A Graphical Interview Simulator

Aonghus McGovern
B.A. (Mod) Computer Science, Linguistics and French
Final Year Project
Supervisor: Dr. John Dingliana

April 4, 2010
Declaration

I hereby declare that this thesis is entirely my own work and that it has not been submitted as an exercise for a degree at any other university.

________________________________________  April 4, 2010
Aonghus McGovern
Abstract

An interview situation can be very stressful, and the best way to handle this stress is by practicing your technique beforehand. The aim of this project was to create an application that would graphically simulate an interview situation with a view to helping people hone their interview techniques. The system generates a room with two interviewers who then ask the user questions. The questions asked and/or the interviewers can be specified by the user themselves, should they wish to do so.
Acknowledgements

I would like to thank my supervisor Dr. John Dingliana for his help and support throughout this project. I would also like to thank my friends James and Lydia as well as my family for their support and encouragement. I am particularly grateful to *The Game Creators*, who developed Dark GDK, the graphics kit I used for this project.
# Contents

1 Introduction ................................................. 5  

2 Background .................................................. 6  
   2.1 Inspiration ............................................... 6  
      2.1.1 Serious Games ................................. 6  
      2.1.2 Presence ........................................ 9  
         2.1.2.1 Fear of Public Speaking ............... 9  
         2.1.2.2 Neuroscience Approach to Virtual Reality Experience ................... 10  
   2.2 Similar Applications ................................. 11  
      2.2.1 reed.co.uk Interview Simulator ............. 11  
      2.2.2 My Interview Simulator ....................... 12  
      2.2.3 Paiwastoon Interview Simulator .......... 13  
   2.3 Relevant Technology ................................. 14  
      2.3.1 Graphics API ................................... 14  
         2.3.1.1 Graphics APIs not chosen ............ 14  
         2.3.1.2 Dark GDK ................................... 16  
   2.4 Autodesk 3ds Max ................................... 17  
   2.5 DirectX file format ................................. 17  
   2.6 Microsoft SAPI ....................................... 17
5.2.3 Camera ......................................................... 33
5.2.4 Editability .................................................... 34
5.3 Problems ..................................................... 34
  5.3.1 Assumptions made with regard to Questions.txt and
       Scene.txt .................................................... 34
  5.3.2 Interactivity ............................................... 35
  5.3.3 Editing of Animations .................................... 35
  5.3.4 Question Times .......................................... 35
  5.3.5 Voices ..................................................... 36
      5.3.5.1 Potential superfluous creation of WAVE files . 37
5.4 Further Work ............................................... 37
5.5 Final Comments ............................................. 38
List of Figures

2.1 Against All Odds .................................................. 7
2.2 Balance of the Planet ........................................... 8
2.3 3rd World Farmer .................................................. 9
2.4 reed.co.uk Simulator .............................................. 12
2.5 My Interview Simulator ......................................... 13
2.6 Paiwastoon Interview Simulator ............................... 14

3.1 Front view of 3ds Max Biped ................................. 19
3.2 Side view of 3ds Max Biped .................................... 19
3.3 Front view of Wilson Mesh .................................... 20
3.4 Side view of Wilson Mesh ..................................... 21

4.1 Interview Scene .................................................... 27
Chapter 1

Introduction

This report describes an application that attempts to simulate an interview environment. The application was built from scratch using C++ and Dark GDK, a games development package designed by The Game Creators, a software company located in Lancashire, England. The application has been designed with editability in mind, thus users can specify their own questions and/or interviewer models.

The report will be structured as follows:

- Chapter 2 details the background to this project, as well as its inspiration and similar applications. Information on the various technologies used in its implementation is also given here.

- Chapter 3 explains the system’s design and the ways in which the user can modify it.

- Chapter 4 explains how the application was implemented as well as the problems that were encountered.

