School of Engineering Science • Burnaby, BC • V5A 1S6
sensit-ensc@sfu.ca

March 4, 2007

Dr. Lakshman One
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

Re: ENSC 440 Design Specification for Motion Capture System

Dear Dr. One:

The attached document, *Design Specification for a Motion Capture System*, outlines the design specifications for our ENSC 440 project. We are in the process of designing and developing a real time system which captures the position and orientation of an object, providing a 2D digital representation and simulation of the object's movements on a computer. The system may be used later as an analysis tool to study the human body movements in different applications such as dance performances.

The purpose of this design specification is to provide a detailed overview of the design criteria and the testing plans that our completed Motion Capture System will fulfill. The document lists the applicable design plan required for a successful project completion in April 2007. Please note that some sections of the document are fairly detailed since group members will be using this document as a reference for their design.

SensIT Technology Ltd. consists of four inventive, motivated, and devoted engineering students: Azadeh Jamalian, Ata Naemi, Sa'ed Abu-Alhaija, and Ivan Lee. If you have any questions or concerns about our proposal, please do not hesitate to contact me by phone at (604) 780-6583 or by e-mail at sensit-ensc@sfu.ca.

Sincerely,

Azadeh Jamalian
President and CEO
SensIT Technology Ltd.

Enclosure: *Design Specification for a Motion Capture System*

Design Specification for a

Motion Capture System

Project Team: Azadeh Jamalian
Atae Naemi
Sunghoon Ivan Lee
Sa'ed Abu-Alhaija

Contact Information: Azadeh Jamalian
sensit-ensc@sfu.ca

Submitted to: Lakshman One (ENSC 440)
Steve Whitmore (ENSC 305)
School of Engineering Science
Simon Fraser University

Submitted Date: March 4, 2007

Revision: Version 1.0

Executive Summary

During the past decade, utilizing the motion capture technology to compose or conduct electrical music based on the movements of the dancers has been of high interest for composers and choreographers, bringing together visual artists, composers, dancers and finally engineers. The SensIT Motion Capture System is the starting platform for the motion capture system which will be utilized by professional dancers as an analysis tool to study their dance movements. Our low cost system will be capable of capturing precision and real time movements of an object.

The development of the Motion Capture System will occur in two stages. After the completion of the first stage of development, the system will provide the position and orientation of only some specified joints of a doll's body sufficient to illustrate an overview of the motion in 2D. The device will have the following features:

1. Captures the position and orientation of different joints of a doll and writes the horizontal and vertical coordinates as well as the angle of each specified joint to a specific file.
2. Simulates the motion of the doll in the form of real time 2D animation on a computer monitor.
3. Provides a user friendly menu to start and end the application, change the location of the object on the screen for study purposes, and to visualize the coordinates of each joint.
4. Provides a document which outlines the resolution, precision, limitations and specifications of the system for potential users and perhaps future developments.

After the second phase of development, the device may also:

1. Be extendable for tracking more number of joints of an actual human body.
2. Provide the simulation of the motion in 3D, i.e. allow more degrees of freedom for the animation.
3. Have a bigger working volume and be more precise.
4. Have a wider range of applications such as music composition, biomedical and robotic researches, and video games development.

The first phase of development in the SensIT Motion Capture System will be completed in April 2007.

Table of Contents

List of Tables

List of Figures

1 Introduction

The Motion Capture System is a system which captures and simulates the movement of a dancing doll. Currently, at Simon Fraser University three departments of Computing, Music, and Engineering are intrigued by the project which focuses on studying dance movements and composing or conducting music inspired from these movements. Particularly, these departments are interested in our project since it has the potential to become the initial step of the bigger project involving the three departments.

SensIT Motion Capture System is a real time system which captures the 2D movement of an object (a small doll) and through further developments it can be utilized as an analysis tool to study the movements of an actual dancer. The data obtained from the analysis tool can be fed to composition software to create composition data or directed to synthesizer software to generate the corresponding audio sounds.

The system uses infrared (IR) LED's which are mounted on the object, as well as a USB webcam which is connected to the computer that displays the animation. The images/frames obtained by the webcam will be analyzed by customized analytical software tools and the results are eventually used by the animation software for simulating movements. The project can later be developed to capture and study human movements, which will involve making a wearable suit with infrared (IR) LED's mounted on it.

1.1 Scope

This document describes the design specifications of the Motion Capture System. A full list of design requirements and testing plans is provided for the proof of concept as well as to define the framework for the design and development of the system. The completed SensIT system may be modified to develop a more sophisticated system that can be used by an actual dancer. Minor modifications might be required upon further tests and analyses.

1.2 Glossary

IR Infrared
LED Light Emitting Diode
MFC Microsoft Foundation Class
MOS Metal-Oxide Semiconductor
RGB Red, Green, Blue
SDK Software Development Kit
USB Universal Serial Bus

1.3 Referenced Documents

[1] http://www.discovercircuits.com/DJ-Circuits/revbat.htm
[2] http://www.pages.drexel.edu/~kws23/tutorials/PICTutorial/PICTutorial.html

[3] http://ww1.microchip.com/downloads/en/devicedoc/35007b.pdf
[4] "Design with Operational Amplifiers and Analog integrated Circuits", Sergio Franco, 3rd edition.
[5] "Introduction to Robotics' Mechanics and Control", John J. Craig, 3rd edition
[6] LTI1961 Data Sheet, Linear Technology Corporation 2001
[7] Design Notes on Boost Regulator, Linear Technology Corporation 2003
[8] www.designers_guide.org
[9] Product Catalog for Logitech Quickcam 6.0
[10] http://www.lpi.usra.edu/education
[11] Krssagar, Simultaneous Previewing & Video Capture suing DirectShow, April 16, 2004, < http://www.codeproject.com/audio/DXCapture.asp>
[12] PIC 16F84A Data Sheet, Microchip Technology Inc.

1.4 Intended Audience

This document provides the framework for the hardware and software engineers to develop their modules and create the complete system. The SensIT engineers will use this document as a reference for the system design.

Through the use of this document, the CEO can manage the group and ensure that each required task is completed on time. The hardware and software managers will also find this document valuable for verifying the project design.

Finally, the marketing manager may use this document to develop initial promotional material.

2 System Overview

The system can be divided into four sub-systems: a control unit, an optical capture unit, image analysis software, and 2D animation software. The control unit consists of the microcontroller and the IR LED's which are attached to the object under study. The control unit sends infrared radiation to the optical capturing device, which consists of a USB webcam. The raw information captured by the USB webcam is then sent to the main computer. The image analysis software analyzes the raw data and finally sends the calculated coordinates of the object to the 2D animation software. The functional description of the 2D motion capture system can be illustrated in the following diagrams.

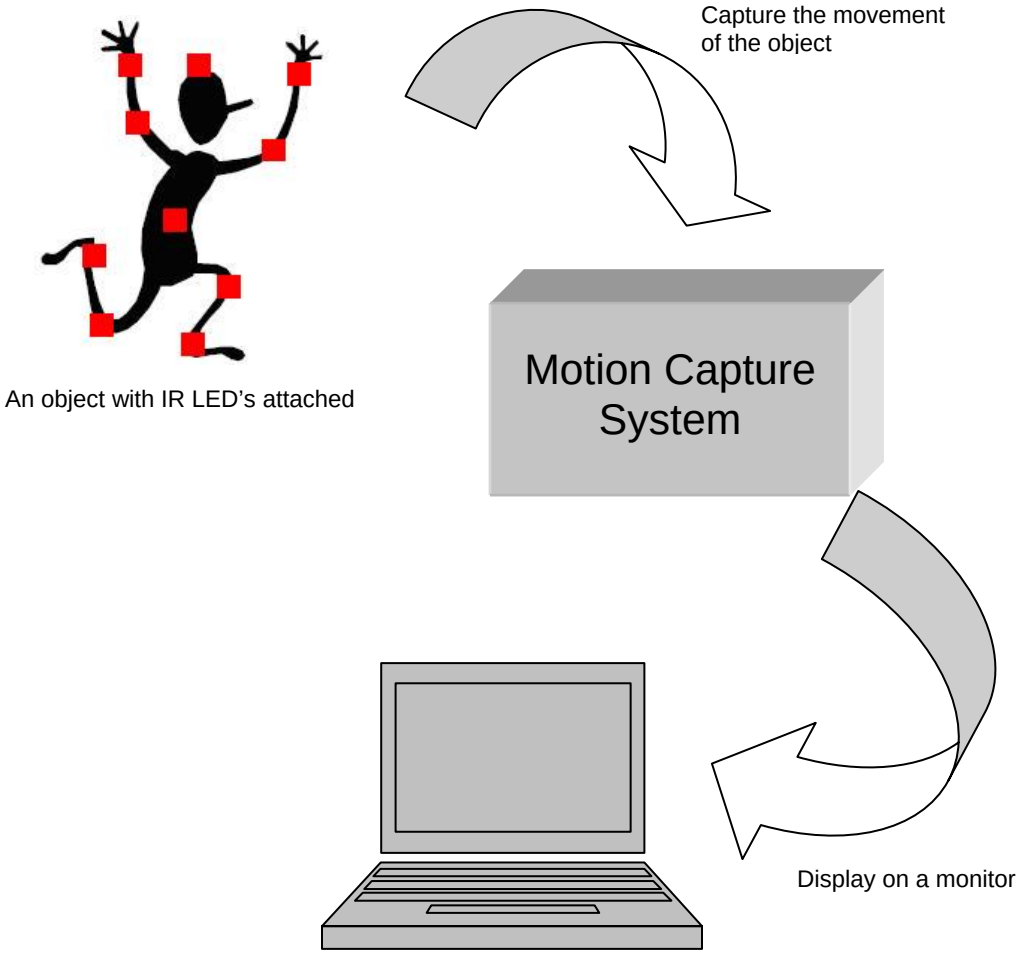

Figure 1: Graphical description of the system

Figure 2 illustrates how the system will be processed. When the system is activated by the user, the USB webcam starts capturing frames from the doll in a way that in each frame only one of the markers is on. These frames will be analyzed and processed in analytical software. The animation software then, creates a two dimensional animation image based on the data received from the analytical software and displays it on a monitor. The process is repeated until the user deactivates the system.

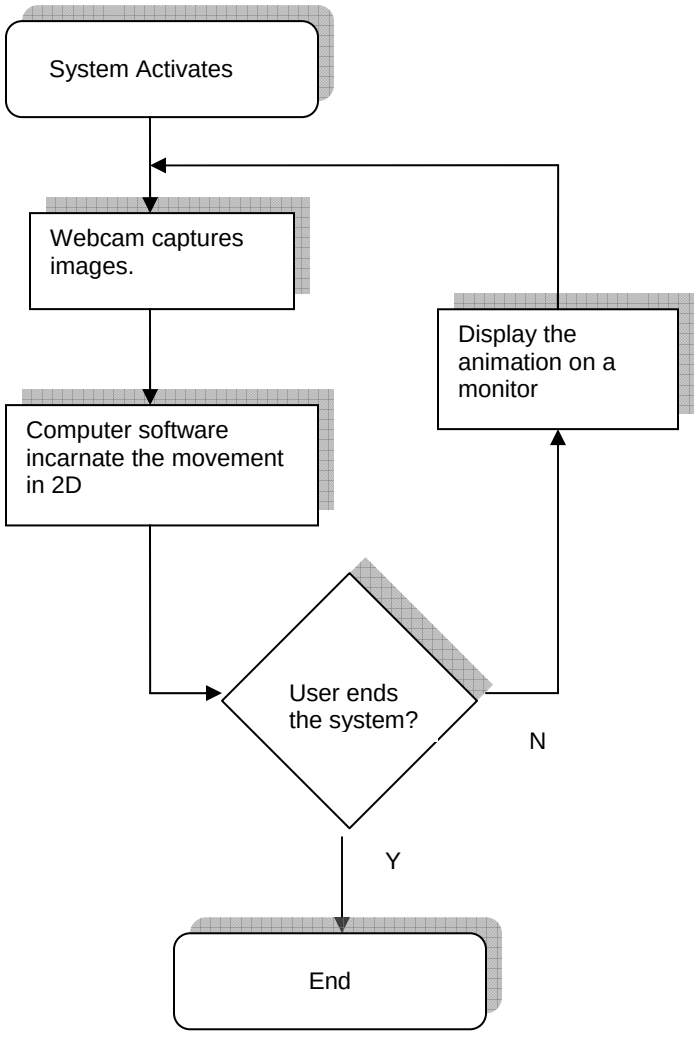

Figure 2: Flow Chart of the System

3 System Hardware

The hardware of the system consists of three sub-systems, the control unit, the optical markers network, and the optical capture unit. The control unit basically controls the optical markers (IR LED's) to turn on one LED at a time. Synchronization is required between the optical markers network and the optical capture unit for proper operation. Time and financial limitations have affected many of the decisions for selecting the parts. Generally, the readily available components are preferred if they meet the minimum requirements specified in the *Functional Specification – Section 4: Hardware Requirements* as well as *Section 3.5: Environmental Requirements*.

