Virtual Puppetry: Real-time gestural control of 3D facial models

by

Rithvik Gambhir

A Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

Supervised by

Dr. Joe Geigel
Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

December 2016
Virtual Puppetry: Real-time gestural control of 3D facial models

Abstract

The Leap Motion controller has created a new way that people interact with their computers. Using hand gestures, users now have a way of interacting with their computers and virtual objects without having to touch a keyboard or mouse. With the Leap, this project aims to explore controlling 3D facial models for applications in medicine, education and entertainment.
Chapter 1

Introduction

1.1 Problem Description

Virtual reality technology has had a tremendous growth in the recent decade with applications in gaming, training and simulations. Immersing oneself in a virtual reality environment using avatars is an exciting way to experience the virtual space. With shared virtual environments, users can communicate, collaborate on training and simulation projects and share experiences in multiplayer games.

The primary objective of this project is to explore the use of hand gestures to control face models in a virtual 3D space. While there exist methods to control characters in a virtual environment, these methods do not quite capture the expression that the user would like to convey as these are limited to keyboard presses and mouse clicks.

With the Leap Motion, a whole new idea of using hand tracked data in the form of gestures is available that lets the user use simple hand gestures to interact with the computer. This opens the possibility of communication in the form of avatars in virtual space using a touch-less interface.

1.2 Related Work

In [1], Liang et al explored the use of a gesture based interaction system for digital storytelling. Here, a novel multiplayer game was designed where children would develop their narrative skills by sharing experiences and ideas in a virtual environment. Using hand gestures captures by a depth sensor camera, young children could manipulate virtual puppets to develop a story.

In [2], a full body character motion control interface was developed inspired by the puppet mechanism. The system allowed for controlling the character’s head, pelvis, legs and arms. While this work allowed for complete character control, it required training to use the system effectively.
1.3 Project Goal

The goal of this project is to discover gestures that can control the face model for communication using avatars in virtual reality environments. In this project, an interactive facial manipulation interface inspired by puppet mechanism was explored. The project explores the use of the Leap motion controller to control the face model using hand gestures.

Inspired by the puppet mechanism, the current work focusses on using puppet-like motions to control a head model. A leap motion controller was used as the hand tracking device. The gestures explored to control the model are simple to reciprocate and the primary goal of this project is to be able to make these manipulations consistently and with very little training with the interface. The system also makes use of the leap motion controller for tracking hand data sparing the user from investing in expensive facial motion capture systems.

1.4 Background

1.4.1 Leap Motion Controller

The leap motion controller (Figure 1) is a 80mm x 30mm x 11.25mm sized device that can connect to a Mac or PC using the USB port. It can be placed on a tabletop or mounted on a VR headset. The controller uses optical sensors and infrared light to track the hands and fingers to report position, velocity and orientation. It has a field of view of 150 degrees and an effective range of 1 inch to 2 feet above the device.

1.4.2 Animation

Computer assisted animation has been a topic of research for many years now. Though devised for 2D animation, the principles of animation have still remained to present day 3D animation. Only key framing animation will be discussed here as it is used in this project.

Keyframe animation is based on interpolation. Keyframes are defined at specific time intervals (say 10, 20, 30 and so on), and the in-betweens are filled so as to make
the sequence complete. These frames are shown in rapid succession creating the animation. Traditionally done manually by artists, the concept of key frame animation is still the same except that the process of filling the in-betweens can now be done on a computer using a specific interpolation method as required by the animator.

Figure 1: Leap Motion Controller (https://developer.leapmotion.com/documentation/v2/unity/unity/Unity_Overview.html)
Chapter 2

Project Design and Architecture

2.1 System Overview

In this section, an overview of the system, the hardware and software components used for the project are discussed. The inputs to the system are hand gestures captured by the Leap Motion controller and the output is animated head model. The leap enabled application, Unity, must run in the foreground in order to receive the hand tracked data.

