<tr>
<td>Voice File</td>
<td>The file that has the voice command recorded.</td>
</tr>
<tr>
<td>External Request</td>
<td>A request to Poly or Sketchfab to get external assets.</td>
</tr>
<tr>
<td>Pointer</td>
<td>The beam of light that leaves the remote.</td>
</tr>
<tr>
<td>Wireframe</td>
<td>When an object is selected an wireframe will appear around the object.</td>
</tr>
<tr>
<td>Headset</td>
<td>The virtual reality headset the user uses.</td>
</tr>
<tr>
<td>Teleporter</td>
<td>The way the user can move around the map.</td>
</tr>
</tbody>
</table>

An asset, in a more detailed explanation, is a 3D Model, that represent a group of points, that connected by geometric entities like lines, triangles, etc... Said assets are what the user will be able to create, edit. When talking about Poly and Sketchfab assets, they are regular assets, however they were imported from Poly and Sketchfab, respectively.

Editing Action refers to all the actions that users can take. Currently the user can create objects, scale their size, rotate them, move them, switch positions between two objects, duplicate an object (meaning an exact copy of the object will be created). However, it is important to denote that the copy and original are completely independent entities after the duplication. Finally there is the group action, this actions allows us to conglomerate objects, so that any action done to the group, or any of its members, will be replicated through out it’s members.

A Microsoft Request is a Web request made to Microsoft’s Bing Speech API, this request is made as an attempt to convert an audio file, which contains the users voice command, to a text, which should contain what the user has said.

A Wit Ai Request, can do what a Microsoft Request does i.e. attempting to convert the audio to a text, but is also able to parse the user command. The reason for using Microsoft Speech To Text API was due to an apparently better performance and results from Microsoft Speech To Text API when compared to the Wit Ai Speech to Text.

A file is a discrete representation of data on a storage device. In this scenario a Voice File is a discrete representation of the Voice Command. We take the byte samples that constitute the AudioClip and save them as ".wav" file. This allows for
Architecture

re-utilization of voice clips in further testing and training of the system, as well as a way to send the Voice commands to the necessary Services.

A Wireframe is a box defined only by its edges, either red or black, that contains an object that was selected through the usage of Touch Commands.

A Headset is the Virtual Reality Equipment used by the user, it can either be an Oculus Rift or a HTC Vive, currently both devices are supported by the application.

A Pointer, Figure 3.2, is the beam that is emitted from the remotes. This beam allows the user to select Objects and teleport around the scene.

![Pointer](image)

(a) Valid Location.  
(b) Invalid Location.

Figure 3.2: Pointer.

3.5 Proposed Solution

This section will cover high level description of the system as a whole, the different interactions between the multiple modules that make up the system.
In this editor, the user is inside the scene, in which he can create, edit and model the surrounding area.

To achieve this, there was an attempt to create an editor, that would allow the user to create content while using different interaction paradigms (voice commands, touch inputs) and while running Virtual Reality. The editor allows the user to obtain assets automatically from multiple sources, such as Poly, Sketchfab and the primitives from Unity. In the Figure 3.3 the workflow of how the user should use the application is showed.

First there would be a voice command such as “Create 5 cubes in a circle with radius 2”, afterwards, the application would attempt to create the object either through external requests or Unity Primitives. In this case the application would generate a cube, then the procedural generation would attempt to find in the command if there is a layout. In this example the layout is "Circle", in the end the layout would place 5 cubes in a circle with radius 2. The details of how this is implemented are explained.
3.5.1 Package Diagram

Diagram 3.4 shows how the Application can be split into packages. There are four big areas of the Application: Application, Speech To Text, External Requests and Procedural Generation.

The Speech To Text, is the module that will handle the conversion of Speech to Text and the parsing of what the user said. The module itself can be divided further, by splitting it into Microsoft Speech and Wit Ai, which are the different APIs that were used. Microsoft Speech handles the Speech To Text, while Wit Ai can both handle converting voice into text and the parsing of command. The parsed command is returned as a JSON.

The External Requests module, is responsible for querying the external repositories and is also responsible for handling the import of the Assets, from said repositories. Like the Speech To Text module, External Requests can also be divided into submodules. In this situation the split can be done into Poly and Sketchfab. They represent the different repositories that can be queried.

The Procedural Generation which is responsible for creating the Layouts and assigning Positions to all the Assets that are created or imported by the user.

Finally the Application, behaves like a centralized point, it is responsible for receiving the voice Commands from the Microphone, starting the requests to the Speech To Text and External Requests Modules. It acts as a controller for the entire process.

Figure 3.4: Package Diagram
3.5.2 Concept Blocks

The Block Diagram 3.5 gives us an overall view of how the different parts of the system interact with one another. As it can be observed, when the user makes a Voice Command, the Speech Processing block will handle the conversion of the Voice Command to a string. It will use that string to return a JSON that contains the instructions, that the program should follow to satisfy what the user has requested.

The instructions will be used by the Procedural Generation Module. Said Module will attempt to create what it was instructed in the JSON, which was created by the Speech Processing Module. In the process of creating the Objects there are multiple paths that the Application can follow, but in a more high level view of the process, it can either make a request to the External Requests Module or attempt to make use of Unity Primitives. At first it will attempt to create any of the Primitives, if that fails it will make the, aforementioned, requests.

The External Requests Module, does not guarantee that it will return an Asset. It only guarantees that a request to the Poly & Sketchfab databases will be done. Both requests support a number of constraints that can be placed on the assets as a way to narrow them down. In this section however it is only mentioned, as a way to supplement the Unity primitives, that there is support assets from both of this stores.
Chapter 4

Implementation

This chapter will address the application was developed and explain the most relevant technical details relating to it.

The application can be split into two main areas: the Main App and the Speech to Text. The Main App itself can be split into 3 sub areas: the voice command, the parse of text and the creation of content.

![Overall System Diagram](image)

Figure 4.1: Overall System Diagram.
Implementation

![System Diagram](image)

**Figure 4.2: System Diagram.**

### 4.1 Speech to Text

Before the application can start to understand what the user has said, it first needs to understand it. Therefore there is the need to take audio files, and transcribe what was being said. The solution was to use external APIs, that will receive the audio files and return a text string translating their understanding of what was being spoken, in the file. Wit.Ai and Bing Speech to Text were the chosen ones to handle the task.

**Wit Ai**

Table 4.1: Wit Ai Characteristics [55]

<table>
<thead>
<tr>
<th>Time Limit(s)</th>
<th>Content Type</th>
<th>Bits</th>
<th>Rate</th>
<th>Endian</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>audio/wav</td>
<td>8</td>
<td>Integer</td>
<td>Big</td>
<td>Text, Entities, msg_id</td>
</tr>
<tr>
<td></td>
<td>audio/mpeg3</td>
<td>16</td>
<td>Little</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>audio/ulaw</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>audio/raw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Implementation

**Time Limit** refers to how many seconds the audio can have, in this case a stream can only have up until to 10 seconds.

**Content type** refers to the File formats that are supported. Currently there are four types that Wit ai supports.

WAVform Audio File Format (WAVE) but more commonly known as WAV due to the filename extension. Is an audio file standard developed and maintained by Microsoft and IBM. uncompressed audio. It stores information about the file’s number of tracks, sample rate, bit depth, and whether it’s mono or stereo.

