Animating Realistic Eye Motions

Dana Moore*
Computer Science
Winthrop University

Sophie Joerg†
School of Computing
Clemson University

Abstract

Animating eyes in a realistic manner is a difficult task to accomplish. In this paper we propose two approaches to animate 3D eyes. The first method uses procedural animation by utilizing a series of commands programmed through Python scripting. The second method is data-driven animation, and makes use of eye-tracking and motion capture technologies to animate our model. In the process of each method we will evaluate which features of eye motions (eyeball rotations, eyelid saccades and blinks) are most important for the creation of convincing eye motions. Through these two approaches, we determine the importance of human eye movements in virtual characters.

Figure 1: 3D head model used in the study.

*e-mail:moored29@winthrop.edu
†e-mail:sjoerg@clemson.edu

1 Introduction

The generation of realistic virtual characters has become a requirement underlying many applications in human-machine interactions [RSB+14]. The way we relate to an animated character is centered around an important attribute: the eyes. People want to interact with a character they feel is relatable to their natural environment. A face model appears natural only if it produces eye movements consistent with human ocular behavior [GLBB07]. Realistic head models must display eye motions similar to that of humans if one wanted them to appear normal. Investigating methods for doing so can increase the realism and likeability of a virtual character. Different programming languages are used in creating animations, however, we implement our procedural approach in Python.

In motion capture, movement is monitored through the use of markers on placed on objects or people. [GD13]. However, information from these devices are not exactly what tracks eye movement. Eye-tracking is more involved, using methods such as eye imaging that uses a binocular camera to to track pupil eye location [Ker01]. Data obtained from this method provides eye motion used in animation. When eye-tracking systems like Erogeneer's Dikablis, an infrared-based gaze tracking device, and Vicon's Tracker, a set of motion capture software, become compatible and integrated, they enhance the ability to use eye-tracking data for animation purposes. [GD13].

We introduce two methods: animation through calculated rules, and data from Dikablis and Vicon, an implement them in Autodesk Maya. These methods are important in accurately depicting eye motions for animations.

2 Background

2.1 Eye Motions

Eye motions feature saccades, eyelid saccades, and eye blinks. Saccades are rapid movements of both eyes from one gaze position to another [LZ06]. They are the only eye movement that can be readily, consciously, and voluntarily executed by human subjects. These quick movements at a swift 40 degrees/sec, [GLBB07] make it difficult to recreate eye rotations procedurally. Eyelid saccades are quick jerk movements of the eyelids from one position to another with the exception of no horizontal movement. They move vertically 5ms after the eye saccades, but peak velocity at the same time [BF88]. Eye and eyelid saccades can be characterized by their magnitude, direction and velocity. The magnitude (amplitude) of a saccade is how far, or what angle the eyes/eyelids rotate to. Saccade direction describes the eye rotation in polar coordinates. Human eye blinks include a down and up phase, with the down phase twice the velocity of the up phase [EMS91]. On average, humans blink 10 times every minute, or once every 6 seconds at a speed of 300-400 milliseconds [TCMH11].

2.2 Measuring eye motions

Prior to implementing eye animations, one must measure/calculate the rotations. Eye rotations can be evaluated through electro-oculography (EOG), magnetic search coil technique, and corneal reflection to name a few. Electro-oculography devices are used by placing markers on the face to follow muscle movement [GD13]. Used in past years, the search coil technique monitored lid position and velocity. Our method for measuring eye motions with Dikablis use video to track the corneal reflection and pupil center.

2.3 Animation Approaches

Different types of research exist where methods for recreating eye movement are proposed. Data-driven, procedural, a hybrid of data-driven and procedural and statistical are some categories for animation. Data-driven approaches have been used with statistical model generated from eye-tracking data [GLBB07]. The data-driven is often a favored method because it implements animated eye motions with information collected straight from the eye itself, using an eye-tracker. The Gu et. al approach was based on empirical models of saccades, which is verified by observation and experience. These estimates could result in faultiness, producing an inaccurate animation.

Research that involves a procedural method usually focus on gaze shifts [NBF+13]. These models are based on existing physiology and sociology literature, which can also result in inaccurate animations.

3 Methodology

3.1 Python in Maya

Autodesk Maya, a 3D animation software, supports the use of Python-style scripting. The implementation of Python scripting in Maya provides an easy way of functionality in Maya. It is a simple way for Maya to perform actions with the graphical interface, saving time while producing similar effects. The Python bindings for all of the Maya commands are in a maya.cmds module [Aut13]. In order to successfully code Python in Maya, maya.cmds must be imported

in the script editor. The script editor lets you enter multi-line scripts in Python and view its output in the history pane (see Figure 2). Once the module is imported, one has the ability to access Maya's commands and also create their own functions to program an animation.

