# How to use abdnthesis.cls

*Timothy J. Norman*

A dissertation submitted in partial fulfilment
of the requirements for the degree of
**Doctor of Philosophy**
of the
**University of Aberdeen**.


Department of Computing Science

2010

# Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

Signed:

Date: 2010

# Abstract

An expansion of the title and contraction of the thesis.

# Acknowledgements

Much stuff borrowed from elsewhere

# Contents

## Chapter 1

# Introduction

The game industry now is bigger than ever before and still growing. Along with technological advancements as well as rise in popularity, games themselves become bigger and more polished. With increasing size and quality, the number of man-hours rises drastically. A lot of work is being put into creating tools that enable faster creation of content. However, there is still a lot left to be optimized and automated.

One domain of game development that suffers that drains lots of man-hours into monotonous processes that could potentially be automated is animation. Most games will rarely animate everything by hand as there is too much content to cover. While some animated content needs to be very polished - usually called cutscenes (action sequences, parts of game that greatly influence the plot development), some animation might be cruder (dialogue sequences). Games like RPGs will feature a lot of dialogue - during the dialogue the characters cannot stand still as it would negatively impact player's immersion in the game world. The characters must move and perform gestures that naturally underline their speech. These animations cannot be all done by hand because of the sheer amount of content needed. For instance:

- Mass Effect Andromeda and Fallout: New Vegas features 65,000 lines of dialogue[1][2].
- The Witcher 3 features roughly 35 hours of dialogues[3].

The main challenge is to make the dialogue scenes (and other automatically generated animation) look indistinguishable from cutscenes. In many games, player will be shown good, well-polished animation that immediately switches to poor, clunky and unrealistic animation. The less perfected dialogue scenes break the immersion of the player and negatively impact the overall experience. Easier, faster and better quality methods of generating dialogue scenes would be a great asset to the gaming industry.

## 1.1 Motivation

The purpose of this project is to develop a tool that helps generate animated dialogue scenes and minimizes the amount of manual work by using natural language processing. Generating the scenes directly from script would pose several benefits:

[1] https://www.pcgamer.com/mass-effect-andromeda-has-over-1200-speaking-characters/
[2] https://www.pcinvasion.com/fallout-new-vegas-will-have-65000-lines-of-dialogue
[3] https://www.pcgamer.com/most-of-the-witcher-3s-dialogue-scenes-was-animated-by-an-algorithm/

- The script is written to outline the plot of the game. The same script could be fed into a program to generate the animations.
- The script is semi-structured natural language. By allowing natural language, the program helps shorten the gap between artists and writers and animators and technicians.
- The program can be used for prototyping scenes quickly and easily.
- The program can be used by people who know nothing about animation.

The program I propose would create prototype animation with almost no amount of work required. This can be use to either save time or to use the saved time to manually adjust the animations.

## 1.2 Objectives

The Projects Objectives are as following:

**Develop a tool able to interpret a natural language script**

The tool must be able to read a semi-structured script and recognize dialogue lines, emotions of the characters and actions performed by them.

**Develop a tool able to blend a final dialogue scene**

The tool must be able to output a fully editable dialogue scene. The scene is assembled using pre-made motion capture clips.

The scenes created by the software will be very crude and unpolished. Scenes generated by the tool will need to be polished manually, the amount of polish can be decided by assessing the importance of a given scene. However there is a chance that the scene quality will be much inferior to scenes generated by similar tools that use different approaches. Therefore an important question this project tries to answer is whether taking the NLP approach to animation is feasible in the games industry given current technology.

## Chapter 2

# Background and Related Work

## 2.1 Background

Manually crafting every animation in the game is unrealistic due to cost and time requirements. Many games have employed various approaches to computer generated animation in order to generate hours of realistic content. No game however has succeded in making the dialogue animation indistinguishable from manually animated cutscenes.

## 2.2 Existing Systems

There has been a variety of approaches featured in games. Many of them ended up generating dialogue scenes that are higly repetitive, not very realistic and in general not mathing the sentiment and emotion of the speech with movement. The only system that did not seek to find cheap workarounds around the issue and instead embraced the full complexity of the problem is the dialogue sene generator used in The Witcher 3. The system developed by CD Projekt Red made computer generated dialogue sequences in many cases barely distinguishable from those made by an artist, allowing less important scenes to be left completely untouched by a human animator. [6]

The tool created by CD Projekt Red takes information on initial state of involved characters (position, stance, emotions, etc.) and audio recording of the dialogue lines. The tool chooses matching premade animated clips and outputs a fully editable animated scene of characters conversing with one another. The tool uses audio recordings to aid the animated clips (analyzing the audio waves may help decide when characters accent or underline some information). This tool is the current state-of-the-art and has produced the best effects in terms of amount of work to quality of animation ration. However, there are some serious drawbacks to this project. It still requires a fair amount of work as for every scene the initial state must be specified manually. Moveover, the system requires audio to be recorded first. Moreover, the tool is not released to be used commercially outside CD Projekt. [14]

The tool I propose would hardly be able to compete with that of CD Projekt Red, however it would have some significant advantages. It would make generating the scenes even faster (requiring less manual work and preparation) and would in general be more appealing to small developers and people who are not animation experts.

## 2.3 Related Work

The main focus of this project is the usage of NLP for generating animated sequences. While this project puts particular emphasis on generating scenes of dialogue, there exist a multitude of

projects that explores the usage of NLP in animation in a variety of ways.

A very early research (1991) explores usage of NLP for creating animations that would help engineers demonstrate tasks in an easy and safe way (demonstrating tasks personally might be unsafe, reading manuals might be insufficient to understand the task in full) [9]. The system would take as input a set of natural language *directives* or *commands* (e.g. *move cup to table*). The system would interpret such an instruction into a series of steps (tasks) that are carried out in a given order. Based upon that sequence an animation would be generated.

The project hovewer seems to have a few significant problems. Most importantly, the end results was not editable. In my research I believe that the end results will not be immediately satisfactory without any manual improvements and I believe that the outputted scenes should be fully editable. The other issue with this project is that the end result is not realistic or immersive (this was not a priority of that research, but is important for me). The animations were automatically generated in full, which I do not believe to be a viable approach for my project. To improve realism of the scenes, the animation should be created using motion capture clips.

A lot more research has been done in this area with certain degree of success. Another interesting paper explores a topic more similar to the focus of this report. `GeneratingAnimationfromNaturalLangua` proposes using a database of pre-recorder motion data instead of filly generating the movements [12]. The paper argues that motion synthesis requires too much manual setup (which a normal user might find too hard). The paper also proposes an architecture of a database - a way to store motion capture clips and their associated metadata. The system described in this paper is however essentially different from what I propose in this report, as the described report focuses on action-driven animation, while my tool would focus on dialogue-driven animation.

While many other reasearch focused on animation in general, some reasearch focused on usage on NLP generated animation for use in games specifically. It aimed to create a parser that would transform natural language text to a set of instructions for the animation layer. The end goal was realtime character control by natural language commands. The animations were not intended to increase realism, but rather to accomplish goals in a strategic manner. The research was generally succesful proving the potential usefulness of NLP in game development. [3]

So far, the main focus of similar research has been generating animated scenes from text with focus on actions, often by employing motion synthesis. There exists no record of using sentiment analysis to create animated dialogue scenes.

## 2.4 Emotion Analysis

The task of emotion analysis is a subset of natural language processing. The task of analysing natural language text in search of subjective information such as sentiment or emotion is known as sentiment analysis.

### 2.4.1 Sentiment Analysis

Sentiment Analysis can be broadly defined as a computational approach for discovering opinions and attitudes expressed in text by opinion holders. In its most basic form it focuses on binary classification of the sentiment of the opinion holder (positive or negative) [13], but can be extended to mine for more complicated opinions such as emotions or detecting sarcasm. One of the popular uses of sentiment analysis is predicting stock market behaviour, as well as getting immediate

feedback on products, political campaigns, decisions by monitoring social media [5].

Approach to sentiment analysis might differ depending whether focus is on document-level analysis or sentence level analysis - where document-level analysis assumes the entire document to have one clear area of focus, while sentence level analysis analyzes each sentence individually. Apart from those, there are more types of sentiment analysis such as aspect-based analysis (which is use when the document or sentence does not focus on a single entity), or comparative analysis (when direct opinion is not desired, but it is needed to contrast opninions with each other) [5]. For the purposes of this project, it seems that the sentence-leve approach is the most suitable, as dialogues comprise of mostly single-sentence lines (rarely more than three sentences per line) and emotional payload may change as the dialogue progresses.

Traditionally sentiment analysis is performed by various classification mathods (usually supervised learning). Naive Bayes classifier and support vector machines were proven to yield pretty acurrate results (over 80% accuracy in general) [13]. The major constrait of those methods is that their quality is tightly linked with the quality of the lexicon used and size and quality of training datasets.

