Media Engineering and Technology Faculty
German University in Cairo

Bachelor Thesis Title

Bachelor Thesis

Author:	Author
Supervisors:	Sup 1
	Sup 2
	Sup 3
Submission Date:	XX July, 20XX

Media Engineering and Technology Faculty
German University in Cairo

Bachelor Thesis Title

Bachelor Thesis

Author:	Author
Supervisors:	Sup 1
	Sup 2
	Sup 3
Submission Date:	XX July, 20XX

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

Author
XX July, 20XX

Acknowledgments

Text

Abstract

Abstact

Todays technology is getting smarter producing amazing new opportunities and making our lives easier. This project allows you to control home devices through a touchless interface. Some might think it's not worth the technology expenses. However it saves power, time, effort and removes obstacles. This is a thesis presented in the development of Microsoft Kinect sensor and home automation technology by using Depth tracking, voice recognition, gesture recognition and touchless graphical interface

Contents

Chapter 1

Introduction

Lately, touchless interface devices have appeared in the technology world making our lives easier. Currently the average household pays over 100 dollars a month for electricity according to the U.S. Energy Information Administration. Half of those costs are to keep the motor and compressors in operation in air conditioners and heating systems. If consumers had the ability to control their homes climate control systems remotely, they could save money without sacrificing convenience and comfort. The idea is to extend the Kinect's potential uses from gaming to be used in different regions making more and better use of it's capabilities. This is a handicapped-friendly project that provides an easy way to control home devices using Microsoft Kinect sensor and Arduino board by giving voice commands, doing gestures with the help of a touchless graphical interface.

1.1 Kinect sensor

Kinect is a motion sensing input device by Microsoft for the Xbox 360 video game console and Windows PCs. Based around a webcam-style add-on peripheral for the Xbox 360 console, it enables users to control and interact with the Xbox 360 without the need to touch a game controller through a natural user interface using gestures and spoken commands to provide full-body 3D motion capture, facial recognition and voice recognition capabilities.

1.1.1 Capabilities and constraints

The kinect sensor contains of an infrared projector, a LED light indicator, a color camera, an IR camera, a Microphone array and a tilt motor. The area required to play Kinect is roughly 6 meters squared, although the sensor can maintain tracking through an extended range of approximately (2.320 ft). The sensor has an angular field of view of 57 horizontally and 43 vertically, while the motorized pivot is capable of tilting the sensor up to 27 either up or down. The horizontal field of the Kinect sensor at the minimum viewing distance of 0.8 m (2.6 ft) is therefore 87 cm (34 in), and the vertical field is 63 cm (25 in), resulting in a resolution of just over 1.3 mm (0.051 in) per pixel. The microphone array features four microphone capsules and operates with each channel processing 16-bit audio at a sampling rate of 16 kHz.

1.1.2 history

Kinect itself was first announced on June 1, 2009 at E3 2009 under the code name "Project Natal". Following in Microsoft's tradition of using cities as code names. In March 2012, Microsoft announced that next version of the Kinect for Windows would be available in May 2012. On June 17, 2011, Microsoft finally released the Kinect SDK beta to the public under a non-commercial license after demonstrating it for several weeks at events like MIX. As promised, it included the skeleton recognition algorithms that make an initial pose unnecessary as well as the AEC technology and acoustic models required to make Kinect speech recognition system work in a large room. Every developer now had access to the same tools Microsoft used internally for developing Kinect applications for the computer. on May 21, 2012, Kinect for Windows 1.5 was released. It adds new features, support for many new languages and debut in 19 more countries.

1.2 Microsoft SDK

The Kinect for Windows SDK is the set of libraries that allows us to program applications on a variety of Microsoft development platforms using the Kinect sensor as input. With it, we can program WPF applications, WinForms applications, XNA applications and, with a little work, even browser-based applications running on the Windows operating systemthough, oddly enough, we cannot create Xbox games with the Kinect for Windows SDK. Developers can use the SDK with the Xbox Kinect Sensor. In order to use Kinect's near mode capabilities, however, we require the official Kinect for Windows hardware. Additionally, the Kinect for Windows sensor is required for commercial deployments. The Kinect for Windows 1.5 SDK includes 'Kinect Studio' a new app that allows developers to record, playback, and debug clips of users interacting with applications. Support for new "seated" or "10-joint" skeletal system that will let apps track the head, neck, and arms of a Kinect user - whether they're sitting down or standing; which would work in default and near mode. Support for four new languages for speech recognition French, Spanish, Italian, and Japanese. Additionally it would add support for regional dialects of these languages along with English.

