Learning Anatomy for Pre Schools Via Kinect Technology

by

Mohd Hazwan B. Sahabuzan

12019

Dissertation submitted in partial fulfilment of

the requirement for the

Bachelor of Technology (Hons)

(Business Information System)

1st August 2012

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Learning Anatomy via Kinect
For Pre-Schools

By

MohdHazwan B Sahabuzan
(12019)

A project dissertation submitted to the
Business Information SystemProgramme
UniversitiTeknologi PETRONAS
In partial fulfillment of the requirement for the
Bachelor of Technology (Hons)
(Business Information System)

Approved by,

……
(Assoc Prof. Dr. DayangRohayabtAwangRambli)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

May 2012

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

_____

MohdHazwan B Sahabuzan

ABSTRACT

In this project, we will discuss about the development and implementation of Kinect in learning body parts for pre-schoolers. The objective of this project is to introduce a new form of learning to children, to explore the use of Kinect based technology on science subject in schools, to develop a Kinect system for biology subject and make it more interactive, and finally to evaluate the reaction and acceptance of this technology in learning. The reason this project was decided is that to improve the current method of learning in schools. Traditional learning styles are usually boring and linear. This had caused some students to lose interest in the subject. So teachers are in constant need to do something to attract student's attentions and gain their interest. This is where my project comes in.We will develop software that uses Kinect technology to make learning more fun. Since we only have about 3 months time to develop this project, the methodology chosen for the system development is Throw-away-Prototyping. The reason this method is chosen is that it is fast and it helps to give clearer view of what the final product will looks like.In the course of the development of this project, there is a few problems that encountered. One of them is that it is not possible to use 3D model in the project within the time frame. So the solution is that pictures or images will be used instead of 3D models. The final prototype will have 3 main functions; the head where the students can learn about parts of their head or faces, the second part is the body where the student can learn which part of the body is called what. The final part is the extra where the pre-schoolers can have some fun.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES and Flow Charts

CHAPTER 1

INTRODUCTION

1.1 Background

Motion sensing has been available for a long time, but its application and availability to the public are very limited. Being expensive and far too complex to be operated by average Joe, it is unpractical for public uses. But in the past few years, these technologies are made public with the rise of gaming industries. A few major names in gaming industries introduce motion sensing gaming style to the public. In 2006 Nintendo introduces Nintendo Wii, its first motion sensing console. This is the first motion sensing gaming device made available to the public. This was then followed by the releasing of Sony Play station move and finally Xbox with its motion sensing project codename project NATAL, that later on named Kinect. Kinect is one of the latest motion sensing devices that are available to the public that are affordable. It was first release in November 2010 in Europe [1].

With this development in gaming industries, the public are made available to access motion sensing technologies with a cheap and affordable price. This has sparked a new interest in the public whereas users try to manipulate and customize these technologies for other purposes aside from gaming. Kinect works in web-cam style add-ons to the console. What Kinect does is that it detects user's movements' gestures with its infrared projectors and cameras, and is able to detect spoken commands [2].

On February 2011, Xbox announced that they will release Kinect System Development Kit (SDK) for windows. This SDK was then released for Windows 7

on 16 June 2011. The SDK allows developers to build applications for Kinect using Microsoft Visual Studio 2010, C++, and C#. Through this SDK, developers are able to access Kinect capabilities on:

1. Raw sensor Streams: - access low streams from the depth sensor, colour camera sensor, and four-element microphone array.
2. Skeletal Tracking: - The capability to track the skeletal image of one or two people, moving within the Kinect field of view for gesture-driven applications.
3. Advanced Audio Capabilities: - Audio processing capabilities include sophisticated acoustic noise suppression and echo cancellation, beam formation to identify the current sound source, and integration with the windows speech recognition API.
4. Sample code and Documentations.

For this project, the aim is to integrate these new technologies with learning process where we use Kinect to make learning more interesting and more interactive. These way students will feel more interested in the subject and the method of teaching will not be as boring as the students will be able to interact with the subject in more creative and fun ways with the help of Kinects [3].

1.2 Problem Statement

The learning process in class usually boring and are somewhat very linear. This is quite a problem as most students become uninterested in the subject and find the subjects taught are boring. [6] This is a common thing that happens in education regardless of which levels. This includes those pre-school students as well. Being that young, the only things that these children have interest in is to play. So they can hardly focus in class as much as we wanted them to.

So, more often than not, teachers find it hard to gain students attention and focused on the subject being taught. This has always been a problem in learning institutes especially in pre-schools where students are more interested to have fun than to study. Teachers need to find a way to attract student's attention and keep them focus, but they are having a hard time doing so and there are no tools to help them as well.

1.3 Objectives

The aims of this project are:

1. To introduce new form of learning to children.
2. To explore and develop a Kinect based software on learning anatomy.
3. To create a system that teacher can use to attract student's attentions and help to improve learning process by really visualizing what they learned.
4. To evaluate the reaction and acceptance of this technology in learning.

1.4 Scope of Study

The focus of this project is to improve the method of learning and teaching. Mainly this project will focus in learning Anatomy or body parts of the human body for pre-school students. The aim is to create learning systems where students and teachers can use Kinect in teaching for more interactive learning. This will introduce a new learning experience for students and teachers as well. The learning process will be more interactive and interesting.

The concept for this project is that we use the pre-schools syllabus we try to make a Kinect program that can help to make the learning more interactive and fun. Traditionally, what had been done to teach these children about the human body part is that we learn it through drawings and picture books. The idea for this project is basically using the same concept with a few changes, for example when learning; students can interact with 3D model through Kinect. The students can manipulate and interact with the subject through the help of Kinect. This will give students better understanding as they are able to see and feel themselves interacting with the system. This will make learning more fun and interesting, thus helping students to focus more on the subject that they learn.

1.5 The Relevancy of The Project

Kinect is one of the latest motions tracking device that are made available to the public, so far its usage are mainly in the gaming industries. What the project aims to do is to open a new field where these Kinect technologies can be used. There are many real world applications where motion detecting devices can be a real use. But right now its purposes and functions are still small and narrow, mainly because of the mindset of the people. Most people will associate Kinect with games as that is what they are intended to do in the first place. But the possibilities of its usage and implementation are almost limitless. And the developments of this project are as such. Not only the project seeks a new way to implements Kinect, the project also seeks a new fun way to learn and facilitate the learning process. If the project was a

success, this will change how subjects are being taught at schools; learning will be more fun and more interactive. Students will be more focused and more involved in the learning process and the teachers will be able to have their student's attentions better in class.

1.6 Feasibility within the Time Frame

For this project, the time frame is from six to seven months. The first 3 months will be on FYP 1, where the main planning and designing will take place. The next 3 month will be on FYP 2, where the real development and the implementation of the project are made. In terms of feasibility of time frame, the project is feasible.

CHAPTER 2

LITERATURE REVIEW

2.1 Interactive Learning

In the beginning of the year 2000, the learning style of the students began to change. People and educations institutions began looking into a new form of learning, called interactive learning. Interactive learning is a new learning style that incorporates social networking and urban computing into the design of the syllabus and delivery of the subject. In other words, it is a learning style that requires involvement of students at a level more than passive learning. Students will be able to have hands on experience and interact with the learning system more. So far, generation Y are the only generation that grew up with and are constantly in contact with digital media. It has become as natural as using a pen or a pencil and has become part of the cultural make up of generation Y. That is why it is very important to implement a new learning style that suits this new generation more. The old learning style that is the passive learning are simple inadequate for this new generation that are more connected with media and technology. [4]

The older generation must realize that it is essential to apply interactive learning. For the people of generation Y, computers, hand phones, gadgets, and media has become part of their lives. With the current development of technology, there are many new technologies that we can use and integrate in the learning process that could help in applying interactive learning. For example, the usage of multimedia tools such as Laptops and LCD projectors. Although some schools and higher learning institutes have implements these new teaching methods, there are still many areas in which it can be improved. Alferd Dork – University of California in his journal of Interactive Learning state that, not all of the current use of computers in education is progressive and useful for the future. [5] This is because most of the materials used in the teaching are less than impressive. But, we still make progress

every day and learn how to implement the usage of computers better in the learning process better. Today, it has become natural to expect University and colleges to use these multimedia tools in their delivery of knowledge and teaching style.

2.1.2 Passive learning

Passive learning is the most common learning style that was implemented in schools and in most education institutions in the past. Right now, more and more learning educations are moving towards interactive learning. But what is exactly passive learning. Passive learning is a classic learning method where students or learners just sit back, observe and listen to what ever has been taught to them.[6] They did little more other just listening to the lectures given by the teachers. Traditionally, this is what has been implemented in the past. Sometimes teachers or educators tried to make learning more fun and interactive by trying to communicate with the students by asking a few questions on related topics of what has been taught. But these kinds of communications are brief and usually ended with few short answers. These learning styles are no longer suitable to the current generation, which is generation Y.

2.1.3 Passive Learning VS Interactive Learning

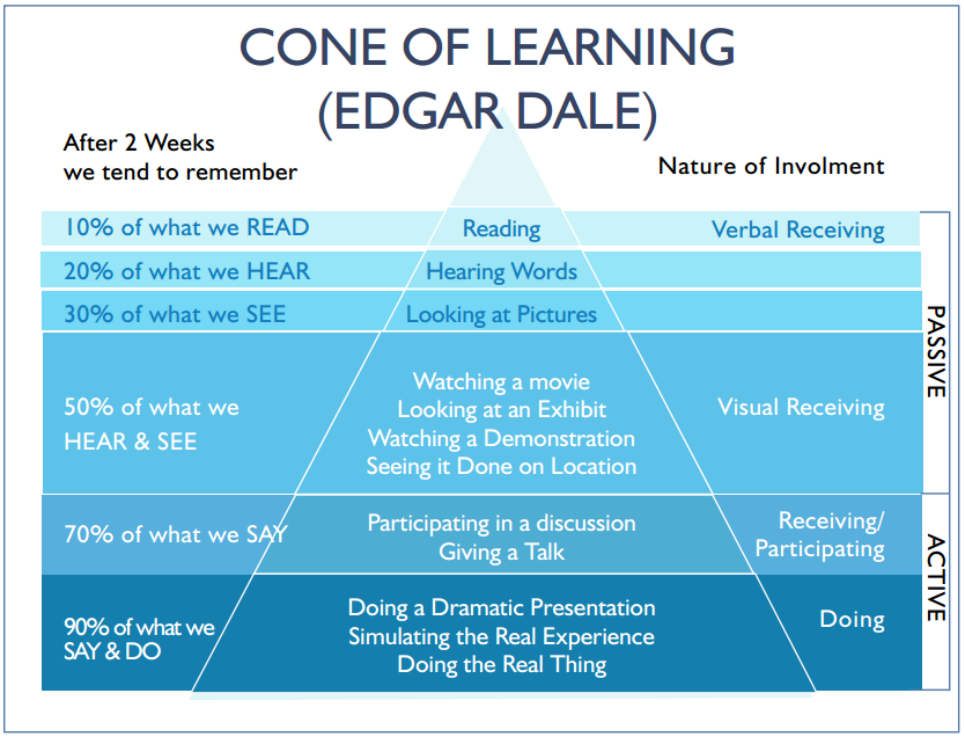


Figure 1: Cone of Learning

This is the cone of learning illustrated by the educationist Edgar Dale. [10] This is a study or a research that compares and collects data after two weeks of learning something new. This cone of learning compares what we tend to remember after two weeks based on a few categories. That is read, hear, see, hear & see, say, say and do. Basically it evaluates what we remember based on our involvement and interaction in certain activities. According to the cone of learning, if the levels of involvement are only through reading, as in we read a book; we will only remember about 10% of what we have read. Which is can be considered little or next to none at all. If we listen to lectures or hearing words from someone, we tend to remember 20% of what we hear. This percentage is still very low to be considered as an effective learning method. However, though seeing something, we tend to remember about 30% of the details. This is considerably higher than just listening. In passive learning, the process usually involves us to just sit back, observe and just listen to what the teachers taught in class. In other words, we can say that the process involve in passive learning is just hearing and seeing. So, according to Cone of Learning by

Edgar Dale, at most we will only remember 50% of what we actually learned in class.

However, we can see significant improvement in the interactive learning scale. For example, we will remember 70% of what we say. This can be relate to discussion and talk that we participate in class. If we are actively engaging discussion and talk in class, we could more effectively than just listening to lecturers by the teachers. To make things better, the study shows that we are likely to remember about 90% after two weeks if our involvement level is say and do. In other words, we are really did the real thing and have a hands on experience or participate in a dramatic presentations. As we can see using Edgar Dale cone of Learning, interactive learning is clearly more effective than passive learning with about 40% difference of effectiveness.

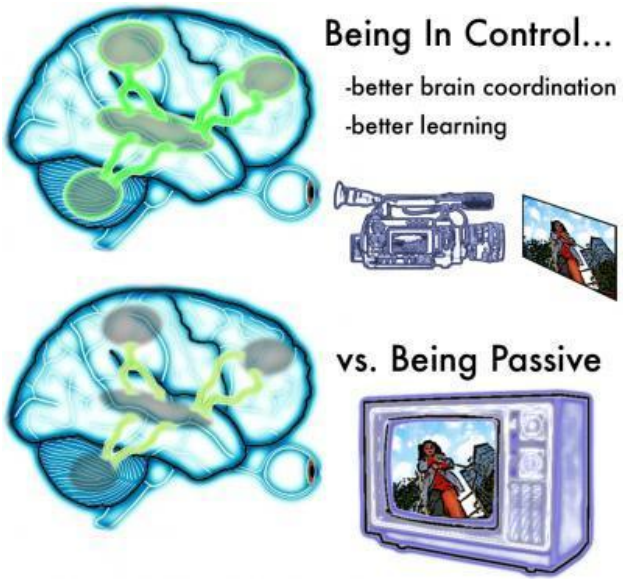


Figure 2: Passive Learning VS Active Learning

Hippocampus is the part of the brain that plays important roles when learning something new. It is a part of the brain than that coordinates other brain structures to do different task. For example, recognizing and remembering an object that has been seen before.

Joel Voss, Brian Gonsalves, Kara Federmeir, Daniel Tranel and Neal Cohen had design a few experiment for their study/paper Hippocampal Brain-Network Coordination During Volitional Exploratory Behaviour Enhances Learning.[11] The main idea of this study is to see whether voluntary control over what and when a subject learns increases the subject ability to recall that information. Certain aspects of memory are the result of correlations in activity between the hippocampus and other brain regions. Volitional control improves these interactions. According to their study, they can see significant difference in the brain activity on active and passive learners. Those who are active learners shows more brain activity than those who of passive learners. [12]

2.2 Augmented Reality

Augmented Reality or AR for short is a term used when the view of real world are augmented by computer generated sensory input such as video, sound, graphics or GPS data. These views are a live view, direct or indirect of the physical world. In simpler terms, it is a view of the world modified with computers. Usually augmented reality involves real time processing, where the users can interact and manipulate digitally the information about the real word. It works by overlaying the real world with artificial information or objects. [13]

According to the Journal On the use of Augmented Reality technique in Learning and Interpretation of Cardiologic Data by Edgar Lamounier, Jr. Arthur Bucioli, Alexandre Cardoso, Adriano Andrade, and Alcimar Soares, they stated that augmented reality is a technology which provides people with more intuitive ways of interaction and visualization, close to those in the real world.[14] Thus the application for augmented reality has grown tremendously, from the fields of medicine, training, entertainment and even educations. Other than that, Hannes Kaufmen in his paper collaborative Augmented Reality in Eduaction, he stated that using augmented reality can contribute to raise interest and motivations in students with high potential to enhance learning experience.

2.2.2 Augmented Reality in Education

If augmented reality were to be applied in education, there are a few things that we must consider. Those things are how the technology should be implementing in teachings, whether there will be change in the current teaching style and how to integrate these technologies in the teaching process. Other than that, we must think of the platform that this technology should use. For example, does this system of learning requires an entirely new set up for its platform or we can integrate it with the current usage of laptops for educations Hannes Kaufmen in his paper collaborative Augmented Reality in Eduaction he also stated that one of the most important purposes in educational environments is to promote social interaction among users located in the same physical space.[15] In term of augmented reality, it is important for users to share a digital space populated with virtual objects. He stated that this technique is very effective since multiple users can communicate directly while interacting with the same virtual objects.

Figure 3: Augmented Reality in Education

2.2.3 Augmented Reality in Learning Anatomy

Augmented Reality versatility has leads to many projects that use Augmented Reality as their main focuses. Nowadays, there are already projects that use Augmented Reality to teach Anatomy. There are also papers written on these subjects. One of those papers is written by Chien-Huan Chien, Chien-Hsu Chen and Tay-Sheng Jeng, and their paper is called 'An Interactive Augmented Reality System for Learning Anatomy Structure' [21]. In their paper, they stated that, by having this AR to help in the studies, students will be able to understand easily what they have learned. This is because with the help of Augmented Reality and pop-up style labelling, the students can manipulate the 3D model and look at it from different angle. This allows the students to have a better view on where the parts of the human body are located [21].

Another example of paper about Augmented Reality in Anatomy learning is written by Soon-ja Yeom. The Paper is called "Augmented Reality for Learning Anatomy". In his paper, he stated that learning Anatomy is a hard thing to do. The contemporary learning method for anatomy is inadequate because they lack the depiction of 3D spatial tissue in a three dimensional manner. This is because most of the explanations are in terms of diagrams and pictures. All of these are in 2D [22].

Figure 4 : Difference in learning From 2D and 3D

As we can see above, learning using 3D images is better than learning from 2D images. The students can manipulate the images and view it from different angle, giving the students better perspective and thus improve their understanding in anatomy. 2D images although widely used, it is unable to provide the best view and depiction of human anatomy to students. Soon-ja Yeom also stated in his paper that, another common learning method for human anatomy is using the cadaver dissection. Although this method gives the best views and depiction of human anatomy as they dissect a real human body, this method might not be the best for learning. He stated that this is because to use this method, there are a lot of costs are involved on top of very strict procedures to follow. To obtain a dead human body to for learning purposes are not as easy one might think. To make it worse, this type of anatomy learning method lack the ability to be used more than a few times [22]. This is different in the case of Augmented Reality where the cost is only for developing, purchasing and maintaining the software, which is much lower than the cost to obtain a dead human body for dissections.

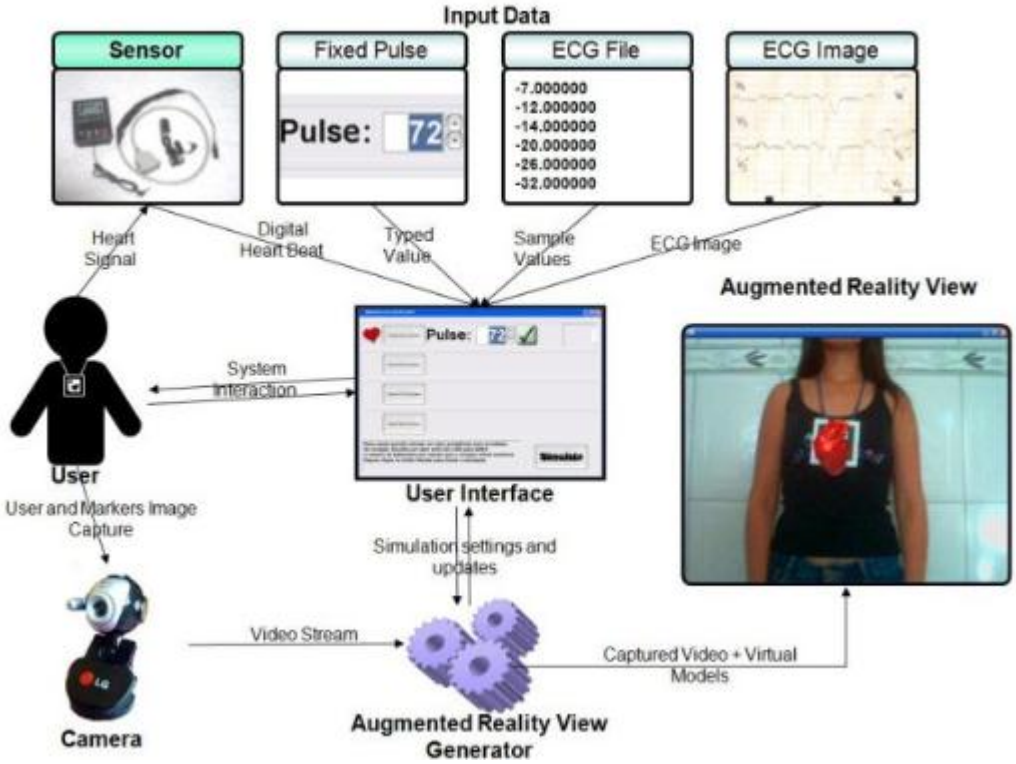


Figure 5: Proposed Architecture

The project above is another sample on the implementation of augmented reality on human anatomy is from Edgar Lamounier, Jr., Arthur Bucioli, Alexanre Cardoso, Adriano Andrade and Alcimar Soares. They wrote a paper on their project, and their paper is called 'On the Use of Augmented Reality Technique in Learning and Interpretation of Cardiologic Data'. Their project is about using the augmented reality and sensors to simulate or give live views on a subject heart beats [23]. This projects works where a sensors a placed on the subject to capture the real time data input on the cardiac pulsation of the subject, and this data are constantly updating the virtual environment. Then a camera is used to capture visual information of the real world in video format. The third part is the ECG file, that is a file containing the amplitude values of electrocardiogram graph with fixed visualization rate. The fourth part is the ECG image; it is images of recorded real cardiac exam inline graph. Finally, the Augmented Reality Visualization Generator. This AR generator is the one that is responsible for generating the real world scenes captured by the camera and the virtual objects (heart an graph) [23]. This camera uses monochromatic market to place the 3D images on.

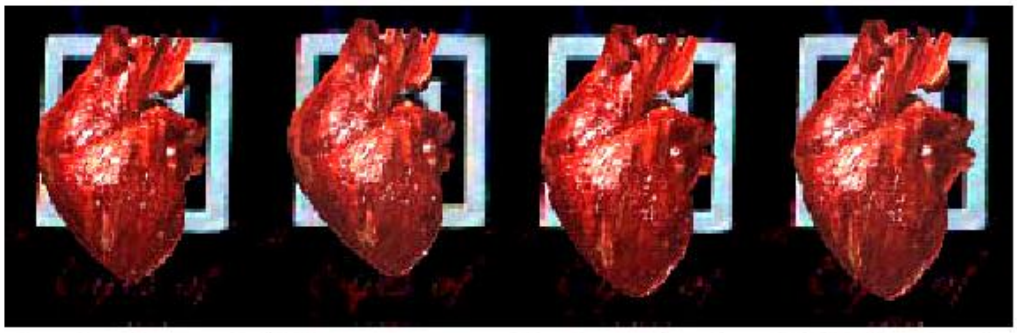


Figure 6: Different heart beat images simulated by the system

2.3 Kinect

Kinect is a new motion sensing technology from Microsoft Xbox 360. It is created to introduce a new style of gaming to the public. Even though motion sensing technology has been around for quite a while, it was never an easy access to the public and is usually costly. With the current development in the gaming industries, these technologies are now available to the public at a much lower cost. Kinect works by using 2 sets of camera, one camera will detect the depth of the surrounding and the other camera will detect the user's movements. Combining both, Kinect are able to map the gestures and movement of the users.[16] What makes Kinect different from other motion sensing technologies is that Kinect encourages the public to mod and modified Kinect's code for other purposes. This has spark new interest in the public and has opened up a lot of new fields where Kinect can be implemented. One of those fields is the field of educations.

2.3.2 Advantages

In our project, the implementation of Kinect in education requires a few adjustments in terms of the settings of the classroom, the practice in the classroom and the method of teachings. The teachers are most likely going to change their teaching style to implement and integrate these new technologies in their teaching process. To ensure that the implementations of the project are successful, teachers are going to be trained and taught on how to use the Kinect first. These topics are also being discussed by Hui-mei Justina Hsu. In her paper, she discussed these problems in the chapter of Affordance. She questioned on how the Kinect will be implementing, and will Kinect bring change on the method of teaching that has been practiced for so long. She stated that, even if we were to implement new technologies in education but if we did not change our teaching style and method, it won't bring any change. [6] This has happen before in when ICT resources are introduced to schools. Little impact if none at all are being made. The cause of this is mainly as stated above. Even with the new technology, the teachers are not changing their

teaching style, thus, the access to new technologies are not being fully utilized. But, in terms of Kinect the possibilities are better. Mainly because using Kinect in education is a unique and creative innovation. Its Kinestatic features and gesture-based interaction will surely motivate educators to devote themselves to use Kinect accordingly in class and fully utilize its capabilities.

