August 2014

Development and Usability Evaluation of Low-cost Virtual Reality Rehabilitation Games for Patients with Upper Limb Impairment

Jayashree Arunkumar
University of Wisconsin-Milwaukee

DEVELOPMENT AND USABILITY EVALUATION OF LOW-COST VIRTUAL REALITY REHABILITATION GAMES FOR PATIENTS WITH UPPER LIMB IMPAIRMENT

by

Jayashree Arun Kumar

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Engineering

at

University of Wisconsin-Milwaukee

August, 2014

ABSTRACT

DEVELOPMENT AND USABILITY EVALUATION OF LOW-COST VIRTUAL REALITY REHABILITATION GAMES FOR PATIENTS WITH UPPER LIMB IMPAIRMENT

by

Jayashree Arun Kumar

The University of Wisconsin-Milwaukee
Under the Supervision of Na Jin Seo, PhD

Stroke is one of the primary causes of long-term disability in adults in the United States which leads to mild to severe sensorimotor impairments. Long-term continuous rehabilitation therapies are needed to facilitate sensorimotor recovery and empower patients in performing daily living activities. Currently, the opportunity of receiving post stroke rehabilitation in the chronic stage (> 6 months post stroke) is limited due to a lack of insurance and the high cost of therapy. Low-cost virtual rehabilitation games with motion tracking devices have tremendous potential to assist physical rehabilitation. Motion tracking devices such as Kinect (Microsoft, Redmond, CA; $100) and P5 Glove (Essential Reality, LLC, NY; $40) have become available to enable development of low-cost virtual rehabilitation games. Such low-cost games may encourage continuous, repeated, and intensive rehabilitation therapies thereby enhancing recovery post stroke. However, current virtual rehabilitation games emphasize on gross arm movements using Kinect or fine finger movements using P5 Glove, but not both at the same time. Since most daily living activities require coordination of the gross shoulder/elbow movement and fine finger movement such as reaching to grasp and transferring a jar to a shelf, effective upper limb rehabilitation must involve coordination of the arm and finger

movements. In addition, many virtual rehabilitation games have been developed without user input and feedback, which may be the primary reason why virtual rehabilitation games are not prominently used at home by patients. This thesis presents the development and usability evaluation of low-cost virtual rehabilitation games. In addition to the archery and puzzle games previously developed in the laboratory, a low-cost rehabilitation kitchen game was developed to encourage patients to practice various functional tasks involving coordinated arm and finger movements that were detected by using Kinect and P5 Glove, respectively. Usability of the three games was assessed with ten chronic stroke survivors using pre-game and post-game surveys. The games met patients' expectations of providing challenging movements. The House of Quality analysis revealed that technical characteristic needing the most improvement was device reliability. The future research should address device reliability by developing a better instruction manual to facilitate device set-up and use. In addition, filtering data can also improve quality of virtual arm movements in future versions of the games. In summary, this thesis presents promising evidence for low-cost rehabilitation games using commercially available motion tracking devices of Kinect and P5 Glove together with free Blender software.

To my beloved husband Arun Kumar, who has been the backbone of all my endeavors.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisor, Na Jin Seo, PhD, who has been very helpful and supportive since our first meeting. I would like to thank my thesis committee members, Naira Campbell-Kyureghyan, PhD and Brooke Slavens, PhD, for all their guidance. I would like to extend my thanks to all of the research assistants in the Hand Rehabilitation Laboratory. I feel indebted to my family and friends who have been a great source of inspiration and encouragement at every stage of my research work. I owe a special thanks to my sweet daughter Saadhana Arun Kumar.

Finally, I would like to thank the stroke survivors who participated in the study and provided valuable feedback.

Thank you all.

1. INTRODUCTION

Stroke in adults is a cerebrovascular accident which happens due to blockage or rupture of the blood vessels in the brain (O'Sullivan and Schmitz 2007). Stroke causes brain cell death or brain damage (O'Sullivan and Schmitz 2007). Stroke is one of the primary causes of adult long-term disability in the United States and the fourth leading cause of death (O'Sullivan and Schmitz 2007, Towfighi and Saver 2011). Nearly 6.4 million Americans suffer from long-term disability often associated with upper limb impairment post stroke (Broeks 1999, Lloyd-Jones, Adams et al. 2010).

Long-term continuous rehabilitation therapies are needed to facilitate sensorimotor recovery and empower patients in performing daily living activities (Wang, Phua et al. 2009). Currently, not all stroke survivors receive rehabilitation in the chronic stage (>6 months post stroke) due to lack of insurance coverage and the high cost of physical therapy (Burdea 2002, Burke, McNeill et al. 2009). Additionally, rehabilitation therapy is often primarily focused on lower limb rehabilitation in order to regain patients' walking abilities, rather than upper-limb rehabilitation (Putman, De Wit et al. 2006). However, approximately 60% of post stroke patients suffer from persistent upper-limb impairment and are challenged in performing daily activities using the upper limb (Wade, Langton-Hewer et al. 1983, Hackett, Duncan et al. 2000, Roger, Go et al. 2011)

Virtual upper-limb rehabilitation games have tremendous potential to assist upper limb physical rehabilitation (Morrow, Docan et al. 2006). Virtual games can provide patients with a motivating environment for intense and continuous practices of active functional movements and can be accustomed to varying levels of disabilities, hence facilitating positive rehabilitation outcomes for a wide range of patients (Crosbie, Lennon

et al. 2006). For this reason, virtual reality has been identified as one of the promising tools used in many fields of therapy and rehabilitation such as physical therapy, psychiatry, and cognitive rehabilitation (Rizzo, Bowerly et al. 2002, Zimand, Anderson et al. 2002, Glanz, Rizzo et al. 2003). Over the past couple decades, such virtual rehabilitation games have been developed to facilitate stroke survivors' upper-limb functional recovery (Subramanian, Knaut et al. 2007, Duff, Chen et al. 2010).

Specifically, virtual rehabilitation games for stroke survivors have been designed using high-tech systems. For instance, three dimensional (3-D) infrared motion capture systems such as Optotrak Certus^{TM} (Northern Digital Inc., Waterloo, Ontario, Canada) (Subramanian, Knaut et al. 2007) and Motion Analysis (Motion Analysis Corporation, Santa Rosa, CA, USA) (Duff, Chen et al. 2010) have been used to track and record patients' upper-limb motion for virtual games in real-time, allowing for goal-oriented interaction that encourages repetitive training in arm movements. In addition, since upper-limb functional tasks often involve not only the arm movement but also the finger movement, CyberGlove (CyberGlove Systems LLC, San Jose, CA, USA) has been used to track finger posture (Merians, et al., 2011). While these high-end motion tracking systems have high accuracy, rehabilitation games using these systems are often unaffordable for patients and most clinics and thus impractical. Therefore, there is a need for low-cost virtual rehabilitation games, allowing for a cost-effective alternative for patients and clinics.

Affordable virtual rehabilitation games can be possible by using free software and low-cost motion tracking devices. Specifically, Blender is free and open-source 3-D computer graphics software to create animation, visual effects, art, interactive 3-D

models and video games. In addition, many commercially-available low-cost motion tracking devices are widely used in gaming. Such commercially available low-cost motion tracking devices include Nintendo Wii (Nintendo, Redmond, WA, USA), Leap Motion (Leap Motion Inc., San Francisco, CA, USA), P5 Glove (Essential Reality Inc., LLC, NY, USA), and Kinect (Microsoft Inc., Redmond, CA, USA).