MPEG3, in this scenario, refers to the more commonly known MP3 format. It uses lossy data compression to encode data while using approximations and the partial discarding of data. This will allow for large reduction in file size when compared to uncompressed audio. The compression works by reducing/approximating the accuracy in parts of the sound that are considered to be beyond of what most people are able to hear. It was designed by the Moving Picture Experts Group (MPEG).

\(\mu - \text{law}\) is an encoding mostly used in North America and Japan for digital telephony. \(\mu - \text{law}\) samples encoded in 8 bits, logarithmically. It is used because speech has a wide range.

RAW Audio format used for storing uncompressed audio. It can be compared to formats like WAV or AIFF, when it comes to size. Raw Audio files do not include any header information (sampling rate, bit depth, endian, or number of channels).

The **number of bits** bit depth, represents how much is stored, in bits, in each sample. It has a direct correspondence to resolution of the sample. For example if the number of bit was 8 the maximum value per sample would be 256.

Usually in Audio bit rate stands for the amount of bits that are conveyed or processed per unit of time. The usual standard for quantifying bit rate is bits per sec (bits/s).

**Endianness** refers to the order in which the bytes are arranged into larger numerical values, there are two possibilities Big and Little Endian. They depend on whether bits or bytes are ordered from the big end (most significant bit) or the little end (least significant bit).

In big-endian, when memory is addressed, the most significant byte is stored first (has the lowest address).

Little-endian reverses the sequence of addresses, this means the least significant byte is stored first, and has the lowest address, whereas the most significant byte is stored last, has the highest address.

Big-endian is mostly commonly used format in data networking, IPv4, IPv6, TCP, and UDP, are transmitted in big-endian order. Because of this, big-endian byte order is referred to as network byte order. Little-endian storage is popular with microprocessors, due to significant influence on microprocessor designs done by Intel Corporation.

For last is **the Response**, which is what Wit.Ai returns when it receives an audio file. **Text** is what the service believes to have been said in the file. **Msg_id** is an unique identifier. Finally **Entities** is how the program interpreters the text.

An user, to make a request, is required to have an header, as seen in Figure 4.3, this header values the ones described in the characteristics table 4.1.
Implementation

<table>
<thead>
<tr>
<th>Authorization</th>
<th>Headers (5)</th>
<th>Body</th>
<th>Pre-request Script</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>☑</td>
<td>content-type</td>
<td>audio/wav</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☑</td>
<td>bits</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☑</td>
<td>rate</td>
<td>8000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☑</td>
<td>endian</td>
<td>big</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.3: Wit Ai Request Header.

```json
{
  "_text": "create a capsule",
  "entities": {
    "editingAction": [
      {
        "confidence": 1,
        "value": "Create",
        "type": "value"
      }
    ],
    "object": [
      {
        "confidence": 1,
        "value": "a capsule",
        "type": "value"
      }
    ],
    "msg_id": "02WkJIjCcrPLZcmK3"
  }
}
```

Listing 4.1: Wit Ai response
Implementation

Microsoft Service

Table 4.2: Speech To Text Characteristics [3]

<table>
<thead>
<tr>
<th>Time Limit</th>
<th>Recognition Modes</th>
<th>Recognition Status</th>
<th>Profanity Handling</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Interactive</td>
<td>Success</td>
<td>Masked</td>
<td>Simple</td>
</tr>
<tr>
<td></td>
<td>Conversation</td>
<td>NoMatch</td>
<td>Raw</td>
<td>Detailed</td>
</tr>
<tr>
<td></td>
<td>Dictation</td>
<td>InitialSilenceTimeout</td>
<td>Removed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BabbleTimeout</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Error</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As it was mentioned in with Wit.Ai, Table 4.1, **Time Limit** refers to the maximum duration a command can have, in this scenario it can have up to 15 seconds duration.

The recognition mode allows the speech recognition to adapt based on how likely the users are to speak.

**Interactive** is when the user makes short requests and expects the application to perform a certain action. In this recognition mode, the users usually know they are talking to a robot, make smalls statements, have knowledge of what the application can do, therefore also know what they can say.

In the **Conversation** users are engaged in a talk with another human and the Speech to Text has an auxiliary role. Its job is to aid the parties involved in the conversation by displaying what has been said by the users.

Finally in the **Diction Mode** user speak longer sentences with the purpose of said utterance being used for further processing. It can be characterized by the fact that users know they are talking to a machine, the speech is usually planned in advance, full sentences with a duration of 5 to 8 seconds are the most common.

**Profanity Handling**, like the name implies, is how the API will handle when the user utters profanity. Currently it supports three modes. Masked, if this is option was selected it will replace the profanity with "**". In the Raw scenario profanities are left as regular text, this is in contrast with the Removed case, where the profanity is erased from the API response.

**Recognition Status** field is the equivalent of HTTP Codes, they provide information on how success full or not the request.

**Success** indicates that the Speech To Text was able to transcribe what was said.

**NoMatch** means it was not possible to match what was said to any known words in the selected language.

**InitialSilenceTimeout** means the audio stream contained only silence and the serviced timeout while waiting for speech.

**BabbleTimeout** means the audio stream contained only noise and the service timed out waiting for speech.

**Error** is used when the service encounters an error and is unable to process the audio stream.

Besides the **Recognition Status** a response will include the following values **Offset, Duration** and **DisplayText**.

**Offset** indicates how much time passed since the start of the stream until audio was detected, it is displayed in 100-nanosecond units. **Duration** indicates how long the
Implementation

speech phrase was, also in 100-nanosecond units. Finally the response will include the DisplayText, which was what the service understood that was spoken in the stream, if the Recognition Status was “Success”.

The response can also have different formats, if the value selected is Simples, in which case the response will only have DisplayText, RecognitionStatus, Offset and Duration. Whereas the response format will be different if the Detailed option is selected.

The Detailed option contains the fields Offset, Duration, Recognition Status, it will also include the N-best values, each N-best value entry also contains extra information.

Table 4.3: Detailed Information

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidence</td>
<td>Score from 0 to 1, how comfortable the service is with the answer</td>
</tr>
<tr>
<td>Lexical</td>
<td>Recognized Text without punctuation and capitalization</td>
</tr>
<tr>
<td>ITN</td>
<td>Converts the words &quot;five five five one two one two&quot; to the inverted canonical form 555-1212</td>
</tr>
<tr>
<td>Masked ITN</td>
<td>Masks Profinity</td>
</tr>
<tr>
<td>Display</td>
<td>The recognized text but with punctuation and capitalization</td>
</tr>
</tbody>
</table>
### 4.2 Process Text

This section discusses the different approaches to parsing the spoken commands by the user to Text, so that they can be split up into the different sections of a command. Afterwards, the program will attempt to understand what the user has commanded and act upon it. Currently there are two ways of parsing the text received from audio stream, one is using Wit AI exclusively, the other involves combining Microsoft Speech To Text, Figure 4.4a, and then send the Text for tokenization.
As seen in Figure 4.4b when using Wit Ai one can combine the Speech to Text and the Tex Processing in one step. Speech To Text done by Wit.Ai, has led to sub par results most of the times. Therefore there was the need to improve on the transcription of the voice commands that led to the creation of the intermediary step.