There are a lot of benefits in implementing Kinect in education. In this project, the aim is to increase participation of students and allowing them to have more interactive learning in the science subject. Through this interactivity, the goal is to gain the students interest, thus making them able to learn better and have more interest in the subject. It has always been known that interactivity plays an important part in learning. Interactivity here means how much can the teachers control the classroom and have the attention of their students. The matter of interactivity and how it affects the process of learning in the classroom are also being discussed by Hui-mei Justina Hsu. In her paper, she stated that traditionally, the process of teaching in class involved a long talk from the teachers and occasionally a little feedback from the students. [6] This is true and this is what is being practiced right now in most schools. As we can see here, in terms of interactivity, it is very low. Students are rarely involved in the learning process other than listens to the teacher's talk. This situation has created a boring and dull learning environment where students lose interest in the subject and find the subject hard to understand and master. She stated that interactive classroom learning should consist of four elements, reciprocal opportunities to talk and participate, appropriates guidance and modelling, environment for participation, and finally an increase in students autonomy. Having Kinect as a learning tool could bring all these 4 elements together [4].

Since Kinect are one of the most advanced motion sensing technologies available to the public in the market, one might think that the costs of implementing this system are going to be expensive. But this is not the case, the price for the Kinect itself would only cost from RM 400 to RM 600 depending on the types. As most schools are already implementing the usage of multimedia at schools which

implement usage of laptops and projectors which cost thousands, the implementation of Kinect can be considered as a cheap add-on. The costs for implementing this system are cheap but the advantages and the change that it will provide in the learning system are huge. The schools and teachers can simply integrate this new Kinect system with the available system. Other than that, there are already software developed to allow users to control the computer desktop using motions and gestures. This will allow greater flexibility for teachers in class. In other words, the implementation of Kinect in class is very feasible and useful compared with its cost.

2.3.3 Disadvantages

According to Hui-mei Justina Hsu, in her paper "The Potential of Kinect in Education", she stated that by using Kinect in a class room, teachers will be able to create an enjoyable, interesting interaction types, boost student's motivation and will promote learning itself. However she also stated that the implementations of Kinect in a classroom have its own limitations. The limitations that she mentions are Kinect needs a large classroom space, lack of easy-to-use development tools, and long calibration time and pedagogical constrains such as the difficulties in shifting to kinaesthetic pedagogical practices and limited understanding of its effects. Other than that, one of the main constrains of using Kinects is that Kinect can't work as a stand-alone system. [4]. To implement Kinect, we will need to sync it with computers that run the software. This is true for our project as well, there are certain limitations that we must face, but the benefits that the learning institutes can obtain by implementing this new style of learning are worth the investment. Today's children are constantly looking for something new and fun to explore and learn, but the methods used in schools are always boring and are not interesting. We need something unique that will pique the interest of the students and gaining their attentions in class

Other than that, the obvious disadvantage of Kinect is that the accuracy of its technology. Although Kinect can be considered as advance motion sensing

equipment, it is still far from perfect. The stereo optics used in Kinect is fairly old. Not only is that, the level of accuracy that Kinect can detect still low. For example, Kinect will encounter errors when an object stands between the users and the system. This is largely to the weakness of the skeletal system that Kinect implements. Kinect are supposed to map user's body with this skeletal system and the skeletal will move according to user's gestures. But if there is an obstacle or other object between Kinect and users, the system will mistake that the object are part of the skeletal system as well. This will gives incorrect and inaccurate input to the system.

Kinect was originally developed for gaming. Usually a local gaming system at home will focuses on one or two players at most. As Kinect was initially developed as an add-ons to the Xbox 360 console, Kinect was developed with this concept in mind as well. So, mainly it focuses on single player and at most, the system right not are being developed to only allow two users at a time. This limitation has posed another problem when we are planning to implement the system in the classroom. In a classroom, generally there are about 30 to 40 students per class. So obviously these systems are not build for this huge number of participant. So the challenge is how to manage and coordinate so that everyone in the classroom can experience and benefit from the implementation of the system. One of the possible methods is having the teachers to have centralized control and have a few numbers of students to participate at a time per activity. This technique should be carried on through the subject syllabus and by the end of the subject syllabus, all students have already participated [7].

2.3.4 Kinect in Robotic Education

Kinect is a versatile technology; it can be used and implemented for various purposes. In terms of educations, the applications of Kinect are not restricted by certain group of age or level of educations. Michal Tölgyessy and Peter Hubinský have proved this by implementing Kinect in Robotics Educations. They wrote a paper on this subject called The Kinect Sensor in Robotics Educations. This paper focuses on the field of educations of robotic sensing abilities. Since the papers are called The Kinect Sensor in Robotic Educations, it focuses more on Kinect as a sensor device for robotics. It focuses on what Kinect can do and what the students can learn by using Kinect.

In the paper, they stated that there are a lot of applications in which Kinect can be implements in the field of Robotics.

a) Data Fusion: Kinect consist of a few parts, that are the RGB camera, 3D depth sensing system, Multi-Array Microphone, and Motorized tilt. Since Kinect are equipped with RGB camera, it is very useful for robotic visions. By implementing and use Kinect, students can learn to fuse different data. The RGB information can be converted to any commonly used colour space, such as Normalized RGB, HIS, HSV, HSL, TSL, YCbCr, CIELAB or CIELUV. These entire colour space can be used for different task for Robotics visions, thus, this information can be used to create a detection and segmentation programs.
b) Obstacle Avoidance and Collision Detection: As Kinect are also equipped with 3D depth sensing technology, applications can also be created to serve these purposes. Kinect ability to provide depth map with good resolution make it a suitable technology for obstacle avoidance and collision detection.
c) Gesture Control: Kinect ability to provide Gesture control does not limited to in games only; this ability can also be extended in Robotics fields. It can be used to enhanced and improve Human-Robot

Interaction (HRI). For example, a robotics system that synchronized with the movement of the users.[2]

2.3.5 Kinect in English Language Learning

In the Research paper: Using the Xbox Kinect in Foundation Phase English Language Acquisition by Peter de Lisle, they try to implement learning using Kinect in a rural primary schools in South Africa. The aim of this program is that they try to test the hypothesis that English language skills can be improve through exposure to English while playing games using Xbox and Kinect technology.[8] The teachers at these schools tried to create an interactive environment that can help students and encourages them to explore concepts and skills in English. They need to make sure that the students participate in the learning actively by creating activates that are fun, relevant and challenging to the students.

So far, in terms of using games for education purposes, we can divide it into categories.

- Computer as a teacher :
  - The basic idea for this concept is that the computers or a game provides everything that students need to learn. The software will provide the materials, activities and keep track on the student's performance and achievements.
- Computer as Creator of Context :
  - This is more practical way to learn as the program from Kinect or Computers does not dictates everything. Instead it relies on the skills of the teachers to create immersive experience that simulates real life using Kinect as the tools. It will become a tool for the teachers to encourage and create interactive environment for the students to explore what have been learned in class.

As the students in the program are very poor in their mastery of English, the programs/games used must be suitable for beginners in English. The games must serve simple and basic purposes. There are a few ways that Kinect can be used as learning tools for the students.

1. Use it as a reward: e.g., rewards for good behaviour.
2. Create a communication gap : have the class to instruct a blind folded players how to progress in games using simple English
3. Expand on the game: e.g., when certain games are finished, ask the class to write about the game, and what have they learn from the game in English.

There are many other ways that Kinect can help in improving Basic English at schools. The important things are that teachers took part in the learning process and guide students toward the right directions.

2.3.6 Kinect in Science

Being a versatile technology, the application of Kinect can be used an implement in many areas. The technology is so user friendly and versatile has enables them to suits almost any purposes in any fields. The fields of science are also beginning to look into new possibilities in Kinect. In 2011, SEIMENS has conducted a competition called SEIMENS Competition in Math, Science & Technology. The aim of this competition is to explore the new possibilities in Math, Science & Technologies and recognize young talents and fosters growth in them. One of the winners for 2011 is using Kinect as their project. Ziyuan Liu and Cassee Cain uses Kinect and Computer Vision to create a program that analyze Human walking pattern. This technology will then be used to help to prescribe treatment to those who suffers from injuries or ailment that effects movements, people who need amputations and hand gone through joint replacement surgery. The team was awarded $100,000 for their projects. [9]

2.4 Conclusion

Another study from Michal Tolgyessy and Peter Hubinsky on their paper, "The Kinect Sensor in Robotics Education", they suggested that the application of Kinect in educations should be divided into two areas. The first one focuses on standard practical classes and the second one consists of large projects, such as bachelor's and diploma projects. In my project, the project will focused on primarily on the science subject taught in schools for primary schools students. For this project, a few chapters will be chosen from the syllabus of the science subject taught in schools. A program will be created based on that chapter to facilitate the learning process of the students in the class. The program will allow the students to interact and play around while learning. This will make learning more fun and interesting for the students. The system will be more as a tool for the teacher to use and create and interactive environment rather than relying on the system alone for the teaching process. Further explanation about how the system would be will be explained in chapter 3.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In this chapter, the methodology on how the system will work will be explained. This will explain in further details on how the project will be done. The project will involve the learning of human body parts or anatomy for pre-schools students, but we will not take the whole syllabus of the subject. Only a few parts will be taken and developed as a prove of concept. Then a system/program will be created based on the chapters chosen. To ensure the projects runs smoothly and this project is evaluated based on System Development Life Cycle (SDLC). But this project focuses more on Planning, Implementing and Analysis.

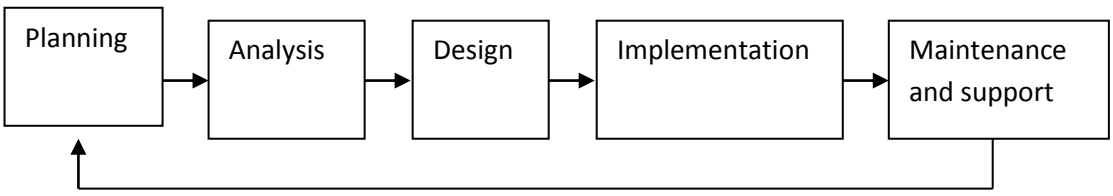


Figure 7: SDLC Phase

But for this project, it only focuses on Planning, Implementing, and Analysis.

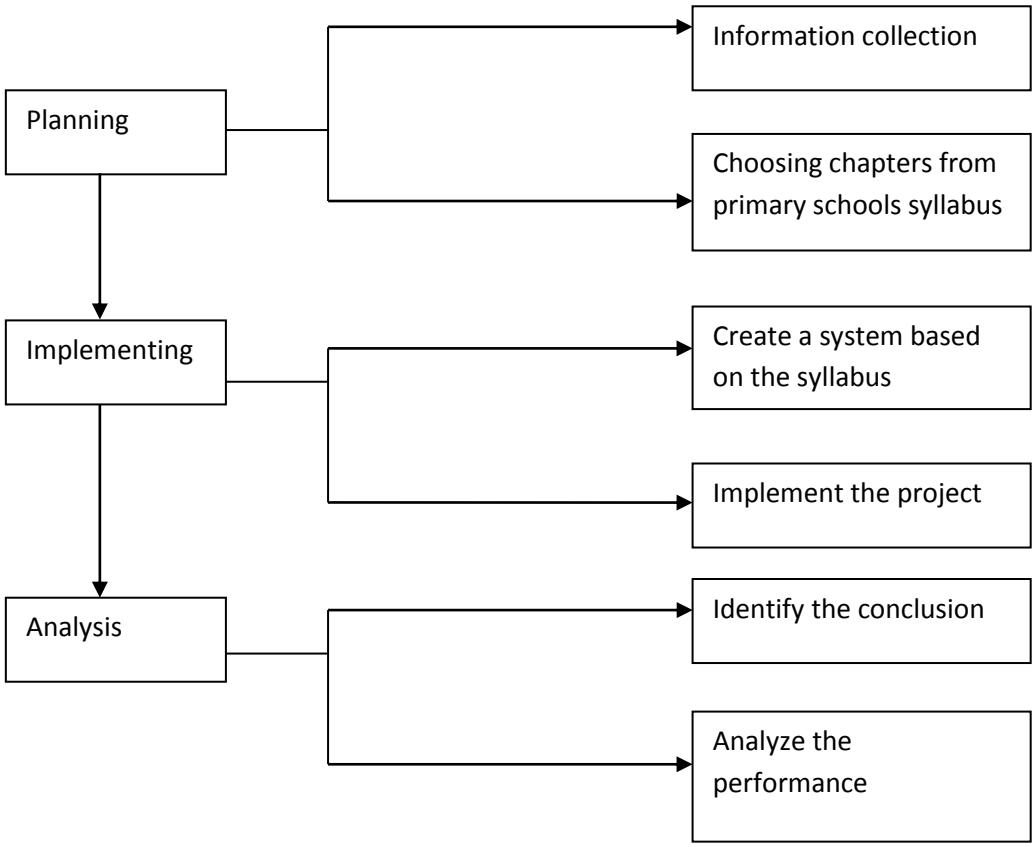


Figure 8: Steps of Methodology

3.2 Planning

As the diagram above the planning phase will be in two parts, which is gathering information and choosing a few parts of the syllabus will be taken and developed into a program. The reason only a few chapters from the syllabus will be chosen for the project is because of the time constrains.

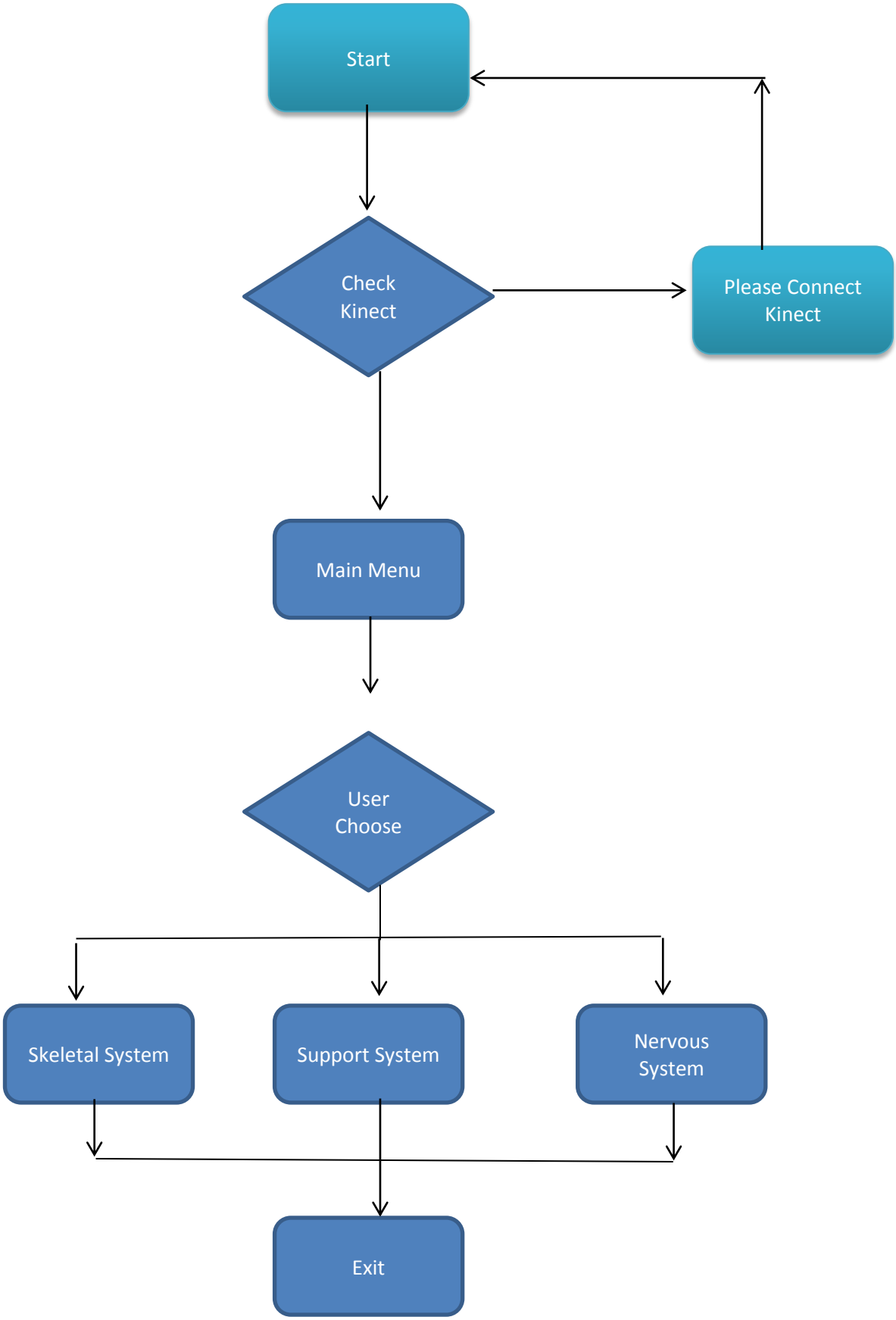
3.2.1 Information Collection

Since the project will use Kinect, the project will be developed using Microsoft Visual Basic 2010. This software was chosen because I already familiar with the language used. Other than that, Microsoft Visual Basic 2010 has one of the best and user friendly interfaces compared to other programming language. Other than that, a few information gathering will be done on the science subject syllabus taught at schools.

3.2.2 Choosing Chapters from School's Schools Syllabus

The reason only a few chapters from the pre-schools syllabus will be chosen for the project is because of the time constrains. So far, we plan to do only the head parts, and the whole body.

3.3 Flow Chart 1



3.4 Implementing

For the implementing parts, it will also be divided into two parts. The first part is designing the systems, and the second part refers to the Implementation of the systems. The first part is to create based on the chapter chosen for this project, the second part is to implement the system. Throw away prototyping are a process where we create a prototype and then throw it away rather than be part of the final project. This technique of development are very useful because we will have clearer views of the needs and requirement by coming up with a working prototype, we can also see right away which area needs improvement or adjustment. These types of development process save times. The steps involve in Throw-Away-Prototyping are:

1. Write preliminary requirements
2. Design the prototype
3. User experience using the prototype. Specify new requirement and area of improvement
4. Repeat if necessary
5. Write and made the final design and requirement
6. Develop the real final project

This is the initial planning on how the project will looked like before any changes to the topics and the project was made.

Figure 9: Sample start page

Figure 10: Second user interface

Figure 11: Example for human skeletal system

Figure 12: Example of human muscles

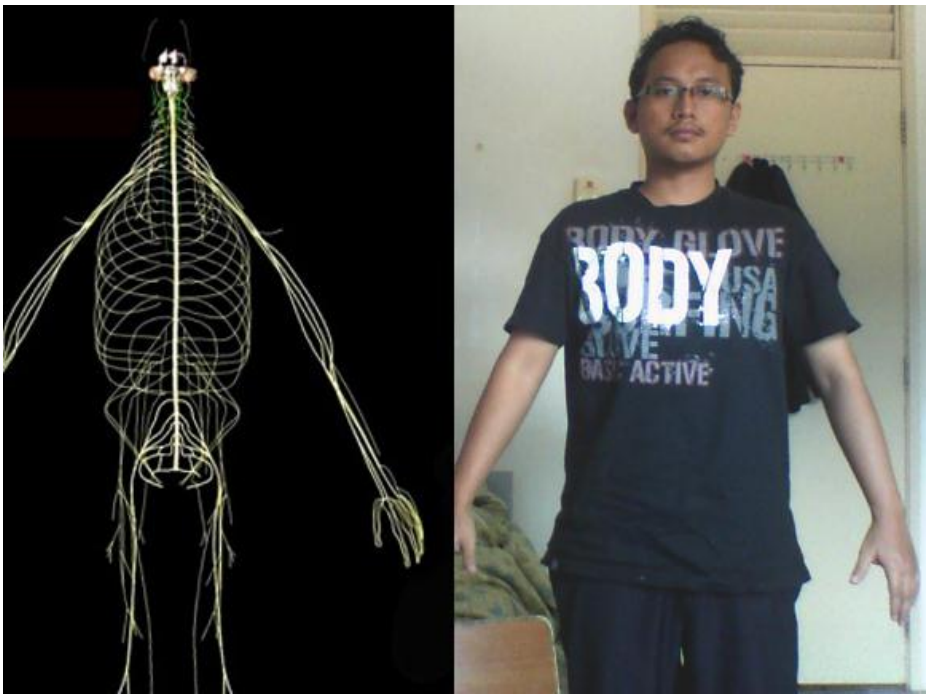


Figure 13: Example for human nervous system

For the second part of the Implementing phase, that is the actual implementation of the system. The system will be a part of the learning process in class, for example, when the teachers are teaching about the human skeletal system, he/she can use the Kinect system to give better views to the students. The teachers can also ask students to use and try the system themselves and see how our body works. These way students will have clearer view on the topics taught. At first, the plan for this project is to use 3D model and map the skeleton of the user through Kinect technology to the 3D model. But with the limited resources and time available to complete this project, only images are being used instead. As the project progresses, a few changes was made. Instead of going on more serious theme, we choose the project to be more fun as a theme. The project was redesigned to fit and be more suitable for pre-schoolers. We include more pictures and made the project simpler as to make it easier for the pre-schoolers to operate and interact with the system.

3.5 Analysis

After the project is done, the project will be analyzed in terms of its performance. How reliable the systems. Finally, improvement will be made so that the project will work efficiently.

3.5 Gant Chart and Key Milestone

Project	Project Activities	Week No											
		1	2	3	4	5	6	7	8	9	10	11	12
FYP1	Planning												
	Project Proposal												
	Information Gathering												
	Extended Proposal												
	Materials and Tools Acquisition												
	VIVA												
	Interim Report												

Table 1: Gant Chart Project Progress

3.6 Tools

1. Kinect
2. Visual Basic 2010
3. Kinect SDK

CHAPTER 4

RESULT

4.1 Result

This is simple methods that are being created to check whether the Kinect is properly connected or not. The program only works when the users unplug the Kinect from the usb port, the power are not turned on or when the Kinect is initializing. So it is basically a basic program to make sure that your systems are being setup correctly.

Figure 14: When Kinect are not plug in into the USB port

Figure 15: Kinect is Initializing

Figure 16: Initialization successful

Figure 17: The power are not turned on

These simple methods are being placed for each form of the complete system. So whenever there is an error in which the Kinect is unplugged, or not turned on, we will know.

Figure 18: The Main Menu

This is the main menu of the project after adjustment to the title and the scope has been made. Right now the project focuses on the Pre-Schoolers to teach them about the human body parts. That is the body parts of the head such as the eyes, nose and ears and the parts of their body such as their legs, hands and body. As we can see in the picture above, the component of the main menu consist of the depth image of the user in the bottom left of the menu, the Kinect angle adjuster on the bottom and selections for the students to choose on the right. For Kinect depth image, the user will be able to know whether they are in the right distance to interact with Kinect and use it. As we can see in the image above, the user depth image at the bottom left is being made brighter than the grey surrounding. This means that the system recognizes that there is a user and it maps the user with colour to differentiate the user from the surrounding. Another component of the main menu is the Kinect angle adjuster. It can be used to adjust the angel of the Kinect. The value ranges from -27 being the lowest, and 27 being the highest.

Figure 19: Mix and Match the face

This is the window that pops up when the user select the Head component on the main menu. This page is like a Mix and Match game, where the user needs to match the pictures with their spellings. It is a simple game like the exercise that pre-schoolers used to do. The difference is that, right now for this kind of exercises, teachers are using the exercise books but this project uses Kinect.