1.2.1 Requirements

Unlike other Kinect libraries, the Kinect for Windows SDK, as its name suggests, only runs on Windows operating systems. Specifically, it runs on x86 and x64 versions of

Windows 7. It has been shown to also work on early versions of Windows 8. Because Kinect was designed for Xbox hardware, it requires roughly similar hardware on a PC to run effectively. Hardware Requirements Computer with a dual-core, 2.66-GHz or faster processor Windows 7compatible graphics card that supports Microsoft DirectX 9.0c capabilities 2 GB of RAM (4 GB or RAM recommended) Kinect for Xbox 360 sensor Kinect USB power adapter Use the free Visual Studio 2010 Express or other VS 2010 editions to program against the Kinect for Windows SDK. You will also need to have the DirectX 9.0c runtime installed. Later versions of DirectX are not backwards compatible. You will also, of course, want to download and install the latest version of the Kinect for Windows SDK. The Kinect SDK installer will install the Kinect drivers, the Microsoft Research Kinect assembly, as well as code samples. Software Requirements Microsoft Visual Studio 2010 Express or other Visual Studio 2010 edition: Microsoft .NET Framework 4 The Kinect for Windows SDK (x86 or x64): For C++ SkeletalViewer samples: DirectX Software Development Kit, June 2010 or later version: DirectX End-User Runtime Web Installer: To take full advantage of the audio capabilities of Kinect, you will also need additional Microsoft speech recognition software: the Speech Platform API, the Speech Platform SDK, and the Kinect for Windows Runtime Language Pack. Fortunately, the install for the SDK automatically installs these additional components for you. Should you ever accidentally uninstall these speech components,

1.2.2 Hacking history

Almost as soon as the Kinect was released, people began trying to figure out how Kinect worked. Then came the hacks, then the alternate applications On November 6, a hacker known as AlexP was able to control Kinects motors and read its accelerometer data. On Monday, November 8, AlexP posted video showing that he could pull both RGB and depth data streams from the Kinect sensor and display them. Hector Martin, a computer science major in Bilbao, Spain, had just purchased Kinect and managed to write the driver and application to display RGB and depth video. Martin became a contributor to the OpenKinect group and a new library, libfreenect, became the basis of the communitys hacking efforts.Throughout November, people started to post videos on the Internet showing what they could do with Kinect. Kinect-based artistic displays, augmented reality experiences, and robotics experiments started showing up on YouTube. Sites like KinectHacks.net sprang up to track all the things people were building with Kinect. By November 20, someone had posted a video of a light saber simulator using Kinectanother movie aspiration checked off. Microsoft, meanwhile, was not idle. The company watched with excitement as hundreds of Kinect hacks made their way to the web. On December 10, PrimeSense announced the release of its own open source drivers for Kinect along with libraries for working with the data. This provided improvements to the skeleton tracking algorithms over what was then possible with libfreenect and projects that required integration of RGB and depth data began migrating over to the OpenNI technology stack that PrimeSense had made available. Without the key Microsoft Research technologies, however, skeleton tracking with OpenNI still required the awkward T-pose to initialize skeleton recognition.

1.2.3 Kinect vs OpenNI

The Official SDK is developed by Microsoft which also develops the hardware and therefore should know internal information about the device that the open source society must reverse engineer. Having economy behind it gives many advantages. On the other hand, never underestimate the force of the open source society: ”The OpenKinect community consists of over 2000 members contributing their time and code to the Project. Our members have joined this Project with the mission of creating the best possible suite of applications for the Kinect. OpenKinect is a true ”open source” community!” OpenKinect was released long before the official SDK as the kinect device was hacked on the first or second day of its release.

Programming languages supported: Official SDK: C++, C, or Visual Basic by using Microsoft Visual Studio 2010. OpenKinect: Python, C, C++, C, Java, Lisp and more! Obviously not requiring Visual Studio.