Nintendo Wii and Kinect both convey arm movement data to the console and into the gaming system. Nintendo Wii, which is widely commercially available, uses game controllers that are grasped by the entire hand that measure hand velocity and displacement. However, Nintendo Wii only conveys information relating to the position of the hand in space, and does not provide positional data of the individual upper-limb joints. Kinect recognizes joint positions of the whole-body and allows the position of 20 joints in a 3-D space to be conveyed to the system (Fig. 1). Since upper limb physical rehabilitation focuses on achieving various arm postures such as elbow extension and shoulder elevation, not only the hand location in space but also information on the whole arm posture is important for virtual rehabilitation games. Therefore, Kinect is a better alternative for relaying upper limb posture for virtual rehabilitation game development compared to Nintendo Wii.

Kinect costs approximately $100. Kinect captures joint position at a 30 Hz sampling rate which is adequate for tracking users' movements in real time (LaBelle 2011) and can detect users standing between 1.2 m to 4.7 m from the device. Specifically, Kinect has a capture volume of 17 m^3 with a horizontal field of view of 4.9 m, vertical field of view of 2.8 m, and a depth field of view that starts at a distance of 1.2 m from Kinect and is up to 4.7 m from Kinect (Fig. 2). Even though Kinect is superior in

measuring upper-limb joint position data compared to Nintendo Wii, both systems lack the ability to measure finger joint posture which is important in designing games involving hand grasping tasks.

Figure 1: (a) Kinect and (b) The 20 joint positions detected by Kinect

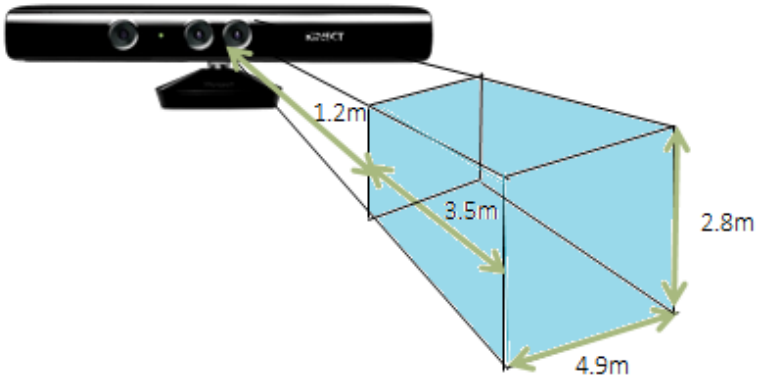


Figure 2: Capture volume of Kinect

Both Leap Motion and P5 Glove are low-cost devices that measure finger bending and wrist position. Leap Motion provides motion capture for the fingers and wrist. However, Leap Motion can detect finger posture only when fingers are open, but not when the fingers are closed in a fist. Thus, Leap Motion is limited in measuring finger postures during tasks involving grasping, which is one of the major components of upper-

limb functional tasks. Alternatively, P5 Glove is capable of measuring finger opening and closing postures. Therefore, P5 Glove is a better alternative for measuring finger bending for virtual rehabilitation games.

P5 Glove (Essential reality, New York, New York, USA) (Fig. 3) is a commercially available device that costs approximately $40. P5 Glove can track the hand's 3-D position (i.e. X, Y, and Z coordinates shown in Fig.3, (Davison 2007) yaw, pitch, roll (Fig.4), and finger bending (flexion/extension) angles. P5 Glove is plugged into the sensor receptor (Fig. 3), which is hooked up to a computer through a USB port. P5 Glove has a maximum 45 Hz sampling rate for the 3D position/orientation of the hand and maximum 60 Hz for the finger bending. The hand position is tracked optically using an infrared LED sensor receptor (Fig.3). The glove is portable, has an ergonomic design, weighs approximately 0.12 kg, and wearable on the hand (Morrow, Docan et al. 2006), although stroke survivors with spasticity may have trouble putting it on as with any other wearable devices. P5 Glove can track hand movements up to a distance of 1.2 m from the sensor receptor.

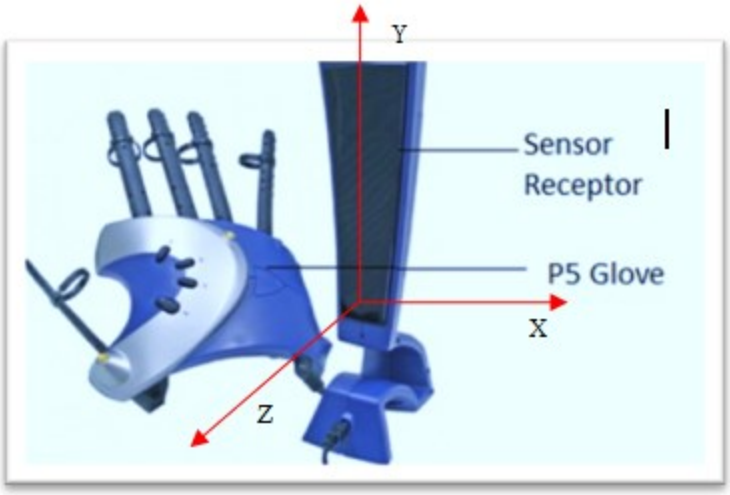


Figure 3: P5 Glove and sensor receptor showing the 3-D coordinates (Davison 2006)

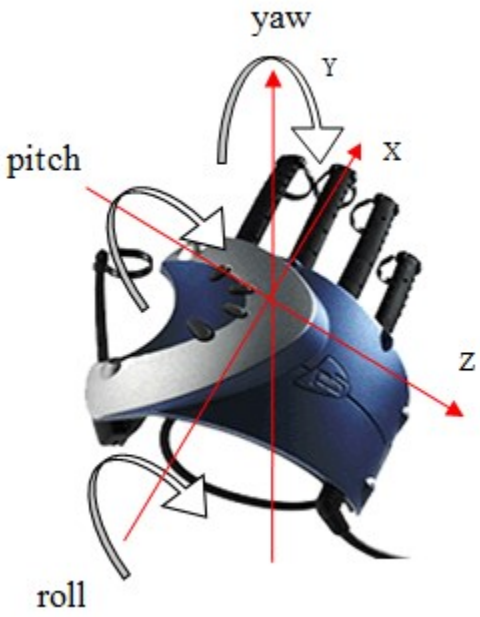


Figure 4: P5 Glove showing the 3-D orientation information

Together, Kinect and P5 Glove have the capability to communicate position information of the upper-limb joints and finger bending for virtual rehabilitation games. While there exists rehabilitation games that use Kinect for gross arm movements only

(Roy, Soni et al. 2013) or P5 Glove for fine finger movements only (Morrow, Docan et al. 2006), there is currently no rehabilitation game that involves both of the devices to measure gross arm and fine finger movements at the same time. Finger and arm coordination is critical for upper limb function in daily activities such as reaching to grasp a cup and transferring a jar to a shelf (Carroll 1965, Wade, Langton-Hewer et al. 1983). Therefore, there is a need for virtual rehabilitation games to focus on improving both gross and fine motor abilities of the arm and hand. In addition, many games are developed by engineers with minimal feedback from patients. Therefore, developed games may not effectively motivate patients nor be liked by patients, especially those suffering from very limited range of motion, such as stroke survivors. Lack of user input and feedback in the development of rehabilitation games is a major problem, since motivation and likability are crucial for patients to adhere to a rehabilitation regime for successful outcomes (Luck 2003). Therefore, in order to address these issues in current virtual rehabilitation games, this thesis focuses on the development and usability evaluation of low-cost virtual rehabilitation games for coordinated arm and finger movements of stroke survivors using Kinect and P5 Glove. Specifically, the following two aims are investigated.

Aim 1: To develop a low-cost virtual rehabilitation game for finger and arm coordination.