One state-of-the-art solution to emotion analysis problem has recently became publicly available. Created by IBM, Watson Tone Analyzer is a tool capable of accurate emotion labeling (fear, joy, anger, etc.) as well as tone labeling (analytical, confident, tantative, etc.). The system is based on the *Big Five personality traits* model [4]. Thanks to this approach, Watson is capable of much more detailed analysis of natural language than most other tools.

### 2.4.2 Emotions

To fully understand how to approach the problem of animation generation from emotional natural language it is important to look at how can emotions be understood and represented in computational terms. Emotions are complex and subjective in nature, so it is important t deconstruct emotions into something that can be worked with. A traditional approach to this problem is using the six basic emotion model. Specified by Paul Ekman and Wallace V. Friesen, who by studying nativa papua-new guinean tribes have discovered six universally recognized and understood emotions: [2]

- Joy
- Anger
- Surprise
- Disgust
- Fear
- Sadness

Although various research proposes changes to the six basic emotions model (such as reducing it to just four basic emotions [8]), this model is widely accepted and used in literature and software (such as IBM Watson tone analyzer, or various lexicons such as the NRC Word-Emotion Association Lexicon [1].

[1] `http://www.saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm`

Psychologists theorize that the basic emotions can be used as building blocks to achieve more complex emotions. The specific emotion can be described as a mixture of basic emotions and their intensities. Robert Plutchik, a supporter of that theory, has represented emotions on a wheel-like diagram ( 2.1) that depicts how two basic emotions combine to create a more complex emotion [2].

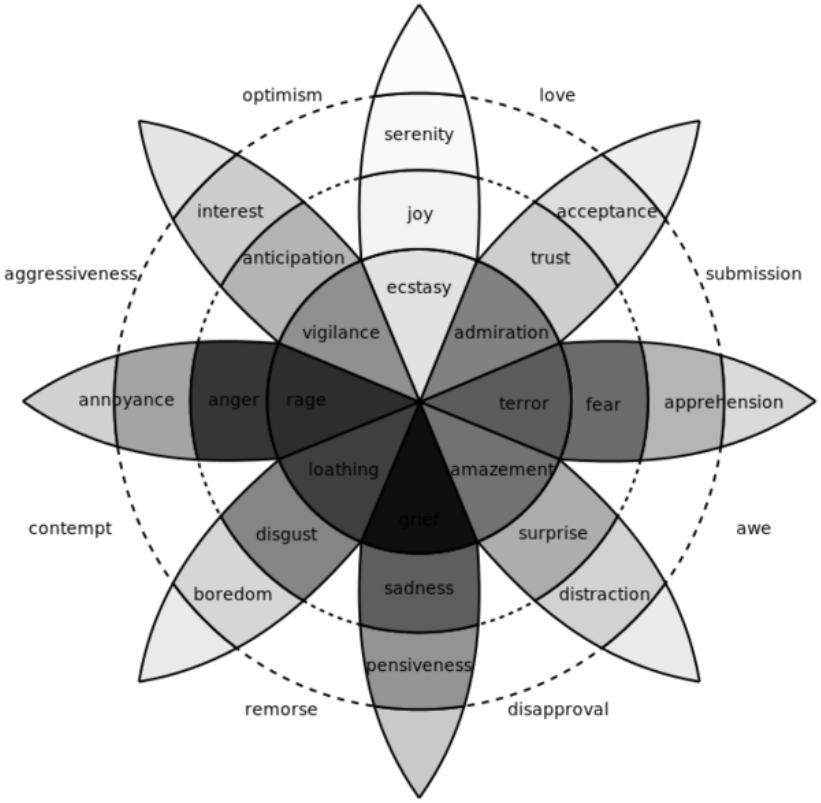


**Figure 2.1:** Plutchik's wheel

It seems that it is enough to extract the very basic emotions from the text, as more complex emotions can be understood as combinations of those basic ones.

## 2.5 Animation Software

The output scene must be created in some software able to play and use the scene. Developing a tool for this would be too time consuming - such software is not a small undertaking and with the time constraints this would result in a very rudimentary framework that would be very limiting. Therefore a choice must be made between existing frameworks.

The animation software must satisfy the following requirements:

- Flexibility and editability - As the scenes created by the generator will not be perfect, the software must provide powerful animation editing features.
- Exporting - It is desirable for the animation to be able to be exported into variety of different formats that can be used by other frameworks useful within game development.

- Availability - As the proposed tool is developed with small developers in mind, it is most desirable for the tool to be available without any unnecessary fees or licensing. The proposed tool shoul also provide help for people unfamiliar with animating, who would be unwilling to pay additional fees for something they are not familiar with.

- Familiarity - Although most animation software is based on similar concepts, due to time constraints my previous experience with the software is also important.

### 2.5.1 Maya

Maya is a 3D computer animation software developed by Autodesk. It supports modeling, rendering, simulation, texturing, and animation. This software is an industry standard and has been used for such projects as the Halo franchise. It supports all the animation editing and exporting features needed for this project, however it comes with a pretty harsh pricetag of $180 per month [2]; a price which would make the potential reach of the proposed tool much smaller. Moreover, although I am familiar with animation concepts, I am not familiar with Maya.

### 2.5.2 Blender

Blender is an open source animation software. Similarly to Maya, it supports modeling, rendering, simulation, texturing and animation. Due to its open source nature the software may be less usable or stable at times however it is still very powerful and recognized within the industry. Blender's rendering engines are not as sophisticated as those of Maya. It supports exporting the animations to Collada, Alembic, 3D Studio, FBX, Motion Capture (.bvh), Wavefront, X3D and Stl file formats. This means that the final animation could be exported and used by pretty much any other tool. Blender is completely free to use and available to anyone. I am familiar with the tool

### 2.5.3 Unreal Engine

Unreal Engine is by far the most popular and most powerful publicly accessible game engine. Since the proposed tool would find most use in games it would make sense to create the scenes directly in a game engine. This approach however has some disadvantages - animation editing features in game engines are much more limited than those of a software dedicated to creating animation. Moreover, any game engine will not support the same exporting features (however, since the animation would already be in the engine, the need for exporting is arguable). Unreal engine is free to use, however Epic Games will seize a portion of income generated by a product developed with Unreal Engine. [3]

### 2.5.4 Unity 3D

Unity 3D is the most popular freely available engine after Unreal. It suffers from the same drawbacks regarding animation editing features and is in general less stable and sophisticated. The only reason why Unity would be more suitable than Unreal for this project is my familiarity with the Unity 3D framework.

### 2.5.5 Final Choice

Upon taking a closer look on the available software, I conclude that Blender is the most suitable tool for the task as it satisfies the requirements best. It provides all the necessary editing exporting

[2] `https://www.autodesk.com/products/maya/subscribe`
[3] `https://www.unrealengine.com/en-US/faq`

and editing features, is free and easily available and I already have experience with using it.

## 2.6 Motion Database

The dialogue scenes in this project are going to be assembled from existing short motion capture clips. While a potential user of the tool proposed by this paper (a game studio) would most likely possess resources to create their own motion capture clips and be in posession of legacy motion capture data recorded for past projects, I must resort to use other resources. There exists a multitude of motion capture data freely available online[4]. Among them, the most famous is the Carneigie Mellon University's CMU Graphics Motion Capture Database[5]. It contains thousands of recordings of people walking, dancing, running, performing everyday activities, playing sports and performing gestures.

Only a small portion of that database would be useful to this project. There is however another database that focuses on matters more aligned with this project - the Emotional Body Motion Database [6]. The database was created by the Max Planck Institute for Biological Cybernetics and features solely recordings of people performing gestures associated with some emotion. The recordings feature all the basic emotions listed earlier as well as other emotions such as pride, relief, shame and more. There are three main types of motion capture recordings in this database:

- Narration - Actors were recorded while reading a story - their gestures were captured while narrating a part of the story that emphasizes some emotion.
- Nonverbal - Actors were recorded performing a gesture associated with some emotion in a nonverbal setting.
- Sentence - Actor were recorder while performing gestures when talking.

The database contains over 1400 clips, each labeled with what emotion it is supposed to represent, and what emotion would an observer assoiate the recording with [15] [16]. An example entry in the EBMD can be seen in 2.2.

| ID | Download as | Intended emotion | Intended polarity | Perceived category | Perceived polarity | Accurate category | Accurate polarity | Duration | Peaks | Speed | Span | Acting task |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | anger | --- | anger | --- | --- | --- | | | | | Narration |
| 1354 | bvh mvnx | anger | negative | anger | negative | 1 | 1 | 3.125 | 3.667 | 0.09641 | 0.1206 | Narration |

| Acting subtask | Actor | Gender | Age | Handedness | Native tongue | Responses | Consistency | Text |
|---|---|---|---|---|---|---|---|---|
| --- | --- | --- | | --- | English | search | | search |
| tale_six_swans | SiGl | f | 21 | right | English | anger, anger, anger, sadness, neutral, sadness, anger, neutral, anger, neutral, anger | 0.545 | NA |

**Figure 2.2:** An example emdb entry

## 2.7 Uncanny Valley

When dealing with animation, modelling, sculpting, etc. it is important to remember about the phenomenon of the uncanny valley. Not taking this phenomenon into account might cause unexpected negative results.

