Design and Implementation of an Interactive Animatronic System for Guest Response Analysis

Brian Burns

Follow this and additional works at: https://digitalcommons.georgiasouthern.edu/etd

Part of the Biomechanical Engineering Commons, Electro-Mechanical Systems Commons, and the Interactive Arts Commons

Recommended Citation
http://thebrianburns.wordpress.com/category/dragon-kronos/

This thesis (open access) is brought to you for free and open access by the Graduate Studies, Jack N. Averitt College of at Digital Commons@Georgia Southern. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of Digital Commons@Georgia Southern. For more information, please contact digitalcommons@georgiasouthern.edu.
DESIGN AND IMPLEMENTATION OF AN INTERACTIVE
ANIMATRONIC SYSTEM FOR GUEST RESPONSE ANALYSIS

By
Brian Christopher Burns
(Under the Direction of Biswanath Samanta)

ABSTRACT

In theme park based entertainment applications, there is a need for interactive, autonomous animatronic systems to create engaging and compelling experiences for the guests. The animatronic figures must identify the guests and recognize their status in dynamic interactions for enhanced acceptance and effectiveness as socially interactive agents, in the general framework of human-robot interactions. The design and implementation of an interactive, autonomous animatronic system in form of a tabletop dragon and the comparisons of guest responses in its passive and interactive modes are presented in this work. The dragon capabilities include a four degrees-of-freedom head, moving wings, tail, jaw, blinking eyes and sound effects. Human identification, using a depth camera (Carmine from PrimeSense), an open-source middleware (NITE from OpenNI), Java-based Processing and an Arduino microcontroller, has been implemented into the system in order to track a guest or guests, within the field of view of the camera. The details of design and construction of the dragon model, algorithm development for interactive autonomous behavior using a vision system, the experimental setup and implementation results under different conditions are presented. Guest experiences with the dragon operating in passive and interactive configurations have been compared both quantitatively and qualitatively through surveys and observations, for different age groups, from elementary school children to college students. Statistical significance of the survey results are presented along with a discussion on the scope of further work.

INDEX WORDS: Animatronics, Human-to-Machine Interaction, Image processing, people-tracking, Arduino, Guest response analysis, Infrared camera, human-robot interaction
DESIGN AND IMPLEMENTATION OF AN INTERACTIVE ANIMATRONIC
SYSTEM FOR GUEST RESPONSE ANALYSIS

by

BRIAN CHRISTOPHER BURNS

B.S., Georgia Southern University, 2013

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial
Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

STATESBORO, GEORGIA

2015
DESIGN AND IMPLEMENTATION OF AN INTERACTIVE ANIMATRONIC SYSTEM FOR GUEST RESPONSE ANALYSIS

by

Brian Christopher Burns

Major Professor: Biswanath Samanta

Committee: Brian L. Vlcek
Christopher Kadlec

Electronic Version Approved:

May 2015
DEDICATION

To Adam, Kevin, Eric, David, Mom and Dad.
ACKNOWLEDGEMENTS

I would like to acknowledge Justin Hinckley for his craftsmanship on the sculpture of Kronos the dragon. Dr. Biswanath Samanta for the support as my advisor and providing the research opportunity, Dr. Brian Vlcek for support and convincing me to stay to complete my graduate degree, Dr. Christopher Kadlec for taking time to review my thesis and make suggestions, Andrew Michaud, Spencer Harp, and Rex Miller for providing necessary assistance for fabrication. I would also like to acknowledge the following for being a helping hand at some point during the process: David Burns, Eric Burns, Rebecca Whitlock, Urmil Patel, Mallory Morgan, Ashley Striplin, Chris Reid, Derrick Herrin and Jonathan Turner.
# TABLE OF CONTENTS

**LIST OF FIGURES** .......................................................................................................................... x

**ABBREVIATIONS** ............................................................................................................................... xv

**Chapter 1: Introduction** ....................................................................................................................... 1

1.1 Problem Statement .............................................................................................................................. 1
1.2 Hypothesis ........................................................................................................................................... 1
1.3 Contributions to Knowledge .............................................................................................................. 3
1.4 Organization of Thesis ....................................................................................................................... 4

**Chapter 2: Literature Review** ............................................................................................................... 5

2.1 Overview ........................................................................................................................................... 5
2.2 Human-Robot Interaction .................................................................................................................... 5
2.3 Animatronics ...................................................................................................................................... 5
2.4 Themed Entertainment Industry ......................................................................................................... 10
2.5 Psychology of Entertainment ............................................................................................................. 13
2.6 Current Interactive Entertainment Research ...................................................................................... 14
2.7 Hardware and Specifics ...................................................................................................................... 17
2.7.1 Human Identification ................................................................................................................... 17
2.7.2 Arduino Mega 2560 ................................................................................................................... 19
2.7.3 ROS and Processing .................................................................................................................. 19
2.7.4 Dragons ....................................................................................................................................... 20

**Chapter 3: Method** ................................................................................................................................ 21

3.1 Method Overview ............................................................................................................................... 21
3.2 Hyperion - Animatronic Dragon Prototype ....................................................................................... 21
3.2.1 Hyperion Aesthetic Design ......................................................................................................... 21
3.2.2 Hyperion Mechanical Design ..................................................................................................... 26
3.2.3 Hyperion Control ....................................................................................................................... 32
3.2.4 Hyperion Feedback .................................................................................................................... 33
3.3 Kronos Aesthetic Design and Fabrication ......................................................................................... 33
3.3.1 Kronos Sculpture ....................................................................................................................... 38
3.3.2 Kronos Mold ............................................................................................................................ 43
3.3.3 Kronos Casting .......................................................................................................................... 47
Appendix D: Processing code for Kronos. .............................................................. 126
Appendix E: Arduino code for Kronos. .................................................................. 138
Appendix F: Mini Maestro passive behavior script.................................................. 146
LIST OF FIGURES

Figure 1: Table from the 2012 Theme Index. Only two out of the top twenty theme parks in the United States experienced a reduction in attendance from 2011 to 2012 (TEA/AECOM 2012). ................................................................................................................................. 12

Figure 2: Sony’s AIBO, the animatronic pet. Photo credit: Sony (Carnoy 2010) ............ 6

Figure 3: Lucky the dinosaur, developed by Walt Disney Imagineering, became the first free-roaming animatronic. Photo credit: happysteve via photopin cc ......................... 8

Figure 4: Animatronic T-Rex from the film Jurassic Park (Jurassic Park's T-Rex - Sculpting a Full-Size Dinosaur 2012). ........................................................................................................ 9

Figure 5: Screen capture of an animatronic creature created by willettfx. Video link: http://www.youtube.com/watch?v=5RMZZ97ssQ8 ......................................................... 10

Figure 6: Maslow's Hierarchy of Needs ........................................................................ 13

Figure 7: Animatronic playing catch sequence............................................................... 15

Figure 8: PrimeSense Carmine 1.08. Photo credit: PrimeSense 3D Sensors ............... 18

Figure 9: Sketch for the first animatronic dragon .......................................................... 22

Figure 10: The clay sculpture for the first animatronic dragon ....................................... 23

Figure 11: Releasing clay head from the mold ............................................................... 24

Figure 12: Casting head with latex ................................................................................ 25

Figure 13: Casting of the head and jaw for the first dragon ........................................... 25

Figure 14: The castings of the dragon were painted using latex paint ......................... 26

Figure 15: Mock-up for neck mechanism for the first dragon ....................................... 27

Figure 16: Simplified sketches to solve for servo arm length ...................................... 27

Figure 17: Servomotor assembly to pull cables around arc ........................................... 29
Figure 18: Neck mechanism for the first animatronic dragon. ................................................. 30
Figure 19: The first dragon mechanism without aesthetic components. ............................ 31
Figure 20: The first dragon with aesthetic components.......................................................... 31
Figure 21: Hardware flow diagram for Kronos 1.0. ................................................................. 32
Figure 22: Hyperion appearance at the STEM Festival 2013.............................................. 78
Figure 23: Concept drawing for Hyperion............................................................................. 34
Figure 24: Inspiration from a painting made by “rah-bop” on deviantART (rah-bop 2013). ................................................................................................................................ 35
Figure 25: Sculpting, molding, and casting process................................................................. 37
Figure 26: CAD model for the table and top surface. Dimensions are in inches. .............. 38
Figure 27: Duct-tape wrapped Styrofoam body for Kronos. Photo taken January 22, 2014. ........................................................................................................................................... 39
Figure 28: Early aesthetic design showcasing the dragon sizing on the table. January 27, 2014................................................................................................................................... 40
Figure 29: The sculpture begins to take shape as Justin Hinckley lies on slabs of clay. March 14, 2014 .......................................................................................................................... 41
Figure 30: Staining the table. April 27, 2014 ........................................................................... 41
Figure 31: Justin detailing the sculpture by adding scales. September 6, 2014................. 42
Figure 32: The final clay sculpture. September 8, 2014 ....................................................... 43
Figure 33: Applying silicone mold over the clay sculpture. September 9, 2014............ 44
Figure 34: The sculpture with the complete silicone mold. September 11, 2014.......... 44
Figure 35: Mother mold shells to retain the overall shape. September 23, 2014 .......... 45
Figure 36: Mold applied to the head sculpture. November 7, 2014 ................................. 46
Figure 37: Removing the clay from the mold. September 23, 2014.......................... 46
Figure 38: November 7, 2014. ............................................................................. 47
Figure 39: Building additional support structure around the mold. October 28, 2014..... 48
Figure 40: Mold flipped to remove paper and prepare for casting. October 28, 2014. .... 48
Figure 41: Mold ready for polyurethane foam. October 30, 2014................................. 49
Figure 42: Expanded Flex Foam It III inside the body mold. October 30, 2014......... 50
Figure 43: Removal of the casting from the body mold. October 30, 2014. ............... 51
Figure 44: Internal mechanisms wrapped in tape for use in casting process. February 8, 2015.......................................................... 52
Figure 45: Foam casting inside the mold of the head. February 22, 2015................... 52
Figure 46: Kronos foam testing with mechanism movement. March 1, 2015.............. 53
Figure 47: Kronos testing foam with mechanism movement. March 1, 2015.......... 53
Figure 48: Kronos foam patching. Head was trimmed and wing claws were added. March 6, 2015.............................................................. 54
Figure 49: Kronos after painting and re-assembly. March 8, 2015. ......................... 55
Figure 50: Underneath the table.............................................................................. 56
Figure 51: Tail motor mechanism.......................................................................... 57
Figure 52: CAD file for the head assembly. ............................................................... 59
Figure 53: Laser cut parts for the head. February 6, 2015........................................ 60
Figure 54: Eye blinking 3D printed parts. February 7, 2015.................................... 60
Figure 55: The head assembly. February 7, 2015..................................................... 61
Figure 56: CAD of the wings.................................................................................... 62
Figure 57: The laser-cut wings installed on Kronos for testing. February 26, 2015. ..... 62
Figure 58: CAD Model of the neck. ................................................................. 63
Figure 59: The completed neck mechanism for Kronos. February 28, 2015. ............ 63
Figure 60: Head, neck and lift assemblies fully installed. March 1, 2015............... 64
Figure 61: The body and tail support structures and mechanisms. March 1, 2015. .... 65
Figure 62: The tail lifting mechanism ............................................................... 65
Figure 63: The internal structure without the wings attached .............................. 66
Figure 64: Face tracking with OpenCV ............................................................. 67
Figure 65: Testing of the Depth Image program through Processing ..................... 69
Figure 66: Servomotor assembly to test head tracking calibration. Note: bottom servo motor not used ............................................................. 70
Figure 67: Control flow diagram for head tracking servomotors ............................ 71
Figure 68: Schematic for the head tracking servomotor setup .............................. 72
Figure 69: Control Flow Diagram .................................................................... 73
Figure 70: Overhead layout of Kronos and camera field of view ............................ 74
Figure 71: The Smile-Scale used for kids ages 12 and under .................................. 75
Figure 72: Burnt H-Bridge after an hour of continuous use. March 12, 2015 ........... 80
Figure 73: Screen capture of tracking the closest pixel. A red circle was drawn over the pixel ......................................................................................... 81
Figure 74: Tracking a person by coloring the space blue ....................................... 82
Figure 75: Tracking the head and center of mass of the person in the image ......... 83
Figure 76: Calibration curve for the pan motion .................................................... 84
Figure 77: Calibration curve for the tilt motion ..................................................... 85
Figure 78: Calibration surface curve for the neck bend ...................................... 87
Figure 79: Calibration surface curve for the neck pivot. .................................................. 87
Figure 80: Neck bend calibration curve two................................................................. 90
Figure 81: Head Rotate Calibration Curve .................................................................. 91
Figure 82: Computer system display tracking the person............................................. 92
Figure 83: Kronos real time tracking right. ................................................................. 93
Figure 84: Kronos real time tracking left................................................................. 93
Figure 85: Kronos real time tracking. Angry when guest is in close proximity.......... 94
Figure 86: Kronos sleeps when there are no moving people in the environment........ 95
Figure 87: Results from feature comparison............................................................. 96
Figure 88: Results from Passive versus Interactive Behavior Comparison.................. 98
Figure 89: Kronos located at LES for survey testing. March 12, 2015...................... 107
Figure 90: Kids petting Kronos asking him to wake up during passive behavior. March 12, 2015.............................................................. 108
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-IRIS</td>
<td>Biologically Inspired Robotics and Intelligent Systems</td>
</tr>
<tr>
<td>GSU</td>
<td>Georgia Southern University</td>
</tr>
<tr>
<td>OpenCV</td>
<td>Open Source Computer Vision</td>
</tr>
<tr>
<td>OpenNI</td>
<td>Open Natural Interaction</td>
</tr>
<tr>
<td>ROS</td>
<td>Robot Operating System</td>
</tr>
<tr>
<td>RGB</td>
<td>Red, Green, Blue</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>IR</td>
<td>Infrared</td>
</tr>
<tr>
<td>SLS</td>
<td>Selective Laser Sintering</td>
</tr>
<tr>
<td>RC</td>
<td>Remote Control</td>
</tr>
<tr>
<td>LAC</td>
<td>Linear Actuator Control</td>
</tr>
<tr>
<td>STL</td>
<td>Stereolithography (file format)</td>
</tr>
<tr>
<td>DXF</td>
<td>Drawing Interchange Format</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

1.1 Problem Statement

For nearly half a century, animatronic figures have provided entertainment in the theme park industry by simulating life-like animations and sounds. These figures enhance the storytelling experience by stimulating visual and audio senses in audiences. While animatronic technology has improved tremendously over the past years, producing smoother motions and more realistic sounds, the majority of animatronic experiences are passive. The robot creatures and humanoids seen in theme parks and other entertainment venues generally operate on pre-programmed scripts. Interactivity in animatronics is a fairly recent endeavor. With the size and number of themed entertainment venues on the rise, the market is becoming increasingly competitive. New technologies must be employed in the theme park environment to better engage guests in stories in effort to create more entertaining and compelling experiences. The design and implementation of an interactive, autonomous animatronic system in form of a tabletop dragon and the comparisons of guest responses in its passive and interactive modes are presented in this work.

1.2 Hypothesis

The hypothesis for this research on interactive animatronics was twofold. The hypothesis statements are as follows:

- A dynamic, people-aware, autonomous, animatronic system can be designed and fabricated for the purposes of testing the effectiveness of interactive elements.
Interactive, animatronic behaviors will provide a more entertaining experience amongst guests than the passive behaviors.

While some interactivity does exist today in animatronics, such as Pleo and Lucky the dinosaur, mechatronic technologies can be implemented to further the guest experience (Brandon n.d.). In fact, as of March 2015, when a web search is conducted through Google for “Interactive Animatronics”, the portfolio of the author of this research shows on the first page of results (Burns 2015). The first hypothesis is broken into three main objectives. These objectives are listed and explained below:

1. Develop a robust people tracking control system. The control system must be able to import image data, locate people in the field of view, and translate coordinates into physical outputs.

2. Design and fabricate an aesthetically pleasing, tabletop autonomous dragon model. The animatronic aesthetic elements must hide mechanical components. The features must also be able to bend or actuate in accordance with mechanical movement.

3. Design and fabricate the mechanical system. This includes the mechanisms, housings and hardware components.

The successful completion of these three objectives test the first hypothesis to design and fabricate a dynamic, people-aware, animatronic system. The second hypothesis is tested by running trial studies with guest volunteers to determine if the interactive behaviors rank higher than passive behaviors. Survey questions include specifics about the realism, level of fun, life-likeness, and overall entertainment value.
1.3 Contributions to Knowledge

The research presented contributes knowledge in multiple arenas. First, a fully documented design and fabrication method for an animatronic system has been produced in the form of a tabletop dragon model. The system consists of an interactive, autonomous dragon model, designed and fabricated in the course of this work, and a vision system with a depth camera, see Figure 1. The dragon model is named Kronos. The relevant hardware and the desktop running the control software are housed underneath the table of the mobile platform.

Figure 1: The Kronos Animatronic System.

A method for identifying people and using that information has been established and implemented as an autonomous control system using a popular microcontroller,
namely, Arduino. Second, interactive and passive behaviors have been identified to provide greater entertaining experiences amongst guests.

1.4 Organization of Thesis

The rest of the thesis is organized as follows.

The literature review is presented in Chapter 2. The literature review covers background information on animatronics as they pertain to themed entertainment. Individual components, such as hardware, are documented as these pieces relate to the research.

The methodology for completing the research is discussed in Chapter 3. The methodology covers the prototype and successor animatronic systems. Each of the animatronic systems are divided into several subsections consisting of aesthetic design and fabrication, mechanical design and fabrication, the control system, and conducting guest interaction trials.

The results and analysis of the methodology are discussed in Chapter 4. Like the methodology, the chapter contains results from the prototype and successor animatronic systems. The four main components of the research (mechanics, control, aesthetics, and guest feedback) are presented.

An overview of the results and recommendations for future work are presented in Chapter 5. Additional documentation related to the research is provided in Appendices. The appendices include a list of publications, sample surveys, updates to the system, codes and scripts.
Chapter 2: Literature Review

2.1 Overview

The literature review begins with background information on Human-Robot Interaction (HRI). Afterwards, background information on animatronics is provided followed by information on the themed entertainment industry. Afterwards, more specific tools and systems are discussed regarding hardware and current research developments related to interactive electro-mechanical systems.

