JESTER: A DEVICE ABSTRACTION AND DATA FUSION API FOR SKELETAL TRACKING SENSORS

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Kevin Schapansky

June 2014

COMMITTEE MEMBERSHIP

TITLE:	Jester: A Device Abstraction and Data Fusion API for Skeletal Tracking Sensors
AUTHOR:	Kevin Schapansky
DATE SUBMITTED:	June 2014
COMMITTEE CHAIR:	Professor Zoë, Ph.D., Department of Computer Science
COMMITTEE MEMBER:	Assistant Professor Chris Lupo, Ph.D., Department of Computer Science
COMMITTEE MEMBER:	Professor Franz Kurfess, Ph.D., Department of Computer Science

ABSTRACT

Jester: A Device Abstraction and Data Fusion API for Skeletal Tracking Sensors

Kevin Schapansky

Humans naturally interact with the world in three dimensions. Traditional personal computers have relied on 2D mice for input because 3D user tracking systems were cumbersome and expensive. Recently, 3D input hardware has become accurate and affordable enough to be marketed to average consumers and integrated into niche applications. Presently, 3D application developers must learn a different API for each device their software will support, and there is no simple way to integrate sensor data if the system has multiple 3D input devices. This thesis presents Jester, a library that aims to simplify the development and improve the accuracy of 3D input-supported applications by providing an easily-extensible set of sensor wrappers that abstract the hardware specific details of capturing skeletal data and fusing sensor data in multiple 3D input device systems. Jester is then successfully used to create a toy application that uses real time skeletal tracking and sensor fusion of data from a Leap Motion and a PrimeSense Carime.

ACKNOWLEDGMENTS

Thanks to:

- My advisor Zoë Wood, for being literally the coolest.
- My parents for money and love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Hardware mice have been the dominant way that humans interact with the digital world ever since the first personal computers that introduced the concept of interacting mainly with a two dimensional graphical user interface instead of a text based terminal. Computer mice are very simple to manufacture and are an effective tool for interacting with strictly two dimensional applications such as word processing or managing an email inbox because they are bound to two dimensional sensing surfaces like a table top or flat laptop trackpad. Traditional computer programs and operating systems are designed to follow the Windows, Icons Menus, Pointer (WIMP) computer interaction paradigm. WIMP attempts to make a good digital analogue for a physical desktop where documents and folders are essentially managed in two dimensions [citation needed].

Even though using a mouse for WIMP-style two dimensional applications is a fairly simple task that even small children master quickly, it quickly becomes a handicap when trying to manipulate 3D applications, games, or even general computer navigation tasks that do not necessarily conform to the WIMP style, such as switching workspaces or web browser tabs [citation needed]. Since humans live in a 3D world by default, the human body itself makes an excellent 3D input device, just like how a computer mouse bound to sensing a two dimensional table top makes an excellent two dimensional sensing device.

1.1 Gesture HCI

Humans live every day of their lives in a 3D world. Our bodies are 3D and every physical object with which we interact is 3D. This makes the human body, and generally the hands, a natural choice for interacting with computers in a more intuitive manner. For example, when a user first attempts to interact with 3D Computer Aided Design (CAD) software with a mouse, there is a steep learning curve before he can fluidly and correctly navigate through the world. Different mouse or keyboard buttons are generally used to differentiate between scaling, rotating, and translating the object that is being designed [citation needed]. Application specific key combinations are not innately intuitive, even if they do become second nature after extended use of the software.The users hands are a perfect input choice because they are the interface he has been using to interact with the world around him ever since he was able to correctly place a triangle shaped block in a triangle shaped hole. Tracking the users hand and finger position allows object manipulation in CAD software to become innately intuitive through the use of standard grabbing, rotating, and pushing or pulling gestures [citation needed].

Nintendo introduced the concept of games that are controlled by the human body to millions of users in 2006. The Nintendo Wii includes sensors that can detect the position of users hands and comes bundled with a simulated sports package that allows users to use their arms to intuitively play digital sports [citation needed]. In 2011, Microsoft brought marker-less gesture interaction to mainstream gaming with the introduction of the Kinect for Xbox [citation needed]. The Kinect uses a technology called structured light, which will be discussed in the next chapter, to track the position of a players entire body.

Human body tracking allows the users to control the Xbox using arm movements without picking up a controller or even play a game that allows users to compete against each other in a virtual dance off [citation needed].

Gesture interaction has also been applied to tasks that do not have as clear of a mapping to real life as CAD or video games. For example, gestures can be used to easily switch between browser tabs or navigate web pages [citation needed]. Also, work is being done on creating 3D file browsers where users can shuffle through digital documents and spreadsheets just like they would with physical paper. The task of arranging and moving files between folders can also be intuitively modeled using grab, move, and drop gestures that simulate arranging physical objects [citation needed].