We will now discuss the design and components selection of each of the sub systems which together, make up the entire hardware system. Several professors such as Dr. Lucky One and Dr. Patrick Leung have been consulted for their professional opinion to help us choosing the appropriate components.

3.1 Control Unit

The purpose of the control unit is the turn on one IR LED at a time synchronized with the USB webcam. The schematic drawing of the control unit is shown in Figure 23. The control unit includes many components based around the microcontroller. In the following sections, the components selected are presented along with justifications of how they meet the requirements set by the *Functional Specification – Section 4.1: Control Unit*.

3.1.1 Microcontroller

The choice of the microcontroller is crucial for proper operation of the system. At the start of the selection process, many factors were taken into account such as power requirements, number of I/O's, availability along with the development tools, and re-programmability. We found out that the engineering laboratory at SFU has a development kit (PICSTART PLUS) available for students. The development kit is for programming Microchip PIC microcontrollers. The code for the microcontroller can be written in assembly language using MPLAB software which is also available at the engineering laboratory.

We considered all of the PIC microcontrollers available from Microchip and we took a trip to the local components stores to find the readily available parts. The requirements specified in the Functional Specification are satisfied by the microcontroller PIC16F84A. The low power and re-programmability features of this PIC made it most suitable for our application.

A switch is used to turn on and off the system. A delay is added through the software to act as a switch debounce. The power supply as well as the power protection circuits will be discussed in more detail in later sections.

The higher the frequency of the oscillator, the more power the microcontroller consumes. A group member already had a 4MHz crystal which fits well in our design since the microcontroller only takes a current of 1.8mA at 5.5 VDD with the 4MHz crystal [12]. We intend to run the microcontroller with lower VDD (3V) which lowers the current taken even more.

As shown in Figure 23, PORTA(0..3) is used to control the decoder. The flow chart of the code is attached at the end of this document. In order to further lower the power consumption of the microcontroller, it is placed in the power-down mode upon execution of the "sleep" instruction. This instruction is executed when the microcontroller is waiting for the timer to turn on the next IR LED through the decoder. Initially, we considered using a timer interrupt to wake up the microcontroller from the sleep mode. However, the timer interrupt function with the PIC16F84A does not wake-up the microcontroller from sleep mode. Therefore, we decided to use the Watchdog Timer (WDT) to wake-up the microcontroller from sleep to select the next LED to turn on. The Watchdog Timer is a free running On-Chip RC Oscillator which does not require any external components. While the device in sleep mode and WDT is enabled, it draws about 5uA. Finally, the microcontroller's small size and light weight make it ideal for our application.

3.1.2 4-6 Line Decoder

We initially intended to have 10 IR LED's placed on the object to meet the minimum requirements specified in the *Functional Specification – Section 4.1: Control Unit*. Upon further investigation of the animation software, we realized that having 14 IR LED's would be more suitable. Therefore, the purpose of using the decoder is to expand the outputs of the microcontroller as well as allow for a more flexible design by freeing I/O's of the microcontroller to be utilized for other functions. The high speed C-MOS technology 4-to-16 line decoder with part number 74HC154 is suitable for our purposes and is also available from the engineering department.

3.1.3 Battery Power Supply

The power supply is implemented using a battery to allow for the object to move freely with now attachments to cables for power supply. The criteria for the battery are its weight, power capacity and availability. We found that the lithium ion batteries are light weight, cheap, available at regular retail stores, and generally have enough power for the LED's and the microcontroller. The lithium ion battery with part number CR2032 has a typical capacity of 230mAh. We intend to turn on the LED's one at a time drawing current of 10-20mA each, the amount of current will be decided at the integration stage with the webcam and the software). We also intend to turn on the LED's for a very short period of time (approximately 0.004second per cycle), the amount of time will also be decided at the integration stage. Therefore, we have 14 LED's and each of them is on about twice per second which results in a total current drawn by the LED's of 2.24mA. The microcontroller spends about 90% of the time in sleep mode; therefore, it approximately takes 0.225mA. Consequently, the total current drawn is 2.5mA which

results in a lifetime of the battery of about 9 hours. Experienced people in the field (such as Dr. Lucky) suggested that 9 hours is a decent lifetime and for the actual demo or performance, the user should replace the old battery with a new one. Finally, we are looking into using two batteries at the same time to reduce the amount of times batteries need to be replaced.

3.1.4 Power Supply Noise Reduction

Real power supplies can cause noise and false oscillations that can results in potential issues in system's performance. Bypassing and decoupling are two methods that electronic engineers usually apply to reduce power supply noise. In this section, we provide a brief explanation of each method.

The output impedance of all voltage regulators increases with frequency due to their finite bandwidth. This increase in impedance can be modeled as an inductor in series with the output. When an active load is connected, the time varying current creates a noise voltage across these inductors which can be reduced in only two ways: reduce the rate of change of the current (d_i / d_t) passing through the inductor, or reduce the inductance. Bypassing reduces the rate of change of the current through the inductor and is applied to reduce the noise current on power supply lines. [8]

Specifically, bypassing is the reduction of high frequency current flow in a high impedance path by shunting that path with a bypass, usually a capacitor. In bypassing, a secondary, high frequency low impedance path (a capacitor) is provided for the varying currents from the load that shares as little inductance as possible with the power supply leads. For successful bypassing, properly determining the flow of current from a load and supplying a return path that is not common with any other part of the circuit is really critical (see Figure 3). [8]

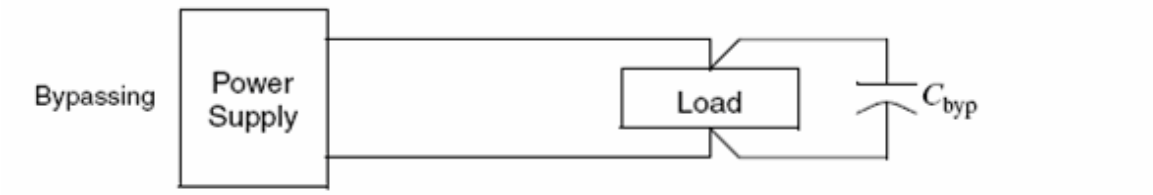

Figure 3: Bypassing Schematics

Further, the bypass path must have significantly lower impedance than the power supply leads at the frequency of interest. Using many small parallel capacitors than one large one is always recommended since the equivalent series inductance does not vary significantly with capacitance, and the parallel bypass paths achieved with the small capacitors results in a much lower total inductance.

As we mentioned earlier, decoupling is another method to reduce power supply noise. Decoupling is the isolation of two circuits on a common line, and it is used to prevent transmission of noise from one circuit to another. The decoupling network is usually a low pass filter and the isolation is rarely equal in both directions. In Figure 4 which shows a simple schematic for decoupling, a bypass capacitor, C_{byp}, is shown along with the decoupling circuit, L_{dec} and C_{dec}. This is because in practice bypassing is always used when decoupling. [8]

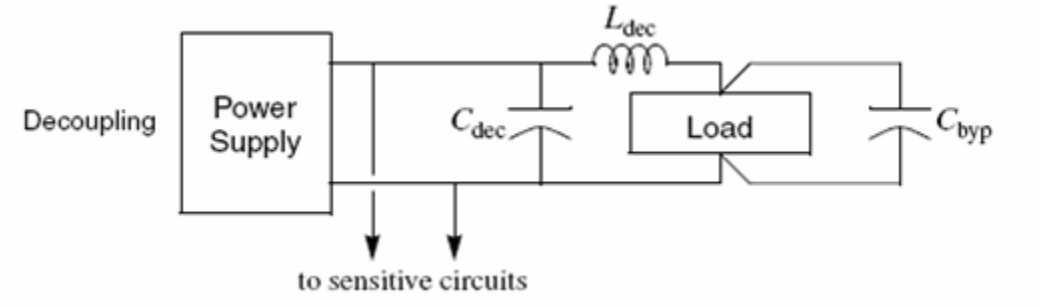

Figure 4: Decoupling Schematics

Decoupling decreases noise transmission in two ways as it is shown in Figure 5. First, since decoupling always consists of a high impedance element in series with the supply line, it assists the bypassing; assuring that the noise current will flow through the low impedance bypass element rather than the supply. Second, it acts as a low pass filter so that the high-frequency content of any current that does pass through the series element will be attenuated, thus, the regulator will be more likely able to react and keep the supply voltage stable.[8]

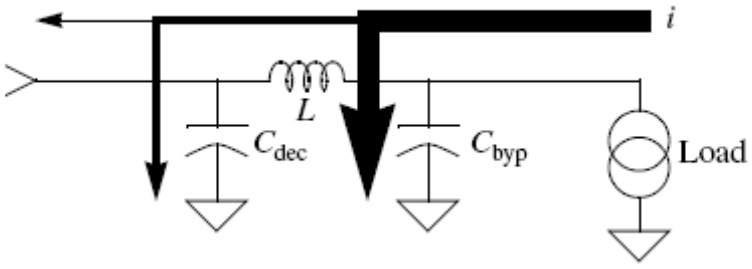

Figure 5: Current Path in Decoupling Circuit

3.2 Optical Markers Network

3.2.1 IR LED

The reason of choosing IR LED's over other types of LED's is that IR LED's have high intensity for the amount of power they require. The webcam is also very sensitive to IR radiation, and as we explained in section 3.3.1, we have modified the webcam so that it only detects IR radiation.

The choice of the IR LED had to meet the requirements specified in the *Functional Specification – Section 3.3: Safety Requirements and Section 4.1: Control Unit*. The power consumption, intensity, viewing angle, and availability are many of the consideration taken into account for selecting the IR LED's. We have carried out basic tests with three types of IR LED's, KIE7304, and KIE7305. We found that the KIE7305 has the most intensity for the same amount of current. The KIE7305 also seems to have a good viewing angle for our application. It is also available at the local electronics components store at an affordable price.

3.2.2 Boost Regulator

A voltage regulator transfers power from an unregulated input source V_I to a load at a prescribed regulated voltage V_O. Hence, its function is to maintain a stable, constant DC output voltage. In general there are two kinds of voltage regulators: Linear regulators, and switching regulators. In linear regulators the BJT operates in the forward active region, and ignoring the base current and current drawn by the control circuitry compared to load current I_O, there is power dissipation of:

$$P \cong (V_I - V_O) I_O .$$

It is this dissipation that limits the efficiency of a linear regulator.

However, switching regulators achieve higher efficiency by operating the transistor as a periodically commutated switch. In this case the BJT is either in cutoff, dissipating

$$P \cong V_{CE} I_C \cong (V_I - V_O) \times 0 = 0 ,$$

or in saturation dissipating

$$P \cong V_{SAT} I_C ,$$

which is generally small because the saturation voltage across switch is small. Thus a switched BJT dissipates much less power than a forward active BJT.

Indeed for a switch-mode operation we need a coil to provide a high frequency transfer of energy packets from V_I to V_O, and a smoothing capacitor to ensure a low output ripple. And since L and C control energy without dissipating any power, the combination of switches and low-loss reactive elements makes switching regulators more efficient than the linear regulators. [4]

Since our system is battery-powered, the high forward voltage of the IR LED's requires a high bias voltage which is an obvious issue for our system. Further, our automotive, distributed battery-powered system operates at a voltage that is derived from a variable bus voltage between all the LED's that are getting power from the microcontroller, thus, the operating voltage may fall somewhere in the middle of the bus voltage range. To resolve these issues, our system requires a DC/DC converter, i.e. a voltage regulator that can step up or step down, depending on the voltage present on the bus.