[4] http://jeroenvanboxtel.com/MocapDatabases.html
[5] http://mocap.cs.cmu.edu/
[6] http://ebmdb.tuebingen.mpg.de/index.php

### 2.7.1 What is the uncanny valley

The uncanny valley is a phenomen regards aesthetics. It states that when a humanoid object (a character model, a robot) behaves in a more realistic, human-like fashion, it might not be necessairly perceived as more human-like. The *valley* suggests that there exists a certain *dip* in human observer's affinity for a human replica [10].

The concept was first identified byrobotics professor Masahiro Mori in 1970. In his essay on the uncanny valley he explains the relationship between a humanoid object's human likeness and observer's affinity towards the object. He has noted, that as the object becomes more human like, the observer's affinity increases. However, when the human likeness of the object reaches certain point, the human affinity for it decreases drastically. Then again, when the human likeness becomes very high, the affinity drastically rises again. The observers might perceive humanoid objects that are not quite human like as funny, cute, in need of repair, etc. The obsever perceive very human like objects are almost indistinguishable from reality - and while they may not be perfect, there is nothing alarming about them. However, the observers have described *fairly* human like humanoid replicas as *creepy* or *eerie* [10] [11].

The concept is easily depicted on a graph. Figure 2.3 draws the relationship between human likeness and observer affinity. According to this representation, when a replica's human likeness reaches about 80%, observers affinity quickly becomes negative. Figures 2.4 2.5 2.6 shows an example of this concept, by showing three different robot examples and their relation to the uncanny valley principle.

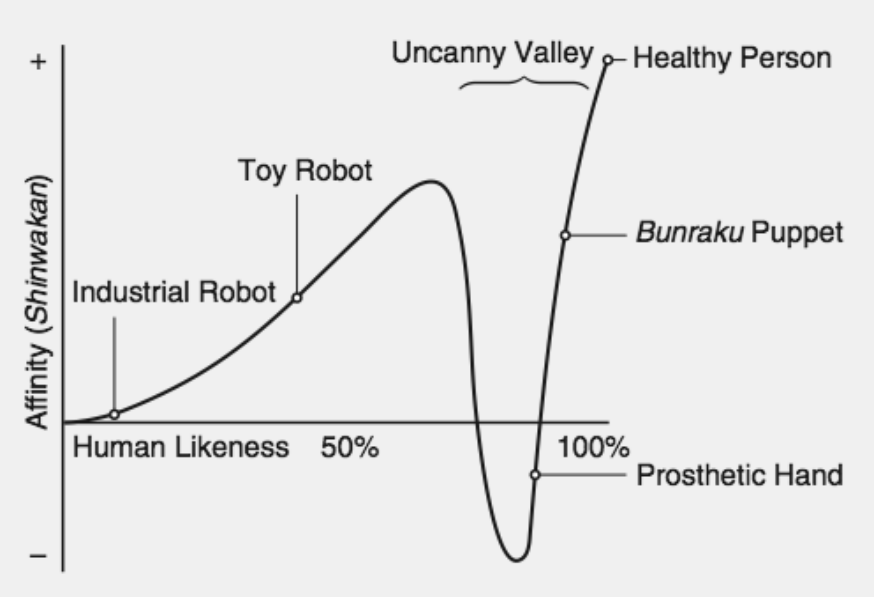


**Figure 2.3:** The uncanny valley

### 2.7.2 Why uncanny valley matters in games and animation

Since animation and modelling deals with humanoid objects, the uncanny valley principle applies to it as well. Many game studios h will prefer to use comic-like graphics and other solutions in order to purposefully make the game less realistic. Oftentimes a less realistic game will be received better than a game that is quite realistic but not exactly so. Characters that fall into the uncanny

**Figure 2.4:** This robot does not fall into the uncanny valley category. It is a humanoid but it is not realistic - it is usually perceived as funny or cute.

**Figure 2.5:** A robot that falls into the uncanny valley category - It is realistic enough to be human-like, but it is not human-like enough to be perceived positively and described as realistic. It provokes a feeling of eerieness or being alarmed in the observers.

**Figure 2.6:** This robot does not fall into the uncanny valley as it is realistic enough to feel familiar and normal.

valley category are often no longer perceived as humans, they are not realatable and laughable [1].

This problem is exactly what caused the specacular failure of Mass Effect Andromeda. BioWare's process of creating large amounts of dialogue animation has produced animations which are quite realistic but do not feel human. The players quickly lost interest in the characters, as serious scenes behaved in a fashion that was out of place, unpredictable, and did not highlight emotions they were trying to convey through speech. The case study of this game shows not only why a good system of generating animation is important, but also that a generation system that is generally realistic might be too little and too much at the same time [7].

This project is also subject to uncanny valley. This principle might cause a set of pretty decent computer generated animations to be perceived very negatively by the audience. I will evaluate on that in section 3.3.

## Chapter 3

# Analysis

The methodology used to develop the project, technologies involved in the develpment and a breakdown of risks that concern the project are described in this chapter..

## 3.1 Methodology

The activities required in order to develop the project are listed below:

- Develop a project plan.
- Review literature. Review existing software, learn about other approaches, analyze existing research.
- Review technology. Find out about software may be useful to this project and learn how to use it.
- Analyze risks. Specify what might be the potential pitfalls that may deem this product unsuccesful.
- Prototype a module that uses processes the script and outputs a structured file that can be used to generate the animated scene.
- Obtain a sufficient amount of animated clips. Create a custom downloader/importer if necessary.
- Tag and classify animation clips. Categorize clips by emotion, length, etc.
- Prototype a Blender extension that uses the created structured files to output an animated scene.
- Iterate on the existing software adding new features.
- Obtain more animated clips, classify and store them.
- Evaluate the prototype with real audience.

The minimum working prototype will be developed first. When all three modules are ready and in place, more features will be added iteratively. The software will work better if many animation clips are available, therefore as many clips as possible will be obtained and classified.

The generated animation must be evaluated with help of a real audience. How animation is perceived is subjective and cannot be decided by one person only. The evaluators will be asked to watch animated dialogue clips from various games and then watch this dialogue recreated using the proposed software. The evaluators will be asked to fill an evaluation survey. This way it will be possible to determined the usefulness and succesfulness of the proposed software.

## 3.2 Technologies and Resources

1. **Emotion analysis** - The first module of the project focuses on NLP. This module should be extract emotions and actions from the script. The most important tool used will be IBM Watson Tone Analyzer. If that tool turns out to be for any reason ineffective or imperfect, a customary tool (naive bayes classifier or a keyword classifier) can be built for that task. Actions can be extracted using a variety of information extraction software such as MITIE or Ollie.

2. **Motion capture** - The EMBD (emotional body motion database) will be used to source the animations of body movement and gestures. The database provides the recordings as BVH files which is convenient as most animation software (such as Blender or Maya) can import BVH files. The emotional metadata about the animated clips will be stored in an SQLite database. Storing the data using this method will allow the data to be easily and quickly searchable while less complicated than using a full fledged database software (such as MySQL od PostgreSQL) (very few tables are needed, there is no need to use advanced DB software).

3. **Animation software** - As aforementioned in section 2.5, the animation software that satisfies all the specified requirements is Blender. Blender supports all the necessary modelling and animation features. It also supports creating extensions allowing the animation to be created and assembled by code.

4. **Programming** - Python 3 will be used to implement the software. Python is the only language supported for add-on development for Blender. For other modules that do not rely on Blender, Python was chosen due to its development speed and in order to keep consistency among modules.

## 3.3 Risk Analysis

This subsection evaluates on ways in which this project can fail. Since reception of animation is subjective it may be hard to pinpoint what exactly went wrong with the software. It is possible that the audience will decide that the animations are not good enough but they will not be able to describe why. Because of that it is important to understand those outline those issues before conducting any tests.

### 3.3.1 Failure due to emotion analysis

The first way in which the software may fail is due to the emotion analysis component. If the final animation does not reflect by body language the emotions of the characters it may mean that the

dialogue were not analyzed properly, meaning that the results provided by emotion analysis are simply not accurate enough.

In this project I will be using IBM Watson which is a pretty good benchmark regarding emotion analysis. It is safe to say that is Watson is unable to perform the task well enough there is no software that would be. If a failure is identified to be caused by the emotion analysis, this might mean that:

- The NLP methods are not yet advanced enough to perform this task. This project will remain unfeasible until we see an improvement in emotion analysis techniques.
- NLP alone is not enough to solve this problem. Other approaches should be used alongside NLP emotion analysis, such as analysis of the tone of the recorded voice or emotional analysis of facial expressures of the voice actors.

Since there is no significant research done regarding combining emotion analysis and animation, there is no way to know for certain whether this approach will work. Because of that the risk of such a failure is moderately high.