2.2 Human-Robot Interaction

Human-Robot Interaction (HRI) is a field of study “dedicated to the understanding, designing, and evaluating robotic systems for use by or with humans” (Goodrich and Schultz 2007). Features of HRI addressed include a natural exchange of information with exhibited intelligence and the ability for the robot system to operate dynamically in an unpredictable environment. HRI can be implemented in a variety of applications such as human-machine interaction and surveillance. Artificial emotions are being implemented in robots in order to develop behaviors from interacting with people (Castro-Gonzales, Malfaz and Salichs 2013). Interactive animatronics is an example of HRI. Therefore, many of the techniques of HRI can be applied to interactive animatronics.

2.3 Animatronics

The word “animatronics” originated in the 1970’s and is defined by the Oxford Dictionary as “the technique of making and operating lifelike robots, typically for use in
film or other entertainment” (Oxford Dictionaries 2013). Mechatronics is a word combination of the two disciplines electronics and mechanical engineering. After adding a third component, animation, the result is the word “animatronics”. Animatronics have been used extensively in the film industry, theme park applications, and toys. The Sony AIBO, Figure 2, is a robot “pet” that was first released in 1999 with new editions appearing until 2005 (Moon 2001). Other animatronic toy products include Furby, and Pleo the Dinosaur (Brandon n.d.). Furby was first released in the 1990’s, selling over 40 million units, and recently, in 2012, Hasbro released a new version (Furby for the Connected Generation 2012).

![Figure 2: Sony’s AIBO, the animatronic pet. Photo credit: Sony (Carnoy 2010)](image)

The term Audio-Animatronic is a registered trademark by Walt Disney Imagineering filed in 1967 (Sreetef and Chakravarthi n.d.). The first Audio-Animatronics that Walt Disney Imagineering, originally called WED Enterprises, created were for The Enchanted Tiki Room at Disneyland, California. The attraction opened in June, 1963 with a cast of Audio-Animatronic birds. Digital controls were used to control the pneumatic valves in the Tiki birds. The design, mechanically driven by air pressure,
limited all animated movements to either an on or off position. For example, the beak was either open or closed, or, a wing was either raised or lowered. Walt Disney, with the rest of the WED Enterprises team, took the knowledge gained from these early animatronics and developed a system with analog programming. The new analog system, with hydraulics, allowed for both an increase in power and varied movement between two extremes (Anderson n.d.). In 1964, Walt Disney introduced a fully animated Abraham Lincoln. The figure had 57 different moves that the animator, Wathel Rogers, programmed using a harness to capture his own movements (The History of Disney's Audio Animatronics n.d.).

Over the following decades, animatronic technology improved allowing for more realistic animations with life-like personalities. In 2002, the first portable and all-electric Audio-Animatronic, Meeko, made his debut in Disney’s Animal Kingdom in Florida. Then, two years later, Lucky the Dinosaur became the first free-roaming Audio
Animatronic as seen in Figure 3 (Kimberly n.d.).

Figure 3: Lucky the dinosaur, developed by Walt Disney Imagineering, became the first free-roaming animatronic. Photo credit: happysteve via photopin cc

Animatronics are also used extensively in the film industry. Some of the most well-known animatronics to be seen in films, according to Jason Serafino, include the Tyrannosaurus Rex from the 1993 release of *Jurassic Park*, Jabba the Hutt from the 1983 release of *Star Wars Episode VI: Return of the Jedi*, and Jaws from the 1975 release of *Jaws* (Serafino 2011). Stan Winston, television and film special effects supervisor, created the full size T-rex for Steven Spielberg’s Jurassic Park as seen in Figure 4. Winston commented on the project, “Steven figured that if we could build a fourteen-foot-tall alien queen, we’d be able to build a twenty-foot-tall T-rex,” (Jurassic Park's T-Rex - Sculpting a Full-Size Dinosaur 2012).
One major difference between film animatronics and theme park animatronics is that the film industry only needs the animatronic to operate successfully long enough to capture the animations on film, while theme park animatronics may experience thousands of cycles 365 days a year. Robust design of mechanical, electrical and outer layers of the theme park animatronic is imperative.

Today, the cost of many technologies has reduced to a level to where hobbyists and animatronic enthusiasts can afford to participate in creature making activities. Halloween, in particular, is a holiday that brings together many of these hobbyists that has sparked online sites such as FrightProps (The FrightProps Crew n.d.). The website includes parts and do-it-yourself instructions for a variety of Halloween animatronic figures. One such animatronic creature, posted by willetfx on YouTube, can be seen in the video screen capture in Figure 5. The initial response to viewing the video caused
GSU alumnus, David Burns, to comment, “That’s creepy.” The creature features blinking eyelids, shifting eyes, moving jaw, and moving lips.

![Figure 5: Screen capture of an animatronic creature created by willettfx. Video link: http://www.youtube.com/watch?v=5RMZZ97ssQ8](image)

### 2.4 Themed Entertainment Industry

The word entertainment is defined by Merriam-Webster as “something diverting or engaging,” (Merriiam-Webster n.d.). Entertainment comes in a wide variety. Some examples include movies, music, gambling, video games, theater, fine dining, museums, board games, sports and theme parks. Entertainment is all around. Themed entertainment is a specific type of entertainment that includes all engaging activities that revolve around a central theme; usually, to immerse guests into a story. Themed entertainment can include museums, zoos, concerts, parties and theme parks. The largest sector within themed entertainment involves theme parks, which are an evolution of fairs
and carnivals. When Walt Disney, pioneer of the first theme park, was asked by his wife, “Why do you want to be in that business? Amusement parks are dirty, unsafe, and the people who work there are nasty!” Mr. Disney replied, “My park won’t be anything like those,” (Imagineers and Malmberg 2010). History has proved Walt Disney correct.

Walt Disney opened the first theme park, Disneyland, in 1955 in Anaheim, California. Today, the Walt Disney Company operates 11 theme parks around the world and is in the process of constructing another in Shanghai, China. In 2012, the top 20 theme parks in the United States accumulated 131.6 million visits resulting in a 3.6% increase from 2011 (TEA/AECOM 2012). A summary is presented in Figure 6.
Figure 6: Table from the 2012 Theme Index. Only two out of the top twenty theme parks in the United States experienced a reduction in attendance from 2011 to 2012 (TEA/AECOM 2012).

Brian Sands, Vice President of Economics for the Americas for AECOM, was quoted with saying “A lot of that growth was, as expected, driven by re-investment – the eternal necessity to drive additional attendance.” The theme park business is becoming increasingly competitive. Parks, such as Universal’s Islands of Adventure, are making large park investments and, consequently, increasing park attendance at a higher rate than
the overall average. Universal’s Islands of Adventure in Orlando witnessed a major, quarter billion dollar, expansion in 2011 resulting in a miraculous 29% attendance increase from 2010 to 2011 (TEA/AECOM 2011). The Walt Disney Company, still with a stronghold on the top five visited parks in the United States, invested over 3.2 billion dollars in their parks and resorts in 2011 (The Walt Disney Company 2011). The theme park industry is thriving, competitive, and the future looks bright.

2.5 Psychology of Entertainment

The goal of a resulting animatronic figure is ultimately to create a compelling, entertaining experience for a guest, or guests. Therefore, psychological and physiological aspects of entertainment are considered during animatronic design. John Wesner, author of *Entertainment Engineering*, refers to Abraham Maslow’s “hierarchy of needs”, from Maslow’s published work in 1954 (Wesner 2013), as shown in Figure 7.

![Maslow's Hierarchy of Needs](image)

**Figure 7: Maslow's Hierarchy of Needs**

Maslow describes how a person must satisfy the needs at the bottom of the pyramid before they will concern themselves with a higher need. Therefore, in an entertainment environment, one must make sure the guest feels safe, welcome and not overwhelmed. Only after these aspects are covered can the guest experience the peak of
the pyramid where entertainment or fun exists. Two things that Wesner says that keep
guests coming back is providing the *Total Guest Experience* and keeping current (Wesner
2013).

Andrew Sempere conducted research, at the Massachusetts Institute of
Technology, on introducing young children to computational concepts. Sempere says
that “faces are intimately familiar objects to all of us. Children, by the age of four,
understand control of their own faces, and are interested in what expressions indicate
sadness, happiness, or anger (Sempere 2003). Therefore, animatronics with facial
expressions have a greater ability to trigger responses in a human.

Guest feedback for this research will include results from younger audiences.
Research on measuring fun has been conducted. The “Smiley-o-meter” and the Again-
Again table were designed as part of a research to best quantify fun for children (Read,
MacFarlane and Casey 2002). When it comes to satisfaction in the themed environment,
emotions can be broken into two components: arousal and pleasure (Bigne, Andreu and
Gnoth 2004). Bigne, Andreu and Gnoth performed research on pleasure and arousal
satisfaction. There were six major areas that were assessed on the questionnaire:
disconfirmation, pleasure, arousal, satisfaction, loyalty, and willingness to pay more
(Bigne, Andreu and Gnoth 2004).

2.6 Current Interactive Entertainment Research

There are current research efforts to create interactive experiences with
animatronics. Disney Research, Pittsburgh, has recently published research for “real-
world human-robot interaction by means of throwing and catching balls between a human
and humanoid robot” (Kober, Glisson and Mistry 2012). The humanoid animatronic
uses an external camera system to acquire the position of the ball and the participant. Ball detection is registered with OpenCV while the participant is tracked using the OpenNI skeleton tracker. Inverse kinematic equations were used to create an algorithm to perform the desired task. One of the catch sequences can be seen in Figure 8.

![Figure 8: Animatronic playing catch sequence. Photo credit: Disney Research (Kober, Glisson and Mistry 2012).](image)

The Disney Research found the interactive animatronic platform successful. They made note that subtle animations in the humanoid cued reactions from the participants. One example of reaction noted was that one of the participants apologized when the humanoid reacted after it dropped the participant’s tossed ball. (Kober, Glisson and Mistry 2012).

Disney Research is also engaging with a technology for creating interactive plants (Poupyrev, et al. 2012). The technology allows a plant to become an input from the environment for experiential, entertainment and aesthetic purposes. The system responds to a variety of gestures which include sliding fingers on plant, touching leaves, and the proximity of touch. The technology can be used with both real and artificial plants.
(Poupyrev, et al. 2012). A snapshot of the plant system identifying touch can be found in Figure 9.

Universal Creative launched interactive wands as part of the *The Wizarding World of Harry Potter - Diagon Alley* expansion for a park expansion (Strickland 2014). The wands utilize an infrared camera system to identify wand gestures, which trigger show elements. The show elements are disguised around the park to add for an immersive, interactive experience.

Another interactive platform being studied is the robotic fish, Commodore, from the New York University Polytechnic School of Engineering (Phamduy and Porfiri 2015). The robotic fish is controlled by user defined path through a tablet. The fish was demonstrated in public with an optional questionnaire. The average enjoyable level was indicated as 4.6 out of 5 stars. The robotic fish was created for advancements in informal science learning. Informal learning is the type of learning that lacks intention and objectives from the learner’s viewpoint (Ainsworth and Eaton 2010).
2.7 Hardware and Specifics

2.7.1 Human Identification

The ability to perceive humans is imperative for human-robot interaction. Commercially available, depth cameras have allowed human perception with robotic applications (Zhang, Reardon and Parker 2013). The Kinect sensor was originally created as a peripheral for Microsoft’s Xbox 360 video game console and released on November 4, 2010. The hardware was developed by the Israeli company, PrimeSense. The Kinect sensor has been utilized in a variety of applications such as HRI and other projects that include object tracking, human activity analysis, hand gestures and indoor three-dimensional mapping (Shao, Xu and Shutton 2013). Unlike an ordinary RGB camera, the Kinect is a depth camera. The Kinect has an IR projector, RGB camera, and an IR camera (Borenstein 2012). While RGB cameras obtain pixel information on the color of light that reached the camera, the Kinect can gather three pieces information on a pixel: x-coordinate, y-coordinate and the z-coordinate values. The RGB camera works like any regular camera with a 640 x 480 resolution. The IR projector emits an irregular pattern of infrared dots of varying intensities. The infrared camera recognizes the pattern and constructs a depth image. Quite similarly, the PrimeSense Carmine, Figure 10, a limited release from PrimeSense, has specifications similar to the Microsoft Kinect except for motor control, body size and the ability to be powered solely by USB. The field of view of Carmine 1.08 is 57.5° × 45° × 69° (horizontal, vertical, depth) and the VGA depth image size is 640 × 480 pixels. The camera supports both color and audio. The range for depth sensing is 0.8m – 3.5m compared to the Kinect’s range of 0.8m to 4 meters (PrimeSense 3D Sensors n.d.).
There are a variety of methods and research projects involving recognition of a person utilizing a color-depth camera such as the Kinect. Current research on optimizing algorithms for head and shoulder recognition of a person is being conducted (Nghiem, Auvinet and Meunier 2012). Tracking just the head and shoulders profile can reduce the processing power for tracking an individual (Lu, Chen and Aggarwal 2011) and (Kirchner, Alempijevic and Virgona 2012). Adaptive methods to tracking the head and arms in cluttered environments are being studied (Huang, Chen and Fu n.d.). The Kinect has also been used to learn motions for sports such as martial arts and dance (Chye and Nakajima 2012). Outside individual algorithms, there are several software tools available for working with data from the Kinect sensor. These tools include OpenNI, Microsoft Kinect SDK and OpenKinect. OpenNI works well with NITE middleware and includes many processing tools for skeletal recognition. The software can also be used in a Windows or Linux operating environment and is also compliant with the Carmine (OpenNI 2014). The Microsoft Kinect SDK includes many built-in processing tools such as face tracking, skeletal tracking and scene detection; however, the software is limited to Windows based systems (Microsoft Kinect SDK 2014).
2.7.2 Arduino Mega 2560

The Arduino Mega 2560 is a microcontroller based on a previous version, the ATmega2560. The board has 54 digital input and output pins, 16 analog inputs, 4 serial ports, a 16 MHz crystal, USB connection, power jack, ICSP header, and a reset button. The microcontroller supports in an input voltage of 7-12 volts and has 256 KB of flash memory for storing code (Arduino 2013). The Arduino language, based on C and C++, is used to program the board through the latest Arduino integrated development environment software. The software is on iteration 1.0.6 and is optimized for Windows, MAC OS X, and Linux. The Arduino serves as the main processor unit aboard the animatronic dragon.

2.7.3 ROS and Processing

ROS (Robot Operating System), while not an operating system, provides similar functionality. ROS is developed by Willow Garage and operates on Linux based operating systems such as Ubuntu. ROS is completely open source and provides libraries and tools for robot related applications (Quigley, et al. 2009). ROS is being used as the primary software for integration and control for the development of a heterogeneous robot swarm system for the B-IRIS (biologically inspired robotics and intelligent systems) laboratory at Georgia Southern University.

Processing 2 is an open source development environment built on the Java language initiated in 2001 (Processing 2 2014). Processing can be used in both Windows and Linux, therefore, allowing cross-over development between the two operating systems.
2.7.4 Dragons

Though the dragon has appeared in many different cultures and time periods, most people believe the dragon to be a mythical creature. Dragons have been described as monsters, serpents and reptiles. Common traits recurring in creature descriptions include wings, fire breathing, scales, claws and horns. Many dragons are very intelligent and live in remote locations. The word “dragon” comes from the Greek word, draconta, which means “to watch.” (Radford 2012).
Chapter 3: Method

3.1 Method Overview

The method is split into two sections. The first section covers the first animatronic prototype, Hyperion. The second section covers the successor, animatronic system called Kronos. The two animatronics will be referred to as Hyperion and Kronos. Each of these sections is broken down into four main parts: aesthetic design and fabrication, mechanical design and fabrication, control and guest feedback. The aesthetic design and fabrication portion discusses the concept and features for the animatronic creature, as well as, the process for creating the aesthetic components. The mechanical section covers the design and fabrication for all mechanisms, as well as sizing, housing, and installation of the actuators, motors, and hardware. The control section of the method covers the method to implement control for the animatronic system. Lastly, the guest feedback section discusses the methods for conducting trials for guest response.

3.2 Hyperion - Animatronic Dragon Prototype

The first animatronic dragon was the result of an undergraduate fund request to design and fabricate an animatronic dragon. The following is an overview of the design, fabrication, and implementation of this first iteration.

3.2.1 Hyperion Aesthetic Design

An original sketch was made to establish a concept of where the project was headed. The sketch can be seen in Figure 11.
The dragon was to be in a lying position with the bulk of the movement coming from the neck, head, and tail. The next step in the process was to create a clay sculpture of the dragon. Monster Maker’s oil-based clay was acquired and the sculpture was created. Gradually, working the clay by hand, the sculpture began to resemble a dragon with more and more detail. Figure 12 displays the final sculpture for the dragon. Wings were not fully sculpted because the clay could not support the weight.
The sculpture was sealed with Krylon Crystal Clear spray paint as a release agent for the molding process. With the clay model complete, the next step in the process was to create the mold. The mold is a negative of the clay sculpture. The process of making the mold involved the use of gypsum cement called Ultra Cal 30. Ultra Cal 30 is rapid setting and has a low expansion rate making it an ideal material for the process. Pre-planning was necessary to decide where to break the clay model to cast the mold in separate pieces. Figure 13 displays the clay head being removed from the mold. Separate mold pieces included the following: head, jaw, left back leg, right back leg, tail, and body.
RD-407 liquid latex was used to cast the skin. The process involved filling the mold with latex, letting the latex dwell, and pouring the excess latex out. The latex filled mold dried completely over the course of approximately 24 hours. Figure 14 displays the head being cast with the liquid latex. Air bubbles in the latex caused minor defects in the castings. All of the latex castings were successful with the exception of the neck. The mold of the neck was inadequate to produce a formidable neck casting. Figure 15 displays the castings of the head and jaw. The latex darkens as it dries, that is the reason for the jaw being darker than the head.
Figure 14: Casting the head with liquid latex.

Figure 15: Casting of the head and jaw for Hyperion.
The Monster Makers latex mask Paint is flexible, bonds to latex, and is mixable. This paint was used to paint the latex castings. Painting was completed by using a brush.

Figure 16: The castings of the dragon were painted using latex paint.

After painting, the latex castings were installed onto the mechanical body.

3.2.2 Hyperion Mechanical Design

The priority of this animatronic figure was to have a neck with a high number of degrees-of-freedom. Cardboard, string, and tape was used to create a mock-up for the mechanism. The tentacle-like mechanism, inspired from a YouTube video posted by user zerocool1027, would allow the neck to bend fluidly. The mock-up can be seen in Figure 17.
Servomotors were acquired to be used for all mechanisms in the project. Lengths of the arms needed to achieve the desired motion of a 90° bend were calculated.