Figure 2: Leap system architecture (https://developer.leapmotion.com/documentation/v2/unity/devguide/Leap_Architecture.html)

2.2 Hardware
The Leap motion controller connected to either a PC or Mac is the only hardware needed for the project.

2.3 Software

The game engine used is Unity 5.2 Personal edition, Leap Motion v2.3.1 software for development. Additionally Leap Motion core assets for Unity [4] were used for the hand models and the sand box. 3DS Max 2016 software is used for creating all the required bone-based keyframe animations.
Chapter 3

Implementation

3.1 Experimental Setup

In this section, the experimental setup is discussed. The foreground Leap enabled application runs on the Unity game engine and the main scene is set up as follows.

3.2 Sand Box

A closed sand box i.e., walls on all sides is centered in the scene. This acts as the boundaries inside which the Leap Motion Controller class tracks the activity/gesture that the user performs and tracks these patterns. The sand box includes a hand model from the Leap Motion assets for Unity. The hand model used for this experiment was the PolyHand1 model as it was aesthetically pleasing, fully rigid, dark colored for best visual appearance against the bright sand box and head model in the scene. Scripts to animate the head model depending on the input gesture were attached as a component to this object.

A white directional light with intensity set to 1 was used as the primary light in the scene.

3.3 The Head Model

The rigged head model (Figure 3) obtained from [3] was used in all experiments. The model (shown below) has 178,553 polygons and 94,328 vertices was imported to 3DS Max 2016. Bone-based animations were explored in 3DS Max. The bones available for bone-based animations are shown in the following screenshot from the Layer Explorer in 3DS Max.

Several bone-based animations such as those involving movements to the whole head (such as turning left and right, looking up and down), eye blinking and mouthing
certain letters were explored. These were set using Auto set key feature in 3DS Max. Using this feature, the user has to only select the keyframes while Max interpolates the in-between frames in order to create the desired animation clips. Animations were created and are exported along with the complete head model to Unity. These

Figure 3: Head Model (obtained from [5])

animations are attached to the head model game object in Unity.

3.4 Playing the scene/game in Unity

The animated model is produced when running the scene in Unity with all the gestures enabled through a script attached to the head model game object. One must
make sure that the Leap Motion controller is properly connected and the hands are visible in the scene. Making one of the enabled gestures with the virtual hand within the sand box with the Leap motion controller attached results in the animated head model.

When a gesture is recognized, it is added as a Gesture object to the frame gesture list. The gestures supported by this class and the ones used in this project are the following:

a. swipe - Swiping the hand either in the left or right direction  
b. circle - Making a circular movement by a finger  
c. screen tap - Making a forward tapping movement by a finger  
d. key tap - Making a downward tapping movement by a finger

The above gestures are mapped to the head movements as follows

a. swipe - turns the head to the left or right depending on the swipe direction  
b. circle - causes the model to blink once  
c. screen tap - makes the head model bounce back and forth once  
d. key tap - makes the head look down and come back up

Due to the fact that mesh animations are not supported for the current head model, expressions on the head model such as that of anger, surprise etc were not explored in this project.
Chapter 4

Results

In this section, a few screenshots of the animated head model are shown.

Figure 4: Head model in the sandbox (hands not shown)

In the above image, a screenshot was captured just before and after the circle gesture was recognized resulting in the model blinking.

For the above capture, the MinRadius and MinArc values are set to 10f and 5f respectively. These values allowed the user to comfortably and consistently perform the circle gesture for the above effect.
Chapter 5

Conclusion and Future Work

5.1 Conclusion

With the Leap Motion and Unity software installed, the project can be tested by playing the scene/running the scene in Unity. With the Leap enabled, the swipe, tap and circle gestures produce one of the above animations. The gestures in the frame are stored as a Gesture object in the frames gesture list and the associated mapped animations on the head model.

The main problems with controlling models using the puppet-like mechanism is the learning curve involved in using the system and reciprocating the gestures consistently to achieve the results. The current work takes this into consideration and provides basic gestures to effectively control the model.

5.2 Future Work

Since the current head model only allows for bone-based deformations, only these animations and gestures to control these animations were explored. The system is fairly simple to use and requires little learning curve to use even for first-time Leap users.

Manipulating the meshes on the face to create expressions such as anger, happiness, frowning etc and discovering intuitive gestures to animate the model accordingly could be direction for future work. However, adding more gestures would further increase the problem of correctly reciprocating the gesture so that it may mapped to the correct animation. There’s also the problem of recall i.e., too many gestures would make it harder for the user to remember all the possible gestures in the application.
References