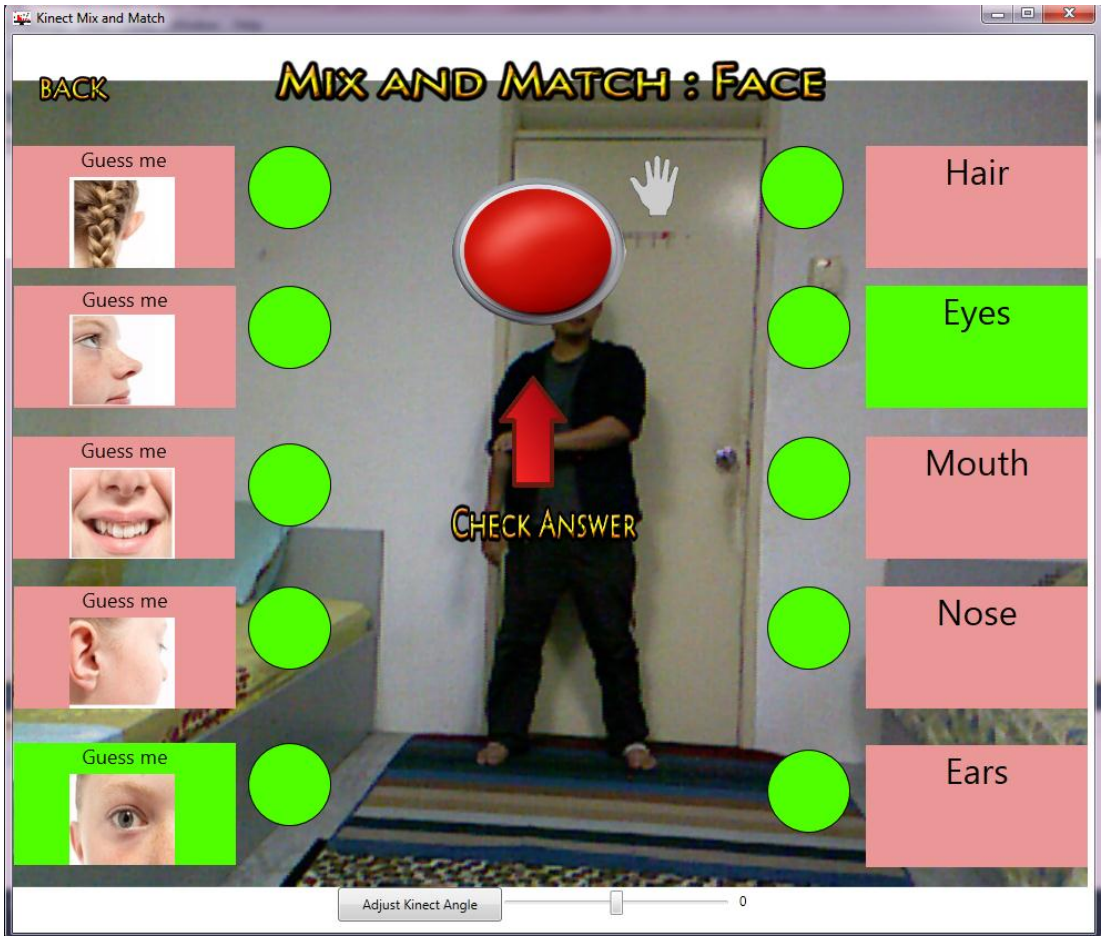


Figure 20: Mix and Match the Face Correct Pair

The program works where the user needs to select a pair. A picture and a word to paired it with. User simple needs to hovers their hand on the green button at the side of both pictures and their key words. After the user has selected a pair, he/she can check it by hovering their hand on the big red button. If the user manage to select the right pair of picture and key word, it will changes colour as the image above. If not, everything will turn back to its default state.

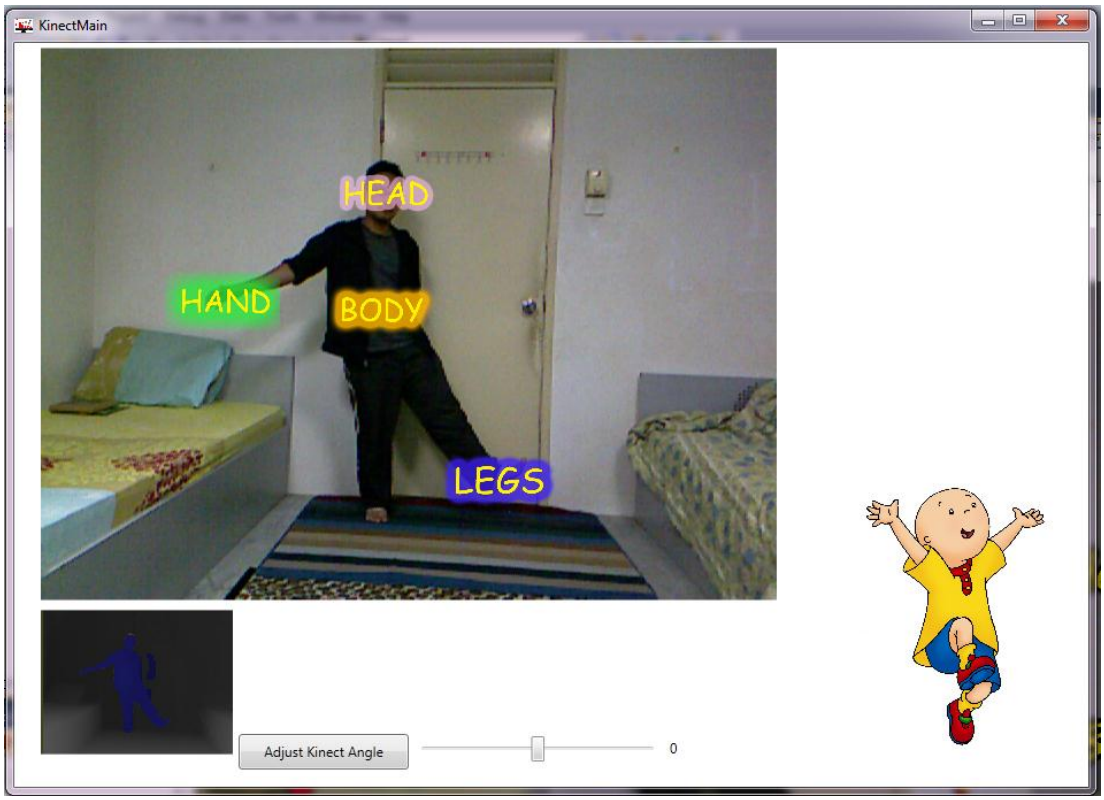
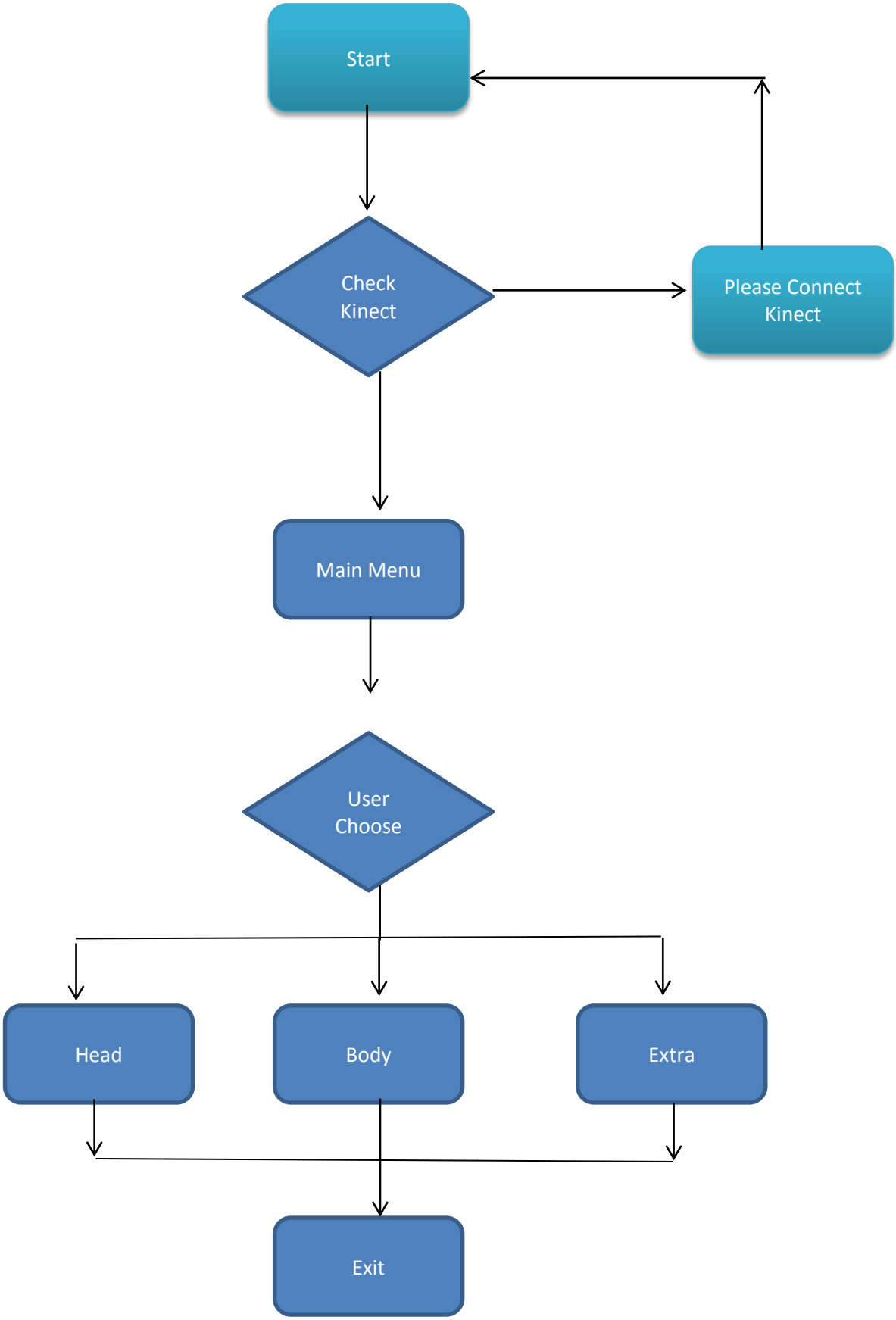


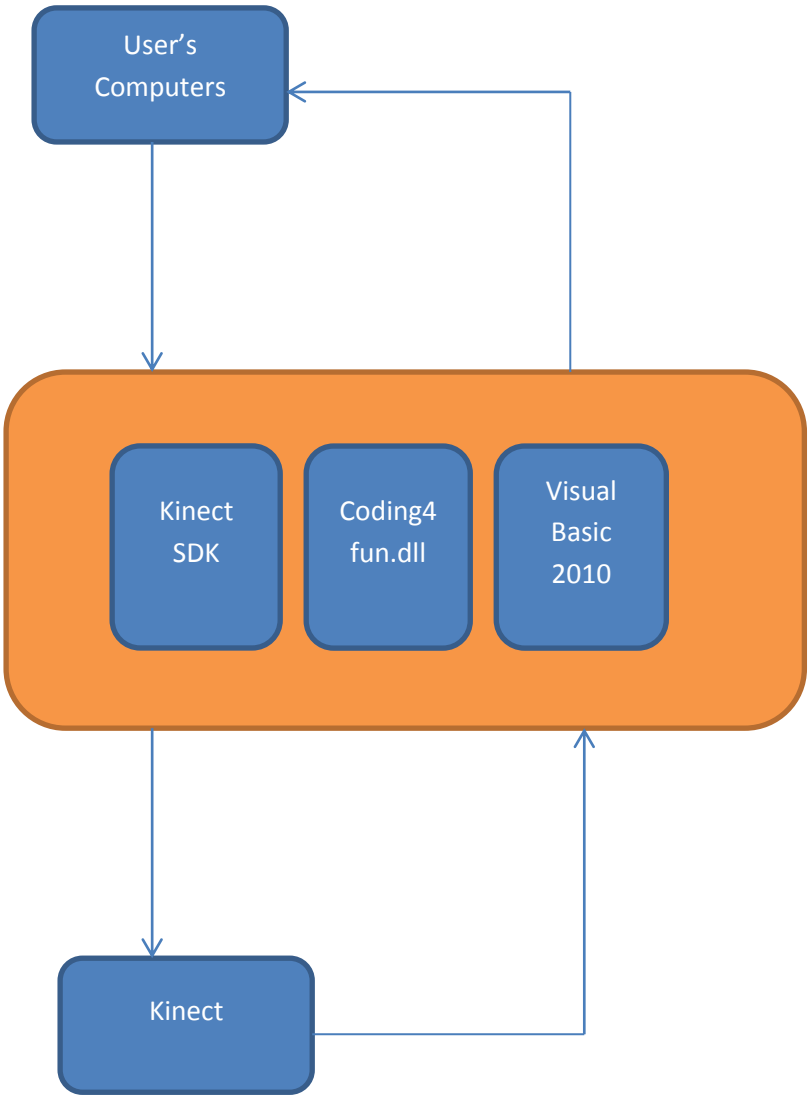
Figure 21: Body Menu

This is the window that opens when the user select the body component from the main menu. At this page, the aim of this window is to map the user's body with its keyword. It can help the pre-schoolers to know which of their body parts are called what. As the screen shots above show, when the user moves around the key word also moves around. So it is a fun way to learn as you can see how you can interact with the system.

4.2 Flow Chart 2



4.3 System Architecture



4.4 Challenges and Solution

Working with Kinect has many challenges as this is a new technology. The resources available to use and to refer to are very low. Since this is a relatively new technology, there are not many tutorials that can be used to learn on how to code programs for Kinect, and to make it worst most of the tutorials available are made for the beta SDK release for Kinect. This means that we can no longer uses this tutorial and resources as right now Kinect SDK are already in its 1.5 version which is totally different from the Beta release. Initially this project was supposed to include many 3D objects and make the object can be manipulated by the users. But since there are no resources that teaches how it is being done, the idea had to be abandoned. The project idea was changed to make it simpler so that it can be done within limited resources and time. Instead of using 3D models, the project will use images instead.

4.5 Field Testing

To further prove that my project is suitable for young children, I try to implement field testing. In this test, I find a few subjects to use the project and try to get their feedback. The feedbacks that I get from the research are quite promising. They seems to really have fun with the system. Having said that, the system still needs some improvements. The first thing I find during the field test is that the system cannot work on outdoors environment. This is mainly due to the limitation of Kinect itself and not the program. What happens during outdoors usage is that the glare and as well as the infrared from the sun are too much. This hinders Kinect from functioning properly and the Kinect itself cannot distinguish users from the environments. As a result, the program doesn't work at all so the system can only be used for indoors purposes.

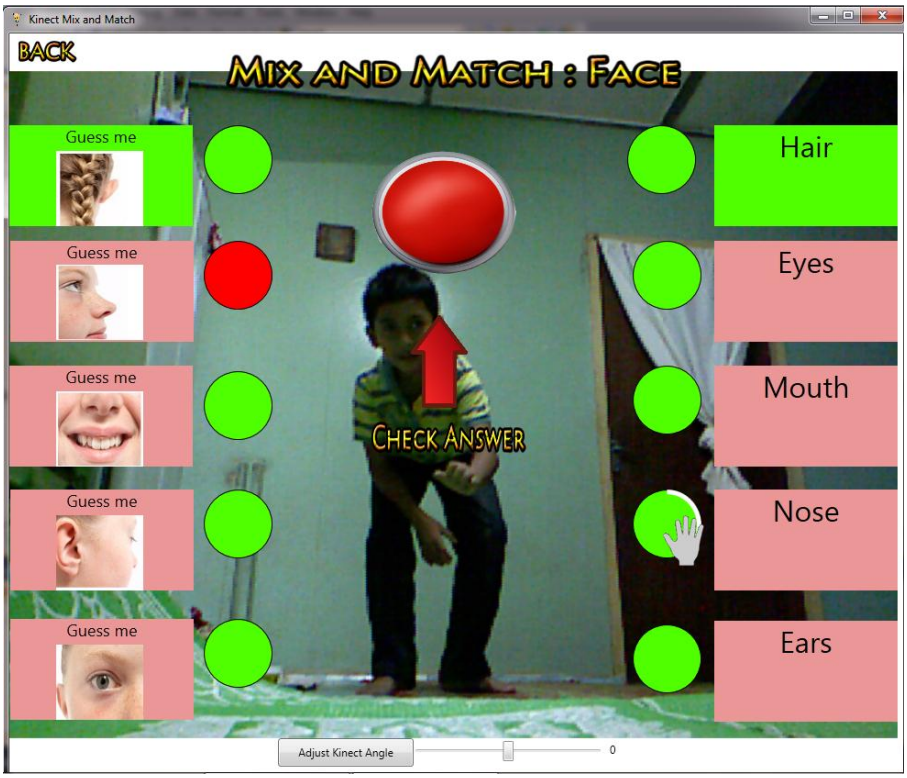


Figure 22: Sample subject using the Quiz in the program

Another finding that I found is that kids are really enjoying the Kinestatic interface where they can use their whole body as a medium to manipulate and use the system. But as the system is only on its initial stage, the resources or materials in the system are too little. As a result, they can easily and quickly finish all of the resources provided in the system. So improvement can be made by adding more resources to the system and make it more comprehensive.

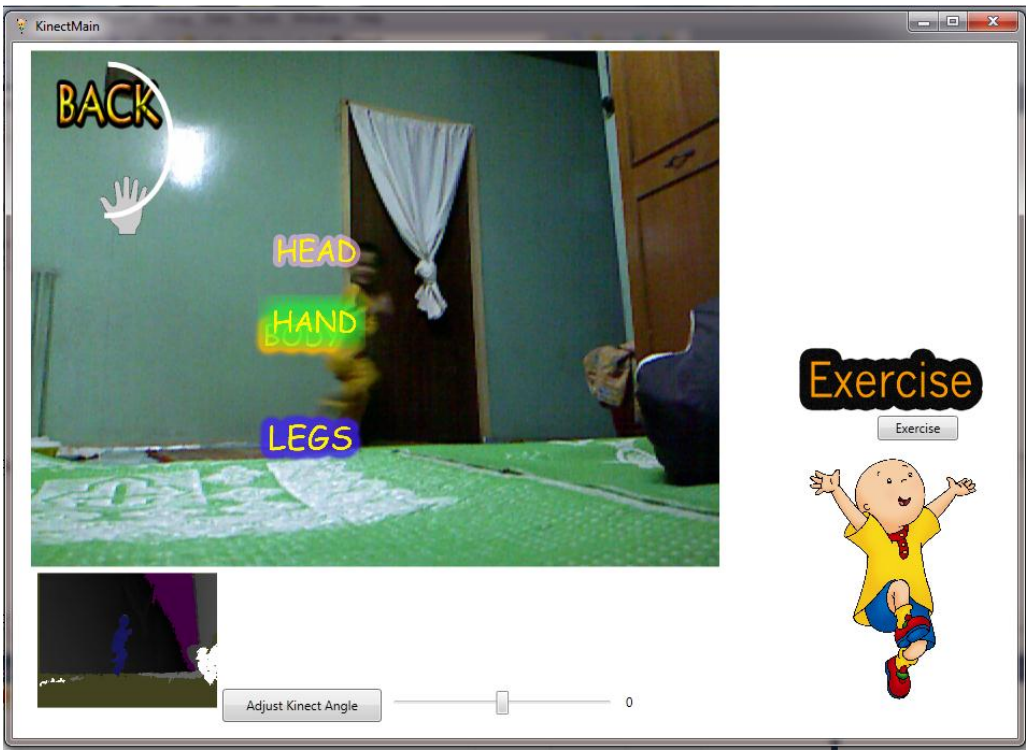


Figure 23: Sample subject having fun with Kinestatic features of the system

CHAPTER 5

CONCLUSION

5.1 Conclusion

As a conclusion, this project is about developing a new learning system for learning basic anatomy for pre-schools schools. The aim is to facilitate learning by making the learning process more fun and interactive, so that the students will be more interested in the subject and focused more in class. From the research that I made, I found that I have met the objectives of my project. I had managed to introduce a new form of learning for kids by developing a Kinect based system. Through this system, I had managed to make learning more fun for Kids and then I also manage to conduct fields test to see their acceptance of the technology. The system is easy to use and to learn but at the same time it is fun for the children. Thus it makes them able to concentrate more. This is because they feel like they are playing a game instead of learning, making it easier for them to understand and grasp what had been taught.

5.2 Future Work

Since this project only covers the simple syllabus that is being used for pre-schoolers, there are many rooms for more improvements. This project will only serve as a prove of concept or as a tool to measures how effective learning with Kinect will be and how are people able to adapt with the technology. For the future work, what can be done is that, we can increase more content to the project, for examples include more exercises, quizzes and we can create a scoring board in the system so that the students as well as the teachers can see how well the students are doing. If this project were successful, this concept can be further enhanced and be used to teach other subjects as well. These subjects can be for the pre-schoolers even to college students.

REFERENCE :

[1] Wikipedia, 2012. Retrieved March 1, 2012 from Wikipedia website <http://en.wikipedia.org/wiki/Kinect>

[2] Michal Tölgyessy, Peter Hubinský (2011). The Kinect Sensor In Robotic Education. Retrive 1 march 2012, from http://scholar.google.com.my

[3] Microsoft, 2012. Retrieved March 1, 2012 from Microsoft website <http://www.microsoft.com/en-us/kinectforwindows/.>

[4] Wikipedia, 2012. Retrieve March 27, 2012 from Wikipedia website <http://en.wikipedia.org/wiki/Interactive_Learning.>

[5] Alferd Dork. University of California-irvine. Interactive Learning.

[6] Hue Mei Justina Hsu (2011). The Potential of Kinect In Education. Retrive 1 march 2012, from http://scholar.google.com.my

[7] Mark Evans (2011). University of North Texas. Gestural Interface Learning. Retrive 1 march 2012, from http://scholar.google.com.my

[8] Peter de Lisle(2010). SchoolNet South Africa. Research: Using the Xbox Kinect in Foundation Phase English Language Acquisition.

[9] Seimens, 2011. Retrieve March 27, 2012 from Seimens website <http://inr.synapticdigital.com/Siemens/Competition2011/.>

[10] Percepsys, 2011. Retrieve March 27, 2012 from Percepsys website

< http://percepsys.com/contentdev.htm>

[11] Joel Voss, Brian Gonsalves, Kara Federmeier, Daniel Tranel, and Neal Cohen (2011). Hippocampal Brain-Network Coordination During Volitional Exploratory Behaviour Enhances Learning.

[12] News Bureau, Illinois, 2010. Retrieve March 27, 2012 from News Bureau website

<http://news.illinois.edu/news/10/1206brain_neal_cohen.html.>

[13] Wikipedia, 2011. Retrieve March 27, 2012 from Wikipedia website

<http://en.wikipedia.org/wiki/Augmented_reality.>

[14] Edgard Lamounier, Jr., Arthur Bucioli, Alexandre Cardoso, Adriano Andrade and Alcimar Soares (2010). On the Use of Augmented Reality Technique in Learning and Interpretation of Cardiologic Data.

[15] Hannes Kaufmann. Institute of Software Technology and Interactive Systems Vienna University of Technology 2. Collaborative Augmented Reality in Education.

[16] Kinect, 2012. Retrieve March 27, 2012 from Xbox website

<http://www.xbox.com/en-US/Kinect.>

[17] Education and Technology, 2012. Retrieve March 26, 2012 from Blogspot

<http://cfrehlichteach.blogspot.com/2011/11/power-of-interactivity-and-engagement.html.>

[18] Kinect, 2012. Retrieve March 27, 2012 from openkinect.org

<http://cfrehlichteach.blogspot.com/2011/11/power-of-interactivity-and-engagement.html.>

[19] Kinect, 2012. Retrieve 25, 2012 from Kinect.me

<http://www.kinect.me/.>

[20] Kinect, 2012. Retrive 24, 2012 from endgadjet.com

<http://www.engadget.com/2010/11/04/kinect-for-xbox-360-review/.>

[21] Chien-Huan Chien, Chien-Hsu Chen and Tay-Sheng Jeng (2010).

'An Interactive Augmented Reality System for Learning Anatomy Structure'

. [22] Soon-ja Yeom. School of Computing and Information Systems University of Tasmania (2010). 'Augmented Reality for Learning Anatomy'

[23] Edgar Lamounier, Jr., Arthur Bucioli, Alexanre Cardoso, Adriano Andrade and Alcimar Soares Jeng (2010). 'On the Use of Augmented Reality Technique in Learning and Interpretation of Cardiologic Data'.