For our system hardware, we will make use of a LT1961, which is a monolithic, highly efficient switching regulator. The LT®1961 is a 1.25MHz monolithic boost switching regulator. A high efficiency 1.5A, 0.2W switch is included on the die together with all the control circuitry required to complete a high frequency, current-mode switching regulator. Current-mode control provides fast transient response and excellent loop stability. Its block diagram is shown in Figure 6: [6]

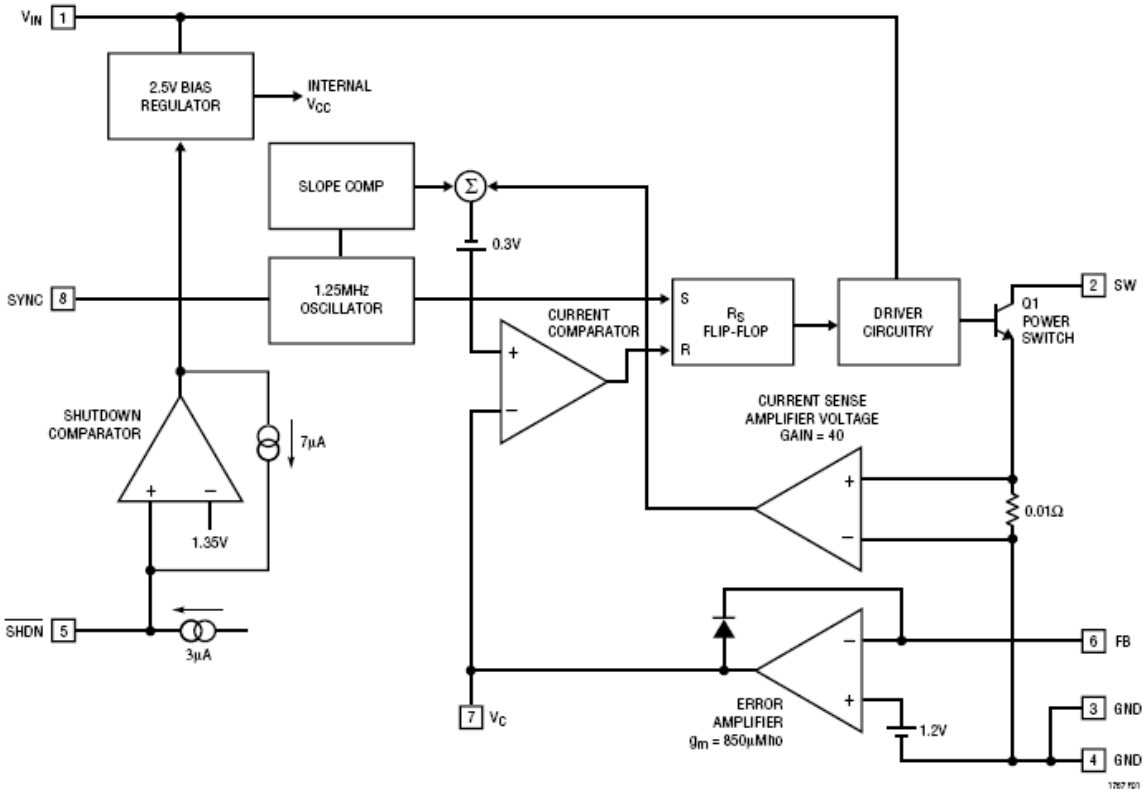

Figure 6: Block Diagram for LTI1961

Further, Figure 7 shows a 3V to 20V input, 5V output 3mm maximum height SEPIC using the LT1961, a 1.25MHz, current mode, monolithic, 1.5A peak switch current, boost converter. The tiny coupling capacitor used here is large enough to handle the RMS ripple current transferring between the primary and secondary sides of the circuit, and to maintain a voltage equal to the input voltage in order to provide good regulation and maximum output power. The current mode control topology of the LT1961 and the small 10mF ceramic output capacitor provide excellent transient response over the wide input voltage range. [7]

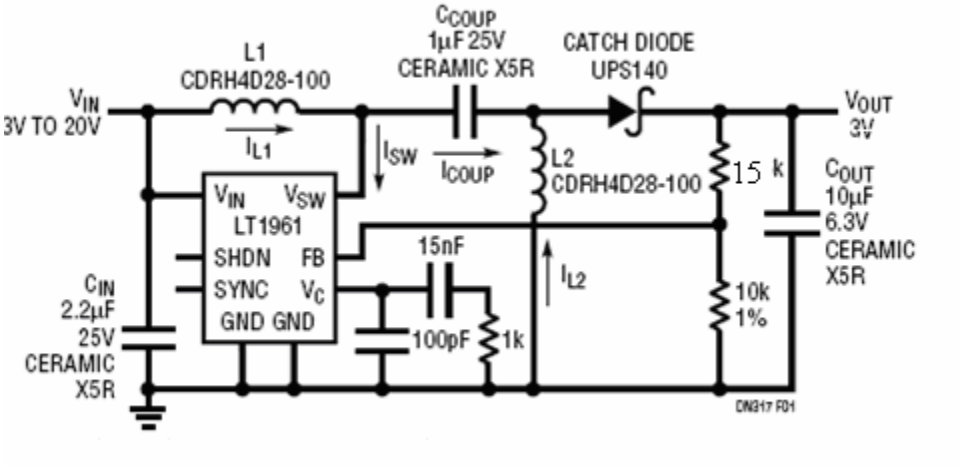

Figure 7: Boost Regulator

The feedback pin, FB shown in Figure 7 is used to set output voltage using an external voltage divider that generates 1.2V at the pin with the desired output voltage.
As it is shown in Figure 6, the internal 1.2V precision band-gap reference is internally connected between the substrate terminal and the inverting input of the high-gain comparator. Therefore, the output of the circuit is sensed through a resistor divider $(R1-R2)$ by non-inverting input of the comparator. Thus, the following relationship holds:

$$R_1 = \frac{R_2(V_{out} - 1.2)}{1.2 - R_2(0.2\mu A)}$$

The suggested resistance (R2) from FB to ground is 10k 1%. This reduces the contribution of FB input bias current to output voltage to less than 0.2% [6].Thus, for an output voltage of 3V, the value for $R1$ should be set to 15K.

Step-up regulators supply current to the output in pulses, and the rise and fall times of these pulses are very fast; thus the output capacitor is required to reduce the voltage ripple this causes. The LT1961 will operate with both ceramic and tantalum output capacitors. Ceramic capacitors are generally chosen for their small size, very low ESR (effective series resistance), and good high frequency operation, reducing output ripple

voltage. Their low ESR removes a useful zero in the loop frequency response, common to tantalum capacitors. To compensate for this, the VC loop compensation pole frequency must typically be reduced by a factor of 10. Typical ceramic output capacitors are in the 1µF to 10µF range. Since the absolute value of capacitance defines the pole frequency of the output stage, an X7R or X5R type ceramic, which have good temperature stability, is recommended by the data sheet of LT1961 [6]. For our system, we are using a 10µF X5R type ceramic as the output capacitor.

Unlike the output capacitor, RMS ripple current in the input capacitor is normally low enough that ripple current rating is not an issue. At higher switching frequency, the energy storage requirement of the input capacitor is reduced so according to LT1961's data sheet values in the range of 1µF to 4.7µF are suitable for most applications [6]. Further, since the absolute value of capacitance is less important and has no significant effect on loop stability, we decided to use 1µF X5R ceramic as the input capacitor.

The suggested catch diode (D1) by is a UPS140 by the Linear technology design guide [7]. It is rated at 1A average forward current and 20V/30V reverse voltage. Typical forward voltage is 0.5V at 1A. The diode conducts current only during switch off time, peak reverse voltage is equal to regulator output voltage, and average forward current in normal operation is equal to output current.

When choosing an inductor, there are 2 conditions that limit the minimum inductance; required output current, and avoidance of sub harmonic oscillation. The recommended minimum inductance is [6]:

$$L_{\min} = \frac{(V_{in})^2 (V_{out} - V_{in})}{0.4(V_{out})^2 (I_{out})(f)}$$

For our system, we are using a CDRH 4D28-100 inductor.

According to Linear Technology design guide [7], one alternative to a transformer-based topology is to use two low profile inductors and a SEPIC coupling capacitor which transfers the energy between the two inductors much like the core of a transformer. The coupling capacitor provides a low impedance path for the inductor currents to pass either from the input, primary, inductor through the catch diode and to the output, or from the output, secondary, inductor back through the switch to ground. Both inductors act continuously and independently, making their selection easier than selecting the transformer for a flyback or a typical SEPIC circuit.

3.2.3 Power Protection Circuit

Generally any electronic network or system requires some sorts of power protection circuitries in order to protect the system from power irregularities. The control unit of our system along with the optical sensors is designed to be powered using the Lithium batteries. Even though batteries are considered to be one of the simplest and most basic

sources of power to be utilized in electronics, but the polarity of the batteries can be a source of a risk to our system.

In order to protect the circuit against the potential incorrect polarity connection of the battery to our system, the Power protection circuitry is utilized. The power protection circuitry is a simple circuit that protects a sensitive electronic circuit from an accidental connection of a battery with a reversed polarity. The schematic of the power protection circuitry is depicted in Figure 8. [1]

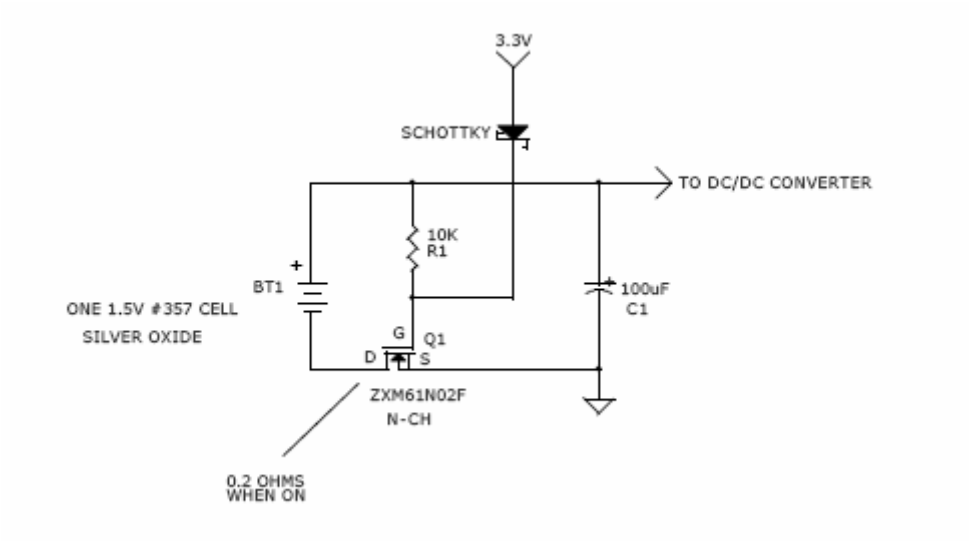

Figure 8: Power Protection Circuitry

The N-Channel FET connects the electronic device to the battery only when the polarity is correct. The circuit depicted in Figure 8 was designed for a device powered from a single 1.5 volts button cell battery. However, the circuit will operate with higher voltages as well. This circuitry is to be utilized in our control unit to protect the microcontroller, the decoder, and also all the optical sensors from reversed polarity connection of the corresponding batteries. The last point to note is that, we have decided to use the SCHOTTKY diode in our design since it only has a smaller voltage drop and it is more efficient compared to its alternatives.

3.3 Optical Capture Unit

3.3.1 USB Webcam Circuitry

The webcam that we chose to utilize for our system is Logitech QuickCam 6.0. We chose this webcam because one of the group members had it available, and it meets the requirements stated in the Functional Specification of the system, R[44] to R[51].

The camera should be able to capture the frames fast enough so that the system would not loose any movement of any of the joints. In this way, the simulation of the object's

motion will be smooth and the motion captured will be accurate and close to the actual motion of the object. The frequency of the Logitech QuickCam 6.0 is 100Hz which is much more than the possible speed of a dancer. Further, the frame rate of the webcam is more than 30 frames per second; thus, it has the ability to capture required number of frames per second and transfer it to the analytical software for further analysis [9].

The other critical requirement of the webcam is its viewing angle. The viewing angle of the webcam should be large enough so that at any moment none of the LED's is outside the capturing range. Logitech QuickCam 6.0 webcam has a minimum viewing angle of 90° which is sufficient for our application [9].

Further, the webcam has a USB connection and is compatible with Platform SDK. Logitech QuickCam 6.0 has 1.1 or 2.0 USB port available and its USB cable has a length of 6 ft. which is more than the minimum required length specified in the Functional Specification.

Logitech QuickCam 6.0 has a weight of 0.4kg, and it has dimensions (W × D × H) of 8cm × 6cm × 8cm which are compatible with the physical conditions stated in the Functional Specification of the system [9].

Further, as it is explained in section 4.3, the algorithm used in the analytical software requires that the webcam blocks completely any visible light and detects the infrared light. Thus, the resulting frames captured by the webcam shall have a completely black background and the LED that is on at the captured time being shown as a white dot in the frame. Figure 9 shows such a frame:

Figure 9: Frames Captured Before and After Blocking the Visible Light

The regular webcams have an IR filter which blocks the infrared light results in capturing the color images. Logitech QuickCam 6.0 uses the light sensors of the type charge-coupled detector, CCD. This type of the sensors are more sensitive to infrared light near

the visible spectrum than they are to visible light, as shown in Figure 10. So, for a CCD camera to match what a human sees, all of the infrared light (wavelengths longer than 700 nanometers) must be cut out with a filter. By removing this IR-cutout filter, the webcam will no longer block the IR light which is required for our application.