Aim 2: To evaluate usability of the low-cost virtual rehabilitation games.

2. AIM 1:

TO DEVELOP A LOW-COST VIRTUAL REHABILITATION GAME FOR FINGER AND ARM COORDINATION

2.1 Introduction

We developed a virtual rehabilitation kitchen game for enhancing finger and arm coordination in stroke survivors using the low-cost motion tracking devices of Kinect and P5 Glove. This kitchen game simulates tasks that are generally performed in a kitchen setting and involves grasping, moving, and putting away plates and utensils. Thus, the kitchen game focuses on functional activities of daily living to a greater extent than the previous games that were developed in the laboratory. The previous games that were developed in the laboratory are the archery game and puzzle game (Crocher, Hur et al. 2013). The archery game requires patients to control and orient a bow and arrow with their arm, while requiring patient to use opening and closing finger motions to release the arrow and shoot at the targets. The puzzle game requires patients to grab virtual puzzle pieces resembling states within the United States and place/orient them in their correct locations on a United States map.

2.2 Kitchen Game Features

The low-cost virtual kitchen game (Fig. 5) had a virtual arm that mimicked a user's upper-limb movements in real-time using Kinect and P5 Glove. The game required the user to perform a variety of functional tasks that involved grasping, moving, and putting away kitchen utensils. The tasks were inspired by the clinical test, Fugl-Meyer Assessment (Duncan, Propst et al. 1983) and focused on flexion/extension of the digits, grasping objects of different sizes and shapes, forearm pronation/supination, elbow

extension, and shoulder abduction. Different scenes were designed for different tasks (Fig. 6).

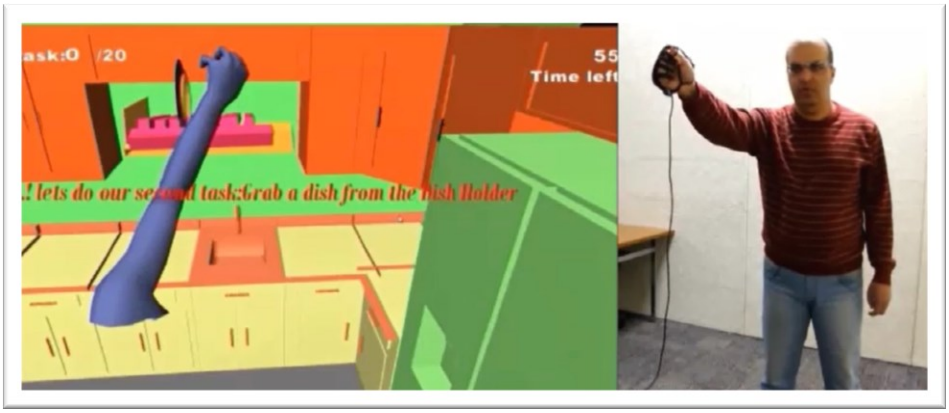


Figure 5: The virtual arm in the game (left) mimics the user's upper arm movements (right)

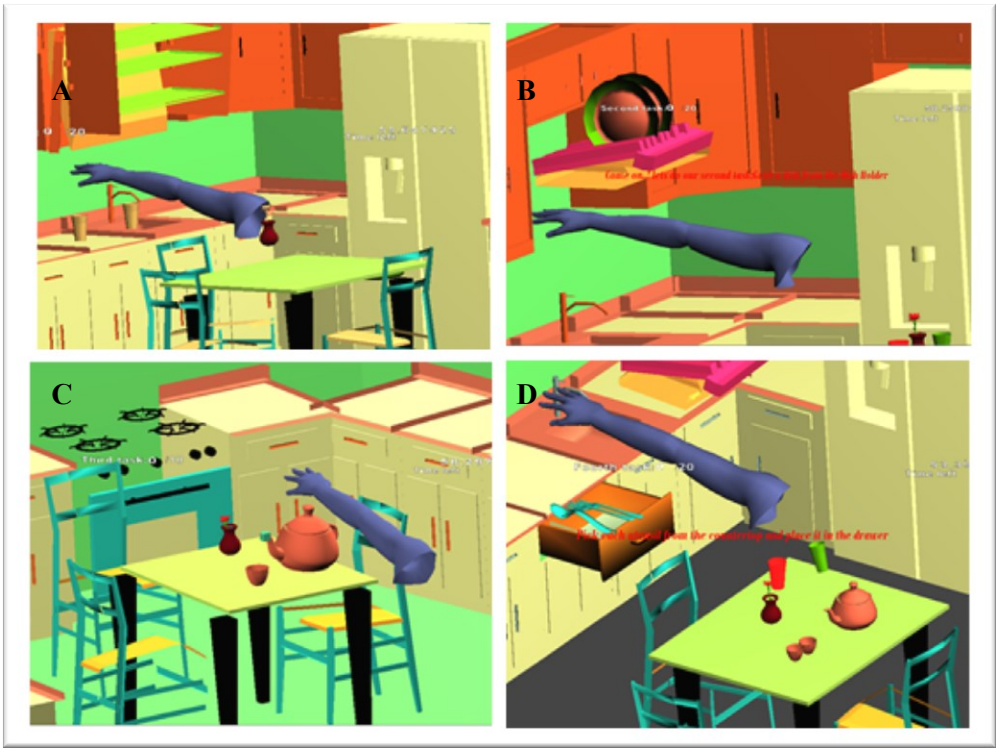


Figure 6: Different scenes used for different tasks in the kitchen game: (A) Task for grasping cups from the countertop and placing them on the overhead compartment, (B) Task for grasping the dishes from the dish holder and placing them on the countertop,

(C) Task for grasping the teapot handle and pouring water from the teapot to a teacup, and (D) Task for opening a drawer, grasping and moving silverware from countertop to inside the drawer.

Instructions were provided in the game so that the users could understand the gameplay and the sequence of each task that needed to be played. Users could track their game score in the upper left corner of the monitor (Fig. 7). Also, they could monitor the elapsed time while playing (Fig.7). Motivation to keep playing the game and move to the next task was provided in the form of a visual and audio cue. For instance, after successfully completing a task, the user was awarded stars and cheering phrases such as "Wow", "Good Job", "Keep up the good work", and "One task to go" (Fig. 8).

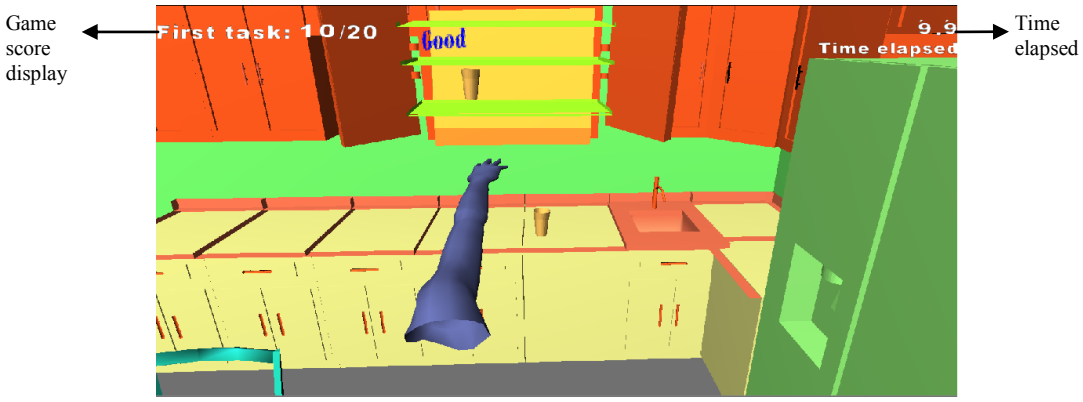


Figure 7: During the kitchen game, the game score was shown in the upper left corner and the elapsed time was shown in the upper right corner of the screen.