A request is made to another Speech To Text service, in this case, Microsoft Speech To Text API. Afterwards, when the application has acquired the transcribed text from the audio stream, it will be necessary to take the extra step. Said step requires us to take extracted text and make a GET request to https://api.wit.ai/message where it will take our sentence and isolate the proper entities.

However, in the end both paths that can be taken, return the same values 4.1, the major difference is that one of the paths offers more correct values than the other.

Is important to mention, that when comparing with the other speech to text options available, the speech to text Microsoft option offered an easier integration with the application in its current state. It also performed in a similar way to other services available, in which the difference was minimal [24].
### 4.3 Structure of a Request

Most requests must follow a basic structure, the only section of a command that is mandatory is the Editing Action. The only exception to this rule is the Create command which must always have an object specified.


#### Editing Action

This part of the request that identifies the user’s intended action.

<table>
<thead>
<tr>
<th>Editing Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
</tr>
<tr>
<td>Delete</td>
</tr>
<tr>
<td>Duplicate</td>
</tr>
<tr>
<td>Group</td>
</tr>
<tr>
<td>Move</td>
</tr>
<tr>
<td>Rotate</td>
</tr>
<tr>
<td>Scale</td>
</tr>
<tr>
<td>Switch</td>
</tr>
</tbody>
</table>

- **Create** is used to instantiate an object or a group of objects into the scene. Delete removes the objects or group completely from the scene.
- **Duplicate** re-instantiates an object already placed in the scene.
- **Group** allows the users to create conglomerate of objects. Any action made to the group or to an object inside the group, will be propagated to all the objects the conglomerate.
- **Move** allows to change the position of an object. The user can change the position on only one or all the axis.
- **Rotate** allows us to rotate the object in either X,Y or Z axis. The rotation is done in degrees.
- **Scale** allows us to increase the size of an object.
- Finally **Switch** allows for two objects or group of objects to swap their position between each other.

#### Object

This part of the request represents what will be acted upon. This means it will be the subject of the Editing Action, restricted by other variables such as Object Colours, Number, ID and Layout.
Implementation

**Object Colour**

The number of colours supported so far is 16 Colours. The colour name is used to attempt to re-create them as Unity Colours.

**Number**

Represents the number of objects to be created, it is part of the Object "text group". Can either be in a numerical or written way. Articles such as "an", "a", "the" are supported and assumed to represent singular entity.
Implementation

Layout

Table 4.5: Layouts

<table>
<thead>
<tr>
<th>Layout</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circle</td>
<td>It will place N objects in a display that mimics a circle with a given radius</td>
</tr>
<tr>
<td>Grid</td>
<td>It will place A x B objects in a display that mimics a grid with A width and B height</td>
</tr>
<tr>
<td>Square</td>
<td>It will place objects in corners of a square with side of value N</td>
</tr>
</tbody>
</table>

Circle Layout

In this layout objects are displayed as if they were on circumference with radius N, effectively creating a circular area.

Create 8 green spheres in a circle with radius 10.

- editingAction
- object
  - t  wit/number
  - t  objectColour
- layout
  - t  wit/number

Figure 4.8: Circle Layout

Square

This layout is the more specific, as it places the objects in the vertices of the square, whose size N is previously defined by the user.
Implementation

Grid Layout

The Grid Layout places objects as if they were part of a Matrix with dimensions MxN, where M is the number of objects per Line and N the number of objects, both of this values are specified by the user in the command.

4.4 The creation of Objects

Throughout this section the process of creating an object will be explained. There are two main groups when looking at object creation in this application. On one hand, the first group is defined by the attempt to create an object, that is a primitive of Unity 4.6. On the other hand the second group is defined by requests made to external databases, each database could be a subgroup of their own. However for
Implementation

simplicity sake, all external requests are put in the same abstraction level, despite how different the way the request to the APIs is and how different the handling of the requests is.

Primitives

Table 4.6: Unity Primitives

<table>
<thead>
<tr>
<th>Primitives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cube</td>
</tr>
<tr>
<td>Sphere</td>
</tr>
<tr>
<td>Capsule</td>
</tr>
<tr>
<td>Cylinder</td>
</tr>
<tr>
<td>Quad</td>
</tr>
<tr>
<td>Plane</td>
</tr>
</tbody>
</table>

Figure 4.11: Unity Primitives

External Requests

When a request is made to create an object is not one of the Unity Primitives, as shown Table 4.6 and Figure 4.11, it means it is an External Asset. Therefore the next step will be to make a query to Poly and Sketchfab. It is through these two asset repositories that the application will complement what is offered by Unity, through its primitives.

Table 4.7: Poly Request

<table>
<thead>
<tr>
<th>Variables</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>keywods</td>
<td>strings</td>
</tr>
<tr>
<td>Curated</td>
<td>true/false</td>
</tr>
<tr>
<td>PolyMaxComplexityFilter</td>
<td>Complex, Medium, Simple, Unspecified</td>
</tr>
<tr>
<td>PolyFormatFilter</td>
<td>Blocks, FBX, GLTF, GLTF_2, OBJ, TILT</td>
</tr>
<tr>
<td>PolyOrderBy</td>
<td>Best, Liked, Newest, Oldest</td>
</tr>
</tbody>
</table>
Implementation

**Keywords** represent what the user wants to create, for example if the user were to issue a command like "Create a robot", the keywords would be robot. Since this is the object the user is attempting to create and add to the scene.

The curated flag is a way to restrict the assets search. If the flag is set to true, it means the user will only received content that has been previously approved by Poly. Furthermore it is also possible to restrict the assets that will be received by limiting their complexity. However, this restriction is not mandatory, the user is free to ignore it, if he so desires, when searching for content.

Furthermore, Poly offers a way to restrict, even further, the assets that were received, by allowing us to filter the format in which the assets are serialized/encoded.

Finally, it is also possible sort the query results in different orders. There is the Best, which sorts by highest rating asset, the Liked, which sorts by most likes, the Newest, which puts first the more recent assets, and finally the Oldest, this sorting orders delivers the oldest assets first.

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocks</td>
<td>As the name indicates, these are simple 2D blocks, they are the &quot;basic&quot;.</td>
</tr>
<tr>
<td>FBX</td>
<td>Proprietary format. It is mostly used in cinema and video games.</td>
</tr>
<tr>
<td>GLTF</td>
<td>A file format for 3D scenes/models that uses the JSON standard. It has been described, by its creators, as the &quot;JPEG of 3D.&quot;</td>
</tr>
<tr>
<td>GLTF_2</td>
<td>Enhanced version of GLTF.</td>
</tr>
<tr>
<td>OBJ</td>
<td>A geometry definition file format developed by Wavefront Technologies to be used in its Advanced Visualizer animation package. The OBJ file format is a simple data-format that represents 3D geometry alone.</td>
</tr>
<tr>
<td>TILT</td>
<td>Is the format used by Tilt Brush an App developed for HTC Vive and Oculus Rift.</td>
</tr>
</tbody>
</table>

### Sketchfab Request

<table>
<thead>
<tr>
<th>Variables</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Animated</td>
<td>true/false</td>
</tr>
<tr>
<td>Category Name</td>
<td>string</td>
</tr>
<tr>
<td>Staff picked</td>
<td>true/false</td>
</tr>
<tr>
<td>Max Faces Count</td>
<td>[0, 10K], [10K, 50K], [50k, 100k], [100k, 250k], +250K</td>
</tr>
<tr>
<td>Query</td>
<td>Keywords</td>
</tr>
<tr>
<td>Sort</td>
<td>Likes, Recent, Views, Relevance</td>
</tr>
</tbody>
</table>

**Animated** is a boolean flag, that allows the user to select assets that have animations integrated.

**Category Name** is a way to further restrict the assets one of, as of the moment, 19 categories in Sketchfab. It is important to denote that this is an optional value, and that is possible to search for assets across all categories.
Implementation

The **Staff picked** flag, works like the Curated flag in Poly, is a way to pre-validate assets. This means, that the user will only see assets that the Staff at Sketchfab validated.

