KAUNAS UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATICS

DEPARTMENT OF MULTIMEDIA ENGINEERING

Cenker Canbulut

A STUDY OF USER INTERFACES BASED ON HAND GESTURES

Master Thesis

Supervisor
Doc. Dr. T.Blažauskas

KAUNAS, 2017

KAUNAS UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATICS
DEPARTMENT OF MULTIMEDIA ENGINEERING

Cenker Canbulut

A STUDY OF USER INTERFACES BASED ON HAND GESTURES

Master Thesis

Supervisor
Tomas Blažauskas

_____
(date, signature)

Reviewer
Mikas Binkis

_____
(date, signature)

Master Student
Cenker Canbulut

_____
(date, signature)

KAUNAS, 2017

KAUNAS UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATICS

(Faculty)

CENKER CANBULUT

(Student's name, surname)

621I0003 INFORMATICS

(Title and code of study programme)

DECLARATION OF ACADEMIC HONESTY

2017 - 05 - 25d.

Kaunas

I confirm that a final project by me, **Cenker Canbulut**, on the subject "**A Study of User Interfaces Based on Hand Gestures**" is written completely by myself; all provided data and research results are correct and obtained honestly. None of the parts of this thesis have been plagiarized from any printed or Internet sources, all direct and indirect quotations from other resources are indicated in literature references. No monetary amounts not provided for by law have been paid to anyone for this thesis.

I understand that in case of a resurfaced fact of dishonesty penalties will be applied to me according to the procedure effective at Kaunas University of Technology.

(name and surname filled in by hand) *(signature)*

SUMMARY

In this project, implementation of a testing application which will measure the performance of *"Kinect"* and *"Leap Motion"* has developed. In the beginning, a study was conducted to see which gestures would fit as a comparison point between two devices as well as their technological capabilities. Also, study on User Interface designs provided us the idea to develop a User Interface which used for all gesture types to establish same conditions between devices. An evaluation process for each implemented gesture followed by comparing both devices in terms of execution time. The results provided us to see in which device and hand gesture users perform faster. Also, usability of both devices is measured to see which device users prefer to interact with User Interfaces. The results from the user test also have been quantified to see which reasons for each gesture effected the execution time on a specific device. Despite technological differences between both devices, it is observed that *"Kinect"* still need improvement in hardware part of the device for better results. This research might give insight to others who want to explore tracking devices to see capabilities and understand how interactions may be used when developing a User Interfaces for *"Kinect"* and *"Leap Motion"* devices.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY OF TERMS AND ABBREVIATIONS

GUI – Graphical User Interface

NUI – Natural User Interface.

Mockup - A model or replica of a machine or structure, used for instructional or experimental purposes.

"Kinect" - is an add-on device for the Microsoft Xbox 360 and Personal Computer gaming system that enables users to control games, movies and music with physical motion or voice commands and without the need for a separate input controller like a joystick or keyboard.

Manual Testing -Manual testing is the process of manually testing software for defects. It requires a tester to play the role of an end user and use most of all features of the application to ensure correct behavior.

IR Laser - Far-infrared laser or terahertz laser (FIR laser, THz laser) is a laser with output wavelength in between 30-1000 μm (frequency 0.3-10 THz), in the far infrared or terahertz frequency band of the electromagnetic spectrum.

User-Friendly - A machine or system that easy to use or understand.

Console – A small electronic device for playing computerized video games.

"Leap Motion" **-** Leap Motion, Inc. is an American company that manufactures hardware sensor device that detects hand and finger motions as input.

INTRODUCTION

Today, Kinect is used for different cases such as games, business applications, simulation platforms and more.

When developing a Kinect application, one of the key concept is to design and develop the Graphical User Interface. GUIs has two main factors that has effect on the usability measurements. One is the placement of graphical elements and the other is their gestures. There is different type of motion controls can be used when developing an application. Different known companies decide to use how to use gestures after many debates or meetings. Cisco, developed a sample application for cloth companies that lets users to change and try clothes. The company decided to use Swap action on the application to change the clothes on specific body part. And wait to select control to select the body part. These examples can be expanded by looking in games such as Dance 3, or Nike Sports Xbox games and many others.

Unproven part of these applications is to decide which gesture suits best to applications been developed. Users cannot always get use to the hand gestures that are provided by the developers. Problems after the development, such as, lack of controlling hand gesture on application menu can be seen in various applications.

These issues can depend on many reasons such information provided to user before controlling the GUI or colors that are picked for the GUI menu, or even the placement of the menu itself.

These reasons will be revealed to understand which factors do affect the usability of the hand gesture when controlling GUI of an application using "*Kinect*" and execution time testing procedure will be performed for 20 attendants to see which common gesture performs faster in terms of execution time. After analyzing and revealing all these factors, analysis for optimal solutions will be applied, and resulted respectively.

Purpose of the Document

This document provides *"Kinect"* and "*Leap Motion*" devices' analysis as well as the hand gestures in *"Kinect"* controlled applications that contains usability, understandability, and controllability of it. It also overviews the different solutions have been found for this manner. Document describes all possible methods been used nowadays in *"Kinect"* controlled applications with hand gesture.

Scientific Novelty

A study on usability of two related technologies *"Kinect"* and *"Leap Motion"*. Evaluation of common hand-gestures in terms of performance used to control GUI elements on *"Kinect"* and *"Leap Motion"* devices. Study of controlling hand gestures using both devices will help the implementation of hand gestures on created testing application.

Aim and Objectives

The aim of the project is to make Research on common hand gestures for *"Kinect"* and *"Leap Motion"* devices and evaluating their performance by creating a testing system.

To achieve this aim we should follow these objectives;

1. Analyze widespread natural user interfaces devices and the hand gestures dedicated for these devices.
2. Propose the system which will be used for gesture usability study.
3. Design and implement proposed system.
4. Evaluate the usability of the hand gestures dedicated for *"Kinect"* and *"Leap Motion"* devices.

1 ANALYSIS

Before 2000s, interacting with NUI elements were limited with mostly Keyboard and Mouse, even if there were devices developed for interaction with NUI elements, they were either costly or just about prototypes. After, 2000s remote controlled devices started to become the part of global market, and now in our days, there are many companies with improved technologies develop their own devices to make NUI interactions for health care, gaming and business industry. In our research, we analyze two devices used today for interacting with Graphical User Interfaces. The purpose we choose these two devices for analyzing is their similarity in terms of behavior when interacting with User Interfaces. One is *"Kinect"* device Microsoft's top innovation for remote interaction with content components of applications (GUI or Game object, etc.). The technology how *"Kinect"* device detects gestures can be found in section **1.1.** Another device is *"Leap Motion"* which is used as an attachable device on *"Oculus Rift"* or as a single device to interact with application content (GUI or Game Content, etc.). The technology how *"Leap Motion"* device detects gestures can be found in section **1.2**. These two devices interaction gestures differ in particular cases but also shows same characteristics as well. *"Kinect"* hand gestures are separated in two groups, they are called common gestures and custom hand gestures. The way they separate gestures in such comes from the SDK support of the Microsoft itself. The SDK supports 3 hand gestures as *"Push to Select"*, *"Wait to Select"* and *"Grab to Select"* in default. The ones aren't supported by the SDK are called custom gestures. Widespread custom hand gestures for *"Kinect"* device dedicated as *"Swipe to Select"* in section **1.3.4**, *"Hold and Swipe to Select"* in section **1.3.5** gesture. *"Leap Motion"* from SDK supports only 1 gesture type to interact with GUI elements which is *"Push to Select"* the device beside supports all other gesture types for Object interaction such as *"Grab"*, *"Swipe"*, *"Hold and Swipe", "Pinch to Grab"* gesture.

User Interface design for remotely controlled applications is another concept which determines usability factor of an application. The applications developed for *"Kinect"* device has both custom and common gesture controls over the interface designs of those applications. Popular games' user interface design and their gesture controls are going to guide us to find out unique solution to implement our own gesture testing application's user interface for all developed hand gestures.

1.1 Analysis of Microsoft *"Kinect"* Technology

"Kinect's" impact is not only capable for the gaming industry. It actually went beyond that and it as a new way to interact with machines and to perform other tasks. Its wide availability for many platforms with its low cost, took the attention of many researchers and practitioners in computer science, robotics and even in medicine where the device use is helping children with autism to assisting doctors in operating rooms [1].

The *"Kinect"* sensor incorporates several advanced sensing hardware. Most notably, it contains a depth sensor, a color camera, and a four-microphone that provides full 3D motion capture of human with facial recognition, and voice recognition capabilities.

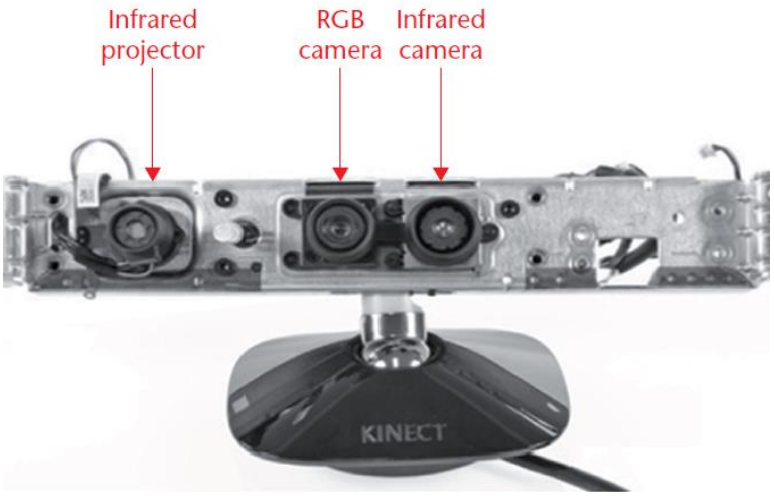

Figure 1- Microsoft "Kinect" Hardware Components **[2]**.

The depth sensor consists of the IR projector combined with the IR camera, which is a monochrome complementary metal oxide semiconductor (CMOS) sensor. The depth-sensing technology is licensed from the Israeli company Prime Sense. Although the exact technology is not disclosed, it is based on the structured light principle. The IR projector is an IR laser that passes through a diffraction rating and turns into a set of IR dots. The relative geometry between the IR projector and the IR camera as well as the projected IR dot pattern are known. If we can match a dot observed in an image with a dot in the projector pattern, we can reconstruct it in 3D using triangulation. Because the dot pattern is relatively random, the matching between the IR image and the projector pattern can be done in a straightforward way by comparing small neighborhoods using, for example, normalized cross correlation [1].

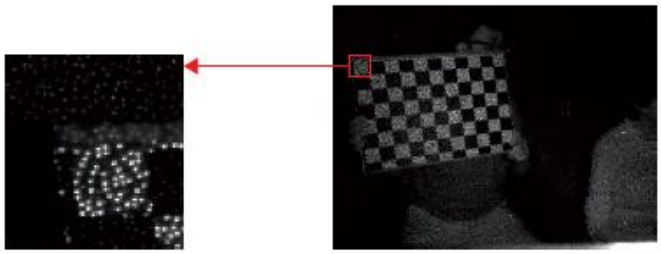

Figure 2 - IR dots seen by the IR camera **[3]**.

1.1.1 Skeletal Tracking in Microsoft *"Kinect"*

The innovation behind *"Kinect"* depends on advances in skeletal tracking. In Microsoft *"Kinect"*. In skeletal tracking, a human body is represented by a number of joints representing body parts such as head, neck, shoulders, and arms. Each joint is represented by its 3D coordinates. The goal is to determine all the 3D parameters of these joints in real time to allow fluent interactivity and with limited computation. We will be using *"Kinect II*" for best tracking and motion control in our experiment.

1.2 Analysis of Leap Motion Technology

In hardware part of the device, "Leap Motion" consists of two cameras and three infrared LEDs. Infrared provides a couple of advantages; by using bright LEDs along a narrow band of the spectrum, and looking just at that part of the spectrum, device can track through external light sources like the sun. Tracking becomes immensely more complicated (and slower) once you have a lot of background stuff to filter out. The two cameras are used for increased field of view to provide users more wide range of tracking space. This gives viewing range with 60 cm limitation. With latest updates this range increased to 80 cm. 20 cm increased field of view comes from the Orion Software that developed by the company.

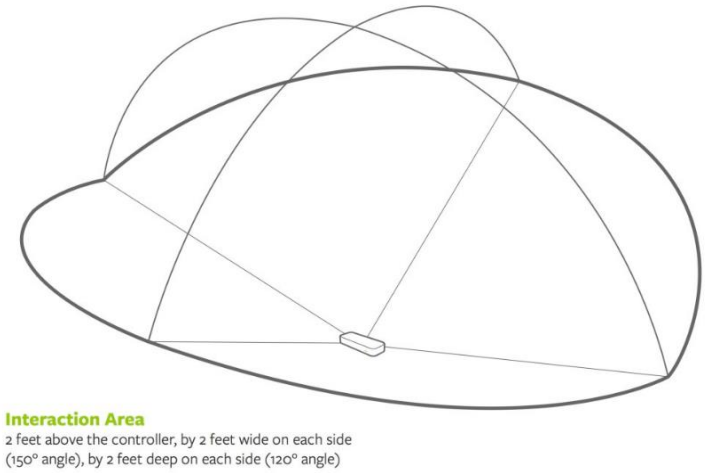

Figure 3 - Representing Interaction area of "Leap Motion" Device **[4]**.

