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 (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].

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).

This tool makes creating the generating 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.

---

[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/

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:

- The script needs to be written anyway. Feeding it into the program directly is very quick.

- 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.

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 create very crude, unpolished scenes. 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 given current technology.

Chapter 2

Background and Related Work

ur mom gay [2] [1]

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 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

FUCK mE DDAAD

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. $\Phi = P \cup F$

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] Piskorski, J. and Yangarber, R. (2013). Information extraction: Past, present and future. In Poibeau, T., Saggion, H., Piskorski, J., and Yangarber, R., editors, *Multi-source, Multilingual Information Extraction and Summarization*, Theory and Applications of Natural Language Processing, pages 23–49. Springer Berlin Heidelberg.