- Chapter 5 is the conclusion, and evaluates the success of the project as well as providing ideas for extending it.
Chapter 2

Background

2.1 Inspiration

2.1.1 Serious Games

One of the main sources of inspiration for this project is the Serious Games initiative. This movement tries to find 'uses for games in exploring management and leadership challenges facing the public sector.'[1] The initiative has two main areas of focus:

- Games for Health. This organisation was founded by the Serious Games Initiative in an effort to establish best-practice standards for applications that are being developed for the health care community. The Games for Health movement aims to form a partnership between: 1. Medical professionals and researchers and 2. Developers and users of game applications to 'create entire new ways of improving the management, quality, and provision of healthcare worldwide.'[2]

- Games for Change. This organisation aims to promote the use of video game techniques to address current issues in our society, including hu-
man rights, the environment and global conflict. They provide support, visibility and shared resources to individuals and organizations using and designing digital games for social change.[3] Example applications include:

- **Against All Odds.** This game focuses on 'the global refugee experience from the time people are forced to leave their countries of origin to the beginning of their new life abroad.'[4] A screen shot of this game can be seen in figure 2.1.

![Figure 2.1: Against All Odds](image)

- **Balance of the Planet**[5]. In this game, the player has to change society’s attitude to energy by means of funding and subsidies. A screen shot of this game can be seen in figure 2.2.

![Figure 2.2: Balance of the Planet](image)
Figure 2.2: Balance of the Planet

- 3rd World Farmer[6]. This game seeks to educate people about the difficulties of owning a farm in the 3rd World, including corruption, drought and armed conflict. A screen shot of this game can be seen in figure 2.3.
2.1.2 Presence

Another source of inspiration for this project is Presence (full name: Presence: Teleoperators and Virtual Environments), a journal published by the Massachusetts Institute of Technology (MIT) which discusses techniques for creating ‘believable’ virtual environments (VEs) i.e. environments which make an impact on the user and make them engage with the application.

2.1.2.1 Fear of Public Speaking

One paper published in this journal[7] investigated how people were affected when they presented a topic to a virtual audience of eight avatars. Subjects had to give talks to two of three different kinds of audience:
• positive audiences, i.e. an audience where the avatars nodded encouragingly, smiled often etc.

• negative audiences, i.e. an audience where the avatars slouched in their chairs, fell asleep etc.

• neutral audiences, i.e. an audience where the avatars remained static throughout the talk.

By monitoring the participants’ heart rates, Pertaub et al. found that their Virtual Environment generated a strong emotional response. This shows that virtual environments can illicit an emotional response from a human being. This particular paper is important because the seminar situation it describes is not massively different from an interview situation. Also, as can be seen on reading the paper, the models used are not of very high quality. This suggests that, for the purposes of this application, the 3D models used do not have to be of the quality observed in many modern games, but simply good enough to be recognised as approximations of human beings. Indeed, as Pertaub et al. put it: ‘In spite of the fact that the avatars had low representational quality, with pre-programmed behaviors independent of the actual behavior of the speaker, people still responded according to type.’

2.1.2.2 Neuroscience Approach to Virtual Reality Experience

Rey et al. also performed an experiment to investigate the effects Virtual Environments could have on humans[8]. For their experiment, subjects navigated a virtual environment under two conditions:

• 1. Manual, where they were allowed to navigate the environment themselves.

• 2. Automatic, where they watched as the scene was navigated for them.
By using Transcranial Doppler sonography (measuring blood flow in main cerebral vessels) to analyse brain activity, Rey et al. found that exposure to the VE affected a response from the participants but that this effect was most pronounced in the active navigation condition. This suggests that when a person is presented with a virtual environment they engage with it to a certain degree. However, when they are given control over how they move around it, they begin to immerse themselves in it.

2.2 Similar Applications

As one might expect, given the difficulty many people experience with their interview techniques, there are many interview simulators currently available. Some of these are described below.