The device's USB controller reads the sensor data into its own local memory and performs any necessary resolution adjustments. This data then transferred to device's tracking software [4].

In the software part of the device, despite of popular misconceptions the device doesn't generate a depth map instead it applies advanced algorithms to the transferred sensor data [4].

The Leap Motion Services software processes images. After processing and comparing the background by distinguishing objects such as head in human body, ambient lighting, the images are analyzed to reconstruct a 3D representation of what original data will look like [4].

Next, the tracking layer matches the data to extract tracking information such as fingers. Our tracking algorithms interpret the 3D data and infer the positions of occluded objects. Filtering techniques are applied to ensure smooth temporal coherence of the data. The Leap Motion Service then feeds the results – expressed as a series of frames, or snapshots, containing all of the tracking data – into a transport protocol [5].

Through this protocol, the service communicates with the Leap Motion Control Panel, as well as native and web client libraries, through a local socket connection (TCP for native, WebSocket for web). The client library organizes the data into an object-oriented API structure, manages frame history, and provides helper functions and classes. From there, the application logic ties into the Leap Motion input, allowing a motion-controlled interactive experience [5].

1.3 Hand Gestures for Microsoft *"Kinect"* device

Studies aiming at tracking information of hands, including shape, position, and motion, can be traced back to the 1970's. Optical, magnetic, or acoustic sensing devices were attached to hands to report their positions. Later on, a glove-based system was described and implemented, which became a common and matured approach in the field of hand tracking. Throughout the past 20 years, a large variety of glove devices as input media for HCI have been built, of which some have remained in research labs and others have reached the marketplace. [3]

Hand Gesture in *"Kinect"* used commonly to control GUI elements on an application. *"Kinect"* sensor can track 20 places of joint. It also can carry out easily hand detected and gesture recognition from this data and detected gestures [6]. *"Kinect"* provides you with the position (X, Y and Z) of the users' joints 30 times (or frames) per second. If some specific points move to specific relative positions for a given amount of time, then you have a gesture. So, in terms of *"Kinect"*, a gesture is the relative position of some joints for a given number of frames. [7]. As gesture can be anywhere in human body also the hand gesture can be recognized by the special algorithm of the Microsoft *"Kinect"*. Using this hand gesture recognition, we will try to create possible moves.

Usability of *"Kinect"* not only depends on hand gesture by itself but it is also connected with understandability and usability of Graphical User Interface (GUI) of the application. To develop a usable and understandable application there are also things to consider such as;

1.3.1 Push to Select Hand Gesture

Push to Select hand gesture is one of the method being used in *"Kinect"* to enter to specific GUI element of application. Most known issues with this movement are lack of precision when pushing a correct GUI element. Common mistake occurs when users do this specific gesture is that they might lose control over their hand movement and move out of the button desired to select. This might lead unclicked button and result in time spent to trigger button enter event.

Figure 4 - Reaction of user when making Push to Select Gesture **[8]**.

1.3.2 Wait to Select Hand Gesture

This move is maybe the most common move between all moves mentioned before. This move is used highly and often to select specific GUI item on the menu. Simply user stands on a specific item during counter time duration and when the duration is finished it enters the section of that GUI element. This move is the move that has lowest issues during the execution of the event. Therefore, this move is highly used by the developers as solution to enter section of specific GUI element. Today, games such as Dance 3, Fitness Evolved uses this hand gesture to make menu interactions.

1.3.3 Hold to Select Hand Gesture

Hold to Select gesture is mostly used to grab a GUI element to interact with it. This hand gesture is mostly used instead wait to select hand gesture and can be seen in various applications today. A different version of this hand gesture is used to zoom in and out by adding swipe functionality for both hands. Also, one hand hold to swipe is a custom Gesture that can be used for menu interactions.

Figure 5 - Hold to Select Gesture example **[8]**

1.3.4 Swipe to Select Hand Gesture

This gesture type is a custom gesture type which can be seen in game applications to User interface controls. Concept of this gesture is, a user interacts with GUI component with open hand. The user swipes the hand at certain velocity treshold in order to trigger button event over the UI element. The example can be seen in game called "Nike+ Kinect Training". It is fitness video game that developed by Microsoft with cooperation with athletics company Nike [9].

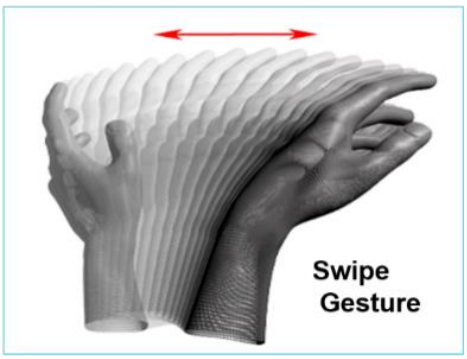

Figure 6 – Swipe Hand Gesture **[10]**

1.3.5 Hold and Swipe to Select Hand Gesture

Different than Swipe hand gesture implementation, the gesture confirms user's current UI element selection to eliminate miss clicks by holding hand motion. This kind of gesture implementation mostly can be seen in TV applications used by *"Kinect"* device. It is because smart TV's transition between menus can be slower than the PC or Xbox Consoles and each miss click might be too much time consuming for the user.

1.3.6 Hand Gestures for *"Leap Motion"* device

We are using *"Leap Motion"* as comparison point beside *"Kinect"*. When we compare these two devices' performances in terms of execution time, both devices must fulfill same functionality under same conditions. It means if hand gesture developed for *"Kinect"* has *"Grab to Select"* gesture. The same hand gesture type should be developed for *"Leap Motion"* device. *"Leap Motion"* supports interaction with UI and objects with given hand gesture types shown above for *"Kinect"* in sections **1.3.1, 1.3.2, 1.3.3, 1.3.4, 1.3.5**. The ones which are for object interactions will be developed for UI interactions to create comparison point between those two devices.

1.4 GUI Design Principles

Informative design is the information given when users attempt to control a GUI. Many applications do find different solutions depending on their purpose. Informative messages should better not give distractive use when you play a game or control an application [11].

Figure 7 - Giving information tip so user can control specific event. **[12]**

In **Figure 7** it is clear to see that the application gives a tip to make user understand which move to do to control specific event in the application menu. Example such and more so can be seen in Xbox console games and applications where *"Kinect"* is needed.

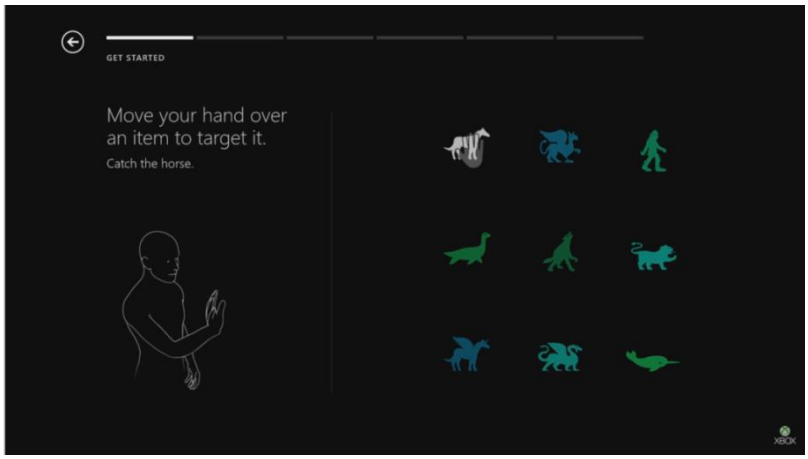

Figure 8 - Giving information tip so user can hand over and item to target it **[12]**

In **Figure 8**, it is also clear to see that the information is given to user what to do to control the event of the game.

These solutions are endless in *"Kinect"* controlled applications and games. Different companies and creators do find different solutions to be more informative for user and to make things more clearly served for them to understand.

GUI Design of application plays key role on to simplify the control of the menu items by the users. A well-designed GUI of an application helps to create more basic and understandable structure of the GUI so it becomes more understandable by the users. Because users might not always be experienced about *"Kinect"* control of an application, to consider all kind of users, GUI design must reflect the purpose of the application clearly. There is wide range of different GUI designs on market. Every creator does find different solution onto this. When talking about a whole design of a GUI the colors and the structure of GUI play important role on that. A successful design should never be tiring the eyes of users so they will not be confused and lost in the GUI. It must be clear and user-friendly.

That there are no universal standards for gestural interactions yet is a problem because the UI cannot rely on learned behavior. The "Kinect" has a few system-wide standards, however, which do help users [11].

Figure 9 - Activated by holding hand still over button **[11]**

For example, there is a standard way to pause a game to bring up the menu: stand with your right arm straight down and your left arm held at a 45-degree angle. Definitely not a natural stance — indeed, it was probably chosen because it will never occur in normal gameplay [11].

The pause gesture becomes the user's lifeline, but there are no standards for other, commonly desired generic commands, such as "back." This makes it harder for users to

navigate the UI, because they need to figure out these basic operations every time. Less learning takes place when the same thing is done differently in different games [11].

As two screenshots show **Figure 7** and **Figure 8** there's not even consistency as to which hand is used for "back," let alone which gesture to use or how the feature is presented visually [11].

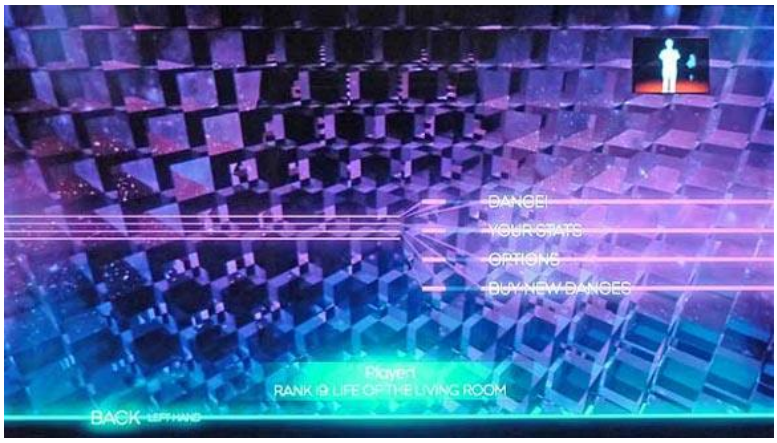

Figure 10- Activated by swiping left hand rightwards. (Dance Central) **[11]**

Designer uses green background with big iconic buttons represents each activity user can select. At the far background of the screen it silhouettes the motion of the user. Approach on GUI in this has been thought this way and provided user as optimal way to present. GUI design differs according to concept of the game and theme.

As it is seen, it is also depending on NUI Natural User Interface Design of and application beside the GUI element design. [13] NUI is a common term for interfaces that are more or less invisible to the user. The user can interact with a machine without the need to learn and adapt an artificial set of skills.

For our testing system, we will be taking this analysis in consideration and be developing it in accordance that the users will have correct understanding of how intended gestures do work.

1.5 Functionality of Hand Gestures

The *"Kinect"* system tracks user movements. When a predefined movement (or gesture) is completed an action occurs on the interface "*"Kinect"*" device supports 4 motion types such as forward and backward move of hand, Swap of hand, waiting, hold and swipe [14].

Those 4 types can be a mixture of hand gestures. User can hold and push or hold and pull even user can hold and move down or upwards to make specific move. It really depends on the developer's decision to make the mixture between the moves but this document takes most popular ones that being commonly and widely used nowadays.

Also, the development of all gesture functionalities will be implemented for the Leap Motion device to have a comparison point for our research.

1.6 System volume and software resources needed to develop system

Visual needs to create graphical elements for scenarios:

- Adobe Photoshop to create UI design.

Needed tools to create Microsoft *"Kinect"* and *"Leap Motion"* hand gestures:

- Microsoft *"Kinect"* SDK for Windows 10 and Unity Engine.
- *"Leap Motion"* SDK for Windows 10 and Unity Engine.
- Unity Engine for development area to polish every aspect of the work and make system ready for testing.

To design all necessary files and develop the system these additional works are required:

- 1 Software developer.
- 1 Graphic Designer.

2 PROJECT DESIGN

Design of the system will give us representation of how system will work when it is developed. It also will be our guide to stick on implementing gestures for both devices in same condition.

2.1 System Functionality

To make the system flow more understandable we will have representation with Use Case and Activity diagrams. These diagrams will give us more clear understanding of how system will work for testing purpose.

In testing part, more restrictions will be added to control GUI elements on the system to test the usability of hand gestures.

To analyze testing data for execution time, we will create a user interface with timer. The timer will start as soon as the user is ready to test the system. Buttons will be highlighted randomly and each click will have timeout for 5 seconds to let user prepare for next randomly highlighted button. After each button click the system will write down time to a txt file. The system will work in same way for each gesture. And user will select which hand gesture he wants to test before starting testing session.