1.2 The Human Skeleton

In order to accurately track the position of a humans body it is necessary to pick some form of conceptual model that can accurately represent it. Muscle and connective tissue systems are extremely complex, and would be very difficult to represent intuitively or with any reasonable accuracy. Eliminating muscles and tendons leaves the skeletal system as the only remaining option. The human skeleton is an excellent choice for representing body position since it is the frame to which every other part of the body attaches. It is not necessary to track all 206 bones in the body since many bones can be approximated as a single bone, such as the bones in the feet or spinal column, or, like the ribs, are simply not necessary to establish the position of a person. A simplified skeletal rigging system is frequently used in computer animation to recreate how muscle and skin move in response to changing skeletal position [citation needed]. The reduced

skeleton is both conceptually simple and can be used to accurately represent body position. Most 3D input devices have associated APIs that provide data about either skeletal joints, which can be trivially used to infer bone position, or direct bone position.

1.3 3D Input Devices

Historically, 3D input systems have been cumbersome and very expensive. They required complex sensor systems that either mounted to the body, or relied on the user wearing a skinsuit covered in motion markers [citation needed]. There are also early version that use computer vision techniques, but they were notoriously inaccurate and sensitive to lighting and user clothing [citation needed]. The expense of setting up a sensor room or the hassle of wearable instrumentation essentially guaranteed that these systems were only used in academic research or high budget special effects.

Within the last two years, there has been an explosion of fairly accurate and consumer-affordable 3D input devices. In 2011 Microsoft unveiled the Kinect for the Xbox 360 [citation needed]. The Kinect is capable of fairly accurately tracking the users torso and limbs, but not finer details like fingers. Several other devices like the Leap Motion Controller, released in 2013, and the Creative Senz3D, released in 2014, are capable of accurately tracking hands and fingers. Finger tracking sensors generally have a limited field of view and range so they cannot track a users entire body. There are also hardware-based sensors like the Razer Hydra or the Oculus Rift that are capable of tracking with very high precision and minimal setup time. Each of these sensors has its own unique API for querying skeletal data.

1.4 3D Applications

Many of the initial applications designed to take advantage of 3D input devices were games and targeted to be more entertaining than useful. There are several games for Xbox that use the Kinect to judge a users dancing skill [citation needed]. Several PC games have also been ported to use the Oculus Rift and Razer Hydra to provide a more immersive gaming experience [citation needed]. Some scientific visualization and CAD software has begun to support the Leap Motion Controller, but a majority of its applications are still games. There has been work on using the Leap Motion and other short range finger sensing devices to navigate desktop workspaces or web browsing, but it is mostly experimental. All of these applications only support a small subset of the sensors that could be used to accomplish the same level of control.

1.5 Jester's Contribution

The human body is a proven method for intuitively and accurately interacting with simulated 3D environments and new and affordable 3D input devices targeted at consumers are flooding the market. However, there is still a significant barrier preventing 3D applications from taking off. Developers must individually tailor their applications to support specific sensors. Also, since sensors that are good at tracking details can rarely see a users whole body and wide angle sensors cannot see enough detail to accurately track fingers, developers must custom craft a data fusion system if they want the best features of both sensors. Jester provides an easily extensible library that simplifies the process of creating an application that can use 3D input devices by providing a framework for creating thin-wrappers to map the skeletal data of any sensor into Jesters internal model

of the skeleton as well as a framework for filtering and fusing data from multiple sensors. Application designers only need to write a thin wrapper for their sensor of choice that maps positional data to any bone or joint that is being tracked. Their application the has access to a world space skeleton and effortlessly adapted to use any available 3D input device.

CHAPTER 2

Background

Jesters core function of providing a solid framework for passing data from a sensor to a skeletal model requires only an understanding of how 3D positions can be transformed into different coordinate spaces and how quaternions handle rotations. The peripheral technologies like ranging solutions and filtering algorithms also warrant explanation. These technologies will be briefly explained in the following subsections.

2.1 Spatial Mathematics

Jester stores its skeletal knowledge as a hierarchy of bones where each bone stores its position and orientation relative to its parent bones coordinate space. It also provides the ability to set bone positions from skeletal joint coordinates in arbitrary space. The design and implementation of the bone hierarchy and joint transforms will be discussed in sections [section needed] and [section needed]. Understanding sections [section needed] and [section needed] will require a basic understanding of matrix spatial transforms and quaternions.

2.1.1 Spatial Transforms

The space of an object is the 3D extents within which other objects have relative position. In the cartesian style spaces used in Jester, each dimension of the space

is specified by a 3D vector, or basis vector, that is defined in its parent space. All of Jesters basis vectors are orthonormal, meaning they are orthogonal and unit length. Orthonormal bases simplify the process of reasoning about skeletal positioning because they guarantee that all of the dimensions can be measured with the same units. Spaces do not have to be specified in this manner, but more complex spaces are unnecessary for this application and lie outside the scope of this thesis.