2.2.1 reed.co.uk Interview Simulator

reed.co.uk is the UK’s biggest job site. Its interview simulator[9] generates random questions when the user presses the ‘Next Question’ button. Its question bank is quite comprehensive, covering a range of topics. However, it is not graphical. Also, its questions are not read aloud, merely printed to the screen. A screen shot can be seen in figure 2.4:
2.2.2 My Interview Simulator

This simulator[10] was created by Arabian Morgan Enterprises, Inc. It allows users to practice answering questions and gives them advice on how to do so. It also allows the user to run a simulation that asks the user questions as if they were in an actual interview. However, it does not use 3D graphics, and merely displays pictures alongside the questions asked. Also, the user is given as long as they want to answer a question; the interview only moves forward when they click on the next question. This somewhat detracts from the interview experience, as it gives the user a certain degree of control that would not be given to them in an actual interview. A screen shot can be
seen in figure 2.5:

Figure 2.5: My Interview Simulator

2.2.3 Paiwastoon Interview Simulator

This interview simulator[11], developed by Paiwastoon Networking Services, walks the user through the entire interview experience, from deciding on what clothes to wear to the telephone call informing the user as to whether or not they got the job. The application poses questions and the user chooses their response from three options. The questions are read aloud, which contributes to the realism of the experience. However, it does not use 3D graphics either, and instead displays pictures with the questions. Also, there is no
randomisation of questions i.e. the user is asked the same questions each time they run the application. A screen shot can be seen in figure 2.6:

![Figure 2.6: Paiwastoon Interview Simulator](image)

2.3 Relevant Technology

2.3.1 Graphics API

There are many graphics APIs available for free download. Thus, it was necessary to carefully evaluate a number of them before deciding on one.

2.3.1.1 Graphics APIs not chosen

The following Graphics APIs were evaluated and not chosen.
• **DIVE**

DIVE (Distributed Interactive Virtual Environment)\[12\] is an online 3D environment through which players navigate while interacting with other players and applications. Pertaub et al. used DIVE in their experiment, described in Section 2.1.2.1. There are three main reasons it was not chosen for this project:

- DIVE models are not of very high quality. There are a variety of free graphics packages that allow the manipulation of better 3D models than are available in DIVE.

- Pertaub et al. state that ‘significant customisation was required, both to the geometry and fitting of texture maps’ when using DIVE. The author felt that the time spent learning how to do this and then actually doing it was not worth the result, considering what other graphics packages offer for significantly less time and effort.

- DIVE’s objects’ behaviours are described by Tcl scripts. The author is not familiar with the Tcl language and felt that the time spent learning it would take away from development time, particularly given that the two previous problems meant he could not guarantee he would be using DIVE to implement the project.

• **OGRE**

OGRE (Object-Oriented Graphics Rendering Engine)\[13\] is a 3D engine written in C++. The main reason it was not chosen is that it is not designed for game development. OGRE is designed to generate 3D environments, not games. This means that it would have to be integrated with a games engine, and also a sound library, to create the interview application. Doing this would have taken away from the time that could be spent developing the low-level aspects of the application.
• **Processing** Processing[14] is a language and environment that allows people to draw objects and create animations. The main reason it was not chosen is that it is more of a prototyping engine than anything else. Indeed, it is described on its homepage as being a ‘software sketchbook’. Thus, the author felt that it did not offer the capabilities he would require to implement the project.

### 2.3.1.2 Dark GDK

Dark GDK is a graphics kit developed by *The Game Creators*, a software company based in Lancashire, England. The main reasons it was chosen are:

- It was created specifically for game design.

- Its intuitive commands make it very easy to learn. For example, the command that positions an object in a scene is `dbPositionObject()`, the command to play an object’s animations is `dbPlayObject()` etc.

- It is designed specifically for use with Microsoft Visual C++ 2008, the editor used when implementing the project.

- It provides detailed documentation and tutorials which explain exactly how to use it.

- It allows people to load and manipulate 3D models that they have created themselves.

- It has a large online community which means that questions that cannot be answered by the documentation can be posed on online forums.
2.4 Autodesk 3ds Max

Autodesk 3ds Max is a program developed by Autodesk Inc. that allows users to create many different kinds of 3D models. It also allows users to create animations for these models. It has been used in the production of many films, including 2012, Black Hawk Down and Spiderman 3. The full capabilities of 3ds Max (lighting, physics etc.) are vast and, as most of them were not used in the creation of this project, they will not be detailed here.