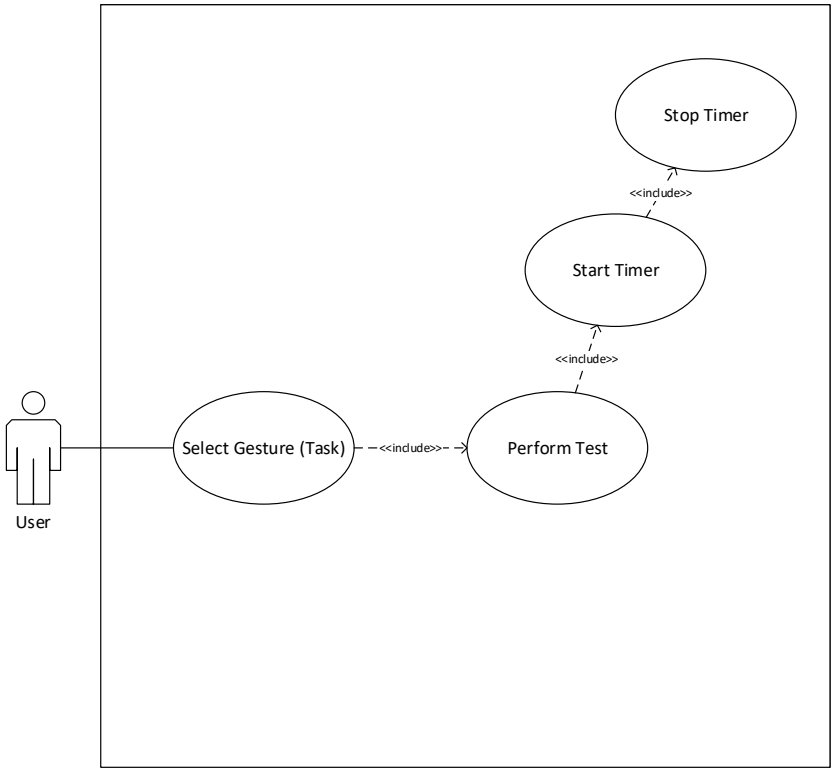

Figure 11 – Usability testing system usage scenario

User action functionality:

- Select Gesture – User selects gesture to proceed testing session.
    - Perform Test – With selected gesture user start testing with randomly highlighted buttons by the system.
    - Start Timer – System starts the timer as soon as button click made by user and also if the 5 second preparation phase finishes
    - Stop Timer – Stops the timer if user does button click for a highlighted button and gives 5 second preparation phase for next generated button.

2.2 User Interface specification

The user interface specification is presented using the so-called wireframes. Intermediate user interfaces are made using Photoshop and validated by the client.

Since the system will only have one particular GUI and one additional pop up menu. The GUI will be presented in 4 different forms in wireframe.

2.2.1 Mockup Design of System

The first GUI that the user will interact by using mouse is the Gesture selection for any device that the user desires to test (**Figure 12**). When the user selects a hand gesture, system automatically generates randomly clickable buttons ready for the user.

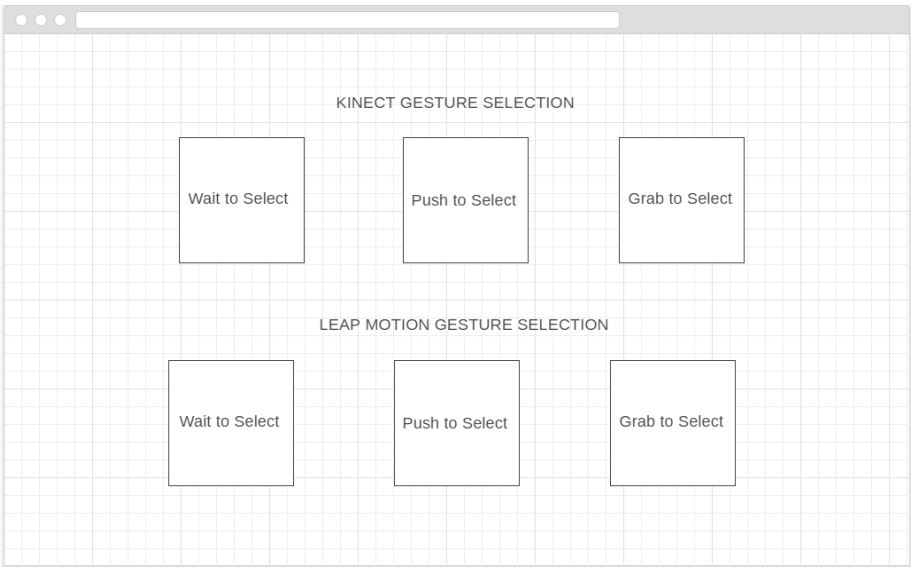

Figure 12 - Selection of Hand Gesture for Desired Device

After user selection for specific hand gesture on *"Kinect"* or *"Leap Motion"*, user will be directed to another interface where selected hand gesture for selected device will be ready to make interaction. User now can interact with one of the tracking device and click Start Test button when he or she feels ready to proceed for testing (**Figure 13**).

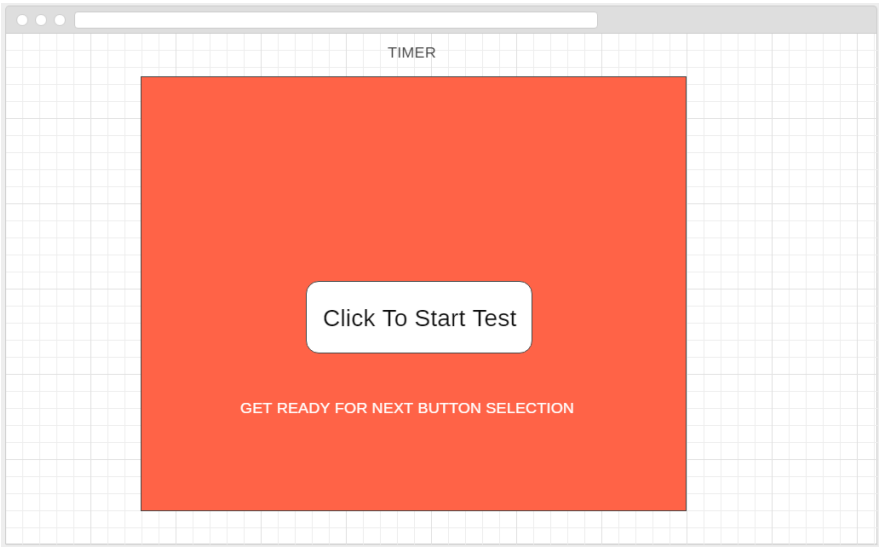

Figure 13 - Interface that ready to start for testing process

When user feels ready to follow testing process, he or she clicks "Click to Start Test" button. After the button selection is done, user gets timer running with highlighted button. In this case, the highlighted button can be any button on the screen chosen by system randomly.

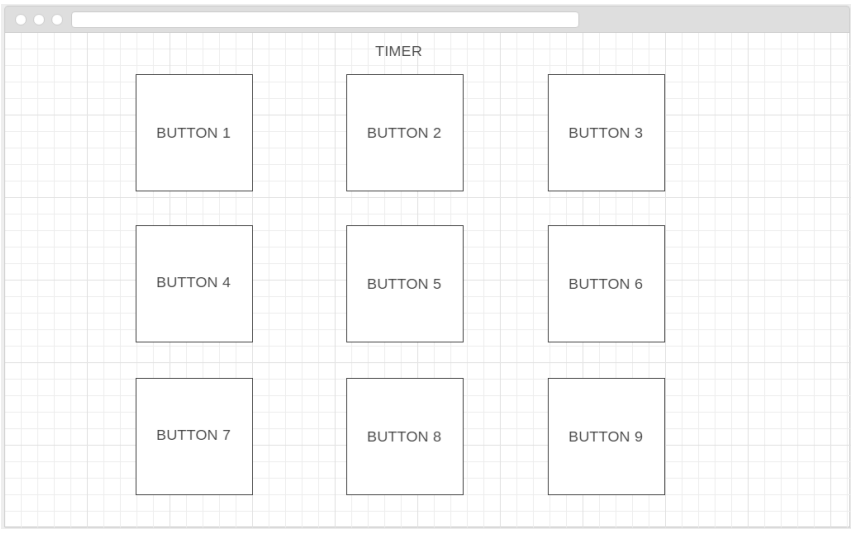

Figure 14 - Button formation for randomly highlighted buttons

When highlighted button click is successfully triggered, system writes down the button clicked by the user with its time and shows 5 second time-out for next button click as shown in **Figure 15**. This lets user to get prepared for next randomly highlighted button. It

continues until all buttons are clicked. After all buttons are clicked system turns back to gesture selection menu as shown in **Figure 12**.

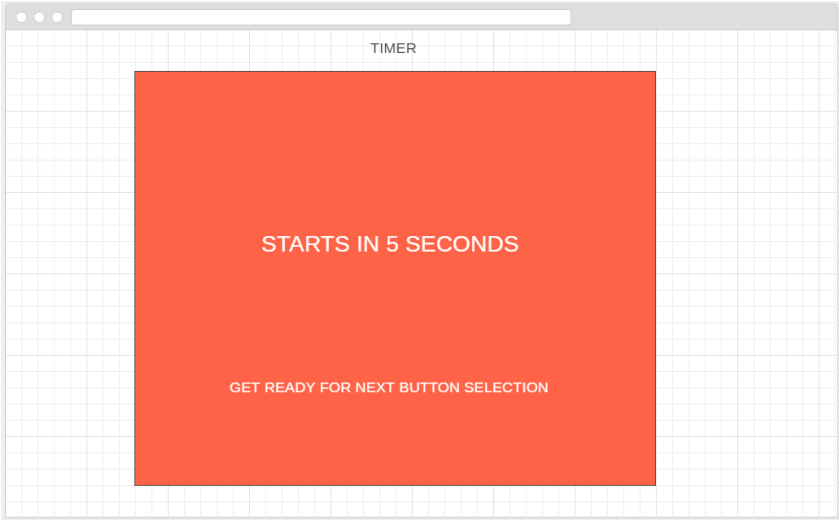

Figure 15 - After Button Click 5 second time-out.

2.3 System Architecture

Before the development stage, we should have overlook upon our development plan to be able construct a working system. For this reason, we will be looking what devices and software components the system demand. The representation of whole system will be shown in deployment model, class diagram, and activity diagram sequence.

2.3.1 System Deployment

System will be running on a desktop or a laptop computer that has installed Microsoft *"Kinect"* SDK and Orion Software to work and communicate with *"Kinect"* and *"Leap Motion"* within the Unity 3D engine. The unity 3D engine is compiling all the system components and giving an executable (.exe) file for launching the system. All GUI elements will be created and programmed inside the unity 3D Engine. User interface controls, system automation, hand gestures for *Kinect"* and *"Leap Motion"* will also be programmed inside Unity 3D Engine. To make interaction inside Unity for both devices and to have a development environment for them we will be running Gesture Recognition Library plugin that is supported by Unity for both devices. In total, system will have 3 physical devices and 3 software clients working.

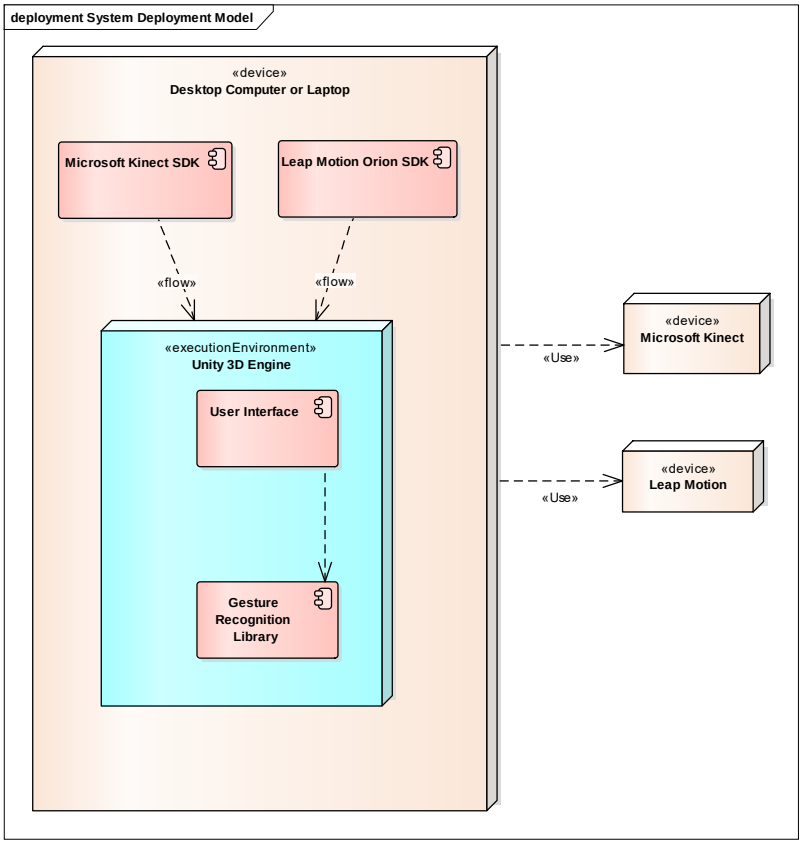

Figure 16 - Deployment Model of the system

2.3.2 Kinect Control Components

To make our system understandable, we split our class diagram into 2 seperate parts. One for *"Kinect"* device and another for *"Leap Motion"* device. These two devices use different SDKs with different instances in the system. The rest work principle will have common work flow. Below **Figure 17**, we see how "*Kinect"* device's work flow in the system will be established be shown.