Appendices:

4.3 Code

Main Window XAML Code

```
<Window x:Class="KinectQuiz.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="KinectMain" Height="840" Width="921" Loaded="Window_Loaded"
        xmlns:Controls="clr-
namespace:Coding4Fun.Kinect.Wpf.Controls;assembly=Coding4Fun.Kinect.Wpf"
        xmlns:my="clr-
namespace:Microsoft.Samples.Kinect.WpfViewers;assembly=Microsoft.Samples.Kinec
t.WpfViewers"
        Closing="Window_Closing"
Icon="/KinectQuiz;component/Images/Caillou2.png">
    <Grid x:Name="theGrid">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="649*" />
            <ColumnDefinition Width="319*" />
            <ColumnDefinition Width="0*" />
        </Grid.ColumnDefinitions>
        <Canvas Background="Transparent" Height="803" VerticalAlignment="Top"
Grid.ColumnSpan="2" Margin="0,-2,0,0">
            <Image Canvas.Left="0" Canvas.Top="0" Height="797" Name="image6"
Stretch="Fill" Width="899" Source="/KinectQuiz;component/Images/cartoon-
illustration-vector-187093.jpg" />
            <Slider Canvas.Left="325" Canvas.Top="768" Height="29"
Name="slider1" Width="212" Maximum="27" Minimum="-27"
ValueChanged="slider1_ValueChanged" />
            <Label Canvas.Left="533" Canvas.Top="765" Content="0" Height="32"
Name="Degree" Width="30" />
            <Button Canvas.Left="179" Canvas.Top="766" Content="Adjust Kinect
Angle" Height="31" Name="btnangle" Width="148" Click="btnangle_Click" />
            <my:KinectSensorChooser Canvas.Left="239" Canvas.Top="247"
Name="kinectSensorChooser1" Width="396" Height="236" />
            <Ellipse Height="71" Width="73" Name="RightHand" Canvas.Left="100"
Canvas.Top="215">
                <Ellipse.Fill>
                    <ImageBrush
ImageSource="/KinectQuiz;component/Images/hand.png" />
                </Ellipse.Fill>
            </Ellipse>
            <Image Canvas.Left="350" Canvas.Top="626" Height="107"
Name="image1" Stretch="Fill" Width="187"
Source="/KinectQuiz;component/Images/Start.png" />
            <Controls:HoverButton Canvas.Left="350" Canvas.Top="626"
Height="107" ImageSize="146" Name="hoverButton2" TimeInterval="1000"
Width="187" />
            <Image Canvas.Left="6" Canvas.Top="398" Height="255" Name="image4"
Stretch="Fill" Width="185" Source="/KinectQuiz;component/Images/Caillou2.png"
/>
```

```
            <Image Canvas.Left="192" Canvas.Top="0" Height="167" Name="image5"
Stretch="Fill" Width="539" Source="/KinectQuiz;component/Images/MainTitle.png"
/>
            <my:KinectDepthViewer Canvas.Left="6" Canvas.Top="659"
Name="kinectDepthViewer1" Height="138" Width="153" Kinect="{Binding
ElementName=kinectSensorChooser1, Path=Kinect}" />
        </Canvas>
    </Grid>
</Window>
```

Main Window Code

```
// (c) Copyright Microsoft Corporation.
// This source is subject to the Microsoft Public License (Ms-PL).
// Please see http://go.microsoft.com/fwlink/?LinkID=131993 for details.
// All other rights reserved.
//This project is created by Mohd Hazwan b Sahabuzan 12019
//but some of the code and resources are taken, modify and used from tutorial
websites
//such as http://channel9.msdn.com/Series/KinectQuickstart and
http://raychambers.wordpress.com

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.Media;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using Coding4Fun.Kinect.Wpf;
using Coding4Fun.Kinect.Wpf.Controls;
using System.IO;



namespace KinectQuiz
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {

        private static double _topBoundary;
        private static double _bottomBoundary;
        private static double _leftBoundary;
        private static double _rightBoundary;
```

```
        private static double _itemLeft;
        private static double _itemTop;


        public MainWindow()
        {
            InitializeComponent();

            hoverButton2.Click += new RoutedEventHandler(hoverButton2_Click);

        }



        bool closing = false;
        const int skeletonCount = 6;
        Skeleton[] allSkeletons = new Skeleton[skeletonCount];

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            kinectSensorChooser1.KinectSensorChanged += new
DependencyPropertyChangedEventHandler(kinectSensorChooser1_KinectSensorChanged
);


        }


        void hoverButton2_Click(object sender, RoutedEventArgs e)
        {
            Main1 Main = new Main1();
            Application.Current.Windows[0].Close();
            this.Hide();
            Main.Show();
        }



        void kinectSensorChooser1_KinectSensorChanged(object sender,
DependencyPropertyChangedEventArgs e)
        {
            KinectSensor old = (KinectSensor)e.OldValue;

            StopKinect(old);

            KinectSensor sensor = (KinectSensor)e.NewValue;

            if (sensor == null)
            {
                return;
            }



            var parameters = new TransformSmoothParameters
            {
                Smoothing = 0.3f,
                Correction = 0.0f,
                Prediction = 0.0f,
                JitterRadius = 1.0f,
```

```
                MaxDeviationRadius = 0.5f
            };
            sensor.SkeletonStream.Enable(parameters);

            //sensor.SkeletonStream.Enable();

            sensor.AllFramesReady += new
EventHandler<AllFramesReadyEventArgs>(sensor_AllFramesReady);

sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);

sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);

            try
            {
                sensor.Start();
            }
            catch (System.IO.IOException)
            {
                kinectSensorChooser1.AppConflictOccurred();
            }
        }

        void sensor_AllFramesReady(object sender, AllFramesReadyEventArgs e)
        {
            if (closing)
            {
                return;
            }

            //Get a skeleton
            Skeleton first = GetFirstSkeleton(e);

            if (first == null)
            {
                return;
            }

            GetCameraPoint(first, e);

            //set scaled position
            //ScalePosition(headImage, first.Joints[JointType.Head]);
            //ScalePosition(leftEllipse, first.Joints[JointType.HandLeft]);
            ScalePosition(RightHand, first.Joints[JointType.HandRight]);


        }

        void GetCameraPoint(Skeleton first, AllFramesReadyEventArgs e)
        {

            using (DepthImageFrame depth = e.OpenDepthImageFrame())
            {
                if (depth == null ||
                    kinectSensorChooser1.Kinect == null)
                {
                    return;
                }


                //Map a joint location to a point on the depth map
                //head
```

```
                DepthImagePoint headDepthPoint =
depth.MapFromSkeletonPoint(first.Joints[JointType.Head].Position);
                //left hand
                DepthImagePoint leftDepthPoint =
depth.MapFromSkeletonPoint(first.Joints[JointType.HandLeft].Position);
                //right hand
                DepthImagePoint rightDepthPoint =
depth.MapFromSkeletonPoint(first.Joints[JointType.HandRight].Position);


                //Map a depth point to a point on the color image
                //head
                ColorImagePoint headColorPoint =
                    depth.MapToColorImagePoint(headDepthPoint.X,
headDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //left hand
                ColorImagePoint leftColorPoint =
                    depth.MapToColorImagePoint(leftDepthPoint.X,
leftDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //right hand
                ColorImagePoint rightColorPoint =
                    depth.MapToColorImagePoint(rightDepthPoint.X,
rightDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);


                //Set location
                //CameraPosition(headImage, headColorPoint);
                //CameraPosition(leftEllipse, leftColorPoint);
                CameraPosition(RightHand, rightColorPoint);


                CheckButton(hoverButton2, RightHand);


            }
        }


        Skeleton GetFirstSkeleton(AllFramesReadyEventArgs e)
        {
            using (SkeletonFrame skeletonFrameData = e.OpenSkeletonFrame())
            {
                if (skeletonFrameData == null)
                {
                    return null;
                }


                skeletonFrameData.CopySkeletonDataTo(allSkeletons);

                //get the first tracked skeleton
                Skeleton first = (from s in allSkeletons
                                  where s.TrackingState ==
SkeletonTrackingState.Tracked
                                  select s).FirstOrDefault();
```

```
                return first;


            }
        }


        private void btnangle_Click(object sender, RoutedEventArgs e)
        {
            if (kinectSensorChooser1.Kinect.ElevationAngle !=
(int)slider1.Value)
            {
                kinectSensorChooser1.Kinect.ElevationAngle =
(int)slider1.Value;
            }

            //if (runtime.NuiCamera.ElevationAngle != (int)slider1.Value)
            //{
            //    runtime.NuiCamera.ElevationAngle = (int)slider1.Value;
            //}

        }

        private void slider1_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
        {
            int n = (int)slider1.Value;

            Degree.Content = n.ToString();

        }

        private void StopKinect(KinectSensor sensor)
        {
            if (sensor != null)
            {
                if (sensor.IsRunning)
                {
                    //stop sensor
                    sensor.Stop();

                    //stop audio if not null
                    if (sensor.AudioSource != null)
                    {
                        sensor.AudioSource.Stop();
                    }


                }
            }
        }

        private void CameraPosition(FrameworkElement element, ColorImagePoint
point)
        {
            //Divide by 2 for width and height so point is right in the middle
            // instead of in top/left corner
            Canvas.SetLeft(element, point.X - element.Width / 2);
            Canvas.SetTop(element, point.Y - element.Height / 2);

        }
```

```
        private static void CheckButton(HoverButton button, Ellipse
thumbStick)
        {


            if (IsItemMidpointInContainer(button, thumbStick))
            {
                button.Hovering();
            }
            else
            {
                button.Release();
            }
        }


        public static bool IsItemMidpointInContainer(FrameworkElement
container, FrameworkElement target)
        {
            FindValues(container, target);

            if (_itemTop < _topBoundary || _bottomBoundary < _itemTop)
            {
                //Midpoint of target is outside of top or bottom
                return false;
            }

            if (_itemLeft < _leftBoundary || _rightBoundary < _itemLeft)
            {
                //Midpoint of target is outside of left or right
                return false;
            }

            return true;
        }

        private static void FindValues(FrameworkElement container,
FrameworkElement target)
        {
            var containerTopLeft = container.PointToScreen(new Point());
            var itemTopLeft = target.PointToScreen(new Point());

            _topBoundary = containerTopLeft.Y;
            _bottomBoundary = _topBoundary + container.ActualHeight;
            _leftBoundary = containerTopLeft.X;
            _rightBoundary = _leftBoundary + container.ActualWidth;

            //use midpoint of item (width or height divided by 2)
            _itemLeft = itemTopLeft.X + (target.ActualWidth / 2);
            _itemTop = itemTopLeft.Y + (target.ActualHeight / 2);
        }


        private void ScalePosition(FrameworkElement element, Joint joint)
        {
            //convert the value to X/Y
            //Joint scaledJoint = joint.ScaleTo(1280, 720);

            //convert & scale (.3 = means 1/3 of joint distance)
            Joint scaledJoint = joint.ScaleTo(900, 800, .3f, .3f);
```

```
            Canvas.SetLeft(element, scaledJoint.Position.X);
            Canvas.SetTop(element, scaledJoint.Position.Y);

        }


        private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            closing = true;
            StopKinect(kinectSensorChooser1.Kinect);
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            Main1 Main = new Main1();
            Application.Current.Windows[0].Close();
            this.Hide();
            Main.Show();
        }




    }
}
```

Mix and Match XAML Code

```
<Window x:Class="KinectQuiz.MixMatch"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Kinect Mix and Match" Height="847" Width="991"
Loaded="Window_Loaded"
        xmlns:Controls="clr-
namespace:Coding4Fun.Kinect.Wpf.Controls;assembly=Coding4Fun.Kinect.Wpf"
        xmlns:my="clr-
namespace:Microsoft.Samples.Kinect.WpfViewers;assembly=Microsoft.Samples.Kinec
t.WpfViewers"
        Closing="Window_Closing"
Icon="/KinectQuiz;component/Images/Caillou2.png">
    <Grid x:Name="theGrid" Height="809">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="649*" />
            <ColumnDefinition Width="319*" />
            <ColumnDefinition Width="0*" />
        </Grid.ColumnDefinitions>
        <Canvas Background="Transparent" Height="808" VerticalAlignment="Top"
Grid.ColumnSpan="2" Margin="0,-2,0,0">
            <my:KinectColorViewer Canvas.Left="0" Canvas.Top="2"
Name="kinectColorViewer1" Height="809" Width="969" Kinect="{Binding
ElementName=kinectSensorChooser1, Path=Kinect}" />

            <TextBlock Canvas.Left="212" Canvas.Top="458" Height="23"
Name="textBlock1" Text="TextBlock" Foreground="White" Visibility="Hidden" />
            <Slider Canvas.Left="438" Canvas.Top="773" Height="29"
Name="slider1" Width="212" Maximum="27" Minimum="-27"
ValueChanged="slider1_ValueChanged" />
            <Label Canvas.Left="650" Canvas.Top="769" Content="0" Height="32"
Name="Degree" Width="30" />
```

```
            <Button Canvas.Left="293" Canvas.Top="770" Content="Adjust Kinect
Angle" Height="31" Name="btnangle" Width="148" Click="btnangle_Click" />
            <my:KinectSensorChooser Canvas.Left="-139" Canvas.Top="128"
Name="kinectSensorChooser1" Width="121" Height="64" />
            <Ellipse Canvas.Left="212" Canvas.Top="102" Height="75"
Name="ellipse1" Stroke="Black" Width="75" Fill="#FF50FF00" />
            <TextBlock Background="#FFEB9797" Canvas.Left="0" Canvas.Top="102"
Height="110" Name="language1text" Text="TextBlock" Width="200"
TextAlignment="Center" FontSize="18" />
            <Ellipse Canvas.Left="212" Canvas.Top="228" Fill="#FF50FF00"
Height="75" Name="ellipse2" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="212" Canvas.Top="370" Fill="#FF50FF00"
Height="75" Name="ellipse3" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="212" Canvas.Top="499" Fill="#FF50FF00"
Height="75" Name="ellipse4" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="212" Canvas.Top="640" Fill="#FF50FF00"
Height="75" Name="ellipse5" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="674" Canvas.Top="102" Fill="#FF50FF00"
Height="75" Name="ellipse6" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="680" Canvas.Top="228" Fill="#FF50FF00"
Height="75" Name="ellipse7" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="680" Canvas.Top="364" Fill="#FF50FF00"
Height="75" Name="ellipse8" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="680" Canvas.Top="499" Fill="#FF50FF00"
Height="75" Name="ellipse9" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="680" Canvas.Top="646" Fill="#FF50FF00"
Height="75" Name="ellipse10" Stroke="Black" Width="75" />
            <Controls:HoverButton Canvas.Left="212" Canvas.Top="101"
Height="75" ImageSize="78" Name="hoverButton11" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="212" Canvas.Top="227"
Height="75" ImageSize="78" Name="hoverButton1" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="212" Canvas.Top="370"
Height="75" ImageSize="78" Name="hoverButton2" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="212" Canvas.Top="499"
Height="75" ImageSize="78" Name="hoverButton3" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="212" Canvas.Top="642"
Height="75" ImageSize="78" Name="hoverButton4" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="674" Canvas.Top="102"
Height="75" ImageSize="78" Name="hoverButton5" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="674" Canvas.Top="227"
Height="75" ImageSize="78" Name="hoverButton6" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="680" Canvas.Top="364"
Height="75" ImageSize="78" Name="hoverButton7" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="680" Canvas.Top="499"
Height="75" ImageSize="78" Name="hoverButton8" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="680" Canvas.Top="646"
Height="75" ImageSize="78" Name="hoverButton9" TimeInterval="1000" Width="75"
/>
            <TextBlock Background="#FFEB9797" Canvas.Left="1" Canvas.Top="228"
Height="110" Name="language2text" Text="TextBlock" Width="200"
TextAlignment="Center" FontSize="18" />
```

```
            <TextBlock Background="#FFEB9797" Canvas.Left="0" Canvas.Top="364"
Height="110" Name="language3text" Text="TextBlock" Width="200"
TextAlignment="Center" FontSize="18" />
            <TextBlock Background="#FFEB9797" Canvas.Left="1" Canvas.Top="499"
Height="110" Name="language4text" Text="TextBlock" Width="200"
TextAlignment="Center" FontSize="18" />
            <TextBlock Background="#FFEB9797" Canvas.Left="1" Canvas.Top="640"
Height="110" Name="language5text" Text="TextBlock" Width="200"
TextAlignment="Center" FontSize="18" />
            <TextBlock Background="#FFEB9797" Canvas.Left="769"
Canvas.Top="102" Height="110" Name="answer1text" Text="TextBlock" Width="200"
FontSize="32" TextAlignment="Center" FontStretch="SemiExpanded" />
            <TextBlock Background="#FFEB9797" Canvas.Left="769"
Canvas.Top="228" Height="110" Name="answer2text" Text="TextBlock" Width="200"
FontSize="32" TextAlignment="Center" FontStretch="SemiExpanded" />
            <TextBlock Background="#FFEB9797" Canvas.Left="769"
Canvas.Top="364" Height="110" Name="answer3text" Text="TextBlock" Width="200"
FontSize="32" TextAlignment="Center" FontStretch="SemiExpanded" />
            <TextBlock Background="#FFEB9797" Canvas.Left="769"
Canvas.Top="499" Height="110" Name="answer4text" Text="TextBlock" Width="200"
FontSize="32" TextAlignment="Center" FontStretch="SemiExpanded" />
            <TextBlock Background="#FFEB9797" Canvas.Left="769"
Canvas.Top="642" Height="110" Name="answer5text" Text="TextBlock" Width="200"
FontSize="32" TextAlignment="Center" FontStretch="SemiExpanded" />
            <TextBlock Canvas.Left="212" Canvas.Top="102" Height="74"
Name="language1number" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="212" Canvas.Top="227" Height="74"
Name="language2number" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="212" Canvas.Top="365" Height="74"
Name="language3number" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="212" Canvas.Top="500" Height="74"
Name="language4number" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="680" Canvas.Top="102" Height="74"
Name="answernumber11" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="680" Canvas.Top="228" Height="74"
Name="answernumber22" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="680" Canvas.Top="370" Height="69"
Name="answernumber33" Text="TextBlock" Width="69" Visibility="Hidden" />
            <TextBlock Canvas.Left="680" Canvas.Top="500" Height="74"
Name="answernumber44" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="212" Canvas.Top="646" Height="74"
Name="language5number" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="680" Canvas.Top="642" Height="74"
Name="answernumber55" Text="TextBlock" Width="75" Visibility="Hidden" />
            <Image Canvas.Left="438" Canvas.Top="309" Height="101"
Name="image2" Stretch="Fill" Width="62"
Source="/KinectQuiz;component/Images/arrow.png" />
            <TextBlock Canvas.Left="293" Canvas.Top="31" Height="60"
Name="textBlock2" Text="TextBlock" Width="74" Visibility="Hidden" />
            <TextBlock Canvas.Left="600" Canvas.Top="26" Height="65"
Name="textBlock3" Text="TextBlock" Width="75" Visibility="Hidden" />
            <Controls:HoverButton Canvas.Left="377" Canvas.Top="100"
Height="211" ImageSize="146" Name="hoverButton10" TimeInterval="1000"
Width="217" />
            <Ellipse Height="71" Width="73" Name="RightHand" Canvas.Left="21"
Canvas.Top="100">
                <Ellipse.Fill>
                    <ImageBrush
ImageSource="/KinectQuiz;component/Images/hand.png" />
                </Ellipse.Fill>
            </Ellipse>
```

```
            <Image Canvas.Left="375" Canvas.Top="112" Height="175"
Name="image1" Stretch="Fill" Width="200"
Source="/KinectQuiz;component/Images/button.png" />
            <Image Canvas.Left="51" Canvas.Top="668" Height="82" Name="image7"
Source="/KinectQuiz;component/Images/eyes.jpg" Stretch="Fill" Width="95" />
            <Image Canvas.Left="51" Canvas.Top="527" Height="82" Name="image3"
Stretch="Fill" Width="95" Source="/KinectQuiz;component/Images/ears.jpg" />
            <Image Canvas.Left="51" Canvas.Top="130" Height="82" Name="image4"
Stretch="Fill" Width="95" Source="/KinectQuiz;component/Images/Hair.jpg" />
            <Image Canvas.Left="51" Canvas.Top="392" Height="82" Name="image5"
Stretch="Fill" Width="95" Source="/KinectQuiz;component/Images/Mouth.jpg" />
            <Image Canvas.Left="51" Canvas.Top="254" Height="82" Name="image6"
Stretch="Fill" Width="95" Source="/KinectQuiz;component/Images/Nose.jpg" />
            <Image Canvas.Left="234" Canvas.Top="2" Height="89" Name="image8"
Stretch="Fill" Width="499"
Source="/KinectQuiz;component/Images/Title%20Face.png" />
            <Image Canvas.Left="377" Canvas.Top="413" Height="61"
Name="image9" Stretch="Fill" Width="198"
Source="/KinectQuiz;component/Images/CheckAnswer.png" />
            <Image Canvas.Left="1" Canvas.Top="2" Height="41" Name="image10"
Stretch="Fill" Width="79" Source="/KinectQuiz;component/Images/Back.png" />
            <Controls:HoverButton Canvas.Left="2" Canvas.Top="2" Height="58"
ImageSize="146" Name="hoverButton12" TimeInterval="1000" Width="78" />
        </Canvas>
        <Image Height="175" Name="deed" Stretch="Fill" Width="202"
Source="/KinectQuiz;component/Images/button.jpg" Grid.Column="2"
Margin="20,160,-222,474" />
    </Grid>
</Window>
```