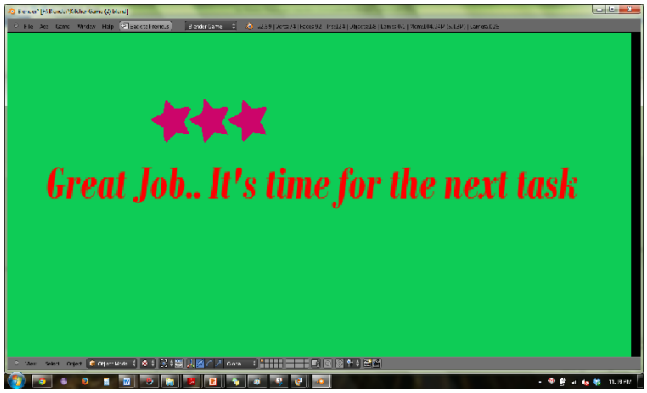


Figure 8: Praises were provided to a user upon successful completion of a task

2.3 Kitchen Game Workflow

The basic game workflow included motion tracking devices to acquire the motions performed by the user and then a program to process the input data provided by the motion tracking devices (Fig.9). First, the client program received input from the motion tracking devices and sent the data to the server program. Secondly, the tracked data from the server program was used in joint angle computation. Hence, the joint angle information (section 2.3 iv) was then used in the interaction script and that data was processed and sent to the game engine. Finally, the game engine sent the data to the audio and video output modules and these output modules generated the virtual reality environment that could be experienced by the users through computer monitor and speakers. Internal Client/Server architecture within the program received input from the motion tracking devices, computed the joint angles, and sent the results to the game engine. The virtual arm model was designed using the armature bones and mesh in Blender. The interface with the motion tracking devices to mimic the users' movements and simulate interaction with objects in the game was programmed using Python. The objects in the game obeyed the laws of physics and reacted to gravity to make the objects fall to floor when a user dropped the object. Fig.9 shows the basic workflow of the virtual rehabilitation games and each section is explained in detail below.

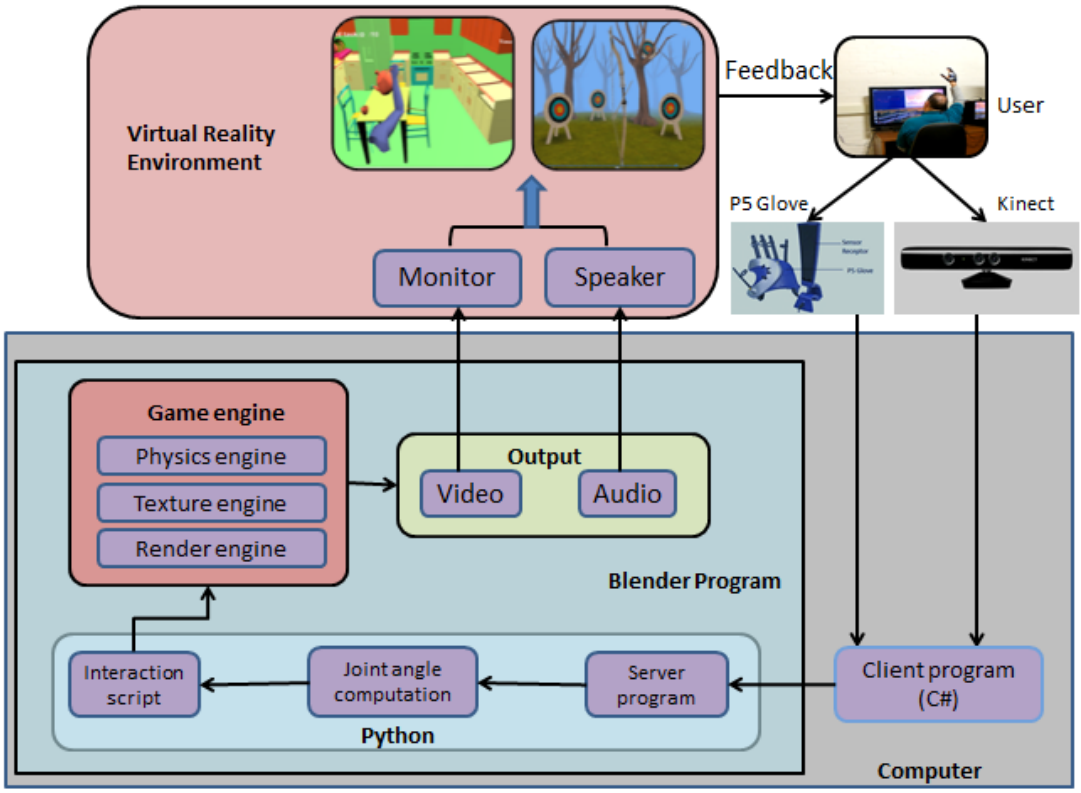


Figure 9: Basic workflow and system components of the kitchen virtual rehabilitation game

i. Motion Tracking

The user's upper-limb movements were tracked using the motion tracking devices of Kinect and P5 Glove. Kinect was used to detect gross arm movements and P5 Glove was used to detect finger motions (Fig. 10). Manufacturer-provided calibration software was used to calibrate P5 Glove (Fig. 11).

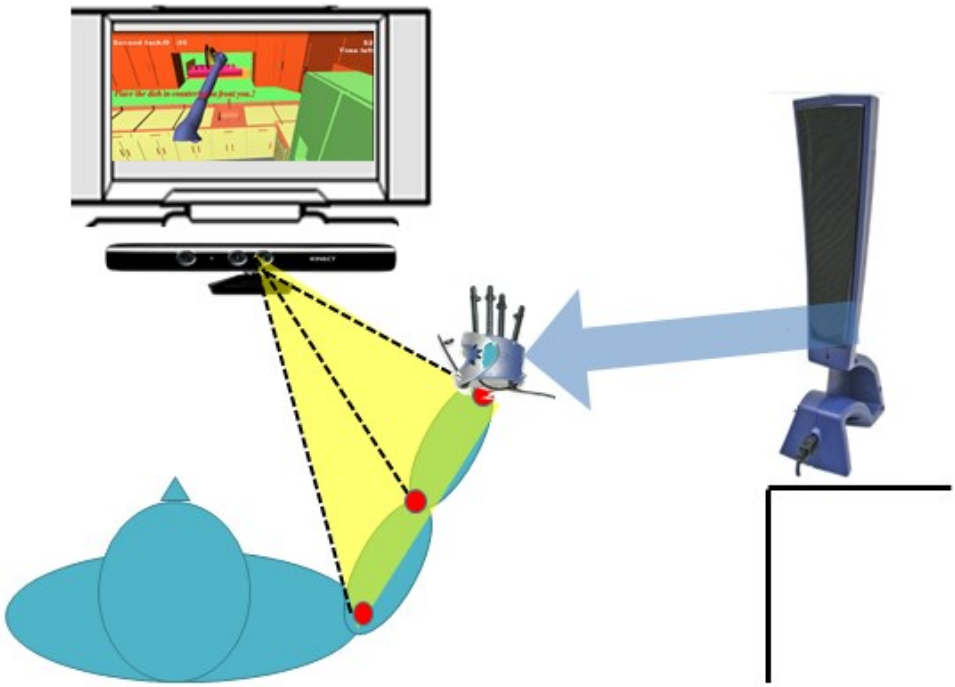


Figure 10: Schematic diagram shows a user playing the kitchen game with Kinect detecting the gross arm movements and P5 Glove detecting finger movements