Operating systems support: Official SDK: only installs on Windows 7. OpenKinect: runs on Linux, OS X and Windows

License: The Official SDK is in its current beta state only for testing. The SDK has been developed specifically to encourage wide exploration and experimentation by academic, research and enthusiast communities. commercial applications are not permitted. Note however that this will probably change in future releases of the SDK. Visit the FAQ for more information OpenKinect appers to be open for commercial usage, but online sources state that it may not be that simple. I would take a good look at the terms before releasing any commercial apps with it.

Documentation and support:

Official SDK: well documented and provides a support forum OpenKinect: appears to have a mailing list, twitter and irc. but no official forum/QA? Documentation on website is not as rich as I would like it to be.

the official SDK came with drivers that installed out of the box came with examples and code for easy getting into business

1.2.4 What Kinect can’t do

Voice recognition doesn’t work as expected when youre surrounded by ambient noise from other guests can’t detect the kinds of movements shown,If you want the Kinect to see you do something, you have to move in very pronounced motions because it seems only large movements can be seen by the stereoscopic cameras on the sensor. This works great for exercise games where you have to run in place or otherwise be full body active, but when all youre doing is making a pushing motion, the Kinect cant see this and doesnt do anything. Additionally, the Kinect doesnt like when theres a bunch of movement taking place all at once. its just too much input and it will simply not detect the motion

See fig:kinectsensor fig:datastream

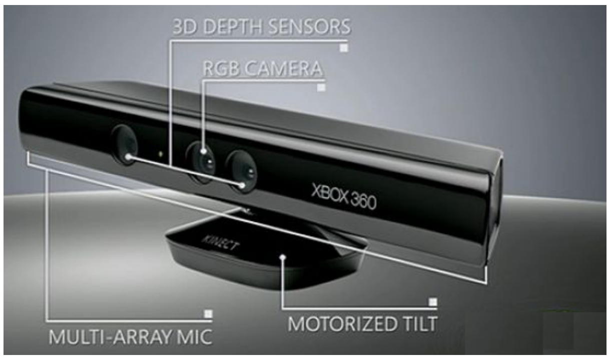

Figure 1.1: Kinect sensor

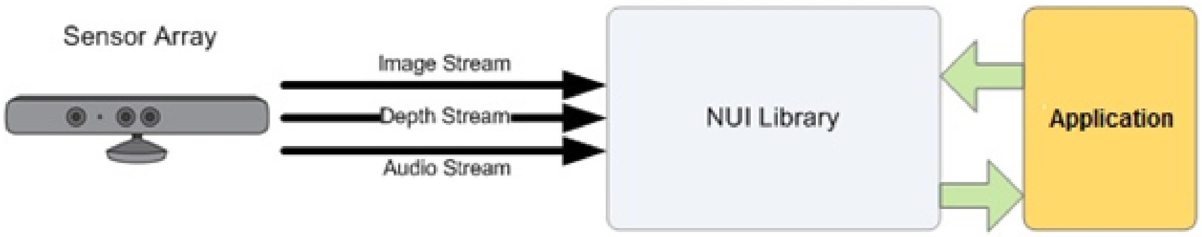

Figure 1.2: Data stream

1.3 Microsoft XNA

1.3.1 Definition and Overview

Microsoft XNA is a set of tools with a managed runtime environment provided by Microsoft that facilitates video game development and management. XNA is based on the .NET Framework for Xbox 360 development and .NET Framework on Windows. It includes an extensive set of class libraries, specific to game development, to promote maximum code reuse across target platforms.

1.3.2 Documentation

Microsoft.Xna.Framework: Provides commonly needed game classes such as timers and game loops. Microsoft.Xna.Framework.Audio: Contains low-level application programming interface (API) methods that can load and manipulate XACT-created project and content files to play audio. Microsoft.Xna.Framework.Content: Contains the run-time components of the Content Pipeline. Microsoft.Xna.Framework.Design: Provides a unified way of converting types of values to other types. Microsoft.Xna.Framework.GamerServices: Contains classes that implement various services related to gamers. These services communicate directly with the gamer, the gamer's data, or otherwise reflect choices the gamer makes. Gamer services include input device and profile data APIs. Microsoft.Xna.Framework.Graphics: Contains low-level application programming interface