The following equation relates the distance, $d$, that the cable would need to be pulled to achieve a 90° bend with the distance, $r$, between the cable connection and the center of the mechanism.
From measurements of the dragon sculpture, \( r \) was approximately 1.75 inches based on the width of the neck. After calculating necessary arm length, the force required to lift the head was determined, see Figure 19.

![Free body diagram for head weight.](image)

**Figure 19:** Free body diagram for head weight.

The following equation was used to determine the force the servo would need to provide:

\[
\text{Equation 4}
\]

Using an approximate weight of the head as 4 pounds, and the distance from the center of gravity to the pull location as 12 inches the equation becomes:
To achieve the required force, two Hitec 7950TH servos were used to lift the neck. With a two inch arm, each servo provided 12.5 pounds of force. The servomotors were assembled so that the arms would pull a bike brake cables around an arc. An assembly of the motors can be seen in Figure 20.

![Figure 20: Servomotor assembly to pull cables around arc.](image)

Utilizing the design from the cardboard mock-up, a functional neck mechanism was produced, Figure 21. The mechanism had the ability to bend half way into two different directions. The majority of the structure was fabricated out of compressed PVC. Standard bike brake cables were used for the pull-pull cables.
Figure 21: Neck mechanism for the first animatronic dragon.

The final version of the first dragon included the following mechanical motions: opening jaw, rotating head bending neck, lifting neck and shifting arm. Other mechanisms that had foundations included a bending tail, shifting right arm, expanding wings and blinking eyes. The ‘naked’ version of the dragon can be seen in Figure 22.
Figure 22: The first dragon mechanism without aesthetic components.

Plans were made for a fully functional wing mechanism; however, this mechanism was never fabricated. The dragon with aesthetic components is shown in Figure 23.

Figure 23: The first dragon with aesthetic components.
3.2.3 Hyperion Control

Hyperion was outfitted with an Arduino Mega 2560 R3 and Pololu Mini Maestro 24-Channel servo controller. The servo controller contains built-in acceleration and speed control while the Arduino board allowed for external input and logic. The hardware flow diagram for the animatronic can be found in Figure 24.

The Wi-Fi shield and PING sensors were not used in the final program but existed for future implementation. The main source of input for the dragon came from micro switches that were placed on either side of the dragon’s belly. When the dragon was “touched” on the left side, Hyperion would rise and look left. A similar action resulted from the right side. The Maestro receives commands from Arduino using the following protocol:

Serial1.write(0xFF); //Prepares Maestro to receive data for compact protocol
Serial1.write(3); //servo channel number  
Serial1.write(100); //send the input position

For the above protocol, the servo channel number ranges from 0-23 for the 24 channels available on the servo controller. The input position is the target position for the servomotor. The target position ranges from 0-254.

3.2.4 Hyperion Feedback

Hyperion succeeded in simulating a behavior and producing fluid mechanical motions. Hyperion made its public appearance for the 2013 S.T.E.M. (Science, Technology, Engineering, and Mathematics) Festival at Georgia Southern University. The festival hosted hundreds of school aged children to participate in S.T.E.M. activities. Other than this event, Hyperion made other smaller appearances for the GSU Mechanical Engineering department during recruiting events and volunteer activities. There were no formal feedback methods utilized for Hyperion. The qualitative observations that were made from public appearances were taken into account for future designs. These designs are discussed later in this chapter for the Kronos system.

3.3 Kronos Aesthetic Design and Fabrication

Before beginning the animatronic project, a conceptual design was necessary. The requirements set by the hypothesis required the animatronic system to have the ability to identify people and interact with them. The concept for the animatronic system for this research was inspired by the original Hyperion system. The new animatronic system, Kronos, was conceptualized as a non-mobile platform that would have the mechanical capabilities to produce life-like behavior when amongst guests. The new system would be part of a table that housed necessary components for function. The idea
to house hardware underneath came from the experience with the Hyperion prototype. While the motions were smooth and effective, Hyperion suffered from noisy servomotors and a lack of space for additional components. With the servos housed underneath a table, there would be a reduction in mechanical noise and ample space for hardware. The original concept drawing for Hyperion can be seen in Figure 25.

*Figure 25: Concept drawing for Hyperion.*

For Kronos, the concept was to take the original design, Figure 25, and move the dragon into a more relaxed resting as seen in Figure 26. Both of these paintings served as references for the rest of the animatronic design and fabrication process that follows. Kronos sits approximately five feet from the tip of his head to the tip of his tail.
After establishing the concept for the new system, a list of features that were to be included in the design was established. These features were ranked as required for the project or as a bonus add-on. The information on whether the feature was included in the system at the time of this thesis publication can be found in the right column of Table 1. These features were established at the beginning of the research as minimum requirements to succeed in testing the hypothesis. The bonus features can be added in future upgrades of the animatronic system.

Figure 26: Inspiration from a painting made by “rah-bop” on deviantART (rah-bop 2013).
Feasibility for adding the bonus features was determined by constraints on the aesthetic components to achieve desired feature as well as time to install the feature.

After the concept was established for the look and features, a plan was made for the types of materials and processes that would be used for the aesthetic components. The process and materials were derived from multiple phone calls and emails with the Atlanta based company, *The Engineer Guy*, as well as previous experience with Hyperion. The physical form of the animatronic dragon would begin with a rough cut of a Styrofoam form to achieve the desired size and shape. The Styrofoam would not include details such as scales, eyes or claws. Upon the Styrofoam layer, oil based clay was used to sculpt the details of the creature. After the sculpture was completed, the clay

Table 1: List of Features for the Kronos animatronic system.

<table>
<thead>
<tr>
<th>Item Number</th>
<th>Feature</th>
<th>Necessity</th>
<th>Included</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lifting head</td>
<td>Required</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Rotating head</td>
<td>Required</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Bending neck</td>
<td>Required</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Blinking eyes</td>
<td>Required</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Shifting eyes</td>
<td>Bonus</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>Opening jaw</td>
<td>Bonus</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Shifting arms</td>
<td>Bonus</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>Lifting tail</td>
<td>Bonus</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>“Breathing” sides</td>
<td>Bonus</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>Audio</td>
<td>Bonus</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>People tracking system</td>
<td>Required</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Touch response</td>
<td>Bonus</td>
<td>No</td>
</tr>
<tr>
<td>13</td>
<td>Smoke blowing effect</td>
<td>Bonus</td>
<td>No</td>
</tr>
<tr>
<td>14</td>
<td>Expanding wings</td>
<td>Bonus</td>
<td>Yes</td>
</tr>
</tbody>
</table>
was sealed with a coat of clear spray paint. Rebound 25, a soft platinum-cure silicone rubber, was to be painted over all of the details and undercuts to create the *soft mold* (Smooth-On 2011). After the soft mold has cured, Free Form AIR, lightweight epoxy putty, would be placed on the soft mold (Smooth-On 2011). This hard mold is also referred to as the *mother mold*. When the mother mold had cured, the clay and Styrofoam would be removed. Ease Release 200 would then be applied into the mold to cover the soft silicone as a release agent. Dragon Skin, a high performance silicone rubber, was to be painted inside the mold (Smooth-On 2011). This layer would be the final output product. Expanding polyurethane foam, Flex Foam-It III, would be poured inside the mold to create a light, flexible support for the skin. The basic sculpting, molding and casting process can be seen in Figure 27.

![Figure 27: Sculpting, molding, and casting process.](image)

The actual process followed the plan outlined, except the finished casting was not silicone, rather, the casting was polyurethane foam. The reason for this change is discussed in the following sections.
3.3.1 Kronos Sculpture

The first step, in the process of fabricating the aesthetic components of the dragon, was to create a sculpture. The author of this thesis teamed with a student from the College of Liberal Arts & Social Sciences, Justin Hinckley, to bring realism and detail to the sculpture. Before beginning the sculpture, constraints were made on the size. It was determined the table would be semi-circle with the flat edge being the front “stage”. The depth of the table was designed to be 36” in order for the table to fit through standard doorways. The table was designed in SolidWorks and the CAD model can be seen in Figure 28.

![CAD model for the table and top surface. Dimensions are in inches.](image)

After the table size was determined, the dragon dimensions were maximized based on the size of the tabletop. The table, in the above figure, allowed space for a computer as well as a shelf for hardware components. The table was cut and assembled out of pinewood and deck screws. The sculpture began with a duct-tape wrapped, Styrofoam body. The foam reduced the required amount of clay to complete the
sculpture and can be seen in Figure 29. The wings were not included in the main body sculpture because the wings would not be part of the same mold.

The dragon was positioned so that Kronos would appear to be curling in a sleeping position on the table. From experience with the first animatronic endeavor, this version was to be sculpted in the resting position. The figure would “stretch” to other mechanical positions, but the aesthetic components would not be forced into a position while the system was not running. The skull in the above figure was hand carved out of floral foam.

Figure 29: Duct-tape wrapped Styrofoam body for Kronos. Photo taken January 22, 2014.
Monster Maker’s oil-based clay was used for sculpting. The oil-based clay softens when heated. Heating was accomplished by either using a hair dryer or kneading the material in hands. Justin adds clay slabs to the foam body in Figure 31.
As the clay took shape, the table neared completion as well. Sanding and staining were the final processes to complete the table, Figure 32. The table was stained “willow” from the Rust-Oleum branded wood stain and sealed with the Rust-Oleum Polyurethane satin sealant.
The table would undergo additions of holes and other add-ons as the process moved forward. The precise locations for the holes could not be determined until the sculpture was completed.

![Figure 33: Justin detailing the sculpture by adding scales. September 6, 2014.](image)

Justin modeled the scales after a pangolin, which is a mammal found throughout tropical regions of Africa and Asia. The finished sculpture can be seen in Figure 34; however, the head was altered slightly afterwards to reduce the size. The claws were not included in the sculpture because they were to be carved and painted out of wooden dowels. This would add more realism by having harder claws.
With the sculpture finished, the next step in the aesthetic design was mold-making.

3.3.2 Kronos Mold

The mold for Kronos was created with two different materials to satisfy different purposes. The first part of the mold was a soft silicone that would capture the fine details of the sculpture. The second part would be a harder mother-mold to retain the shape. The silicone used for the process was Rebound 25. Rebound 25 comes in two parts that are mixed in equal volumes. The silicone was applied, by brush, in three layers with a 24 hour curing time in between each layer. The sculpture covered with two layers of silicone can be seen in Figure 35. The silicone appears to be wet, even though product has cured. The sculpture with the completed silicone mold can be seen in Figure 36.
Figure 35: Applying silicone mold over the clay sculpture. September 9, 2014.

Figure 36: The sculpture with the complete silicone mold. September 11, 2014.
After excess silicone was removed from the mold, the mother-mold was applied. The mother-mold material was a lightweight epoxy called Free Form AIR. The epoxy came in two parts and was mixed in even ratios by hand to apply to the soft mold. The intention, in Figure 37, was to create many shells that fit together like a puzzle. However, this method proved to make re-assembling the mold into the original shape a bit more challenging.

Figure 37: Mother mold shells to retain the overall shape. September 23, 2014.

The same process was applied to the head of the sculpture, Figure 38. The mother molds for the head fit together nicely for re-assembly. Once the molds had cured, the clay and foam sculpture was removed from the inside, Figure 39 and Figure 40. The molds were cleaned and Ease Release 200 was applied.
Figure 38: Mold applied to the head sculpture. November 7, 2014.

Figure 39: Removing the clay from the mold. September 23, 2014.
Clay from the sculpture was salvaged as much as possible for future use.

3.3.3 Kronos Casting

After the molds were completed, the next step was to cast. Unfortunately, the mother-molds for the body sculpture did not encompass enough of the soft mold to support the entire shape. This fault was remedied by stuffing the mold with paper to achieve a close approximation to the original shape, and building a support structure around the mold, Figure 41. The mold was then flipped, Figure 42, to remove the paper and to prepare for casting.
Figure 41: Building additional support structure around the mold. October 28, 2014.

Figure 42: Mold flipped to remove paper and prepare for casting. October 28, 2014.
Another coat of Ease Release 200 was applied, lightly by brush, to the silicone mold. DragonSkin was the brand silicone to be used for the final casting of Kronos. Three layers were applied in the mold but, unfortunately, not enough of the material was available to achieve a thick enough compound to remove it from the mold without tearing. Due to time constraints, and a test of the polyurethane foam, the decision was made to use the foam as the final casting rather than wait for more silicone. The molds were prepared to accept the polyurethane foam as seen in Figure 43. Blocks of wood were inserted into the mold to approximate the location of the supporting mechanism structure. This reduced the amount of foam needed and the amount of foam that needed to be cut out later in the process.

![Figure 43: Mold ready for polyurethane foam. October 30, 2014.](image-url)
The mold with the expanded foam inside can be seen in Figure 44. The foam begins expanding within a minute of mixing equal parts together. Total cure time was approximately one hour before the casting could be removed, Figure 45.

Figure 44: Expanded Flex Foam It III inside the body mold. October 30, 2014.
Figure 45: Removal of the casting from the body mold. October 30, 2014.

For the casting of the head, the internal mechanisms were taped together, Figure 46, and Ease Release 200 was applied. The mechanism assembly was placed inside the mold. This allowed the foam casting to expand tightly around the mechanism. The expanded foam in the head mold can be seen in Figure 47.
The foam castings were trimmed and fitted onto the external mechanisms. The wing mechanisms were fitted with fabric and stapled to the plywood structure of the wings. The mechanisms were moved and trimming was done to foam parts that did not
flex as intended. Figure 48, Figure 49, and Figure 50 display Kronos in different positions as the foam was analyzed for various trim points.

Figure 48: Kronos foam testing with mechanism movement. March 1, 2015.

Figure 49: Kronos testing foam with mechanism movement. March 1, 2015.
Figure 50: Kronos foam patching. Head was trimmed and wing claws were added. March 6, 2015.

The trimmed foam castings were removed from the internal mechanisms to be prepared for painting.

3.3.4 Kronos Painting and Patchwork

After the foam castings were patched and trimmed, Kronos was ready to be painted. A darker color was chosen to contrast the lighter color of the table. Kronos was sprayed with Krylon Crystal Clear to help the Monster Maker’s latex paint stick to the surface. Painting was achieved with an airbrush gun and air supply. The foam castings and wings were removed for this process. After painting, Kronos was re-assembled onto the table, Figure 51.
After painting, claws and patching around neck areas were completed.

3.4 Kronos Mechanical Design and Fabrication

This section covers the mechanical design and fabrication of the mechanisms for Kronos. Methods for design and assembly are discussed in the following section.

3.4.1 Mechanical Components

Much of the original mechanism concepts for Hyperion were persevered for Kronos. However, the majority of the hardware was placed underneath the table. Figure 52 displays the backside of the table. Following the figure, a description of each of component is made.
From Figure 52, item A is the HB-25 motor controller. The motor controller accepts PWM signals from the servo controller and powers the tail motor based on the response. Item B is the tail motor attached with a belt mechanism. The mechanism can be better seen outside the table as shown in Figure 53. On the larger gear, a ball and socket joint connects to a steel cable that raises though the top of the table for lifting the tail. The mechanism translates a rotational input into a linear output.
Figure 53: Tail motor mechanism.

Item C is an air flow output fan. Item H is an air flow input fan. When all of the panels enclose the table, the air flow seemingly helped keep the servomotors cooler than without the fans. This was based on a recommendation from Hitec that if the servomotors are too hot to touch by hand, then the servomotors are in danger of overheating. Item D is the shaft about which the neck can pivot. A lock collar was used to connect the shaft to a small linear actuator. This mechanism translates linear motion into rotational. Item E contains three switches. The leftmost switch is being preserved for future use but is wired to the system. The middle switch is used as the behavior mode switch. Up puts the dragon into the interactive mode. When the switch is down, Kronos is set into passive mode. The right switch is power for the servomotors from the 6 volt power supply. Item F is an assembly that contains the Arduino board, Mini Maestro, MP4 processor and other inputs and outputs. This is the center of control for the entire system.
Item G is a power strip with a switch that is used to apply power to the entire system. Item I is an assembly for the servomotors that cause the bend in the dragon neck. The blue cables are bike cable housings that lead up through the body to the neck. The bracket, that the motors rest, was 3D printed through the company Shapeways with an SLS (Selective Laser Sintering) 3D printing process. The servomotors contain a pulley wheel that contains the steel cable.

Item J is a Firgelli Actuator. The actuator operates on 12 volts and can deliver 150lbs of force. The end of the actuator contains a simple pulley mechanism that essentially halves the output force in favor of twice the output speed. The actuator has the ability to move 0.5”/second, which was on the slow end of what was visually appealing of the entire neck assembly lifting. Item J is the Linear Actuator Control (LAC) board by Firgelli. The board accepts the RC servomotor input form the servo controller and simplifies the control required by taking care of the potentiometer for the large linear actuator internally. The board controls the large linear actuator for the neck lift.

Item L contains the two servomotors for lifting the wings. The bottom motor operates the right wing, while the top operates the left. The motors are equipped with a small pulley wheel to pull the steel cable that raises the wings. Item M is a USB hub that combines the signals needed for the Arduino board, Mini Maestro servo board, PrimeSense Carmine and speaker power into one USB port to connect with the computer or Wi-Fi adapter. Item N contains two buttons that are used for manually positioning the dragon that override any program that is currently running. The top button resets the dragon to a resting position.
The head assembly contains everything needed for the eye mechanisms, rotating the head, and operating the jaw. The assembly was designed in SolidWorks, Figure 54, so that all of the parts could either be 3D printed or laser cut.

The mesh structure of the dragon head came from a 3D laser scan using the Faro Arm. The head was scanned as an STL file which as imported into the free software, MeshLab. Inside MeshLab, a quadratic reduction filer was applied and then converted into a DXF file format. The DXF file was then imported into SolidWorks for use to design the internal head structure. The laser cut parts and the 3D printed mini-servo motor housing can be found in Figure 55. The 3D printed eye mechanism parts can be seen in Figure 56.
Additional hardware needed for assembly included machine screws, hex nuts, and bell-crank assemblies. A small set of ball joints linked the mechanisms together. The eyes were purchased from an Etsy store online called artistJP. The eyes were listed as
30mm Handmade Glass eye for jewelry making. The head assembly can be seen in Figure 57.