Unity uses "*MonoBehaviour"* type as many 3D engines use. The reason Unity uses this type of approach is logically, it enables *Start()*, *Awake()*, *Update()*, *FixedUpdate()*, and *OnGUI()* for execution order. This, makes development more flexible for different control points of the software.

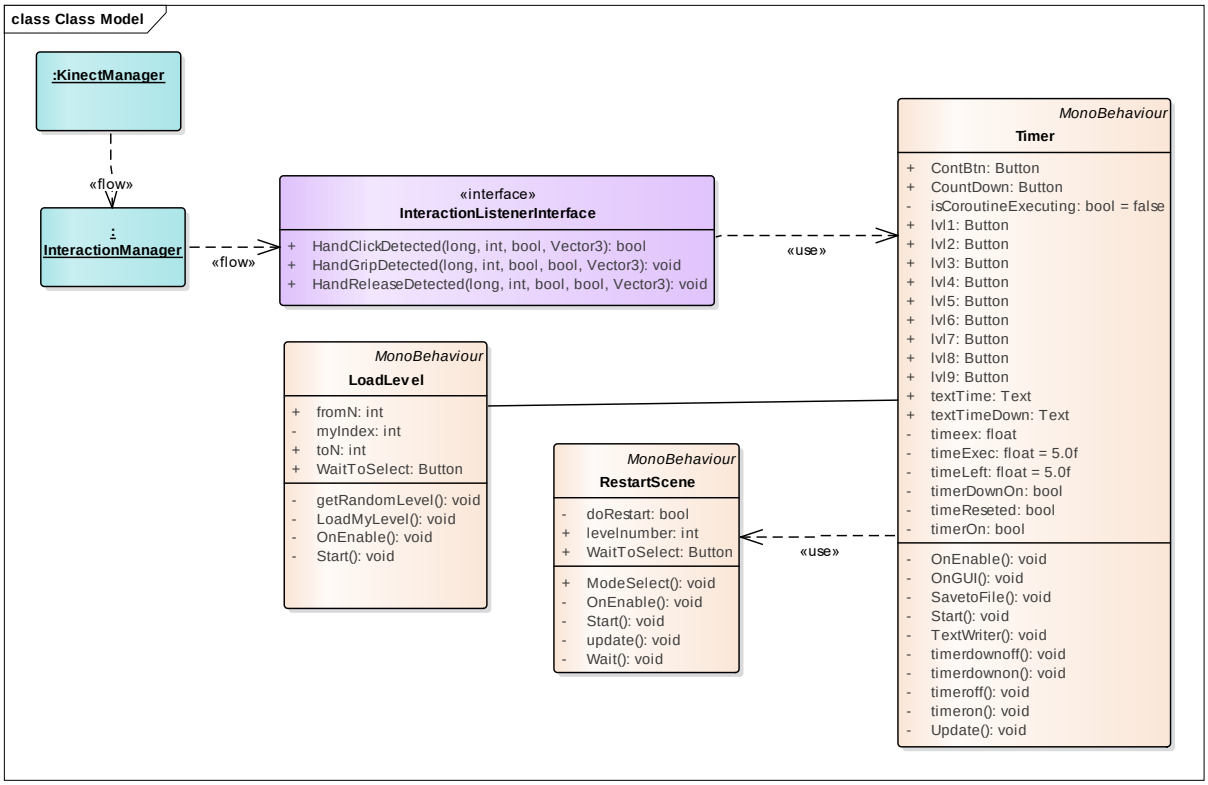

Figure 17 - Kinect Class Diagram Work Flow in the System

- Kinect Manager(Instance) – This whole big C# script is responsible of transferring SDK data to Unity ‘s framework. This lets us to have and use all *“Kinect”* data communications inside the Unity framework.

- Interaction Manager(Instance) – One another C# script provided by the SDK for Unity, ables *“Kinect”* device to interact with Unity ‘s Graphical User Interface property.

- InteractionListenerInterface(Interface) – This separate script we created uses Interaction Manager(Instance) provided by SDK. We take necessary X, Y, Z coordinates to be able to create hand gestures that we need to include in the system.

- LoadLevel(Script) – Script, lets us to have random highlighted buttons each time the testing application run. We use this approach in purpose that the user will not be aware of upcoming button highlight.

- RestartScene(Script) – This script detects last clicked button by the user, finishes the testing procedure and lets the system get back to its idle state which is selection of hand gesture.

- Timer(Script) – Timer script contains all 9 clickable buttons and a timer with preparetion phase. Those buttons get highlighted randomly, once a highlighted button is clicked, it changes the clicked button to uninterruptable state. It also controls time-out (preparation phase) of 5 second.

2.3.3 Leap Motion Control Components

Different than the *"Kinect's"* SDK, Leap Motion doesn't contain InteractionManager(Instance) to be able to make interactions possible with Unity's Graphical User Interface component and its property. The SDK only comes with Object component interactions and lets us to use only this component's property. Because of this reason, we develop our own Interaction manager for *"Leap Motion"* device to have same conditions when we compare two devices.

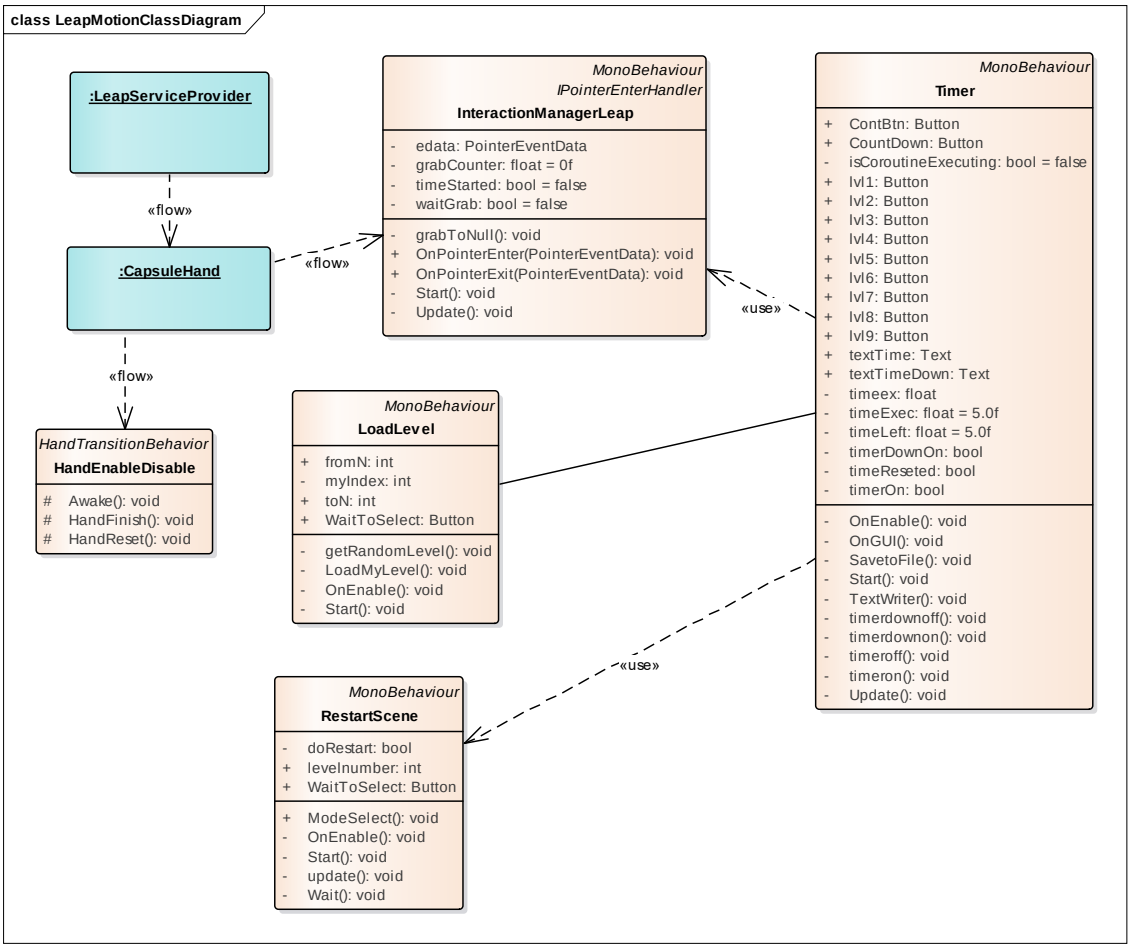

Figure 18 – Leap Motion Class Diagram Work Flow in the System

- LeapServiceProvider(Instance) - This C# script is responsible of transferring SDK data of *"Leap Motion"* to Unity 's framework. This lets us to have and use all *"Leap Motion"* data communications inside the Unity framework.

- CapsuleHand(Instance) – CapsuleHand is another script keeps hand data of 3D hand model. Custom 3D model of any hand can be integrated in the *"Leap Motion"* system. This scrip also used to specify which hand the user is using, so user can match a 3D hand model to a specific hand side.

- InteractionManagerLeap – The script helps us to take hand position data from CapsuleHand and make it intractable with the Unity's Graphical User Interface component with its properties. Depending on the position we can create the gestures we need for testing in our system.

- HandEnableDisable – Script provides users not to get confuse when their hand is not detected. If hand is out of Leap Motion's detectable range, the hand gesture becomes invisible. So, users attempt their physical hand towards the device to let it become visible again.

2.3.4 Action or Flow of System

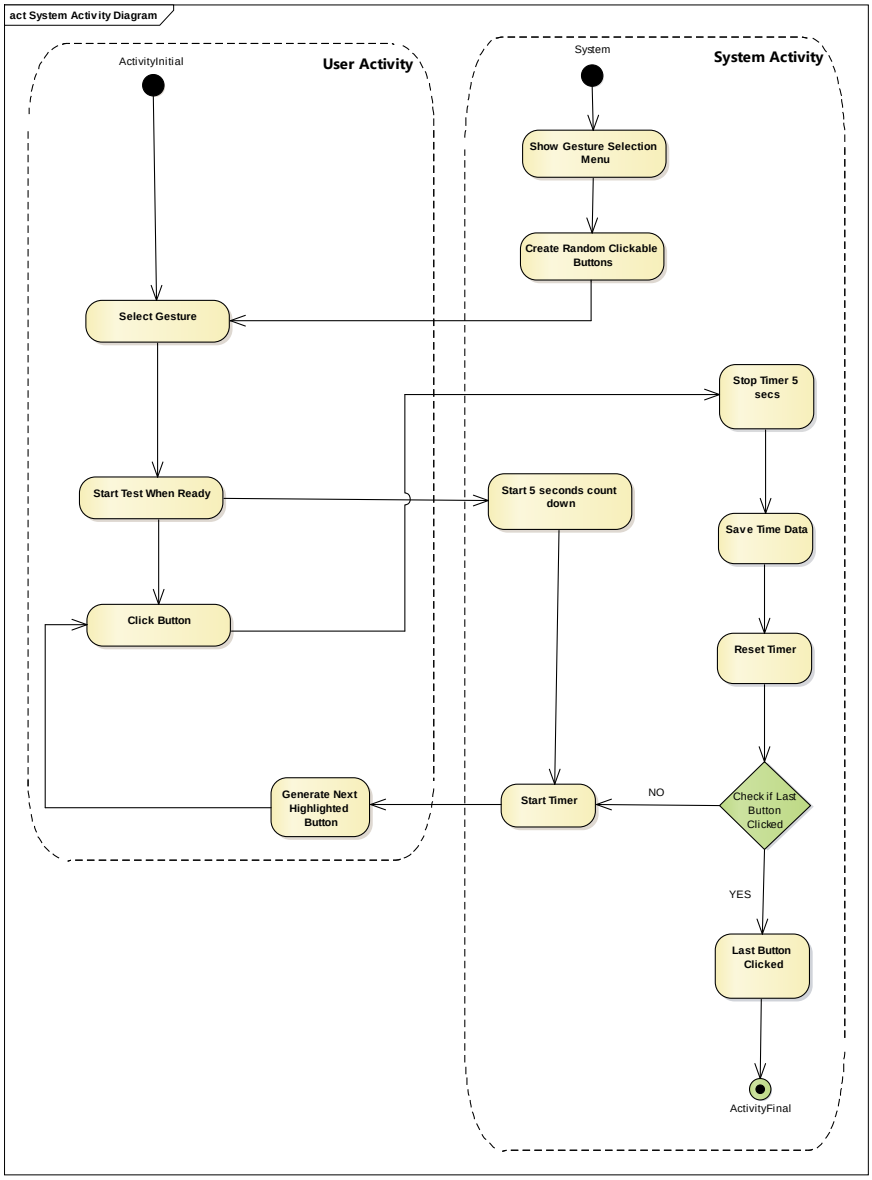

Figure 19 - Hand Gesture Usability Testing System Activity Diagram

- Show Gesture Selection Menu – The first menu the user sees before the start of testing session. Here, user does which gesture type he or she wants to perform.