(API) methods that take advantage of hardware acceleration capabilities to display 3D objects. Microsoft.Xna.Framework.Graphics.PackedVector: Represents data types with components that are not multiples of 8 bits. Microsoft.Xna.Framework.Input: Contains classes to receive input from keyboard, mouse, and Xbox 360 Controller devices. Microsoft.Xna.Framework.Input.Touch: Contains classes that enable access to touch-based input on devices that support it. Microsoft.Xna.Framework.Media: Contains classes to enumerate, play, and view songs, albums, playlists, and pictures. Microsoft.Xna.Framework.Net: Contains classes that implement support for Xbox LIVE, multiplayer, and networking for XNA Framework games. Microsoft.Xna.Framework.Storage: Contains classes that allow reading and writing of files.

1.3.3 Requirements

Since XNA games are written for the runtime, they can run on any platform that supports the XNA Framework with minimal or no modification. Games that run on the framework can technically be written in any .NET-compliant language, but only C in XNA Game Studio Express IDE and all versions of Visual Studio 2008 and 2010 (as of XNA 4.0) are officially supported.

1.3.4 Basics

Any class that implements subclass Microsoft.XNA.Game DrawableGameComponent should override 4 main methods and call them in order Initialize(): Allows the game to perform any initialization it needs to before starting to run. This is where it can query for any required services and load any non-graphic related content.

LoadContent(): LoadContent will be called once per game and is the place to load all of your content eg: textures, animations, fonts or any external files. UnLoadContent(): works as a destructor Update(GameTime): Allows the game to run logic such as updating the world,checking for collisions, gathering input, and playing audio. it works as a while(true) loop according to certain flags. gameTime Provides a snapshot of timing values. Draw(GameTime): This is called when the game should draw itself.

1.4 Arduino

Arduino is a single-board microcontroller designed to make the process of using electronics in multidisciplinary projects more accessible. It's a tool for making computers that can sense and control more of the physical world than your desktop computer and thus can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board. The hardware consists of a simple open source hardware board designed around an 8-bit Atmel AVR microcontroller, though a new model has been designed around a 32-bit Atmel ARM. The software consists of a standard programming language compiler and a boot loader that executes on the microcontroller. Arduino projects can be stand-alone, or they can be communicate with software running on your computer. The boards can be assembled by hand or purchased preassembled. fig:arduinoboard

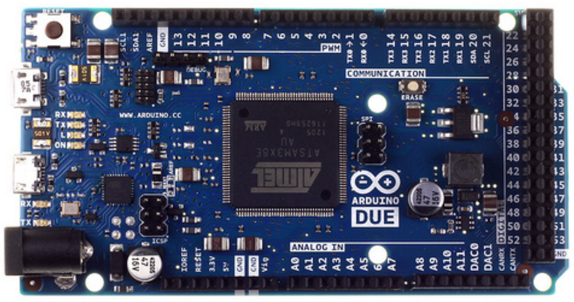
Figure 1.3: Arduino Board

1.4.1 Requirements

The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows

1.4.2 IDE

Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino

1.4.3 How it works

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

The Arduino is a microcontroller board based on the ATmega328 . It has 14 digital input/output pins of which 6 can be used as PWM outputs, 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. The power pins are as follows: The ATmega328 has a 32 KB memory. It also has 2 KB of SRAM and 1 KB of EEPROM.

1.4.4 IDE basics

The Arduino integrated development environment (IDE) is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit makefiles or run programs on a command-line interface. A program or code written for Arduino is called a sketch. Sketches are saved with the file extension .ino. It has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino environment including complete error messages and other information. The bottom righthand corner of the window displays the current board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor Arduino programs are written in C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much easier. Users only need define two functions to make a runnable cyclic executive program: setup(): a function run once at the start of a program that can initialize settings loop(): a function called repeatedly until the board powers off to execute your code you only need two buttons: Verify which Checks your code for errors. Upload which Compiles your code and uploads it to the Arduino I/O board. See

Chapter 2

Background

2.1 KinectControl Overview

2.1.1 User Stories

The user stories and the program features are as follows:

- As a user, I can turn the lights on/off by entering/leaving the room.
- As a system, I should be able to recognize certain gestures and control the device accordingly.
- As a user, I can interact with a touchless interface.
- As a user, I can give certain voice commands.
- As a system, I should have a circuit connection between the arduino board and the device I want to control.
- As a system, I should be able to control the device and switch it on/off when needed.
- As a system, I should create an interface to allow the communication between the kinect sensor and arduino board.
- As a user, I should see pop-up screens to show feedback from the interface at certain circumstances.
- As a user, I should see an avatar which represents my distance to the kinect.
- As a user, I should see an introductory fading screen to the application.
- As a user, I can play music.