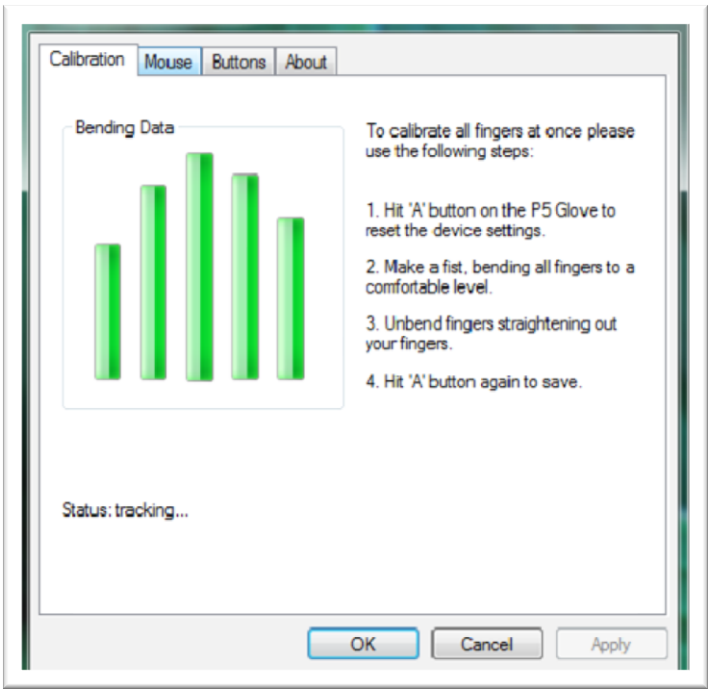


Figure 11: P5 Glove calibration

ii. Client program

A client program used Kinect to locate the upper-limb joints of the user in 3-D space and P5 Glove to detect bending of the fingers and track their movements in real time. The program was written in C# because; Application Programming Interfaces (API) for both Kinect and P5 Glove are available in C#. The Kinect for Windows Software Development Kit (SDK) provides the tools and APIs, needed to develop Kinect-enabled applications. Skeletal tracking information provided in Microsoft developer network (msdn) was appropriately used to tweak and incorporate the respective user joint positions to build the program. This program was used to send real time Kinect and P5 Glove data that provided the arm postures and finger flexion/extension of the user to the server program. Data was sent from the client program to the server via a secured networking protocol called User Datagram Protocol (UDP). UDP is used for sending data over the network using minimal protocol mechanism (Postel 1980).

iii. Server program

The server program received real-time Kinect and P5 Glove data sent by the client program (Fig.9). The server program was developed in Python because Blender supports Python.

iv. Joint angle computation

Using the user's arm and finger posture data received by the server program, elbow and shoulder joint angles were computed in Python. The wrist angle was not computed since accuracy for Kinect to detect the wrist angle is poor based on unpublished data in our laboratory. Each of the elbow and shoulder angles was computed as follows:

a) Elbow angle

The elbow angle was defined as the angle made by the forearm and extension of the upper arm (Fig. 12). If the 3-D positions of shoulder, elbow, and wrist are shown as S, E, and W vectors respectively, then the forearm vector (V1) and the upper-arm vector (V2) was calculated as:V1 = W-E, V2 = E-S. Equation (1) was used to perform elbow angle computation.

$$\theta = \cos^{-1}\left(\frac{V_1 . V_2}{|V_1||V_2|}\right) \quad (1)$$

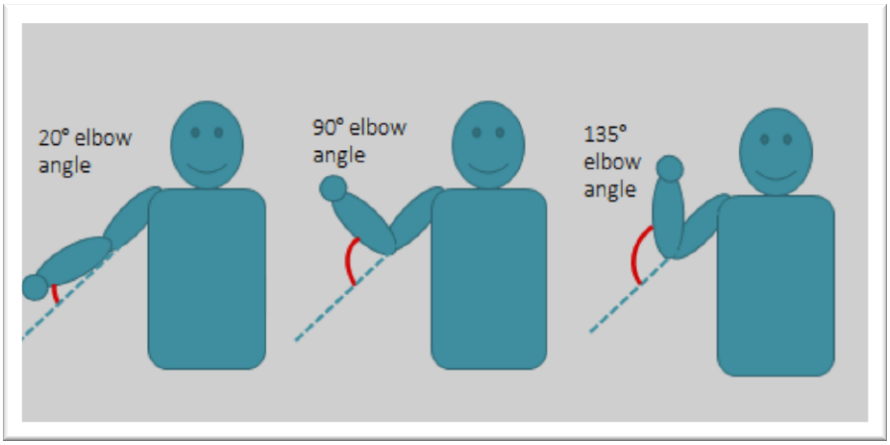


Figure 12: Diagram depicting 20 º, 90 º, and 135 º elbow angles.

b) Shoulder azimuth angle

The shoulder azimuth angle was defined as the angle between the upper arm and the sagittal plane as seen in the Fig.13. Equation (1) was used to compute the azimuth angle with V1 being the vector of the upper arm projected on the horizontal plane and V2

being the unit vector of the forward horizontal direction (intersection between the sagittal and horizontal planes).

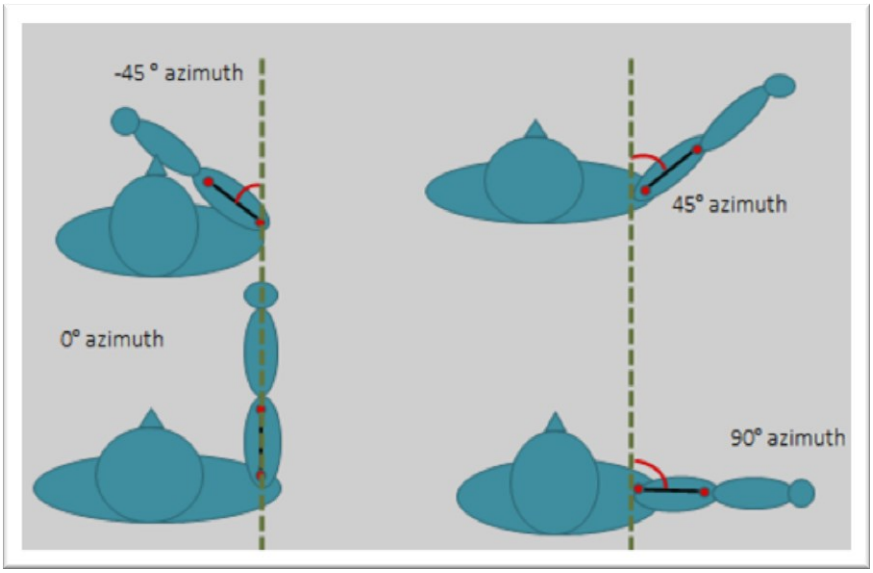


Figure 13: Diagram depicting the -45 º, 45 º, 0 º and 90 º shoulder azimuth angles

c) Shoulder elevation angle

The shoulder elevation angle was defined as the angle made by the upward-downward motion of the arm with respect to the body (Fig.14). Equation (1) was used to compute the shoulder elevation angle with V1 being the vector of the upper arm and V2 being the unit vector of the downward direction (intersection between the sagittal and frontal planes).