2.5 DirectX file format

DirectX is an architecture- and context-free file format. It exists within the Microsoft DirectX framework for Windows operating systems and is dedicated to handling multimedia tasks, particularly gaming and video. It allows very good models to be developed and is supported by Dark GDK, making it ideal for use in this application. In order to convert the 3ds Max models to the DirectX file format, I used the Panda DirectX Exporter[15].

2.6 Microsoft SAPI

The Microsoft Speech Application Programming Interface (SAPI) is an API that was developed by Microsoft to enable speech processing within Windows applications. It allows programmers to 'speak' lines of text into WAVE\(^1\) files. Dark GDK can then load these WAVE files and play them back within an application.

\(^1\)The WAVE(Waveform Audio File Format) is a Microsoft and IBM audio file that stores an audio bit stream. Its file extension is .wav.
Chapter 3
Design

There are three main aspects to this application:

- Graphics i.e. the interviewers and the room.
- Sounds i.e. having a voice speak the questions.
- Interaction between the user and the application.

This chapter explains each of these, as well as the editability of the system. A comparison is also drawn between the techniques used in this application and those used to create video games.

3.1 Graphics

3.1.1 Interviewers

The creation of the characters involved 3 steps, detailed below.

3.1.1.1 The 3ds Max Biped

According to the 3ds Max documentation, the biped is ‘a linked hierarchy of objects that by default resemble those of a human’. The root object
in this hierarchy is the biped’s Center of Mass (COM), located near the center of the its pelvis. The biped’s structure allows for its fingers, toes, arms, legs, head and torso to be moved in a way that provides a very good approximation of the movement of the human skeleton, making it very useful for this application. Front and side views of the biped can be seen in figures 3.1 and 3.2, respectively:

Figure 3.1: Front view of 3ds Max Biped

Figure 3.2: Side view of 3ds Max Biped
3.1.1.2 Skinning

Obviously, the biped on its own is not enough. Having a multi-coloured skeleton ask the user questions will certainly detract from the reality of the experience! Thus, the biped needs to be given a human exoskeleton. This involves placing a 3D mesh around the biped by means of a process known as skinning. The mesh used in this application comes with Autodesk 3ds Max 2010, and is called Wilson. Front and side views of Wilson can be seen in figures 3.3 and 3.4 respectively:

![Figure 3.3: Front view of Wilson Mesh](image)
The skinning process is somewhat complicated but, in simplified terms, it involves:

- Positioning the biped inside the mesh and then positioning its limbs so that they lie within the center of the corresponding mesh limbs e.g. the biped’s arm must be in the middle of the mesh arm.

- Posing the biped so that it fits the Wilson mesh.

- Attaching the mesh to the biped. This is done by applying the *Physique modifier*\(^2\) to the biped. The Physique modifier makes it possible for the mesh to be controlled by the biped’s motion by assigning each mesh vertex to one or more bones on the biped.

- Adjusting the biped’s envelopes so that the mesh moves correctly.

We then animate the biped and when it moves, the mesh moves with it. Explanations on posing the biped and adjusting its envelopes can be found in section 4.

\(^{2}\)3ds Max modifiers allow the user to change geometry and/or properties of an object.
3.1.1.3 Animations

Once the biped has been skinned, it is ready to be animated. Creating animations in Autodesk 3ds Max is a relatively simple process. The first step is posing the biped in a starting position and setting a frame, which essentially saves that position. The next step is to pose the biped in a final position, and set another frame. 3ds Max then fills in the motions that take the biped from the starting position to the final position.

Five animations were created for each biped:

- An ask animation, to be played when the interviewer is asking a question. This animation does not contain a mouth movement, as the Wilson model does not have movable lips. Instead, it consists of an arm movement and a turning of the head towards the camera.