- As a user, I should see a main screen where I can edit settings and enable/disable features like face/depth/voice/skeletal tracking.
- As a user, I should see a screen for each controlled device where I can edit it's settings manually.
- As a system I should provide a way for the user to enter a password to be able to manage the application.

2.1.2 Software Architecture

The software architecture of the project is shown in the figures below Classes documentation See fig:softwarearchitecture

2.1.3 Kinect Arduino Communication

2.2 Voice Recognition

The kinect's voice recognition system is implemented with the helpe of Microsoft.Speech library and accessing the audiostream of the kinect sensor. The system takes as an input an array of commands which is the grammar of words it should detect and compares each of them to the heard word with a certain error ratio which is the confidence, The gesture recognition system is done by a gesture controller which contains all gestures defined and updates them with the skeleton data and constantly checks if any is detected through a certain amount of frames

2.2.1 defects

however its not fully efficent as it differs various words and various accents so sometimes you may want to lower the confidence. It also may have some bugs according to the surrounding noise Speech recognition is only supported in the following locales English,French,German,Italian,Japanese,Spanish. Because of the complexity of solving for languages and accents voice support will roll out across

2.3 Gesture Recognition

The Microsoft SDK provides the player's skeleton as an array of joints of a Vector4 value each, which contains each joint's x,y,z and w values; it's x,y,z position values in 3d space and the w value that indicates the quality level (Range between 0-1)) of the position that indicates the center of mass for that skeleton which is only 1 for passive fully tracked players. The kinect sensor can detect up to 6 players at the same time, however it can only track 2 at the same time while it only stores some basic information like position values for the other 4. There's many ways you could build a gesture recognition system using the kinect SDK for example you can store your gestures as frame images and use image proccessing to compare them, or you can just compare the angle between certain joints but this won't be as efficient as it may detect other random gestures too, so it's better to compare joint positions relatively to eachothers or draw imagionary vectors between the joints and compare them to the expected directions. You can divide the gesture into smaller gesture parts/segments if needed and check if they succeed in each frame according to their topological order in a certain number of frames then the gesture succeeds. (See fig:skeletonjoints)

2.3.1 How it works

a gesture is defined by an array of gesture segments and the gesture type a segment is a division of a gesture such that a gesture can be one segment or more so basically a gesture is a sequence of segments if they all are accepted in order the gesture itself is accepted aswell each segment check can return three values, fail, pause and succeed, the pause event is raised meaning that the gesture is in the middle of being performed, it waits for a few more frames representing a suitable time factor if the user does not complete the gesture it raises a fail gesture event which means this segment was not performed as expected, see fig:gesturearchitecture1 and fig:gesturearchitecture2

The check can be made by comparing the skeleton joints x,y and z positions relatively to each others and/or by meassuring the angle between joints using the help of the skeleton analyzer class.

2.3.2 How to add a new gesture

To Add a new gesture simply create a class for each gesture segment implementing the IRelativeGestureSegment interface and override the checkgesture result and write the algorithm for your gesture.

2.3.3 Defects

However this system may have some disadvantages if the gestures are somehow similar or overlapping meaning if a gesture sequence of segments is a prefix or a suffix of another both gestures will be detected to avoid this conflict the implementor should wait for more frames to see if the bigger gesture is detected but this will rule out the real time detection factor of the system which is important there was no gesture recording software compatible with the microsoft sdk found. also some gesture may not be able to be detected due to the fact that the microsoft sdk detect the user as a skeleton as an array