Mix and Match Code

```
// (c) Copyright Microsoft Corporation.
// This source is subject to the Microsoft Public License (Ms-PL).
// Please see http://go.microsoft.com/fwlink/?LinkID=131993 for details.
// All other rights reserved.
//This project is created by Mohd Hazwan b Sahabuzan 12019
//but some of the code and resources are taken, modify and used from tutorial
websites
//such as http://channel9.msdn.com/Series/KinectQuickstart and
http://raychambers.wordpress.com

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.Media;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
```

```
using Microsoft.Kinect;
using Coding4Fun.Kinect.Wpf;
using Coding4Fun.Kinect.Wpf.Controls;
using System.IO;




namespace KinectQuiz
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MixMatch : Window
    {

        private static double _topBoundary;
        private static double _bottomBoundary;
        private static double _leftBoundary;
        private static double _rightBoundary;
        private static double _itemLeft;
        private static double _itemTop;

        string language1;
        string language2;
        string language3;
        string language4;
        string language5;
        //string language6;
        //string language7;
        //string language8;

        string languagenumber1;
        string languagenumber2;
        string languagenumber3;
        string languagenumber4;
        string languagenumber5;
        //string languagenumber6;
        //string languagenumber7;
        //string languagenumber8;

        string answer1;
        string answer2;
        string answer3;
        string answer4;
        string answer5;
        //string answer6;
        //string answer7;
        //string answer8;

        string answernumber1;
        string answernumber2;
        string answernumber3;
        string answernumber4;
        string answernumber5;
        //string answernumber6;
        //string answernumber7;
        //string answernumber8;

        int finallanguage1;
        int finalanswer1;
```

```
public MixMatch()
{
    InitializeComponent();

    finallanguage1 = 1;
    finalanswer1 = 2;

    TextReader tr = new StreamReader("otherlanguage.txt");

    int NumberOfLines = 9;

    string[] ListLines = new string[NumberOfLines];
    for (int i = 1; i < NumberOfLines; i++)
    {
        ListLines[i] = tr.ReadLine();
    }
    int b = 0;
    {
        string[] parts = ListLines[1].Split(',');
        foreach (string part in parts)
        {
        }
        b++;
        language1 = parts[0];
        languagenumber1 = parts[1];
    }
    int c = 0;
    {
        string[] parts = ListLines[2].Split(',');
        foreach (string part in parts)
        {
        }
        c++;
        language2 = parts[0];
        languagenumber2 = parts[1];
    }

    int g = 0;
    {
        string[] parts = ListLines[3].Split(',');
        foreach (string part in parts)
        {
        }
        g++;
        language3 = parts[0];
        languagenumber3 = parts[1];
    }

    int h = 0;
    {
        string[] parts = ListLines[4].Split(',');
        foreach (string part in parts)
        {
        }
        h++;
        language4 = parts[0];
        languagenumber4 = parts[1];
    }

    int j = 0;
    {
```

```
    string[] parts = ListLines[5].Split(',');
    foreach (string part in parts)
    {
    }
    j++;
    language5 = parts[0];
    languagenumber5 = parts[1];
}

TextReader trr = new StreamReader("english.txt");

int NumberOfLines2 = 9;

string[] ListLines2 = new string[NumberOfLines2];
for (int i = 1; i < NumberOfLines2; i++)
{
    ListLines2[i] = trr.ReadLine();
}
int d = 0;
{
    string[] parts = ListLines2[1].Split(',');
    foreach (string part in parts)
    {
    }
    d++;
    answer1 = parts[0];
    answernumber1 = parts[1];
}
int e = 0;
{
    string[] parts = ListLines2[2].Split(',');
    foreach (string part in parts)
    {
    }
    e++;
    answer2 = parts[0];
    answernumber2 = parts[1];
}

int f = 0;
{
    string[] parts = ListLines2[3].Split(',');
    foreach (string part in parts)
    {
    }
    f++;
    answer3 = parts[0];
    answernumber3 = parts[1];
}

int n = 0;
{
    string[] parts = ListLines2[4].Split(',');
    foreach (string part in parts)
    {
    }
    n++;
    answer4 = parts[0];
    answernumber4 = parts[1];
}

int o = 0;
```

```
            {
                string[] parts = ListLines2[5].Split(',');
                foreach (string part in parts)
                {
                }
                o++;
                answer5 = parts[0];
                answernumber5 = parts[1];
            }

            // button1.Click += new RoutedEventHandler(button1_Click);
            hoverButton1.Click += new RoutedEventHandler(hoverButton1_Click);
            hoverButton2.Click += new RoutedEventHandler(hoverButton2_Click);
            hoverButton3.Click += new RoutedEventHandler(hoverButton3_Click);
            hoverButton4.Click += new RoutedEventHandler(hoverButton4_Click);
            hoverButton5.Click += new RoutedEventHandler(hoverButton5_Click);
            hoverButton6.Click += new RoutedEventHandler(hoverButton6_Click);
            hoverButton7.Click += new RoutedEventHandler(hoverButton7_Click);
            hoverButton8.Click += new RoutedEventHandler(hoverButton8_Click);
            hoverButton9.Click += new RoutedEventHandler(hoverButton9_Click);
            hoverButton10.Click += new
RoutedEventHandler(hoverButton10_Click);
            hoverButton11.Click += new
RoutedEventHandler(hoverButton11_Click);
            hoverButton12.Click += new
RoutedEventHandler(hoverButton12_Click);
        }



        bool closing = false;
        const int skeletonCount = 6;
        Skeleton[] allSkeletons = new Skeleton[skeletonCount];

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            kinectSensorChooser1.KinectSensorChanged += new
DependencyPropertyChangedEventHandler(kinectSensorChooser1_KinectSensorChanged
);

            language1text.Text = language1;
            language1number.Text = languagenumber1;

            language2text.Text = language2;
            language2number.Text = languagenumber2;

            language3text.Text = language3;
            language3number.Text = languagenumber3;

            language4text.Text = language4;
            language4number.Text = languagenumber4;

            language5text.Text = language5;
            language5number.Text = languagenumber5;

            answer1text.Text = answer1;
            answernumber11.Text = answernumber1;

            answer2text.Text = answer2;
            answernumber22.Text = answernumber2;
```

```
            answer3text.Text = answer3;
            answernumber33.Text = answernumber3;

            answer4text.Text = answer4;
            answernumber44.Text = answernumber4;

            answer5text.Text = answer5;
            answernumber55.Text = answernumber5;

        }

        void hoverButton11_Click(object sender, RoutedEventArgs e)
        {
            ellipse1.Fill = new SolidColorBrush(Colors.Red);
            ellipse2.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse3.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse4.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse5.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finallanguage1 = Convert.ToInt32(language1number.Text);
            textBlock2.Text = "1";
        }

        void hoverButton1_Click(object sender, RoutedEventArgs e)
        {
            ellipse2.Fill = new SolidColorBrush(Colors.Red);
            ellipse1.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse3.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse4.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse5.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finallanguage1 = Convert.ToInt32(language2number.Text);
            textBlock2.Text = "2";
        }

        void hoverButton2_Click(object sender, RoutedEventArgs e)
        {
            ellipse3.Fill = new SolidColorBrush(Colors.Red);
            ellipse2.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse1.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse4.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse5.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finallanguage1 = Convert.ToInt32(language3number.Text);
            textBlock2.Text = "3";
        }

        void hoverButton3_Click(object sender, RoutedEventArgs e)
        {
            ellipse4.Fill = new SolidColorBrush(Colors.Red);
```

```
            ellipse2.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse3.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse1.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse5.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finallanguage1 = Convert.ToInt32(language4number.Text);
            textBlock2.Text = "4";
        }
        void hoverButton4_Click(object sender, RoutedEventArgs e)
        {
            ellipse5.Fill = new SolidColorBrush(Colors.Red);
            ellipse2.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse3.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse4.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse1.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finallanguage1 = Convert.ToInt32(language5number.Text);
            textBlock2.Text = "5";
        }

        void hoverButton5_Click(object sender, RoutedEventArgs e)
        {
            ellipse6.Fill = new SolidColorBrush(Colors.Red);
            ellipse7.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse8.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse9.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse10.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finalanswer1 = Convert.ToInt32(answernumber11.Text);
            textBlock3.Text = "1";
        }

        void hoverButton6_Click(object sender, RoutedEventArgs e)
        {
            ellipse7.Fill = new SolidColorBrush(Colors.Red);
            ellipse8.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse6.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse9.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse10.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finalanswer1 = Convert.ToInt32(answernumber22.Text);
            textBlock3.Text = "2";

        }

        void hoverButton7_Click(object sender, RoutedEventArgs e)
```

```
        {
            ellipse8.Fill = new SolidColorBrush(Colors.Red);
            ellipse7.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse6.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse9.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse10.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finalanswer1 = Convert.ToInt32(answernumber33.Text);
            textBlock3.Text = "3";
        }

        void hoverButton8_Click(object sender, RoutedEventArgs e)
        {
            ellipse9.Fill = new SolidColorBrush(Colors.Red);
            ellipse7.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse8.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse6.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse10.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finalanswer1 = Convert.ToInt32(answernumber44.Text);
            textBlock3.Text = "4";
        }

        void hoverButton9_Click(object sender, RoutedEventArgs e)
        {
            ellipse10.Fill = new SolidColorBrush(Colors.Red);
            ellipse7.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse8.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse9.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse6.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finalanswer1 = Convert.ToInt32(answernumber55.Text);
            textBlock3.Text = "5";

        }

        void hoverButton10_Click(object sender, RoutedEventArgs e)
        {
            if (finalanswer1 == finallanguage1)
            {
                //to reset all of the ellipse colors
                ellipse1.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse2.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse3.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse4.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
```

```
                ellipse5.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse6.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse7.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse8.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse9.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse10.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

                //to check which text block is being used and change its color
                if (textBlock2.Text == "1")
                {
                    language1text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock2.Text == "2")
                {
                    language2text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock2.Text == "3")
                {
                    language3text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock2.Text == "4")
                {
                    language4text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock2.Text == "5")
                {
                    language5text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }

                if (textBlock3.Text == "1")
                {
                    answer1text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock3.Text == "2")
                {
                    answer2text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock3.Text == "3")
                {
                    answer3text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock3.Text == "4")
                {
                    answer4text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock3.Text == "5")
                {
```

```
                    answer5text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }

            }
            else
            {
                //MessageBox.Show("Incorrect");
                // to reset the text block color to basic color
                ellipse1.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse2.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse3.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse4.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse5.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse6.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse7.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse8.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse9.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse10.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            }
        }

        void hoverButton12_Click(object sender, RoutedEventArgs e)
        {

            Head headInfo = new Head();
            this.Hide();

            Application.Current.Windows[0].Close();
            headInfo.Show();
            headInfo.Activate();
            //throw new NotImplementedException();
        }


        void kinectSensorChooser1_KinectSensorChanged(object sender,
DependencyPropertyChangedEventArgs e)
        {
            KinectSensor old = (KinectSensor)e.OldValue;

            StopKinect(old);

            KinectSensor sensor = (KinectSensor)e.NewValue;

            if (sensor == null)
            {
                return;
            }
```

```
            var parameters = new TransformSmoothParameters
            {
                Smoothing = 0.3f,
                Correction = 0.0f,
                Prediction = 0.0f,
                JitterRadius = 1.0f,
                MaxDeviationRadius = 0.5f
            };
            sensor.SkeletonStream.Enable(parameters);

            //sensor.SkeletonStream.Enable();

            sensor.AllFramesReady += new
EventHandler<AllFramesReadyEventArgs>(sensor_AllFramesReady);

sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);

sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);

            try
            {
                sensor.Start();
            }
            catch (System.IO.IOException)
            {
                kinectSensorChooser1.AppConflictOccurred();
            }
        }

        void sensor_AllFramesReady(object sender, AllFramesReadyEventArgs e)
        {
            if (closing)
            {
                return;
            }

            //Get a skeleton
            Skeleton first = GetFirstSkeleton(e);

            if (first == null)
            {
                return;
            }

            GetCameraPoint(first, e);

            //set scaled position
            //ScalePosition(headImage, first.Joints[JointType.Head]);
            //ScalePosition(leftEllipse, first.Joints[JointType.HandLeft]);
            ScalePosition(RightHand, first.Joints[JointType.HandRight]);




        }

        void GetCameraPoint(Skeleton first, AllFramesReadyEventArgs e)
        {

            using (DepthImageFrame depth = e.OpenDepthImageFrame())
```

```
            {
                if (depth == null ||
                    kinectSensorChooser1.Kinect == null)
                {
                    return;
                }


                //Map a joint location to a point on the depth map
                //head
                DepthImagePoint headDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.Head].Position);
                //left hand
                DepthImagePoint leftDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.HandLeft].Position);
                //right hand
                DepthImagePoint rightDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.HandRight].Position);


                //Map a depth point to a point on the color image
                //head
                ColorImagePoint headColorPoint =
                    depth.MapToColorImagePoint(headDepthPoint.X,
headDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //left hand
                ColorImagePoint leftColorPoint =
                    depth.MapToColorImagePoint(leftDepthPoint.X,
leftDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //right hand
                ColorImagePoint rightColorPoint =
                    depth.MapToColorImagePoint(rightDepthPoint.X,
rightDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);


                //Set location
                //CameraPosition(headImage, headColorPoint);
                //CameraPosition(leftEllipse, leftColorPoint);
                CameraPosition(RightHand, rightColorPoint);

                CheckButton(hoverButton11, RightHand);
                CheckButton(hoverButton1, RightHand);
                CheckButton(hoverButton2, RightHand);
                CheckButton(hoverButton3, RightHand);
                CheckButton(hoverButton4, RightHand);
                CheckButton(hoverButton5, RightHand);
                CheckButton(hoverButton6, RightHand);
                CheckButton(hoverButton7, RightHand);
                CheckButton(hoverButton8, RightHand);
                CheckButton(hoverButton9, RightHand);
                CheckButton(hoverButton10, RightHand);
                CheckButton(hoverButton12, RightHand);



            }
```

```
        }


        Skeleton GetFirstSkeleton(AllFramesReadyEventArgs e)
        {
            using (SkeletonFrame skeletonFrameData = e.OpenSkeletonFrame())
            {
                if (skeletonFrameData == null)
                {
                    return null;
                }


                skeletonFrameData.CopySkeletonDataTo(allSkeletons);

                //get the first tracked skeleton
                Skeleton first = (from s in allSkeletons
                                  where s.TrackingState ==
SkeletonTrackingState.Tracked
                                  select s).FirstOrDefault();

                return first;



            }
        }


        private void btnangle_Click(object sender, RoutedEventArgs e)
        {
            if (kinectSensorChooser1.Kinect.ElevationAngle !=
(int)slider1.Value)
            {
                kinectSensorChooser1.Kinect.ElevationAngle =
(int)slider1.Value;
            }

            //if (runtime.NuiCamera.ElevationAngle != (int)slider1.Value)
            //{
            //    runtime.NuiCamera.ElevationAngle = (int)slider1.Value;
            //}

        }

        private void slider1_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
        {
            int n = (int)slider1.Value;

            Degree.Content = n.ToString();

        }

        private void StopKinect(KinectSensor sensor)
        {
            if (sensor != null)
            {
                if (sensor.IsRunning)
                {
                    //stop sensor
                    sensor.Stop();
```

```
                    //stop audio if not null
                    if (sensor.AudioSource != null)
                    {
                        sensor.AudioSource.Stop();
                    }


                }
            }
        }

        private void CameraPosition(FrameworkElement element, ColorImagePoint
point)
        {
            //Divide by 2 for width and height so point is right in the middle
            // instead of in top/left corner
            Canvas.SetLeft(element, point.X - element.Width / 2);
            Canvas.SetTop(element, point.Y - element.Height / 2);

        }

        private static void CheckButton(HoverButton button, Ellipse
thumbStick)
        {


            if (IsItemMidpointInContainer(button, thumbStick))
            {
                button.Hovering();
            }
            else
            {
                button.Release();
            }
        }


        public static bool IsItemMidpointInContainer(FrameworkElement
container, FrameworkElement target)
        {
            FindValues(container, target);

            if (_itemTop < _topBoundary || _bottomBoundary < _itemTop)
            {
                //Midpoint of target is outside of top or bottom
                return false;
            }

            if (_itemLeft < _leftBoundary || _rightBoundary < _itemLeft)
            {
                //Midpoint of target is outside of left or right
                return false;
            }

            return true;
        }

        private static void FindValues(FrameworkElement container,
FrameworkElement target)
        {
            var containerTopLeft = container.PointToScreen(new Point());
```

```
            var itemTopLeft = target.PointToScreen(new Point());

            _topBoundary = containerTopLeft.Y;
            _bottomBoundary = _topBoundary + container.ActualHeight;
            _leftBoundary = containerTopLeft.X;
            _rightBoundary = _leftBoundary + container.ActualWidth;

            //use midpoint of item (width or height divided by 2)
            _itemLeft = itemTopLeft.X + (target.ActualWidth / 2);
            _itemTop = itemTopLeft.Y + (target.ActualHeight / 2);
        }


        private void ScalePosition(FrameworkElement element, Joint joint)
        {
            //convert the value to X/Y
            //Joint scaledJoint = joint.ScaleTo(1280, 720);

            //convert & scale (.3 = means 1/3 of joint distance)
            Joint scaledJoint = joint.ScaleTo(1280, 720, .3f, .3f);

            Canvas.SetLeft(element, scaledJoint.Position.X);
            Canvas.SetTop(element, scaledJoint.Position.Y);

        }


        private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            closing = true;
            StopKinect(kinectSensorChooser1.Kinect);
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {

        }







    }
}
```

Main1 XAML Code

```
<Window x:Class="KinectQuiz.Main1"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="KinectMain" Height="840" Width="921" Loaded="Window_Loaded"
        xmlns:Controls="clr-
namespace:Coding4Fun.Kinect.Wpf.Controls;assembly=Coding4Fun.Kinect.Wpf"
        xmlns:my="clr-
namespace:Microsoft.Samples.Kinect.WpfViewers;assembly=Microsoft.Samples.Kinec
t.WpfViewers"
```

```
        Closing="Window_Closing"
Icon="/KinectQuiz;component/Images/Caillou2.png">
    <Grid x:Name="theGrid">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="649*" />
            <ColumnDefinition Width="319*" />
            <ColumnDefinition Width="0*" />
        </Grid.ColumnDefinitions>
        <Canvas Background="Transparent" Height="803" VerticalAlignment="Top"
Grid.ColumnSpan="2" Margin="0,-2,0,0">
            <my:KinectColorViewer Canvas.Left="6" Canvas.Top="-3"
Name="kinectColorViewer1" Height="800" Width="900" Kinect="{Binding
ElementName=kinectSensorChooser1, Path=Kinect}" />
            <Slider Canvas.Left="325" Canvas.Top="768" Height="29"
Name="slider1" Width="212" Maximum="27" Minimum="-27"
ValueChanged="slider1_ValueChanged" />
            <Label Canvas.Left="533" Canvas.Top="765" Content="0" Height="32"
Name="Degree" Width="30" />
            <Button Canvas.Left="179" Canvas.Top="766" Content="Adjust Kinect
Angle" Height="31" Name="btnangle" Width="148" Click="btnangle_Click" />
            <my:KinectSensorChooser Canvas.Left="239" Canvas.Top="247"
Name="kinectSensorChooser1" Width="396" Height="236" />
            <Controls:HoverButton Canvas.Left="612" Canvas.Top="234"
Height="150" ImageSize="146" Name="hoverButton10" TimeInterval="1000"
Width="146" />
            <Ellipse Height="71" Width="73" Name="RightHand" Canvas.Left="100"
Canvas.Top="215">
                <Ellipse.Fill>
                    <ImageBrush
ImageSource="/KinectQuiz;component/Images/hand.png" />
                </Ellipse.Fill>
            </Ellipse>
            <Controls:HoverButton Canvas.Left="612" Canvas.Top="386"
Height="150" ImageSize="146" Name="hoverButton1" TimeInterval="1000"
Width="146" />
            <Controls:HoverButton Canvas.Left="612" Canvas.Top="534"
Height="150" ImageSize="146" Name="hoverButton2" TimeInterval="1000"
Width="146" />
            <Image Canvas.Left="0" Canvas.Top="281" Height="255" Name="image4"
Stretch="Fill" Width="185" Source="/KinectQuiz;component/Images/Caillou2.png"
/>
            <Image Canvas.Left="179" Canvas.Top="82" Height="167"
Name="image5" Stretch="Fill" Width="539"
Source="/KinectQuiz;component/Images/MainTitle.png" />
            <Image Height="146" Name="image1" Stretch="Fill" Width="287"
Source="/KinectQuiz;component/Images/HeadMain.png" Canvas.Left="612"
Canvas.Top="238" />
            <Image Canvas.Left="612" Canvas.Top="534" Height="150"
Name="image3" Stretch="Fill" Width="294"
Source="/KinectQuiz;component/Images/Extra.png" />
            <my:KinectDepthViewer Canvas.Left="6" Canvas.Top="659"
Name="kinectDepthViewer1" Height="138" Width="153" Kinect="{Binding
ElementName=kinectSensorChooser1, Path=Kinect}" />
            <Image Height="150" Name="image2" Stretch="Fill" Width="287"
Source="/KinectQuiz;component/Images/Body2.png" Canvas.Left="612"
Canvas.Top="386" />
        </Canvas>
    </Grid>
</Window>
```

Main1 Code

```
// (c) Copyright Microsoft Corporation.
// This source is subject to the Microsoft Public License (Ms-PL).
// Please see http://go.microsoft.com/fwlink/?LinkID=131993 for details.
// All other rights reserved.
//This project is created by Mohd Hazwan b Sahabuzan 12019
//but some of the code and resources are taken, modify and used from tutorial
websites
//such as http://channel9.msdn.com/Series/KinectQuickstart and
http://raychambers.wordpress.com

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.Media;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using Coding4Fun.Kinect.Wpf;
using Coding4Fun.Kinect.Wpf.Controls;
using System.IO;



namespace KinectQuiz
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class Main1 : Window
    {

        private static double _topBoundary;
        private static double _bottomBoundary;
        private static double _leftBoundary;
        private static double _rightBoundary;
        private static double _itemLeft;
        private static double _itemTop;

        //string language1;
        //string language2;
        //string language3;
        //string language4;
        //string language5;
        //string language6;
        //string language7;
```

```
        //string language8;

        //string languagenumber1;
        //string languagenumber2;
        //string languagenumber3;
        //string languagenumber4;
        //string languagenumber5;
        //string languagenumber6;
        //string languagenumber7;
        //string languagenumber8;

        //string answer1;
        //string answer2;
        //string answer3;
        //string answer4;
        //string answer5;
        //string answer6;
        //string answer7;
        //string answer8;

        //string answernumber1;
        //string answernumber2;
        //string answernumber3;
        //string answernumber4;
        //string answernumber5;
        //string answernumber6;
        //string answernumber7;
        //string answernumber8;

        //int finallanguage1;
        //int finalanswer1;

        public Main1()
        {
            InitializeComponent();



            hoverButton10.Click += new
RoutedEventHandler(hoverButton10_Click);
            hoverButton1.Click += new RoutedEventHandler(hoverButton1_Click);
            hoverButton2.Click += new RoutedEventHandler(hoverButton2_Click);

        }



        bool closing = false;
        const int skeletonCount = 6;
        Skeleton[] allSkeletons = new Skeleton[skeletonCount];

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            kinectSensorChooser1.KinectSensorChanged += new
DependencyPropertyChangedEventHandler(kinectSensorChooser1_KinectSensorChanged
);


        }

        void hoverButton1_Click(object sender, RoutedEventArgs e)
```

```
        {

            Body bdyForm = new Body();
            this.Hide();
            Application.Current.Windows[0].Close();
            bdyForm.Show();
        }

        void hoverButton2_Click(object sender, RoutedEventArgs e)
        {
            Extra extraForm = new Extra();
            Application.Current.Windows[0].Close();
            this.Hide();
            extraForm.Show();
        }

        void hoverButton10_Click(object sender, RoutedEventArgs e)
        {
            Head headLearn = new Head();
            Application.Current.Windows[0].Close();
            this.Hide();
            headLearn.Show();

        }

        void kinectSensorChooser1_KinectSensorChanged(object sender,
DependencyPropertyChangedEventArgs e)
        {
            KinectSensor old = (KinectSensor)e.OldValue;

            StopKinect(old);

            KinectSensor sensor = (KinectSensor)e.NewValue;

            if (sensor == null)
            {
                return;
            }



            var parameters = new TransformSmoothParameters
            {
                Smoothing = 0.3f,
                Correction = 0.0f,
                Prediction = 0.0f,
                JitterRadius = 1.0f,
                MaxDeviationRadius = 0.5f
            };
            sensor.SkeletonStream.Enable(parameters);

            //sensor.SkeletonStream.Enable();

            sensor.AllFramesReady += new
EventHandler<AllFramesReadyEventArgs>(sensor_AllFramesReady);

sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);

sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);

            try
```

```
            {
                sensor.Start();
            }
            catch (System.IO.IOException)
            {
                kinectSensorChooser1.AppConflictOccurred();
            }
        }

        void sensor_AllFramesReady(object sender, AllFramesReadyEventArgs e)
        {
            if (closing)
            {
                return;
            }

            //Get a skeleton
            Skeleton first = GetFirstSkeleton(e);

            if (first == null)
            {
                return;
            }

            GetCameraPoint(first, e);

            //set scaled position
            //ScalePosition(headImage, first.Joints[JointType.Head]);
            //ScalePosition(leftEllipse, first.Joints[JointType.HandLeft]);
            ScalePosition(RightHand, first.Joints[JointType.HandRight]);







        }

        void GetCameraPoint(Skeleton first, AllFramesReadyEventArgs e)
        {

            using (DepthImageFrame depth = e.OpenDepthImageFrame())
            {
                if (depth == null ||
                    kinectSensorChooser1.Kinect == null)
                {
                    return;
                }


                //Map a joint location to a point on the depth map
                //head
                DepthImagePoint headDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.Head].Position);
                //left hand
                DepthImagePoint leftDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.HandLeft].Position);
                //right hand
                DepthImagePoint rightDepthPoint =
```

```
depth.MapFromSkeletonPoint(first.Joints[JointType.HandRight].Position);


                //Map a depth point to a point on the color image
                //head
                ColorImagePoint headColorPoint =
                    depth.MapToColorImagePoint(headDepthPoint.X,
headDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //left hand
                ColorImagePoint leftColorPoint =
                    depth.MapToColorImagePoint(leftDepthPoint.X,
leftDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //right hand
                ColorImagePoint rightColorPoint =
                    depth.MapToColorImagePoint(rightDepthPoint.X,
rightDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);


                //Set location
                //CameraPosition(headImage, headColorPoint);
                //CameraPosition(leftEllipse, leftColorPoint);
                CameraPosition(RightHand, rightColorPoint);

                CheckButton(hoverButton10, RightHand);
                CheckButton(hoverButton1, RightHand);
                CheckButton(hoverButton2, RightHand);


            }
        }


        Skeleton GetFirstSkeleton(AllFramesReadyEventArgs e)
        {
            using (SkeletonFrame skeletonFrameData = e.OpenSkeletonFrame())
            {
                if (skeletonFrameData == null)
                {
                    return null;
                }


                skeletonFrameData.CopySkeletonDataTo(allSkeletons);

                //get the first tracked skeleton
                Skeleton first = (from s in allSkeletons
                                  where s.TrackingState ==
SkeletonTrackingState.Tracked
                                  select s).FirstOrDefault();

                return first;



            }
        }
```

```
        private void btnangle_Click(object sender, RoutedEventArgs e)
        {
            if (kinectSensorChooser1.Kinect.ElevationAngle !=
(int)slider1.Value)
            {
                kinectSensorChooser1.Kinect.ElevationAngle =
(int)slider1.Value;
            }

            //if (runtime.NuiCamera.ElevationAngle != (int)slider1.Value)
            //{
            //    runtime.NuiCamera.ElevationAngle = (int)slider1.Value;
            //}

        }

        private void slider1_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
        {
            int n = (int)slider1.Value;

            Degree.Content = n.ToString();

        }

        private void StopKinect(KinectSensor sensor)
        {
            if (sensor != null)
            {
                if (sensor.IsRunning)
                {
                    //stop sensor
                    sensor.Stop();

                    //stop audio if not null
                    if (sensor.AudioSource != null)
                    {
                        sensor.AudioSource.Stop();
                    }


                }
            }
        }

        private void CameraPosition(FrameworkElement element, ColorImagePoint
point)
        {
            //Divide by 2 for width and height so point is right in the middle
            // instead of in top/left corner
            Canvas.SetLeft(element, point.X - element.Width / 2);
            Canvas.SetTop(element, point.Y - element.Height / 2);

        }

        private static void CheckButton(HoverButton button, Ellipse
thumbStick)
        {


            if (IsItemMidpointInContainer(button, thumbStick))
            {
```

```
                button.Hovering();
            }
            else
            {
                button.Release();
            }
        }


        public static bool IsItemMidpointInContainer(FrameworkElement
container, FrameworkElement target)
        {
            FindValues(container, target);

            if (_itemTop < _topBoundary || _bottomBoundary < _itemTop)
            {
                //Midpoint of target is outside of top or bottom
                return false;
            }

            if (_itemLeft < _leftBoundary || _rightBoundary < _itemLeft)
            {
                //Midpoint of target is outside of left or right
                return false;
            }

            return true;
        }

        private static void FindValues(FrameworkElement container,
FrameworkElement target)
        {
            var containerTopLeft = container.PointToScreen(new Point());
            var itemTopLeft = target.PointToScreen(new Point());

            _topBoundary = containerTopLeft.Y;
            _bottomBoundary = _topBoundary + container.ActualHeight;
            _leftBoundary = containerTopLeft.X;
            _rightBoundary = _leftBoundary + container.ActualWidth;

            //use midpoint of item (width or height divided by 2)
            _itemLeft = itemTopLeft.X + (target.ActualWidth / 2);
            _itemTop = itemTopLeft.Y + (target.ActualHeight / 2);
        }


        private void ScalePosition(FrameworkElement element, Joint joint)
        {
            //convert the value to X/Y
            //Joint scaledJoint = joint.ScaleTo(1280, 720);

            //convert & scale (.3 = means 1/3 of joint distance)
            Joint scaledJoint = joint.ScaleTo(900, 800, .3f, .3f);

            Canvas.SetLeft(element, scaledJoint.Position.X);
            Canvas.SetTop(element, scaledJoint.Position.Y);

        }


        private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
```

```
        {
            closing = true;
            StopKinect(kinectSensorChooser1.Kinect);
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            Head headLearn = new Head();
            Application.Current.Windows[0].Close();
            this.Hide();
            headLearn.Show();

        }

        private void button2_Click(object sender, RoutedEventArgs e)
        {
            Body bdyForm = new Body();
            this.Hide();
            Application.Current.Windows[0].Close();
            bdyForm.Show();
        }

        private void button3_Click(object sender, RoutedEventArgs e)
        {
            Extra extraForm = new Extra();
            Application.Current.Windows[0].Close();
            this.Hide();
            extraForm.Show();
        }




    }
}
```