- A head shake animation.

- A head nod animation.

- A mid-question animation. This animation consists of the interviewer looking at their watch, or indeed the place where their watch should be (The Wilson model unfortunately does not have a watch).

- A default animation. This animation is looped while the interviewer does not have anything to do, i.e. they are not asking a question, performing a mid-question animation or performing a reaction animation (i.e. either a head nod or a head shake).

3.1.2 Scene

The interviewers’ chairs were not modeled separately, and were instead included with the interviewers themselves. The room is not very detailed, merely five planes (a plane is essentially a four-sided shape with no volume):
one with a carpet texture\textsuperscript{3} applied to it (the floor) and four with wood paneling textures applied to them (the roof and the left, right and back walls). The user can move the camera left, right, up and down by moving the mouse.

### 3.2 Interaction

There were some features the author wanted to include but which had to be left out due to time constraints. These were:

- An eye tracker to monitor the user’s eye movements to make sure they kept their eyes on the interviewers and didn’t look at the floor or the roof too much etc.

- A speech processing component that would record the user’s voice and then perform analysis on it. This would be useful as it would allow the program to determine whether the user had waited too long to answer a question or if they had paused for an extended period of time during an answer, in which case, the program would move on to the next question.

### 3.3 Editability

As has been stated previously in this report, one very important feature of this application is editability. With this in mind, the system has been designed so that the user can specify information relating to both the questions that are asked and the interviewer and room models. The question information is specified in a file named Questions.txt. The format for specifying the questions is as follows:

\textsuperscript{3}Texturing is the process of placing images onto the surface of objects. In this case, an image of a carpet was placed on the surface of the plane.
\-qText/-q. 'qText' is the question text.

\-startTime/t. 'startTime' is the time at which the question is to be asked.

\-rReaction/r. 'Reaction' is the reaction the interviewer will give at the end of the time allocated for the question to be answered. The options are 'shake' and 'nod', which correspond to a shake and a nod of the head, respectively. This feature was included to allow for the creation of a positive or negative atmosphere for the user. For example, if a negative atmosphere were required, the interviewers could be made to shake their head after each question. Conversely, if we wanted a positive atmosphere, the interviewers could be made to nod their head after each question. It is of course possible to have an interviewer nod their head after some answers and shake their head after others.

\-dDuration/d. 'Duration' is the length of time the user is given to answer the question.

\-mMid/m. 'Mid' is a mid-question animation. This line is optional.

\-imidQTime/i. 'midQTime' specifies how far into the question the mid-question animation will be played, if there is one. This line should only appear in the file if the previous line also appears.

\-e. Marks the end of a question description.

The interviewer and room information is specified in the interviewers.txt file. Users can modify animation information for the five animations mentioned in Section 3.1.3. The format for specifying this information is as follows:

\-fFilePath/f. 'filePath' is a file path for an interviewer.
3.3.1 Caveat

Certainly, the average user will not have the expertise required to create their own models for use within this application. However, it is possible for one person with the required knowledge to create their own models, load them into the application and then distribute the application to as many people as they wish.

3.4 Voices

The voices are generated using the Microsoft SAPI, mentioned in Chapter 2. Each time the application is run, the Questions.txt file is processed and the text for each question is spoken into a WAVE file. The file name for each WAVE file is written to a text file filePaths.txt. The application then reads filePaths.txt, loads the sound files described by the file names and plays them at the appropriate time.
Chapter 4

Implementation

This application is written entirely in C++. It has been tested on Microsoft Windows but unfortunately, there was not enough time to test it on any other systems (Linux, Mac OS X etc.).

4.1 Interface

The scene is very simple. It consists of a room with two interviewers, who ask the user questions at intervals based on the values in the Questions.txt file. The text for each question remains on the screen for the length of time allocated to that particular question. If the interviewers do not have a specific animation to perform e.g. asking a question, nodding their head etc. they revert to their default animation. The user can move the camera left and right, up and down. If a user has finished answering a question or decides they do not want to answer it, they can skip ahead to the next one by pressing the right arrow key. If there are no more questions, pressing the right arrow key will cause the program to close.

