International Journal of User-System Interaction 13(2) 2020, 73-88

Voice-controlled 3D modelling with an intelligent personal assistant

Sonia Grigor, Constantin Nandra, Dorian Gorgan

Technical University of Cluj-Napoca, Computer Science Department
G. Baritiu 28, 400027, Cluj-Napoca, Romania
E-mail: sonia.grigor@student.cs.utclu.ro,
{nandra.constantin, dorian.gorgan}@cs.utcluj.ro

Abstract. In this paper we look into the feasibility of controlling the process of 3D modelling with the help of voice commands. The main motivation is represented by the potential increase in the accessibility of cluttered graphical interfaces to novice users, helping them to smoothen the learning curve. Throughout this paper, we present a solution that we have implemented to augment the interface of an existing 3D modelling tool for the purpose of creating and editing 3D objects. We describe the overall architecture of the solution, detailing the structure of the command set it employs and the main challenges it was designed to overcome. Finally, we seek to demonstrate the potential of the voice-interaction model by employing its functionality in a 3D modelling use-case and analyzing its relative performance when compared to the use of the graphical interface.

Keywords: voice based user interaction, 3D graphics editing, personal assistant, Amazon Web Services

DOI: 10.37789/ijusi.2020.13.2.2

1. Introduction

Besides the obvious artistic dimension of 3D modelling, making it indispensable in areas like game asset creation and special effects, it also has an important role in engineering, best exemplified by the empolyment of Computer Aided Design (CAD) tools. Therefore, 3D modelling, whether for artistic or engineering purposes, is a skill in high demand. However, the creation and editing of 3D models are very complex processes, typically presenting steep learning curves to novice users. Existing tools feature complex sets of menus and shortcut combinations that can be difficult to find and access (Xue, Kou, & Tan, 2009). Even for users with 3D modelling background, switching to another environment would require time and effort

spent in getting acquainted with a new layout and the different approaches to operations such as transformations or basic object editing.

Considering this context, one could see the merits of providing alternative interaction models to supplement the capabilities of graphical user interfaces. The goal would be to facilitate access to the wealth of features packed into 3D modelling software. Alternative means of interaction could prove more intuitive and allow for smoother learning curves for novice users, or simply increase the accessibility of the tools to users with disabilities.

Within this paper, we explore the possibility of using voice commands as an alternative means of interaction with a 3D modelling tool. More specifically, we seek to employ the speech recognition capabilities of the *Amazon Alexa* smart personal assistant to drive and control the visual interface of the *Blender* 3D modelling tool (Blender, 2020). By harnessing the power of natural language through voice commands, there might be potential to improve the effectiveness of the existing interface or make it more easily accessible to new users. In the following sections, we will present the solution that we have implemented to control some basic aspects of 3D modelling in *Blender*. We will also describe the structure of the command set that we are employing and show the results of their application within a modelling use-case.

2. Related work

The first implemented solution capable of speech recognition is credited to Davis, Biddulph and Balashek (1952). The system was designed to solve the problem of digit recognition and worked by matching the estimated frequencies of uttered vowels within a digit to reference patterns. Over the decades, incremental improvements have led to systems capable of utilizing speech waveform pattern matching technology (Rabiner, Levinson, Rosenberg, & Wilpon, 1979) to recognize hundreds of different words.

More recent developments usually adopt approaches based on deep neural networks, typically trained on large acoustic data sets. This approach is considered a quantum leap in acoustics modelling and has garnered the attention of research groups from technology giants such as *Microsoft* and *Google* (Deng, Hinton, & Kingsbury, 2013). The ability of neural networks to refine and continually improve speech recognition models by processing massive amounts of data, collected over the years, has led to some remarkable

performance improvements within the last decade. One such example is the *Google Voice* service, available to virtually all smartphone users today (Sak, Senior, Rao, Beaufays, & Schalkwyk, 2015).

Consistent, real-time speech recognition has enabled the development and mass deployment of intelligent personal assistants, like *Cortana* (Microsoft, 2020), *Google Assistant* (Google, 2020), *Siri* (Apple, 2020) and *Alexa* (Amazon, 2020). At first glance, they offer the user a multimodal interface for a given device, allowing for hands-free operation. However, they go beyond that, as they are created to adapt to the user's habits and preferences, based on data gathered while operating - hence the *intelligent* attribute.

The intelligent assistants can run on a user's personal computer or mobile device, and are constantly improving their performance using the data gathered from said devices. *Cortana*, for example, is present on virtually all computers running the *Windows 10* operating system, while both *Apple* and *Google* have deployed their virtual assistants on mobile devices running *iOS* and *Android*. In this regard, *Alexa* presents an interesting case, as it is typically deployed on a dedicated smart speaker – *Amazon Echo* – and can be employed for various types of tasks, either device specific – like searching the internet, streaming news and music – or user-customizable, like controlling various smart devices. The potential for user-customization is a very important aspect of *Alexa*, as it offers an extensive API that allows for the development and publishing of new behaviours – called skills – by users over the internet (Amazon, 2020). This includes the definition of commands and associated actions, opening the door to the creation of personalized voice-command interfaces for various software tools.