Head XAML Code

```
<Window x:Class="KinectQuiz.Head"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="KinectMain" Height="685" Width="950" Loaded="Window_Loaded"
        xmlns:Controls="clr-
namespace:Coding4Fun.Kinect.Wpf.Controls;assembly=Coding4Fun.Kinect.Wpf"
        xmlns:my="clr-
namespace:Microsoft.Samples.Kinect.WpfViewers;assembly=Microsoft.Samples.Kinec
t.WpfViewers"
        Closing="Window_Closing"
Icon="/KinectQuiz;component/Images/Caillou2.png">
    <Grid x:Name="theGrid" Height="628" Width="900">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="649*" />
            <ColumnDefinition Width="319*" />
            <ColumnDefinition Width="0*" />
        </Grid.ColumnDefinitions>
        <Canvas Background="Transparent" Height="630" VerticalAlignment="Top"
Grid.ColumnSpan="2" Margin="0,-2,0,0">
```

```
            <my:KinectSensorChooser Canvas.Left="272" Canvas.Top="188"
Name="kinectSensorChooser1" Width="396" Height="236" />
            <Image Canvas.Left="0" Canvas.Top="0" Height="587" Name="image3"
Stretch="Fill" Width="894" Source="/KinectQuiz;component/Images/Slide11.PNG"
/>
            <Slider Canvas.Left="332" Canvas.Top="595" Height="29"
Name="slider1" Width="212" Maximum="27" Minimum="-27"
ValueChanged="slider1_ValueChanged" />
            <Label Canvas.Left="548" Canvas.Top="593" Content="0" Height="32"
Name="Degree" Width="30" />
            <Button Canvas.Left="178" Canvas.Top="593" Content="Adjust Kinect
Angle" Height="31" Name="btnangle" Width="148" Click="btnangle_Click" />
            <my:KinectDepthViewer Canvas.Left="6" Canvas.Top="473"
Name="kinectDepthViewer1" Height="151" Width="167" Kinect="{Binding
ElementName=kinectSensorChooser1, Path=Kinect}" />

            <Image Canvas.Left="709" Canvas.Top="0" Height="106" Name="image2"
Stretch="Fill" Width="191" Source="/KinectQuiz;component/Images/Exercise.png"
/>
            <Image Canvas.Left="10" Canvas.Top="10" Height="83" Name="image1"
Stretch="Fill" Width="123" Source="/KinectQuiz;component/Images/Back.png" />
            <Controls:HoverButton Canvas.Left="10" Canvas.Top="10" Height="83"
ImageSize="146" Name="hoverButton5" TimeInterval="1000" Width="123" />
            <Controls:HoverButton Canvas.Left="709" Canvas.Top="0"
Height="106" ImageSize="146" Name="hoverButton6" TimeInterval="1000"
Width="191" />
            <Ellipse Height="71" Width="73" Name="RightHand" Canvas.Left="505"
Canvas.Top="197">
                <Ellipse.Fill>
                    <ImageBrush
ImageSource="/KinectQuiz;component/Images/hand.png" />
                </Ellipse.Fill>
            </Ellipse>
        </Canvas>
    </Grid>
</Window>
```

Head Code

```
// (c) Copyright Microsoft Corporation.
// This source is subject to the Microsoft Public License (Ms-PL).
// Please see http://go.microsoft.com/fwlink/?LinkID=131993 for details.
// All other rights reserved.
//This project is created by Mohd Hazwan b Sahabuzan 12019
//but some of the code and resources are taken, modify and used from tutorial
websites
//such as http://channel9.msdn.com/Series/KinectQuickstart and
http://raychambers.wordpress.com

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.Media;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
```

```
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using Coding4Fun.Kinect.Wpf;
using Coding4Fun.Kinect.Wpf.Controls;
using System.IO;



namespace KinectQuiz
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class Head : Window
    {

        private static double _topBoundary;
        private static double _bottomBoundary;
        private static double _leftBoundary;
        private static double _rightBoundary;
        private static double _itemLeft;
        private static double _itemTop;



        public Head()
        {
            InitializeComponent();



            //hoverButton10.Click += new
RoutedEventHandler(hoverButton10_Click);
            //hoverButton1.Click += new
RoutedEventHandler(hoverButton1_Click);
            //hoverButton2.Click += new
RoutedEventHandler(hoverButton2_Click);

        }



        bool closing = false;
        const int skeletonCount = 6;
        Skeleton[] allSkeletons = new Skeleton[skeletonCount];

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            kinectSensorChooser1.KinectSensorChanged += new
DependencyPropertyChangedEventHandler(kinectSensorChooser1_KinectSensorChanged
);


            hoverButton5.Click += new RoutedEventHandler(hoverButton5_Click);
            hoverButton6.Click += new RoutedEventHandler(hoverButton6_Click);
        }
```

```
        void hoverButton6_Click(object sender, RoutedEventArgs e)
        {
            MixMatch headMatch = new MixMatch();

            this.Hide();

            Application.Current.Windows[0].Close();
            headMatch.Show();
            headMatch.Activate();
        }

        void hoverButton5_Click(object sender, RoutedEventArgs e)
        {
            Main1 MainForm = new Main1();
            this.Hide();

            Application.Current.Windows[0].Close();
            MainForm.Show();
            MainForm.Activate();
        }


        void kinectSensorChooser1_KinectSensorChanged(object sender,
DependencyPropertyChangedEventArgs e)
        {
            KinectSensor old = (KinectSensor)e.OldValue;

            StopKinect(old);

            KinectSensor sensor = (KinectSensor)e.NewValue;

            if (sensor == null)
            {
                return;
            }



            var parameters = new TransformSmoothParameters
            {
                Smoothing = 0.3f,
                Correction = 0.0f,
                Prediction = 0.0f,
                JitterRadius = 1.0f,
                MaxDeviationRadius = 0.5f
            };
            sensor.SkeletonStream.Enable(parameters);

            //sensor.SkeletonStream.Enable();

            sensor.AllFramesReady += new
EventHandler<AllFramesReadyEventArgs>(sensor_AllFramesReady);

sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);


sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);

            try
            {
```

```
                sensor.Start();
            }
            catch (System.IO.IOException)
            {
                kinectSensorChooser1.AppConflictOccurred();
            }
        }

        void sensor_AllFramesReady(object sender, AllFramesReadyEventArgs e)
        {
            if (closing)
            {
                return;
            }

            //Get a skeleton
            Skeleton first = GetFirstSkeleton(e);

            if (first == null)
            {
                return;
            }

            GetCameraPoint(first, e);

            //set scaled position
            // ScalePosition(headImage, first.Joints[JointType.Head]);
            //ScalePosition(leftEllipse, first.Joints[JointType.HandLeft]);
            ScalePosition(RightHand, first.Joints[JointType.HandRight]);


        }

        void GetCameraPoint(Skeleton first, AllFramesReadyEventArgs e)
        {

            using (DepthImageFrame depth = e.OpenDepthImageFrame())
            {
                if (depth == null ||
                    kinectSensorChooser1.Kinect == null)
                {
                    return;
                }


                //Map a joint location to a point on the depth map
                //head
                DepthImagePoint headDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.Head].Position);

                DepthImagePoint bodyDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.Spine].Position);
                //left hand
                DepthImagePoint leftDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.HandLeft].Position);
                //right hand
                DepthImagePoint rightDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.HandRight].Position);
```

```
                //right leg
                DepthImagePoint rlegDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.FootRight].Position);

                //Map a depth point to a point on the color image
                //head
                ColorImagePoint headColorPoint =
                    depth.MapToColorImagePoint(headDepthPoint.X,
headDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //body
                ColorImagePoint bodyColorPoint =
                    depth.MapToColorImagePoint(bodyDepthPoint.X,
bodyDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //left hand
                ColorImagePoint leftColorPoint =
                    depth.MapToColorImagePoint(leftDepthPoint.X,
leftDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //right hand
                ColorImagePoint rightColorPoint =
                    depth.MapToColorImagePoint(rightDepthPoint.X,
rightDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);

                //right leg
                ColorImagePoint rlegColorPoint =
                   depth.MapToColorImagePoint(rlegDepthPoint.X,
rlegDepthPoint.Y,
                   ColorImageFormat.RgbResolution640x480Fps30);

                //Set location
                //CameraPosition(headImage, headColorPoint);
                //CameraPosition(bodyImage, bodyColorPoint);
                //CameraPosition(leftHand, leftColorPoint);
                //CameraPosition(rightLeg, rlegColorPoint);

                CameraPosition(RightHand, rightColorPoint);

                //CheckButton(hoverButton1, RightHand);
                //CheckButton(hoverButton2, RightHand);
                //CheckButton(hoverButton3, RightHand);
                //CheckButton(hoverButton4, RightHand);
                CheckButton(hoverButton5, RightHand);
                CheckButton(hoverButton6, RightHand);

            }
        }


        Skeleton GetFirstSkeleton(AllFramesReadyEventArgs e)
        {
            using (SkeletonFrame skeletonFrameData = e.OpenSkeletonFrame())
            {
                if (skeletonFrameData == null)
                {
                    return null;
                }
```

```
                skeletonFrameData.CopySkeletonDataTo(allSkeletons);

                //get the first tracked skeleton
                Skeleton first = (from s in allSkeletons
                                  where s.TrackingState ==
SkeletonTrackingState.Tracked
                                  select s).FirstOrDefault();

                return first;



            }
        }


        private void btnangle_Click(object sender, RoutedEventArgs e)
        {
            if (kinectSensorChooser1.Kinect.ElevationAngle !=
(int)slider1.Value)
            {
                kinectSensorChooser1.Kinect.ElevationAngle =
(int)slider1.Value;
            }

            //if (runtime.NuiCamera.ElevationAngle != (int)slider1.Value)
            //{
            //    runtime.NuiCamera.ElevationAngle = (int)slider1.Value;
            //}

        }

        private void slider1_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
        {
            int n = (int)slider1.Value;

            Degree.Content = n.ToString();

        }

        private void StopKinect(KinectSensor sensor)
        {
            if (sensor != null)
            {
                if (sensor.IsRunning)
                {
                    //stop sensor
                    sensor.Stop();

                    //stop audio if not null
                    if (sensor.AudioSource != null)
                    {
                        sensor.AudioSource.Stop();
                    }


                }
            }
        }
```

```
        public void CameraPosition(FrameworkElement element, ColorImagePoint
point)
        {
            //Divide by 2 for width and height so point is right in the middle
            // instead of in top/left corner
            Canvas.SetLeft(element, point.X - element.Width / 2);
            Canvas.SetTop(element, point.Y - element.Height / 2);

        }

        private static void CheckButton(HoverButton button, Ellipse
thumbStick)
        {


            if (IsItemMidpointInContainer(button, thumbStick))
            {
                button.Hovering();
            }
            else
            {
                button.Release();
            }
        }


        public static bool IsItemMidpointInContainer(FrameworkElement
container, FrameworkElement target)
        {
            FindValues(container, target);

            if (_itemTop < _topBoundary || _bottomBoundary < _itemTop)
            {
                //Midpoint of target is outside of top or bottom
                return false;
            }

            if (_itemLeft < _leftBoundary || _rightBoundary < _itemLeft)
            {
                //Midpoint of target is outside of left or right
                return false;
            }

            return true;
        }

        private static void FindValues(FrameworkElement container,
FrameworkElement target)
        {
            var containerTopLeft = container.PointToScreen(new Point());
            var itemTopLeft = target.PointToScreen(new Point());

            _topBoundary = containerTopLeft.Y;
            _bottomBoundary = _topBoundary + container.ActualHeight;
            _leftBoundary = containerTopLeft.X;
            _rightBoundary = _leftBoundary + container.ActualWidth;

            //use midpoint of item (width or height divided by 2)
            _itemLeft = itemTopLeft.X + (target.ActualWidth / 2);
            _itemTop = itemTopLeft.Y + (target.ActualHeight / 2);
        }
```

```
        private void ScalePosition(FrameworkElement element, Joint joint)
        {
            //convert the value to X/Y
            //Joint scaledJoint = joint.ScaleTo(1280, 720);

            //convert & scale (.3 = means 1/3 of joint distance)
            Joint scaledJoint = joint.ScaleTo(1280, 720, .3f, .3f);

            Canvas.SetLeft(element, scaledJoint.Position.X);
            Canvas.SetTop(element, scaledJoint.Position.Y);

        }


        private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            closing = true;
            StopKinect(kinectSensorChooser1.Kinect);
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            MixMatch headMatch = new MixMatch();

            this.Hide();

            Application.Current.Windows[0].Close();
            headMatch.Show();
            headMatch.Activate();
        }




    }
}
```

Extra XAML Code

```
<Window x:Class="KinectQuiz.Extra"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="KinectMain" Height="685" Width="950" Loaded="Window_Loaded"
        xmlns:Controls="clr-
namespace:Coding4Fun.Kinect.Wpf.Controls;assembly=Coding4Fun.Kinect.Wpf"
        xmlns:my="clr-
namespace:Microsoft.Samples.Kinect.WpfViewers;assembly=Microsoft.Samples.Kinec
t.WpfViewers"
        Closing="Window_Closing"
Icon="/KinectQuiz;component/Images/Caillou2.png">
    <Grid x:Name="theGrid" Height="628" Width="900">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="649*" />
            <ColumnDefinition Width="319*" />
```

```
            <ColumnDefinition Width="0*" />
        </Grid.ColumnDefinitions>
        <Canvas Background="Transparent" Height="630" VerticalAlignment="Top"
Grid.ColumnSpan="2" Margin="0,-2,0,0">
            <my:KinectColorViewer Canvas.Left="0" Canvas.Top="0"
Name="kinectColorViewer1" Height="480" Width="640" Kinect="{Binding
ElementName=kinectSensorChooser1, Path=Kinect}" />
            <Slider Canvas.Left="332" Canvas.Top="595" Height="29"
Name="slider1" Width="212" Maximum="27" Minimum="-27"
ValueChanged="slider1_ValueChanged" />
            <Label Canvas.Left="548" Canvas.Top="593" Content="0" Height="32"
Name="Degree" Width="30" />
            <Button Canvas.Left="178" Canvas.Top="593" Content="Adjust Kinect
Angle" Height="31" Name="btnangle" Width="148" Click="btnangle_Click" />
            <my:KinectSensorChooser Canvas.Left="117" Canvas.Top="137"
Name="kinectSensorChooser1" Width="396" Height="236" />
            <Ellipse Height="71" Width="73" Name="RightHand" Canvas.Left="453"
Canvas.Top="32">
                <Ellipse.Fill>
                    <ImageBrush
ImageSource="/KinectQuiz;component/Images/hand.png" />
                </Ellipse.Fill>
            </Ellipse>
            <Image Canvas.Left="709" Canvas.Top="369" Height="255"
Name="image4" Stretch="Fill" Width="185"
Source="/KinectQuiz;component/Images/Caillou2.png" />
            <my:KinectDepthViewer Canvas.Left="6" Canvas.Top="473"
Name="kinectDepthViewer1" Height="151" Width="167" Kinect="{Binding
ElementName=kinectSensorChooser1, Path=Kinect}" />
            <Image Canvas.Left="10" Canvas.Top="10" Height="83" Name="image1"
Stretch="Fill" Width="123" Source="/KinectQuiz;component/Images/Back.png" />
            <Controls:HoverButton Canvas.Left="10" Canvas.Top="10" Height="83"
ImageSize="146" Name="hoverButton5" TimeInterval="1000" Width="123" />
            <Image Canvas.Left="34" Canvas.Top="265" Height="61"
Name="leftHand" Stretch="Fill" Width="67"
Source="/KinectQuiz;component/Images/hand1.png" />
            <Image Canvas.Left="557" Canvas.Top="265" Height="61" Name="hhand"
Stretch="Fill" Width="66" Source="/KinectQuiz;component/Images/hand2.png" />
            <Image Canvas.Left="253" Canvas.Top="109" Height="109" Name="body"
Stretch="Fill" Width="118" Source="/KinectQuiz;component/Images/Untitled-
2.png" />
        </Canvas>
    </Grid>
</Window>
```