**Max Faces Count** is a way to restrict the assets, through their complexity. Like it can be seen on the table 4.9 there are multiple ranges to pick from.

**Query** is the equivalent of keywords in Poly. They represent what the user is looking for in the repositories.

**Sort**, like with Poly, is the order the assets are shown to the user. There is **the Views**, which sorts by how many user views an asset has on its page, **the Likes**, which sorts by most likes, **the Recent**, which puts first the newest assets, and finally **Relevance**, this sorting orders delivers the according to a score give the assets by Sketchfab.

4.5 Object Selection

Before diving into how the Editing Actions work 4.6, it is important to understand how the user is able to select objects.

This section will be divided into two parts, the selection that can be done through Voice or Touch Commands.

**Selection With Voice Commands**

When the user makes a command he can choose to specify certain aspects that allow the application to pinpoint which objects he plans to act upon.

Currently the proof of concept supports selection through the colour, ID tag, name of the mesh/object. When the user utters the voice command he will specify what characteristics the object should possess.

If for some reason no object is selected the command will fail. However if the number of required objects is not attained - for example, the user requests 3 cubes, but there are only 2 - the command will still be enforced on the 2 cubes available.

```csharp
//Returns the set of checks and object must pass to satisfy the user query
Dictionary<string, Func<GameObject, string, bool>> GetEvaluateFunctions()

//prop is the name of the property being looked up (can be one of the following "objectcolour", "object", "objectid")
Func<GameObject, string, bool> GetEvaluateFunction(string prop)
```

**Selection With Touch Commands**

Selecting with the touch commands, Figure 4.12, is far easier, the user needs only to point the remote and activate the beam, pointer, once the beam hits a valid object it will count as selected. To de-select the user needs only to point the beam to another valid object. This selection has the limitation of only being able to select up to 2 objects/groups at the same time.
Any command said will affect all the selected objects, if the user wishes for an object to not be affected by the Command he should de-select the object in question.

4.6 Editing Actions

This section is going to unravel the details of how the multiple editing actions (table 4.4) work and details of their implementation.

Create

This command is responsible for the creation of objects be it either Primitives from Unity or from one of the external repositories. In this command there is a mandatory part the Object that must be created. The optional parts, pertain to the number of objects to create, the colour they should have, the ID tag, and finally their placement in the scene that will depend on the kind of layout chosen.

(a) External Objets (Poly or Sketchfab).  
(b) Unity Primitive.

Figure 4.13: Create Action.
Implementation

```csharp
// attempt prevents infinite loop
public void Create(string command, bool attempt)
{
    bool createdFlag = createUnityPrimitive(command);
    if (!createdFlag && !attempt)
    {
        // Check for cases like command = cute but user meant cube
        Create(WordDistance(command, PrimitiveNames), true)
        ExternalRequest.query(command);
    }
}
```

Delete

It is through the delete command that the user can remove objects from the scene. This command can be used with either of the voice commands "Delete" or "Remove". Most of the times, with different users, the Speech to Text would not understand "Delete" properly therefore the need to find a replacement arose. To answer that problem the option "Remove" was added as a synonym for Delete.

Duplicate

With this command the users are able to replicate an object/group that has been placed in the scene. It is important to mention, that despite the Copy being an exact match to the Origin they are, after the instantiation moment, completely different objects that can be interacted separately.
Implementation

**Duplicate Action.**

1. Position newPos = CreatePos(obj);
2. instance = Instantiate(obj, newPos, obj.rotation);
3. instance.SetActive(true);
4. instance.parent = WorldParentObject;

**Group Action.**

1. Object newParent = GetNewParent(obj, obj1);
2. if (newParent == null)
   return;
3. obj.parent = newParent.transform;
4. obj1.parent = newParent.transform;
5. obj.UpdateBrothers(obj1);
6. obj1.UpdateBrothers(obj);

The group Action has a few particularities, the first one is that it is the only action, other than Swap/Switch that can only be done with Touch Input. In its the current state of the application the command will only work if the objects were selected with the touch input.

The other particularity lies in how the groups are created. To create a group the application must first attain the parents of the objects it wishes to join. Afterwards it will face one of 3 possible situations:
Both Objects Parents are the World Object  If this is indeed the case then the only thing that needs to be done by the application is to create a new Object that will act as parent.

One of the Objects Parent is the World Object  Like the case before this one is also an easy solution, the object, whose Parent is the World Object is going to become a child of the other object Parent.

The case where neither of the parents are the World Object  This is the more complex situation, but essentially what happens is one of Parents merges with the other. The chosen object to act as parent will receive all of the other Parent Object.

It is important to denote that, regardless of the case, in all three scenarios all objects involved must update the List that they keep of whom their brothers are, even if they were not the ones selected. All objects inside the group must have an updated and accurate list of brothers.

Rotate

![Figure 4.17: Rotate Action.](image)
Implementation

```csharp
// get the string with rotation values
string rotateString = GetRotateString(components);
// Creates a Dictionary Like { [1,0,0]: 45 }  // the key is the axis of the rotation and value how many degrees
// it should rotate
Dictionary<Vector3, float> rotateTemp = GetRotateValue(rotateString, components);
if (rotateTemp == null || rotateTemp.Count == 0)
    // Nothing To Rotate
    return;
foreach (Vector3 axis in rotateTemp.Keys)
    obj.Rotate(axis, rotateTemp[axis]);
```

By using Rotate the user is able to enact rotations of the objects/groups placed in the scene. For this to happen an object must be selected. After an object/group has been chosen the application must find out how many degrees and in which axis it should enact the rotation. For this to happen there must a rotate string, this string has to meet one of the following criteria.

Be a single digit, like "90", in which the application will rotate 90 degrees in all of the axis (X, Y, Z). Or a combination of digits and axis, like "45 X 50 Z", with this string the object would be rotated 45 degrees in X-axis and 50 in Z-axis. The order in which the axis appear does not matter, neither does the order between number, axis, the only important criteria is that they must exist.

**Move**

By using this command that the users can translate an object through out the scene.

Much like Rotate this command follows an almost identical syntax. An object must be specified, either through voice or touch selection then a destination must be specified. A string to be considered a destination must meet one of the following criteria.

Be a single digit like 5, in this case the application will believe the user want an object to be moved to (5, 5, 5).

Be a combination of digit and axis, this means that the string should look something like this "5 X", which would lead the application to move the object to (5, Obj-PosY, ObjPosZ). The remaining axis can be added, a command could look like, as long as it follows the rule Number,Axis: "5 Y 2 Z", "1 X 4 Y 6 Z". The order between number and axis is irrelevant, the only requisite is that they both exist.
Implementation

Figure 4.18: Move Action.