The addition of intelligent assistants could benefit software tools that rely on complex, cluttered, visual interfaces. Often novice users can be discouraged by the steep learning curve such software typically entail. One of the earliest attempts to remedy the problem using smart assistants was *Microsoft's Office Assistant*, integrated within the *Office 97* software suite (Microsoft, 1996). It was meant to offer users interactive and personalized assistance, give advice related to ongoing tasks and provide help when requested. Although it was poorly received by the public and seen as distracting and intrusive, it did provide insight into the psychological aspect of human-computer interaction, thus paving the way for today's intelligent assistants.

GPS car navigation systems have been some of the first applications to

see widespread use of voice-based interfaces. The main line of reasoning was the elimination of elements that could divert the driver's attention. Starting with detailed, recorded sets of voice navigatin cues, these systems have evolved to provide basic voice-command recognition capabilities (Dobesova, 2012). The field is continously improving, with recent developments demonstrating the integration of general-purpose natural language processing tools to analyze unstructured user queries and create route plans in *Google Maps* (Withanage, Liyanage, Deeyakaduwe, Dias, & Thelijjagoda, 2018).

A somewhat specialized area in which voice-control interaction has found use is that of solutions designed to assist people with disabilities. In this case, accessibility is the strong point of the voice interaction model. Examples include the work described within (Gawari & Bakuli, 2014) – a system that can be employed to assist in the navigation of visually impaired people within urban environments, integrating voice-control with GPS navigation and obstacle detection. A slightly different solution, designed to help users with motor disabilities, is described within (Puviarasi, Ramalingam, & Chinnavan, 2014). It allows users to control a wheelchair and a set of home appliances through voice commands.

Generally, 3D modelling tools employ complex, crowded visual interfaces, presenting steep learning curves to novice users. This is the case for most of the existing artistic 3D modelling and CAD tools. Efforts to improve the usability of existing tools have been exploring the concept of multimodal interfaces as a means of providing a more flexible and effective interaction with the system (Stivers & Sidnell, 2005). In this respect, virtual reality and voice control are some of the main research directions. In regards to the former, Huang, Lin, & Cai (2020) conducted a study showing the benefits of employng virtual reality (VR) technology to augment the traditional CAD modelling methods.

Voice2Cad (Voice2CAD, 2020) is an interesting example of a CAD modelling tool controlled through voice commands. In its description, the developers state that the voice interaction mode was implemented specifically to overcome the problem of searching through cluttered visual interfaces or relying on shortcuts – which are features of virtually all 3D modelling tools. A similar solution is presented by the authors of (Xue, Kou, & Tan, 2009). Called *Natural Voice Enabled CAD*, the system is beased on a semantic verb searching algorithm implemented by its authors, which is also presented within the paper. In another paper (Kou, Liu, & Tan, 2009), some of the same

authors show the effectiveness of a voice-augmented CAD system by measuring the reduced number of mouse movements when said feature is activated. They employ a quadtree-based method of tracking the mouse trajectory distributions.

3. Proposed solution – Voice controlled 3D modelling

Our solution was developed under the assumption that natural language presents an intuitive and easy to employ interaction model, which could prove valuable for novice users trying to navigate through the typically cluttered visual interfaces offered by 3D modelling tools.

3.1. System Architecture

Implementing a voice-based interaction model required two main contributions: the definition of a suitable command set and the design of a control module to respond to the commands. To provide the voice recognition capabilities for our solution, we rely on *Alexa'*s configurable skills functionality. This allows users to define customized commands, also reffered to as "skills". The API provided by *Amazon* is quite flexible, allowing for complex, composite commands, which can integrate numerous keywords and keyphrases. For the 3D modelling tool, we went with *Blender v2.82* (Blender, 2020), as it is one of the best choices in terms of interface access and control. Aside from being open-source, it features an extensive, well documented Python API, which grants programmatic control over virtually all of its features.

Figure 1 shows the architecture of the system that we have developed to add voice interaction capabilities to the *Blender* tool. The typical workflow for the execution of a voice command starts with *Alexa's* deployment device (either a smart speaker, or a mobile device) recording the spoken command and sending it to *Amazon's* cloud service for interpretation. Once received, the recording is analyzed by the AI speech recognition deployed as part the *Alexa* application, and a response is prepared to be delivered to the user. This response is registered as the feedback component of the custom command created by developers. The feedback message is delivered vocally, through the text-to-speech functionality provided by *Alexa*. As part of the registered command response, we have implemented a component that records the

interpreted command within a *DynamoDB* database hosted on an *Amazon* server, accessible through *Amazon Web Services (AWS)*.