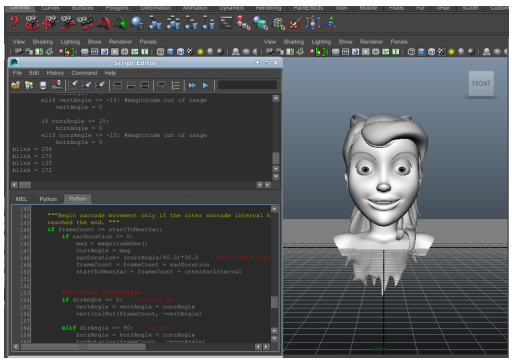

Figure 2: Screen capture of Autodesk Maya. The script editor is displayed left of the 3D model.

3.2 Eye-Tracking and Motion Capture

The eye-tracker used is a customized edition by Erogeneers (see Figure 3) called the Dikablis. It is a lightweight binocular head unit with unlimited head movement. The tracker is operated using D-Lab 3.0 software, which is ran on a Dell desktop computer. There are three cameras: right eye, left eye and field view. Our device had been altered to include markers for motion capture. The motion capture system consist of a total of 14 Vicon cameras (6T40S cameras and 8 MX40 cameras) suspended from the ceiling of a motion capture lab. Vicon Tracker (ran on a Windows XP machine) is the software used in combination with the Dikablis Eye Tracking system which allows gaze data to be collected.

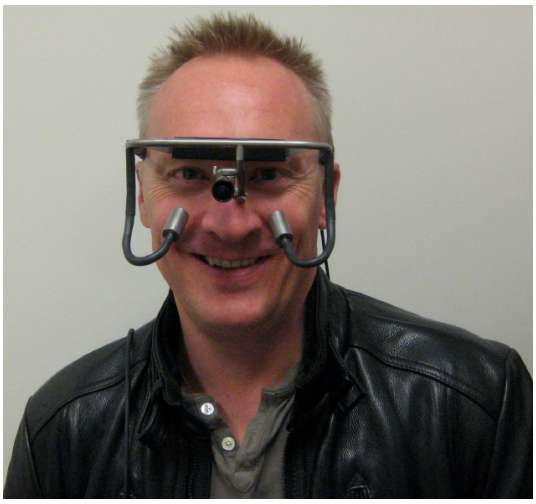

Figure 3: Example of the Dikablis eye-tracker being worn.

4 Design

The first approach, procedural animation, was composed of numerous steps to produce a relatively natural-looking animated model. To begin, we created an algorithm to be scripted in Maya. The algorithm is as follows:

1. Determine saccade and eyelid direction
2. Determine saccade magnitude
3. Calculate saccade duration based on magnitude
4. Rotate eyes and eyelids within eye range based on calculated data
5. Calculate blink occurrence

From eyelid movements, saccades, and multiparty conversations, Gu informs us of the percent chances the eyes will rotate to a particular position. Up-down

and left-right movements occur more than twice as often as diagonal movements [GLBB07]. Once obtaining eye and eyelid direction, we calculate how far they will rotate in that direction (magnitude). Gu's paper also produces the percent chance of occurrence the eyes will move to a specific angle. Next, the eyes are rotated based on its calculated magnitude. Following the calculations of saccade direction, magnitude and duration, a Python command is called to animate the eyes based on this information. While steps 1-4 of the algorithm are ongoing, there is a separate function which randomly animates blink movement. Once the script was run, the 3D model is automatically animated with a time frame of 25 seconds.

Following implementation, 8 ten second mini clips of the animation were created. There were a total of 4 conditions: original, no blink motion, no saccade motion and no eyelid saccade motion. Each condition had a total of two clips.

In data-driven animation there are less manual calculations. It involves an experiment on an individual doing a specific task. Prior to recording, Dikablis and Vicon are calibrated separately. After calibration, the participant will be asked to read a one page paper with a topic different from the actual investigation. When they complete the reading task, the recording from Vicon Tracker is played back to save the frames to a flat file. The flat file is then processed through a Python script in Maya, to automatically keyframe the appropriate objects in the model. The resulting animation will display natural eye motions based on the recorded data. See Figure 5 for details.

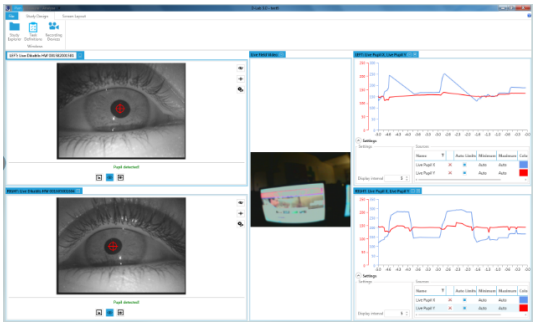

Figure 4: Screen capture of live feed eye-tracking from Dikablis.