Both of the bottom eyelids are actuated by one servomotor and both of the top eyelids are forced by the other servomotor. After the head was assembled, the mechanism were taped up and used for creating the foam casting of the head.

The wings were also designed utilizing SolidWorks. The wings were designed to be laser cut from plywood. The laser is a CO$_2$ laser. The CAD model can be seen in Figure 58. Two sections of the designed wing were not laser cut because steel cables were used instead because the only necessary action those sections completed was in tension, not compression. The freshly, laser-cut wings can be seen in Figure 59.
Figure 58: CAD model of the wings for Kronos.

Figure 59: The laser-cut wings installed on Kronos for testing.  February 26, 2015.
Then neck disks were designed to be 3D printed. The disks were attached together using door hinges. The SolidWorks design for the neck mechanism can be seen in Figure 60. This design was different than that of Hyperion in that this one can only bend horizontally. The Kronos design greatly enhanced durability and reduced the force required from the servomotors that drive the mechanism. Figure 61 displays the neck fully assembled. The black rods in the upper part of the mechanism add stiffness to the design, so that the neck bends evenly.

Figure 60: CAD Model of the neck.

Figure 61: The completed neck mechanism for Kronos. February 28, 2015.
The head assembly was attached to the neck assembly which was then attached to the lifting mechanism as found in Figure 62.

![Image](image.png)

**Figure 62: Head, neck and lift assemblies fully installed. March 1, 2015.**

The body structure for Kronos is displayed in Figure 63. The cables for the wings are routed to the body and attached at the base of the wing attachment points. The body also contains the swivel pulley that the head lifting mechanism utilizes for routing the steel cable. The neck pivot point is also shown in the figure. The tail lifting mechanism can be found in Figure 64. The cable from the tail motor is routed through a housing to pull the end structure up. A small weight was added to the end of the structure to increase the speed in which the tail fell back down after being lifted.
Figure 63: The body and tail support structures and mechanisms. March 1, 2015.

Figure 64: The tail lifting mechanism.
The entire internal structure, minus the wings, can be seen in Figure 65. The speaker for sound effects sits underneath the center body structure. Much of the support structure for the center body was constructed utilizing Actobotics parts.

Figure 65: The internal structure without the wings attached.

3.5 Kronos Control System

The Control System section is broken into three main categories. The first covers the method for identifying people. The second section covers the hardware for the animatronic system. Lastly, the third section will cover the programming logic that brought the control system together.

3.5.1 Vision System

In order for the animatronic system to interact with people in the environment, the system needed a method of identifying those people. A couple of different identification methods were explored. The first method that was investigated was identifying a human face. The camera used was a high definition webcam by Logitech. The program would
find a face in the field of view then coordinates or possibly expressions could be extracted from the result. The program utilized OpenCV video algorithms to identify and draw a box around the face, Figure 66.

![Face Tracking with OpenCV](image)

**Figure 66: Face tracking with OpenCV.**

The weakness with the face tracking method was that when the face turned away from the camera, approximately 45°, the face was lost. Consequently, this would result in a process that required many cameras, fixed at different angles, to continually identify faces in a given space. This method was quickly discarded.

After researching other possibilities, a common method of people identification utilized the Kinect Sensor by Microsoft. The PrimeSense Carmine 1.08, which is very similar to the Kinect, was readily available and ultimately used as the primary sensor input. The PrimeSense Carmine 1.08 was chosen for a few reasons. First, as previously stated, the sensor was readily available. Second, in addition to the color camera, the Carmine contained an infrared emitter and collector to provide pixel information based
on depth. Depth data has an advantage over color data in many object tracking situations. With two pixels, a simple differentiation between the pixels that is close versus one that is far away can be made. Color pixel data require more than pixel to make this determination.

With the depth cameras, such as the Carmine and Kinect, there were several software tools available for accessing the data from the sensors. These tools included OpenNI, Microsoft Kinect SDK and OpenKinect. OpenNI included many processing tools for skeletal recognition and could be used in a Windows or Linux operation environment. The Microsoft Kinect SDK included many built-in processing tools such as facial recognition, skeletal tracking, and scene detection. However, the SDK was limited to Windows only based systems. OpenKinect contained tools for accessing sensor components but lacked pre-developed libraries for skeletal recognition. For the reasons mentioned above, OpenNI was chosen as the developmental tool to be used to accomplish person detection. OpenNI allows for future implementation with ROS.

The book, Making Things See by Greg Borenstein, was heavily used for the initial utilization of OpenNI and the depth camera (Borenstein 2012). Processing was chosen as the wrapper code where the logic would be written for the OpenNI and NITE libraries. First, the drivers, libraries and software had to be installed. The following were installed in order to access the Carmine through Processing: Processing 2.1.1 -32 bit, NITE 1.5.2.21 for Windows 32 bit, Kinect for Windows SDK v1.7, OpenNI 1.5.4.0 for Windows 32 bit. The “Kinect for Windows SDK” was installed just for the automatic installation of the drivers; however, the drivers could be installed without the SDK. After
installing the software, libraries, and drivers, the sensor was tested to see if a video feed could be produced. The result is pictured in Figure 67.

![Figure 67: Testing of the Depth Image program through Processing.](image)

Next, the first test in building the code was to track the nearest pixel in an image from the depth camera. The images were coming from the infrared collector at a resolution of 640 pixels across and 480 pixels high. The “carmine.depthMap” function stores the pixels as a single array of data. The following pseudo code was used to accomplish the task of locating the closest pixel in an image.

1. Look at each row in the depth image
   a. Look at each pixel in the row
   b. Get depth value for each pixel
      i. If value is less than the current one,
         save the value and position
      else, check the next pixel
      continue checking all pixels in the row
   continue checking all the rows
2. After all pixels have been looked at
   a. The value saved last will be the nearest pixel
3. Draw red circle at x and y coordinates of the nearest pixel
The pseudo-code provided the framework for continued development of the program. The full code is listed in the Appendix. Results for the people tracking are discussed in Chapter 4.

3.5.2 Hardware

A trial was conducted to test the communication from the people tracking software to the physical output devices. This trial run was conducted with a two servomotor system, Figure 68. The bottom servomotor was not used in the test. The assembly allowed for one of the motors to control horizontal motion and the other to control vertical. This gave the cardboard owl face two degrees of freedom at 180° each.

Figure 68: Servomotor assembly to test head tracking calibration. Note: bottom servomotor not used.

The hardware setup for this test can be seen in Figure 69.
Figure 69: Control flow diagram for head tracking servomotors.

The depth sensor was placed so that the 57.5° field of view allowed the person to travel around the servomotor assembly and stay within view of the camera, Figure 70. The x value in the diagram is 60 inches.
The results for the hardware communication test for the pan and tilt system are discussed in Chapter 4.

Much of the hardware for Kronos was salvaged from the previous animatronic dragon, Hyperion. The control hardware includes an Arduino Mega 2560, Pololu Mini Maestro, HB-25 motor controller, Firgelli Linear Actuator Controller, and the Vizic MP4 processor. The hardware flow schematic can be seen in Figure 71. Sensor input hardware includes the PrimeSense Carmine and four switches. The switches allowed for switching between different behaviors and manual positioning of the dragon. The system is powered with two separate AC to DC power supplies by Mean Well. One power supply
outputs 12 volts DC for the tail motor and large linear actuator. The other power supply is a 6 volt DC for all other actuators.

![Kronos Control Flow Diagram](image)

**Figure 71: Kronos Control Flow Diagram.**

### 3.5.3 Calibration

In order for a mechanical output to correctly communicate with a person that was being tracked, the input and output had to be calibrated. Mechanical positions were recorded to create a calibration curve through Microsoft Excel.

After the mechanical components were completed for Kronos, calibration was conducted. The camera was situated at an angle off to the side of Kronos; therefore, the calibration was not linear. The basic overhead schematic for Kronos can be found in Figure 72.
Figure 72: Overhead layout of Kronos and camera field of view.

Calibration was achieved by recording mechanical position values alongside coordinate locations for a tracked person. The calibration utilized two inputs, x-coordinate and z-coordinate, to create an output calibration surface curve. The surface curve was achieved through Matlab and is discussed in Chapter 4.

3.6 Kronos Guest Feedback

3.6.1 Kronos Features Trial

A study was conducted amongst 20 high school students on February 24th, 2015. The 20 students were split into groups of two. Five of the groups saw Kronos with only two physical features, while the other five saw Kronos with five physical features. The two features included blinking eyes and a shifting head. The five feature mode included blinking eyes, shifting head, shifting neck, shifting wings and lifting tail. The students were asked to rank the following on a scale of 1-10: Realistic, engaging, lifelike, fun, and interesting. The data was compiled to test if more features ranked higher.
3.6.2 Kronos Behavior Survey with Elementary Students

A study was conducted among nearly 70 elementary school students on March 12th, 2015. The students were brought in groups of 3-4 with instructions to grade the dragon on his behavior. 37 students graded a passive behavior while 12 students graded an interactive behavior. The rest of the students were brought in larger groups, therefore, their surveys were not counted in the study. Qualitative feedback was still recorded. Teachers defined the survey words if the student did not understand. The students were asked to grade dragon on the following: Realistic, engaging, lifelike, fun, interesting, and to choose a smile that best described the entertainment experience, Figure 73.

![Figure 73: The Smile-Scale used for kids ages 12 and under.](image)

The surveys provided quantitative data to analyze, however, qualitative data was not completely disregarded. Notes were taken for any surprising or significant responses from guests. The results from this study are discussed in Chapter 4.

3.6.3 Kronos Behavior Survey with College Students

A study was conducted with college students to analyze feedback for passive and interactive behaviors. This study was conducted on April 2, 2015. Student volunteers
were asked to view Kronos then grade the animatronic based on the behavior. Forty-seven students were surveyed in total with 23 students surveyed for passive behavior and 24 surveyed for interactive behavior. The survey can be seen in Appendix A. The results for this study are discussed in Chapter 4.

3.6.4 Survey Data Analysis

Statistical analysis of the data was conducted to test the hypothesis. The Student T-test was used to analyze the results. Graphs and charts were created to compare different results of animatronic simulations. The results from the elementary school were analyzed to determine appropriate refinement for future tests.
Chapter 4: Findings and Analysis

4.1 Hyperion Results

This section of Chapter 4 discusses the results from the prototype animatronic system, Hyperion.

4.1.1 Hyperion Aesthetic Results

Despite the success of this animatronic endeavor, Hyperion suffered a few flaws. First, the aesthetic components were not consistent and the missing wings distracted from the overall look. The neck of the dragon was missing because the mold was not adequate to produce the casting. Even if the neck casting was produced, the latex would not have the flexibility to bend the full range of the mechanism capabilities. These flaws were noted and taken into consideration for the next form of the animatronic system.

4.1.2 Hyperion Mechanical Results

The space inside the dragon was tight and did not bode well for improvements or repairs. The servomotors that actuated the bulk of the movement became “too hot to touch” after 10 minutes of continuous use. The system was also powered solely through battery. The duty cycles for short runs and testing were inconvenienced by the battery being the sole power supply. The whining noise from the servomotors distracted from the simulation. These notes were rectified for the Kronos system.

4.1.3 Hyperion Control Analysis

The control system for Hyperion was achieved through the Arduino microcontroller and maestro servo controller. The programming for the system was
easily modified to add new motions to the dragon routine. Overall, the control implementation for Hyperion was successful and reliable.

4.1.4 Hyperion Feedback

Many of the children enjoyed watching and petting the dragon at the STEM Festival. Their expressions were of excitement and wonder. Figure 74 is a picture from the festival.

Figure 74: Hyperion appearance at the STEM Festival 2013.

During the event, it was noted that the most common forms of interaction from the kids were waving, talking to the dragon, or petting the dragon. The kids immediately treated the dragon as if it were a real animal. The kids observed the face of the dragon most often.
4.2 Kronos Results and Analysis

In this section, an analysis is made on the Kronos animatronic system. The analysis is broken to three components. These components include the aesthetics, mechanical components and control with people identification.

4.2.1 Kronos Aesthetics Features

The organic, or aesthetic, features of Kronos were pleasing. The polyurethane foam captured the detail from the mold. The foam features were able to bend with the neck mechanism without too much wrinkle. The paint gave the dragon uniformity. The one misstep on the aesthetic process was that the final parts were not made of silicone because of a misjudgment in materials needed for casting. The silicone was industrial grade used for animatronics by the largest of themed entertainment companies. This is something that could be addressed and replaced in the future; however, the foam seems to be holding up fine at the time of this thesis publication. The molds are still intact and could be used for future upgrade if necessary. However, despite the one misstep, the aesthetic features satisfied the requirement set forth by the hypothesis to create an aesthetically pleasing animatronic creature.

4.2.2 Kronos Mechanical Analysis

In this section, a discussion is made covering the mechanical components of the system. Overall, the mechanisms were pleasing to the eye and were successful in outputting desired motions. However, upgrades to specific parts could benefit the system. A ball bearing assembly would benefit the wing joints by reducing friction, thus reducing torque and power required to lift the wings. This would also improve lifecycles.
Ball bearing joint could also be installed for the tail mechanism as well as a replacement shaft hub on the motor to reduce the wobble. The eye mechanisms could also use mini ball bearings.

The main neck bend mechanisms undergo the majority of the stress during operation. At the conclusion of the 1+ hour testing of continuous use, one of the servomotors completely burnt out, see Figure 75. The two servomotors for bending the neck could be replaced with either larger servomotors that can handle the continuous load applied, or a linear actuator with potentiometer feedback. A plan was established to replace the bend actuators with an industrial grade servomotor.

Figure 75: Burnt H-Bridge after an hour of continuous use. March 12, 2015.

Another weak spot mechanically, was the use of the small actuator for the neck pivot mechanism. The actuator whines loudly in certain positions, which indicates the
load is near the maximum output. A plan was established to replace the small actuator with a larger actuator.

Besides those specific weak spots, the system operates satisfactorily and met the hypothesis requirement for the motor/mechanism outputs. The mechanisms produce organic like motions. When assembled with aesthetic components, actuator methods are not apparent upon observation of Kronos.

4.2.3 Kronos Control System Results and Analysis

This section covers the results from the test pan and tilt mechanism, as well as, the results for Kronos. The technique to identify the closest pixel to the camera resulted with successful live images of the closest pixel to the sensor. The result is shown in Figure 76. A running average was implemented to reduce the amount of “jitteriness” of the red circle. The minimum distance that the Carmine could detect was approximately 20 inches, which matched the hardware specifications.

Figure 76: Screen capture of tracking the closest pixel. A red circle was drawn over the pixel.
The next test was to successfully identify a person utilizing the OpenNI and NITE libraries. This was accomplished by simply calling the function to the library and adding the function to color in the identified person. The result can be found in Figure 77.

![Figure 77: Tracking a person by coloring the space blue.](image)

The servomotor position commands from Arduino to the Pololu ranged from 0 to 254. The values from the head tracking, from Figure 78, were converted into a usable range for the servomotor commands.
Figure 78: Tracking the head and center of mass of the person in the image.

The servomotors were moved manually to the most ideal position through the Pololu software on the computer while both the servo position and head coordinate data were recorded. The servos were actuated to appropriate positions by sending a PWM (pulse width modulation) signal which is measured in micro-seconds (µs). The servo-controller accepts an 8-bit position target for a servomotor in a range from 0-254. The 255 in the table was a mistake due to an early misunderstanding of the 8-bit value the servo-controller accepts. The 255 position value did not result in errors for the resulting calibration. The results are presented in Table 2 below.
Table 2: Servo calibration data for the 2 DOF assembly.

<table>
<thead>
<tr>
<th>No.</th>
<th>Pan x-coord (pixels)</th>
<th>Pan Servo Target (µs)</th>
<th>8-Bit Position</th>
<th>Tilt y-coord (pixels)</th>
<th>Tilt Servo Target (µs)</th>
<th>8-Bit Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>121</td>
<td>640</td>
<td>0</td>
<td>439</td>
<td>640</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>179</td>
<td>856</td>
<td>32</td>
<td>355</td>
<td>856</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>176</td>
<td>1072</td>
<td>64</td>
<td>307</td>
<td>1072</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>226</td>
<td>1288</td>
<td>96</td>
<td>267</td>
<td>1288</td>
<td>96</td>
</tr>
<tr>
<td>5</td>
<td>291</td>
<td>1504</td>
<td>128</td>
<td>222</td>
<td>1504</td>
<td>128</td>
</tr>
<tr>
<td>6</td>
<td>351</td>
<td>1720</td>
<td>159</td>
<td>194</td>
<td>1720</td>
<td>159</td>
</tr>
<tr>
<td>7</td>
<td>398</td>
<td>1936</td>
<td>191</td>
<td>83</td>
<td>1904</td>
<td>187</td>
</tr>
<tr>
<td>8</td>
<td>470</td>
<td>2152</td>
<td>223</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>504</td>
<td>2368</td>
<td>255</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The data was imported into Microsoft Excel and two calibration curves were generated. The calibration curves can be seen in Figure 79 and Figure 80.

![Pan Servomotor Calibration](image)

Figure 79: Calibration curve for the pan motion.
The calibration equation for the pan servomotor was computed as follows:

\[ P_{\text{servo}} = 3 \times 10^{-6}x^3 - 0.0034x^2 + 1.7361x - 168.03 \quad \ldots \text{Equation 4} \]

From the above equation, \( x \) represents the input x-coordinate value from the head tracking portion of the code.

The calibration equation for the tilt servomotor is as follows:

\[ T_{\text{servo}} = 1 \times 10^{-5}y^3 - 0.0078y^2 + 1.2741y - 129.63 \quad \ldots \text{Equation 5} \]

The calibration equations were implemented into the Processing code and the results were much more visually pleasing than a linear conversion.

The program was expanded in order to track multiple people at once and to provide coordinate data of the location of the person in the field of view. For Kronos, in order to best track a person in a horizontal field, both the X and Z coordinates had to be taken into account. Fifteen positions of a person within the field of view were matched with fifteen positions of the head location. The data can be seen in Table 3. The coordinate values come from the pixel number from the camera (max X-coordinate is 64).
Table 3: Kronos calibration data for head tracking.