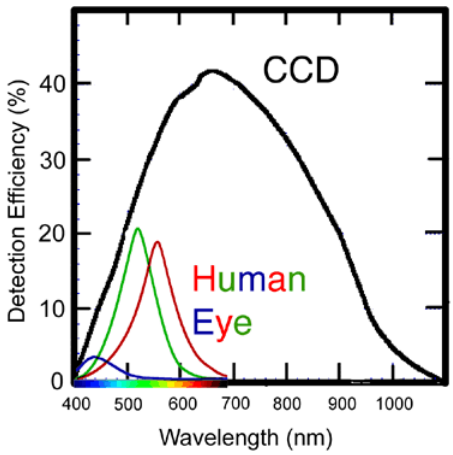

Figure 10: Light Detection Efficiency of a CCD

Figure 10 shows the light detection efficiency of a CCD sensor versus the three types of cone cells in the human eye. CCD detectors are more efficient than cones, and are sensitive to infrared light with wavelengths out to ~1000 nanometers nm [10].

To remove the IR filter the following steps should be carried out:

1. Unscrew the body of the webcam and open the case.
2. Remove the circuit board from the case.
3. Remove the circuit board from the case which had the lens attached to it.
4. Once extracted the lens assembly, look for the IR-blocking filter. It is a greenish/bluish glass and/or a surface that reflects pink or red. Most commonly, the IR-block filter sits between the lens assembly and the CCD / CMOS light detector (the flat black shiny plate on the green circuit board).
5. Remove the filter as shown in Figure 11.

Figure 11: Removing the IR Filter

Thus, by removing the IR filter the infrared lights will no longer be blocked. Further, to have a black background we need to block the visible light. Film negatives of black scenes act as filters permitting only near-infrared; therefore, by putting pieces of exposed film negatives in front of the webcam's lens all the visible lights will be blocked and only infrared light can pass. Thus, the image that we get is a totally black background with a dot of infrared lights which indicates the position of the on IR LED in the scene.

4 System Software

The software of the system can be divided in four main parts: optical markers control unit, optical capture control unit, image processing, and animation. Mainly, the first unit is responsible for choosing the specific LED to be turned on ad off, and also it is responsible for controlling the timing of powering the LED's. The optical capture control unit is in charge of automating the capturing process of the webcam, adjusting the frame rates and saving the captured frames as files. The image processing unit is responsible for analyzing the captured frames and thus provides an algorithm which for recording the coordinates of each joint in a two dimensional plane. The coordinates of the joints will be sent to the animation unit. The animation unit is responsible for generating the motion on the screen of the computer.

In the following sections we will explain about the design specifications of each of these units.

4.1 Optical Markers Control Unit

In sections 3.1.1 we have discussed about the hardware of the microcontroller. In this section we will explain about the programming specifications of each.

The software code to run the PIC 16F84A microcontroller is written is assembly language given the time critical operations. The flow chart of the code is attached at the end of this document in Figure 24. A switch debounce is implemented in the software through the user of a delay before the start of the code. This delay makes sure that the switch has been turned on for a while and the microcontroller is ready to execute the code. The four control signals of the decoder are mapped to PORTA(0..3). As discussed in the control unit hardware design, the WDT timer is utilized to wake-up the microcontroller from the low power sleep mode. Count is the variable declared to keep track of which LED is turned on. Once all of the LED's are turned on, count is reset to start a new cycle. The duration of each LED turned on is controlled through the software and can be modified as the project progresses.

4.2 Optical Capture Control Unit

The optical capture control unit is responsible of capturing the image of an object to be tracked and send the information it retained to the image processing unit. The capture control unit is a direct interface between the computer mounted software system and the IR hardware system. The main task's of optical capture control unit is listed in Table 1.

Name of Task	Description
Adjusting Frame Rate	The unit automatically adjust the frame rate in order to optimize with the hardware IR sensor duration time.
Capturing Image	The unit captures using web camera at the designated t ime rate.
Creating Matrix	The unit converts the raw image data received from the web camera into human-familiar image matrix.

Table 1: List of tasks of optical capture control unit

In this section, we start with development environment followed by detailed description of each task listed in Table 1.

4.2.1 Development Environment

The design of the optical capture control unit was referenced the design of DXCapture system [11]. The unit is programmed with Visual C++ language with Microsoft Foundation Class Library (MFC Library.) The reason that the software team chose such language is because MFC Application is the most realistic way to develop graphical programming in Windows system. The software was built on Microsoft Windows XP SP2 with Microsoft Visual Studio 2005. The web camera that the unit is communicating must support USB 1.1 or 2.0 as it is stated in the Functional Specification R[53]. According to the research of software team, computer web cameras that potential users might purchase in the market are mostly supports USB interface. In order to give flexibility of choosing web cameras for potential users, the software team designed the unit so that it communicates with the web camera through USB 1.1 or 2.0. The unit uses two software development kits (SDK); (1) Microsoft DirectX SDK 2007 February and (2) Microsoft Platform SDK for Windows Server 2003 R2. Table 2 describes the environment on where the program is developed.

Category	Chosen Environment
Operating System	Microsoft Windows XP
Programming Lang uage	Visual C++ (MFC)
Program Tool Kit	Microsoft Visual Studio 2005
SDK	1. Microsoft DirectX SDK 2007 February 2. Microsoft Platform SDK for Windows Server 2003 R2
Supported Interface	USB 1.1 or 2.0

Table 2: Development Environment

4.2.2 Adjusting Frame Rate

Given the information that the fourteen sensors mounted on the doll are turned on one sensor at a time with designated time duration, the optical control unit adjust the frame

rate to optimum value. Figure 25 in Appendix C illustrates the algorithm of adjusting the flow rate.

When the software is tuned on by the user, the software runs the adjusting frame rate process in order to initialize the system. Since the software does not have the direct communication with the IR hardware sensor, the software system must automatically optimize its frame rate so that the frame rate is synchronized with that of the IR hardware. The system first sets the frame rate, R_{frame}, so that it is must larger than that of the IR hardware. The default value of the frame rate is,

$$R_{frame} = 25 \quad \text{Frames/sec}$$

Then the system checks if there is any IR light captured from the web camera. If there is no IR light captures, then the system checks until it finds one.

Once the system detects there is an IR light point shown in the frame captured by the camera, it calculates the coordinate of that point. Since the frame rate is fast enough, we are expecting more than one frame with same coordinates will be recognized by the system. The following example shows the possible outcome in the process.

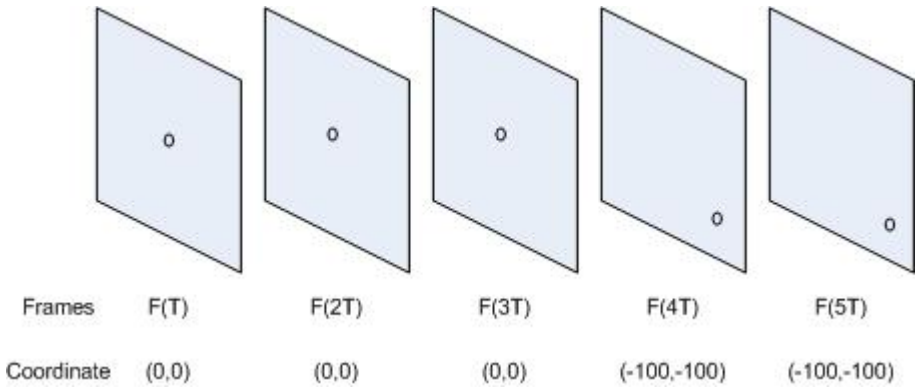

Figure 12: Example of Adjusting Frame Rate

The system detects the coordinate of an IR dot using Image Processing Unit. Then it determines whether the coordinate of the detected dots have been moved. The algorithm used to return the coordinate of the dot will be explained in detail in the next section. Then the optical capture control unit calculates how many frames were captured with same IR dot coordinates. The system then calculates the optimal frame rates using the formula below.

$$R_{frame,opt} = \frac{R_{frame}}{n}$$

where *n* is the number of frames with same IR dot coordinates.

Then the system captures images using web camera with new frame rate and checks if there is not consecutive frames with same IR dot coordinates. If this condition is satisfied, then the system starts capturing images as well as image processing

4.2.3 Capturing Image

One of the important tasks of the optical capture control unit is to capture images using web camera automatically. As it is mentioned in the introduction of the section 4.2, the capturing image module is referenced by [11]. The unit simultaneously captures images from a web camera using DirectShow SDK which is included in MS Platform SDK for Windows Server 2003 R2.

4.2.3.1 Init(int iDeviceID, HWND hWnd, int iWidth, int iHeight)

This function initialized the program to make it ready to connect to the web camera. Since there can be more than one camera connected to the computer, the program automatically finds a default camera and returns its index number as iDeviceID. hWnd is the handler of the display window and iWidth and iHeight are with and height of the resolution respectively. The default resolution is 320 pixels by 240 pixels.

4.2.3.2 DWORD GetFrame(BYTE ** pFrame)

Calling this function will allow the program to capture an instance image from the web camera and put it into a buffer. The return value will be the size of the buffer which refers to the buffer pointer.

4.2.3.3 DWORD ImageCapture(LPSTSTR szFile)

This is a function which calls GetFrame() function and convert the bits received from the web camera into the raw RGB images. The returned values will contain red, green and blue components of the image. However, the returned value is just a series of 24 bit numbers which must later be converted into a 320 by 240 matrix. The algorithm that finds the coordinate of an IR point in the frame will later be discussed in the next section.

Figure 26 in Appendix C illustrates the algorithm for capturing images.

4.2.4 Creating Matrix

The last important task of the optical capture control unit is to convert the series of unorganized image bytes into a three different matrices which are red, green and blue components respectively. The size of the array of the image bytes is

$$3 \times 320 \times 240 = 230400$$

The software team found out the size of the matrix testing various possible frame sizes. The conversion of the array of bytes into three different matrices is as follows.

```
for (i = 0; i < 230400; i++)
{
      y = (int)(i/960);
      x = (int)((i - (y * 960)) / 3);
      //x and y coordinate conversion.
      if((i % 3) == 0)
            r_matrix[x][y] = *(pFrame + i);

      else if ((i % 3) == 1)
            g_matrix[x][y] = *(pFrame + i);
      else if ((i % 3) == 2)
            b_matrix[x][y] = *(pFrame + i);
}
```

In this algorithm, r_matrix[320][240] represents red component of RGB image and each values in the matrix is the 24-bit integer of red components in that pixel position. pFrame represents the pointer for the first member of the image array.

4.3 Image Processing Unit

4.3.1 Background Knowledge

The image processing unit is responsible of finding a true coordiate of the IR point in a captured frame. As it was discussed in the system hardware section, two layers of flim is allocated on the web camera to absorb all human visible light but infrared light. Therefore, when the web camera captures an image, the frame will be in black background with small cluster of white IR light. The following picture is an example of frame image of captured IR light.

Figure 13: Example of Original Image and Visible Light Filtered Image

The image on the left is the original image capturing an intrared light and the one of the right is the visible light filtered out image. The reason showing this example is to let you know that camera captures the infrared light as an white color light source. In RGB-format, the white color is represented by having maximum value for all red, green and blue components. In other words a true white color pixel in 24-bit RGB will have the following values.

$$[R,G,B] = [255, 255, 255]$$

Therefore, in order to find the coordate of the IR point, we are free to use either red, green or blue matrices of a captured image. However, as you can see in the Figure 13, the IR light shows very white color in the middle with blueish dimmer color surrounding it. We use the red matrix to make sure that we have a vivid circle IR image in the frame.

4.3.2 Algorithm

Keeping the backgound knowledge in mind, we are now presenting the algorithm to find the coordinate of the IR image captured in a frame. The coordinate will be given in (x,y) which is the pixel cooridination in (320, 240) frame. When we anaylze the IR filter point using MATLAB® we obtain a similar number pattern as in Figure 14. The figure is directly captured from MATLAB® windows and this represents the white point in Figure 13.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	26	45	49	60	29	9	0
0	0	44	68	80	118	94	72	28	6
0	56	70	202	255	224	157	121	43	36
20	58	236	255	255	255	245	150	93	72
45	111	255	254	255	255	245	193	117	77
29	177	255	255	255	255	249	249	83	70
50	135	255	255	255	255	249	197	72	113
45	124	254	254	255	255	228	147	72	22
20	104	134	237	255	224	152	118	35	65
29	0	54	107	132	146	108	73	39	10
43	36	26	58	81	62	41	29	27	0
0	0	10	63	11	1	10	7	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 14: Example of Numerical Analysis of the IR point Captured

When the software analyzes the red matrix, it gives a threshold value to distinguish strong white point from dimmer points. This is illustrated in Figure 14, with a threshold value of 80 and every critical values greater than this threshold value is included in the

red box. In the software, we actually mark these critical values as one and rest of them zero for convenience. As we can easily see, the red box can be recognized as a circle.