- Select Gesture – User already selected his or her hand gesture type for testing. After the selection, system creates randomly clickable buttons in sequence for the user.

- Start Test When Ready – User performs clicking on start button, System warns user for upcoming button click after 5 seconds.

- Click Button – User has 9 clickable buttons where only one is highlighted and indicating which button the user should click. After the click performed;

- Stop Timer, Highlighted button click was performed for testing purpose, system stops timer and gives 5 second timeout to let user prepare for next upcoming highlighted button. and resets the timer.

- Save Time Data - During the 5 second preparation phase, system saves the time data of the button that was clicked previously to created txt file

- Reset Timer – After Saving previously clicked button's time data, system resets the timer.

- Check Last Button Clicked – System Checks the previously clicked button's state;

    - If No – If the button previously clicked was not the last button, system generates new highlighted clickable button and starts the timer.

    - If Yes – If the button previously clicked was last button, system finishes the activity and gets back to initial state.

3 IMPLEMENTATION OF THE SYSTEM

Implementation of the system covers the techniques and solutions we found during the development stage of the testing software as well as the descriptions about creativity and approaches to create such a system that will provide us the testing and evaluating measurements. We will only uncover important parts of implementation of the system and explain them with pseudo codes.

3.1 Implementation of GUI Order

Menu creation of the game made using powerful Unity 3D engine components and scripts to get the menu work in order. In this section, we will see what we got step by step to create those menu components. Below we can see the User Interface Hierarchy of system.

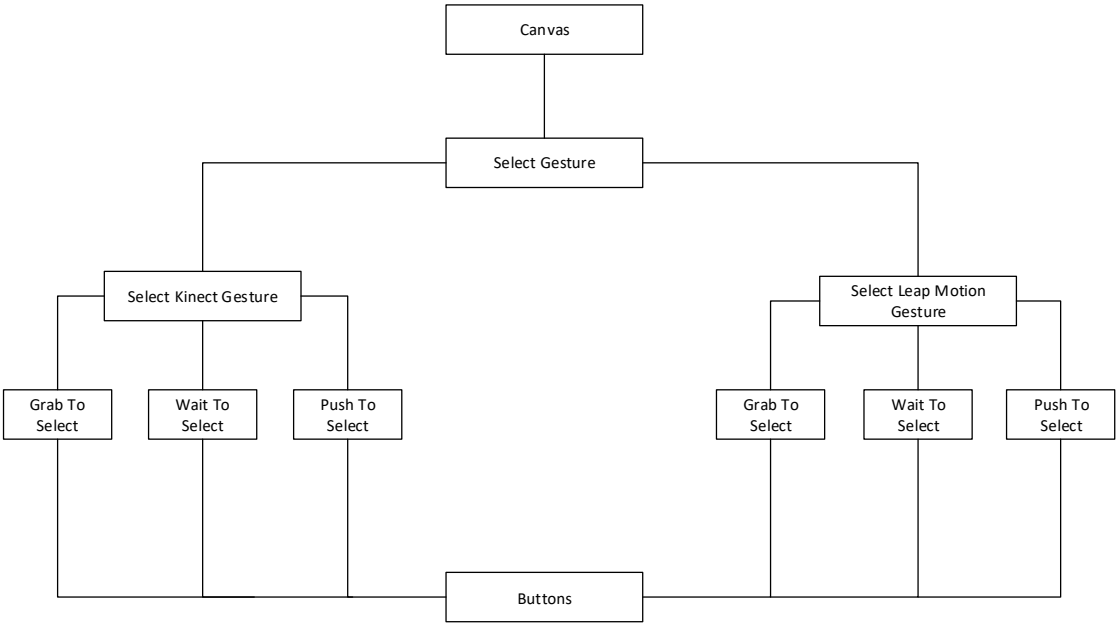

Figure 20 - Menu GUI Hierarchy Diagram

- Canvas – Canvas is GUI container of Unity Engine. It splits GUI elements from other elements of the engine so the developer can find and work on them more flexible.

- Select Gesture – Select Gesture is the Main menu of the GUI. User sees this menu once the tester is launched.

- Select Kinect Gesture and Select Leap Motion Gesture – Menu splits into two parts for both devices and their gesture types.

- Gesture Selections (Grab to Select, Wait to Select, Push to Select) – User decides which of the device's gesture type he or she is going to test and clicks that button.

- Buttons – Every gesture selection gets to this menu after selection is completed. Randomly Clickable buttons generated and presented for the user.

3.2 Implementation of Graphic Design for Graphical User Interface

For the implementation of GUI elements, we used 1 Background sprite and 2 button state changing backgrounds. Since, we only will have software for testing purpose and not for public, this simplistic approach was giving us less load on Graphic Card Memory and was easy to modify for gesture types. We also use, 3 hand gesture graphics for *"Kinect"* device. One prefab for unity used to present "Leap Motion's" 3D hand modelling. Other info giving elements were taken from the unity's text component.

3.2.1 Button Designs

Simple button design principles are used for testing purposes of our research work. To access this simplicity, two major button types are designed. One is for default state of button and another is for hovering of the button. They are square shaped, and cannot be deformed when resized. This gives us flexibility to find a nice fitting formation for the buttons when locating on GUI of the system.

Figure 21 - Normal and Hover Background Design for Button Dynamics

If the button has highlight effect, we change its green outer line's width with animation component of Unity, giving it more visible green color with transition between wider and thinner border of the button. It loops between wider and thinner border until the button click is triggered by the user.

3.2.2 Text inputs

Text inputs are done using Unity's text UI component. So, it is not designed in another application to fit inside the button background but instead, for each button element, text object created. Also, Timer uses same text component. This approach is flexible and

easier to use when we aim to develop a testing software. If, we want in our application a different looking text later. We can do it by uploading any text font to Unity project and change the text component's font property.

3.3 Implementation of UI Elements

Research project contains scripts that let the Menu transitions for both devices within unity. In this section, we are going to describe how our GUI elements behaves within the system.

3.3.1 Main Menu Manager

The menu manager is created to know which Menu container is in current active state. This will allow the transition to be triggered properly. Because, if we know which menu has the current state we can easily set not active stance for unused menu containers. This component is developed and assigned to Canvas of the Menu, where the canvas has the highest hierarchic order in Unity where the script can be accessed by another menu elements.

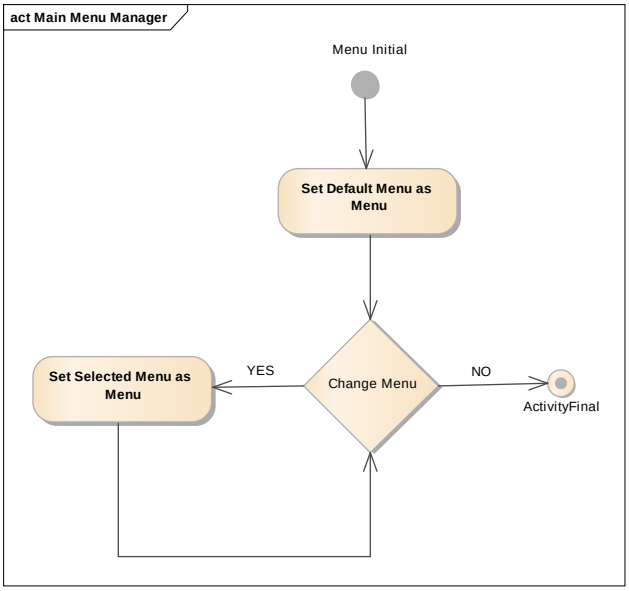

Figure 22 – Main Menu Manager Activity.

- Set Default Menu as Menu – The menu assigned as default menu element at the beginning of the run-time of the system.

- Set Selected Menu as Menu – If user clicks a button assigned to another menu item, system sets new menu element as menu component.

- Change Menu – If user changes menu item again the menu becomes as Selected menu if not System activity finishes.

3.3.2 Timer

Timer is created to track the speed factor of the user while controlling UI. The script has two holder buttons. Those two assigned buttons used one for starting the timer and the other stopping the timer. The script also contains 9 button objects which are randomly get the state of clickable as reference from *"loadLevel"* script. The script contains another GUI object which is task manager. Task manager, states time-out for next upcoming button click. *"Lastactive"* script does look if the last active button was clicked and if it is clicked, it brings application to its idle state.

The brain of the code is the controlling of the button states when time-out needs to be given for the user. When the system in timeout state the system shouldn't lose the control of randomly highlighted buttons and also buttons shouldn't be clicked mistakenly after the timeout.

After each button click, system should write the time in a txt file this takes button name of the button and time of the button when clicked to approach the click and write down time we created this line of code. We add listener to *OnEnable()* monobehavior of unity. And when we see a button click we start *TextWriter()* class.

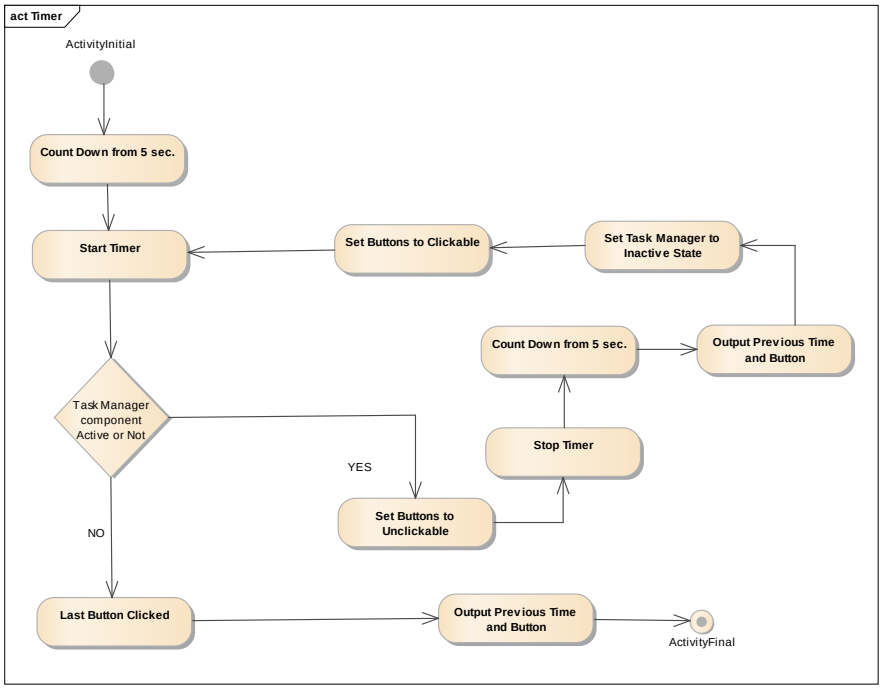

Figure 23 Timer Script Activity Diagram

- Count Down from 5 Secs – System Starts counting from 5 secs down when the begin button clicked.

- Start Timer – System starts timer from 0 to increasing initial value as seconds:miliseconds.

- Task Manger Active or Inactive – Timer checks if the task manager component is active, if yes it sets buttons to unclickable state to prevent misclicks by the user.

- Stop Timer – Timer stops timer seconds:miliseconds.

- Count down from 5 secs – After the timer stops system counts down for next task from 5 seconds.

- Output previous time and button – Timer outputs the stopped time and specific button clicked.

- Set Task Manager to Inactive state – Timer sets Task Manager component to Inactive state to let the users see button that is highlighted.

- Set Buttons to Clickable – Timer sets highlighted button to clickable state.

- Last Button Clicked – Timer determines that the last button is clicked and outputs its time and button.

3.4 Implementation of Hand Gesture Behaviors in the System

As I have mentioned previously, SDKs from the device developers are one of the reason why we choose to develop the testing system in unity. The library they provide for developers reduces our work load to create complex physical algorithms to find hand behaviors in the system. We call necessary parts we want to develop from SDK and modify to create our own custom behaviors in the system. We will see how we implemented all hand gestures in this section for both devices.

3.4.1 Implementation of Hand Gestures for Leap Motion device

To create three custom hand gestures for "*Leap Motion "*. First we had to look the "*Leap Motion's"* library where we could modify the referenced part of the script for our needs. We get this part of code for our reference point and create our own to give best result for push to select.

From the SDK, we get hand velocity of hand, and give it z axis which means towards the front of the user. Then, the speed of z motion for hand object should be faster than given value and *isStationary()* class which checks if the user's hand is stationary. If not and if hand pushes forward in optimal speed. We apply Push to Select gesture trigger on a button.

3.4.2 Implementation of Hand Gestures for Kinect device

Same as the *"Leap Motion"*, we also followed similar methods for implementation of *"Kinect"* hand gestures. We got reference data from the SDK and modified it for ours needs.

Briefly we make restrictions to let *"Kinect"* device to listen clicks for only UI component of unity. From the SDK, we take the distance between our current position and z position and check if it is higher. *"Kinect"* hand graphic in Unity doesn't trigger button clicks on buttons for SDK reasons. So, we set mouse cursor to the *"Kinect"* hand graphic and let the system trigger mouse clicks instead of button clicks on buttons. To make it possible we use the mouse controller asset [15], This asset manipulates Microsoft's original Mouse cursor not only for read but also makes it writable. we set the cursor position of the mouse to the hand gesture position to make possible clicks. So, if the hand gesture does correct movement we apply click to the mouse cursor and it triggers click for the specific UI element.