<table>
<thead>
<tr>
<th>X-Coordinate (pixels)</th>
<th>Z-Coordinate (centimeters)</th>
<th>Neck Bend Output (8-bit position)</th>
<th>Neck Pivot Output (8-bit position)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>150</td>
<td>254</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>277</td>
<td>180</td>
<td>90</td>
</tr>
<tr>
<td>24</td>
<td>354</td>
<td>127</td>
<td>127</td>
</tr>
<tr>
<td>24</td>
<td>163</td>
<td>254</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>250</td>
<td>180</td>
<td>90</td>
</tr>
<tr>
<td>36</td>
<td>317</td>
<td>127</td>
<td>127</td>
</tr>
<tr>
<td>38</td>
<td>240</td>
<td>180</td>
<td>90</td>
</tr>
<tr>
<td>40</td>
<td>180</td>
<td>254</td>
<td>1</td>
</tr>
<tr>
<td>42</td>
<td>288</td>
<td>127</td>
<td>127</td>
</tr>
<tr>
<td>44</td>
<td>378</td>
<td>90</td>
<td>180</td>
</tr>
<tr>
<td>46</td>
<td>343</td>
<td>90</td>
<td>180</td>
</tr>
<tr>
<td>50</td>
<td>310</td>
<td>90</td>
<td>180</td>
</tr>
<tr>
<td>57</td>
<td>330</td>
<td>1</td>
<td>254</td>
</tr>
<tr>
<td>58</td>
<td>306</td>
<td>1</td>
<td>254</td>
</tr>
<tr>
<td>60</td>
<td>270</td>
<td>1</td>
<td>254</td>
</tr>
</tbody>
</table>

The data was taken for the neck bend and neck pivot and imported into Matlab’s surface solver to generate a surface curve and equation. These calibration curves are three dimensional because there are two inputs that define the output. The generated surface curves for the neck bend and pivot outputs can be found in Figure 81 and Figure 82, respectively.
Figure 81: Calibration surface curve for the neck bend.

Figure 82: Calibration surface curve for the neck pivot.
The best fit equations from these curves both came from polynomial functions with three degrees for \( x \) and two degrees for \( z \). The resulting equations were as follows for the neck bend curve.

\[
Output \ Pivot = \ -242.2 + 0.7987x + 2.052z - 0.2977x^2 + 0.03496xz \\
- 0.004357z^2 + 0.004916x^3 - 0.0004588x^2z + 0.00002285xz^2
\]

\[
Output \ Bend = \ 340 + 0.2653x - 0.3369z + 0.3693x^2 - 0.06686xz \\
+ 0.0004915z^2 - 0.005062x^3 + 0.0002431x^2z + 0.0000710xz^2
\]

Unfortunately, the neck pivot was too slow and the reaction time was not pleasing to the eye for tracking the head linearly alongside the quicker servomotors that controlled the bending of the neck. However, the pivot actuator was still useful for putting Kronos into different positions that did not involve tracking a person. Therefore, new calibration data was generated as found in Table 4.
Table 4: Kronos head tracking calibration data without pivot.

<table>
<thead>
<tr>
<th>X-Coordinate (pixels)</th>
<th>Z-Coordinate (centimeters)</th>
<th>Neck Bend Output Position (8-bit target)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>206</td>
<td>254</td>
</tr>
<tr>
<td>20</td>
<td>320</td>
<td>180</td>
</tr>
<tr>
<td>27</td>
<td>350</td>
<td>127</td>
</tr>
<tr>
<td>29</td>
<td>198</td>
<td>254</td>
</tr>
<tr>
<td>30</td>
<td>260</td>
<td>180</td>
</tr>
<tr>
<td>34</td>
<td>305</td>
<td>127</td>
</tr>
<tr>
<td>40</td>
<td>245</td>
<td>180</td>
</tr>
<tr>
<td>39</td>
<td>203</td>
<td>254</td>
</tr>
<tr>
<td>43</td>
<td>275</td>
<td>127</td>
</tr>
<tr>
<td>32</td>
<td>392</td>
<td>90</td>
</tr>
<tr>
<td>39</td>
<td>338</td>
<td>90</td>
</tr>
<tr>
<td>46</td>
<td>294</td>
<td>90</td>
</tr>
<tr>
<td>46</td>
<td>387</td>
<td>1</td>
</tr>
<tr>
<td>48</td>
<td>345</td>
<td>1</td>
</tr>
<tr>
<td>55</td>
<td>296</td>
<td>1</td>
</tr>
</tbody>
</table>

With the new data, a new surface calibration curve, Figure 83, was generated using the same method as before.
Figure 83: Neck bend calibration curve two.

The new calibration curve equation is as follows:

This surface curve did not require the same number of degrees as the previous curves did. As the head rotates, because of how the neck assembly lifts, it was required to calibrate the head rotation. The head rotation calibration data can be seen in Table 5. The rotation required was slight, but necessary.

Table 5: Head rotation calibration data.

<table>
<thead>
<tr>
<th>Kronos Head Rotate Calibration Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neck Bend Position (8-bit target)</td>
</tr>
<tr>
<td>------------------------------------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>90</td>
</tr>
<tr>
<td>127</td>
</tr>
<tr>
<td>180</td>
</tr>
<tr>
<td>254</td>
</tr>
</tbody>
</table>
The data was input into Microsoft Excel to compute the 2-Dimensional curve and best-fit equation, Figure 84.

![Head Rotate Calibration Curve](image)

**Figure 84: Head Rotate Calibration Curve**

The best fit equation for the head rotate calibration curve:

\[ O_{\text{Rotate}} = 0.0006x^2 - 0.3055x + 128.4 \]

The calibration equations were programmed into the Arduino microcontroller for head tracking mode only.

One major requirement from the hypothesis was that the system had to have the ability to identify people around the animatronic. An in-software, three frame animation of the system tracking a person can be seen in Figure 85. Each frame displays the infrared image as well as the RGB camera image. The system is capable of tracking up to six people at a time. This information can be sent to Arduino for calibration and logic of the output. The computer system sends coordinates to Arduino to calibrate the output.
The research attempt to identify people in an area was successful. Next, the system had to be able to output the animatronic to utilize the data from the tracked person. The real time people tracking capabilities can be seen through Figure 86, Figure 87, and Figure 88. The figures display the dragon “watching” the person in the figures. Figure 88 displays that when a person comes in close proximity of the dragon, Kronos will growl and spread wings.
Figure 86: Kronos real time tracking right.

Figure 87: Kronos real time tracking left.
Kronos will growl for a short period of time. The growl comes from a small audio speaker located just behind the base of the neck. If the person does not move away from Kronos, Kronos will get bored and go to sleep. If the person moves back into range of head tracking, Kronos will watch the person move about. If there is no person tracked or the person is far away and no longer moving, Kronos goes to sleep, as seen in Figure 89. Kronos will remain asleep until the system identifies another human body moving about in the field of view.
During the passive behavior mode, the dragon sleeps for a specified period of time, raises up and looks around, stretches and lies back down to sleep again. The entire passive script lasts approximately 40 seconds before the cycle is repeated. Overall, the people tracking method, calibration, and output was very successful and satisfied the requirement set forth by the hypothesis.

4.3 Kronos Guest Feedback Results and Analysis

The results for guest feedback were recorded both quantitatively and qualitatively. This was important because reasons for the quantitative results could be found from qualitative observations.

4.3.1 Quantitative Results

The results from the features study as well as the passive and interactive trials are discussed in the following sections. Only quantitative results are discussed here while qualitative results are in the following section.
4.3.1.1 Features Survey Results

A survey was conducted before completion of the dragon. The survey was testing to see if the number of features, or motions, that Kronos output affected guest opinions on realism, level of engagement, lifelikeness, fun and interest levels. The hypothesis for this test was that guests would rate the five feature mode higher than the two feature mode in the areas of realism, engagement, lifelikeness, fun and interest levels. The survey results can be found in Figure 90.

![Number of Animated Features Comparison](image)

**Figure 90: Results from feature comparison.**

The averages for all five questions for five features utilized were all higher than the averages for utilizing only two features. The two feature mode utilized the blinking eyes and shifting head. The head did not lift, however, the head rotated a little side to side. The five feature behavior mode utilized the blinking eyes, shifting neck, rotating
head, shifting wings and lifting tail. The wings were not fully extended nor did the head have the ability to move the full range. The T-Test data for this study can be found in Table 6.

Table 6: T-test data for comparing number of features.

<table>
<thead>
<tr>
<th>Group</th>
<th>Behavior Mode</th>
<th>n</th>
<th>x</th>
<th>s</th>
<th>t</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reality</td>
<td>Five Features</td>
<td>10</td>
<td>7.80</td>
<td>1.83</td>
<td>0.261</td>
<td>0.400</td>
</tr>
<tr>
<td></td>
<td>Two Features</td>
<td>10</td>
<td>7.61</td>
<td>1.37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engaging</td>
<td>Five Features</td>
<td>10</td>
<td>6.80</td>
<td>1.72</td>
<td>0.272</td>
<td>0.396</td>
</tr>
<tr>
<td></td>
<td>Two Features</td>
<td>10</td>
<td>6.56</td>
<td>2.27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lifelike</td>
<td>Five Features</td>
<td>10</td>
<td>7.50</td>
<td>2.06</td>
<td>0.296</td>
<td>0.387</td>
</tr>
<tr>
<td></td>
<td>Two Features</td>
<td>10</td>
<td>7.17</td>
<td>2.91</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fun</td>
<td>Five Features</td>
<td>10</td>
<td>8.50</td>
<td>2.11</td>
<td>1.107</td>
<td>0.149</td>
</tr>
<tr>
<td></td>
<td>Two Features</td>
<td>10</td>
<td>7.33</td>
<td>2.58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interesting</td>
<td>Five Features</td>
<td>10</td>
<td>9.40</td>
<td>1.02</td>
<td>1.702</td>
<td>0.062</td>
</tr>
<tr>
<td></td>
<td>Two Features</td>
<td>10</td>
<td>8.33</td>
<td>1.70</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

With a 90% confidence level, the only question that had a statistically higher mean was the one asking how interesting Kronos was.

4.3.1.2 Elementary Student Survey Results

The next survey tested one of the thesis hypotheses that interactive behaviors would rank higher in entertainment than passive behaviors. Like the previous survey, five questions were asked on realism, engagement, lifelikeness, fun and interest levels. The full survey form can be seen in Appendix A. This survey included an additional question regarding the entertainment value. The results can be seen in Figure 91.
Figure 91: Results from Passive versus Interactive Behavior Comparison.

Realistic and engaging resulted in lower average means for the interactive average than the passive. For the survey results, a two sample T-Test was conducted. The tests were all one tailed and assumed a confidence level of 95%, therefore, $\alpha=0.05$. The null hypothesis, $H_0$, for the experiment was that the means would be equal and there would be no statistical difference between the survey results for passive and interactive behaviors. The alternative hypothesis, $H_a$, was that the interactive behavior would produce a higher average mean, $\mu_2$, than the passive mean, $\mu_1$. These hypotheses are given below.

$$H_0: \mu_1 = \mu_2$$

$$H_a: \mu_1 < \mu_2$$
The t-test is represented by the equation that follows where $\bar{x}$ represents the mean, $s$ represents standard deviation and $n$ represents the sample size. The sample sizes were unequal for the test. The following equation was utilized for calculating the t-value.

$$ t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} $$

A t value for each of the six survey questions were computed as seen in

<table>
<thead>
<tr>
<th>T-Test Data for Survey Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>Realism</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Engaging</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Lifelike</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Fun</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Interesting</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Smiley Meter</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Table 7.
Table 7: T-test data for survey results.

The t critical value is a table value based on an alpha of 0.05 and the smallest sample size minus one.

\[ t_{0.05,11} = 1.796 \]

The statistic above the t-critical value was the question about how interesting Kronos was. The p-value for the level of interest was as follows: \( p_i = 0.026 \). The p-value for the entertainment statistic was as follows: \( p_e = 0.252 \). The null was only rejected for the survey question called “interesting”. The interest level of the guest with the interactive behavior mode was statistically higher than the interest level of the student with the passive behavior animatronic. Therefore, the hypothesis statistic for the level of entertainment being higher in interactive behavior over passive behavior with a 95% level of confidence level was not proved, though, testing among a more diverse population may be able to prove the hypothesis. The interest level statistic was the only statistic that merited as significant for both of the tests.

4.3.1.3 College Student Survey Results

Georgia Southern University students were surveyed in the B-IRIS laboratory to test the hypothesis that the entertainment value of interactive behavior is higher than passive behavior in animatronics. Forty-seven students were surveyed after viewing the animatronic dragon. The students viewed the dragon one at a time and were not told what
kind of behavior mode Kronos was running. The survey can be viewed in Appendix A. 

The average rating for each survey question is seen in Figure 92.

![Passive vs Interactive Modes with College Students](image)

**Figure 92: Averages for the 47 college students.**

The data for testing passive versus interactive modes with college students can be seen in Table 8. Like before, a two sample T-Test was conducted. The tests were all one tailed and assumed a confidence level of 95%, therefore, \( \alpha = 0.05 \). The null hypothesis, \( H_0 \), defines the means being equal and is no statistical difference between the results for passive and interactive behaviors. The alternative hypothesis, \( H_\alpha \), states that interactive behavior produces a higher average mean, \( \mu_2 \), than the passive mean, \( \mu_1 \). These hypotheses are given below.
The results for all 47 college students did not produce any means with a p-value lower than 0.05, therefore the null hypothesis was not rejected.

Table 8: T-Test data for testing passive versus interactive modes with college students.

<table>
<thead>
<tr>
<th>Group</th>
<th>Behavior Mode</th>
<th>n</th>
<th>x</th>
<th>s</th>
<th>t</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Realism</td>
<td>Passive</td>
<td>23</td>
<td>7.57</td>
<td>1.53</td>
<td>0.998</td>
<td>0.165</td>
</tr>
<tr>
<td></td>
<td>Interactive</td>
<td>24</td>
<td>7.96</td>
<td>1.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engaging</td>
<td>Passive</td>
<td>23</td>
<td>7.96</td>
<td>1.68</td>
<td>0.559</td>
<td>0.291</td>
</tr>
<tr>
<td></td>
<td>Interactive</td>
<td>24</td>
<td>8.21</td>
<td>1.38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lifelike</td>
<td>Passive</td>
<td>23</td>
<td>7.74</td>
<td>1.45</td>
<td>0.176</td>
<td>0.431</td>
</tr>
<tr>
<td></td>
<td>Interactive</td>
<td>24</td>
<td>7.67</td>
<td>1.37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fun</td>
<td>Passive</td>
<td>23</td>
<td>9.26</td>
<td>0.94</td>
<td>0.721</td>
<td>0.239</td>
</tr>
<tr>
<td></td>
<td>Interactive</td>
<td>24</td>
<td>9.04</td>
<td>1.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interesting</td>
<td>Passive</td>
<td>23</td>
<td>9.61</td>
<td>0.71</td>
<td>0.633</td>
<td>0.266</td>
</tr>
<tr>
<td></td>
<td>Interactive</td>
<td>24</td>
<td>9.46</td>
<td>0.91</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entertainment</td>
<td>Passive</td>
<td>23</td>
<td>9.17</td>
<td>1.05</td>
<td>0.650</td>
<td>0.261</td>
</tr>
<tr>
<td></td>
<td>Interactive</td>
<td>24</td>
<td>9.38</td>
<td>1.07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smiley Meter</td>
<td>Passive</td>
<td>23</td>
<td>4.57</td>
<td>0.495728</td>
<td>1.359</td>
<td>0.094</td>
</tr>
<tr>
<td></td>
<td>Interactive</td>
<td>24</td>
<td>4.75</td>
<td>0.433013</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

However, one of the questions on the survey asked if the student had any prior exposure with the dragon. If the student had previous exposure, the opinions of the animatronic behavior might have bias. Therefore, the results were analyzed excluding all students that had previously been exposed to Kronos. The resulting averages can be seen in Figure 93.
From the above figure, the resulting averages are different from the total averages from the college students. The only statistic that had a lower average for the interactive mode was for engaging. The table results for the t-test can be seen in Table 9.
Table 9: T-test data for college students without prior exposure to Kronos.

<table>
<thead>
<tr>
<th>Group</th>
<th>Behavior Mode</th>
<th>n</th>
<th>x</th>
<th>s</th>
<th>t</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reality</td>
<td>Passive</td>
<td>6</td>
<td>7.83</td>
<td>1.07</td>
<td>0.045</td>
<td>0.483</td>
</tr>
<tr>
<td></td>
<td>Interactive</td>
<td>14</td>
<td>7.86</td>
<td>1.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engaging</td>
<td>Passive</td>
<td>6</td>
<td>8.17</td>
<td>1.21</td>
<td>0.440</td>
<td>0.339</td>
</tr>
<tr>
<td></td>
<td>Interactive</td>
<td>14</td>
<td>8.43</td>
<td>1.24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lifelike</td>
<td>Passive</td>
<td>6</td>
<td>8.00</td>
<td>0.82</td>
<td>0.462</td>
<td>0.332</td>
</tr>
<tr>
<td></td>
<td>Interactive</td>
<td>14</td>
<td>7.79</td>
<td>1.21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fun</td>
<td>Passive</td>
<td>6</td>
<td>9.17</td>
<td>0.90</td>
<td>0.827</td>
<td>0.223</td>
</tr>
<tr>
<td></td>
<td>Interactive</td>
<td>14</td>
<td>9.50</td>
<td>0.63</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interesting</td>
<td>Passive</td>
<td>6</td>
<td>9.33</td>
<td>0.75</td>
<td>0.635</td>
<td>0.277</td>
</tr>
<tr>
<td></td>
<td>Interactive</td>
<td>14</td>
<td>9.57</td>
<td>0.82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entertainment</td>
<td>Passive</td>
<td>6</td>
<td>8.67</td>
<td>0.94</td>
<td>2.335</td>
<td>0.033</td>
</tr>
<tr>
<td></td>
<td>Interactive</td>
<td>14</td>
<td>9.64</td>
<td>0.61</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smiley Meter</td>
<td>Passive</td>
<td>6</td>
<td>4.33</td>
<td>0.47</td>
<td>2.448</td>
<td>0.029</td>
</tr>
<tr>
<td></td>
<td>Interactive</td>
<td>14</td>
<td>4.86</td>
<td>0.35</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From the previous table, the p-value for entertainment was 0.033, which is lower than the alpha, 0.05. Thus, for the entertainment question, the null hypothesis is rejected and the alternative can be accepted for this test. Entertainment was valued higher by college students with no prior exposure to the Kronos system. The results for only students with exposure to the system can be seen in Figure 94.
Figure 94: Results for college students with prior exposure to Kronos.