### 3.3.2 Failure due to the motion capture data quality

The software proposed by this paper can only ever be as good as the motion capture data provided. While the software will be designed to work with custom motion dapture databases, the EBMD is used for this project. This means that if the clips provided by EBMD are too ambiguous - that the recorded body language do not convey the emotion clearly enough, or that the recerded movements are unnatural, too subtle or too over emphasised, the output animation will not be perceived as successful. In this case this might not be the problem of the software or the approach at all - the failure is generated only and specifically by the motion capture recordings.

The EBMD provides quite some metadata regarding each clip. Among the metadata provided there is information about what emotion the movement tries to convey and what emotion was perceived by the audience when watching the clip. Because of that the risk of failure due to motion capture data quality is fairly low as the animation should in general unambiguously convey the intended emotion. There is however still a risk that the motions will be perceived as conveying correct emotion, but being too unnatural and unrealistic.

### 3.3.3 Failure due to the uncanny valley principle

The uncanny valley principle poses a high threat to the project. The final animation satisfying the uncanny valley principle is possibly the worst scenario in which the project may fail. This failure is easiest determined by comparing the final animation with other games. Two types of game animation will be shown to the audience - animation is advanced and generally considered good, and animation that is basic, unrealistic, seemingly random and monotonous. If the audience judges the animations generated by this software to be worse than both of these animation types then it probably means that the animations comply with the uncanny valley principle. It means that while the animation generally is realistic and matches the intended emotion of the speech, the animation still feels unnatural and it might have been better to go with a much simpler approach, such as keeping the characters mostly static.

If the uncanny valley is achieved by the output animation, this might mean a few things. If the uncanny valley effect was created by the nature of the EBMD recordings, this means that replacing the database with a different might just solve the problem. However, it might mean that the NLP emotion analysis approach to generating animation is inherently flawed. It might just prove that text data alone is never enough to produce convincing animation and while this method might be useful in combination with other methods, emotion analysis approach will always produce unconvincing animation.

## Chapter 4

# Requirements Specification

This section describes functional and non-functional requirements of the proposed software. The software must be developed accordingly to the requirements and must satisfy all of them in order to be truly successful.

## 4.1 Functional Requirements

1. **Analysis of a semi-structured script** - The software takes a semi structured script as input. The structure of a script must resemble a structure of a movie script. The script provides 2 kinds of information - characters involved and lines of dialogue spoken by them. The software must be able to extract emotions from the dialogue lines.

2. **Store motion capture data with regards to emotions** - The software must store and tag the motion capture clips with relevant metadata about what kind of emotions they represent. The database must be easily and quickly searchable.

3. **Find relevant animation clips** - The software must take information about character's emotions and actions and choose animation clip that best represent's characters behaviour. The chosen clips must reflext characters emotions, but also need to be of correct length to match the speed of the speech, as well as not repeat too often. The system must be compatible with Emotional Body Motion Database published by Max Planck Research Institute.

4. **Assemble the final scene** - The software must be able to output the final animated dialogue scene. The outputted scene must be fully editable, enabling various adjustments before rendering. The final scene must also be exportable to other formats so it can be used with game engines or other editing software.

## 4.2 Non-Functional Requirements

- The user should be able to use the software with a custom motion-capture database (Usability).

- The user should be able to assign differenct character models to different characters (Usability).

- The user should be able to fully customize and edit the final scene (Usability).

- The software is designed to automatically create big amounts of animated scenes. The time of generating a scene is not a high priority, but it must be reasonable (Peformance).

- The software must support common animation file formats (fbx, bvh) (Portability).
- The software must be modular enough so that different parts of it can be replaced with ease .It mustbe possible to replace emotion analysis software and 3D animation editing software with other software (Portability).
- The output animation must be perceived well by the audience and judged to be acceptable for use in a game given minor adjustments (Quality).

## Chapter 5

# Design And Architecture

This chapter describes the design of the system. This includes both the underlying decisions of the system as well as the user interface design. The chapter also presents on a more detailed view of the system's architecture.

## 5.1 System Design

The system has to work in two major steps. The first step is to take semi-structured natural language script and analyze it. This means that the system must analyze each dialogue line and infer some emotional values from them. Using those emotional values and the length of the dialogue line, the system must find best matching animation clips from the motion capture database. The results of this step is a file that specifies which characters say which lines of dialogue and it also specifies which animated clips accompany those dialogue lines.

After this step is completed, the file must be intepreted into a final scene. The importer must be able to read the file, import required models and animations and generate the final scene. As every animation software and game engine is a little different, each of them would need a custom importer. Those would be very similar in principle, but differ slightly because of the implementation of given software. For my project I have created the importer for Blender. This is because Blender is open source and available to anyone, as well as I am the most familiar with this software.

## 5.2 Emotions

For my project I have decided to only use the following emotions: joy, fear, disgust, anger and sadness. Tradinionally, surprise is also part of the six basic emotions model by Paul Ekman and Wallace V. Friesen. However, IBM Watson Tone Analyzer, that I am using for sentiment analysis, is unable to detect surprise in the text (more on that in section 5.7) - therefore I had to abstain from using this emotion in this project. The emotions are expressed as decimal numbers between 0 and 1 - this identifies whether a given emotion is present in a specific motion clip or text and how intense that emotion is. It also allows to create a mixture of emotions in order to represent more complex emotions. The EMBD describes the animations in terms of their emotion category, but provides no value (the EMDB will not differentiate between an angry gesture, and a *very* angry gesture). Therefore each animation clip has to be manually reviewed in order to assess it's emotional properties.

## 5.3 Architecture

The architecture of the system is essentially a pipeline. The modules process resources and pass them onto the next module. They can be completely unaware of each other. This allows for a lot of flexibility and helps achieve some of the requirements. The emotion analysis API can be replaced with a different one, the user can use a custom motion capture database, and a different importer can be created if the user wishes to use software other than blender.

There are three main modules:

- Text Analysis
- Animation Clip Matcher
- Animation Generator

The Text Analysis module parses the text and analyses emotion using some API. It takes a script as an input and passess the parsed and analyzed text to the Animation Clip Matcher.

The Animation Clip Matcher uses the motion capture database to find the best animation clips to accompany the speech. The Animation Clip Matcher must take into account the emotion analysis of the text and find animations with similar emotional score. Another important constraint is the time of the animation. The animated clips have different lengths and a chosen clip must not be significantly longer than the spoken/read text. In case the animated clip is shorter, the Matcher must choose more than one matching clip. The Matcher outputs a JSON file which specifies all dialogue lines; it states which characters perform which emotional gesture actions and where is the animation file located.

The Animation Generator reads the JSON file and imports all needed animations. The user is now able to assign character model for each character and run the generator. The generator will assemble the animations in correct order, focus the camera on a currently speaking character and add subtitles. The output is an animated scene that can be either manually edited, exported to a file or rendered.

The full diagram of the system can be seen in figure 5.1.

## 5.4 Character Armature and Model

As the system is designed to be used in games, one of the main requirements was for the system to handle various character models. Because in a game each character is represented by a different model, this is a necesity. The model itself is relatively irrelevant - it must hovewer support the same or similar enough armature to the armature the system was designed around. The armature is shown in detail in figure 5.2. For an armature to be supported, the bones might have corresponding names. Any extra bones or bones with unmatching names will be left unanimated.

## 5.5 EMBD Importer and Showcase Generator

The following are small tools created to help automate / bootstrap the process of acquisition and categorization of the animations.

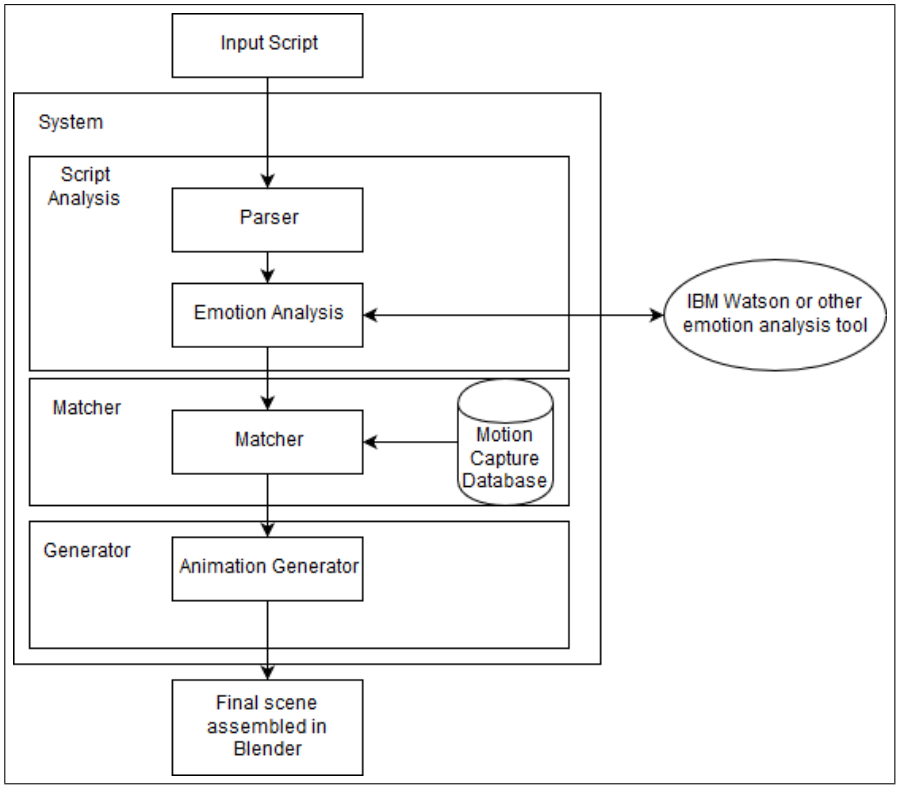


**Figure 5.1:** The pipeline architecture of the system

### 5.5.1 EMDB Importer

The EMDB importer's purpose is to help prepare the animations obtained from EMBD to work with the system. The EMDB animations have a few issues making them incompatible with the system - The armatures are a little different with unmatching bone names, many animations were recorder while actors were seating [1] and are saved in BVH files [2].