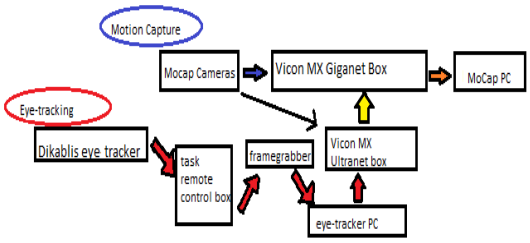

Figure 5: Diagram of the data-driven animation process.

4.1 Procedure

For each condition there were two separate clips viewed by the participants in random order. The participants were prompted to watch each clip, then rate (on a scale of 1-7) how realistic the character appeared based on its eye motions. The students tested were not aware of the variations in each clip.

5 Results

Following the testing of 5 students, data revealed realism was rated highest in the original clip, no eye-

lid saccades, no blinks, and no saccades respectively. Averages are displayed in Figure 6. Technical issues prevented the addition of recorded data from Dikablis onto the virtual character.

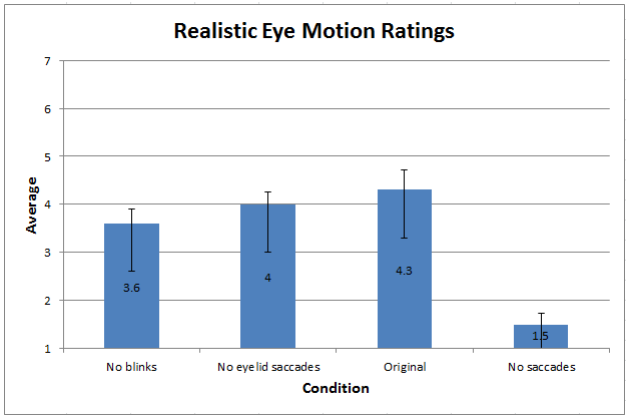

Figure 6: Survey results from the procedural study.

6 Conclusion

Procedural and data-driven animation each have their own advantages and challenges, but can be very helpful for an animator. Despite the difficulties, eye-tracking can recreate lively-looking figures in films or motion pictures. It minimizes the calculations necessary for creating eye motions. Both approaches can be reused, however eye-tracking is an efficient way to complete future investigations dealing with real realistic eye animations.

Acknowledgments

I want to thank Distributed Research Experiences for Undergraduates (DREU) for granting me the opportunity to conduct great research this summer. Next, Dr. Joerg for mentoring me through the process. Also, I want to thank Dr. Duchowski for his assistance with eye-tracking.

References

[Aut13] Autodesk. Scripting, 2013.

[BF88] W. Becker and A. F. Fuchs. Lid-eye coordination during vertical gaze changes in man and monkey. *Journal of Neurophysiology 60*, 4:1227–1252, 1988.

[EMS91] C. Evinger, K.A. Manning, and P.A. Sybony. Eyelid movements. *Investigative Ophthalmology Visual Science*, 32(2), 1991.

[GD13] G.B. Guerrero and A. Duchowski. Animating Eyes. *Class report, Clemson University*, 2013.

[GLBB07] E. Gu, S. Lee, J.B Badler, and N.I. Badler. *Data-Driven 3D Facial Animation*, chapter Eyelid movements, saccades, and multiparty conversations, pages 79–97. 2007.

[Ker01] I.V. Kerlow. The Art of 3D:. *Computer Animation and Effects*, 2001.

[LZ06] R.J Leigh and D.S Zee. The neurology of eye movements. *Oxford University Press*, pages 1227–1252, 2006.

[NBF+13] A. Normoyle, J.B. Badler, T. Fan, N.I Badler, V.J Cassol, and S.R. Musse. Evaluating perceived trust from procedurally animated gaze. *Proceedings of Motion on Games*, pages 119:141–119:148, 2013.

[RSB+14] K. Ruhland, Andrist S., J. B. Badler, C. E. Peters, N. I. Badler, M. Gleicher, B. Mutlu, and R. McDonnell. "Look me in the eyes": A survey of eye and gaze animation for virtual agents and artificial systems. *Eurographics State-of-the-Art Report (EG '14 STARs)*, 2014.

[TCMH11] L.C. Trutoiu, E.J. Carter, I. Matthews, and J.K. Hodgins. Modeling and Animating Eye Blinks. *ACM Transactions on Applied Perception*, 8(3), 2011.