The only statistical difference between the questions for students with prior exposure was the question for fun. Fun was rated higher for the passive mode, rather than the interactive mode.
Table 10: T-test data for college students without prior exposure to Kronos.

<table>
<thead>
<tr>
<th>T-Test Data for College Students with Prior Exposure to Kronos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>Realism</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Engaging</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Lifelike</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Fun</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Interesting</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Entertainment</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Smiley Meter</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

4.3.2 Qualitative Results

While the quantitative survey results were used to disprove or prove the hypothesis, it was important to not discount qualitative feedback from guests. While the students were busy watching or interacting with Kronos, notes were made for particular events that occurred. These events included particular comments or common gestures that occurred during the study period. Kronos in the school library, for the elementary student test, can be seen in Figure 95.
The most common occurrence noted was that many of the kids found the dragon to be interactive even in the passive mode. Comments during passive behavior included, “He is looking at me!” or “Watch what he does when I clap”. Perhaps the mindset of the kids thinking that Kronos was interactive, even when not, inflated the passive average results.

Some of the groups ventured close to pet Kronos. Comments were made such as “Look! He likes it when I pet his nose. If you pet him, he wakes up.” A little greater than half of the groups pet the dragon. There were no rules given on whether or not petting was allowed.

During the interactive behavior mode, only four students realized that the dragon would “watch” them as they moved around. The other students were clapping, waving, or talking to get Kronos to interact, though, the only input Kronos was receiving,
technically, was visually from the Carmine. There were six occurrences of the kids noticing that he growled when they got close to the dragon. One of the younger girls hugged me after seeing the dragon because, the author assumes, she liked it so much.

Figure 96: Kids petting Kronos asking him to wake up during passive behavior. March 12, 2015.

Qualitatively, Kronos was not being fully utilized for his intended interactive behavior. Kronos was interactive to the children in passive behavior and interactive. These results sparked some interactive behavior ideas that need investigation. The following are thoughts on how to improve the real utilization of the interactive elements.

- For the current interactive behavior, have some kind of arc barrier or line that does not allow for the guest to cross. This may cause the guest to pace back and forth more often allowing Kronos to track the individual.
• Program Kronos to switch between people in field of view repeatedly or to the most active person.

• Program Kronos to track hands since guests waved more often than physically moving their bodies.

For the college students, qualitative observations were also made. The responses from the students were overall very positive. The most exciting reaction was an individual, whom had no prior exposure to Kronos, the individual jumped back in surprise when Kronos looked up at him. The individual’s surprise turned into curiosity as the individual began to pace the room as Kronos “watched”.

Many of the students during the interactive mode tested the range of the dragon capabilities. The students walked close to the dragon to witness the growl and paced the room to check the response time. Some the students ducked low and waved hands. One of the students touched the dragon and attributed the growl from the touch, although, the growl came from the proximity. Qualitatively, the students are very intuitive and smart; many of the students identified the range of capabilities in the inactive mode. When the program logic is identified, a sense of intelligence and realism from the animatronic is lost. Further complication and an element of randomness to the behavior may further intrigue the audience.

Overall, there is a lot of programming logic that could be implemented into Kronos that would allow the dragon to change from tracking various body parts or movements. The new logic techniques employed could have a major effect on survey results for comparing passive and interactive animatronic behavior.
Chapter 5: Conclusions, Recommendations for Future Work

5.1 Conclusion

The first hypothesis of the thesis was that a dynamic, people-aware, autonomous, animatronic system can be designed and fabricated for the purposes of testing interactive elements. This hypothesis was proved correct for all parts with the design of the animatronic dragon, Kronos. Through people identification, animatronic system allows extracted location data points for various parts of the body. The system has the ability of identifying up to six of the most recent people that entered the field of view at a given time. The animatronic system includes aesthetic parts that conceal the internal mechanisms and produced pleasing visual results. The program of the system allowed for switching between modes.

The second hypothesis was that interactive, animatronic behaviors provide a more entertaining experience amongst guests versus passive behaviors. The survey results were analyzed using the t-test. For test with elementary students, the average entertainment value for interactive behavior was not greater than the average entertainment value for passive behavior with a 95% level of confidence. Through this feedback, the results did not prove the second thesis hypothesis correct. However, qualitatively, Kronos was interactive to the majority of the participants even when in passive behavior. Perhaps, if the students were given a longer time to engage with Kronos, they might have realized Kronos was acting passively or figure out what interaction Kronos was actually portraying. Other ways to implement interactive behaviors may change the outcome in future tests. For the survey results with college students, there was no difference between the averages, also based a 95% confidence
level. However, when the results were analyzed only with those students with no prior exposure to the Kronos system, the average entertainment ranking by the student proved to be greater during the interactive mode than the passive mode. This analysis concurs with the thesis hypothesis that interactive behaviors are more entertaining than passive behaviors. Further investigation may strengthen the difference in responses for survey feedback.

5.2 Recommendations for Future Work

More survey tests need to be conducted with a larger, more diverse population and with different, perhaps more complex, interactive modes. The results would give insight to what interactive behaviors people engage with best. The future studies would allow testing demographics against one another to see what is most effective for different groups.

General mechanism improvements and motor replacements could be achieved in future. A few of the motors were not built for continuous cycling and would likely not last with continuous tests.

As next phase enhancements, claws might be added into the aesthetic components by carving wooden dowels and painting. Painted cloth might be added between the head and the neck to cover the mechanism that is exposed during bends. Replacing foam with a silicone/foam mixture could benefit the longevity of aesthetic components. However, it is pure speculation on how long the foam will last.

In the near future, Kronos might be integrated into the B-IRIS cloud robotics project. This would offset the computation required for image analysis. The cloud
robotics system runs on a cluster of six high speed servers. The servers run the Robot Operating System and integrate various robot languages into one format for heterogeneous control. This integration would allow remote access and processing for Kronos. Future capabilities may include autonomous interaction with other robots amongst the laboratory.
References


Anderson, Paul F. *Audio-Animatronics*. n.d.
http://labyrinth.tripod.com/animatronics/history/animatronics101-102/aa.html
(accessed November 18, 2013).


Kimberly. Lucky The Dinosaur. n.d.


"PrimeSense 3D Sensors." PrimeSense, n.d.


APPENDICES

Appendix A: List of Publications and Presentations

The following is a list of publications for this thesis:


The following is a list of presentations for this thesis:


Appendix B: Survey for guest response.

The following is the survey for elementary students and college students, respectively.

<table>
<thead>
<tr>
<th></th>
<th>Realistic</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Engaging</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Lifelike</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Fun</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Interesting</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

*Circle the face that describes how entertaining the dragon was?*

![Emoticons ranging from sad face to happy face]
### Kronos Feedback Survey

We are conducting research on different animatronic behaviors. It is important that you answer honestly and not generously.

**Circle your answers below:**

<table>
<thead>
<tr>
<th></th>
<th>Which mode was the dragon in?</th>
<th>Blue</th>
<th>White</th>
<th>Gold</th>
<th>Freedom</th>
<th>Glory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Have you seen this dragon project prior to today?</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Rate the following on a scale of 1-10 with 1 being low and 10 being high.*

<table>
<thead>
<tr>
<th></th>
<th>Realism</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Engaging</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Lifelike</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Fun</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Intriguing</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Entertaining</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Circle the face that best describes your experience with Kronos:*
Appendix C: Updates to Kronos

This appendix item includes updates to the Kronos system. The updates include a number of hardware, software and aesthetic upgrades to the system. The first set of upgrades includes ball bearings for the joints in the system. Ball bearings were installed for the neck lift mechanism, wing pivot points and the tail mechanism as seen in Figure 97, Figure 98 and Figure 99.

Figure 97: Ball bearing addition for the neck lift.
Figure 98: Ball bearing joints for the wings.

Figure 99: Ball bearing installed for the tail mechanism.
Aesthetically, the bases of the wings were re-clothed. The cloth now covers the entire base of the wing without exposing the inner joints. Claws were shaped out of excess foam and glued to the fingers. The panels were finished and installed to fully enclose three sides of the table. These updates can be found in Figure 100.

A new actuator, the Firgelli FA-HF-100-12-3, replaced the lift actuator. The new actuator operates with less sound and at 2.5”/s rather than 0.5”/s. A Torxis i01857 replaced the servomotors for the wings. The Torxis servomotor is industrial grade and is rated for constant torque loads of 800oz-in. The pivot actuator was replaced with a larger Firgelli actuator, specifically the FA-PO-35-12-2. A panel was cut to hold the fans and
monitor while keeping access to buttons for control. These updates can be found in Figure 101.

Figure 101: Hardware upgrades include a new monitor, actuators and industrial servomotor.

The code was refined and now allows for tracking of over 30 people. The program no longer crashes when over six people enter the field of view. An interface was developed to input the mode of which Kronos would operate. The software is a stand-alone application exported out of the processing code. The interface can be found in Figure 102. There are three buttons to change the mode of operation. Currently, only the tracking and passive modes are available. The “sleep” button puts Kronos in resting position. The input boxes below the “Camera Location” text allows the user to input
coordinate location data of the infrared camera in respect to Kronos. At the time of this thesis publication, auto-calibration for the camera location has yet to be implemented.

Figure 102: Interface application for Kronos.
Appendix D: Processing code for Kronos.

//Kronos Processing Revision 29

import SimpleOpenNI.*; // import camera library

import processing.serial.*; //Arduino communication
Serial myPort; // Create object from Serial class

/////////////////////////////for input numbers
import controlP5.*;

ControlP5 cp5;

String url1, url2, url3;

///////////////////////////////for input numbers

/*-----------------------------------------------
Variables
-----------------------------------------------*/
SimpleOpenNI camera; // create camera object
PImage cameraDepth; // image storage from infrared camera
PImage cameraRGB; //image storage for the RGB camera

int[] userID; // int of each user being tracked

color[] userColor = new color[] { color(0,255,0), color(255,0,0), color(0,0,255), // user colors
    color(255,255,0), color(255,0,255), color(0,255,255), color(0,255,0),
    color(255,0,0), color(0,0,255), color(0,255,0), color(255,0,0), color(0,0,255), color(255,0,0), color(0,255,0), color(255,0,0), color(0,0,255), color(255,0,0), color(0,255,0), color(255,0,0), color(0,0,255),};

PVector headPosition = new PVector(); // position of head to draw circle
float distanceScalar; // turn headPosition into scalar form
float headSize = 200; // diameter of head drawn in pixels
float confidenceLevel = 0.5; // threshold of level of confidence

float confidence; // the current confidence level that the camera is tracking

PVector confidenceVector = new PVector(); // vector of tracked head for confidence checking

//Font Variables
PFont titlefont;
PFont datafont;
PFont brianfont;
int a = 10; //title font x coordinate
int b = 710; //title font y coordinate
int c = 15; //servo table x coordinate
int d = 580; //servo table y coordinate

int CameraX = 0;
int CameraY = -36;
int CameraZ = 60;
int KronosX = 0;
int KronosY = 0;
int KronosZ = 0;

long previousMillis = 0; // will store last time output was sent to Arduino
int interval = 15; // interval at which to send output data to Arduino

public static final char HEADER = 'H'; // Arduino starts listening when this is sent
public static final char COORDINATE_TAG = 'C'; // for head tracking mode input
public static final char ANOTHER_TAG = 'A'; // can use for different mode

//-------------------------------------------------------------------------------- Start Stop Button vvvvvv

int restX, restY;   // Position of square button
int trackX, trackY; // Position of circle button
int passiveX, passiveY; // Position of passive button
int simonX, simonY; // Position of simon says button
int restSize = 87;   // Diameter of rect
int trackSize = 93;  // Diameter of circle
int passiveSize = 93; // Diameter of circle for passive button
int simonSize = 93;  // Diameter of circle for simon says button
color restColor, trackColor, baseColor, passiveColor, simonColor;
color restHighlight, trackHighlight, passiveHighlight, simonHighlight;
color currentColor;
boolean restOver = false;
boolean trackOver = false;
boolean passiveOver = false;
boolean simonOver = false;

int buttonPressedState = 0;

//-------------------------------------------------------------------------------- Start Stop Button ^^^^^^^
/*------------------------------------------------------------------------------------------
 Starts new camera object and enables skeleton tracking.
 Draws window
------------------------------------------------------------------------------------------*/

void setup()
{
    camera = new SimpleOpenNI(this); // start a new camera object
    camera.enableDepth(); // enable depth sensor
    camera.enableRGB(); // enable color camera
    camera.enableUser(); // enable skeleton generation for all joints
    strokeWeight(2); // draw thickness of drawer
    smooth(); // smooth out drawing
}
background(125); // create a window //White background
size(1290, 720);

//Create Fonts
titlefont = createFont("Arial",30,true);
datafont = createFont("Arial",24,true);
brianfont = createFont("Georgia", 18, true);
//Title Text
textFont(titlefont,50);
fill(0);
text("Kronos", a,b);
textFont(brianfont, 18);
text("by Brian Burns", 190, 715);
//Camera Labels
textFont(datafont,16);
fill(0);
  textFont(datafont,24);
  fill(0);
text("Infrared Image", 280, 510);
text("RGB Image", 960, 510);
//Table Text
textFont(datafont,18);
fill(0);
  text("X-Coordinate:", c, d);
  text("Y-Coordinate:", c, d+25);
  text("Z-Coordinate", c, d+50);
  text("Camera", c+130, d-30);
  text("Kronos", c+220, d-30);

  textFont(datafont,20);
  fill(0);
  text("Camera Location", c+430, d-40);
  textFont(datafont,16);
  fill(0);
  text("Tracking", c+755, d+65);
  text("Passive", c+875, d+65);
    text("Nothing", c+995, d+65);
    text("Sleep", c+1150, d+65);

String portName = Serial.list()[1];
myPort = new Serial(this, portName, 9600);

//vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
v start stop button
int xButtons = 100;
restColor = color(0);
restHighlight = color(51);
trackColor = color(255);
trackHighlight = color(204);
passiveColor = color(255);
passiveHighlight = color(204);
simonColor = color(255);
simonHighlight = color(204);
baseColor = color(102);
currentColor = baseColor;
trackX = (width/2+trackSize/2+10)+xButtons;
trackY = (height/2)+220;
passiveX = (width/2+trackSize/2+10)+xButtons+115;
passiveY = (height/2)+220;
simonX = (width/2+trackSize/2+10)+xButtons+230;
simonY = (height/2)+220;
restX = (width/2-restSize-10)+590;
restY = (height/2-restSize/2)+220;
ellipseMode(CENTER);

cp5 = new ControlP5(this);

cp5.addTextfield("Input_X_Distance").setPosition(450, 565).setSize(75, 30).setAutoClear(false);

cp5.addTextfield("Input_Y_Distance").setPosition(450, 615).setSize(75, 30).setAutoClear(false);

cp5.addTextfield("Input_Z_Distance").setPosition(450, 665).setSize(75, 30).setAutoClear(false);

cp5.addBang("Submit").setPosition(570, 600).setSize(80, 40);

void draw(){

  camera.update(); // update the camera
  cameraDepth = camera.depthImage(); // get camera data
  cameraRGB = camera.rgbImage(); // get color camera data
  image(cameraDepth, 5, 5); // draw depth image at coordinates (10,10)
  image(cameraRGB, 645, 5); // draw color image at coordinates (650, 10)
  
  //VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV
  strokeWeight(2);
update(mouseX, mouseY);
// background(currentColor);

if (restOver) {
    fill(restHighlight);
} else {
    fill(restColor);
}
stroke(255);
rect(restX, restY, restSize, restSize);

if (trackOver) {
    fill(trackHighlight);
} else {
    fill(trackColor);
}
stroke(0);
ellipse(trackX, trackY, trackSize, trackSize);

if (passiveOver) {
    fill(passiveHighlight);
} else {
    fill(passiveColor);
}
stroke(0);
ellipse(passiveX, passiveY, passiveSize, passiveSize);

if (simonOver) {
    fill(simonHighlight);
} else {
    fill(simonColor);
}
stroke(0);
ellipse(simonX, simonY, simonSize, simonSize);

//^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ start stop button
if (buttonPressedState == 1){ //tracking mode
    strokeWeight(20);

    userID = camera getUsers(); // get all user IDs of tracked users

    for(int i=0;i<userID.length;i++) // loop through each user to see if tracking
    {
        if(camera.isTrackingSkeleton(userID[i])) // if camera is tracking certain user then get joint vectors
            {

confidence = camera.getJointPositionSkeleton(userID[i],
SimpleOpenNI.SKEL_HEAD,confidenceVector); // get confidence level that camera is tracking head

if(confidence > confidenceLevel) // if confidence of tracking is beyond threshold, then track user
{
    stroke(userColor[(i)]); // change draw color based on hand id#
    fill(userColor[(i)]); // fill the ellipse with the same color
    drawSkeleton(userID[(i)]); // draw the rest of the body
    displayData(); //data onscreen
}

} //if(camera.isTrackingSkeleton(userID[i]))
} //for(int i=0;i<userID.length;i++)

} //if buttonPressedState == 1
else {
    // println("Sleeping");
}

} //void draw()

/* --------------------------------------------------------------------------
     Defining all the voids below
--------------------------------------------------------------------------*/