4 EXPERIMENTATION

The experiment is aiming to evaluate gesture's performance comparing both devices by their gesture types as *"Push to Select"*, *"Grab to Select"* and *"Wait to Select"* in terms of execution time. The experiment also performed on 10 attendants for both devices from the user perspective to evaluate their usability measurements. To get both measurements we will perform 2 experiments:

- Performance – This experiment measures the time difference between both devices with 3 gestures mentioned above. The results for this experiment is in section **4.2**.
- System Usability Scale (SUS) – The experiment measures the usability factor of both devices from the user perspective. The results for this experiment is in section **4.3**.

4.1 Experiment Setup

To be able to setup this experiment hardware and software needs shown as follows;

Hardware setup:

- Computer with Windows 10 OS.
- Kinect-II device.
- Leap Motion device.
- Monitor or TV to see necessary information inside the system.
- Kinect-II Computer connector.

Software Setup:

- Unity version 5.5.0f3 (32 or 64-bit)
- Unity Core Assets version 4.1.6 for Leap Motion
- Leap Motion Interaction Engine for Unity
- Leap Motion UI Input Module for Unity
- Kinect SDK for Unity

4.2 Experiment procedure for Performance measures

The experiment is performed using 3 gesture types as mentioned before, *"Wait to select"*, "*Grab to select*" and *"Push to select"* for *"Kinect-II"* and *"Leap Motion"*. Buttons are in a square formation 3 at top, 3 at middle and 3 at the bottom placed side by side. The purpose of this formation is to find out the effectiveness of each hand gesture in differently located buttons on user interfaces.

The system indicates user with begin button to start testing procedure. After the button is clicked, user makes necessary selections on supposedly highlighted button. Each time a

highlighted button triggered, system outputs clicked button with its time. Each button click again indicates user with count down timer starts from 5 seconds, this lets users to get prepared for next randomly highlighted button and when the time reaches 0, user attempts to click another highlighted button. The procedure loops until the last active button is executed.

4.2.1 Gesture Testing Framework

To measure the effectiveness of motion types using *"Kinect-II"* and *"Leap Motion"*, we focused on three main gesture types that are used commonly today. These are stated as, "*Grab to select*" where user makes fist motion of hand to select specific menu button on interface. *"Wait to select"*, the gesture type that provides users to select when the hand gesture stands on specific menu item after a time lapse and *"Push to select"* that user does forward and backward hand motion while on a button to trigger click.

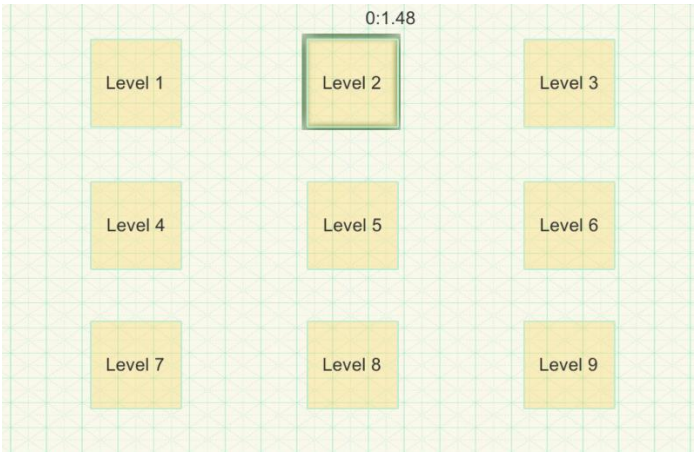

Figure 24 - Formation of the Buttons

To determine usability of *"Kinect-II"* and *"Leap Motion"* gestures a testing procedure will be followed. For this purpose, an application is developed to find execution time of hand gestures with commonly used UI design. This will allow us to get precise data from user interactions. Later, the data is printed to Txt file to be able to note it down for our research goal.

The system is developed using Unity Engine, it has 3 scenarios with 9 randomly highlighted clickable buttons which are placed on the canvas in such a manner that they cover uniformly all the window.

4.2.2 Gesture Execution Results of Kinect Device

Table 1 – "Wait to select" Gesture – Kinect II device

BUTTONS	USERS										Avg
	1	2	3	4	5	6	7	8	9	10	
1	3.98	3.38	3.89	3.83	4.17	4.33	4.50	3.24	3.74	3.65	3.87
2	3.96	3.78	4.54	4.58	4.85	4.01	4.71	3.87	4.01	3.81	4.21
3	4.36	3.62	4.42	5.70	3.58	4.46	5.91	3.67	3.64	4.21	4.35
4	4.22	4.75	4.21	4.76	3.43	4.21	4.45	4.42	3.97	4.14	4.25
5	3.81	4.37	3.73	4.25	3.13	2.33	2.78	2.45	2.83	4.14	3.38
6	3.61	4.11	2.90	3.26	3.31	3.93	3.65	2.87	2.97	3.14	3.37
7	4.82	5.33	4.14	3.94	3.64	4.58	4.62	4.28	3.71	3.46	4.25
8	3.84	3.59	3.81	3.71	5.05	4.42	4.46	4.23	4.41	3.84	4.13
9	3.40	3.65	3.52	3.45	3.26	4.28	3.90	3.80	4.51	3.42	3.71

As we can see from *Table **1*** buttons 5, 6, and 9 have 3 lowest execution times compared to other buttons. During the test, we observed that, most users perform faster executions if the buttons are closer at the side of hand that controls *"Kinect-II"*.

Table 2 – "Grab to select" Gesture – Kinect II device

BUTTONS	USERS										Avg
	1	2	3	4	5	6	7	8	9	10	
1	1.99	3.38	3.86	3.75	2.93	2.43	3.22	2.51	2.74	3.04	2.98
2	2.32	4.04	3.71	2.10	3.03	2.36	3.71	2.41	3.47	3.25	3.04
3	1.97	2.25	1.91	2.08	1.84	2.46	2.69	1.85	2.15	1.91	2.11
4	2.10	3.05	2.81	2.04	2.37	3.03	3.57	2.84	2.79	3.07	2.76
5	1.44	1.69	1.49	2.44	1.72	2.11	2.50	1.41	1.78	2.12	1.87
6	1.70	1.44	1.97	1.52	1.57	2.03	2.18	1.84	2.12	1.77	1.81
7	2.78	4.11	2.78	4.51	4.44	4.16	3.78	3.62	3.14	4.47	3.77
8	2.90	1.91	2.10	2.05	1.97	2.30	2.72	2.55	2.34	2.71	2.35
9	3.43	1.71	3.16	4.69	2.19	2.32	3.10	2.19	3.45	3.84	3.00

In "*Grab to select*" data from *Table **2***, buttons 3, 5, 6, 8 have faster execution times than others. During the test, it is determined that users find and grab the button at center and at the side where they control *"Kinect-II"* easier. When buttons placed at corners users grab is more unstable than other button placements.

Table 3 – "Push to select" Gesture – Kinect II device

BUTTONS	**USERS**										**Avg**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
1	1.96	3.71	3.81	4.44	2.12	5.43	2.29	3.45	2.41	3.14	3.27
2	2.83	2.12	3.88	2.10	1.94	1.94	2.57	1.81	2.45	2.22	2.38
3	2.49	4.77	5.82	2.65	1.97	2.50	3.62	3.53	2.51	2.41	3.22
4	3.61	3.83	4.30	3.55	6.09	2.68	4.97	3.74	4.12	2.98	3.98
5	1.65	2.98	2.07	3.50	1.71	2.59	1.98	1.55	2.47	1.83	2.23
6	2.73	4.11	4.17	2.95	1.82	3.14	3.06	2.65	4.12	2.33	3.10
7	3.95	3.48	4.03	6.69	2.78	2.89	2.78	3.41	3.26	3.85	3.71
8	2.33	3.83	5.55	6.99	2.10	3.02	5.39	2.15	2.87	3.22	3.74
9	2.12	3.64	5.14	4.70	2.17	3.84	2.68	3.98	2.51	3.23	3.39

"Push to select" Gesture values from *Table **3*** shows us that, 5 at the center and 2 at the center top buttons have lower values compared to other buttons. According to test reviews, when users do push movement, it is observed that they partly loose the control of hand movement towards the buttons standing at sides, this causes users to get out of button area and cause *"Kinect-II"* to not register a click trigger. Users find easier to push buttons standing at the center and at top center, this way, they do not lose control of the hand movement.

4.2.3 Gesture Execution Results of Leap Motion Device

Table 4 - "Wait to select" Gesture - "Leap Motion" device

BUTTONS	USERS										Avg
	1	2	3	4	5	6	7	8	9	10	
1	4.44	3.20	3.68	3.40	3.05	3.67	3.74	3.15	2.87	2.78	3.39
2	3.57	3.27	2.95	2.88	2.89	2.31	3.48	2.79	2.88	2.80	2.98
3	3.28	2.93	2.91	3.26	3.50	4.92	3.98	2.87	3.05	2.83	3.35
4	3.61	4.79	3.78	3.56	3.48	3.99	3.93	2.70	2.82	2.73	3.53
5	4.31	3.24	3.32	3.21	3.62	2.91	3.61	2.52	2.63	2.58	3.19
6	3.43	3.16	3.26	2.68	3.15	3.78	3.35	2.63	2.70	2.93	3.10
7	3.96	3.10	2.16	3.25	2.98	3.61	3.86	2.77	2.88	2.75	3.13
8	4.41	2.68	2.68	3.35	3.17	3.65	3.53	2.67	2.63	2.73	3.15
9	2.92	2.85	5.72	2.75	3.13	3.15	3.48	2.93	3.08	2.80	3.28

When users keep their hand at detectable range of *"Leap Motion"* sensors, they centralize their hand on *"Leap Motion"* and when they centralize their hand this lets them to trigger button 5 faster than the other buttons. It is because button 5 stands at the middle of the menu screen.

Table 5 – "Grab to select" Gesture – "Leap Motion"

BUTTONS	USERS										Avg
	1	2	3	4	5	6	7	8	9	10	
1	1.34	1.54	1.25	1.52	1.79	1.88	1.15	2.29	1.79	2.42	1.69
2	1.47	1.54	1.41	1.39	1.89	1.01	1.39	0.88	1.01	1.33	1.33
3	1.96	1.41	1.15	1.77	1.73	1.40	1.31	2.24	1.74	1.38	1.61
4	1.03	0.96	1.28	2.25	1.21	1.93	1.18	1.56	1.24	1.11	1.37
5	0.91	0.89	1.12	1.42	1.39	1.56	1.15	0.88	0.91	1.01	1.12
6	1.37	1.71	1.34	1.69	1.90	1.65	1.48	1.28	1.21	1.11	1.47
7	1.84	1.72	1.54	1.66	1.29	1.08	1.95	1.08	1.33	1.01	1.45
8	1.72	1.14	1.53	1.14	1.39	1.71	1.13	0.67	0.99	1.29	1.27
9	1.42	1.49	1.24	1.61	1.74	1.18	1.88	1.54	1.77	1.18	1.50

Despite *"Kinect-II"*'s "*Grab to select*" gesture (easier to click center or top-center button *Table 2*), *"Leap Motion's"* "*Grab to select*" gesture makes interactions easier if the buttons stand at the center or bottom-center of the screen. Because users mostly lower their

hand in testing session while keeping their hand on *"Leap Motion"*. It gets easier to trigger click on the center-bottom (button 8) and at the center (button 5).

Table 6 – "Push to select" Gesture – "Leap Motion"

BUTTONS	**USERS**										**Avg**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
1	1.34	1.54	1.25	1.52	1.79	1.88	1.15	2.29	1.79	2.42	1.69
2	1.47	1.54	1.41	1.39	1.89	1.01	1.39	0.88	1.01	1.33	1.33
3	1.96	1.41	1.15	1.77	1.73	1.40	1.31	2.24	1.74	1.38	1.61
4	1.03	0.96	1.28	2.25	1.21	1.93	1.18	1.56	1.24	1.11	1.37
5	0.91	0.89	1.12	1.42	1.39	1.56	1.15	0.88	0.91	1.01	1.12
6	1.37	1.71	1.34	1.69	1.90	1.65	1.48	1.28	1.21	1.11	1.47
7	1.84	1.72	1.54	1.66	1.29	1.08	1.95	1.08	1.33	1.01	1.45
8	1.72	1.14	1.53	1.14	1.39	1.71	1.13	0.67	0.99	1.29	1.27
9	1.42	1.49	1.24	1.61	1.74	1.18	1.88	1.54	1.77	1.18	1.50

In *"Push to select"* Gesture for *"Leap Motion"*, it is the same situation here as in "*Grab to select*" Gesture of *"Leap Motion"*.

4.2.4 Results Comparing both devices for Performance Measurements

When we are writing down the results, we take each buttons' average execution time for all gesture types and represent it in graph data to be able to compare the devices' speed factor. Each gesture type is compared with its kind. It means, if the hand gesture is *"Wait to Select"* gesture the graph will share the data of this gesture for both devices. This approach is the idiom called "apples to apples" comparison.

