# 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 challange 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 characters and actions performed by characters.

**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 cutscenes. The system that yielded the best results so far is an in-house tool developed by CD Projekt Red while working on The Witcher 3.

Those animated sequences can be realistic enough to feel natural and be believable but it is not feasible to create them by hand. They have to generated automatically at least in part. The most successful state-of-the-art attempt at this is the conversation system used in The Witcher 3. The tool created by CD Projekt Red takes information on initial state of involved characters (position, pose, 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). The tool I propose makes generating the scenes much faster and enables non-animators to create dialogue scenes. However, 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. These are some serious limitations especially for small developers. [3]

## 2.2 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.

### 2.2.1 Generating Animated Scenes for Training and Instructions

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) [1]. 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.

### 2.2.2 Animation From Text and Motion Database Framework

This research project explores a topic much closer to mine. It does not put any emphasis on emotions or gestures (just actions), however it proposes a usage of motion capture database [2].

## Chapter 3

# Analysis

The project methodology, development tools and technologies and project risk assessment are defined in this following 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.
- Find and assemble a set of relevant motion capture animated clips.
- Prototype a module that uses processes the script and outputs a structured file that can be used to generate the animated scene.
- 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.

My goal is to first build a very minimal prototype of the program. The prototype will focus on emotions, work with a small amount of animations and keep the characters gesturing, but not moving or performing other actions throughout the scene. Afterwards I will focus my effort on making the outputted scenes more realistic and intricate.

## 3.2 Technologies and Resources

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.

The motion capture clips will come from two sources - the Carneige Mellon University motion capture database and mocapdata.com. I will hand-pick relevant animatio clips and preprocess

them so that they can all be easily used with a character model. I will create a SQLite database that will store metadata about the animations (associated emotions, length, etc).

The last part of the project - assembling the final animated scene - will be done using Blender; an open source 3D modelling and animation software. The first module will create a json file that specifies a sequence of dialogue lines and animation clips accompanying them. A Blender extension will read that file and import necessary character models and animations to create a fully editable animated scene.

## 3.3 Risk Analysis

| Risk | Mitigation | Level |
|---|---|---|
| Time delays caused by workload/illness | Follow the project plan closely. Focus on developing a minimum working prototype first. | High |
| Natural Language Approaches are too inaccurate | Use more structured software. Try to find other approaches to this problem. If there seems to be no solution, I can use the findings and existing software to argue that the technology has not yet reached a level that would make this project viable. | low |

**Table 3.1:** Risk Assessment

## Chapter 4

# Requirements Specification

This section describes functional and non-functional requirements of the proposed software.

## 4.1 Functional Requirements

**FR1 Analysis of a semi structure 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 3 kinds of information - characters involved in a scene, dialogue lines and actions performed by characters during the scenes. The software must be able to extract actions and emotions from the script.

**FR2 Find relevant animation clip**

The software must take information about character's emotions and actions and choose animation clip that best represent's characters behaviour.

**FR3 Assemble final scene**

The software takes information generated by other modules and outputs an animated scene. The outputted scene must be fully editable, enabling various adjustments before rendering.

## 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).

## 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
- Anger
- Fear
- Sadness
- Disgust
- Surprise

These are six basic human emotions as specified by Paul Ekman and Wallace V. Friesen. For each animation and dialogue line the system assigns a value between 0 and 1 for each of those

emotions in order to indicate whether a given emotion is present in an action/text and how intense that emotion is. Any combination of those emotions may be used to achieve more specific results.

This set of six basic emotions is generally agreed to be satisfiably accurate in most cases. A lot of existing software (such as IBM Watson Tone Analyzer) and lexicons use those exact emotions.

## 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.

## 5.4 Input

The input of the system looks 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`)

## Chapter 6

# Frequently asked questions

In addition to the information provided in chapter 1, here are some brief notes on references (see section 6.1) and figures (see section 6.2).

## 6.1 References

You can, of course, use any referencing style you like such as `plain`. The `natbib` package, however, allows you to do this with named style citations:

| | |
|---|---|
| `\citet{key}` | Jones et al. (1990) |
| `\citet*{key}` | Jones, Baker, and Smith (1990) |
| `\citep{key}` | (Jones et al., 1990) |
| `\citep*{key}` | (Jones, Baker, and Smith, 1990) |
| `\citep[chap. 2]{key}` | (Jones et al., 1990, chap. 2) |
| `\citep[e.g.][]{key}` | (e.g. Jones et al., 1990) |
| `\citep[e.g.][p. 32]{key}` | (e.g. Jones et al., p. 32) |
| `\citeauthor{key}` | Jones et al. |
| `\citeauthor*{key}` | Jones, Baker, and Smith |
| `\citeyear{key}` | 1990 |

## 6.2 Figures

To include an encapsulated postscript or PDF file (depending on whether you're using LaTeXor PDFLaTeX) as a figure, do something like the following. Note, to ensure correct cross-referencing, it is best to include the figure label within the caption definition. *Note that the graphicx package is already loaded and used to include the University crest on the title page.*

```
\begin{figure}
 \begin{center}
   \includegraphics{myfigure.pdf}
   \caption{This is my figure.\label{fig:mylabel}}
 \end{center}
\end{figure}
```

## 6.3 Frequently used symbols

In LaTeX documents where you want to use a modality or some text consistently in normal text and in equation environments it is often difficult to remember to typeset the text consistently or time-consuming to keep typing in the environment. It may be a good idea to define something like the following in the preamble (i.e. before `\begin{document}`):

```
\def\sfthing#1#2{\def#1{\mbox{{\small\normalfont\sffamily #2}}}}

\sfthing{\PP}{P}
\sfthing{\FF}{F}
```

Then use it in text or math mode. In all cases it looks the same; e.g.

```
\PP\ refers to something, and other things are \FF; $\Phi = \PP\cup\FF$
```

is typeset as:

P refers to something, and other things are F; i.e. $\Phi = \mathsf{P} \cup \mathsf{F}$

Note that you need to put "\" after the command if you want a normal space after it.

# Bibliography

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

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

[3] 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`.