/*---------------------------------------------------------------
Draw the skeleton of a tracked user.  Input is userID
----------------------------------------------------------------*/
void drawSkeleton(int userId){
    camera.getJointPositionSkeleton(userId, SimpleOpenNI.SKEL_HEAD,headPosition); // get 3D position of head
    camera.convertRealWorldToProjective(headPosition,headPosition); // convert real world point to projective space
    distanceScalar = (525/headPosition.z); // create a distance scalar related to the depth in z dimension
    ellipse(headPosition.x,headPosition.y, distanceScalar*headSize,distanceScalar*headSize); // draw the circle at the position of the head with the head size scaled by the distance scalar

    camera.drawLimb(userId, SimpleOpenNI.SKEL_HEAD, SimpleOpenNI.SKEL_NECK); //draw limb from head to neck
    camera.drawLimb(userId, SimpleOpenNI.SKEL_NECK, SimpleOpenNI.SKEL_LEFT_SHOULDER); //draw limb from neck to left shoulder
camera.drawLimb(userId, SimpleOpenNI.SKEL_LEFT_SHOULDER, SimpleOpenNI.SKEL_LEFT_ELBOW); //draw limb from left shoulder to left elbow
camera.drawLimb(userId, SimpleOpenNI.SKEL_LEFT_ELBOW, SimpleOpenNI.SKEL_LEFT_HAND); //draw limb from left elbow to left hand
camera.drawLimb(userId, SimpleOpenNI.SKEL_NECK, SimpleOpenNI.SKEL_RIGHT_SHOULDER); //draw limb from neck to right shoulder
camera.drawLimb(userId, SimpleOpenNI.SKEL_RIGHT_SHOULDER, SimpleOpenNI.SKEL_RIGHT_ELBOW); //draw limb from right shoulder to right elbow
camera.drawLimb(userId, SimpleOpenNI.SKEL_RIGHT_ELBOW, SimpleOpenNI.SKEL_RIGHT_HAND); //draw limb from right elbow to right hand
camera.drawLimb(userId, SimpleOpenNI.SKEL_RIGHT_HIP, SimpleOpenNI.SKEL_RIGHT_KNEE); //draw limb from right hip to right knee
camera.drawLimb(userId, SimpleOpenNI.SKEL_RIGHT_KNEE, SimpleOpenNI.SKEL_RIGHT_FOOT); //draw limb from right knee to right foot

camera.drawLimb(userId, SimpleOpenNI.SKEL_LEFT_SHOULDER, SimpleOpenNI.SKEL_TORSO); //draw limb from left shoulder to torso
camera.drawLimb(userId, SimpleOpenNI.SKEL_RIGHT_SHOULDER, SimpleOpenNI.SKEL_TORSO); //draw limb from right shoulder to torso
camera.drawLimb(userId, SimpleOpenNI.SKEL_TORSO, SimpleOpenNI.SKEL_LEFT_HIP); //draw limb from torso to left hip
camera.drawLimb(userId, SimpleOpenNI.SKEL_LEFT_HIP, SimpleOpenNI.SKEL_LEFT_KNEE); //draw limb from left hip to left knee
camera.drawLimb(userId, SimpleOpenNI.SKEL_LEFT_KNEE, SimpleOpenNI.SKEL_LEFT_FOOT); //draw limb from left knee to left foot

camera.drawLimb(userId, SimpleOpenNI.SKEL_RIGHT_HIP, SimpleOpenNI.SKEL_TORSO); //draw limb from right hip to torso
camera.drawLimb(userId, SimpleOpenNI.SKEL_RIGHT_KNEE, SimpleOpenNI.SKEL_RIGHT_FOOT); //draw limb from right knee to right foot

camera.drawLimb(userId, SimpleOpenNI.SKEL_TORSO, SimpleOpenNI.SKEL_LEFT_HIP); //draw limb from torso to left hip
...
void displayData() {

    int XCoord = (round(abs(headPosition.x/10)));
    int YCoord = (round(abs(headPosition.y/10)));
    int ZCoord = round(abs(headPosition.z/10));

    KronosX = ((0*CameraX)+XCoord);
    KronosY = ((0*CameraY)+YCoord);
    KronosZ = ((0*CameraZ)+ZCoord);

    //draw rectangles to clear area where data is displayed
    stroke(125);          // Setting the outline
    strokeWeight(1);
    fill(125);          // Setting the interior of a shape
    rect(0,485,740,235); // Drawing the rectangle

    fill(125);          // Setting the interior of a shape
    rect(0,0,5,485);
    rect(635,5,10,485);
    rect(0,0,1290,5);


    //Title Text
    //  textFont(titlefont,30);
    //  fill(0);
    //  text("Kronos: Beta", a,b);

    //Camera Labels
    //  textFont(datafont,24);
    //  fill(0);
    //  text("Infrared Image", 280, 510);
    //  text("RGB Image", 960, 510);

    //Table Text
    //  textFont(datafont,24);
    //  fill(0);
    //  text("X-Coordinate:", c, d);
    //  text("Y-Coordinate:", c, d+30);
    //  text("Z-Coordinate", c, d+60);
    //  textFont(titlefont,50);
    fill(0);
    text("Kronos", a,b);
    textFont(brianfont, 18);
    text("by Brian Burns", 190, 715);
}

//Camera Labels
textFont(datafont,16);
fill(0);
textFont(datafont,24);
fill(0);
text("Infrared Image", 280, 510);
// text("RGB Image", 960, 510);
// Table Text
textFont(datafont,18);
fill(0);
text("X-Coordinate:", c, d);
text("Y-Coordinate:", c, d+25);
text("Z-Coordinate", c, d+50);
text("Camera", c+130, d-30);
text("Kronos", c+220, d-30);
textFont(datafont,20);
fill(0);
text("Camera Location", c+430, d-40);
textFont(datafont,18);
fill(0);
text(XCoord, c+145, d);
text(YCoord, c+145, d+25);
text(ZCoord, c+145, d+50);
text(KronosX, c+240, d);
text(KronosY, c+240, d+25);
text(KronosZ, c+240, d+50);

/*----------------------------------------------------------
 Listening to Arduino and Sending data
----------------------------------------------------------*/
// sendCoordinates(COORDINATE_TAG,XCoord,YCoord,ZCoord); //8 bytes
// println("Started this thing");

String inString = myPort.readStringUntil("\n"); //read incoming data until end of line
if(inString != null) { //if the incoming data does not equal non-valid data
  // if(inString == "ready"){
    sendCoordinates(COORDINATE_TAG,KronosX,KronosY,KronosZ); //8 bytes
    println("Master, the Coordinates of the rebel spy have been sent.");
    print( inString ); // echo text string from Arduino
    // delay(15);
    // }
} // if(inString != null)

} //displayData()

// void mousePressed() { //Triggers the start of sending data
// sendCoordinates(COORDINATE_TAG,XCoord,YCoord,ZCoord); //8 bytes
// println("Started this thing");
// } //mousePressed()

void sendCoordinates(char tag, int x, int y, int z){
// send the given index and value to the serial port
myPort.write(HEADER);
myPort.write(tag);
myPort.write((char)(x / 256)); // msb
myPort.write(x & 0xff);       // lsb
myPort.write((char)(y / 256)); // msb
myPort.write(y & 0xff);       // lsb
myPort.write((char)(z / 256)); // msb
myPort.write(z & 0xff);       // lsb
//delay(15);
}

//sendCoordinates(char tag, int x, int y, int z)
//VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVvvvvv
vvvvvvvvvvvvvvvvvvvvvvvvvvvVVVVVVV VVVVVV VVVVVV VVVVV

void update(int x, int y) {
  if (overCircle(trackX, trackY, trackSize)) {
    trackOver = true;
    restOver = false;
    passiveOver = false;
    simonOver = false;
  } else if (overRect(restX, restY, restSize, restSize)) {
    restOver = true;
    trackOver = false;
    passiveOver = false;
    simonOver = false;
  } else if (overCircle(passiveX, passiveY, passiveSize)) {
    passiveOver = true;
    trackOver = false;
    restOver = false;
    simonOver = false;
  } else if (overCircle(simonX, simonY, simonSize)) {
    simonOver = true;
    trackOver = false;
    restOver = false;
    passiveOver = false;
  } else {
    trackOver = restOver = passiveOver = simonOver = false;
  }
}

void mousePressed() {
  if (trackOver) {
    //currentColor = trackColor;
    trackColor = color(0, 153, 0);
    passiveColor = color(255);
    simonColor = color(255);
    buttonPressedState = 1;
  }
  if (restOver) {
    //currentColor = restColor;
    trackColor = color(255);
    passiveColor = color(255);
  }
}
simonColor = color(255);
buttonPressedState = 0;
}
if (passiveOver) {
    //currentColor = trackColor;
    passiveColor = color(0, 153, 0);
    trackColor = color(255);
    simonColor = color(255);
    buttonPressedState = 2;
}
if (simonOver) {
    //currentColor = trackColor;
    simonColor = color(0, 153, 0);
    passiveColor = color(255);
    trackColor = color(255);
    buttonPressedState = 3;
}
}

boolean overRect(int x, int y, int width, int height) {
    if (mouseX >= x && mouseX <= x+width &&
        mouseY >= y && mouseY <= y+height) {
        return true;
    } else {
        return false;
    }
}

boolean overCircle(int x, int y, int diameter) {
    float disX = x - mouseX;
    float disY = y - mouseY;
    if (sqrt(sq(disX) + sq(disY)) < diameter/2) {
        return true;
    } else {
        return false;
    }
}

//^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ start stop buttons

void Submit() {
    print("the following text was submitted :");
    url1 = cp5.get(Textfield.class,"Input_X_Distance").getText();
    url2 = cp5.get(Textfield.class,"Input_Y_Distance").getText();
    url3 = cp5.get(Textfield.class,"Input_Z_Distance").getText();
    print(" x = " + url1);
    print(" y = " + url2);
    print(" z = " + url3);
    println();
    CameraX = Integer.valueOf(url1).intValue();
    CameraY = Integer.valueOf(url2).intValue();
    CameraZ = Integer.valueOf(url3).intValue();
}
}  
//for input numbers
Appendix E: Arduino code for Kronos.

//Arduino for Kronos 2.0 Revision 37

// BinaryDataFromProcessing
// These defines must mirror the sending program:
const char HEADER = 'H';
const char COORDINATE_TAG = 'C'; // for head tracking mode input
const char ANOTHER_TAG = 'A'; // Can use for different mode
const int TOTAL_BYTES = 8; // the total bytes in a message

#include <SMARTWAV.h> // include the SMARTWAV library!
SMARTWAV sWav; // create our object called sWAV (smartWAV)

const int topbutton = 41;
const int bottombutton = 43;
const int middleswitch = 38;

int topbuttonState = 0;
int bottombuttonState = 0;
int middleswitchState = 0; // for switching between passive and interactive modes

int previousj = 1;
int j = 1;
int n = 1;
int growlcounter = 1;
int passivesignal = 1;

const int switchLED = 51;

int ChannelOutput[10] = {254, 127, 127, 1, 1, 127, 1, 1, 254, 30}; // resting positions for dragon

unsigned long tailpreviousMillis = 0; // will store last time tail wagged
unsigned long wingpreviousMillis = 0; // will store last time wings flapped
unsigned long blinkpreviousMillis = 0; // will store last time the eyes blinked
unsigned long jawpreviousMillis = 0; // will store last time the jaw opened
unsigned long trackingpreviousMillis = 0; // will store last time dragon tracked

const long tailInterval = 10000; // interval at which the tail will wag
const long wingInterval = 25000; // interval at which the wings will stretch
const long blinkInterval = 6000; // interval at which the eyes will blink
const long jawInterval = 15000; // interval at which the jaw will open and make noise
const long trackingInterval = 60000; // interval at which the dragon will track before taking a rest.

void setup()
{
    Serial.begin(9600); // Communication with the computer
    Serial1.begin(9600); // Communication with Maestro
sWav.init(); //configure the serial and pinout of arduino board for SMARTWAV support
sWav.reset(); //perform an initial reset
pinMode(1,OUTPUT);
pinMode(0,INPUT);

pinMode(topbutton, INPUT);
pinMode(bottombutton, INPUT);
pinMode(middleswitch, INPUT);
pinMode(switchLED, OUTPUT);

int accelChannel = 1; //channel for sending acceleration command

while (accelChannel < 13) {
    Serial1.write(170); //acceleration protocol
    Serial1.write(12);
    Serial1.write(9);
    Serial1.write(accelChannel); //channel 1
    Serial1.write(10);
    Serial1.write(2);

    accelChannel++;

    if (accelChannel == 12) {
        break; //exit the loop
    } //if (accelChannel == 12)
} //while (accelChannel < 13)

Serial.println("Completed initial Acceleration Commands. Ready.");
} //void setup()

//----------------------------------------------------------------------------
// Start Main Loop
//----------------------------------------------------------------------------

void loop(){
    delay(15);
    middleswitchState = digitalRead(middleswitch);
topbuttonState = digitalRead(topbutton);

    if (middleswitchState == HIGH){
        digitalWrite(switchLED, HIGH);

        if ( Serial.available() >= TOTAL_BYTES) {
            if( Serial.read() == HEADER)
            {
                char tag = Serial.read();
                if(tag == COORDINATE_TAG)
{  
    //Collect integers
    int x = Serial.read() * 256;
    x = x + Serial.read();
    int y = Serial.read() * 256;
    y = y + Serial.read();
    int z = Serial.read() * 256;
    z = z + Serial.read();

    /*--------------------------------------------------------------------------------------
     Calibration Equations
     *--------------------------------------------------------------------------------------*/

    if(x>=42 && z<=280 && growlcounter <= 49)  //if user is too close to the dragon get in
    angry growl mode

        ChannelOutput[0] = 1; //neck lift
        ChannelOutput[1] = 190; //Neck bend
        ChannelOutput[2] = 250; //Neck Pivot
        ChannelOutput[3] = 250; //Top Eyelid
        ChannelOutput[4] = 250; //Bottom Eyelid
        ChannelOutput[5] = 63;  //Head Rotate
        ChannelOutput[6] = 250; //Left Wing
        ChannelOutput[7] = 250; //Right Wing
        ChannelOutput[9] = 240; //Jaw
     Serial1.write(0xFF); //Short Position Command protocol
     Serial1.write(10);   //Channel
     Serial1.write(240);  //Target Position
    sWav.playTrackName("burp");  //call the song by it's name, don't add the
    "\.wav" extension
    growlcounter++;

    }

    if(x>=42 && z<=280 && growlcounter >= 50){

        ChannelOutput[0] = 254; //neck lift
        ChannelOutput[1] = 127;  //Neck bend
        ChannelOutput[2] = 127;  //Neck Pivot
        ChannelOutput[3] = 60;   //Top Eyelid
        ChannelOutput[4] = 90;   //Bottom Eyelid
        ChannelOutput[5] = 127;  //Head Rotate
        ChannelOutput[6] = 1;    //Left Wing
        ChannelOutput[7] = 1;    //Right Wing
        ChannelOutput[9] = 30;   //Jaw
     Serial1.write(0xFF); //Short Position Command protocol
     Serial1.write(10); //Channel
     Serial1.write(30); //Target Position

    }
else if (x<42 && n<=30) { //if no user is too close to the dragon

//j = (((round(ChannelOutput[1]))/10)*10);  

// if(j==previousj) {
//    n++;
// }  

// if(n<=25) {

ChannelOutput[0] = 1; //neck lift

ChannelOutput[1] = constrain(round(510+(8.031*x)-(2.12*z)-(0.04156*x*z)+0.004461*z*z), 1, 254); //Neck Bend  

ChannelOutput[2] = 127;  //Neck Pivot 

ChannelOutput[3] = 250;  //Top Eyelid  

ChannelOutput[4] = 250;  //Bottom Eyelid  

ChannelOutput[5] = round(0.0006*(ChannelOutput[1])*(ChannelOutput[1])-(0.3055*(ChannelOutput[1]))+128.4);  //Head Rotate  

ChannelOutput[6] = 1;  //Left Wing  

ChannelOutput[7] = 1;  //Right Wing  

ChannelOutput[9] = 30;  //Jaw  

Serial1.write(0xFF); //Short Position Command protocol  

Serial1.write(10); //Channel  

Serial1.write(30); //Target Position  

// } //if(n<=100)

j = (((round(ChannelOutput[1]))/10)*10);

if(j=previousj){
    n++;  
    previousj = j;
}  

}

else if(x<42 && n>=30) {  

ChannelOutput[0] = 254; //neck lift  

ChannelOutput[1] = 127;  //Neck bend  

ChannelOutput[2] = 127;  //Neck Pivot  

ChannelOutput[3] = 60;  //Top Eyelid  

ChannelOutput[4] = 90;  //Bottom Eyelid  

ChannelOutput[5] = 127;  //Head Rotate  

ChannelOutput[6] = 1;  //Left Wing  

ChannelOutput[7] = 1;  //Right Wing  

Serial1.write(0xFF); //Short Position Command protocol  

Serial1.write(10); //Channel  

Serial1.write(30); //Target Position  

if(j=! previousj) {
    n=1; //reset here?  
}  

}
// } //if(j>=100)
// } //if(j==previousj)

////////////////////////////////////////////////////////////////////////////////
for resetting if no user tracked
    Serial.print("Master, Arduino has received the following coordinates | x:");
    Serial.print(x);
    Serial.print(" | y:");
    Serial.print(y);
    Serial.print(" | z:");
    Serial.println(z);

previousj = j;

////////////////////////////////////////////////////////////////////////////////
---working in here for if user gets too close, growl
} //if(tag == COORDINATE_TAG)
else {
    //else anything? Add sleeping mode here.
    // } //else
} //if ( Serial.read() == HEADER)
} //if ( Serial.available() >= TOTAL_BYTES)

} //if ( middleswitchState == HIGH)
else { //Eventually passive behavior. For now, resting position.
    digitalWrite(switchLED, LOW);

    passive();
}

} //else

if (topbuttonState == HIGH) { //resting position and hold for 20 seconds
    ChannelOutput[0] = 254; //neck lift
    ChannelOutput[1] = 127; //Neck bend
    ChannelOutput[2] = 127; //Neck Pivot
    ChannelOutput[3] = 60; //Top Eyelid
    ChannelOutput[4] = 90; //Bottom Eyelid
    ChannelOutput[5] = 127; //Head Rotate
    ChannelOutput[6] = 1; //Left Wing
    ChannelOutput[7] = 1; //Right Wing
    ChannelOutput[9] = 30; //Jaw

    int PositionChannel = 1; //channel for sending Position command
while (PositionChannel < 11) {
    Serial1.write(0xFF); //Short Position Command protocol
    Serial1.write(PositionChannel); //Channel
    Serial1.write(ChannelOutput[PositionChannel - 1]); //Target Position

    PositionChannel++;
    if (PositionChannel == 11)
        break; //exit the loop
} //if (PositionChannel == 10)
} //while (PositionChannel < 10)
    delay(2000); //delay to hold the resting position
} /*----------------------------------------------------------------------------------

Writing Out to Mini Maestro
----------------------------------------------------------------------------------*/

int PositionChannel = 1; //channel for sending Position command
while (PositionChannel < 11) {
    Serial1.write(0xFF); //Short Position Command protocol
    Serial1.write(PositionChannel); //Channel
    Serial1.write(ChannelOutput[PositionChannel - 1]); //Target Position

    PositionChannel++;
    if (PositionChannel == 11)
        break; //exit the loop
} //if (PositionChannel == 11)
} //while (PositionChannel < 11)