Extra Code

```
// (c) Copyright Microsoft Corporation.
// This source is subject to the Microsoft Public License (Ms-PL).
// Please see http://go.microsoft.com/fwlink/?LinkID=131993 for details.
// All other rights reserved.
//This project is created by Mohd Hazwan b Sahabuzan 12019
//but some of the code and resources are taken, modify and used from tutorial
websites
//such as http://channel9.msdn.com/Series/KinectQuickstart and
http://raychambers.wordpress.com

using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;

using System.Media;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using Coding4Fun.Kinect.Wpf;
using Coding4Fun.Kinect.Wpf.Controls;
using System.IO;



namespace KinectQuiz
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class Extra : Window
    {

        private static double _topBoundary;
        private static double _bottomBoundary;
        private static double _leftBoundary;
        private static double _rightBoundary;
        private static double _itemLeft;
        private static double _itemTop;



        public Extra()
        {
            InitializeComponent();


        }

        bool closing = false;
        const int skeletonCount = 6;
        Skeleton[] allSkeletons = new Skeleton[skeletonCount];

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            kinectSensorChooser1.KinectSensorChanged += new
DependencyPropertyChangedEventHandler(kinectSensorChooser1_KinectSensorChanged
);


            hoverButton5.Click += new RoutedEventHandler(hoverButton5_Click);
        }

        void hoverButton5_Click(object sender, RoutedEventArgs e)
        {
            Main1 MainForm = new Main1();
```

```
            this.Hide();

            Application.Current.Windows[0].Close();
            MainForm.Show();
            MainForm.Activate();
        }


        void kinectSensorChooser1_KinectSensorChanged(object sender,
DependencyPropertyChangedEventArgs e)
        {
            KinectSensor old = (KinectSensor)e.OldValue;

            StopKinect(old);

            KinectSensor sensor = (KinectSensor)e.NewValue;

            if (sensor == null)
            {
                return;
            }


            var parameters = new TransformSmoothParameters
            {
                Smoothing = 0.3f,
                Correction = 0.0f,
                Prediction = 0.0f,
                JitterRadius = 1.0f,
                MaxDeviationRadius = 0.5f
            };
            sensor.SkeletonStream.Enable(parameters);

            //sensor.SkeletonStream.Enable();

            sensor.AllFramesReady += new
EventHandler<AllFramesReadyEventArgs>(sensor_AllFramesReady);

sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);


sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);

            try
            {
                sensor.Start();
            }
            catch (System.IO.IOException)
            {
                kinectSensorChooser1.AppConflictOccurred();
            }
        }

        void sensor_AllFramesReady(object sender, AllFramesReadyEventArgs e)
        {
            if (closing)
            {
                return;
            }

            //Get a skeleton
            Skeleton first = GetFirstSkeleton(e);
```

```
            if (first == null)
            {
                return;
            }

            GetCameraPoint(first, e);

            //set scaled position
            // ScalePosition(headImage, first.Joints[JointType.Head]);
            //ScalePosition(leftEllipse, first.Joints[JointType.HandLeft]);
            ScalePosition(RightHand, first.Joints[JointType.HandRight]);


        }

        void GetCameraPoint(Skeleton first, AllFramesReadyEventArgs e)
        {

            using (DepthImageFrame depth = e.OpenDepthImageFrame())
            {
                if (depth == null ||
                    kinectSensorChooser1.Kinect == null)
                {
                    return;
                }


                //Map a joint location to a point on the depth map
                //head
                DepthImagePoint headDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.Head].Position);

                DepthImagePoint bodyDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.Spine].Position);
                //left hand
                DepthImagePoint leftDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.HandLeft].Position);
                //right hand
                DepthImagePoint rightDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.HandRight].Position);

                //right leg
                DepthImagePoint rlegDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.FootRight].Position);

                //Map a depth point to a point on the color image
                //head
                ColorImagePoint headColorPoint =
                    depth.MapToColorImagePoint(headDepthPoint.X,
headDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //body
                ColorImagePoint bodyColorPoint =
                    depth.MapToColorImagePoint(bodyDepthPoint.X,
bodyDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
```

```
                //left hand
                ColorImagePoint leftColorPoint =
                    depth.MapToColorImagePoint(leftDepthPoint.X,
leftDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //right hand
                ColorImagePoint rightColorPoint =
                    depth.MapToColorImagePoint(rightDepthPoint.X,
rightDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);

                //right leg
                ColorImagePoint rlegColorPoint =
                   depth.MapToColorImagePoint(rlegDepthPoint.X,
rlegDepthPoint.Y,
                   ColorImageFormat.RgbResolution640x480Fps30);

                //Set location

                CameraPosition(body, bodyColorPoint);
                CameraPosition(leftHand, leftColorPoint);

                CameraPosition(RightHand, rightColorPoint);
                CameraPosition(hhand, rightColorPoint);


                CheckButton(hoverButton5, RightHand);

            }
        }


        Skeleton GetFirstSkeleton(AllFramesReadyEventArgs e)
        {
            using (SkeletonFrame skeletonFrameData = e.OpenSkeletonFrame())
            {
                if (skeletonFrameData == null)
                {
                    return null;
                }


                skeletonFrameData.CopySkeletonDataTo(allSkeletons);

                //get the first tracked skeleton
                Skeleton first = (from s in allSkeletons
                                  where s.TrackingState ==
SkeletonTrackingState.Tracked
                                  select s).FirstOrDefault();

                return first;



            }
        }


        private void btnangle_Click(object sender, RoutedEventArgs e)
        {
            if (kinectSensorChooser1.Kinect.ElevationAngle !=
(int)slider1.Value)
```

```
            {
                kinectSensorChooser1.Kinect.ElevationAngle =
(int)slider1.Value;
            }

            //if (runtime.NuiCamera.ElevationAngle != (int)slider1.Value)
            //{
            //    runtime.NuiCamera.ElevationAngle = (int)slider1.Value;
            //}

        }

        private void slider1_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
        {
            int n = (int)slider1.Value;

            Degree.Content = n.ToString();

        }

        private void StopKinect(KinectSensor sensor)
        {
            if (sensor != null)
            {
                if (sensor.IsRunning)
                {
                    //stop sensor
                    sensor.Stop();

                    //stop audio if not null
                    if (sensor.AudioSource != null)
                    {
                        sensor.AudioSource.Stop();
                    }


                }
            }
        }

        public void CameraPosition(FrameworkElement element, ColorImagePoint
point)
        {
            //Divide by 2 for width and height so point is right in the middle
            // instead of in top/left corner
            Canvas.SetLeft(element, point.X - element.Width / 2);
            Canvas.SetTop(element, point.Y - element.Height / 2);

        }

        private static void CheckButton(HoverButton button, Ellipse
thumbStick)
        {


            if (IsItemMidpointInContainer(button, thumbStick))
            {
                button.Hovering();
            }
            else
            {
```

```
                button.Release();
            }
        }


        public static bool IsItemMidpointInContainer(FrameworkElement
container, FrameworkElement target)
        {
            FindValues(container, target);

            if (_itemTop < _topBoundary || _bottomBoundary < _itemTop)
            {
                //Midpoint of target is outside of top or bottom
                return false;
            }

            if (_itemLeft < _leftBoundary || _rightBoundary < _itemLeft)
            {
                //Midpoint of target is outside of left or right
                return false;
            }

            return true;
        }

        private static void FindValues(FrameworkElement container,
FrameworkElement target)
        {
            var containerTopLeft = container.PointToScreen(new Point());
            var itemTopLeft = target.PointToScreen(new Point());

            _topBoundary = containerTopLeft.Y;
            _bottomBoundary = _topBoundary + container.ActualHeight;
            _leftBoundary = containerTopLeft.X;
            _rightBoundary = _leftBoundary + container.ActualWidth;

            //use midpoint of item (width or height divided by 2)
            _itemLeft = itemTopLeft.X + (target.ActualWidth / 2);
            _itemTop = itemTopLeft.Y + (target.ActualHeight / 2);
        }


        private void ScalePosition(FrameworkElement element, Joint joint)
        {
            //convert the value to X/Y
            //Joint scaledJoint = joint.ScaleTo(1280, 720);

            //convert & scale (.3 = means 1/3 of joint distance)
            Joint scaledJoint = joint.ScaleTo(1280, 720, .3f, .3f);

            Canvas.SetLeft(element, scaledJoint.Position.X);
            Canvas.SetTop(element, scaledJoint.Position.Y);

        }


        private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            closing = true;
            StopKinect(kinectSensorChooser1.Kinect);
        }
```

```
    }
}
```

Body XAML Code

```
<Window x:Class="KinectQuiz.Body"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="KinectMain" Height="685" Width="950" Loaded="Window_Loaded"
        xmlns:Controls="clr-
namespace:Coding4Fun.Kinect.Wpf.Controls;assembly=Coding4Fun.Kinect.Wpf"
        xmlns:my="clr-
namespace:Microsoft.Samples.Kinect.WpfViewers;assembly=Microsoft.Samples.Kinec
t.WpfViewers"
        Closing="Window_Closing"
Icon="/KinectQuiz;component/Images/Caillou2.png">
    <Grid x:Name="theGrid" Height="628" Width="900">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="649*" />
            <ColumnDefinition Width="319*" />
            <ColumnDefinition Width="0*" />
        </Grid.ColumnDefinitions>
        <Canvas Background="Transparent" Height="630" VerticalAlignment="Top"
Grid.ColumnSpan="2" Margin="0,-2,0,0">
            <my:KinectColorViewer Canvas.Left="0" Canvas.Top="0"
Name="kinectColorViewer1" Height="480" Width="640" Kinect="{Binding
ElementName=kinectSensorChooser1, Path=Kinect}" />
            <Slider Canvas.Left="332" Canvas.Top="595" Height="29"
Name="slider1" Width="212" Maximum="27" Minimum="-27"
ValueChanged="slider1_ValueChanged" />
            <Label Canvas.Left="548" Canvas.Top="593" Content="0" Height="32"
Name="Degree" Width="30" />
            <Button Canvas.Left="178" Canvas.Top="593" Content="Adjust Kinect
Angle" Height="31" Name="btnangle" Width="148" Click="btnangle_Click" />
            <my:KinectSensorChooser Canvas.Left="148" Canvas.Top="126"
Name="kinectSensorChooser1" Width="396" Height="236" />
            <Ellipse Height="71" Width="73" Name="RightHand" Canvas.Left="453"
Canvas.Top="32">
                <Ellipse.Fill>
                    <ImageBrush
ImageSource="/KinectQuiz;component/Images/hand.png" />
                </Ellipse.Fill>
            </Ellipse>
            <Image Canvas.Left="709" Canvas.Top="369" Height="255"
Name="image4" Stretch="Fill" Width="185"
Source="/KinectQuiz;component/Images/Caillou2.png" />
            <my:KinectDepthViewer Canvas.Left="6" Canvas.Top="473"
Name="kinectDepthViewer1" Height="151" Width="167" Kinect="{Binding
ElementName=kinectSensorChooser1, Path=Kinect}" />
            <Image Canvas.Left="787" Canvas.Top="6" Height="50"
Name="headImage" Stretch="Fill" Width="100"
Source="/KinectQuiz;component/Images/HeadText2.png" />
```

```
            <Image Canvas.Left="787" Canvas.Top="101" Height="50"
Name="bodyImage" Stretch="Fill" Width="100"
Source="/KinectQuiz;component/Images/Untitled-11.png" />
            <Image Canvas.Left="787" Canvas.Top="53" Height="50"
Name="leftHand" Stretch="Fill" Width="100"
Source="/KinectQuiz;component/Images/HandText.png" />
            <Image Canvas.Left="787" Canvas.Top="149" Height="50"
Name="rightLeg" Stretch="Fill" Width="100"
Source="/KinectQuiz;component/Images/legtxt.png" />
            <Image Canvas.Left="703" Canvas.Top="256" Height="106"
Name="image2" Stretch="Fill" Width="191"
Source="/KinectQuiz;component/Images/Exercise.png" />
            <Controls:HoverButton Canvas.Left="787" Canvas.Top="6" Height="50"
ImageSize="146" Name="hoverButton1" TimeInterval="1000" Width="100" />
            <Controls:HoverButton Canvas.Left="787" Canvas.Top="149"
Height="50" ImageSize="146" Name="hoverButton4" TimeInterval="1000"
Width="100" />
            <Controls:HoverButton Canvas.Left="787" Canvas.Top="53"
Height="50" ImageSize="146" Name="hoverButton2" TimeInterval="1000"
Width="100" />
            <Controls:HoverButton Canvas.Left="787" Canvas.Top="101"
Height="50" ImageSize="146" Name="hoverButton3" TimeInterval="1000"
Width="100" />
            <Image Canvas.Left="10" Canvas.Top="10" Height="83" Name="image1"
Stretch="Fill" Width="123" Source="/KinectQuiz;component/Images/Back.png" />
            <Controls:HoverButton Canvas.Left="10" Canvas.Top="10" Height="83"
ImageSize="146" Name="hoverButton5" TimeInterval="1000" Width="123" />
            <Controls:HoverButton Canvas.Left="703" Canvas.Top="256"
Height="106" ImageSize="146" Name="hoverButton6" TimeInterval="1000"
Width="191" />
        </Canvas>
    </Grid>
</Window>
```

Body Code

```
// (c) Copyright Microsoft Corporation.
// This source is subject to the Microsoft Public License (Ms-PL).
// Please see http://go.microsoft.com/fwlink/?LinkID=131993 for details.
// All other rights reserved.
//This project is created by Mohd Hazwan b Sahabuzan 12019
//but some of the code and resources are taken, modify and used from tutorial
websites
//such as http://channel9.msdn.com/Series/KinectQuickstart and
http://raychambers.wordpress.com

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.Media;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
```

```
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using Coding4Fun.Kinect.Wpf;
using Coding4Fun.Kinect.Wpf.Controls;
using System.IO;




namespace KinectQuiz
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class Body : Window
    {

        private static double _topBoundary;
        private static double _bottomBoundary;
        private static double _leftBoundary;
        private static double _rightBoundary;
        private static double _itemLeft;
        private static double _itemTop;



        public Body()
        {
            InitializeComponent();



            //hoverButton10.Click += new
RoutedEventHandler(hoverButton10_Click);
            //hoverButton1.Click += new
RoutedEventHandler(hoverButton1_Click);
            //hoverButton2.Click += new
RoutedEventHandler(hoverButton2_Click);

        }




        bool closing = false;
        const int skeletonCount = 6;
        Skeleton[] allSkeletons = new Skeleton[skeletonCount];

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            kinectSensorChooser1.KinectSensorChanged += new
DependencyPropertyChangedEventHandler(kinectSensorChooser1_KinectSensorChanged
);

            hoverButton1.Click +=new RoutedEventHandler(hoverButton1_Click);
            hoverButton2.Click +=new RoutedEventHandler(hoverButton2_Click);
            hoverButton3.Click += new RoutedEventHandler(hoverButton3_Click);
            hoverButton4.Click += new RoutedEventHandler(hoverButton4_Click);
            hoverButton5.Click += new RoutedEventHandler(hoverButton5_Click);
```

```
            hoverButton6.Click += new RoutedEventHandler(hoverButton6_Click);
        }

        void hoverButton6_Click(object sender, RoutedEventArgs e)
        {
            BodyMix BodyM = new BodyMix();

            this.Hide();

            Application.Current.Windows[0].Close();
            BodyM.Show();
            BodyM.Activate();
        }

        void hoverButton5_Click(object sender, RoutedEventArgs e)
        {
            Main1 MainForm = new Main1();
            this.Hide();

            Application.Current.Windows[0].Close();
            MainForm.Show();
            MainForm.Activate();
        }

        void hoverButton4_Click(object sender, RoutedEventArgs e)
        {
            //CameraPosition(headImage, headColorPoint);
            //CameraPosition(bodyImage, bodyColorPoint);
            //CameraPosition(leftHand, leftColorPoint);
            //CameraPosition(rightLeg, rlegColorPoint);
        }

        void hoverButton3_Click(object sender, RoutedEventArgs e)
        {
            //throw new NotImplementedException();
        }

        void hoverButton1_Click(object sender, RoutedEventArgs e)
        {

        }

        void hoverButton2_Click(object sender, RoutedEventArgs e)
        {

        }

        void hoverButton10_Click(object sender, RoutedEventArgs e)
        {
            //MixMatch MixFrm = new MixMatch();
            //MainWindow MainWin = new MainWindow();
            //StopKinect(kinectSensorChooser1.Kinect);
            //MixFrm.Show();
            //MainWin.Close();
        }

        void kinectSensorChooser1_KinectSensorChanged(object sender,
DependencyPropertyChangedEventArgs e)
        {
            KinectSensor old = (KinectSensor)e.OldValue;

            StopKinect(old);
```

```
            KinectSensor sensor = (KinectSensor)e.NewValue;

            if (sensor == null)
            {
                return;
            }



            var parameters = new TransformSmoothParameters
            {
                Smoothing = 0.3f,
                Correction = 0.0f,
                Prediction = 0.0f,
                JitterRadius = 1.0f,
                MaxDeviationRadius = 0.5f
            };
            sensor.SkeletonStream.Enable(parameters);

            //sensor.SkeletonStream.Enable();

            sensor.AllFramesReady += new
EventHandler<AllFramesReadyEventArgs>(sensor_AllFramesReady);

sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);


sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);

            try
            {
                sensor.Start();
            }
            catch (System.IO.IOException)
            {
                kinectSensorChooser1.AppConflictOccurred();
            }
        }

        void sensor_AllFramesReady(object sender, AllFramesReadyEventArgs e)
        {
            if (closing)
            {
                return;
            }

            //Get a skeleton
            Skeleton first = GetFirstSkeleton(e);

            if (first == null)
            {
                return;
            }

            GetCameraPoint(first, e);

            //set scaled position
           // ScalePosition(headImage, first.Joints[JointType.Head]);
            //ScalePosition(leftEllipse, first.Joints[JointType.HandLeft]);
            ScalePosition(RightHand, first.Joints[JointType.HandRight]);
```

```
        }

        void GetCameraPoint(Skeleton first, AllFramesReadyEventArgs e)
        {

            using (DepthImageFrame depth = e.OpenDepthImageFrame())
            {
                if (depth == null ||
                    kinectSensorChooser1.Kinect == null)
                {
                    return;
                }


                //Map a joint location to a point on the depth map
                //head
                DepthImagePoint headDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.Head].Position);

                DepthImagePoint bodyDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.Spine].Position);
                //left hand
                DepthImagePoint leftDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.HandLeft].Position);
                //right hand
                DepthImagePoint rightDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.HandRight].Position);

                //right leg
                DepthImagePoint rlegDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.FootRight].Position);

                //Map a depth point to a point on the color image
                //head
                ColorImagePoint headColorPoint =
                    depth.MapToColorImagePoint(headDepthPoint.X,
headDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //body
                ColorImagePoint bodyColorPoint =
                    depth.MapToColorImagePoint(bodyDepthPoint.X,
bodyDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //left hand
                ColorImagePoint leftColorPoint =
                    depth.MapToColorImagePoint(leftDepthPoint.X,
leftDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //right hand
                ColorImagePoint rightColorPoint =
                    depth.MapToColorImagePoint(rightDepthPoint.X,
rightDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);

                //right leg
```

```
                ColorImagePoint rlegColorPoint =
                   depth.MapToColorImagePoint(rlegDepthPoint.X,
rlegDepthPoint.Y,
                   ColorImageFormat.RgbResolution640x480Fps30);

                //Set location
                CameraPosition(headImage, headColorPoint);
                CameraPosition(bodyImage, bodyColorPoint);
                CameraPosition(leftHand, leftColorPoint);
                CameraPosition(rightLeg, rlegColorPoint);

                CameraPosition(RightHand, rightColorPoint);

                CheckButton(hoverButton1, RightHand);
                CheckButton(hoverButton2, RightHand);
                CheckButton(hoverButton3, RightHand);
                CheckButton(hoverButton4, RightHand);
                CheckButton(hoverButton5, RightHand);
                CheckButton(hoverButton6, RightHand);

            }
        }


        Skeleton GetFirstSkeleton(AllFramesReadyEventArgs e)
        {
            using (SkeletonFrame skeletonFrameData = e.OpenSkeletonFrame())
            {
                if (skeletonFrameData == null)
                {
                    return null;
                }


                skeletonFrameData.CopySkeletonDataTo(allSkeletons);

                //get the first tracked skeleton
                Skeleton first = (from s in allSkeletons
                                  where s.TrackingState ==
SkeletonTrackingState.Tracked
                                  select s).FirstOrDefault();

                return first;



            }
        }


        private void btnangle_Click(object sender, RoutedEventArgs e)
        {
            if (kinectSensorChooser1.Kinect.ElevationAngle !=
(int)slider1.Value)
            {
                kinectSensorChooser1.Kinect.ElevationAngle =
(int)slider1.Value;
            }

            //if (runtime.NuiCamera.ElevationAngle != (int)slider1.Value)
            //{
            //    runtime.NuiCamera.ElevationAngle = (int)slider1.Value;
```

```
            //}

        }

        private void slider1_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
        {
            int n = (int)slider1.Value;

            Degree.Content = n.ToString();

        }

        private void StopKinect(KinectSensor sensor)
        {
            if (sensor != null)
            {
                if (sensor.IsRunning)
                {
                    //stop sensor
                    sensor.Stop();

                    //stop audio if not null
                    if (sensor.AudioSource != null)
                    {
                        sensor.AudioSource.Stop();
                    }


                }
            }
        }

        public void CameraPosition(FrameworkElement element, ColorImagePoint
point)
        {
            //Divide by 2 for width and height so point is right in the middle
            // instead of in top/left corner
            Canvas.SetLeft(element, point.X - element.Width / 2);
            Canvas.SetTop(element, point.Y - element.Height / 2);

        }

        private static void CheckButton(HoverButton button, Ellipse
thumbStick)
        {


            if (IsItemMidpointInContainer(button, thumbStick))
            {
                button.Hovering();
            }
            else
            {
                button.Release();
            }
        }


        public static bool IsItemMidpointInContainer(FrameworkElement
container, FrameworkElement target)
        {
```

```
            FindValues(container, target);

            if (_itemTop < _topBoundary || _bottomBoundary < _itemTop)
            {
                //Midpoint of target is outside of top or bottom
                return false;
            }

            if (_itemLeft < _leftBoundary || _rightBoundary < _itemLeft)
            {
                //Midpoint of target is outside of left or right
                return false;
            }

            return true;
        }

        private static void FindValues(FrameworkElement container,
FrameworkElement target)
        {
            var containerTopLeft = container.PointToScreen(new Point());
            var itemTopLeft = target.PointToScreen(new Point());

            _topBoundary = containerTopLeft.Y;
            _bottomBoundary = _topBoundary + container.ActualHeight;
            _leftBoundary = containerTopLeft.X;
            _rightBoundary = _leftBoundary + container.ActualWidth;

            //use midpoint of item (width or height divided by 2)
            _itemLeft = itemTopLeft.X + (target.ActualWidth / 2);
            _itemTop = itemTopLeft.Y + (target.ActualHeight / 2);
        }


        private void ScalePosition(FrameworkElement element, Joint joint)
        {
            //convert the value to X/Y
            //Joint scaledJoint = joint.ScaleTo(1280, 720);

            //convert & scale (.3 = means 1/3 of joint distance)
            Joint scaledJoint = joint.ScaleTo(1280, 720, .3f, .3f);

            Canvas.SetLeft(element, scaledJoint.Position.X);
            Canvas.SetTop(element, scaledJoint.Position.Y);

        }


        private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            closing = true;
            StopKinect(kinectSensorChooser1.Kinect);
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            BodyMix BodyM = new BodyMix();

            this.Hide();

            Application.Current.Windows[0].Close();
```

```
            BodyM.Show();
            BodyM.Activate();
        }




    }
}
```