In order to successfully calculate the coordinate system, the software scans the matrix from the coordinate (0,0) to (320,240) until it finds a '1'; the critical point. When it finds the first critical value, it scans horizontally to see where the critical points end. It is just like making a horizontal range of critical values. In our example illustrated in Figure 14, the first range will be as follows.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	26	45	49	60	29	9	0
0	0	44	68	80	118	94	72	28	6
0	56	70	202	255	224	157	121	43	36
20	58	236	255	255	255	245	150	93	72
45	111	255	254	255	255	245	193	117	77
29	177	255	255	255	255	249	249	83	70
50	135	255	255	255	255	249	197	72	113
45	124	254	254	255	255	228	147	72	22
20	104	134	237	255	224	152	118	35	65
29	0	54	107	132	146	108	73	39	10
43	36	26	58	81	62	41	29	27	0
0	0	10	63	11	1	10	7	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 15: Numerical Analysis with the First Horizontal Range

Then we calculate an average of the x-coordinate of the starting index of the range and the ending index of the range. In our example, the starting index will be the x-coordinate of the value *"80"* and the ending index will be that of *"94"*. Then with that average x-coordinate, we search for a vertical range just like we search for the horizontal range. If we successfully find the vertical range, which will be from "118" through 146" in our case, then we average the y-coordinate and start to search for another horizontal range with that y-coordinate. We continue this searching and averaging process four times to successfully find the coordinate of the IR point. In our example, the searching will give us point as in Figure 16 below.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	26	45	49	60	29	9	0
0	0	44	68	80	118	94	72	28	6
0	56	70	202	255	224	157	121	43	36
20	58	236	255	255	255	245	150	93	72
45	111	255	254	255	255	245	193	117	77
29	177	255	255	255	255	249	249	83	70
50	135	255	255	255	255	249	197	72	113
45	124	254	254	255	255	228	147	72	22
20	104	134	237	255	224	152	118	35	65
29	0	54	107	132	146	108	73	39	10
43	36	26	58	81	62	41	29	27	0
0	0	10	63	11	1	10	7	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 16: Approximated Coordinate of the IR Point

In Figure 15, the return coordinate of the IR point will be the coordinate of the value that is marked with a blue box. This process of finding the middle point will give an approximated point of the IR position in the frame. Figure 27 in Appendix C illustrates the algorithm used to find the coordinate of IR lights.

4.3.3 Challenge in Development

There are few challenging tasks to be concerned while developing the image processing unit to detect the correct coordinate of the IR point. We named it as distorted point which is a distortion in IR light due to rotation of IR light bulb. Additionally, there is another problem of having virtual point which is a reflected IR light captured by the camera. We show example of each of these two problems in the following diagram.

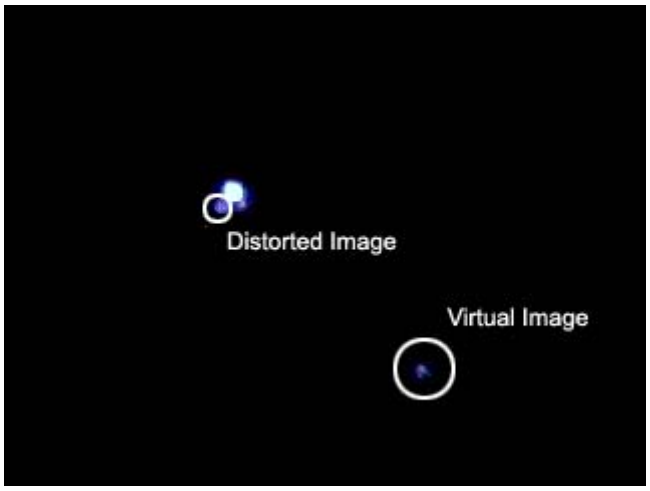

Figure 17: Distorted Image and Virtual Image

In order to be free of these two problems, we can develop an algorithm which can intellectually remove any faulty points other than the true image in the frame. The algorithm is based on the experimental result we found that IR source gives a certain horizontal and vertical range of critical points (as it was discussed in the section 4.3.2) when the light source is placed at a fixed distance from the web camera. Therefore any other images with significantly smaller width and height values compare to the experimental values can be removed from decision making process.

4.4 Animation Unit

In this section we will explain about the design of the animation units in two design levels: high level and low level. In high level section we will explain about the user interface. In low level design we will explain about the logic and implementation of the program.

4.4.1 High Level Design

The user interface of the system consists of three sub-windows,

- The Settings Box (properties)
- The Animation Window
- The Icons Window

Figure 21 in Appendix A shows the user interface of the system. Please note that this window is subject to further modifications. However, it illustrates the layout the user interface and it exemplifies an overview of the user options.

4.4.1.1 Settings Box

The settings box is the right side of the user window on the user interface window in the run time. It basically consists of the following items:

a. Properties: The properties item enables the user to modify the segments resolution of the object model components. In addition it enables the modification of the scale factor applying to the animation window's content. And lastly the object model representation can be changed to Wire-frame mode as well.

b. Lights: This feature enables the user to alter the lighting settings of the simulation environment. Mainly the lighting feature has been divided to two components, the light setting of the object model and the light setting of the background working area. The intensity of each component can be modified independently and the lightings can be disabled as well.

c. Options: The option feature is to enable and disable the current existing items on the animation window. Option feature can enable/disable draw object, Draw axes and Draw text on the simulation window.

d. Text: Using the Text feature, the text being printed on the simulation window can be altered to the provided text choices accordingly.

e. Icon controls: Finally there are four control buttons designed to control the content of icons window. The icons can be disabled/ enabled. The icons can be shown/ hidden. And lastly a Quit button is devised that enables the user to exit the application immediately.

4.4.1.2 Animation Window

The animation window represents the working area under control with the respective preset dimensions. It also provides the 3D representation of the moving object. Note that since our system is a 2D tracking motion system, the 3D model is projected on a 2D plane (table) and the movements are all analyzed just in 2D frame. And lastly the pre-selected text from Settings feature is displayed on the animation window accordingly.

4.4.1.3 Icons window

The last sub-window contains some useful icons which enable the user to perform a set of rotational and translational operations on the object model and the working frame respectively. The designed icons along with their functionalities are as follow. The Object, Sphere and Cube arc balls enable the user to rotate and change the orientations of the entire animation content, the object model, and the working table respectively. The Blue Light arc ball enables the user to change the lighting of the animation window. And finally the translational control icons enable the user to translate the animation content in X, Y, Z directions and also in the XY plane accordingly.

4.4.2 Low Level Design

In this section we will explain about the low level design of the animation unit. First we provide an overview of the logic of the program, then we outline the sequential flow of the diagram. Finally, in the last subsection, we will explain about the implementation of the program.

4.4.2.1 Logic of the program

Having highlighted the main features of the user interface design, the high level software design implementation corresponding to each components of user interface will be discussed in this section accordingly.

The program is basically all integrated into a single main class called, ***main.cpp***. The overall logic of the program is similar to any typical OpenGL and code programming; the main function of the code, is responsible for initializing the glut, window, calling the glut display, reshape and idle functions, handling the exit command, initializing the scene and finally entering the famous infinite loop, glutMainLoop.

There are totally three costume-made functions in the class which are described below.

1. void ***drawCylinder***(…): This function is responsible for drawing a cylinder, with two discs attached on its top and button faces. A quadric object, top and bottom radius size, height, slices and stacks values are passed in as of the required arguments for this function.

2. void ***drawfoot***(): This function is responsible for drawing the last link of our model representing the foot of dancing doll. This task is accomplished utilizing the *gltranslat(), glRotate(), glutSolidSphere(…)* and *glutSolidCube(…)* OpenGL functions.

3. void ***drawhand***(): This function is responsible for drawing a link of the model representing the hand of dancing doll. This task is similarly accomplished utilizing the *gltranslat(), glRotate(), glutSolidSphere(…)* and *glutSolidCube(…)* OpenGL functions.

4.4.2.2 Graphics Update

The OpenGL graphics scene is to be updated continuously in order to be able to track the motion of the object accordingly. The OpenGL animation software is able to simulate the movements of the object based on the coordinates update being input from the image processing software. The image processing software outputs the new coordinates of markers having placed on the object for continuous image captures.

Having the X and Y coordinates of each marker, the corresponding link of the model object can be translated and/or rotated accordingly. In order to be able to accomplish this task the following issues should be considered.

1. In the animation implementation, is has been assumed that the hardware system is perfectly synchronized with the image processing unit. In this case, the animation software would also be synchronized with the hardware system because the animation and image processing units both will be integrated as a Visual C++ package software eventually.
2. Based on the discussion of the sensors' identification algorithms stated in the software implementation section, the animation unit will also be informed of the first sensor that will be captured and analyzed. Having identified the first sensor, by implementing a simple counter, the program will keep track of the sensors in sequence and therefore will assign the received X and Y coordinates (from image processing unit) to the appropriate markers respectively.
3. The propagation through the linkages of any manipulator (for our case, the limbs of body model) can be accomplished in two different ways with respect to assigning the coordinate systems to the linkages.
    a. The Absolute coordinate system: In the absolute coordinate system method, a preset point is chosen as the reference point or the origin. All the coordinate systems assigned to the links will be expressed with respect to this preset origin. The advantage of this method of implementation is

the ease of use and follow in the propagation of the links of our model. The important condition at which this method would function properly in our application, is the fact that the frame rate of the USB webcam is high enough such that the frames are being captured and analyzed fast enough such that the human eye would not be able to detect the discrete movements of the linkages, all with respect to the predefined origin of coordinate.

b. The Relative coordinate system: In the relative coordinate system method, there is no preset origin selected as the reference point of links' coordinate frames. Each link's coordinate frame is referenced to its preceding link's coordinate. Even though this method is more tedious and complicated than the Absolute coordinate system, but this is the more efficient method of propagating through the linkages of our object model; there would be no necessity to keep track of coordinates of newly added links with respect to a predefined origin and each new link will be referenced to its previous one by a simple transformation matrix multiplication. (Please refer to the following algorithm). It is also easier to implement the relative coordinates system in our animation software compared to the Absolute coordinates system because there would be no need to separate the translational and/or rotational commands being assigned to different linkages as it is the case for the Absolute coordinate systems.

Based on fore mentioned reasoning, we have decided to use the Relative coordinate system method to implement our motion tracking algorithm within the animation program. The following algorithm outlines the steps undertaken to accomplish our motion tracking process.

1. Store the previous X and Y coordinates.
2. Record the new X and Y coordinates received from the image processing unit. (Based on earlier discussion, the animation program is able to identify the markers correctly and will therefore assign each pair of X and Y coordinates to the appropriate link (or the corresponding marker) respectively.
3. Calculate the X and Y displacements.

$$\Delta X = X_{current} - X_{previous}$$
$$\Delta Y = Y_{current} - Y_{previous}$$
$$\theta = \tan^{-1}(\Delta Y / \Delta X)$$

4. Construct the transformation matrix.

$$M = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & \Delta X \\ \sin\theta & \cos\theta & 0 & \Delta Y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5. Pre-multiply the transformation matrix from step 4 by the marker (IR LED) this is under control [5].
6. Repeat steps 1-4 for as long as the application hasn't exited. (infinite loop)

4.4.2.3 Sequential flow of the program

The overall flow of the program can be highlighted in following sequential steps.

1. Initializing GLUT.
2. Initializing GLUT window size.
3. Setting glut call back functions

Setting Glut display function

3.1.1. Setting all the camera parameters.
3.1.2. Drawing graphics scene.
3.1.2.1. Calculating the rotation angle and displacement of each model joint based on input data. (Please refer to Graphics update section)
3.1.2.2. Drawing all the parts of the model.
3.1.2.3. Drawing the appropriate text according to user's selection.
3.2. Setting the Glut Reshape function (Call back for reshaping the window).
3.3. Setting the Glut idle function (Call back for idle state).
4. Handling the exit event.
5. Setting the lighting parameters of the scene. (Set up OpenGL lights)
7. Setting up the GUI elements for the side window.
7.1. Setting up the controls for object parameters.
7.2. Setting up the controls for lights.
7.3. Setting up the controls for Show/Hide the objects.
8. Setting up the GUI elements for the Bottom window.
9. Entering the infinite loop, glutMainLoop. (Returns to step 3.1, glutDisplay and runs for ever until user exits the program).