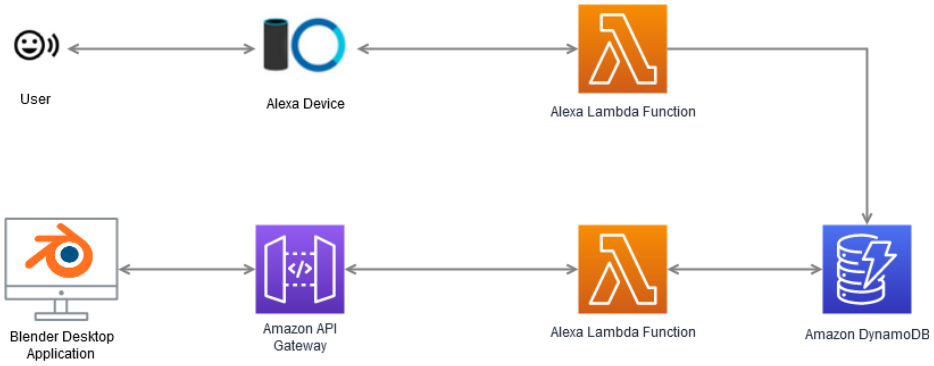

Figure 1 - System architecture

On the client side, within *Blender*, we are running a polling thread, with access to the *AWS* database of commands. It searches for the occurrence of new commands and executes predefined scripts, that we have created for each of the supported voice commands. These can access elements from the scene's internal state and can control various aspects of *Blender's* visual interface.

3.2. Voice command structure and grammar

When designing the command set, one of the most important aspects was the complexity of the commands. In 3D modelling, the most likely sources of complexity are the arguments required by a given command and the necessity to navigate and select elements to operate upon. In particular, the mouse control over the camera is very difficult to replicate with voice commands. This is because of the stream of input data generated by a mouse, which allows continuous control of the camera, as opposed to the discrete actions one could achieve with voice commands. Although we did implement commands for rotating and panning the camera, the usage of a pointing device is still preferable for navigating within the scene. In this sense, we see the vocal command set as an augmentation to the traditional interface, not as a complete replacement.

To formally describe the command set, we employ a graphical, tree-based representation of the grammar (Figure 2 and Figure 3), with elements from

the *BNF* notation, using uppercase letters for terminals and lowercase for non-terminals – elements that can be further broken down into component parts. For formatting reasons, where possible, we expand some branches all the way down to the last level, while creating separate sub-trees for the more complex ones. Also, we have omitted some of the more simple or obvious non-terminals, as we wish to concentrate on the high-level commands.