Jester spaces are specified in the traditional 3D computer graphics style where the x, y, and z basis vectors, a vector specifying the x, y, and z translation of the new space within its parent space, and another basis vector that, for the scope of this explanation, allows the matrix to be used for affine transformation are combined to form a 4x4 matrix, or transformation matrix, that fully defines the new space. A 3D point specified in the new space can be converted, or transformed, into the corresponding 3D point in the parent space by multiplying the point by the transformation matrix of the new space. In order for the multiplication to be possible, a fourth dimension must be added to the point. The fourth dimension is known as the homogeneous component and allows points and vectors to be handled correctly. Similarly, a 3D point can be brought from the parent space to the child space by multiplying the point by the inverse of the childs transformation matrix.

2.1.2 Quaternions

Quaternions are a complex mathematical construct that effectively stores a 3D axis and the amount of rotation around that axis. They are less intuitive, but they allow rotations to be combined without the possibility of the gimbal lock problem that can occur when specifying a rotation as an euler rotation which is

a combination of separate cascading rotations about the x, y, and z axes. Gimbal lock occurs when rotation about one axis causes the other two axes to be on the same plane. If two axes are coplanar a degree of freedom has essentially been lost. Figure [figure needed] demonstrates how a 90 degree rotation about the z axis causes the y and x axes to both specify rotation about the y axis.

2.2 Sensor Types

There are many different sensors that have begun to become popular in the consumer 3D input device market. They can be broadly broken into two categories: devices that require the user to wear trackable markers or sensors, and devices that use some form of imaging technology to observe the scene. Both categories contain many sub-types. Some current commercial examples will be examined and broadly explained. Jester is capable of working with any sensor that can produce data about a joints or bones position and orientation. The process of using the sensor with Jester will be explained briefly below. More detail about the Jester API and its use can be found in sections 4 and 5.

2.2.1 Wearable Sensors

Wearable sensors require that either trackable objects or the actual sensor itself is placed on the body part that the user wishes to track. They have the advantage of being very resistant to interference from ambient light, and are generally more accurate than other approaches. The trade off is that there is more setup time involved to place sensors and each bone or joint requires a separate sensor or marker.

Sixense STEM

The Sixense STEM system can track up to 3 remote tags, STEM packs, and two handheld controllers in the x, y, and z dimensions as will the roll, pitch, and yaw, known as six degrees of freedom, using magnetic field sensors (6 DOF) [citation needed]. Relying on magnetic field rather than some sort of visual sensor means that the STEM system is impervious to any sort of lighting or UV interference. Using the STEM system with Jester would be as simple as getting the controller and STEM pack data from the Sixense API, associating the data points with the correct Jester bones or joints, and calling the newData() function on the Jester Controller.

Oculus Rift

The Oculus Rift is primarily a head mounted display, but it accurately detects head orientation using gyroscopes in order to provide the ability to look around the virtual environment. The orientation returned by the Oculus API could easily be assigned to the head bone and then fused with the head position given by some other sensor in the Jester data fusion framework to provide both head orientation and position with minimal overhead.

PrioVR

Yeti Technology is set to start delivering the first units of its new PrioVR system in Fall of 2014. The PrioVR system provides a set of Inertial Measurement (IMU) sensors that are designed to be attached to all of the main extremities on the user. The sensors reconcile their positions against the position of the main sensor unit which is mounted near the sternum to establish the pose of the user. PrioVR is

perfectly suited to send data to Jester since it already senses the position and orientation of all of the major bones that are tracked by Jester.

2.2.2 Imaging Sensors

Imaging sensors capture the light reflected off of the user in order to determine his position. Some emit their own light in order to simplify the process of ranging and others rely entirely on ambient light. Therefore, all of them are sensitive to ambient lighting conditions and can be rendered ineffective by either too much ambient light or, in the case of passive sensors, not enough. They do not require markers to be placed on the user and can be designed to have very minimal setup times. Some examples of imaging sensors and their consumer implementations are discussed below.

2D Cameras

Traditional 2D cameras can be used to gather skeletal data. Extensive research has been done in attempting to subtract irrelevant background from 2D images and isolate the desired features. However 2D imaging techniques remain very sensitive to differences in ambient light and background color or texture. Due to these limitations, 2D cameras are not frequently used in consumer 3D user input devices that expect and reasonable level of accuracy.

Structured Light

Structured light sensors work by emitting a pattern of infrared light. When the pattern strikes objects, it distorts based on the angle of the object and its distance to the sensor. The sensor has a camera that is designed to only respond

to infrared light that senses how the emitted pattern is reflected off of the user. The sensor then internally maps the emitted pattern to the distorted pattern to figure out patterns transform. The transform gives the distance to that particular part of the image. The sensor combines all of the distances into a depth image where grayscale color corresponds to the distance to that particular pixel. If any ambient light source produces more infrared light than the emitter, structured light sensors lose accuracy very quickly.

The PrimeSense Carmine and the Kinect are both examples of structured light sensors. They have an effective range of approximately 1 to 3 meters and have supporting libraries that map the output depth image to an estimated model of the users joint positions. These joint positions can be fed directly into Jester and Jester will reconstruct the position of the skeletal bones using known mappings from joints to bones. The Jester example program presented in this thesis uses the PrimeSense Carmine and will be discussed in more detail in section [section].