To import an animation, the importer must follow through the following steps:

1. Download animation from link to file (receive http link as input) and import the animation from the downloaded BVH file
2. Remove animation data from the legs
3. Rename the bones that the animation data applies to so that they match those of the target architecture
4. Import the target armature
5. Apply the action from imported from EMDB to the target architecture
6. Export the animation data to FBX file

The EMDB importer help automate the process of acquiring animations by allowing to download and process a huge bulk of animations, saving them in folders that correspond to their emotion category and showing their length in frames in the filename.

[1] Leg animation is irrelevant for this system
[2] BVH files are suitable for motion capture, but can be more problematic for generic animation than FBX

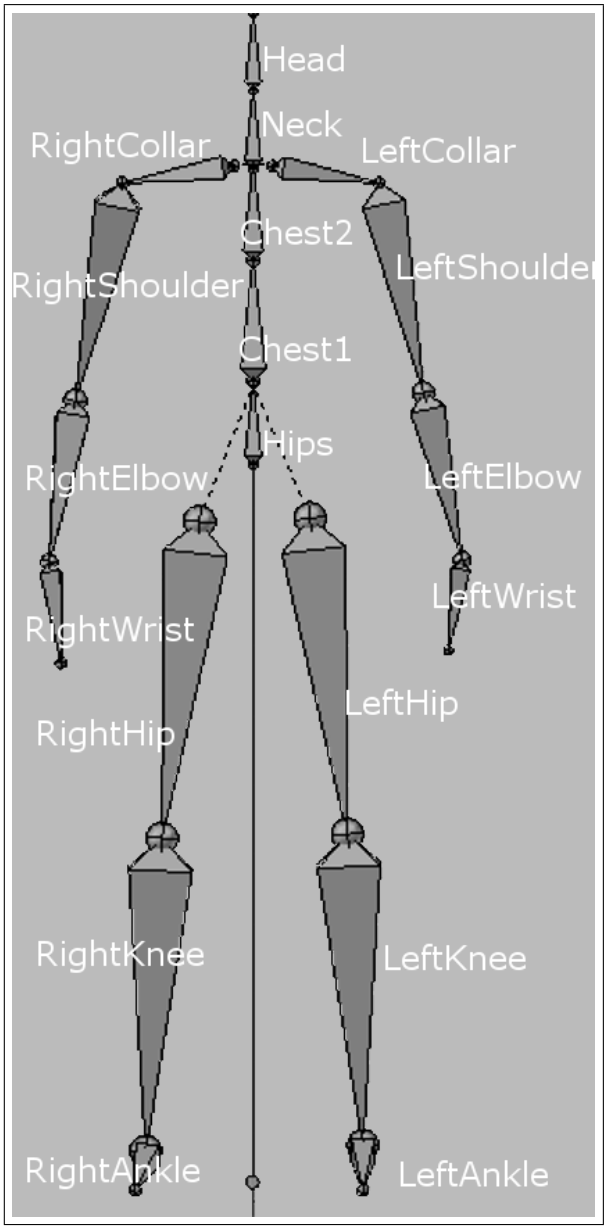


**Figure 5.2:** The armature supported by the system

### 5.5.2 Showcase Generator

The purpose of the showcase generator is to help categorize and store the animations in the database (section 5.8). The showcase generator outputs instructions for the animation generator (section 5.10) which allow the animation generator to create an animation which shows all the animations in a given directory one after another. Since the animations have to be analyzed manually (in order to assess their emotional value), a clip showing all animations allows for a quick and easy analysis.

## 5.6 Input

The input of the system is as follows:

The input is a file that represents a dialogue. Each character that participates in the scene must ble clearly stated. Each dialogue line must have a dialog line must have a character clearly

associated with it. Character names are specified after five tabs. Dialogue Lines are specified after three tabs. Each file must end with "ENDSCRIPT" with no indentation.

I used this format as this format is often used to represent movie scripts. One can find hundreds of scripts saved in this format on The Internet Movie Script Database (`www.imsdb.com`). The example input can be seen in figure 5.3.

```
                                SANDRA
                    Sonny?  Sonny, who is it?
                    What is it?

                                SONNY
                    They shot the old man.

                                SANDRA
                    Oh God...

                                SONNY
                    Honey...don't worry.  Nothing else
                    is going to happen.

ENDSCRIPT
```

**Figure 5.3:** An example of system's input

## 5.7 Emotion Analysis

As aforementioned, there are many ways to perform emotion analysis of the text. I have chosen to use IBM Watson Tone analyzer for this task as it offers a state-of-the-art service accessible for prototyping (a lot of tools offering high quality emotional analysis are bult for commercial purposes and not available without paying high fees).

The module takes each dialogue line and sends to IBM Watson Tone Analyzer for analysis. Watson's REST API is used to accomplish that. Watson returns all the emotions with values between 0 and 1 found in the text. Each dialogue line now has emotional values assigned to it.

## 5.8 Database

The motion capture database consists of two main parts: the files that contain the animated clips and a database that holds the metadata about the animations and character models.

The animation data is stored in FBX files - one animated action per file. The folder structure can be customized - for the purposes of this project the animations are stored in a way that describes the emotions they convey [3]. The folder structure is irrelevant as long as it stays consistent with the entries in the SQL database.

The database is implemented using SQLite. It is the perfect tool for this task as the database needs to be simple, relatively small and easily searchable. The animation metadata is held in a table that look like this FIGURE. The emotional values of each animated clip need to be manually

[3] For example: animations > anger > subtle > disbelief1.fbx

adjusted. When all values are set to zero, it means that the clip carries no emotional impact (neutral). An example of a few database records can be seen in figure 5.4.

| | id | name | file | Frames | anger | joy | fear | disgust | sadness |
|---|---|---|---|---|---|---|---|---|---|
| | | Filter | Filter | F... | | | | F... | Filter |
| 1 | 1 | angry_shrug | anger/anger1... | 40 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2 | 2 | shrug_hand_t... | anger/anger2... | 48 | 0.3 | 0.0 | 0.0 | 0.0 | 0.0 |
| 3 | 3 | shrug_hand_t... | anger/anger3... | 52 | 0.4 | 0.0 | 0.0 | 0.0 | 0.0 |
| 4 | 4 | cower | fear/fear1.fbx | 40 | 0.0 | 0.0 | 0.9 | 0.0 | 0.0 |
| 5 | 5 | cover_head | fear/fear2.fbx | 60 | 0.0 | 0.0 | 0.7 | 0.0 | 0.0 |
| 6 | 6 | disbelief | anger/anger4... | 62 | 0.65 | 0.0 | 0.0 | 0.0 | 0.0 |

**Figure 5.4:** Database of animation clips

The other purpose of the database is to store the information about the models. The important information about a model is its name, file location, camera offset and rotation. Because models may be of different shapes and sizes, camera positioning during dialogue will not be the same. The database allows to specify a desirable camera position relative to the character that will allow to fully capture the character at a good angle. Example of a few model database records can be seen in figure 5.5.

| | ID | name | file | cam_offset_x | cam_offset_y | cam_offset_z | cam_rot_x | cam_rot_y | cam_rot_z |
|---|---|---|---|---|---|---|---|---|---|
| | | F... | Filter | Filter | Filter | Filter | Filter | Filter | Filter |
| 1 | 4 | Blue | blue.fbx | -0.5 | -1.8 | 1.65 | 0.0 | 0.0 | 1.3 |
| 2 | 3 | Green | green.fbx | -0.5 | -1.0 | 2.0 | -0.3 | 0.0 | 1.5 |
| 3 | 2 | Test | untitled.fbx | -0.5 | -1.8 | 1.65 | 0.2 | 0.0 | 3.14 |
| 4 | 1 | Basic | untitled.fbx | -0.5 | -1.8 | 1.65 | 0.0 | 0.0 | 1.3 |

**Figure 5.5:** Database of character models

## 5.9 Matcher

The matcher module depends on the emotion analysis and database modules. It takes the emotion data and searches the database for a suitable animated clip. The matcher's input is a dictionary where the keys are emotions (anger, fear, etc.) and values are e 0 to 1 decimal.