Scale

Figure 4.19: Scale Action.

This command is used to modify the size of the object.

```csharp
float scaleFactor = GetNumberFromSentence(scaleFactorString);
gameObj.Scale *= scaleFactor;
```

Scale is done by multiplying the GameObjects Scale with **Scale Factor**

**Layouts**

All of the Layouts inherit from an Abstract Class Layout, which has implemented most of the methods, since the only change between the layouts is how objects are placed in the World Space.
Implementation

```csharp
// The only function the children class implement
Positions pos = CalculateNewPosOfObjects(layoutComponent, objectComponent, toCreate);
// If no pos was assigned no use in continuing
if (pos == null || pos.Count <= 0)
    return false;

//Create "copies" of the object to place in the layout
Objects objs = GenerateObjects(toCreate, pos.Count);
objs.Add(toCreate);
// Move all the objects to the correct position
PlaceObjects(objs, pos);
//All objects are placed under a group
CreateGroup(objs);
return true;
```

Therefore all the functions like GenerateObjects, PlaceObjects, CreateGroup are only implemented once. While there are, currently, three versions of CalculateNewPosOfObjects, one per Layout.

Circle

```csharp
Positions returnList = new Positions();

//Take the Radius from the string
float radius = ReadRadius(eValue);
// How many Objects the user wants to create
int numObj = ReadNumberOfObjects(objectComponent);

// Add a position for all the numObj that need to be created
for (int i = 0; i < numObj; i++) {
    var angle = (i / (numObj / 2.0)) * Math.PI;
    // For a semicircle, it should use (i / numObj) * Math.PI.
    float x = radius * Math.Cos(angle);
    float z = radius * Math.Sin(angle);
    //New position is added
    returnList.Add(new Positions(x, obj.limits.y, z));
}
```
Implementation

Square

```csharp
Positions returnList = new Positions();
Debug.Log("Square Layout");
// Extract the Size of the square, it’s the side
int size = GetSquareSize(eValue);

// If lower or equal to 0 there is point in creating
if (size <= 0)
    return returnList;

// X dimension of the objects
float widthX = obj.limit.x;
// Z dimension of the objects
float widthZ = obj.limit.z;
// Y dimension of the objects with an added delta
float widthY = obj.limit.y + 0.1f;

AddSide(returnList, -widthZ - size / 2.0f, widthY, -widthZ - size / 2.0f);
AddSide(returnList, -widthZ - size / 2.0f, widthY, widthZ + size / 2.0f);
AddSide(returnList, widthX + size / 2.0f, widthY, -widthZ - size / 2.0f);
AddSide(returnList, widthX + size / 2.0f, widthY, widthZ + size / 2.0f);
```
Implementation

Grid

```csharp
public Positions CalculateNewPosOfObjects(string layoutComponent, string objectComponent, GameObject obj) {
    Positions returnList = new List<Vector3>();
    GridSize gridSizes = GetGridSize(layoutComponent);

    if (temp == null)
        return null;

    int X = (int)gridSizes.x;
    int Z = (int)gridSizes.y;
    if (X <= 0 || Z <= 0) {
        // Not valid
        return null;
    }

    // X dimension of the objects
    float widthX = obj.limit.x;
    // Z dimension of the objects
    float widthZ = obj.limit.z;
    Vector3 startPos = Vector3.zero;

    for (int xi = 0; xi < X; xi++) {
        // Give spacing between objects
        Position tempPos = startPos + new Position(0, 0, widthZ * xi);
        for (int zi = 0; zi < Z; zi++) {
            // The new position
            Position pos = tempPos + new Position(widthX * zi + 0.5f, b.size.y, widthZ * xi);
            // Add to Current List
            returnList.Add(pos);
            // Update Pos
            tempPos = tempPos + new Position(widthX, 0, 0);
        }
    }

    return returnList;
}
```

Bot Training

To help improve how well the Wit.Ai bot worked, certain steps were taken to increase the accuracy, the two main steps were an website and a script.
The website (Figure 4.20) allowed users to utter voice commands or provide a text list of commands he wanted the bot to parse. Afterwards a human operator had to go to the Wit.Ai console and validate the parsing suggested by the Wit.Ai.

Figure 4.20: Website, where the users can feed commands to train the bots.

The script was a more automated form of training, the Wit.Ai would receive commands and with the commands he would also receive how the sentence should be parsed. The script would create N entities of all the specified commands. For example the script would send a JSON object like the following:

```json
{
    "text": "Create 18 red trucks",
    "entities": [
        {
            "entity": "editingAction",
            "value": "Create"
        },
        {
            "entity": "objectColour",
            "value": "red"
        },
        {
            "entity": "object",
            "value": "18 red trucks"
        }
    ]
}
```

### 4.7 Final Application

This section is going to cover the outcome of some of the requirements, 3.1, and functionalities, table 3.2, of the thesis. As mentioned in Chapter 3, the user should have the ability to import assets from multiple sources, such as Sketchfab & Poly.
Implementation

In the picture the figure 4.21 it is possible to observe the co-existence of the Unity Primitives, table 4.6 & picture 4.11, with the assets from the mentioned repositories.

Figure 4.21: A scene with assets from multiple sources.

One important step in this dissertation was, how to provide feedback to the user. It was mandatory for the users to understand, in which state, the application was in. For instance after a web request or an action were done the user should received of how things would be, it was also necessary to update the user on state changes that happen, as it is possible to observe in the Picture 4.22.
To finalize, it is important to show the user how the different selection allows him to edit the objects, Figure 4.23.
4.8 Summary

This chapter described the main implementation details of the different areas of the prototype.

It went through how the conversion of speech to text used requests to external APIs. It also covered how the prototype parses the text obtained from the command the user uttered.

Furthermore, it covered how the selection, creation, edition of the objects was implemented in the prototype, how the user could in fact do these actions and the details of how the aforementioned actions were implemented.

Finally, this section covered how the bot, mentioned in the Section 4.2, was trained and why it was necessary to do the training to complement the input given by the users, while using the application.
Chapter 5

Evaluation

This chapter will focus on how the success of the proof of concept can be evaluated. To this end, the assessment will be based on how close the application is to achieve the proposed goal and answer the research questions from section 1.2. It will also be presented what kind of studies were done, their methodology, the results. Finally it will end with a discussion of what can be learned and in what way do these results allow to answer the researched questions posed at the beginning.

To evaluate how well the prototype performed a number of metrics are used:

- Observed Metrics
- Questionnaire Metrics
- Measured Metrics

Observed metrics correspond to what the facilitator noticed in the participants. For instance, what the user said and how he/she acted, during the test execution. This was used, as an attempt, to perceive how well the users behaved and acted during the tasks and while using the application was observed and taken into account.

Another metric will be the Questionnaire, by using this we hope to gather information related to how the users perceived the user studies, the usability of the application, how much did the system aided them in the completion of the tasks, and finally if some thoughts about how the system could simplify the content creation and edition. It will be through this metric that most of the information related to the application will be collected.

Measured Metrics refers to the time the users take to complete the tasks, all users were timed. This was used, as an attempt, to assess if the differences in time between the two study cases would provide any hints to which editing paradigm is the better one.