BodyMix XAML Code

```
<Window x:Class="KinectQuiz.BodyMix"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Kinect Mix and Match" Height="847" Width="991"
Loaded="Window_Loaded"
        xmlns:Controls="clr-
namespace:Coding4Fun.Kinect.Wpf.Controls;assembly=Coding4Fun.Kinect.Wpf"
        xmlns:my="clr-
namespace:Microsoft.Samples.Kinect.WpfViewers;assembly=Microsoft.Samples.Kinec
t.WpfViewers"
        Closing="Window_Closing"
Icon="/KinectQuiz;component/Images/Caillou2.png">
    <Grid x:Name="theGrid" Height="809">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="649*" />
            <ColumnDefinition Width="319*" />
            <ColumnDefinition Width="0*" />
        </Grid.ColumnDefinitions>
        <Canvas Background="Transparent" Height="808" VerticalAlignment="Top"
Grid.ColumnSpan="2" Margin="0,-2,0,0">
            <my:KinectColorViewer Canvas.Left="0" Canvas.Top="2"
Name="kinectColorViewer1" Height="809" Width="969" Kinect="{Binding
ElementName=kinectSensorChooser1, Path=Kinect}" />

            <TextBlock Canvas.Left="212" Canvas.Top="458" Height="23"
Name="textBlock1" Text="TextBlock" Foreground="White" Visibility="Hidden" />
            <Slider Canvas.Left="438" Canvas.Top="773" Height="29"
Name="slider1" Width="212" Maximum="27" Minimum="-27"
ValueChanged="slider1_ValueChanged" />
            <Label Canvas.Left="650" Canvas.Top="769" Content="0" Height="32"
Name="Degree" Width="30" />
            <Button Canvas.Left="293" Canvas.Top="770" Content="Adjust Kinect
Angle" Height="31" Name="btnangle" Width="148" Click="btnangle_Click" />
            <my:KinectSensorChooser Canvas.Left="-139" Canvas.Top="128"
Name="kinectSensorChooser1" Width="121" Height="64" />
            <Ellipse Canvas.Left="212" Canvas.Top="102" Height="75"
Name="ellipse1" Stroke="Black" Width="75" Fill="#FF50FF00" />
            <TextBlock Background="#FFEB9797" Canvas.Left="0" Canvas.Top="102"
Height="110" Name="language1text" Text="TextBlock" Width="200"
TextAlignment="Center" FontSize="18" />
            <Ellipse Canvas.Left="212" Canvas.Top="228" Fill="#FF50FF00"
Height="75" Name="ellipse2" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="212" Canvas.Top="370" Fill="#FF50FF00"
Height="75" Name="ellipse3" Stroke="Black" Width="75" />
```

```
            <Ellipse Canvas.Left="212" Canvas.Top="499" Fill="#FF50FF00"
Height="75" Name="ellipse4" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="212" Canvas.Top="640" Fill="#FF50FF00"
Height="75" Name="ellipse5" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="674" Canvas.Top="102" Fill="#FF50FF00"
Height="75" Name="ellipse6" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="680" Canvas.Top="228" Fill="#FF50FF00"
Height="75" Name="ellipse7" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="680" Canvas.Top="364" Fill="#FF50FF00"
Height="75" Name="ellipse8" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="680" Canvas.Top="499" Fill="#FF50FF00"
Height="75" Name="ellipse9" Stroke="Black" Width="75" />
            <Ellipse Canvas.Left="680" Canvas.Top="646" Fill="#FF50FF00"
Height="75" Name="ellipse10" Stroke="Black" Width="75" />
            <Controls:HoverButton Canvas.Left="212" Canvas.Top="101"
Height="75" ImageSize="78" Name="hoverButton11" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="212" Canvas.Top="227"
Height="75" ImageSize="78" Name="hoverButton1" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="212" Canvas.Top="370"
Height="75" ImageSize="78" Name="hoverButton2" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="212" Canvas.Top="499"
Height="75" ImageSize="78" Name="hoverButton3" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="212" Canvas.Top="642"
Height="75" ImageSize="78" Name="hoverButton4" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="674" Canvas.Top="102"
Height="75" ImageSize="78" Name="hoverButton5" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="674" Canvas.Top="227"
Height="75" ImageSize="78" Name="hoverButton6" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="680" Canvas.Top="364"
Height="75" ImageSize="78" Name="hoverButton7" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="680" Canvas.Top="499"
Height="75" ImageSize="78" Name="hoverButton8" TimeInterval="1000" Width="75"
/>
            <Controls:HoverButton Canvas.Left="680" Canvas.Top="646"
Height="75" ImageSize="78" Name="hoverButton9" TimeInterval="1000" Width="75"
/>
            <TextBlock Background="#FFEB9797" Canvas.Left="1" Canvas.Top="228"
Height="110" Name="language2text" Text="TextBlock" Width="200"
TextAlignment="Center" FontSize="18" />
            <TextBlock Background="#FFEB9797" Canvas.Left="0" Canvas.Top="364"
Height="110" Name="language3text" Text="TextBlock" Width="200"
TextAlignment="Center" FontSize="18" />
            <TextBlock Background="#FFEB9797" Canvas.Left="1" Canvas.Top="499"
Height="110" Name="language4text" Text="TextBlock" Width="200"
TextAlignment="Center" FontSize="18" />
            <TextBlock Background="#FFEB9797" Canvas.Left="1" Canvas.Top="640"
Height="110" Name="language5text" Text="TextBlock" Width="200"
TextAlignment="Center" FontSize="18" />
            <TextBlock Background="#FFEB9797" Canvas.Left="769"
Canvas.Top="102" Height="110" Name="answer1text" Text="TextBlock" Width="200"
FontSize="32" TextAlignment="Center" FontStretch="SemiExpanded" />
            <TextBlock Background="#FFEB9797" Canvas.Left="769"
Canvas.Top="228" Height="110" Name="answer2text" Text="TextBlock" Width="200"
FontSize="32" TextAlignment="Center" FontStretch="SemiExpanded" />
```

```
            <TextBlock Background="#FFEB9797" Canvas.Left="769"
Canvas.Top="364" Height="110" Name="answer3text" Text="TextBlock" Width="200"
FontSize="32" TextAlignment="Center" FontStretch="SemiExpanded" />
            <TextBlock Background="#FFEB9797" Canvas.Left="769"
Canvas.Top="499" Height="110" Name="answer4text" Text="TextBlock" Width="200"
FontSize="32" TextAlignment="Center" FontStretch="SemiExpanded" />
            <TextBlock Background="#FFEB9797" Canvas.Left="769"
Canvas.Top="642" Height="110" Name="answer5text" Text="TextBlock" Width="200"
FontSize="32" TextAlignment="Center" FontStretch="SemiExpanded" />
            <TextBlock Canvas.Left="212" Canvas.Top="102" Height="74"
Name="language1number" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="212" Canvas.Top="227" Height="74"
Name="language2number" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="212" Canvas.Top="365" Height="74"
Name="language3number" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="212" Canvas.Top="500" Height="74"
Name="language4number" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="680" Canvas.Top="102" Height="74"
Name="answernumber11" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="680" Canvas.Top="228" Height="74"
Name="answernumber22" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="680" Canvas.Top="370" Height="69"
Name="answernumber33" Text="TextBlock" Width="69" Visibility="Hidden" />
            <TextBlock Canvas.Left="680" Canvas.Top="500" Height="74"
Name="answernumber44" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="212" Canvas.Top="646" Height="74"
Name="language5number" Text="TextBlock" Width="75" Visibility="Hidden" />
            <TextBlock Canvas.Left="680" Canvas.Top="642" Height="74"
Name="answernumber55" Text="TextBlock" Width="75" Visibility="Hidden" />
            <Image Canvas.Left="438" Canvas.Top="309" Height="101"
Name="image2" Stretch="Fill" Width="62"
Source="/KinectQuiz;component/Images/arrow.png" />
            <TextBlock Canvas.Left="293" Canvas.Top="31" Height="60"
Name="textBlock2" Text="TextBlock" Width="74" Visibility="Hidden" />
            <TextBlock Canvas.Left="600" Canvas.Top="26" Height="65"
Name="textBlock3" Text="TextBlock" Width="75" Visibility="Hidden" />
            <Controls:HoverButton Canvas.Left="377" Canvas.Top="100"
Height="211" ImageSize="146" Name="hoverButton10" TimeInterval="1000"
Width="217" />
            <Ellipse Height="71" Width="73" Name="RightHand" Canvas.Left="21"
Canvas.Top="100">
                <Ellipse.Fill>
                    <ImageBrush
ImageSource="/KinectQuiz;component/Images/hand.png" />
                </Ellipse.Fill>
            </Ellipse>
            <Image Canvas.Left="375" Canvas.Top="112" Height="175"
Name="image1" Stretch="Fill" Width="200"
Source="/KinectQuiz;component/Images/button.png" />
            <Image Canvas.Left="51" Canvas.Top="668" Height="82" Name="image7"
Source="/KinectQuiz;component/Images/body.jpg" Stretch="Fill" Width="95" />
            <Image Canvas.Left="51" Canvas.Top="527" Height="82" Name="image3"
Stretch="Fill" Width="95" Source="/KinectQuiz;component/Images/shoulders1.jpg"
/>
            <Image Canvas.Left="51" Canvas.Top="130" Height="82" Name="image4"
Stretch="Fill" Width="95" Source="/KinectQuiz;component/Images/egg-head-
cartoon-hi1.png" />
            <Image Canvas.Left="51" Canvas.Top="392" Height="82" Name="image5"
Stretch="Fill" Width="95" Source="/KinectQuiz;component/Images/hand.jpg" />
            <Image Canvas.Left="51" Canvas.Top="254" Height="82" Name="image6"
Stretch="Fill" Width="95"
Source="/KinectQuiz;component/Images/6837799f_leg.jpg" />
```

```
            <Image Canvas.Left="234" Canvas.Top="2" Height="89" Name="image8"
Stretch="Fill" Width="499"
Source="/KinectQuiz;component/Images/Title%20Body.png" />
            <Image Canvas.Left="377" Canvas.Top="413" Height="61"
Name="image9" Stretch="Fill" Width="198"
Source="/KinectQuiz;component/Images/CheckAnswer.png" />
            <Image Canvas.Left="1" Canvas.Top="2" Height="41" Name="image10"
Stretch="Fill" Width="79" Source="/KinectQuiz;component/Images/Back.png" />
            <Controls:HoverButton Canvas.Left="2" Canvas.Top="2" Height="58"
ImageSize="146" Name="hoverButton12" TimeInterval="1000" Width="78" />
        </Canvas>
        <Image Height="175" Name="deed" Stretch="Fill" Width="202"
Source="/KinectQuiz;component/Images/button.jpg" Grid.Column="2"
Margin="20,160,-222,474" />
    </Grid>
</Window>
```

BodyMix Code

```
// (c) Copyright Microsoft Corporation.
// This source is subject to the Microsoft Public License (Ms-PL).
// Please see http://go.microsoft.com/fwlink/?LinkID=131993 for details.
// All other rights reserved.
//This project is created by Mohd Hazwan b Sahabuzan 12019
//but some of the code and resources are taken, modify and used from tutorial
websites
//such as http://channel9.msdn.com/Series/KinectQuickstart and
http://raychambers.wordpress.com

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.Media;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using Coding4Fun.Kinect.Wpf;
using Coding4Fun.Kinect.Wpf.Controls;
using System.IO;



namespace KinectQuiz
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
```

```
public partial class BodyMix : Window
{

    private static double _topBoundary;
    private static double _bottomBoundary;
    private static double _leftBoundary;
    private static double _rightBoundary;
    private static double _itemLeft;
    private static double _itemTop;

    string language1;
    string language2;
    string language3;
    string language4;
    string language5;
    //string language6;
    //string language7;
    //string language8;

    string languagenumber1;
    string languagenumber2;
    string languagenumber3;
    string languagenumber4;
    string languagenumber5;
    //string languagenumber6;
    //string languagenumber7;
    //string languagenumber8;

    string answer1;
    string answer2;
    string answer3;
    string answer4;
    string answer5;
    //string answer6;
    //string answer7;
    //string answer8;

    string answernumber1;
    string answernumber2;
    string answernumber3;
    string answernumber4;
    string answernumber5;
    //string answernumber6;
    //string answernumber7;
    //string answernumber8;

    int finallanguage1;
    int finalanswer1;

    public BodyMix()
    {
        InitializeComponent();

        finallanguage1 = 1;
        finalanswer1 = 2;

        TextReader tr = new StreamReader("BodyPic.txt");

        int NumberOfLines = 9;

        string[] ListLines = new string[NumberOfLines];
        for (int i = 1; i < NumberOfLines; i++)
```

```
{
    ListLines[i] = tr.ReadLine();
}
int b = 0;
{
    string[] parts = ListLines[1].Split(',');
    foreach (string part in parts)
    {
    }
    b++;
    language1 = parts[0];
    languagenumber1 = parts[1];
}
int c = 0;
{
    string[] parts = ListLines[2].Split(',');
    foreach (string part in parts)
    {
    }
    c++;
    language2 = parts[0];
    languagenumber2 = parts[1];
}

int g = 0;
{
    string[] parts = ListLines[3].Split(',');
    foreach (string part in parts)
    {
    }
    g++;
    language3 = parts[0];
    languagenumber3 = parts[1];
}

int h = 0;
{
    string[] parts = ListLines[4].Split(',');
    foreach (string part in parts)
    {
    }
    h++;
    language4 = parts[0];
    languagenumber4 = parts[1];
}

int j = 0;
{
    string[] parts = ListLines[5].Split(',');
    foreach (string part in parts)
    {
    }
    j++;
    language5 = parts[0];
    languagenumber5 = parts[1];
}

TextReader trr = new StreamReader("BodyText.txt");

int NumberOfLines2 = 9;

string[] ListLines2 = new string[NumberOfLines2];
```

```
for (int i = 1; i < NumberOfLines2; i++)
{
    ListLines2[i] = trr.ReadLine();
}
int d = 0;
{
    string[] parts = ListLines2[1].Split(',');
    foreach (string part in parts)
    {
    }
    d++;
    answer1 = parts[0];
    answernumber1 = parts[1];
}
int e = 0;
{
    string[] parts = ListLines2[2].Split(',');
    foreach (string part in parts)
    {
    }
    e++;
    answer2 = parts[0];
    answernumber2 = parts[1];
}

int f = 0;
{
    string[] parts = ListLines2[3].Split(',');
    foreach (string part in parts)
    {
    }
    f++;
    answer3 = parts[0];
    answernumber3 = parts[1];
}

int n = 0;
{
    string[] parts = ListLines2[4].Split(',');
    foreach (string part in parts)
    {
    }
    n++;
    answer4 = parts[0];
    answernumber4 = parts[1];
}

int o = 0;
{
    string[] parts = ListLines2[5].Split(',');
    foreach (string part in parts)
    {
    }
    o++;
    answer5 = parts[0];
    answernumber5 = parts[1];
}

// button1.Click += new RoutedEventHandler(button1_Click);
hoverButton1.Click += new RoutedEventHandler(hoverButton1_Click);
hoverButton2.Click += new RoutedEventHandler(hoverButton2_Click);
hoverButton3.Click += new RoutedEventHandler(hoverButton3_Click);
```

```
            hoverButton4.Click += new RoutedEventHandler(hoverButton4_Click);
            hoverButton5.Click += new RoutedEventHandler(hoverButton5_Click);
            hoverButton6.Click += new RoutedEventHandler(hoverButton6_Click);
            hoverButton7.Click += new RoutedEventHandler(hoverButton7_Click);
            hoverButton8.Click += new RoutedEventHandler(hoverButton8_Click);
            hoverButton9.Click += new RoutedEventHandler(hoverButton9_Click);
            hoverButton10.Click += new
RoutedEventHandler(hoverButton10_Click);
            hoverButton11.Click += new
RoutedEventHandler(hoverButton11_Click);
            hoverButton12.Click += new
RoutedEventHandler(hoverButton12_Click);
        }



        bool closing = false;
        const int skeletonCount = 6;
        Skeleton[] allSkeletons = new Skeleton[skeletonCount];

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            kinectSensorChooser1.KinectSensorChanged += new
DependencyPropertyChangedEventHandler(kinectSensorChooser1_KinectSensorChanged
);

            language1text.Text = language1;
            language1number.Text = languagenumber1;

            language2text.Text = language2;
            language2number.Text = languagenumber2;

            language3text.Text = language3;
            language3number.Text = languagenumber3;

            language4text.Text = language4;
            language4number.Text = languagenumber4;

            language5text.Text = language5;
            language5number.Text = languagenumber5;

            answer1text.Text = answer1;
            answernumber11.Text = answernumber1;

            answer2text.Text = answer2;
            answernumber22.Text = answernumber2;

            answer3text.Text = answer3;
            answernumber33.Text = answernumber3;

            answer4text.Text = answer4;
            answernumber44.Text = answernumber4;

            answer5text.Text = answer5;
            answernumber55.Text = answernumber5;

        }

        void hoverButton11_Click(object sender, RoutedEventArgs e)
        {
            ellipse1.Fill = new SolidColorBrush(Colors.Red);
```

```
            ellipse2.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse3.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse4.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse5.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finallanguage1 = Convert.ToInt32(language1number.Text);
            textBlock2.Text = "1";
        }

        void hoverButton1_Click(object sender, RoutedEventArgs e)
        {
            ellipse2.Fill = new SolidColorBrush(Colors.Red);
            ellipse1.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse3.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse4.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse5.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finallanguage1 = Convert.ToInt32(language2number.Text);
            textBlock2.Text = "2";
        }

        void hoverButton2_Click(object sender, RoutedEventArgs e)
        {
            ellipse3.Fill = new SolidColorBrush(Colors.Red);
            ellipse2.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse1.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse4.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse5.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finallanguage1 = Convert.ToInt32(language3number.Text);
            textBlock2.Text = "3";
        }

        void hoverButton3_Click(object sender, RoutedEventArgs e)
        {
            ellipse4.Fill = new SolidColorBrush(Colors.Red);
            ellipse2.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse3.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse1.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse5.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finallanguage1 = Convert.ToInt32(language4number.Text);
            textBlock2.Text = "4";
        }
        void hoverButton4_Click(object sender, RoutedEventArgs e)
        {
```

```
            ellipse5.Fill = new SolidColorBrush(Colors.Red);
            ellipse2.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse3.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse4.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse1.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finallanguage1 = Convert.ToInt32(language5number.Text);
            textBlock2.Text = "5";
        }

        void hoverButton5_Click(object sender, RoutedEventArgs e)
        {
            ellipse6.Fill = new SolidColorBrush(Colors.Red);
            ellipse7.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse8.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse9.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse10.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finalanswer1 = Convert.ToInt32(answernumber11.Text);
            textBlock3.Text = "1";
        }

        void hoverButton6_Click(object sender, RoutedEventArgs e)
        {
            ellipse7.Fill = new SolidColorBrush(Colors.Red);
            ellipse8.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse6.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse9.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse10.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finalanswer1 = Convert.ToInt32(answernumber22.Text);
            textBlock3.Text = "2";

        }

        void hoverButton7_Click(object sender, RoutedEventArgs e)
        {
            ellipse8.Fill = new SolidColorBrush(Colors.Red);
            ellipse7.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse6.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse9.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse10.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finalanswer1 = Convert.ToInt32(answernumber33.Text);
            textBlock3.Text = "3";
        }
```

```
        void hoverButton8_Click(object sender, RoutedEventArgs e)
        {
            ellipse9.Fill = new SolidColorBrush(Colors.Red);
            ellipse7.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse8.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse6.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse10.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finalanswer1 = Convert.ToInt32(answernumber44.Text);
            textBlock3.Text = "4";
        }

        void hoverButton9_Click(object sender, RoutedEventArgs e)
        {
            ellipse10.Fill = new SolidColorBrush(Colors.Red);
            ellipse7.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse8.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse9.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            ellipse6.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

            finalanswer1 = Convert.ToInt32(answernumber55.Text);
            textBlock3.Text = "5";

        }

        void hoverButton10_Click(object sender, RoutedEventArgs e)
        {
            if (finalanswer1 == finallanguage1)
            {
                //to reset all of the ellipse colors
                ellipse1.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse2.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse3.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse4.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse5.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse6.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse7.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse8.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse9.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse10.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));

                //to check which text block is being used and change its color
                if (textBlock2.Text == "1")
```

```
                {
                    language1text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock2.Text == "2")
                {
                    language2text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock2.Text == "3")
                {
                    language3text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock2.Text == "4")
                {
                    language4text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock2.Text == "5")
                {
                    language5text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }

                if (textBlock3.Text == "1")
                {
                    answer1text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock3.Text == "2")
                {
                    answer2text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock3.Text == "3")
                {
                    answer3text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock3.Text == "4")
                {
                    answer4text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }
                else if (textBlock3.Text == "5")
                {
                    answer5text.Background = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                }

            }
            else
            {
                //MessageBox.Show("Incorrect");
                // to reset the text block color to basic color
                ellipse1.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse2.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse3.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
```

```
                ellipse4.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse5.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse6.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse7.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse8.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse9.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
                ellipse10.Fill = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FF50FF00"));
            }
        }

        void hoverButton12_Click(object sender, RoutedEventArgs e)
        {
            Body MainBody = new Body();
            this.Hide();

            Application.Current.Windows[0].Close();
            MainBody.Show();
            MainBody.Activate();
            //throw new NotImplementedException();
        }


        void kinectSensorChooser1_KinectSensorChanged(object sender,
DependencyPropertyChangedEventArgs e)
        {
            KinectSensor old = (KinectSensor)e.OldValue;

            StopKinect(old);

            KinectSensor sensor = (KinectSensor)e.NewValue;

            if (sensor == null)
            {
                return;
            }




            var parameters = new TransformSmoothParameters
            {
                Smoothing = 0.3f,
                Correction = 0.0f,
                Prediction = 0.0f,
                JitterRadius = 1.0f,
                MaxDeviationRadius = 0.5f
            };
            sensor.SkeletonStream.Enable(parameters);

            //sensor.SkeletonStream.Enable();

            sensor.AllFramesReady += new
EventHandler<AllFramesReadyEventArgs>(sensor_AllFramesReady);

sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
```

```
sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);

            try
            {
                sensor.Start();
            }
            catch (System.IO.IOException)
            {
                kinectSensorChooser1.AppConflictOccurred();
            }
        }

        void sensor_AllFramesReady(object sender, AllFramesReadyEventArgs e)
        {
            if (closing)
            {
                return;
            }

            //Get a skeleton
            Skeleton first = GetFirstSkeleton(e);

            if (first == null)
            {
                return;
            }

            GetCameraPoint(first, e);

            //set scaled position
            //ScalePosition(headImage, first.Joints[JointType.Head]);
            //ScalePosition(leftEllipse, first.Joints[JointType.HandLeft]);
            ScalePosition(RightHand, first.Joints[JointType.HandRight]);







        }

        void GetCameraPoint(Skeleton first, AllFramesReadyEventArgs e)
        {

            using (DepthImageFrame depth = e.OpenDepthImageFrame())
            {
                if (depth == null ||
                    kinectSensorChooser1.Kinect == null)
                {
                    return;
                }


                //Map a joint location to a point on the depth map
                //head
                DepthImagePoint headDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.Head].Position);
                //left hand
                DepthImagePoint leftDepthPoint =
```

```
depth.MapFromSkeletonPoint(first.Joints[JointType.HandLeft].Position);
                //right hand
                DepthImagePoint rightDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.HandRight].Position);


                //Map a depth point to a point on the color image
                //head
                ColorImagePoint headColorPoint =
                    depth.MapToColorImagePoint(headDepthPoint.X,
headDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //left hand
                ColorImagePoint leftColorPoint =
                    depth.MapToColorImagePoint(leftDepthPoint.X,
leftDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //right hand
                ColorImagePoint rightColorPoint =
                    depth.MapToColorImagePoint(rightDepthPoint.X,
rightDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);


                //Set location
                //CameraPosition(headImage, headColorPoint);
                //CameraPosition(leftEllipse, leftColorPoint);
                CameraPosition(RightHand, rightColorPoint);

                CheckButton(hoverButton11, RightHand);
                CheckButton(hoverButton1, RightHand);
                CheckButton(hoverButton2, RightHand);
                CheckButton(hoverButton3, RightHand);
                CheckButton(hoverButton4, RightHand);
                CheckButton(hoverButton5, RightHand);
                CheckButton(hoverButton6, RightHand);
                CheckButton(hoverButton7, RightHand);
                CheckButton(hoverButton8, RightHand);
                CheckButton(hoverButton9, RightHand);
                CheckButton(hoverButton10, RightHand);
                CheckButton(hoverButton12, RightHand);



            }
        }


        Skeleton GetFirstSkeleton(AllFramesReadyEventArgs e)
        {
            using (SkeletonFrame skeletonFrameData = e.OpenSkeletonFrame())
            {
                if (skeletonFrameData == null)
                {
                    return null;
                }


                skeletonFrameData.CopySkeletonDataTo(allSkeletons);
```

```
                //get the first tracked skeleton
                Skeleton first = (from s in allSkeletons
                                  where s.TrackingState ==
SkeletonTrackingState.Tracked
                                  select s).FirstOrDefault();

                return first;



            }
        }


        private void btnangle_Click(object sender, RoutedEventArgs e)
        {
            if (kinectSensorChooser1.Kinect.ElevationAngle !=
(int)slider1.Value)
            {
                kinectSensorChooser1.Kinect.ElevationAngle =
(int)slider1.Value;
            }

            //if (runtime.NuiCamera.ElevationAngle != (int)slider1.Value)
            //{
            //    runtime.NuiCamera.ElevationAngle = (int)slider1.Value;
            //}

        }

        private void slider1_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
        {
            int n = (int)slider1.Value;

            Degree.Content = n.ToString();

        }

        private void StopKinect(KinectSensor sensor)
        {
            if (sensor != null)
            {
                if (sensor.IsRunning)
                {
                    //stop sensor
                    sensor.Stop();

                    //stop audio if not null
                    if (sensor.AudioSource != null)
                    {
                        sensor.AudioSource.Stop();
                    }


                }
            }
        }

        private void CameraPosition(FrameworkElement element, ColorImagePoint
point)
        {
```

```
            //Divide by 2 for width and height so point is right in the middle
            // instead of in top/left corner
            Canvas.SetLeft(element, point.X - element.Width / 2);
            Canvas.SetTop(element, point.Y - element.Height / 2);

        }

        private static void CheckButton(HoverButton button, Ellipse
thumbStick)
        {


            if (IsItemMidpointInContainer(button, thumbStick))
            {
                button.Hovering();
            }
            else
            {
                button.Release();
            }
        }


        public static bool IsItemMidpointInContainer(FrameworkElement
container, FrameworkElement target)
        {
            FindValues(container, target);

            if (_itemTop < _topBoundary || _bottomBoundary < _itemTop)
            {
                //Midpoint of target is outside of top or bottom
                return false;
            }

            if (_itemLeft < _leftBoundary || _rightBoundary < _itemLeft)
            {
                //Midpoint of target is outside of left or right
                return false;
            }

            return true;
        }

        private static void FindValues(FrameworkElement container,
FrameworkElement target)
        {
            var containerTopLeft = container.PointToScreen(new Point());
            var itemTopLeft = target.PointToScreen(new Point());

            _topBoundary = containerTopLeft.Y;
            _bottomBoundary = _topBoundary + container.ActualHeight;
            _leftBoundary = containerTopLeft.X;
            _rightBoundary = _leftBoundary + container.ActualWidth;

            //use midpoint of item (width or height divided by 2)
            _itemLeft = itemTopLeft.X + (target.ActualWidth / 2);
            _itemTop = itemTopLeft.Y + (target.ActualHeight / 2);
        }


        private void ScalePosition(FrameworkElement element, Joint joint)
        {
```

```
            //convert the value to X/Y
            //Joint scaledJoint = joint.ScaleTo(1280, 720);

            //convert & scale (.3 = means 1/3 of joint distance)
            Joint scaledJoint = joint.ScaleTo(1280, 720, .3f, .3f);

            Canvas.SetLeft(element, scaledJoint.Position.X);
            Canvas.SetTop(element, scaledJoint.Position.Y);

        }


        private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            closing = true;
            StopKinect(kinectSensorChooser1.Kinect);
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {

        }







    }
}
```