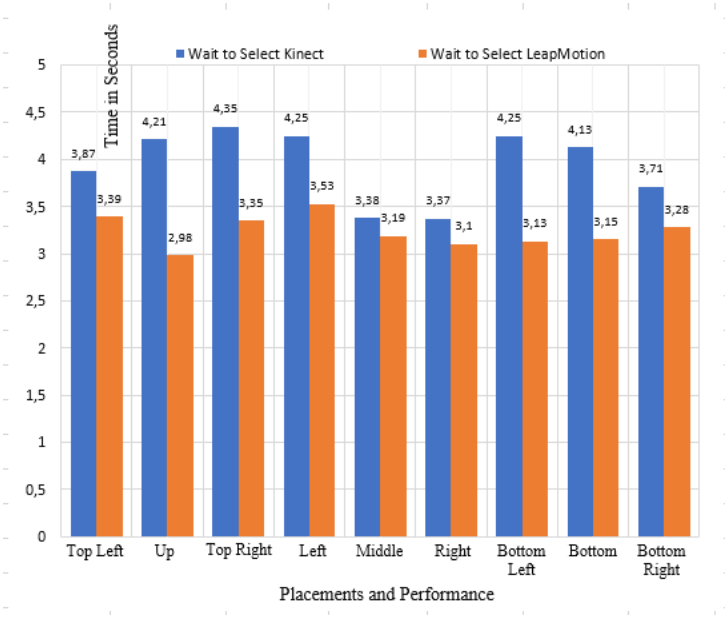

Figure 25 - Comparison of execution time for "Wait to select" Gesture on Kinect and "Leap Motion"

By comparing *"Wait to select"* gesture for *"Kinect-II"* and *"Leap Motion"*. We see that users spend more time on moving hand on *"Kinect-II"* device than the *"Leap Motion"*. Since users need to wait for 2 seconds to trigger button click all the values for this specific gesture stands above 2 seconds.

Even if in some button placements they get close values, *"Leap Motion*'s*"* execution time performs 11.22% faster than *"Kinect-II"* for Wait and Select Gesture.

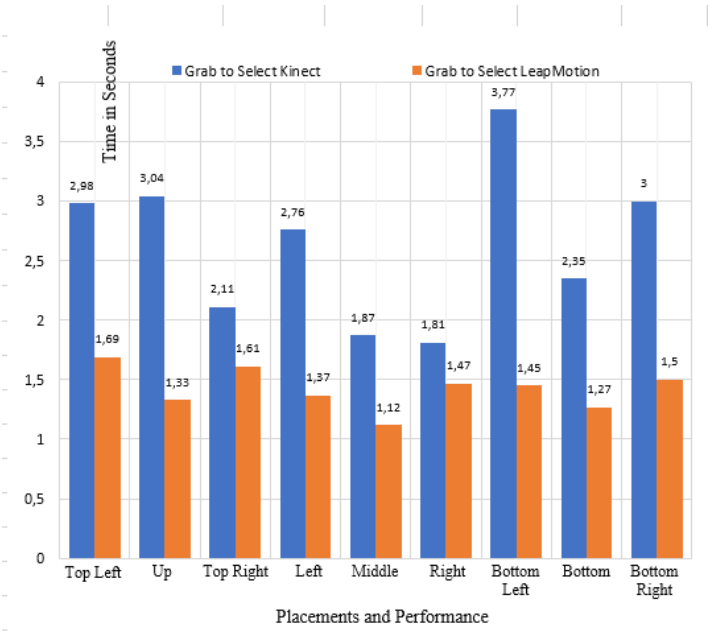

Figure 26 - Comparison of execution time for "Grab to select" Gesture on Kinect II and "Leap Motion"

When we look at the Grab and Select Gesture on both devices the gesture of *"Leap Motion's"* execution time performs 60.38% faster than *"Kinect-II"*.

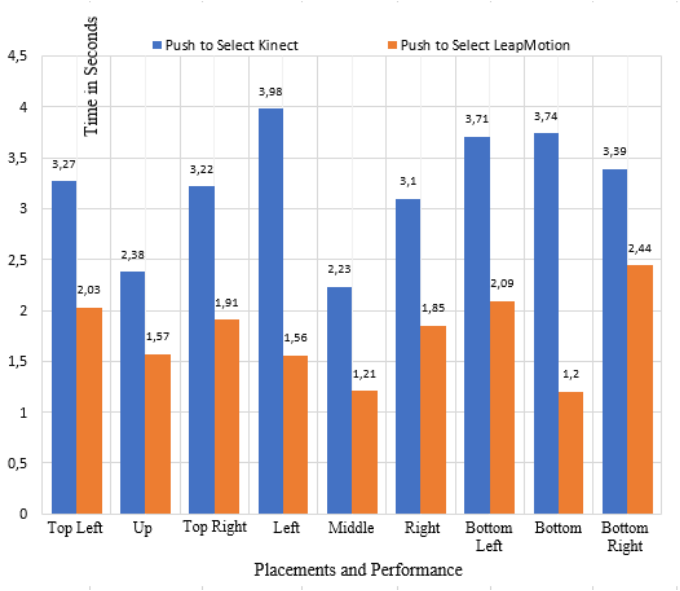

Figure 27 - Comparison of execution time for "Push to select" Gesture on Kinect II and "Leap Motion"

In *"Push to select"* gesture type for both devices, *"Leap Motion"* performs 58.64% faster than the Kinect II in terms of execution time.

4.3 System Usability Scale (SUS)

The System Usability Scale (SUS) provides a reliable tool for measuring the usability of a wide variety of products and services such as hardware, software, mobile devices, websites and applications [16]. Using this method, we prepared questionnaire asking 10 attendants to measure "Kinect" and "Leap Motions" devices' usability. The measurements in SUS method rely on 0-100 though the scores are 0-100, these are not percentages and should be considered only in terms of their percentile ranking [16]. Based on research, a SUS score above a 68 would be considered above average and anything below 68 is below average, however the best way to interpret your results involves "normalizing" the scores to produce a percentile ranking [16].

4.3.1 Kinect Questionnaire

By asking 10 attendants to measure Kinect's usability, we used known method System Usability Scale. Depending on the questions shown in **Figure 29** we evaluated the usability factor of "*Kinect*" device.

We see the results in **Figure 28** that "Kinect" devices usability is evaluated by 70.45 points which is just above average 68.5 points according to SUS.

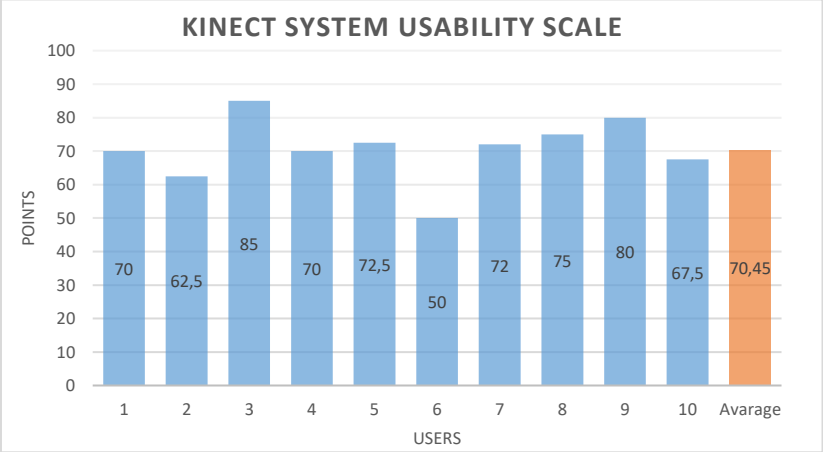

Figure 28 – Kinect System Usability Scale Graph

If to see individually how attendants answered given questions from the **Figure 29**. Users most likely satisfied using *"Kinect"* as a User Interface interaction device.

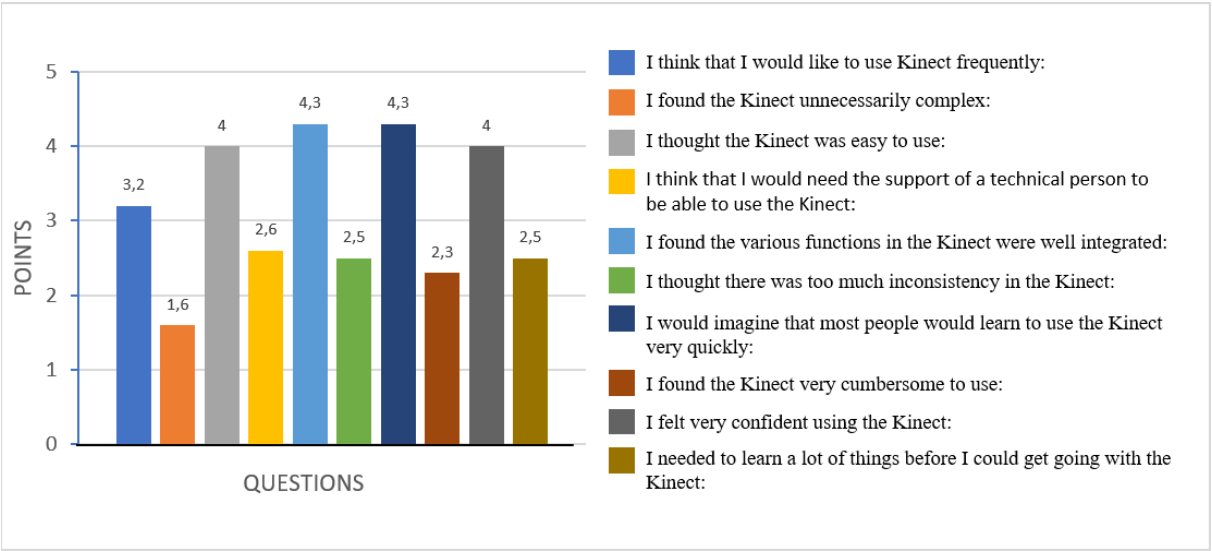

Figure 29 Results from what users think of the *"Kinect"* device. Each bar represents the score for each of the questions in the questionnaire. The left most bar is the score for the top most question, and the right most bar is the score of the bottom question.

"Kinect" for users seems to be a regular device which doesn't contain some complex setups or adjustments. They find *"Kinect"* easy to use when interacting with GUI elements but at the same time half of attendants require an assistant or a technical person to be able to use *"Kinect"*. They also seem to find *"Kinect"* technology and its functionality well integrated. Despite, its well integration users do not find it too constant in terms of responsiveness when interacting with GUI elements in the system. For most of users,

"Kinect" device is not complex to learn but it is also not the fastest device to control GUI elements when interacting with UI. After users get use to *"Kinect"* device they mostly feel confident when using it. During Interactions people half of attendants think that they need to learn more things about it before using the device.

4.3.2 Leap Motion Questionnaire

By asking 10 attendants to measure Leap Motion's usability, we used known method System Usability Scale. Depending on the questions shown in **Figure 31** we evaluated the usability factor of "*Leap Motion"* device. By looking **Figure 30** we can say that User's find usability of "Leap Motion" device for interacting with User Interfaces more efficient than "Kinect" device since its usability scale point is 87,25 which makes it a greater product.

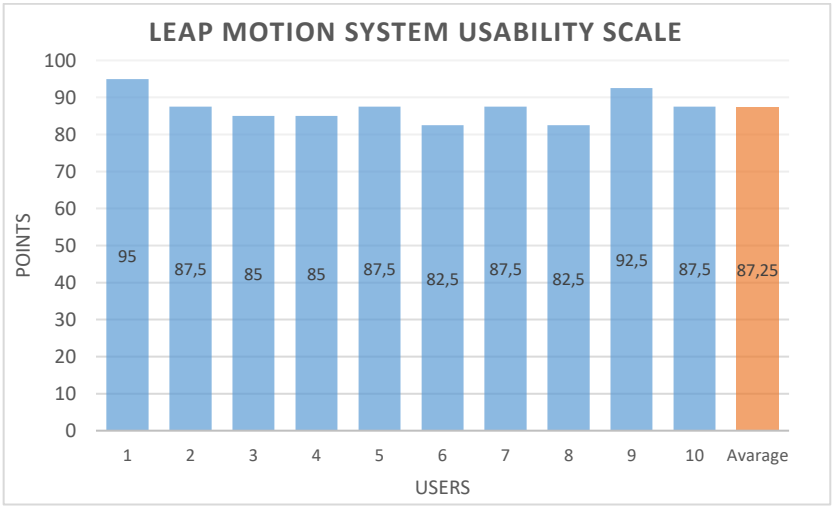

Figure 30 Leap Motion System Usability Scale Graph

Users would use more frequently the *"Leap Motion"* towards *"Kinect"* and for them Leap Motion isn't a complex device to get used to. Attendants find *"Leap Motion"* device easier to use than *"Kinect"* by small margin. Most of attendants doesn't require for an assistance or technical support to use *"Leap Motion"* device so it is also another good measurement to see that the *"Leap Motion"* is more user-friendly device than *"Kinect"* device. While attendants using *"Leap Motion"*, even if they have faced some inconsistent responses from the device, they think it wasn't interrupting the process of action. Every single attendant completely agrees on the simplicity of *"Leap Motion's"* learning process which they think it would be very quick to learn. Users think that *"Leap Motion"* is a much faster and responsive device than *"Kinect"*. Almost all attendants completely agree on the confidence of using *"Leap Motion"* which let them complete their tasks without worrying

about device's functionality. Attendants also disagree on the things that they must have learnt before using the device for completing tasks.