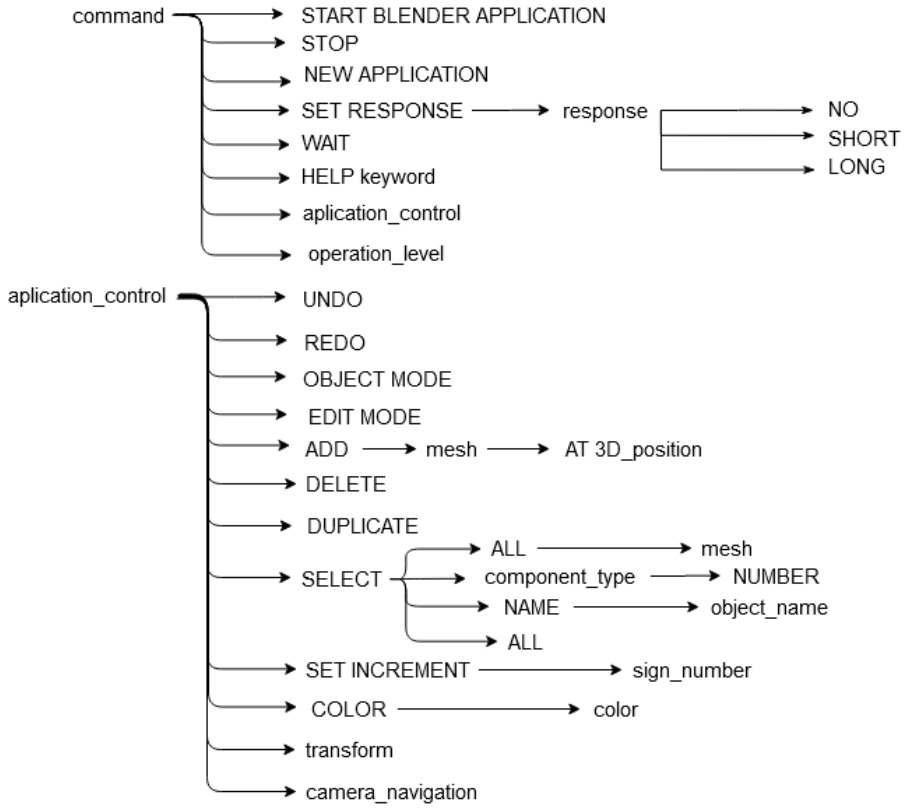

Figure 2 - Command structure – level 1

At the highest level (Figure 2 - top), we have created commands for starting and stopping the application, setting the detail level of *Alexa's* feedback messages, making *Alexa* wait for a predefined amount of time (a standby function) and providing help with the command structure. Allowing the user to set the level of feedback to "short" or "no feedback" could support

a more rapid and streamlined interaction with *Alexa*, minimizing the time the user would have to wait for the vocalization of the feedback message. The "help" command is intended to support novice users, by supplying the template based on a given command keyword.

At the level of application control (Figure 2 - bottom), we have commands for adding and removing mesh primitives, selecting primitives and mesh elements, undo, redo, and color application. Also at this level, we have commands for switching between the external and internal representations of a mesh ("Object" and "Edit" modes). These allow alternating operations to be made on a mesh or its internal structure.

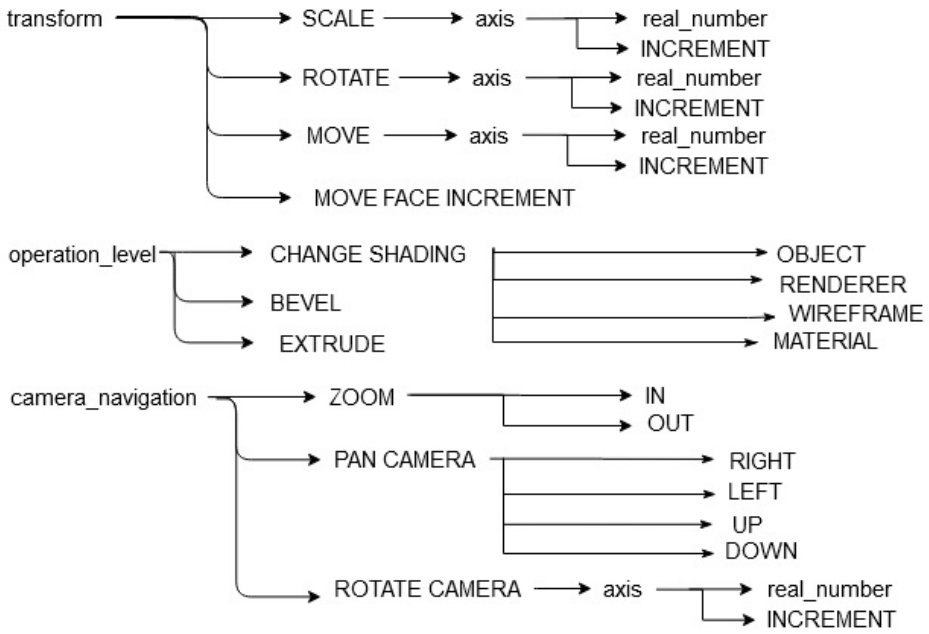

Figure 3 - Command structure – level 2

Going at the operations level (Figure 3), we have the basic translate, scale and rotate operations, all of which support axis-level operating modes. They can be accompanied by numerical arguments to specify the amplitude of the transformation or, lacking a numerical value, operate by default with an "increment" global variable that can be set by the user. Other operation-level

commands include setting the shading mode (solid/wireframe), setting the object's color and two modifier operations: edge bevel (split) and face/edge extrusion (extend). Both work with the implicit "increment" variable mentioned previously.

The commands for navigating the scene include camera zooming, panning and rotation. Panning and rotation are done on two axes: left-right & up-down. All three operate using the value of the "increment" variable. Controlling these three operations through voice commands is the main weakpoint of the system, as it is quite ineffective to navigate the scene in discrete steps. An alternative potential use of voice commands to direct and control continuous movements of the camera is also problematic because of the lack of accuracy stemming from the time lag between the uttered command and the effect of the action - about 1 to 1.5 seconds.

Object selection was another challenging operation to achieve without the use of a pointing device. At the moment, we select objects (meshes) using the name that each mesh is assigned by *Blender* when created. These follow a very simple format, consisting of the type of the mesh and an order number - Figure 4 - right.