### 5.9.1 Score matching

Firstly, the matcher will decide whether the text carries enough emotions. If no emotion surpasses the constant relevancy threshold (in this project the threshold is set to 0.2) the animation is deemed as neutral. If there is only one emotion, the module will look for an animation matching that emotional value. If more than one emotion exceed the threshold, only the two most important (highest value) will be considered. That is because certain emotions often go in pairs (such as anger and sadness), but combinations of more than two emotions are rare, confusing and hard to represent with an animation. The searches the database for animations that may be fitting and compares the text emotional values and animations emotional values. The pseudocode behind these calculations can be found in figure 5.6.

### 5.9.2 Length Matching

The matcher must account that even the most emotionally fitting animations may not be the most suitable. It is very important to take into account the speed of speech - gestures performed in an animation must match the length of character's dialogue line. Otherwise there will be situations where a short animation is chosen for a few sentences worth of dialogue, or a very long animation is chosen for just a few words of dialogue.

To account for this, constant frames per second (to define animation length in real world terms) and words per second (to calculate the length of a dialogue line in seconds) must be defined. The matcher uses those to calculate the ratio between an animated clip length and speech length. This ratio is used to adjust the score calculated using emotional values (section 5.9.1). How the two matching operations work together can be seen in the pseudocode in figure 5.6.

Additionally, the matcher will ensure that the animation is not too monotonous. The matcher will not choose twice the same animation in one dialogue - it will prioritize a similarly matching animation that has not yet been used. The animations will be reused if no other matching animations are available.

## 5.10 Animation Generator (Importer)

As a forementioned, in this project I have used Blender to support the animation generator model. The generator is a Blender addon. It can either be used as a script, or be installed as an addon to be fully and permanently available within Blender.

The generator first reads the JSON file prepared by previous modules and loads required animation clips from files. The user needs assign a 3D character model to each character participating in the dialogue. The required 3D character models are animated and the camera is positioned accordingly to the model's metadata stored in the database. The animated characters are positioned to directly face each other (the program currently supports only up to two characters in one scene). Each animated action is assigned to a corresponding character model and then pushed on a separate NLA strip (figure 5.7). The program keeps track of how many frames are being used so that newly added actions begin just when the previous actions have ended.

To setup some basic background and feel, a floor (plane) is added beneath the characters and two directional lights cast light down from above the characters.

The camera is added to the scene and positioned using the metadata retrievec from the database. At the start of each character's actions the camera is positioned and rotated to focus at the character performing an action.

Subtitles are added as Video Editor Sequence Strips spanning between frames that correspond to the animation (figure 5.8).

The final animation can be edited and adjusted in Blender (figure 5.9), rendered (figure 5.10) or exported to file.

## 5.11 User Interface

The Animation Generator module is the only module that features a Graphical UI. The module is embedded into Blender and uses extends Blender's interface.

The UI is minimalistic and simple to use. Upon installation and activation a new tab is added

```
constants: RELEVANCY_THRESHOLD
           WORDS_PER_SECOND
           FRAMES_PER_SECONDS

function get_length_match(text, animation)
  animation_length = animation['frames'] / FRAMES_PER_SECONDS
  text_length = count_words(text) / WORDS_PER_SECOND
  score = min(animation_length, text_length) / max(animation_length, text_length)

  return score
end_function

function get_emotion_match(emotions, animation)
  differences = []
  for emotion in emotions
    value1 = emotion.value1
    value2 = animation[emotion].value
    differences.append(abs(value1 - value2))
  end_for

  average = sum(differences) / length(differences)
  score = 1.0 - average

  return score
end_function

function get_matching_gestures(text, emotions)
  # Sort emotions so that the most relevant
  # is at the start of the array
  emotions = sort(emotions)
  emotions = reverse(emotions)

  matching_animations = set() # Don't allow for duplicates
  neutral = False
  if emotions[0].value < 0.2 #neutral
    neutral = True
    matching_animations = get_emotions_from_database('neutral')
  else
    matching_animations = get_emotions_from_database(emotions[0].emotion)
    if emotions[1].value >= 0.2
      matching_animations.append(get_emotions_from_database(emotions[1].emotion))
    end_if
  end_if

  emotion_scores = {} # a dictionary {animation: score}
  # Calculate scores
  for each animation in matching_animations
    score = 1
    if not neutral # All neutral animations get the same score
      score = get_emotion_match(emotions, animation)
    end_if
    emotion_scores[animation] = score
  end_for

  # Adjust for length
  for each animation in matching_animations
    # Multiply to adjust accordingly
    emotion_scores[animation] *= get_length_match(text, animation)
  end_for

  # Sort so that most matching animation is first
  emotion_scores = reverse(sort(emotion_scores))
  return emotion_scores # Return list of matching animations
end_function

```

**Figure 5.6:** Pseudocode explaining how the matcher calculates the scores

to the menu on the left hand side. Figure 5.11. Rectangle 1 represents the left hand side menu. Number 2 shows the tabs where the bottommost one belongs to the extension. Number 3 shows the actual UI of the extension.

From here the user is able to either clean the scene (might be useful especially if something unexpected happens or the work of the extension is interrupted) or prepare the scene. To prepare the scene the user must first initialize the four variables. The user needs to point the program to where the animations and models are stored as well as the database file and the scene file (JSON file generated by previous modules).

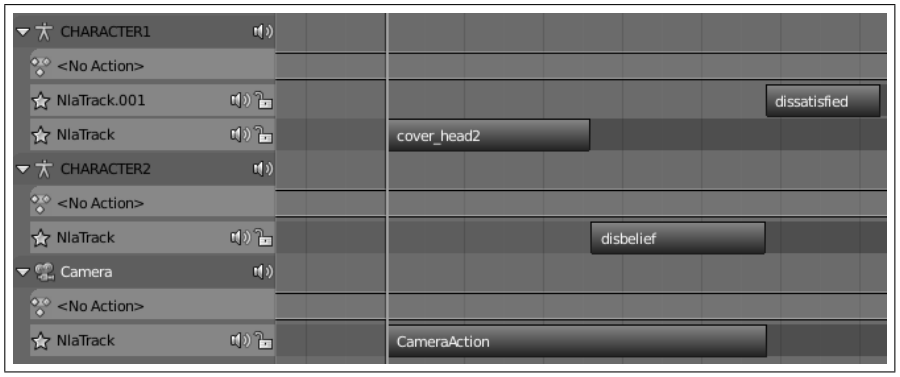


**Figure 5.7:** NLA strips of the final animation

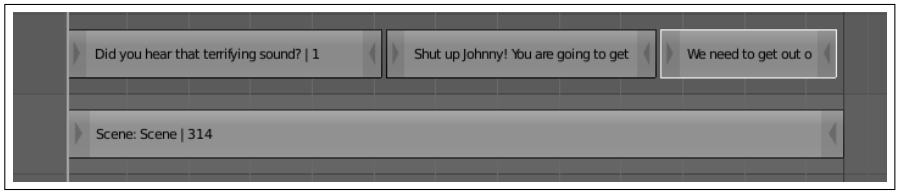


**Figure 5.8:** Video strips featuring subtitles

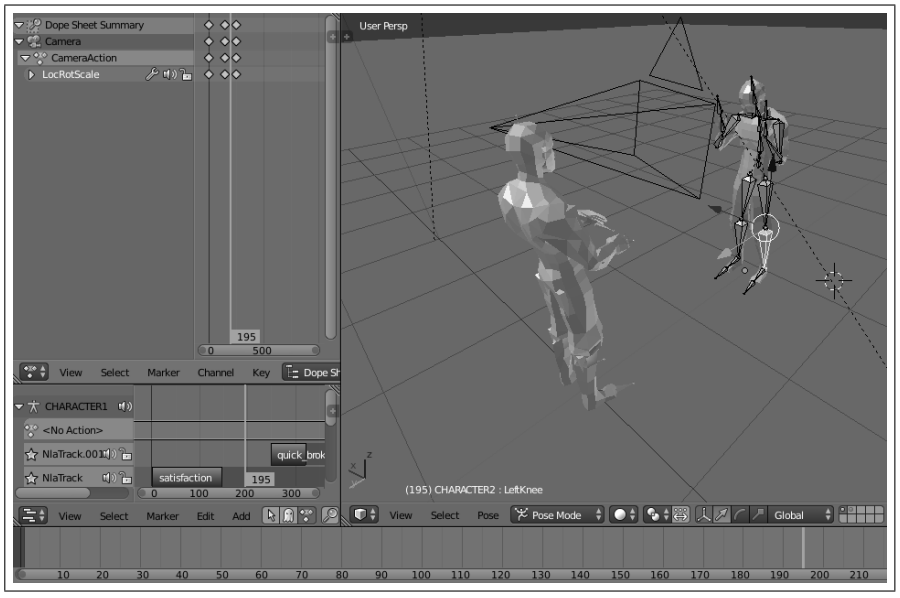


**Figure 5.9:** Editing the final animation in Blender

Upon pressing ‘Prepare‘ the program will prepare data for generating a scene. Required animations will be imported and the scene file will be parsed. There is one more step left to finalize the animation. When preparation is finished, a new menu will pop up below the currently existing one. The menu can be seen in figure 5.12. In the menu the user is required to assign a character model to each character existing in the scene. Upon pressing ‘Finalize‘ the full scene will be generated complete with lighting, camera movement and subtitles.