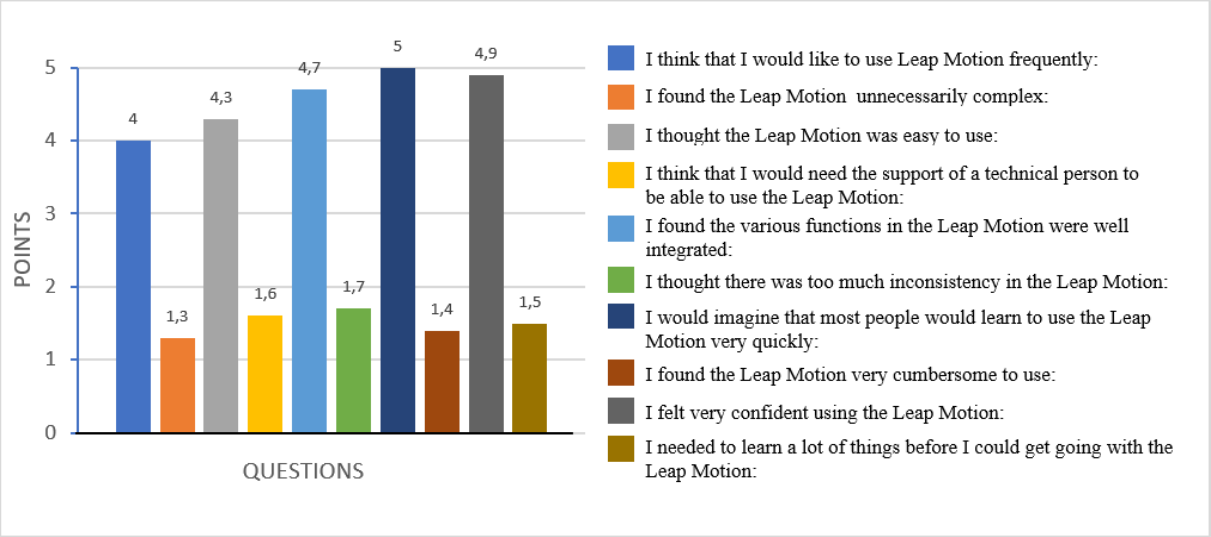

Figure 31 Results from what users think of the *"Leap Motion"* device. Each bar represents the score for each of the questions in the questionnaire. The left most bar is the score for the top most question, and the right most bar is the score of the bottom question.

4.4 Experiments Conclusion

Performance experiment shows that, *"Kinect"* device on 3 hand gestures are performing slower in terms of execution time than *"Leap Motion"* device. Overall data with exact differences for both devices can be seen for all gesture types in **Figure 25**, **Figure 26**, **Figure 27**. This also reflects users experience on both devices when we look at **Figure 32**.

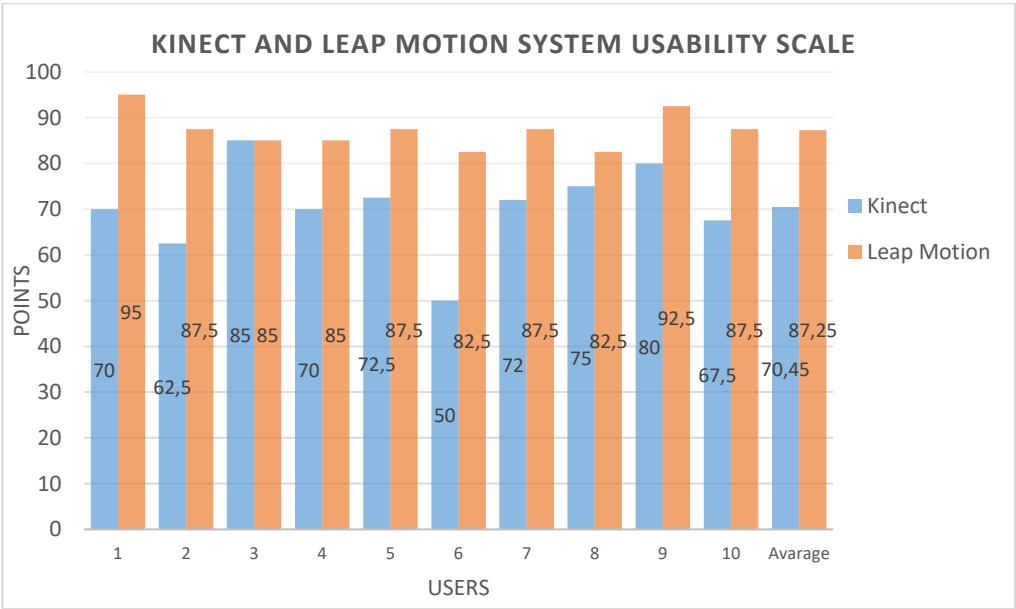

Figure 32 Comparison of Kinect and Leap Motion System Usability Scale

For users, *"Kinect's"* usability factor is lower than *"Leap Motion"*. This means, users controlling Leap Motion in brief find it more; responsive, less complex and choose it as their primary device to interact with UIs if they had option to select between two devices.

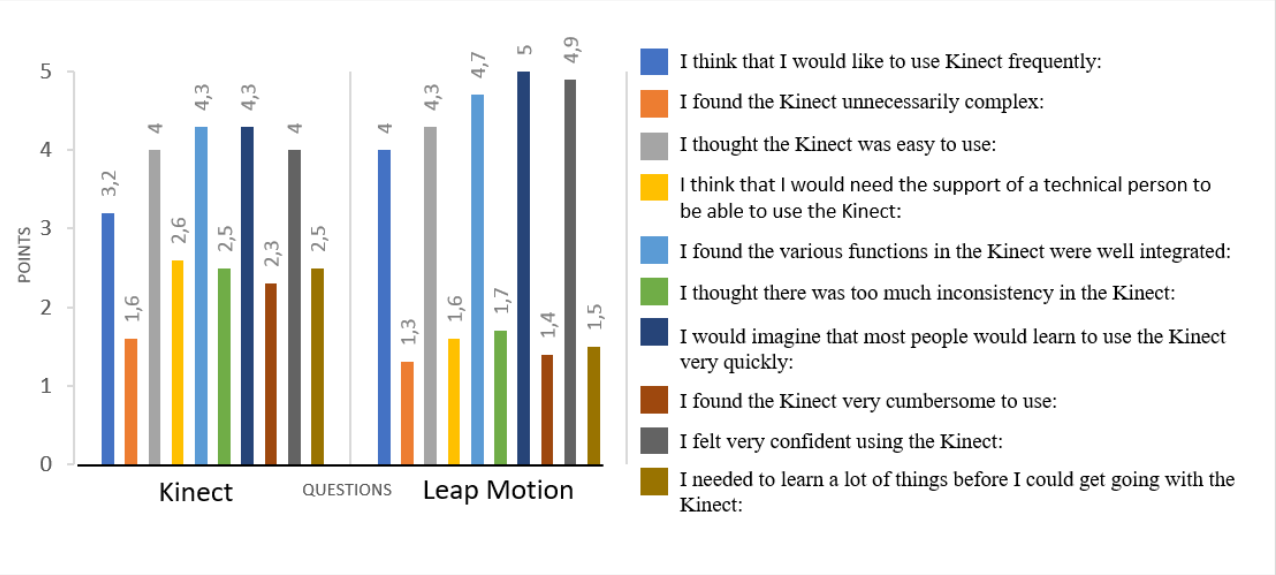

Figure 33 Results comparison from what users think of *"Kinect"* and *"Leap Motion"* devices. Each bar represents the score for each of the questions in the questionnaire. The left most bar is the score for the top most question, and the right most bar is the score of the bottom question.

During the performance experiment, we observed that, users using *"Leap Motion"* device are feeling flexible when interacting with menu than *"Kinect"*. The most reasonable fact for that is because, *"Kinect's"* performance doesn't only rely on the user behavior but it also gets effected by the environmental factors such as; light coming towards "Kinect's" infrared lights, the user's cloths color which effects depth measurements, even the distance between user and the *"Kinect"*. Even if the experiments are performed with following rules [17] the small difference in these factors mentioned before are effecting the performance of the device.

CONCLUSION

- The analyses of NUI devices show that in the market there are two widespread solutions, *"Kinect"* and *"Leap Motion"*. Therefore, we decided to do the usability study on these two devices.

- The study of hand gestures dedicated for these devices helped us to determine which gestures to be used to evaluate devices performance in terms of execution time. The analysis show that, custom gestures do not rely on standardized properties which means their implementation might differ from application to application whereas, common gestures such as; *"Wait to Select"*, *"Push to Select"*, *"Grab to Select"*) come along the *"Kinect's"* SDK which has its concrete properties. These concrete properties guided us to implement equivariant properties for *"Leap Motion"* device.

- To build testing structure of the system, the design studies were conducted to find out what user interface would cover most of the use cases for this research project. After studies, a UI design is implemented to create this structure to be able to compare dedicated 3 hand gestures.

- The decomposition of UI from SDKs allowed us to reuse code for devices to have the same evaluation scheme.

- To evaluate device and gesture usability two experiments were conducted. The experiment on usage efficiency shows that:

    - *"Grab to select"* gesture for *"Leap Motion"* gives fastest execution result at in every placement between all gesture types.
    - *"Grab to select"* gesture for *"Leap Motion"* is 60.38% faster than *"Kinect 's"* grab to select gesture.
    - We observed that *"Wait to select"* gesture for *"Kinect-II"* device performs slow if button placements are at the opposite side of hand that controls the *"Kinect-II"*.
    - We see that users make 39.4% faster executions in overall using *"Leap Motion"* when interacting with the buttons placed at the middle up, center and bottom of the screen.
    - Bottom corner sides for the placements of buttons reduces interaction effectiveness in all gesture types.
    - The experiment about System Usability Scale survey shows that the usability of "Leap Motion" device is more efficient (by 32.5 points) when comparing to "Kinect" device.

BIBLIOGRAPHY

[1] W. Zeng, "Multimedia at Work," Missouri, USA, 2012.

[2] S.-H. L. a. S.-H. Oh, "A Kinect Sensor Based Windows Control Interface," Gyeongju, 2001.

[3] Y. Li, "HAND GESTURE RECOGNITION USING KINECT," Louisville, Kentucky, 2010.

[4] A. COLGAN, "http://blog.leapmotion.com," 9 August 2014. [Online]. Available: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/. [Accessed 05 05 2017].

[5] A. A. V. G. Gabriel Anzaldo Alvarado, "Exploring the outer space with Leap," tech-zone.org, Cochabamba, Bolivia, 2015.

[6] T. M. a. J. Shin, "Hand Gesture and Character Recognition Based on Kinect Sensor," Fukushima, 2014.

[7] V. Pterneas, "Implementing Kinect Gestures," 2014.

[8] [Online]. Available: tundra.csd.sc.edu.

[9] L. Martin, "Wikipedia," Microsoft, Nike, 7 February 2017. [Online]. Available: https://en.wikipedia.org/wiki/Nike%2B_Kinect_Training. [Accessed 14 04 2017].

[10] "xiaotingrunning.files.wordpress.com," 03 2015. [Online]. Available: https://xiaotingrunning.files.wordpress.com/2015/03/java6-011.png?w=640. [Accessed 03 04 2017].

[11] J. Nielsen, "http://www.gamasutra.com," 2016. [Online]. Available: http://www.gamasutra.com/view/feature/6253/kinect_gestural_ui_first_.php?print=1. [Accessed 20 January 2016].

[12] [Online]. Available: http://www.computerworld.com.pt/media/2015/04/Vtouch-Tim-Hornyak_IDGNS.jpg.

[13] C. Benjamin, "Petrel Software Usability Study: Using the Microsoft Kinect v2 for Workflow Execution," http://brage.bibsys.no/xmlui/bitstream/handle

/11250/2352571/12753_FULLTEXT.pdf?sequence=1&isAllowed=y, Trondheim, 2015.

[14] S. Thompson, "INTERFACE ANYWHERE," in *Development of Voice and Gesture System for Spaceflight Opreations*, Sacramento, 2013.

[15] DarknessBlade, "Unity Asset Store," DarknessBlade, 01 May 2014. [Online]. Available: https://www.assetstore.unity3d.com/en/#!/content/17177. [Accessed 09 02 2017].

[16] "www.usability.govm," usability.gov, [Online]. Available: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html. [Accessed 07 03 2017].

[17] M. Kinect, "https://support.xbox.com," Microsoft, [Online]. Available: https://support.xbox.com/en-US/xbox-360/accessories/sensor-placement. [Accessed 17 04 2017].

[18] B. Salvatore, "GRAPHICAL USER INTERFACE," Austin, 2002.

[20] C. Zahumenszky, September 2010. [Online]. Available: http://www.xataka.com/tablets/mimesign-prototipo-de-control-gestual-para-tablets.

[21] F. Rumen, "Unity Assets Store," 14 08 2016. [Online]. Available: https://www.assetstore.unity3d.com/en/#!/content/18708. [Accessed 01 01 2017].