The overall structure of the program can be best described by depicting the corresponding flowchart. The following flowchart explains the different stages and steps undertaken with their corresponding order and hierarchy.

A flow chart of the program is provided in Appendix C, Figure 28 .

4.4.2.4 Implementation

For implementation of the program, we have used all the common functions of typical OpenGL sample programs, but in addition developed three customized functions

mentioned earlier in the logic section of the report. The main functions along with their brief descriptions utilized in our software development, are as follows.

void control_cb (…) This is a GLUI control call back function which is responsible for handling the pressing of any buttons.

void myGlutKeyboard(unsigned char Key, int x, int y) This function is responsible for handling the events related to keyboard input.

void myGlutIdle(void) This function is responsible for operation of the application when the program is in idle mode.

void myGlutMotion(int x, int y) This function is responsible for proper handling of window when the user moves the window frame.

void myGlutReshape(int x, int y) This function is responsible for proper handling of window resizing event.

void myGlutDisplay(void) This function is major OpenGL function which is responsible for drawing the screen. This function will be continuously called ("infinite loop") until the user closes the program or the window. This is by far the most important function in the animation software development design.

void __cdecl exitHandler() This function handler is automatically called when the program is exiting.

int main(int argc, char argv[])* This function is the main function of our program. The application basically starts execution from this function. Within this function all the fore mentioned functions within the class in being called based on the flow of our implementation.

5 Test Plan

In this section we will outline our test plans for the hardware and software of the system. Some of these test plans are designed to check the functionality of the parts and some are designed to verify the successful integration of the different subunits of the system.

5.1 Hardware Test Plan

5.1.1 Optical Marker's Threshold Detector Circuitry

Each IR LED should turn on within the time period that the microcontroller specifies and it should go to sleep the rest of the time in each cycle. If an IR LED fails to turn on during its own on-time then the analytical software will not be able to track the LED's accurately which results in an inaccurate simulation of the motion.

This test is to verify that each IR LED gets on during the required on-time. To perform the test we will utilize a level or threshold detector circuit to monitor the voltage across each LED. As the voltage across each LED rises above 5V, the detector output is used to generate an interrupt to the microcontroller so that programmer is alerted that the LED is on.

As shown in the Figure 18 shows, the detector has three components: a voltage reference, V_{REF}, to establish a stable threshold, a voltage divider R_1 and R_2 to scale the input v_I, and a comparator. The comparator trips whenever v_I is equal to V_T which is equal to:

$$V_T = (1 + R_2 / R_1) V_{REF} \quad .$$

For $v_I < V_T$, the comparator is off and so no interrupt would be generate. For $v_I > V_T$, the comparator saturates and generates an interrupt to the microcontroller, thus providing an indication of when the IR LED is on [4].

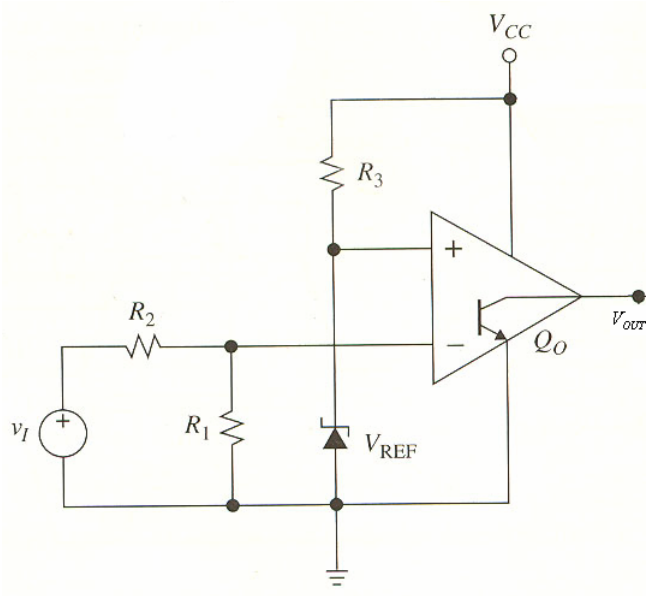

Figure 18: Level Detector Circuit

Note that R_3 is responsible for biasing the diode.

5.1.2 PIC6F84A Functionality Test Circuitry

As mentioned earlier in the system hardware section, the PIC16F84A is chosen to serve as the microcontroller of our system. In order to verify the functionality of the microcontroller a series of test plans were devised. The test plans require hardwire circuitry in which the microcontroller is tested; thereafter the microcontroller is programmed to perform a specific task. Figure 19 depicts the test circuitry utilized for in this test plan. [2]

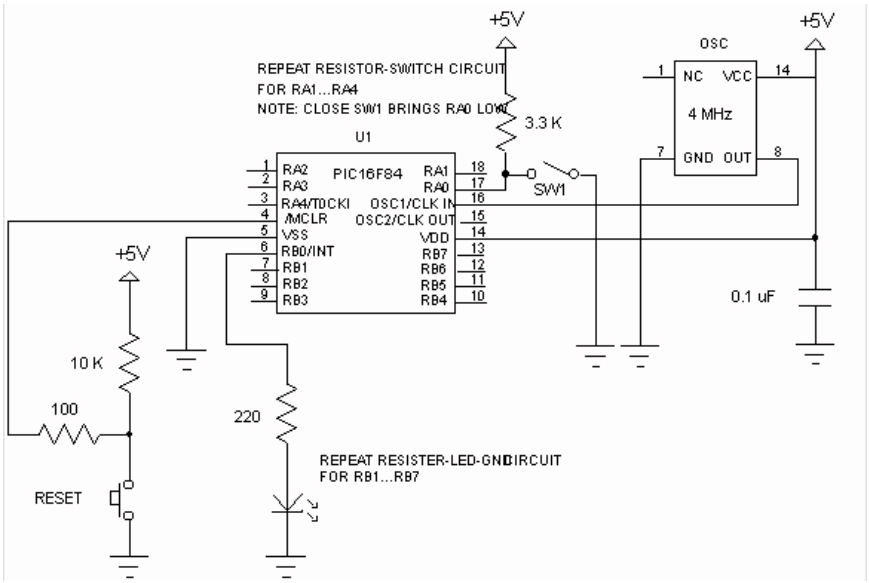

Figure 19: PIC16F84A Functionality Test Circuitry

This circuit is set up to test and display basic PIC functions. PORTB on the PIC (Pins 6-13) is used as an output. LED's are connected to all 8 of the PORTB lines, and will light up when the line is set to logic high, or '1'. Port A (Pins 17, 18, 1, 2 and 3) is used as an input. Its lines are connected to dip switches, which will set the line to logic high when the switch is in the 'On' position.

Having set up the hardware circuitry, now the microcontroller can be programmed to perform the required task and thereafter its behavior can be compared to the expected result to ensure the proper functionality of the microcontroller. The following code has been developed to assign pins of the microcontroller as output and to control the LED's.

```
        list    p=16F84
        radix   hex

;-----------------------------------------------------------------------
;       cpu equates (memory map)
myPortB equ     0x06            ; (p. 10 defines port address)
;-----------------------------------------------------------------------

        org     0x000

start   movlw   0x00            ; load W with 0x00 make port B output
        tris    myPortB         ; copy W tristate, port B outputs

        movlw   b'10101010'     ; load W with bit pattern
        movwf   myPortB         ; load myPortB with contents of W

circle  goto    circle  ; done

        end
```

As intended, the pins 0, 2, 4 and 6 of PORTB was set to logic low and pins 1, 3, 5 and 7 were set to logic high, all operating as output. Having connected the LED's to the corresponding pins, the LED's on the logic high pins of PORTB were all turned on while the ones on logic low remained all off.

This simple test plan verified some basic functionality of PIC16F84A microcontroller and the test result was in fact perfectly in accordance with our expected goal.

5.1.3 PIC6F84 Oscillator Design Circuitry

The microcontroller PIC16F84A utilized in our system design can be operated in four different oscillator modes. The microcontroller can be programmed by two configuration bits to select the four different modes of the oscillator. The four oscillator modes are as follows; the LP: Low Power Crystal, XT: Crystal/Resonator, HS: High Speed Crystal/Resonator, and RC: Resistor/Capacitor.

Due to the better performance and ease of use, the XT oscillator mode has been chosen for our system design. Figure 20 depicts the configuration at which the crystal or ceramic resonator is connected to the OSC1/CLKIN and OSC2/CLKOUT pins of the PIC microcontroller to establish oscillation. [3]

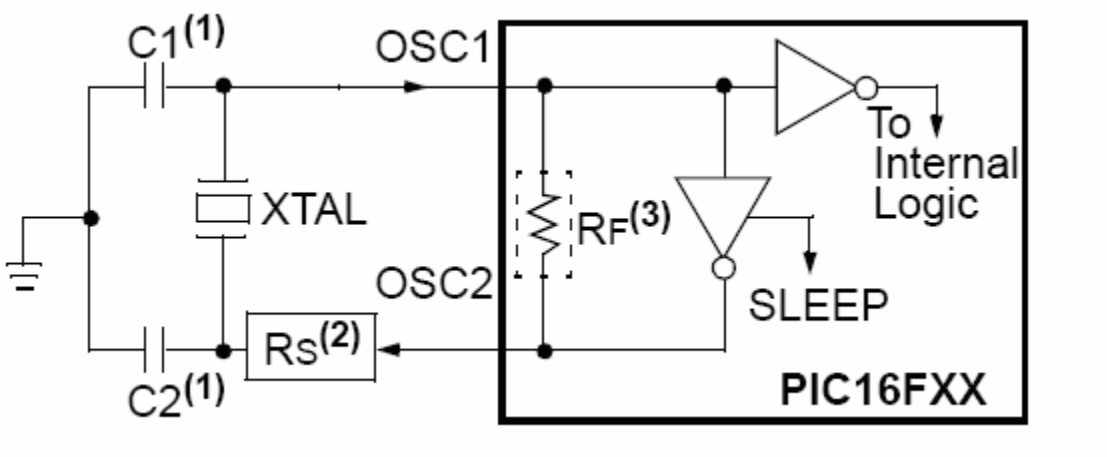

Figure 20: Crystal/Ceramic Resonator operational circuitry

The selection of the capacitors, C1 and C2, for ceramic resonators is based on the operational frequency of the microprocessor. Our system is designed to be operating at 4 MHZ frequency and based on this frequency rate, both the C1 and C2 capacitors corresponding to OSC1 and OSC2 pins respectively, are to be in the range of 15-33 pF. Having established this oscillator circuitry with proper capacitor values, the microcontroller was tested upon all devised test plans and the results were consistent with the expected behavior of the microcontroller and the overall performance of the system.

5.1.4 IR Emitters Sensitivity and Intensity Test

This test plan is devised to investigate the sensitivity of the IR emitters to their corresponding distance and orientation. To accomplish this, a basic test can be performed as follows. An IR emitter is positioned at different distances from the capturing device (the USB webcam) and/or with different orientations (0, 10, 45, 60 and 90 degrees). The intensity of the Infrared radiation is then analyzed using the captured data from the webcam. An appropriate pass condition would be an intensity of at least 20 mW/sr corresponding to the IR radiation captured and observed by the webcam. Please refer to Functional Specification, test plan 9.1.

5.1.5 USB Webcam Frequency and Sampling Rate Test

This test plan is devised to investigate the appropriate frame rate required from the webcam to ensure that the motions of the object under control with corresponding frequency rates will be captured by the webcam.

A simple test procedure can be set up as follows; a series of object's movements with different frequency rates is to be examined and captured by the USB webcam (e.g. 2, 5, and 10 Hz). Thereafter, the corresponding images are to be tested to verify the proper capturing functionality of the webcam considering the webcam's frame rate limitation. The expected behavior of the webcam is that, it should be able to detect all the motions

with a frequency rate of less than 100Hz. Please refer to Functional Specification, test plan 9.2.

5.1.6 Optical Marker's Control Test Utilizing Watchdog Timer

A test plan for the control unit along with the IR LED's has been design. Instead of the IR LED's, we used red LED's so that we can see the sequence of operation visually. The program code is similar to that of the flow chart in Figure 24 except that the delay for which the LED is on increased so that the LED flashing is visible with the naked eye. This test jig will be demonstrated during the oral presentation. We will also test for the proper operation of the timing using an oscilloscope to measure the timing of the proper delays. The oscilloscope channel can be connected to one of the decoder's outputs and we can test the duration of the LED while it is on. We can also connect the other channel of the oscilloscope to another consecutive output of the decoder and measure the time between the transitions of LED's to ensure that the WDT is operating properly.