They key aspect of the interface is its simplicity as it allows users to generate relatively complicated animation sequences by essentially using two buttons, with no knowledge about animation and little Blender expertise required. The UI also provides flexibility as the user can decide upon

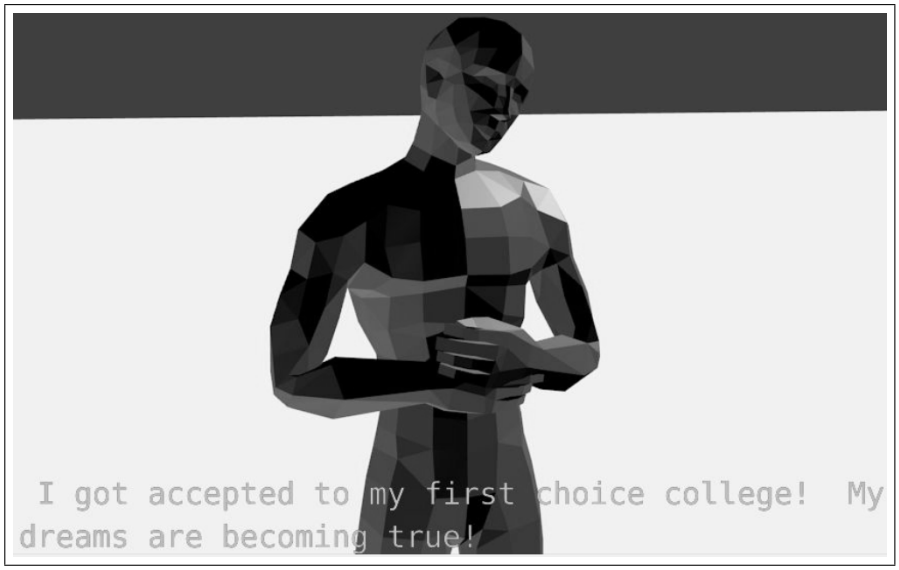


**Figure 5.10:** Rendered final animation

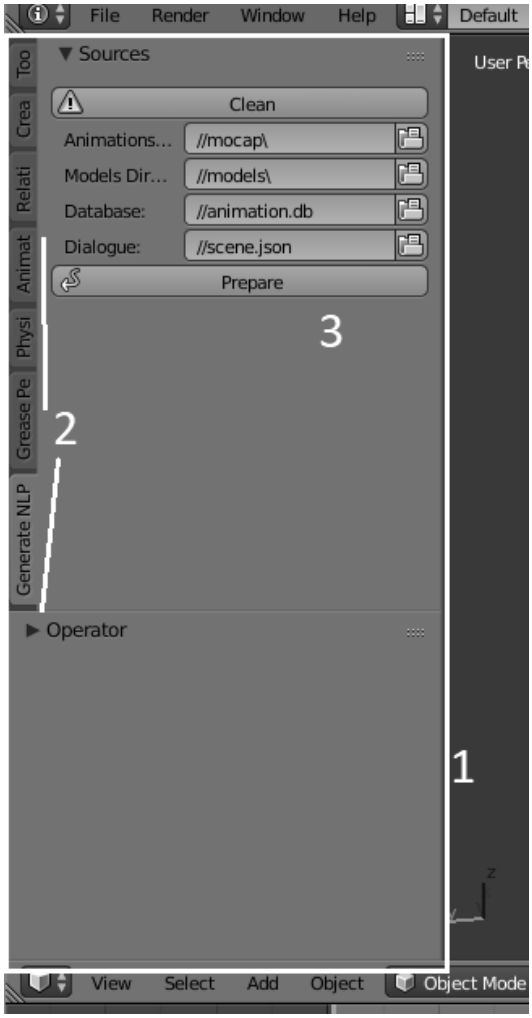


**Figure 5.11:** The addon UI

which character models to use.

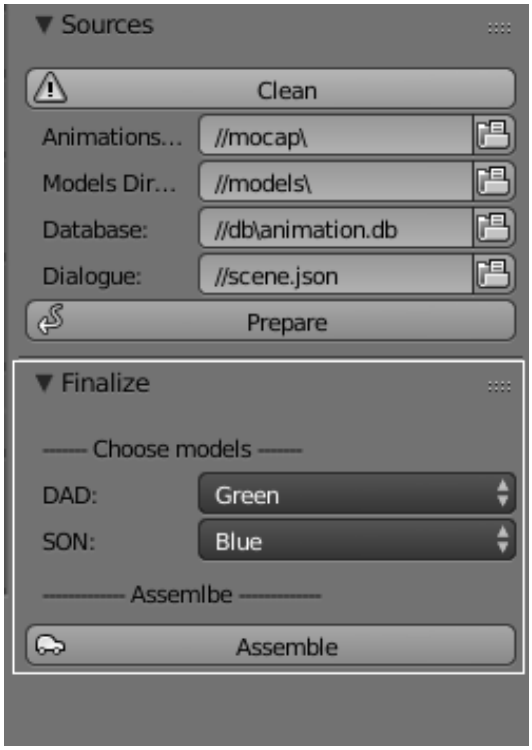


**Figure 5.12:** The finalization step (new menu highlighted by white rectangle)

## Chapter 6

# Testing and Performance

This section describes testing and potential issues of the developed software.

## 6.1 Animation Generator

There are some serious limitations to testing of this module. It is hard for a computer to decide whether the animation looks natural. Some animations imported from EMDB experience import problems which results in an animation with broken and unnatural movements. While manually analyzing the emotional values of the animations, it was important to check each animations for movements that may have been incorrectly imported. Such animations cannot be used with the software.

The animation generator is now able to handle only up to two characters in the scene (although the module was designed with extensibility in mind). If a scene JSON file contains information on more than two characters (or no characters at all) the scene will not be assembled and an error message will be displayed instead. In case the scene file is corrupted and cannot be parsed the user will also be presented with an error message.

## 6.2 Performance

One of the important requirements of the software was the speed of execution. For creating an animation consisting of 6 dialogue lines using the first module takes up to 18 seconds. The matcher module is so quick that it's execution does not influence the total execution time. The generator module takes anotherfour seconds on average. This results in an execution speed of about 3.6 seconds per dialogue line. This fulfils the requirement as the time required to assemble an animation is uncomparably less than doing that manually and is not a limitation for an animator working on the scene.

# Bibliography


[1] Brodkin, J. (2014). Game developers crossing the uncanny valley. https://insights.dice.com/2014/02/14/game-developers-crossing-uncanny-valley/.

[2] Burton, N. (2016). What are basic emotions? https://www.psychologytoday.com/blog/hide-and-seek/201601/what-are-basic-emotions.

[3] Cavazza, M. and I., P. (1999). Natural language control of interactive 3d animation and computer games. Technical report, University of Bradford.

[4] Chang, L. (2015). Ibm's watson may now be able to tell how snarky your email is with its new tone analzyer. https://www.digitaltrends.com/cool-tech/watsons-new-tone-analyzer-can-tell-if-your-email-is-too-snarky/.

[5] Feldman, R. (2013). Techniques and applications for sentiment analysis. *Communications of the ACM*, 56:82–89.

[6] Fenlon, W. (2016). Most of the witcher 3's dialogue scenes were animated by an algorithm.

[7] Hernandez, P. (2017). People are trashing mass effect: Andromeda's animation. https://kotaku.com/people-are-ripping-apart-mass-effect-andromedas-animati-1793334047.

[8] Jack, R. E., Garrod, O. G., and Schyns, P. G. (2014). Dynamic facial expressions of emotion transmit an evolving hierarchy of signals over time. Technical report, University of Glasgow.

[9] Levison, L. (1991). Action composition for the animation of natural language instructions. Technical report, Universirt of Pennsylvania.

[10] MacDorman, K. F. and Chattopadhyay, D. (2014). Reducing consistency in human realism increases the uncanny valley effect; increasing category uncertainty does note. Technical report, Indiana University School of Informatics and Computing.

[11] Mori, M. (1970). The uncanny valley. https://spectrum.ieee.org/automaton/robotics/humanoids/the-uncanny-valley.

[12] Oshita, M. (2009). Generating animation from natural language texts and framework of motion database. Technical report, Kyushu Institute of Technology.

[13] Rohit, W. and Jagdale, R. (2018). An approach to sentiment analysis. Technical report, Department of CS & IT, Dr. BAMU, Aurangabad, Maharashtra, India.

[14] Tominski, P. (2016). Behind the scenes of cinematic dialogues in 'the witcher 3: Wild hunt. http://www.gdcvault.com/play/1022988/Behind-the-Scenes-of-Cinematic.