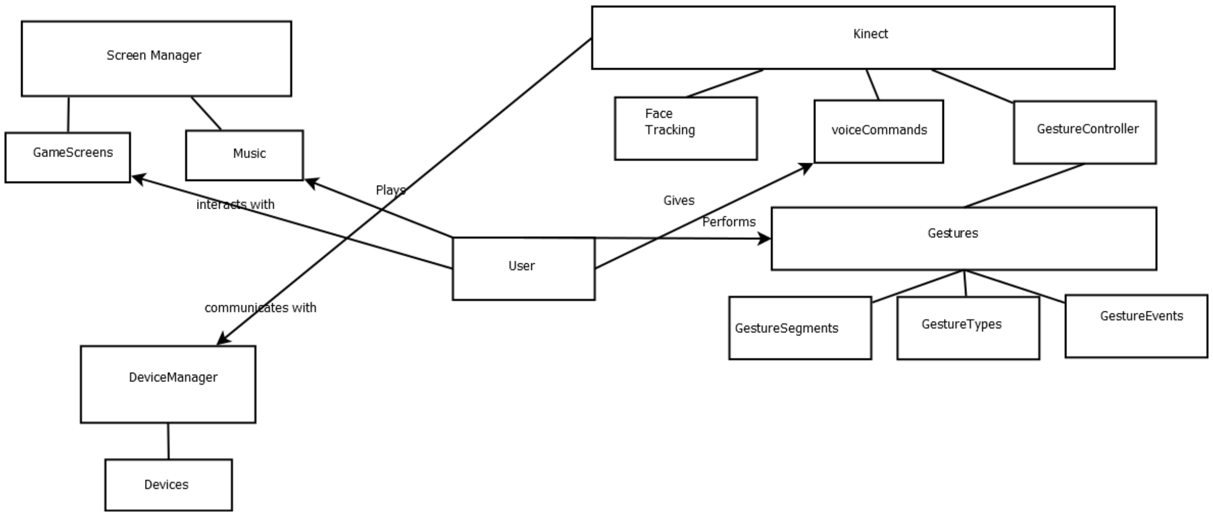

Figure 2.1: Software Architecture

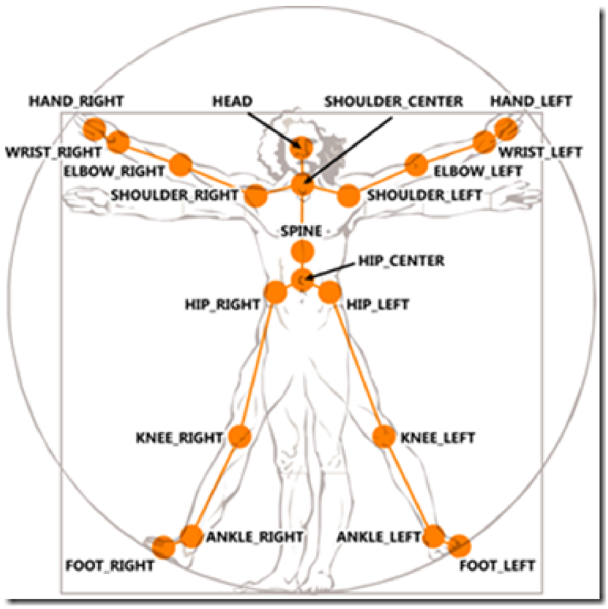

Figure 2.2: Skeleton Joints

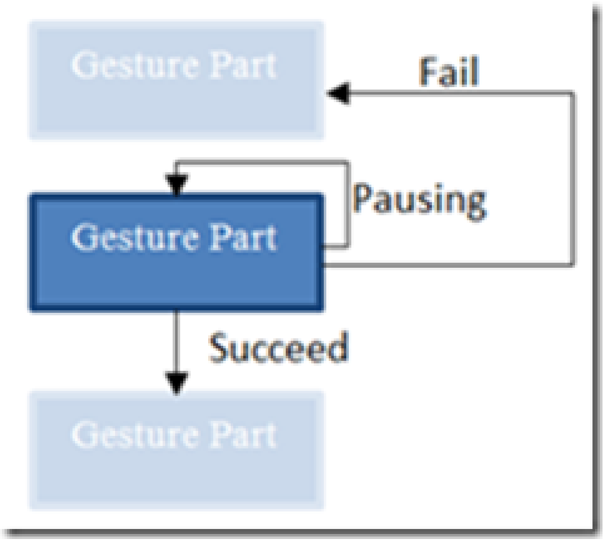

Figure 2.3: Gesture Architecture1

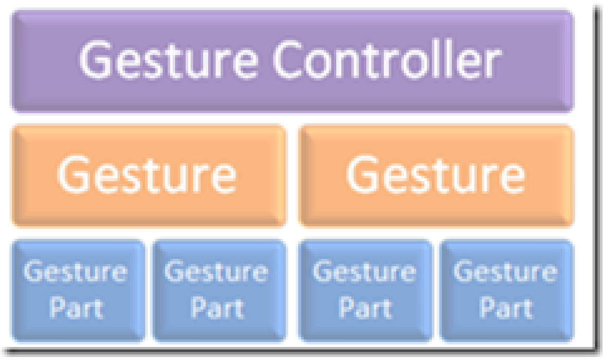

Figure 2.4: Gesture Architecture2

of joints, so for an example the system will not be able to differentiate between a fist bump gesture and a clap gesture. however its really rare the user may want to implement two similar gestures so the above points are not big turn offs for the abilities system

2.4 User Interface

to add a gesture you just have to define its segments, define the gessture as an array of these segments with certain repitions and order then add them to the gesture controller, add the gesture type in the gesture types class

2.4.1 Screen Manager

the ScreenManagers handles how to add a screen on top of another, how to show it, remove it, pause/unpause the creen and what screens comes next each screen is a subclass of the class GameScreen which contains all the things that will be loaded and drawn common in each screens such as the useravatar, voice recognition and the music player

2.4.2 User Avatar

the avatar on top right checks if the uer is standing in correct position, the kinect sensor can detect uesr from 2 to 8 meters distance, it was implemented according to how the kinect sdk sees the skeleton in these distances dead means not detected, red means partially detected- too close- can't function well, green means good position, white means too far, stsarting to lose track of skeleton

2.4.3 Music Player

The music player is implemented with the help of using Microsoft.Xna.Framework.Media library. it's implemented. You can play, pause, stop skip to next or previous songs by saying the correct voice commands. to add your own music just add a folder named "music" into the project content folder