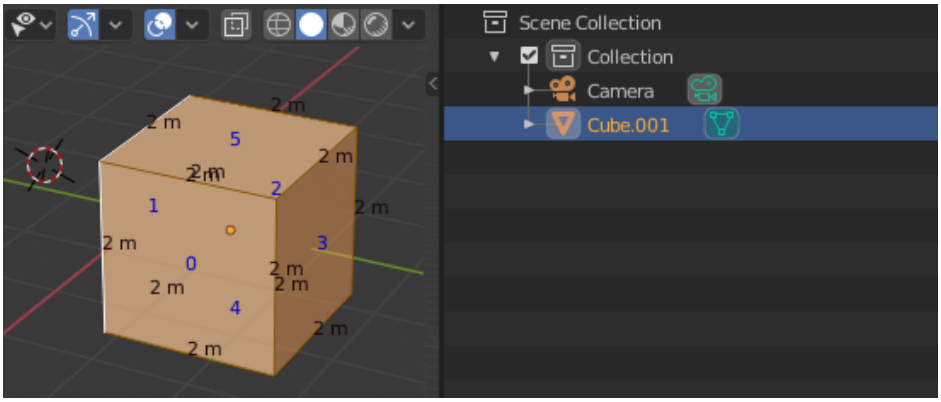

Figure 4 – Blender mesh and component identifiers

Similarly, when working with mesh components – vertices, edges and faces – we select them by specifying the numerical identifiers that are generated by *Blender*. These can be displayed on top of the corresponding elements, for easy visual identification. Figure 4 – left shows the blue numbers identifying the cube faces, each centered on the corresponding face.

Aditionally, *Blender* also has the option to display the lengths of edges, helping the user approximate distances when setting the values of the "increment" variable.

The currently implemented command set offers its users basic control and navigation capabilities within the scene. With some commands having multiple operating modes, each with different additional keywords or parameters, the user is only expected to know the name or keyword of a given command. Speaking it with the wrong arguments or not employing the proper command structure will prompt *Alexa* to present the user with a spoken example of how to formulate that command. This option is available when setting the "long" feedback option for *Alexa* and is in place as a command discovery and help mechanism for novice users.

At the moment, we have the functionality in place for a limited number of operations and transformations to be controlled through voice commands. However, given the structure of the command sets, they can be extended and integrated within the current hierarchy.

4. Experimental evaluation

This paper is intended to look into the feasibility of controlling 3D modelling tools, with the purpose of augmenting the traditional visual interfaces provided by such tools. Therefore, we set out to demonstrate the possibility of using the command set that we have implemented to create and edit 3D models. A use-case was defined, to be tested by two graduate students who volunteered for the task. Both had some basic working experience with *Blender*, having used it to import, export and do slight corrections on 3D models during their undergraduate studies.

Before setting the students to the task of completing our proposed use-case, we had them undergo two training sessions, in which they had to create two simple 3D objects (Figure 5 - left) while under our supervision. We had them undergo the training sessions two times: first using the voice commands (set to the "short-feedback" mode) that we developed and then using the graphical interface offered by *Blender* (with no shortcut keys allowed).

The proposed use-case required the participants to create a simplified model of a toy car (as shown in Figure 5 – right) using a cube and two cylinders as base primitives. The task was to be accomplished with the same operations that were employed during the training sessions. The use-case

required the final object to respect a set of clearly specified positions and dimensions, both when positioning the primitives and when executing operations such as face extension (extrude) and edge splitting (bevel).

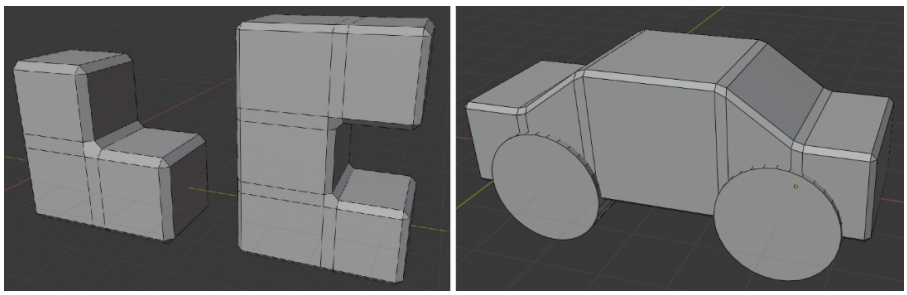

Figure 5 - 3D objects used for the training sessions (left) and for the proposed use case (right)

```
1  open blender application
2  add "cube" at 5,0,2
3  set increment "positive" 1.2
4  scale "x" by increment
5  edit mode
6  selection "face"
7  select "face" 2
8  set increment "positive" 0.7
9  extrude
10 extrude
11 select "face" 0
12 extrude
13 extrude
14 selection "face"
15 select "face" 12
16 set increment "negative" 0.8
17 move face by increment
18 select "face" 21
19 move face by increment
20 selection "edge"
21 set increment "positive" 0.2
22 bevel
23 add "cylinder" at 3.5,0,1
24 set increment "positive" 90.0
25 rotate "x" by increment
26 duplicate
27 move "x" to 6.5
```

Figure 6 - Use-case command listing

The commands listed within Figure 6 produce the results captured – in stages – within Figure 7. The first command is used to load Alexa's command set that we defined for controlling Blender. The second command creates a cube and places it at coordinates (5, 0, 2) – offset from the viewport origin. With the third command setting the global variable "increment" to 1.2, command number four does a scaling operation on the x axis, resulting in a laterally elongated cube - Figure 7 – 1.