5.1 User Studies

User studies were used as an attempt to find out if with the usage of multi-modal inputs that can simplify the process of creating 3D/VR content and as a by product allow users with no technical or with little experience to create tailored 3D scenes.
There was also interest in trying to figure out if the re-utilization of assets help speed up development time and improve quality of the content created.

The studies can be split into two cases, as seen in Table 5.1. One was done in Virtual Reality and utilizing the proof of concept. The second Case was done in a traditional editor; in this situation, the chosen one was Unity, due the application being built upon Unity. This makes it so that in both cases, the actions that the user can take are as similar as possible.

In both studies the user was given a list of tasks to perform, as shown in Figure 5.1 and a list that contains examples of the commands that he can perform, Figure 5.2. Said examples cover all the more complex cases of the commands the user can utter. This means, that most commands in the list can be used in a more simplified way, as it was showed in the Chapter 4.

Table 5.1: Tests Description.

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>Attempt to follow the instructions given</td>
</tr>
<tr>
<td>Case 2</td>
<td>Redo the same instruction set as in Case 1 but this time in the regular Unity Editor</td>
</tr>
</tbody>
</table>

Figure 5.1: User studies instructions.

5.2 Methodology

This section of the dissertation is responsible for providing a deeper look and explanation of how the user studies happened, and what they entailed to.

5.2.1 Voice Testing

Before diving into the work done to evaluate the application, it is important to talk about the tests that were done to find out how to improve the Speech To Text.
To do so, a group of 10 people were asked to read a list of commands similar to the ones at 5.2. The result of requests made to Microsoft Speech to Text would be written in a file. Afterwards, we analyzed the most commons mistakes returned by the request, and concluded that most of these errors had its roots on the accent of participants.

The next step was to gather the most common errors that resulted from the request made to Microsoft Speech To Test API. Afterwards, these errors would be used to correct the text given by the request This approach lead to a decrease in the percentage of errors and increase in the number of successful actions.

A test session would happen as follows:

The user would arrive, afterwards he would be told a small explanation of what the project was about. The user was told about the several challenges surrounding the creation of 3D/VR content. How while using Virtual Reality he/she could feel more focused and immersed during the course of the edition. A small description of the goals of the project was also told. The proof of concept aimed to allow the creation content while using different interaction paradigms such as voice and touch commands. The ability to obtain assets from several automatic sources and procedural content generation. Afterwards he would be asked to read and sign a consent. The consent entailed to the user rights, as well as to what would be done with their data, such as his answers to the Questionnaire and the pictures taken during the duration of the test session. They were also informed that all their be data would stored and used in an anonymised way.

After the formalities were done, a final introduction to what the application could do would be given.

This introduction went through how the user could perform voice commands and an example (Figure 5.2).

**Object Selection** User could select objects either by using voice, touch or both.

**Different selection of objects** The different ways a command could be uttered.

**Brief explanation** Which the editing actions were available and the outcome of said actions.
Evaluation

Now he would be asked to perform the tests.

In the first case, Figure 5.3, the aim was to find out how effortless and intuitive the tool could be for first time users. The user was guided through the case 5.1, but the instructions were given, so that they were only suggestion of what the user must do, not how they should do it. The intent is clear, he/she must complete all the tasks assigned, but the order and how he/she does them is entirely up to them.

The user was also provided with a couple of example commands Figure 5.2, these were made, so that, the user would be able to understand how to use all the editing actions. They covered edition with objects selected and not selected, creation of objects with layouts, colour, number.

While in the second study scenario Figure 5.4 the objective was for the user to complete the same tasks as in first case, but using a more traditional tool, in this case, Unity. The purpose of the second case was an attempt to draw a comparison between the ways of editing in the two modes.

5.3 Results

It is in this section that the data gathered in the user studies, is going to be presented and explored to what it entails and how it relates to the questions at hand.
Evaluation

Users Profile

To assess the usefulness of the developed prototype, there was the need to conduct a live test, in which individuals have a diverse background, concerning experience with VR and 3D computing, gender, as it can be observed in Table 5.2.

The users gathered had a variety of literary abilities, gender, ages and experience. The sample gathered had 75% male and 25% female, with people from High School degree to Master Degrees, Table 5.2, with ages gathered in the interval \([20,31]\) Ggraph 5.5 with an average of 23.05 with a standard deviation of 2.178, as well as with experience in VR/3D and without any previous experience.

When talking about the VR experience of the users it is possible to split them into three different groups. With this we mean the number of times an user has used VR. The experience they have in editing 3D content, how much work has an user done in either creating or editing 3D content. Finally the users can also be categorized by their experience in creating content for Virtual Reality.

The experience in VR, refers to VR as a whole, the concern was to understand how comfortable the participants were in using Virtual Reality and how much of a novelty this experience would be to them.

Experience related to 3D is related to how used to creating and editing 3D models a user is. This information relevant so that we can understand how much knowledge regarding the process and amount of work required to create and edit 3D models a user is aware of. The same can be said about experience related to the creation of Virtual Reality Content, but instead of 3D models, we are concerned with how much experience an user has in creating VR content.

With this sample, we hope to be able to gather multiple opinions from different backgrounds, so that it is possible to reach conclusions that are able to reach a more ample group of the population.

![Figure 5.5: Age of Participants.](image)

<table>
<thead>
<tr>
<th>Avg</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.05</td>
<td>2.178875857</td>
</tr>
</tbody>
</table>
Evaluation

Table 5.2: Characterization of the Participants.

<table>
<thead>
<tr>
<th>Value</th>
<th>Number</th>
<th>Percentage %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>15</td>
<td>75</td>
</tr>
<tr>
<td>Female</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>HighSchool</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>Bachelor</td>
<td>13</td>
<td>65</td>
</tr>
<tr>
<td>Master</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>Total</td>
<td>20</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Experience in using VR</th>
<th>Number</th>
<th>Percentage %</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 times</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1 - 5 times</td>
<td>12</td>
<td>60</td>
</tr>
<tr>
<td>6 - 10 times</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>+ 10 times</td>
<td>6</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Experience in editing 3D content</th>
<th>Number</th>
<th>Percentage %</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>11</td>
<td>55</td>
</tr>
<tr>
<td>I do it yearly</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>I do it monthly</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>I do it weekly</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>I do it daily</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>I have done it but not frequently</td>
<td>7</td>
<td>35</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Experience in editing VR content</th>
<th>Number</th>
<th>Percentage %</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>11</td>
<td>55</td>
</tr>
<tr>
<td>I do it yearly</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>I do it monthly</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>I do it weekly</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>I do it daily</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>I have done it but not frequently</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

A key topic of interest for our study, lies with the overall capacity to gather a better understanding of which functionalities are valued the most amongst those developed in our prototype. From the results, as highlighted in graph 5.6a, the functionalities of creating and deleting stand out as those that users have found most useful in the process with more than 70% of users considering them most useful. On the contrary, functions Switch and Group have not gathered much support in our sample, in fact both of these functions are the only one to gather less than 50% approval 5.6a.
The results regarding the commands group and switch, could indicate room for further improvement on the functionality itself, its accessibility or that it is not adding significant value to the application. From an implementation point of view, those two functionalities are the most restrict ones. To make use of them the user must first select the objects using Touch Commands. Then, only then, can he/she actually recite the voice command.