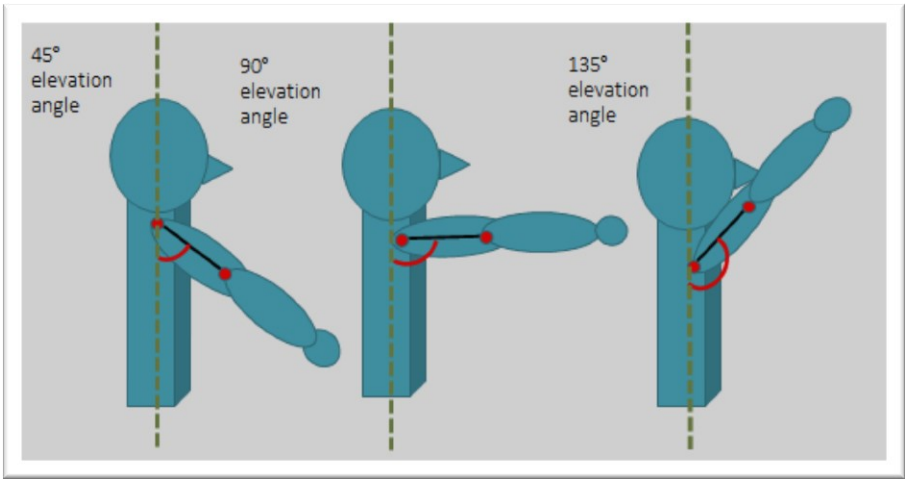


Figure 14: Diagram depicting the 45 º, 90 º and 135 º shoulder elevation angles.

d) Shoulder rotation angle

The shoulder rotation angle was defined as the internal/external rotation angle of the upper arm around its own axis (Fig 15). Equation (2) was used to compute the shoulder rotation angle. First, the vector normal to the plane of the arm (V_3) was computed by finding the cross product between the upper arm vector (V_1) and the forearm vector (V_2). Then, the normal vector (V_3) was projected on the horizontal plane (V_4). The shoulder rotation was calculated as the angle between the normal vector (V_3) and its projection on the horizontal plane (V_4).

$$\theta = \cos^{-1}\left(\frac{\boldsymbol{V_3}.\boldsymbol{V_4}}{|\boldsymbol{V_3}||\boldsymbol{V_4}|}\right) \quad (2)$$

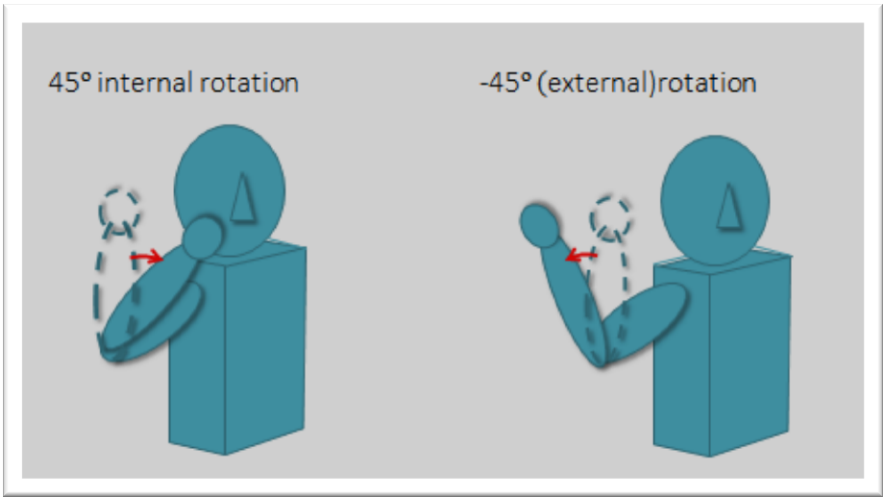


Figure 15: Diagram depicting 45º and -45º rotation of the shoulder.

v. Interaction script

An interaction script was written in Python to control the interaction between the hand and movable objects (cups, dishes, teapot and silverware), so that those objects could be grasped, moved and dropped. Specifically, when the distance between the hand and the desired object was less than a preset value, that object could be grasped by closing of the fingers. In that case, the interaction script changed the state of that object to prevent the physics engine from controlling it, thus its location could be controlled by the hand. On the other hand, by opening the fingers, the grasped object was released, in which case the state of the object was returned back to its default setting allowing the physics engine to control the object movement according to the physics laws again.

vi. Game engine and output

Based on the upper limb posture and object information determined in the interaction script, the game engine updated the virtual reality environment (Fig. 9). Specifically, input modules were updated by incorporating physics engine and textures/render engine, along with the elbow and shoulder joint angles to make the virtual

arm mimic the upper limb movements performed by the user. Physics engine attributed physical laws to the objects in the game making them obey dynamics and gravity. Textures added extra details to the surface of the objects, which was achieved by projecting images or patterns on the surface. Render engine provided a fine quality image of the developed 3-D scene. These game engine data determined video and audio outputs that were fed into a computer monitor and speaker, respectively. These visual and audio displays provided the user with experience in the virtual reality environment.

2.4 Kitchen Game Activities

The kitchen game had game activities to practice functional tasks involving finger and arm coordination to impact daily upper limb function (Stanger, Anglin et al. 1994) and also to improve the clinical upper limb score of the Fugl-Meyer Assessment (a standard motor impairment scale for stroke survivors as an index for rehabilitation outcome). Specifically, the Fugl-Meyer Assessment emphasizes on movements requiring control and coordination of multiple upper-limb joints such as reaching forward and up (shoulder elevation) with elbow and finger extension, shoulder elevation with forearm rotation, and shoulder internal rotation while maintaining the elbow posture. Therefore, kitchen tasks involving coordination of multiple upper limb joints were featured. All tasks involved grasping and releasing of kitchen items with the hand to practice hand-arm coordination. The game involved four different tasks (Fig.6). In the first and second tasks, the user was asked to move glasses from the counter to an overhead compartment and to move plates from the overhead compartment to the counter, respectively. These tasks represent typical functional activities of reaching, grasping, and releasing.

Clinically, these tasks focused on coordinated shoulder elevation, elbow extension, and finger flexion/extension (Fig.16A, B) that are relevant to the Fugl-Meyer Assessment. The third task had the user grasp a teapot and pour water into a cup to practice shoulder internal/external rotation (Fig. 16C). In the fourth task, the user had to open a drawer, pick up a spoon and a fork one by one from the countertop and place it inside the drawer. This task involved practice of forearm pronation and coordinated shoulder, elbow, and hand movements (Fig. 16D) in addition to reaching, grasping, and releasing.

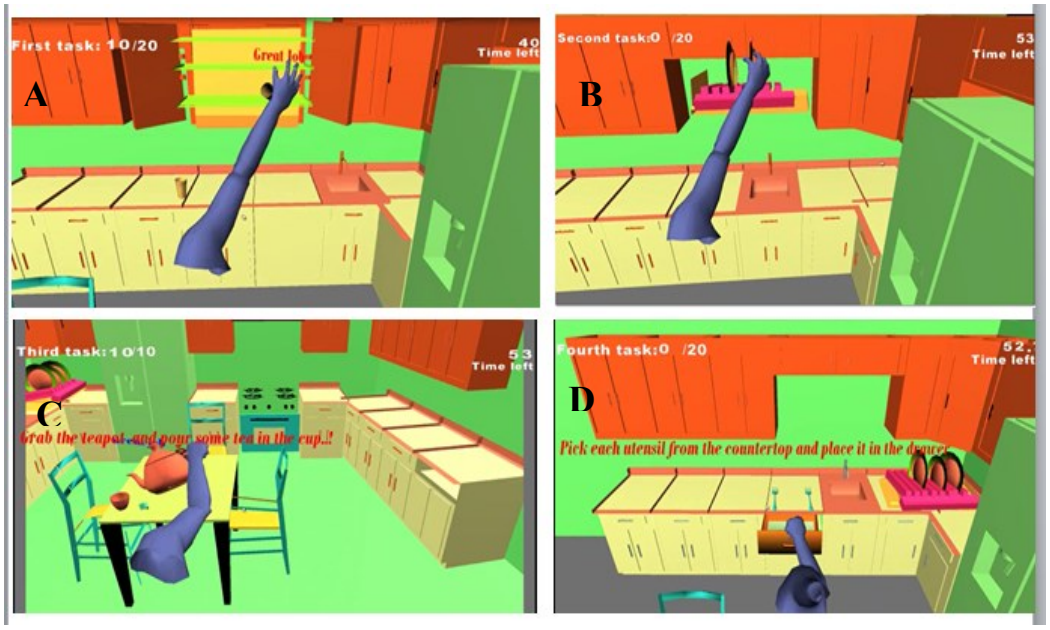


Figure 16: A few postures involved in the kitchen game activities. (A) task involving shoulder elevation, elbow extension, and finger extension posture, (B) task involving shoulder elevation, elbow extension, and finger flexion posture, (C) task involving shoulder rotation posture , and (D) task involving forearm pronation and elbow flexion posture.