Command number 5 activates the object-editing mode, allowing for the selection of various mesh components, such as vertices, edges or faces. Commands 6 and 7 set the selection mode to "face", displaying the identifier for each face of the cube, and then select face number 2 (right side of the cube) - Figure 7 – 2.

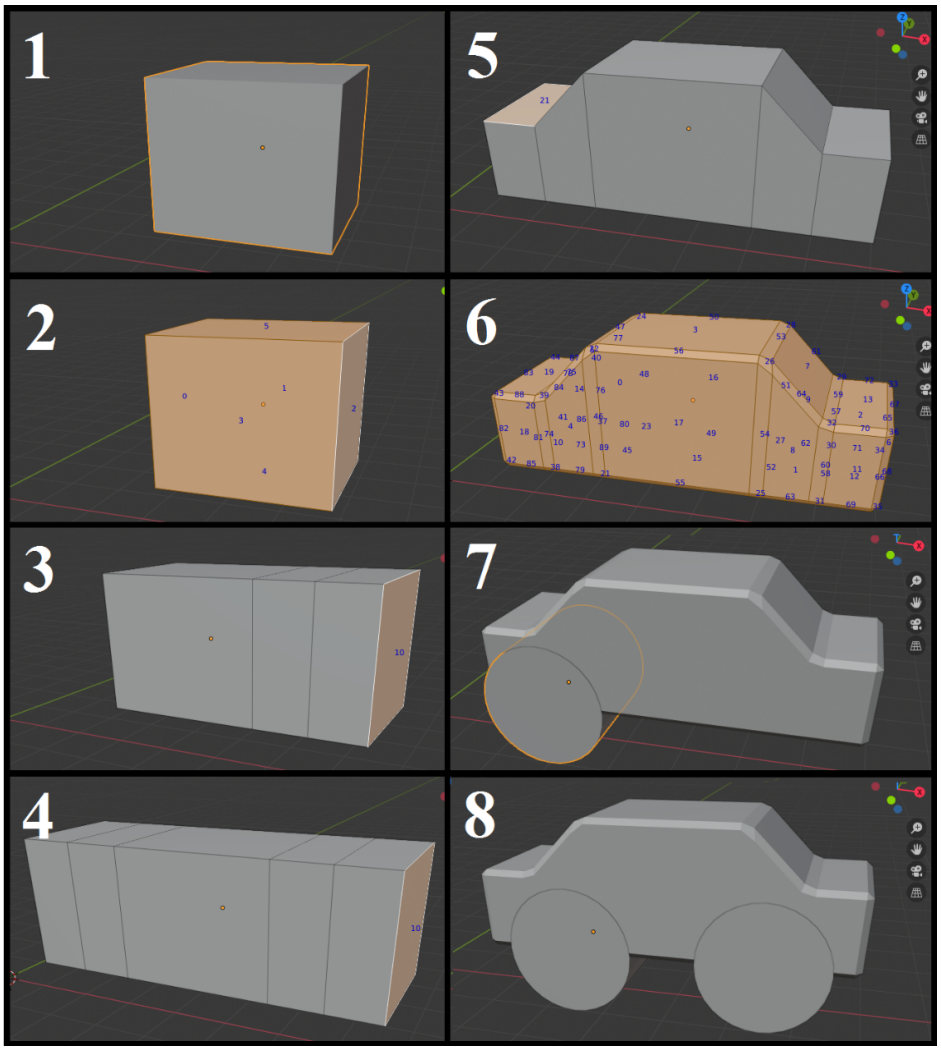

Figure 7 – The results (in stages) of the use-case commands

With face number 2 selected, command 8 sets the increment variable to 0.7, after which the next 2 commands execute two identical extrude operations. The extrude operation creates an extension of the currently selected element – in this case a face – and moves that extension with the amount set within the increment variable, along the direction of the normal.

The result can be seen in Figure 7 – 3. Commands 11 to 13 repeat the extrude operation on face number 0 (left side of the cube) with the same amount – as it is stored within the increment variable. The result is shown in Figure 7 – 4.