A screen shot is given in figure 4.1.
Figure 4.1: Interview Scene

Tell us a bit about yourself
4.2 Challenges

The major difficulties encountered in the implementation of this project are detailed here.

4.2.1 Allowing the user to skip ahead

In order to allow the user to specify question times and durations, as well as mid-question animations, a method for keeping time was required. This was done by using the timer that comes with Dark GDK. When the application starts, a variable, `start`, records the time. As the program is running, another variable, `current`, stores the current value of the timer (updated every millisecond). A third variable, `duration` is used to track how long the interview has been running. The value for `duration` is calculated by subtracting `start` from `current`. In order to skip forward, it was necessary to add another variable, `timeIncrement` to `duration`. This variable is 0 until the user presses the right arrow key. Every time the user presses the right arrow key, it is made to equal the difference between `duration` and the start time of the next question minus 3000. Thus, when the right arrow key is pressed, the value of `duration` is increased to a value very close to the start time of the next question. Problems arose when a question was skipped during an animation such as a head shake or the asking of a question. If this happened, the interviewer in question would completely stop, i.e. they would not revert to their default animation. To stop this from happening, the user is not allowed to skip questions during the ask, mid-question or reaction animations (head shake or head nod).
4.2.2 File Processing

The Dark GDK file reader adds standard return characters to the end of every string it reads. There is a good reason for this. In C++, if strings are not properly terminated a problem called buffer overrun can occur. This essentially means that the program is trying to access elements in an array beyond the memory range allocated to that array. This can have many severe consequences and must be avoided. Important though the addition of these characters therefore is, it makes it difficult to perform operations using data read from files. For example, if we were to compare ’ask’ as read by the Dark GDK file reader with a string we had instantiated and whose value we set to ’ask’ we would in fact be comparing ’ask\r’ with ’ask’ which would return false. The characters cannot be replaced with the backspace character, because even though we will only see ’ask’, the C++ compiler will see ’ask\b’. This means that the strings read by the Dark GDK file reader must be processed in some way.

4.2.2.1 Questions.txt

For the animation descriptions, a function, contain, was written by the author to check if a particular string contains another. This means that a string read by the Dark GDK file reader can be tested to see if it contains, for example ’ask’. For the question times, a function, cleanup, also written by the author, was written. This function replaces all alphanumeric characters with spaces, making it possible to use C++’s strtod function to convert strings to doubles. This is very important because if we are to allow the user to enter question times as strings we need a way to convert these strings to a numeric representation.
4.2.2.2 Interviewers.txt

For the file paths, it is not enough to simply replace the terminating characters with spaces; the paths had to be exactly right. Thus, a function was written by the author called cleanupFile which takes a string and returns a new string that contains all of the information from the original minus the terminating ASCII characters. For the animation frames, cleanup was used to replace the terminating ASCII characters with spaces and the new strings were then converted to a numeric representation.

4.2.3 Skinning

The skinning process was mentioned briefly in Section 3.1.1.2. Certain challenges had to be overcome in order to successfully skin the biped.

4.2.3.1 Posing the biped

In order for the biped animations to properly affect the Wilson mesh, the biped had to be 'posed', i.e. its proportions had to be changed so that it would better fit the mesh. This involved increasing the length and width of the biped’s fingers, arms, legs and feet. It was also necessary to alter the shape of the biped’s head so that its chin lined up with the Wilson Mesh chin.

4.2.3.2 Adjusting the biped’s envelopes

The envelope for each biped part is essentially the area that is affected by its movement. If the envelopes are not of the correct size, the mesh will not 'deform’ (i.e. move) properly. For example, if the envelope for the arm is too narrow, when the biped arm moves, the t-shirt sleeve will not move with it. The envelopes for the biped’s arms, legs and torso had to be adjusted so that the corresponding parts of the mesh would deform correctly.
4.2.4 Screen size

Unfortunately, the application window is not full screen. The window dimensions can be adjusted using the Dark GDK \textit{dbSetDisplayMode} function but the graphics card on the computer used for the development of the application is not powerful enough to display the scene at the dimensions required to generate a full screen image.