2.5 Clinical Relevance

The game can provide not only game scores but also relevant clinical information such as the range of motion, movement speed, and time to complete given tasks that may help clinicians understand the progress that patients are making. To demonstrate that the game is capable of providing relevant clinical information, the range of motion observed

for each task and for each joint as well as time to complete each task obtained while one person played the kitchen game are described in Table 1.

Table 1: Range of motion and time elapsed observed during the kitchen game

Kitchen tasks	Elbow angle range	Shoulder azimuth angle range	Shoulder elevation angle range	Shoulder rotation angle range	Time elapsed
Grasping cup from countertop and placing it in overhead compartment	84º	160º	104º	175º	40sec
Grasping dishes from dish holder and placing it in countertop	124º	108º	141º	179º	30sec
Grasping teapot and pouring tea from it	107º	96º	79º	148º	20sec
Grasping different sizes of silverware and placing it in the table cabinet after opening it.	114º	134º	69º	131º	45sec

2.6 Conclusions

Aim 1 was to develop a low-cost virtual rehabilitation kitchen game for finger/arm coordination. This aim was achieved using affordable Kinect and P5 Glove motion tracking devices and free Blender software. This rehabilitation games

demonstrated a strong potential and feasibility for low-cost rehabilitation systems which could be used at home or a clinical environment. The expected cost is $140 including the hardware and free open-source software. The game requires patients to employ a range of motor functions and repetitive movements that are ideal in upper limb rehabilitation therapies (Sveistrup, McComas et al. 2003). The requested movements in the game have potential to train the upper limb motor functions and may allow for recovery by providing more practice while keeping patients motivated.

3. AIM 2:

TO EVALUATE USABILITY OF THE LOW-COST VIRTUAL REHABILITATION GAMES

Usability assessment is a crucial step in designing a product because it is a way to optimize product design by identifying weak areas in the product's concept, design and user interface (Lange, Flynn et al. 2009, Lange, Rizzo et al. 2011). If the product goes to market without a thorough usability assessment and subsequent design and quality improvements, the product is at a high risk of failure because of the lack of interest and motivation associated with poor usability. Such failure results in a great loss of labor and development cost. Hence, it is an industry standard to evaluate usability of a product.

Usability of a product can be assessed in form of questionnaires, focus groups, task analysis, user observation, interviews and surveys after users interact with the product. The usability of our virtual rehabilitation games was tested using House of Quality (HOQ). HOQ belongs to a management approach called Quality Functional Deployment (QFD). HOQ has been widely adopted in Japan and has gained popularity in the U.S. as well (Hauser 1993). After its inception in Japan in 1972, HOQ is now used by many major developers such as Hewlett-Packard, AT&T, Ford, General Motors and Toyota (Hauser and Clausing 1988). Toyota's auto body startup and production costs have reduced 61% after implementing HOQ and QFD (Sullivan 1986). All three games developed in our laboratory – the kitchen game described in Ch. 2 as well as the archery and puzzle games described in Crocher et al. (2013) – were tested for usability in this study.

3.1 Methods

3.1.1 Subjects

Ten stroke survivors (five males and five females, ages ranging from 43 to 76 years with a mean of 63 year) participated in this study. The inclusion criteria used to recruit subjects was that the subject must be a chronic stroke survivor (>6 months post-stroke). Traditional health insurances do not cover for extended durations of physical rehabilitation, and stroke survivors are commonly believed to have reached a recovery plateau within 6 months post stroke (Page, Sisto et al. 2004, Krakauer 2006). However, recent evidence suggests that targeted therapy and exercises can help stroke survivors achieve meaningful motor function improvements and improve physical fitness and cardiovascular health (Mark and Taub 2004, Page, Sisto et al. 2004, Billinger, Arena et al. 2014). Therefore we were interested in understanding how we can improve our current games to make them more interesting and user-friendly for chronic stroke survivors. Subjects were excluded from the study if they had botulinum toxin injection within the past 3 months from the day of study and/or if the subject had cognitive impairments. Botulinum toxin injection reduces spasticity in stroke survivors. The subjects that undergo the botulinum toxin treatment may not represent the general populace of stroke victims we are aiming for. Also, subjects with cognitive impairment were excluded because the game tasks required ability to understand and follow commands. All subjects signed a consent form and followed a protocol approved by the Institutional Review Board. Table 2 shows the patients' demographic information along with their functional evaluation scores. Subjects' upper extremity function was evaluated by a physical therapist in the lab using Chedoke McMaster Stroke Assessment and Fugl-Meyer score

(Fugl-Meyer, Jääskö et al. 1974, Gowland, Stratford et al. 1993, Sanford, Moreland et al. 1993).

Table 2: Patient demographic information

Characteristics	**Range**	**Mean**
Age in years	43 to 76	63
Years post stroke	3.5 to 13	8
Fugl-Meyer score (out of 66)	2 to 66	43
Chedoke score (out of 7)	1 to 7	5
Modified Ashworth score	0 to 3	1.5

3.1.2 Study Protocol

Subjects answered pre- and post-game surveys before and after interacting with the games for half an hour (Fig. 17). These surveys were then used to construct the HOQ. The pre-game survey was designed to better understand the user expectations about the virtual rehabilitation game systems. The survey included 10 questions related to user expectations on usability and functional implications of the low-cost virtual rehabilitation games. Total ten criteria to be used in HOQ analysis were extracted from these survey questions, which are: (1) easy to understand, (2) easy to use, (3) adaptation of the game to the patient's functional ability/ improvement, (4) interesting (5) challenging, (6) graphics quality, (7) progression score, (8) variety of different scenes, activities and games, (9) integrating clinical assessment (10) proven clinical effect (Fig. 18B).

The post-game survey was similar to the pre-game survey with the same 10 criteria but re-worded to determine which expectations were met with the games. The subjects were asked to select an answer on a Likert scale of 1-5 (least to most satisfactory). Pre and post-game surveys provided data to construct the HOQ matrix in order to quantitatively identify the technical characteristics of the game that should have the highest priority for improvement for future development. Also, the post-game survey included an open ended feedback to provide remarks about the games.