One possible improvement for these commands would be to allow voice commands the ability to support more than 1 object per command, which is a limitation of the current application. Another aspect worth noting is that these results are stronger amongst the community of users with prior experience in VR - all of them found that Create and Delete were considered 100%, by the users as a "Must Have", while Rotate increased to 78%. This could be happening since these users are more familiar with the typical barriers to edit content in VR and 3D. This could be justified since these users are more familiar with the typical barriers to edit content in VR and 3D, and as such are able to have a better grasp of how much the application simplifies the
Evaluation

process.

Furthermore, in a more analytical outlook at the users with experience responses, with one exception, all of the commands gained a minimum of 7%, with an average gain of 11.85% on how the commands are a must have. The only expectation is the Switch command that actually experienced a decrease in usefulness when comparing the user with experience with the overall experience, it suffers a loss of 8%. The loss could, possibly, be explained to the stricter rules surrounding the usage of the command.

All in all, results seem to support robustly that the application, by making use of the multimodal approach as well as by allowing external assets to be manipulated as if they were native to the editor, does indeed help to improve the usability by simplifying the workflow, making it easier for people with experience and non-technical users. It also, might lead to a speed up of the development time, by allowing user to use external assets that represent entities they wanted to add to the scene. Regarding the two main questions of our study, we do find positive evidence. First, the prototype improved experience for non-traditional users. Secondly, feedback regarding the reutilization of assets was also positive, graph 5.7.

From the users’ experience while using the application almost 50% of the users felt like VR made them more focused on the task, more than 50% of the users that editing in VR made them more aware of how the user would experience the scenes they were creating. Furthermore, if a deeper look was to be taken at the data, it would also be possible to see that most of the users believe that the creation of content could be simplified (65%) and 70% believe that the edition could also be simplified, through the usage of multi-modal inputs, Graph 5.8.

However, it is also important to denote that only 40% of the users agreed that using VR helped them focus on the tasks and only 45% of them believed that the combination helped them complete the tasks. This seems to clash with previous user answers that the combination and VR experience helped them and simplified the process, but this unexpected answers could lie with inexperience of using the app or how
Evaluation

the tasks were constructed. This is supported by the fact that only 45% of the users claim that the instructions were easy to follow, this could mean that the users were more focused on the completion of the tasks and following the instructions and not in the editing process itself.

Nevertheless it is important to refer that if one were to combine the users who "Agreed" and "Completely Agreed" into a single group, with the statements regarding the User experience (Graph 5.10), one would see that the approval values see a big increase. For instance the when looking at how easy the users felt that Voice Commands 35% would turn into 85% and the 40% becomes 75%. Such increase could be explained, due to the early stages of the functionalities, which might have led some user to feel more reticent to some of the functionalities.
Evaluation

I think Touch Commands would make it easier to Edit the objects in the task.

I think the combination of Voice Commands and Touch Commands helped me complete the task.

I think the combination of inputs could simplify content creation.

I think editing while in VR made me more aware of the user experience.

I think editing while in VR helped me focus on the tasks.

To understand how successful the proof of concept was, it is necessary to understand how at ease that the users felt while using the commands.

On one hand a quick glance at the graph 5.11 will tell us that most commands were considered very easy to use, with special mention towards delete and duplicate. This favoritism from the users, could have its roots on the fact that both of these commands can be uttered with only a single world as long as the touch selection was used by the user.

On the other hand the more controversial commands were switch, group and rotate. Concerning switch and group, the controversy could originate from the seemingly
lack of immediate impact on the scene that these commands hold. Rotates’ controversy could have originated on the more demanding structure around the action. In the case of this command, its minimal formulation is:

“Rotate $Object$ number Axis”  

From all the commands, this is the one that is the one that has the more complex minimal formulation.

However, it is important to mention that the most complex command that an user can utter is the create command if it is used in its full form, which is with number, colour and layout, Figure 5.2.

5.4 Discussion

From the answers collected from through users, the prototype seems to be able to answer in a positive way to both of the research questions (Section 1.2). It was able to simplify the process of creation to a point that satisfied both users with experience and without it. The satisfaction with the proof of concept seems to be higher with users, who had previous experience in either 3D or VR content creation. This as mentioned, could be attributed to knowledge of the barriers and difficulties, that the creation of content, in the aforementioned areas.

However, participants felt that through the touch and voice commands their work was made easier. It can also be said that more work could be done on procedural generation and certain commands need to be made more accessibly before the application is made available to a wider audience. On one hand, most users found that using external assets allowed them to have access to a lot more content, and much faster. Even if the final assets used in their scenes are not those imported, the tool still allows for the users to build quick prototypes. In is similar to software prototyping applied to the creation of 3D/VR scenes.

1The axis value could be either x,y or z axis.
Evaluation

Figure 5.12: Application Testing.
Chapter 6

Conclusion

In this section it’s important to, once again, remember what the research questions and what this prototype set out to accomplish. This dissertation had the goal to answer these questions:

• Can a multimodal combination of inputs enable non-technical users to create tailored 3D scenes?

• Can the re-utilization of assets help speed up development time and improve quality of a scene?

In the end and in a simple way the answer to both questions is yes. Through the questionnaire made it possible to ascertain, that according to the users, editing and creating content did indeed become simpler and more accessible, the more experienced users had a stronger belief than the less experienced, this could originate from the better understanding of the obstacles a users faces while creating and editing content.

It is also possible to see that the users, according to their experience, believe the re-utilization of assets will indeed speed up development, according to the written feedback some users mentioned that "work would not need to be replicated so often" & "it is quality work already done and reduces the steps needed to finish a product or experience.".

The application could target itself as a prototype creator. So what is software prototyping? It’s the activity to building incomplete applications with the purpose of being able to get feedback faster. By getting feedback in a faster pace, the creator is able to better tailor the application to the requirements asked by the client.

Through the same logic, with VR/3D the application could be used to create content in a fast and efficient way. This make it so that the users would be able to gather more feedback, in a faster way and iterate on it, leaving the final product with a vision more aligned with it’s target goal. The external objects and primitives could work as final objects or as placeholders for better quality assets to be created, added at the end of development.
6.1 Future Work

Three main areas for future exploration have been identified. The first two relate to both the external part and a core part of the application. From the core part, it would be possible to improve the usability, by allowing the commands to be more flexible. The second part resides with changing the core part so that it can adapt to a more flexible structure and allow for more optional parts, to introduce other commands such as positions, Tag and other components. The third area where the application could be improved is the creation of interfaces to give the user more control on how the application works.

User Feedback

One of the suggestions made by the participants of the tests was to allow the user to select in which direction they are facing when teleporting and not only the position.

Another suggestion is that, in scenes with a larger number of objects might be more difficult to find and edit a specific object in VR, therefore objects out of the field of view could be considered "hidden". This suggestion would allow the application to reduce the search time since the objects outside the field of view would not be considered. The only problem, is how to keep the list updated in an efficient way, one would have to consider if the trade off in keeping an updated list of viewable objects isn’t higher than the search. The answer, most likely, depends on the number of objects in scene, after a certain value X this suggestion might prove more efficient, the key part lies in finding the value X.

Other user suggests that a change of colours might make the selection (touch commands) more intuitive.