/*----------------------------------------------------------------------------------*/
} //void loop()

void passive() {
    Serial1.write(0xFF); //Short Position Command protocol
    Serial1.write(1); //Channel
    Serial1.write(254); //Target Position
    delay(5000);
}

void Tail() {
    unsigned long tailcurrentMillis = millis();
}
if(tailcurrentMillis - tailpreviousMillis >= tailInterval) {
    // save the last time dragon wagged tail
    tailpreviousMillis = tailcurrentMillis;
    // if the tail is wagging turn off, if not, turn on
    if (ChannelOutput[8] == 254) {
        ChannelOutput[8] = 80;
    } else
        ChannelOutput[8] = 254;
} //if(tailcurrentMillis - tailpreviousMillis >= tailInterval)
} //void Tail()

/*

void Wing() {
    unsigned long wingcurrentMillis = millis();

    if(wingcurrentMillis - wingpreviousMillis >= wingInterval) {
        // save the last time dragon opened wings
        wingpreviousMillis = wingcurrentMillis;
        // if it's time to open the wings.
        if (ChannelOutput[6] == 1) {
            ChannelOutput[6] = 1;
            ChannelOutput[7] = 1;
        } else
            ChannelOutput[6] = 1;
        ChannelOutput[7] = 1;
    } //if(tailcurrentMillis - tailpreviousMillis >= tailInterval)
} //void Tail()

void Blink() {
    unsigned long blinkcurrentMillis = millis();

    if(blinkcurrentMillis - blinkpreviousMillis >= blinkInterval) {
        // save the last time dragon blinked
        blinkpreviousMillis = blinkcurrentMillis;
        // add another so dragon blinks quickly
        if (ChannelOutput[3] == 60) {
            ChannelOutput[3] = 60;
            ChannelOutput[4];
        } else
            ChannelOutput[3] = 60;
        ChannelOutput[4] = 60;
    } //if(blinkcurrentMillis - blinkpreviousMillis >= blinkInterval)
} //void Blink()

void Jaw() {
    unsigned long jawcurrentMillis = millis();

    if(jawcurrentMillis - jawpreviousMillis >= jawInterval) {
        // save the last time dragon opened mouth
        jawpreviousMillis = jawcurrentMillis;
        // do something
    } //if(jawcurrentMillis - jawpreviousMillis >= jawInterval)
} //void Jaw()
jawpreviousMillis = jawcurrentMillis;
// need to add another so the jaw closes after opening
if (ChannelOutput[5] == 127){
    ChannelOutput[5] = 127;
}
else
    ChannelOutput[5] = 127;
} //if(jawcurrentMillis - jawpreviousMillis >= jawInterval)
} //void Jaw()

void Track() {
    unsigned long trackingcurrentMillis = millis();

    if(trackingcurrentMillis - trackingpreviousMillis >= trackingInterval) {
        // save the last time dragon tracked
        trackingpreviousMillis = trackingcurrentMillis;
        // do something
    } //if(trackingcurrentMillis - trackingpreviousMillis >= trackingInterval)
} //void Track()
Appendix F: Mini Maestro passive behavior script.

<UsrSettings version="1">
  <NeverSuspend>false</NeverSuspend>
  <SerialMode>UART_FIXED_BAUD_RATE</SerialMode>
  <FixedBaudRate>9600</FixedBaudRate>
  <SerialTimeout>0</SerialTimeout>
  <EnableCrc>false</EnableCrc>
  <SerialDeviceNumber>12</SerialDeviceNumber>
  <SerialMiniSscOffset>0</SerialMiniSscOffset>
  <EnablePullups>false</EnablePullups>
  <Channels MiniMaestroServoPeriod="80000" ServoMultiplier="1">
    <!--Period = 20 ms-->
    <!--Channel 0-->
    <Channel name="" mode="Servo" min="3008" max="9024" homemode="Off" home="3008" speed="0" acceleration="0" neutral="6000" range="1651" />
    <!--Channel 1-->
    <Channel name="Neck Lift" mode="Servo" min="4032" max="4608" homemode="Goto" home="4608" speed="0" acceleration="0" neutral="4032" range="3429" />
    <!--Channel 2-->
    <Channel name="Neck Bend" mode="Servo" min="4480" max="7616" homemode="Goto" home="6108" speed="10" acceleration="3" neutral="6000" range="3429" />
    <!--Channel 3-->
    <Channel name="Neck Pivot" mode="Servo" min="5760" max="7232" homemode="Goto" home="6480" speed="0" acceleration="5" neutral="6528" range="3429" />
    <!--Channel 4-->
    <Channel name="Top Eyelid" mode="Servo" min="3968" max="4736" homemode="Goto" home="3968" speed="0" acceleration="5" neutral="3968" range="1905" />
    <!--Channel 5-->
    <Channel name="Bottom Eyelid" mode="Servo" min="5376" max="6656" homemode="Goto" home="5376" speed="0" acceleration="5" neutral="5440" range="1905" />
    <!--Channel 6-->
    <Channel name="Head Rotate" mode="Servo" min="3392" max="7616" homemode="Goto" home="5352" speed="0" acceleration="5" neutral="6000" range="1905" />
    <!--Channel 7-->
    <Channel name="Left Wing" mode="Servo" min="3136" max="8832" homemode="Goto" home="3136" speed="0" acceleration="5" neutral="6336" range="1905" />
    <!--Channel 8-->
    <Channel name="Right Wing" mode="Servo" min="3712" max="8640" homemode="Goto" home="3712" speed="0" acceleration="5" neutral="6000" range="1905" />
    <!--Channel 9-->
    <Channel name="Tail Lift" mode="Servo" min="5568" max="6016" homemode="Goto" home="6000" speed="0" acceleration="5" neutral="6016" range="1905" />
    <!--Channel 10-->
    <Channel name="Jaw" mode="Servo" min="3904" max="6208" homemode="Goto" home="4080" speed="0" acceleration="5" neutral="5888" range="1905" />
    <!--Channel 11-->
    <Channel name="" mode="Servo" min="3648" max="6016" homemode="Goto" home="5400" speed="0" acceleration="5" neutral="5952" range="1905" />
  </Channels>
</UsrSettings>
<Channels>
<!--Channel 12-->
<Channel name="" mode="Servo" min="4416" max="9024" homemode="Off" home="4416"
speed="0" acceleration="0" neutral="6000" range="1905"/>
<!--Channel 13-->
<Channel name="" mode="Servo" min="3072" max="8960" homemode="Off" home="3072"
speed="0" acceleration="0" neutral="6000" range="1905"/>
<!--Channel 14-->
<Channel name="" mode="Servo" min="3968" max="8000" homemode="Off" home="3968"
speed="0" acceleration="0" neutral="6000" range="1905"/>
<!--Channel 15-->
<Channel name="" mode="Servo" min="3968" max="8000" homemode="Off" home="3968"
speed="0" acceleration="0" neutral="6000" range="1905"/>
<!--Channel 16-->
<Channel name="" mode="Servo" min="3008" max="9024" homemode="Off" home="3008"
speed="0" acceleration="0" neutral="6000" range="1905"/>
<!--Channel 17-->
<Channel name="Head Rotate" mode="Servo" min="3200" max="6336" homemode="Goto"
home="5200" speed="0" acceleration="0" neutral="6000" range="1905"/>
<!--Channel 18-->
<Channel name="" mode="Servo" min="3968" max="8000" homemode="Off" home="3968"
speed="0" acceleration="0" neutral="6000" range="1905"/>
<!--Channel 19-->
<Channel name="" mode="Servo" min="3968" max="8000" homemode="Off" home="3968"
speed="0" acceleration="0" neutral="6000" range="1905"/>
<!--Channel 20-->
<Channel name="" mode="Servo" min="3008" max="8000" homemode="Off" home="3008"
speed="0" acceleration="0" neutral="6000" range="1905"/>
<!--Channel 21-->
<Channel name="" mode="Servo" min="3968" max="8000" homemode="Off" home="3968"
speed="0" acceleration="0" neutral="6000" range="1905"/>
<!--Channel 22-->
<Channel name="" mode="Servo" min="2880" max="8000" homemode="Off" home="2880"
speed="0" acceleration="0" neutral="6000" range="1905"/>
<!--Channel 23-->
<Channel name="" mode="Servo" min="3968" max="8000" homemode="Off" home="3968"
speed="0" acceleration="0" neutral="6000" range="1905"/>
</Channels>

<Sequences>
<!--Kronos Motion Demo 1-->
<Sequence name="Kronos Motion Demo 1">
<Frame name="Frame 0" duration="2200">0 6016 6108 6480 3968 5376 5352 3136 3712
6000 4080 5400 0 0 0 0 0 5200 0 0 0 0 0</Frame>
<Frame name="Frame 1" duration="2000">0 6000 6108 6480 5019 6576 5352 3136 3712
6000 4080 5400 0 0 0 0 0 5200 0 0 0 0 0</Frame>
<Frame name="Neck Up" duration="2500">0 4032 4845 6731 5024 6609 5352 3136 3712
6000 4080 5400 0 0 0 0 0 5200 0 0 0 0 0</Frame>
<Frame name="Frame 3" duration="2000">0 6035 4601 7024 5120 6589 5352 3136 3712
6000 6118 5400 0 0 0 0 0 5200 0 0 0 0 0</Frame>
<Frame name="Frame 4" duration="2000">0 6035 7357 5995 5120 6589 5352 3136 3712
5380 4038 5400 0 0 0 0 0 5200 0 0 0 0 0</Frame>
<Frame name="Frame 5" duration="2000">0 6035 5286 6481 5120 6589 4910 3136 3712
5380 4038 5400 0 0 0 0 0 5200 0 0 0 0 0</Frame>
</Sequence>
</Sequences>
<Frame name="Frame 6" duration="2000">0 6035 5286 6481 5120 6589 5768 3136 3712 5380 4038 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 7" duration="2000">0 6035 6078 6481 5120 6589 5225 8638 8472 5380 4038 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 8" duration="2000">0 6035 7128 6310 5120 6589 5225 3274 3712 6009 4038 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 9" duration="500">0 6035 6961 6310 3984 5376 5225 3274 3712 6009 4038 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 10" duration="750">0 6035 6961 6310 5108 6720 5225 3274 3712 6009 4038 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Neck down" duration="2500">0 8000 5987 6567 5108 6720 5225 3274 3712 6009 4038 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 12" duration="10000">0 6000 6108 6480 3968 5376 5352 3136 3712 6000 4080 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Sequence>
<Sequence name="Kronos Passive 1">
<Frame name="Frame 1" duration="2200">0 4608 6000 6528 3968 5376 4622 4446 4110 6016 4080 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 1" duration="900">0 4608 6000 6528 3968 5376 4048 4446 4110 6016 4080 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 2" duration="600">0 4608 6000 6528 3968 5376 4458 4446 4110 6016 4080 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 3" duration="800">0 4608 6000 6528 3968 5376 5032 4446 4110 6016 4080 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 4" duration="1100">0 4608 6108 6480 4736 6656 5352 3136 3712 6000 4080 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 5" duration="2000">0 4032 5560 6480 4736 6656 5352 3136 3712 6000 4080 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 6" duration="1200">0 4032 4860 6781 3968 5376 4458 4446 4110 6016 4080 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 7" duration="1500">0 4032 5895 6781 3968 5376 4806 4446 4110 6000 4080 5400 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 8" duration="1300">0 4032 7433 6131 4736 6656 5352 3136 3712 5568 4172 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 9" duration="1300">0 4032 7265 6038 4736 6656 5011 3136 3712 5568 4172 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 10" duration="1300">0 4141 5454 6939 4736 6656 5011 3136 3712 5568 3904 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 11" duration="1300">0 4141 5454 7210 4736 6656 5011 8804 8640 5568 3904 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 12" duration="1800">0 4141 5454 7210 4736 6656 4786 8804 8640 6016 3904 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 13" duration="1100">0 4141 6656 6403 4736 6656 4786 8804 8640 6016 3904 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 14" duration="1500">0 4141 6656 6403 4736 6656 5421 3136 3712 6016 3904 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 15" duration="1500">0 4141 6656 6403 4736 6656 5093 3136 3712 6016 3904 0 0 0 0 5200 0 0 0 0 0 0</Frame>
<Frame name="Frame 16" duration="1400">0 4141 6656 6403 3968 5376 5093 3136 3712 6016 3904 0 0 0 0 5200 0 0 0 0 0 0</Frame>
</Sequence>
<table>
<thead>
<tr>
<th>Time</th>
<th>Coordinates</th>
<th>Frame Numbers</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>4032 5560</td>
<td>frame_1_2</td>
<td># Frame 5</td>
</tr>
<tr>
<td>1200</td>
<td>4860 6781</td>
<td>frame_2..5_10_11</td>
<td># Frame 6</td>
</tr>
<tr>
<td>1500</td>
<td>5895 4806</td>
<td>frame_2_6</td>
<td># Frame 7</td>
</tr>
<tr>
<td>1300</td>
<td>7433 6131</td>
<td>frame_2..5_10</td>
<td># Frame 8</td>
</tr>
<tr>
<td>900</td>
<td>7265 6038</td>
<td>frame_2..3_6_9</td>
<td># Frame 9</td>
</tr>
<tr>
<td>1000</td>
<td>4141 5454</td>
<td>frame_1..3_10</td>
<td># Frame 10</td>
</tr>
<tr>
<td>1300</td>
<td>7210 8804</td>
<td>frame_3..7_8</td>
<td># Frame 11</td>
</tr>
<tr>
<td>1800</td>
<td>4786 6016</td>
<td>frame_6_9</td>
<td># Frame 12</td>
</tr>
<tr>
<td>1100</td>
<td>6656 6403</td>
<td>frame_2..7_8</td>
<td># Frame 13</td>
</tr>
<tr>
<td>1500</td>
<td>5421</td>
<td>frame_6</td>
<td># Frame 14</td>
</tr>
<tr>
<td>1500</td>
<td>5093</td>
<td>frame_6</td>
<td># Frame 15</td>
</tr>
<tr>
<td>1400</td>
<td>3968 5376</td>
<td>frame_4..5</td>
<td># Frame 16</td>
</tr>
<tr>
<td>1200</td>
<td>4736 6656</td>
<td>frame_4..5</td>
<td># Frame 17</td>
</tr>
<tr>
<td>2100</td>
<td>5439 5568</td>
<td>frame_2..9</td>
<td># Frame 18</td>
</tr>
<tr>
<td>2000</td>
<td>7128 6174</td>
<td>frame_2..3_6</td>
<td># Frame 19</td>
</tr>
<tr>
<td>1600</td>
<td>4693</td>
<td>frame_2</td>
<td># Frame 20</td>
</tr>
<tr>
<td>1600</td>
<td>4571 6810</td>
<td>frame_2..3_9</td>
<td># Frame 21</td>
</tr>
<tr>
<td>1000</td>
<td>4032 5484</td>
<td>frame_1..2_6</td>
<td># Frame 22</td>
</tr>
<tr>
<td>600</td>
<td>5536</td>
<td>frame_10</td>
<td># Frame 23</td>
</tr>
<tr>
<td>600</td>
<td>5729</td>
<td>frame_6</td>
<td># Frame 24</td>
</tr>
<tr>
<td>1500</td>
<td>6991 5917</td>
<td>frame_2..3_6</td>
<td># Frame 25</td>
</tr>
<tr>
<td>1900</td>
<td>5278 4105</td>
<td>frame_6_10</td>
<td># Frame 26</td>
</tr>
<tr>
<td>1700</td>
<td>5180 6488</td>
<td>frame_2..3</td>
<td># Frame 27</td>
</tr>
<tr>
<td>2000</td>
<td>4608 6352</td>
<td>frame_1..2_7</td>
<td># Frame 28</td>
</tr>
<tr>
<td>1700</td>
<td>7276</td>
<td>frame_8</td>
<td># Frame 29</td>
</tr>
<tr>
<td>1900</td>
<td>3136 7396</td>
<td>frame_7..8</td>
<td># Frame 30</td>
</tr>
<tr>
<td>1200</td>
<td>3712</td>
<td>frame_8</td>
<td># Frame 31</td>
</tr>
<tr>
<td>1200</td>
<td>5568</td>
<td>frame_9</td>
<td># Frame 32</td>
</tr>
<tr>
<td>1000</td>
<td>6016</td>
<td>frame_9</td>
<td># Frame 33</td>
</tr>
<tr>
<td>10200</td>
<td>3968 5376</td>
<td>frame_4..5</td>
<td># Frame 34</td>
</tr>
</tbody>
</table>

repeat

**sub frame_0..23**

23 servo
22 servo
21 servo
20 servo
19 servo
18 servo
17 servo
16 servo
15 servo
14 servo
13 servo
12 servo
11 servo
10 servo
9 servo
8 servo
7 servo
6 servo
5 servo
4 servo
3 servo
2 servo
1 servo
0 servo
delay
return

sub frame_6
6 servo
delay
return

sub frame_2..9
9 servo
8 servo
7 servo
6 servo
5 servo
4 servo
3 servo
2 servo
delay
return

sub frame_1_2
2 servo
1 servo
delay
return

sub frame_2..5_10..11
11 servo
10 servo
5 servo
4 servo
3 servo
2 servo
delay
return

sub frame_2_6
6 servo
2 servo
delay
return

sub frame_2..5_10
10 servo
5 servo
4 servo
3 servo
2 servo
delay
return

sub frame_2_3_6_9
9 servo
6 servo
3 servo
2 servo
delay
return

sub frame_1..3_10
10 servo
3 servo
2 servo
1 servo
delay
return

sub frame_3_7_8
8 servo
7 servo
3 servo
delay
return

sub frame_6_9
9 servo
6 servo
delay
return

sub frame_2_3_7_8
8 servo
7 servo
3 servo
2 servo
delay
return

sub frame_4_5
5 servo
4 servo
delay
return

sub frame_2_9
9 servo
2 servo
delay
return

sub frame_2_3_6
  6 servo
  3 servo
  2 servo
delay
return

sub frame_2
  2 servo
delay
return

sub frame_2_3_9
  9 servo
  3 servo
  2 servo
delay
return

sub frame_10
  10 servo
delay
return

sub frame_6_10
  10 servo
  6 servo
delay
return

sub frame_2_3
  3 servo
  2 servo
delay
return

sub frame_1_2_7
  7 servo
  2 servo
  1 servo
delay
return

sub frame_8
  8 servo
delay
return
sub frame_7_8
  8 servo
  7 servo
  delay
  return

sub frame_9
  9 servo
  delay
  return

</Script>
</UscSettings>