Command number 14 is optional and is used in this case only to highlight all the faces of the mesh, with their IDs, allowing the user to identify the faces that are to be modified next. Commands 15 to 19 select the two end faces at the top of the mesh – 12 and 21 – and move them downwards using an increment set to negative 0.8.

In the absence of a specified axis, the "move" operation translates the specified element along the direction of its normal. The transformation is identical for the two faces, since it was done using the same increment - Figure 7 – 5. When modelling with precision and transformation replication in mind, these types of commands could be more effective than manually inputing the numbers into a visual interface. This was one of the points that was also argued by some of the proponents of voice interaction within CAD tools - (Voice2CAD, 2020), (Xue, Kou, & Tan, 2009).

Command number 20 switches the selection mode to "edge", to prepare for a bevel (edge splitting) operation. The value set by command 21 is to be used as the distance between the new edges created after the split. Finally, command 22 does the bevel operation – as seen in Figure 7 – 6.

Command number 23 creates a cylinder at the location (3.5, 0, 1). When adding a primitive, the selection mode is automatically changed to "object mode" to allow for the handling of the new primitive. After adding the cylinder, the next two commands rotate it by 90 degrees along the X axis - Figure 7 – 7. Lastly, the currently selected primitive is duplicated (command 26) and the copy is moved along the X axis to location 6.5, as seen in Figure 7 – 8.

To detect potential improvements in the students' ability to use either interface, they had repeated the use-case three times for each of the two interfaces – first with the voice command set and then with the GUI. Figure 8 shows the average time (rounded to the nearest second) spent on each of the three attempts while creating the 3D model.

The chart shows comparable amounts of time spent by the participants using the two interfaces, with slightly better results obtained using the voice interface. Also, one can notice a trend of improvement in the case of both the graphical user interface (GUI) and the voice command set, with the former seeming to stall after the second attempt. We have found that the participants

seem to gain familiarity with the visual interface more quickly – as evidenced by the more sudden drop in time between attempts 1 and 2. However, this was offset by a degree of difficulty in adhering to exact positions and dimensions when using the graphical interface, since it required the user to type actual numbers into specific text fields. In this respect, dictating the numbers via voice commands seems to be more effective, as proven by the ability of the participants to obtain an even lower time when employing the voice commands during the final attempt.

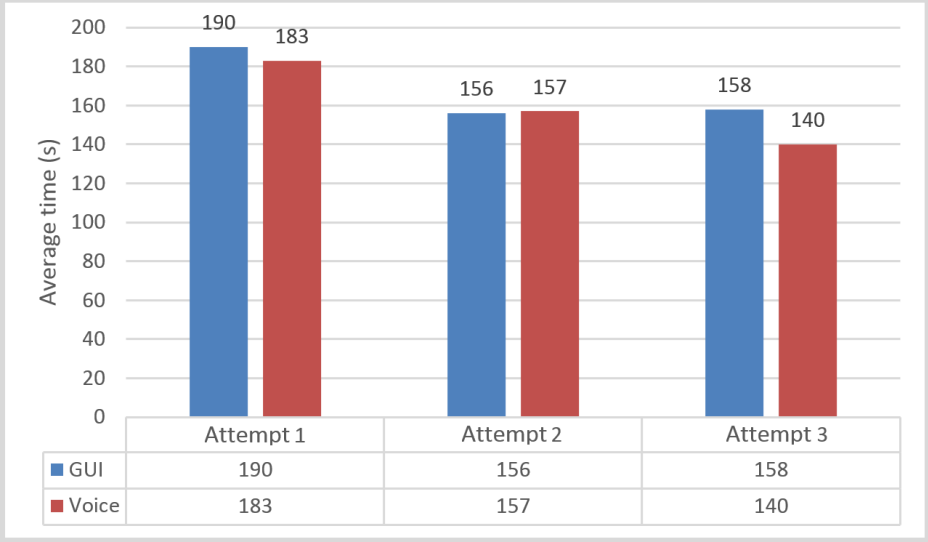

Figure 8 - Average time per use-case attempt

Overall, we have obtained some promising results, with the time required to create the object using voice commands being comparable, if slightly better, to that required by the exclusive use of the GUI. However, this has come at a cost of an average of 1.66 command-interpretation errors for a use-case of 27 voice commands, which equates to an error rate of about 6.1%. This shows us that there are still improvements to be made to both the voice recognition capabilities of the intelligent assistant and the combination of keywords/keyphrases employed witin our command grammar.

5. Conclusions

Throughout this paper we present a solution that we have implemented to allow users to control a 3D modelling tool with the help of voice commands. While we have provided basic control commands for navigating through the scene of objects, they are inadequate for effective operation, since they only allow for discrete parameter modification. One could still argue, however, that these commands might add to the tool's accessibility when it comes to users with motor disabilities, who cannot employ a mouse or a keyboard.