5.2 Software Test Plan

5.2.1 Low Level Testing

5.2.1.1 Trying different frame rates

User Input: Since it is a low level testing, there is no input from user
Conditions: Testing for different frame rates.
Expected Observation: We do not have a specific expected value of the maximum frame rate. However, depending on the complexity of the lower level software design, we expect to find the maximum frame rate to be greater than 25 frames/sec.

5.2.1.2 Synchronization with hardware sensors

User Input: Since it is a low level testing, there is no input from user
Conditions: We will give various frame rate to synchronize with the hardware sensors.
Expected Observation: We expect that the software perfectly synchronize with the hardware as it was discussed in the section 4.2.2.

5.2.1.3 Detecting distorted image

User Input: Since it is a low level testing, there is no input from user
Conditions: We rotate the IR sensor to manipulate a distorted image of an IR light source.
Expected Observation: As it was discussed in the section 4.3.3, software will eliminate the distorted image from its decision making process

5.2.1.4 Detecting virtual image

User Input: Since it is a low level testing, there is no input from user
Conditions: We rotate the IR sensor to manipulate a virtual image of an IR light source.
Expected Observation: As it was discussed in the section 4.3.3, software will eliminate the virtual image from its decision making process.

5.2.2 High Level Testing

In a software debugger, the functionality of the user interface software will be tested to confirm that all menu options act accordingly to the Functional Specification. This test is expected to remove potential bugs that arise in the user interface logic routines.

Further, several people other than the group members will be asked to work with the user menu and report their difficulties and their suggestions for the interface modification. This test will ensure that the system is user friendly and it is easy to be utilized by people who do not have an excessive knowledge about the design of the system.

5.3 Overall System Testing

After testing the hardware and software of the system, the overall integrated system should be tested to ensure that all of the subunits of the system are in synchronization with each other, the system satisfy the required Functional Specification, and the animation unit simulates an accurate model of the motion of the object.

To perform this test we should perform the following steps:

1. Start the hardware so that the LED's start blinking one at a time.
2. Start the webcam capturing applications.
3. Move all the joints of the model in any possible way in the two dimensional plane with different speeds.
4. Compare the resulted animation with the actual movement of the object and verify that each joint is being modeled accurately.

The system should be able to simulate the motion of the object in a working volume of $1\,m^3$ according to the Functional Specification.

6 Conclusion

The proposed design solutions to meet the Functional Specification of the Motion Capture System have been discussed in this document. During the actual development, these design specifications will be adhered to as much as possible to meet the Functional Specification.

By implementing the test plans included in the design specifications, we can ensure that all the required Functional Specification of the system will be met.

Appendix A - Screen Captures

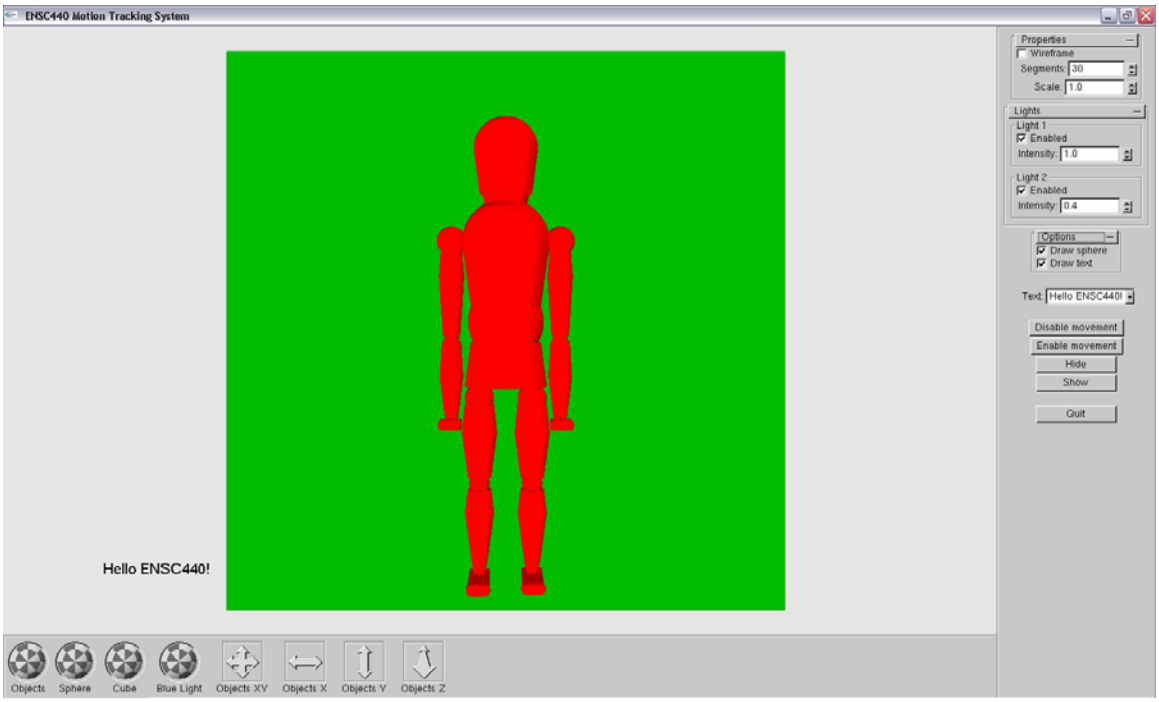

Figure 21: User Interface (Original Position)

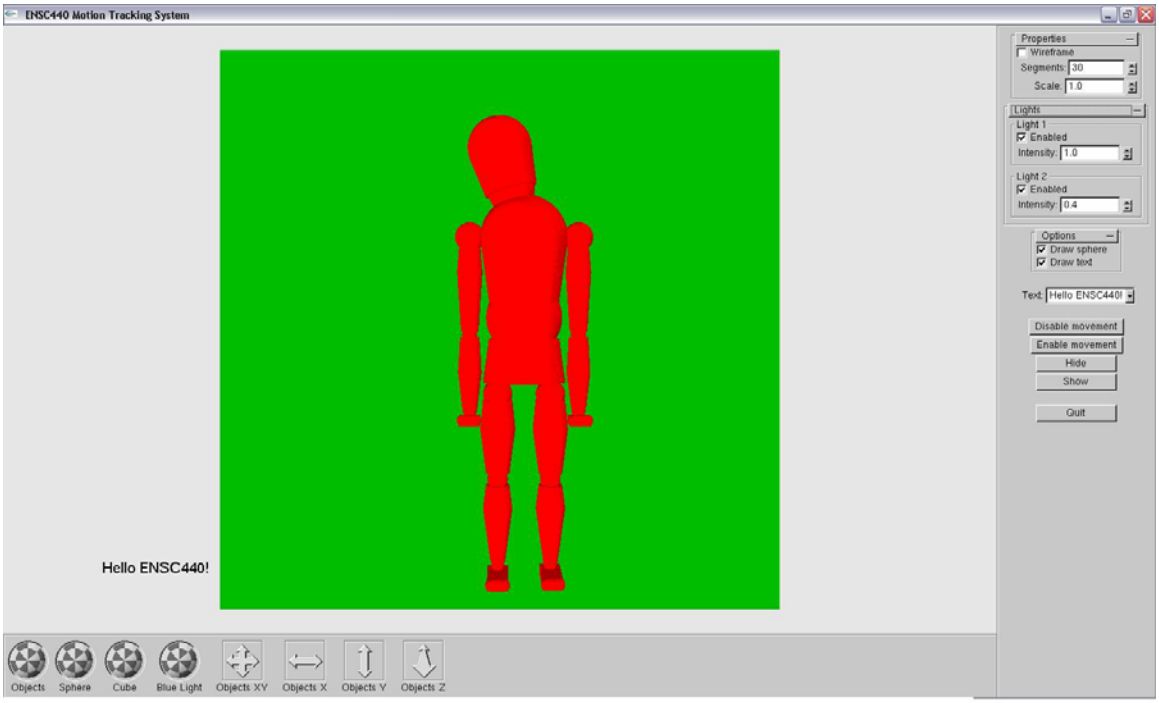

Figure 22: User Interface (Neck Rotated)