2.4.4 Kinect-Arduino Connection

The connection between the kinect and the arduino is done through serial ports with the help of the System.IO.Ports library. See fig:kinectarduinoconnection Serial ports, also called communication (COM) ports, are bi-directional. Bi-directional communication allows each device to receive data as well as transmit it. Serial devices use different pins to receive and transmit data using the same pins would limit communication to half-duplex, meaning that information could only travel in one direction at a time. Using different pins allows for full-duplex communication, in which information can travel in both directions at once. The connection manager system manages the connection details such as baudrate, portname, you can use it to open a port, write data to a port, recieve data and finally close the port when a certain gesture or voice command is detected, the application opens the port, sends certain data to the arduino, the arduino recieves it and does the expected command accordingly for example a "1" turns the internal LED attached to pin 13 on and "0" turns it off. Howeve the problem with this approach is that you need to specify the port name you want to send data through which is where the arduino board is connected, it can not be done dynamically unfortunately.

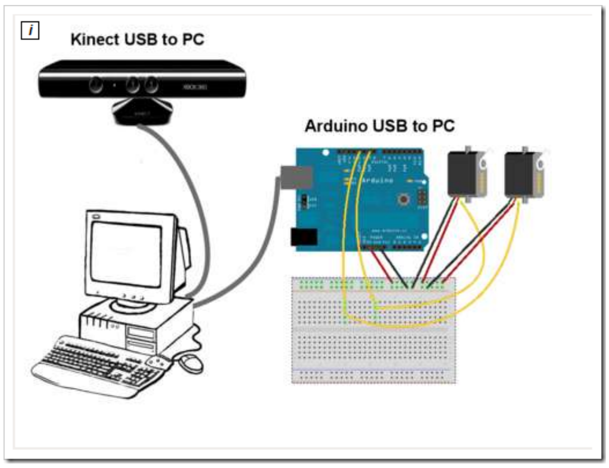

Figure 2.5: Kinect Arduino Connection

Chapter 3

Conclusion

Conclusion

Chapter 4

Future Work

Text

Appendix

Appendix A

Lists

Beginning kinect programming with microsoft kinect sdk by Jarrett Webb and James Ashley. Arduino cookbook by Michael Margolis. http://msdn.microsoft.co http://arduino.cc/en/Main

List of Figures

Bibliography