When it comes to model manipulation and editing operations, the tests show quite promising results, with the voice-based command interface slightly outperforming the graphical interface in terms of average time required for a 27-command use-case. On the other hand, when utilizing voice commands, users have to deal with an average rate of 6.1% command-interpretation errors, which indicates that the speech recognition technology of the personal assistants still has room for improvement. The same can be said about the structure and keywords of the currently implemented voice commands.

At this stage, we have tried to prove the feasibility of the voice interaction model in controlling basic 3D modelling operations. In future developments, we intend to explore its potential in handling more complex, composite command chains, and compare its overall effectiveness to that achieved through the graphical user interface.

Overall, we are of the opinion that, although voice-based interaction cannot fully replace the conventional mouse-and-keyboard control mechanism for 3D modelling, it can nevertheless bring significant improvements to the accessibility and usability of the cluttered graphical user interfaces featured by most modelling tools.

References

Amazon. (2020). Retrieved from Alexa Skills Kit: https://developer.amazon.com/en-US/alexa/alexa-skills-kit?

Amazon. (2020). *Alexa*. Retrieved from https://www.amazon.com/b?ie=UTF8&node=17934671011

Apple. (2020). *Siri*. Retrieved from https://www.apple.com/siri/

Blender. (2020). *Blender*. Retrieved from https://www.blender.org/

Davis, K. H., Biddulph, R., & Balashek, S. (1952). Automatic Recognition of Spoken Digits. *J. Acoust. Soc. Am., 24*(6), 627-642.

Deng, L., Hinton, G., & Kingsbury, B. (2013). New types of deep neural network learning

for speech recognition and related applications: An overview. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (pp. 85-99). doi:10.1109/ICASSP.2013.6639344

Dobesova, Z. (2012). Voice control of maps. *Proceedings of the 35th International Conference on Telecommunications and Signal Processing*, 460-464.

Gawari, H., & Bakuli, M. (2014). Voice and GPS Based Navigation System For Visually Impaired. *International Journal of Engineering Research and Applications, 4*(4), 48-51.

Google. (2020). *Google Assistant*. Retrieved from https://assistant.google.com/

Huang, H., Lin, C., & Cai, D. (2020). Enhancing the learning effect of virtual reality 3D modeling: a new model of learner's design collaboration and a comparison of its field system usability. *Univ Access Inf Soc*. doi:https://doi.org/10.1007/s10209-020-00750-7

Kou, X., Liu, X., & Tan, S. (2009). Quadtree Based Mouse Trajectory Analysis for Efficacy Evaluation of Voice-enabled CAD. *IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurements Systems.* doi:10.1109/VECIMS.2009.5068892

Microsoft. (1996). *Microsoft Office 97* . Retrieved from https://news.microsoft.com/1996/11/19/microsoft-office-97-released-to-manufacturing/

Microsoft. (2020). *Cortana*. Retrieved from https://www.microsoft.com/en-us/cortana

Puviarasi, R., Ramalingam, M., & Chinnavan, E. (2014). Self Assistive Technology for Disabled People – Voice Controlled Wheel Chair and Home Automation System. *IAES International Journal of Robotics and Automation, 3*(1). doi:10.11591/ijra.v3i1.3203

Rabiner, L. R., Levinson, S. E., Rosenberg, A. E., & Wilpon, J. G. (1979). Speaker Independen tRecognition of Isolated Words Using Clustering Techniques. *IEEE Trans. Acoustics, Speech and Signal Proc., Assp-27*, 336-349.

Sak, H., Senior, A., Rao, K., Beaufays, F., & Schalkwyk, J. (2015). *Google voice search: faster and more accurate* . Retrieved from https://ai.googleblog.com: https://ai.googleblog.com/2015/09/google-voice-search-faster-and-more.html

Stivers, T., & Sidnell, J. (2005). Introduction: Multimodal interaction. *Semiotica*(156), 1-20. doi:10.1515/semi.2005.2005.156.1

Voice2CAD. (2020). *Voice2CAD | Voice Activated Software*. Retrieved from https://voice2cad.com/how-does-it-work/

Withanage, P., Liyanage, T., Deeyakaduwe, N., Dias, E., & Thelijjagoda, S. (2018). Road Navi-gation System Using Automatic Speech Recognition (ASR) And Natural Language Processing (NLP). *Proceedings of the IEEE Region 10 Humanitarian Technology Conference*, (pp. 1-6).

Xue, S., Kou, X., & Tan, S. (2009). Natural Voice-Enabled CAD: Modeling via Natural Discourse. *Computer-Aided Design and Applications, 6*(1), 125-136. doi:10.3722/cadaps.2009.125-136.