4.3 Operation

This program is very easy to use. The only potential difficulty with editing the questions and/or the scene is learning the markup language, and that difficulty is not great. Problems may arise with the creation of new models for use in the application as this process is not easy. However, this is not a fault of the system itself.
Chapter 5

Conclusion

Due to time constraints, the addition of features to this project was on a ‘must-have’ basis. This unfortunately meant that certain features, while desired, had to be left out. This chapter discusses the overall success of this project, as well as ways in which it could be improved and expanded.

5.1 Realism vs. Immersiveness

Certainly, the scene is not a very realistic one. However, as can be seen from the paper by Pertaub et al. that I mentioned in Section 2 of this paper[7], one does not need realism to get an emotional response from the user. Indeed, very few video games can claim to create virtual environments that are ‘realistic’, in the true sense of the word. Rather, they aim to be immersive. That is to say, they try to make the user engage with the application, even though they know it is not real. This point is well made by Rey et al.[8]: ‘VEs take advantage of peoples imaginative ability to psychologically transport their presence to another place.’
5.2 Similarity to Video Games

Some of the techniques used in the implementation this application were inspired by modern video games.

5.2.1 Animations

Giving the interviewers a default animation is very similar to the approach adopted in the Half Life video games. These are first person shooter (FPS)\textsuperscript{4} games, developed by \textit{Valve}. At several points in the game, the user has to interact with characters. These characters, when not performing specific animations (e.g. asking a question, moving an object etc.), revert to a default animation which is looped over and over, giving the impression that they are alive. Similarly, in the Interview Simulator described in this report, when the interviewers are not performing specific animations (e.g. asking a question, reacting to an answer etc.) they too revert to a default animation.

5.2.2 Voices

The use of sound files in this application is very similar to almost every video game on the market. There are very few video games that do not have some kind of sound. Indeed, the way in which sound files are used for speech in this application is analogous to the way in which they are used in video games that have interaction between the user and characters within the game.

5.2.3 Camera

The camera view is first-person, identical to FPS video games.

\textsuperscript{4}Games where the camera angle is from the character’s eyes and the user navigates a scene, shooting enemies
5.2.4 Editability

The possibility for users to put their own interviewer models into the game is similar to the functionality provided by the Quake video games, designed by id Software. These games are also of the FPS variety, and can be heavily modified by players. In fact, it is possible to modify the game so much that it no longer represents the original. In 1996 Robin Walker, John Cook and Ian Caughley modified Quake to create a new game, Team Fortress.

As it is possible for the user to load their own interviewers and scene into the Interview Simulator, it is possible to create an application that looks entirely different to the one presented here.

5.3 Problems

This project functions quite well but is not as complete as it could be. The following section details where improvements could be made.

5.3.1 Assumptions made with regard to Questions.txt and Scene.txt

It is assumed that the user will enter the information correctly for both of these files. For example, it is assumed that the user will not enter two questions that start at the same time or two questions that overlap each other, and that they will not enter the same start or end frame for two different animations. This assumption was made because the time that would be required to make absolutely certain the user entered the information correctly would have taken away from the time spent on the general development of the application.
5.3.2 Interactivity

There are many ways I would have liked to integrate interactivity into the application, including voice processing to tell if the user was speaking too loudly or too quietly and an eye tracker to make sure the user kept their eyes focused on the interviewers, but there was simply no time to implement these features. However, there is some implicit interactivity in the application. The user will react to being questioned i.e. by giving an answer, and may also respond to the interviewers’ reactions to their answers. The fact that the interviewers move constantly during the period the user is answering a question could also cause the user to react. Mid-question animations are another way in which the application can cause the user to engage themselves. Thus, while the application doesn’t provide a huge amount of physical interactivity, the author would argue that it does enough to get a response from the user.