Appendix B - Schematics

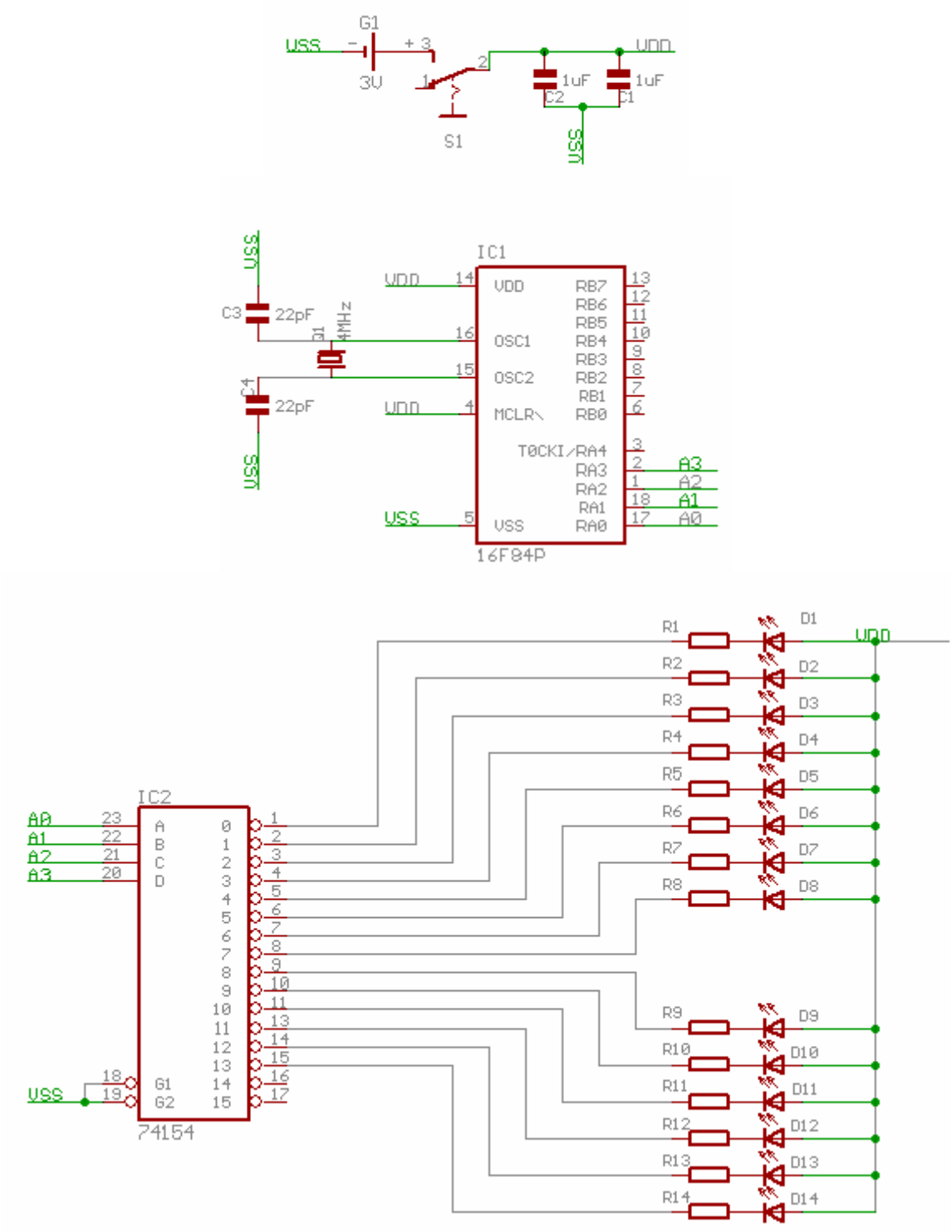

Figure 23: Schematics of the Hardware System

Appendix C - Flow Charts

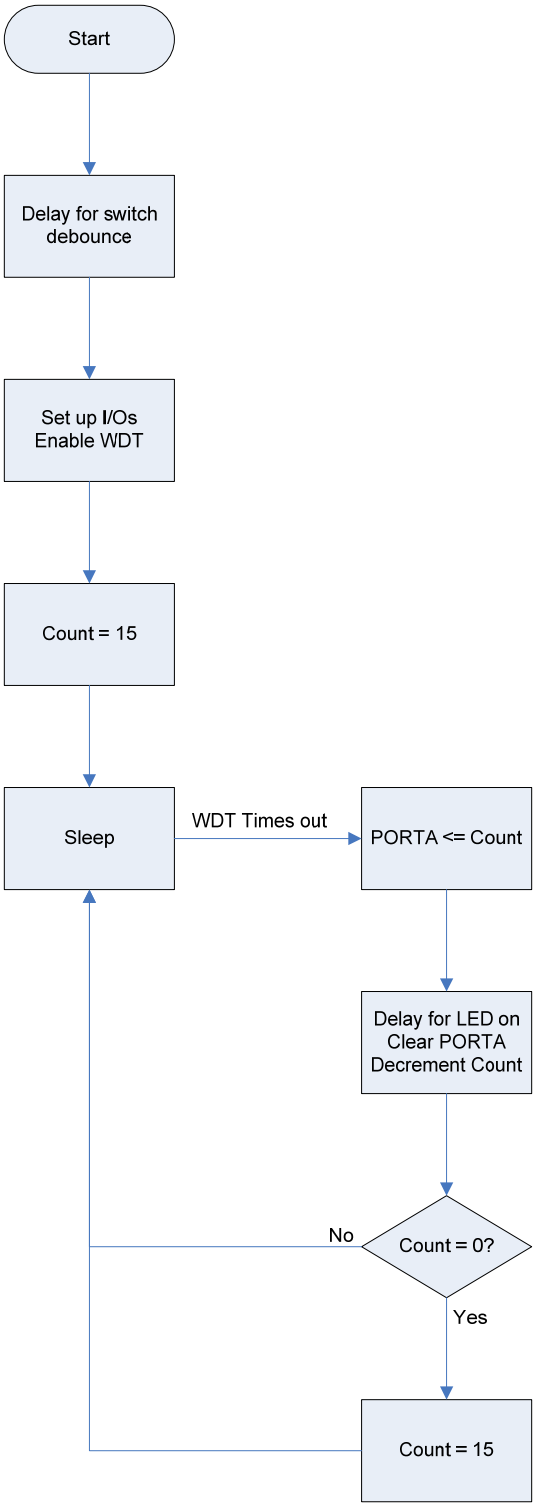

Figure 24: Flowchart of Microcontroller Program Code

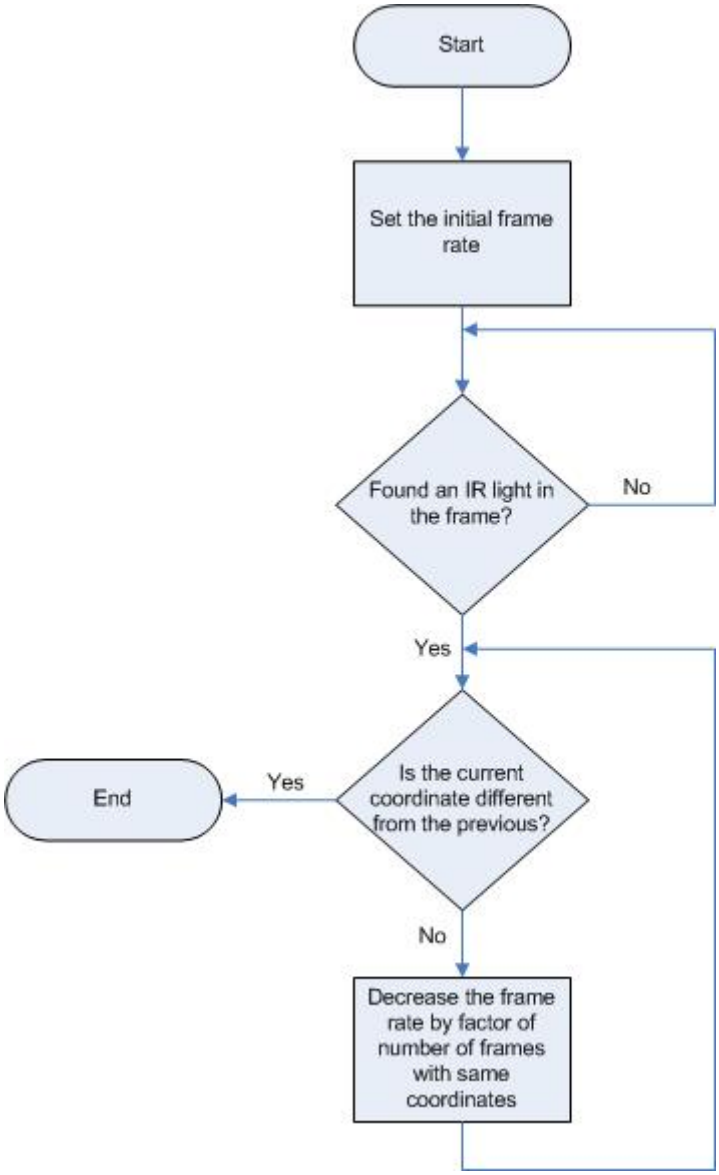

Figure 25: Flowchart of Adjusting Flow Rate

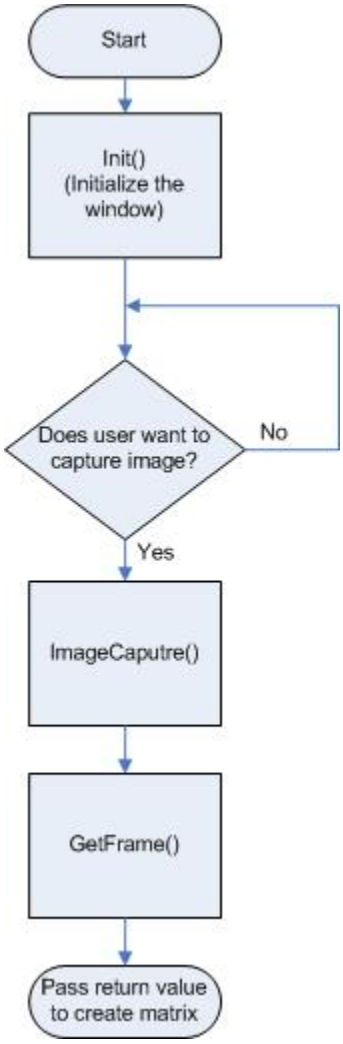

Figure 26: Flowchart of Capturing Image

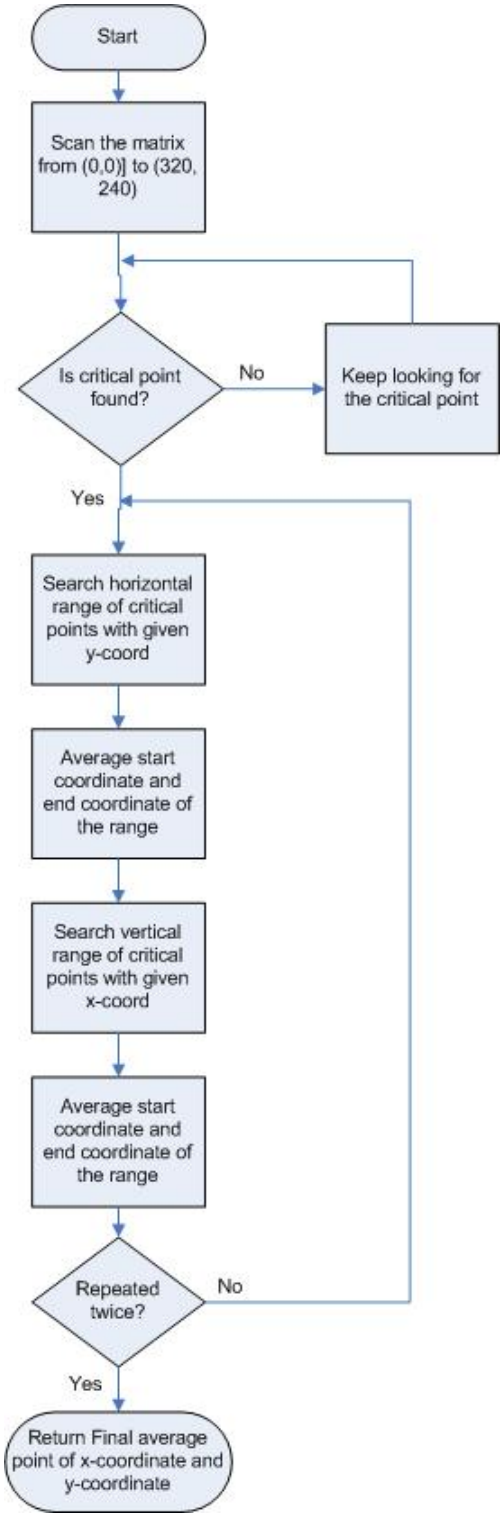

Figure 27: Algorithm to Find Coordinate of IR LED

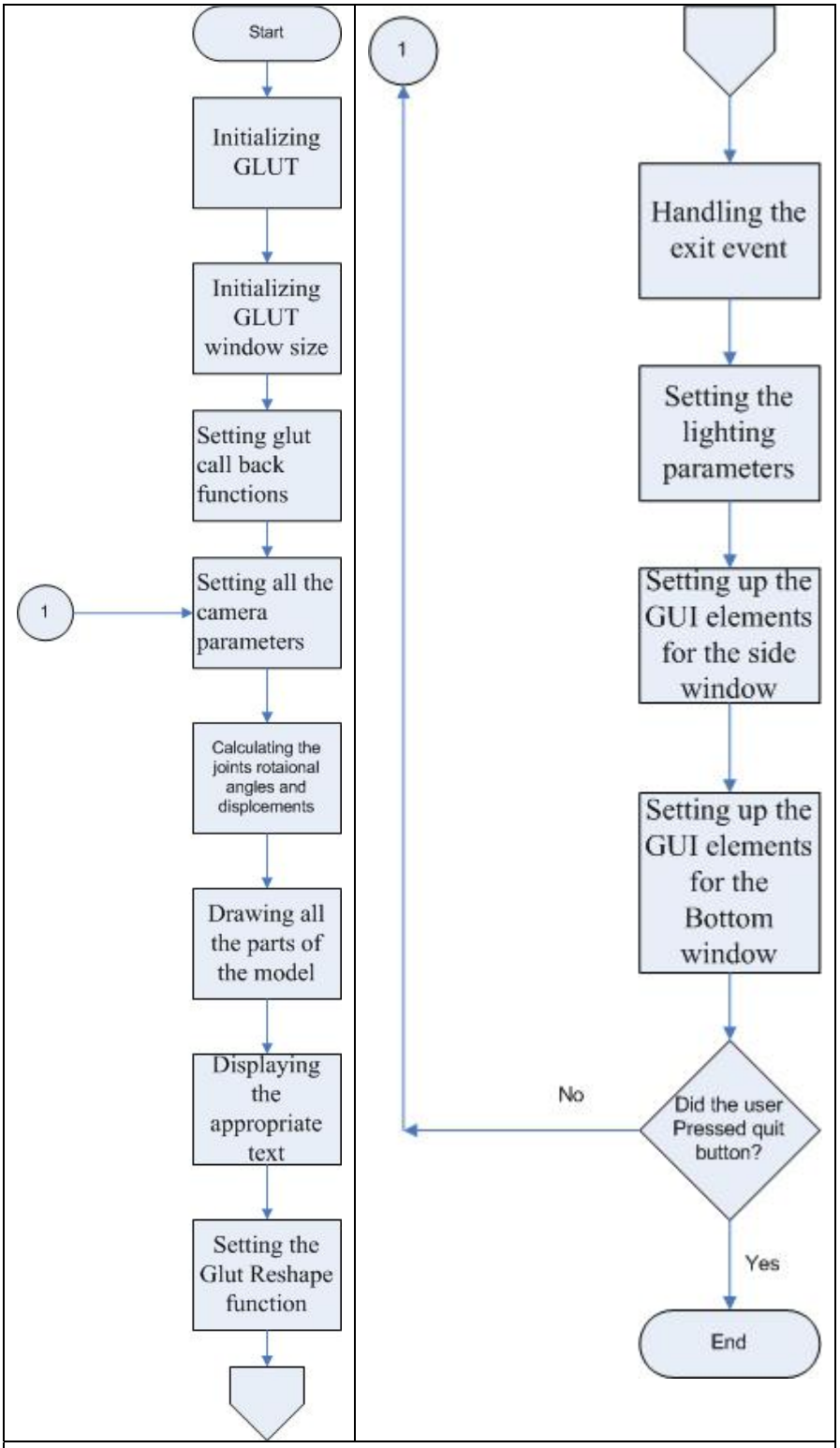

Figure 28: Flowchart of High Level Software