One user suggested that if the application were to process the voice commands locally, the suggestion claimed that following such an approach would probably lead to a faster and even more useful application. This suggestion has already been touched upon in the Replace Wit Ai service. The question raised is how much time would be gained, since there is always the need to make at least one Web Request to convert the audio stream to text. Will the reduction from two to one make the application that faster.

More Flexibility In the Voice Commands

On one hand the flexibility of commands could be achieved by increasing the training of the bot in Wit Ai. By improving it will be possible to increase the number of cases that are covered by the generation. Training the bot is done through the usage of a python script that would create instances of all the commands. As such, it is possible to train the service to allow for more flexible instances of commands.

Another way to increase flexibility might be to swap Wit Ai for another service or way of processing speech. Wit Ai currently is working acceptably, but this raises the question, if the parsing of text couldn’t be improved to be something more flexible, maybe a better division. On the other hand, another option would be to delve deeper into Regex or into Dialogflow, which seems to have a more regex approach than the one currently being used.
Conclusion

**Interfaces**

The idea behind the interfaces is to allow the user to have more power of how the application behaves and what it should do. For instance an interface could be created for the user to be allowed to customize the feedback messages, select which service he would like to use for the Speech to Text, from a list of those available. Be able to have more control on how the queries are made to the external services.

Essentially, these interfaces are more for the user to be able to have his/her own custom experience.

**New Engine**

Unity was built to create games, as such certain functionalities and workflows do not match our needs, or simply overcomplicate certain aspects. For instance Unity was built to have a group of pre-built assets to be spawned multiple times, adding more assets in runtime. Whereas, the system is expecting, that the user to import multiple new assets during run time.

A good replacement would be the XR Editor from Unity’s Technologies. For starters this project is an Editor built for editing in VR, just from that, which could solve many of the limitations inherent to Unity. It is an open source project, therefore outside developers are able to use and develop for it. However, despite this project being an open source the support and learning curve, for outside developers is limited.
Chapter 7

Appendix

7.1 Appendix A - Consent
DECLARAÇÃO DE CONSENTIMENTO
(Baseada na declaração de Helsínquia)

No âmbito da realização da tese de Mestre no Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto, intitulada Mixed interactive/procedural content creation in immersive VR environments, realizada pelo estudante João Moreira, orientada pelo Prof. Rui Rodrigues e sob a coorientação do Prof. Rui Nóbrega.

Declaro que compreendi a explicação que me foi fornecida acerca do estudo irei participar, nomeadamente o carácter voluntário dessa participação, tendo-me sido dada a oportunidade de fazer as perguntas que julguei necessárias.

Tomei conhecimento de que a informação ou explicação que me foi prestada versou os objetivos, os métodos, o eventual desconforto e a ausência de riscos para a minha saúde, e que todos os dados recolhidos serão anonimizados.

Explicaram-me, ainda, que poderei abandonar o estudo em qualquer momento, sem que daí advenham quaisquer desvantagens.

Por isso, consinto participar no estudo e na recolha de imagens necessárias, respondendo a todas as questões propostas.

Porto, __ de ___________ de _____

______________________________
(Participante ou seu representante)
Appendix

7.2 Appendix B - Questionnaire
Mixed interactive/procedural content creation in immersive VR environments

*Required

1. **Education Level** *
   
   Mark only one oval.

   - Highschool
   - Bachelor
   - Master
   - Doctoral

2. **Gender** *
   
   Mark only one oval.

   - Female
   - Male
   - Other: __________________________

3. **Age** *

4. **How many times have you used VR** *
   
   Mark only one oval.

   - 0
   - 1 - 5
   - 6 - 10
   - +10

5. **How much experience do you have editing 3D content?** *
   
   Mark only one oval.

   - None
   - I do it yearly
   - I do it monthly
   - I do it weekly
   - I do it daily
   - I have done it but not frequently
6. How much experience do you have in VR content? *
Mark only one oval.

- None
- I do it yearly
- I do it monthly
- I do it weekly
- I do it daily

7. From my experience I think content/assets from public repositories could speed up development time. *
Mark only one oval.

1 2 3 4 5
Totally Disagree ☐ ☐ ☐ ☐ ☐ Totally Agree

8. Would you like to elaborate your choice?

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

Do the first test case

Evaluate the first test case

9. I found the application easy to use *
Mark only one oval.

1 2 3 4 5
Completely Disagree ☐ ☐ ☐ ☐ ☐ Completely Agree

10. I think the Voice Commands helped me to complete the task *
Mark only one oval.

1 2 3 4 5
Completely Disagree ☐ ☐ ☐ ☐ ☐ Completely Agree

11. I found the commands intuitive? *
Mark only one oval.

- Yes
- No
- Other: ________________________________
Do the second test case

Evaluate the second test case

12. I found the instructions easy to follow *
   Mark only one oval.

   1 2 3 4 5
   Completely Disagree 0 0 0 0 0 Completely Agree

13. How easy were the commands to use (In VR) *
   Mark only one oval per row.

<table>
<thead>
<tr>
<th>Command</th>
<th>Very Hard</th>
<th>Hard</th>
<th>Neutral</th>
<th>Easy</th>
<th>Very Easy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scale</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Switch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duplicate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

14. How useful were the commands *
   Mark only one oval per row.

<table>
<thead>
<tr>
<th>Command</th>
<th>Useless</th>
<th>Not that much</th>
<th>Neutral</th>
<th>Useful</th>
<th>Must have</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scale</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Switch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duplicate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

15. I think Touch Commands would make it easier to Edit the objects in the task. *
   Mark only one oval.

   1 2 3 4 5
   Completely Disagree 0 0 0 0 0 Completely Agree

16. I think the combination of Voice Commands and Touch Commands helped me complete the task. *
   Mark only one oval.

   1 2 3 4 5
   Completely Disagree 0 0 0 0 0 Completely Agree

Final Considerations
17. I think Voice Commands would make it easier to Edit Content compared to Traditional Editing (ex: Unity) *
   Mark only one oval.
   1 2 3 4 5
   Completely Disagree  0 0 0 0 0  Completely Agree

18. I think Touch Commands would make it easier to Edit Content (ex: Unity) *
   Mark only one oval.
   1 2 3 4 5
   Completely Disagree  0 0 0 0 0  Completely Agree

19. I think the combination of Voice Commands and Touch Commands would make it easier to edit Content (ex: Unity) *
   Mark only one oval.
   1 2 3 4 5
   Completely Disagree  0 0 0 0 0  Completely Agree

20. I think the combination of inputs could simplify content creation. *
    Mark only one oval.
    1 2 3 4 5
    Completely Disagree  0 0 0 0 0  Completely Agree

21. I think editing while in VR made me more aware of the user experience. *
    Mark only one oval.
    1 2 3 4 5
    Completely Disagree  0 0 0 0 0  Completely Agree

22. I think editing while in VR helped me focus on the tasks. *
    Mark only one oval.
    1 2 3 4 5
    Completely Disagree  0 0 0 0 0  Completely Agree

23. I feel like Voice Commands would make it easier but not they way it was implemented. *
    Mark only one oval.
    1 2 3 4 5
    Completely Disagree  0 0 0 0 0  Completely Agree
References


REFERENCES


REFERENCES


82
REFERENCES


REFERENCES