[15] Volkova, E., Mohler, B., de la Rossa, J., and Bulthoff, S. (2014a). The mpi database of emo- tional body expressions common for narrative scenarios. Technical report, Max-Planck Institute for Biological Cybernetics in Tuebingen, Germany.

[16] Volkova, E., Mohler, B., Tesch, T., and Bulthoff, S. (2014b). Emotion categorisation of

body expressions in narrative scenarios frontiers in psychology. Technical report, Max-Planck Institute for Biological Cybernetics in Tuebingen, Germany.

## Appendix A

# User Manual

The system is designed to be cross platform, however it is advised to use Windows as it is the only system on which the project was thoroughly tested.

## A.1 Requirements and installation

Before using the software, several requirements must be fulfilled.

### A.1.1 Software and Libraries

Required software

- Python 2.7
- Pip
- Blender version 2.79 or newer

Libraries

Only for Linux systems: Download and install library python-dev (or python-devel)

Following python libraries must be installed:

- setuptools
- watson-developer-cloud
- wget

The packages can be installed in bulk by running:

‘path_to_python/Scripts/pip install -r requirements‘

in the main directory. As Blender usually comes with its own Python environment, you will need to run the command:

‘path_to_blender/version/python/Scripts/pip install -r requirements‘

To install the libraries for Blender.

### A.1.2 Install Generator as Blender Addon

The following steps are optional. Installing the module as addon will enable the module to be permanently incorporated into Blender.

To install the addon:

- Open Blender

- Open user settings (File > User preferences)
- Open Add-ons tab
- On the bottom panel press ‘install add-on from file‘
- Navigate to ‘project folder > generator‘ and double click on ‘addon.py‘
- In the search box on the top right corner type ‘NLPanim‘ - a greyed-out item called ‘Object: NLPAnim‘ should appear
- Check the box next to the item
- Press ‘Save user settings‘ in the bottom left corner
- Exit user preferences

The module is now installed as an add-on. In the 3D-view (default view) on the bottom of the right-hand side menu there should appear a tab called ‘Generate NLP anim‘ (figure A.1).

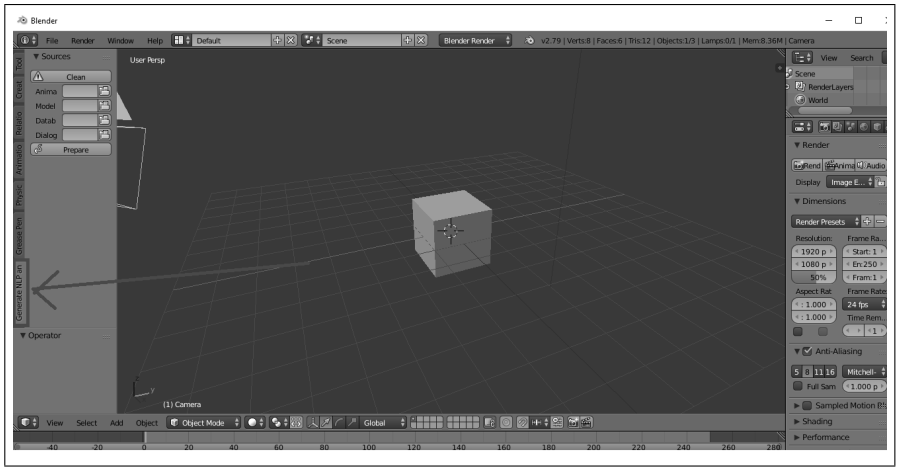


**Figure A.1:** The generator addon menu

## A.2 Generating animations

Now that all the requirements are met, the animation can be generated. There are two main steps to generating. Firstly, use the analyzer module to create a JSON file instructions for the generator. Secondly, use the generator to assemble the animation.

### Analyze script

The input script must resemble the script in figure **??**
The characters must be specified after 5 tabs. The dialogue text is specified after 3 tabs. The file must end with an ‘ENDSCRIPT‘ with no indentation. For reference please refer to either:

- Example script files in folder ‘movies‘
- Movie scripts hosted on `www.imsdb.com`

The script now can be process using the following command:

‘python analyzer/script_analyze.py path_to_dialogue_script db/animation.db‘

The program will output a file called ‘scene.json‘. This file is used to generate the animation.

```
                                SANDRA
Sonny?  Sonny, who is it?
What is it?

                                SONNY
They shot the old man.

                                SANDRA
Oh God...

                                SONNY
Honey...don't worry.  Nothing else
is going to happen.

ENDSCRIPT
```

**Figure A.2:** An example of system's input

## Generate Animation

If you did not follow section A.1.2, please follow these steps:

- Open the file 'rendered.blend' with Blender
- Click 'Run Script' on the bottom of the scripting view (figure A.3)
- The tab called 'Generate NLP anim' should appear on the right hand side menu of the 3D view
- Click on the 'Generate NLP anim' tab

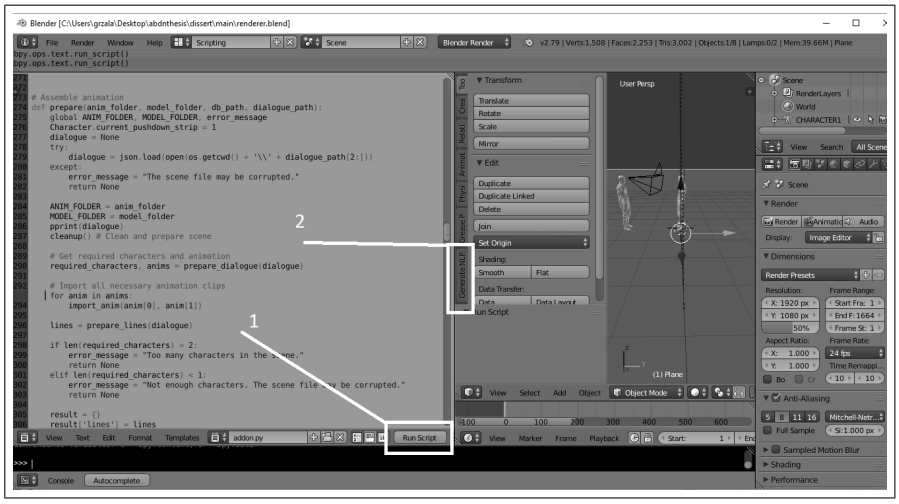


**Figure A.3:** Running the generator without installing it as an add-on

Please follow these steps only if you installed the generator as an add-on:

- Open Blender

- Press ‘File > Save‘ and save the file in a preferred location
- **Important:** close Blender and reopen the saved file
- Open ‘Generate NLP anim‘ tab

After finishing the above instructions, do the following to finalize the animation:

- Press ‘Clean‘ at the top of the menu
- Choose the directory ‘mocap‘ for ‘Animations Directory‘
- Choose the directory ‘models‘ for ‘Models Directory‘
- Choose the file ‘db/animation.db‘ for ‘Database‘
- For ‘Dialogue‘ choose the ‘scene.json‘ file you created when following the section A.2
- Press ‘Prepare‘ and wait until a ‘Finalize‘ menu appears below the ‘Prepare‘ button
- Assign models for the characters from the drop-down lists
- Press ‘Assemble‘ and wait until done

The animation is now finalized. It can now be viewed, edited or rendered. Please refer to Blender official documentation and tutorials for more information on this.

## A.3 Using Custom Animation and Models

This section describes how to extend or replace the animation database.

To insert a new animation file, ensure that the file is saved as .fbx file and contains only one animation (action). Please make sure that the armature is compatible with the one found in ‘EMBD importer/armature.fbx‘ file.

To continue you will need to install ‘DB Browser for Sqlite‘. Use that program to open the file ‘db/animation.db‘. Browse the table ‘animations‘ and add a new record. Provide a name for the animation file (please note the names must be unique) and a path to the animation file. You must also fill the emotion values for the animation (set to 0.0 if the emotion is not carried by the animation).

To insert a new model, browse the ‘models‘ table. Again, provide a unique name and a file path for the .fbx model. You will also need to provide information on how to position and rotate the camera to focus it on the characters face (relative to the origin of the character model).
You can use your own database file as long as the database keeps the following structure:

- ‘animations‘ table:
  - id (Integer, unique, primary key)
  - name (Text, unique)
  - file (Text)
  - Frames (Integer)

  - anger (Numeric)
  - joy (Numeric)
  - sadness (Numeric)
  - fear (Numeric)
  - disgust (Numeric)
  - surprise (Numeric)

- ‘models‘ table:
  - ID (Integer, unique, primary key)
  - name (Text, unique)
  - file (Text)
  - cam_offset_x (Numeric)
  - cam_offset_y (Numeric)
  - cam_offset_z (Numeric)
  - cam_rot_x (Numeric)
  - cam_rot_y (Numeric)
  - cam_rot_z (Numeric)

Keep the emotion values from 0.0 to 1.0.

## A.4 Importing More Animations From EMBD