5.3.3 Editing of Animations

Users cannot enter animations other than the five that are described in Section 3.1.3. The reason for this is that allowing a user to enter specifications for any animation they like would require a large amount of coding. Indeed, it would nearly be a project unto itself. Thus, users can only enter new frames for the animations that are already described in ’Scene.txt’.

5.3.4 Question Times

The times for the questions and mid-question animations must be specified in milliseconds, the format of the Dark GDK timer. The reason for this is that there was no time to write code that would convert some other time representation to milliseconds.
5.3.5 Voices

The application is written to automatically generate WAVE files for each question. However, a problem arose with the voice used. Microsoft SAPI uses the operating system’s default voice which, in Microsoft Vista (the operating system on which the project was implemented), is Microsoft Anna, a female voice. Unfortunately, Anna is the only voice that comes with Windows Vista. The supervisor for this project, Dr. John Dingliana, was kind enough to provide another voice, named Lawrence, distributed by Cepstral. However, the SAPI function that is needed to change voices from Anna to Lawrence, \textit{SpFindBestToken}, which works fine in a normal C++ application, causes a compiler error in a Dark GDK project. After extensive searching through the Dark GDK forums and the Web in general, the author was unable to find a solution to this problem. This meant that a separate application, 'Speak Test', had to be written to read in the \textit{Questions.txt} file, change the voice from Anna to Lawrence, generate the WAVE files and write the file names to a file, \textit{filePathsMale.txt}. This affects how line 257 of the Interview Simulator source code (which specifies the file from which the application will read the file paths for the sounds) should be written. On Windows systems where the default voice is male, the line should read:

\begin{verbatim}
dbOpenToRead(3,"filePaths.txt");
\end{verbatim}

This means that the file names for the sounds will be read from \textit{filePaths.txt}, the file that is created by the main application. However, on Windows systems where the default voice is female, the 'Speak Test' application must be run before the Interview Simulator is run. In this case, line 257 of the Interview Simulator should read:

\begin{verbatim}
dbOpenToRead(3,"filePathsMale.txt");
\end{verbatim}
This means that the file names for the sounds will be read from `filePaths-Male.txt`, the file generated by the 'Speak Test' application. Note however, that changes to the source code are not strictly necessary. At the moment, the Interview Simulator is designed to use the sound files generated by the 'Speak Test' application. This means that as long as the user is willing to first run the 'Speak Test' application, they will be able to run the Interview Simulator, with appropriate voice files, without modifying the source code. As the 'Speak Test' application takes mere seconds to complete, it could be argued that the additional effort involved in running it as well as the Interview Simulator is minimal.

5.3.5.1 Potential superfluous creation of WAVE files

The application generates WAVE files for the questions in `Questions.txt` each time it is run. Thus, if it is run twice using an un-modified `Questions.txt` file, it will generate WAV files that already exist. Indeed, in the case where the main application uses the files generated by the 'Speak Test' application, it will generate WAVE files that it will not even use. Ideally this should be avoided to reduce computation time.

5.4 Further Work

There is a great deal of scope for future work on this project. Situations where one person is questioned by a group are quite numerous. This project could even be expanded to include scenes involving more than 2 questioners e.g. a seminar, a courtroom setting, a disciplinary hearing, a press conference etc.
5.5 Final Comments

This report describes an interview application that presents the user with a scene, in which animated characters ask them questions. Although the 3D models I used are not magnificent, they are certainly recognisable as human beings. Also, the structure of this application means that users can create and animate better 3D models and use them in the application.

The learning gain from this project has been large. I have learned the basics of animation, become familiar with Autodesk 3ds Max and drastically increased my programming skills, both in terms of the writing of algorithms and in C++. I was also introduced to the Microsoft text-to-speech system, Microsoft SAPI. This project has also been my first experience with working at length on a single application and required me to adapt the way I had previously thought about programming.

All in all, this project required me to deal with a great number of things I had previously taken for granted.
Bibliography