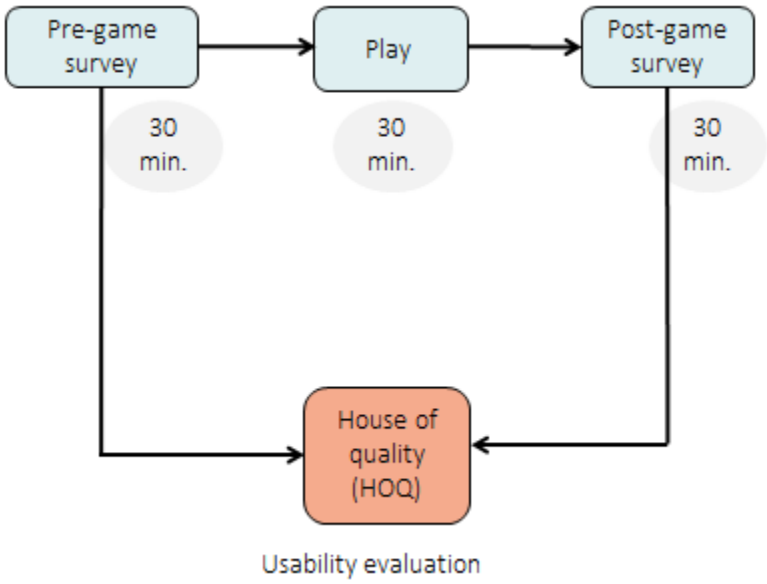


Figure 17: Usability evaluation workflow of the rehabilitation games. Patients' feedback was analyzed in this pattern.

During the gaming session, subjects were given the instruction manual (see Appendix) for the games, a computer running Windows 7 with the games already installed, and the hardware (Kinect and P5 Glove). The instruction manual described where to place the hardware and how to wear P5 Glove, turn the hardware on, and run the game software before being able to play the games.

3.1.3 House of Quality Analysis:

An HOQ matrix is an orderly way of defining improvement priorities among the technical characteristics of a product to effectively respond to user expectations in future designs of the product (Logan and Radcliffe 1997). The HOQ is completed in following steps: (1) The engineer determines the technical characteristics, **j** (Fig. 18A). (2) The engineer determines the patient expectation criteria to be included in the pre-game survey, **i** (Fig. 18B). (3) The relationship between technical characteristics and the patient expectation criteria are determined by the engineer for the interrelationship matrix index, $\mathbf{I_{ij}}$ (Fig. 18C), which has three levels (strong relationship as 9, moderate relationship as 3, and weak relationship as 1, Fig. 18D). (4) The expectation weight for each criterion is calculated as the mean of patients' pre-game survey scores, $\mathbf{W_i}$**.** (5) The response weight for each criterion is calculated from the patients' response in the post-game survey, $\mathbf{R_i}$**.** (6) Priority weight ($\mathbf{P_j}$**)** for each technical characteristic is computed using Equation 3, using the interrelationship matrix index ($\mathbf{I_{ij}}$), the pre-game survey ($\mathbf{W_i}$), and the post-game survey ($\mathbf{R_i}$). (7) The priority weight is expressed into percentage. (8) The technical characteristic with the highest priority weight is considered to have the maximum need to be improved.

$$P_j = \sum_{i=1}^{N} W_i \times I_{i,j} \times (5 - R_{\mathrm{i}}) \tag{3}$$

The technical characteristics included are shown in Fig. 18A. Specifically, the installation manual represents the presence, quality, and understandability of a paper-version installation manual. Game instructions are the instructions that show up on the

computer monitor during the game play to prompt the user for next necessary actions as the user progresses through the game, such as the words shown in Figure 5. Device reliability is the probability of a device performing its required function and producing same results on repeated trials (Miller, Epstein et al. 1985). Game reliability represents the game's ability to provide scores in a consistent manner. Game adaptation represents the game's ability to change its contents according to the user's functional ability and improvement over time, possibly by using Artificial Intelligence (AI). The number of levels represents various levels of difficulty and challenges in the game for the user to work through. Game realism represents the user's perception that the virtual environment is real, facilitated by realistic objects, environment, sounds, textures, and physics simulation. 3-D display is concerned with the technical decision on whether high quality 3-D display would be needed or 2-D display would suffice for rehabilitation games to satisfy users' expectation. Clinical assessment represents administration of clinical tests such as the Fugl-Meyer Assessment within the virtual game setup without clinicians' presence in order for clinicians to track and easily understand recovery and progress of the patients during the course of game usage. Lastly, clinical evidence represents clinical trials that demonstrate that the virtual rehabilitation game effectively enhances users' upper limb function.

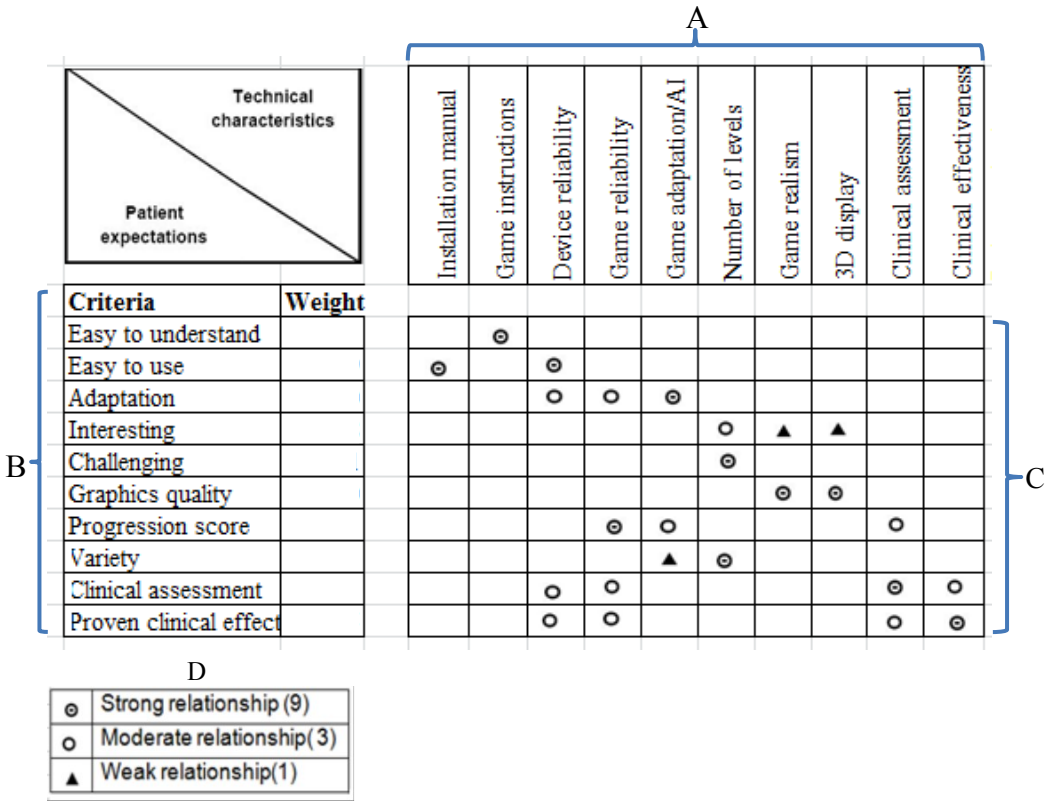


Figure 18: (A) Technical characteristics of the games (j). (B) Patient expectations from pre-game survey (i). (C) Interrelationship matrix showing the relationship between technical characteristics and the patient expectation criteria (I_{ij}), (D) Levels of relationships for the interrelationship matrix.

3.2 Results

3.2.1 HOQ results

The pre-game survey showed that the highest expectation that patients had of virtual rehabilitation game criteria was for the game to be challenging, shown by a mean ± standard deviation (SD) expectation weight of 4.1 ± 0.6 (Fig. 19) out of a highest score of 5 on a Likert scale. These pre-game survey results of the expectation weights were provided in the left column of the HOQ matrix in Fig. 20. The lowest criterion that patients rated for expectation was the graphics quality with the expectation weight of 3.0 ± 1.3. Other criteria were found to be moderately important to the users.

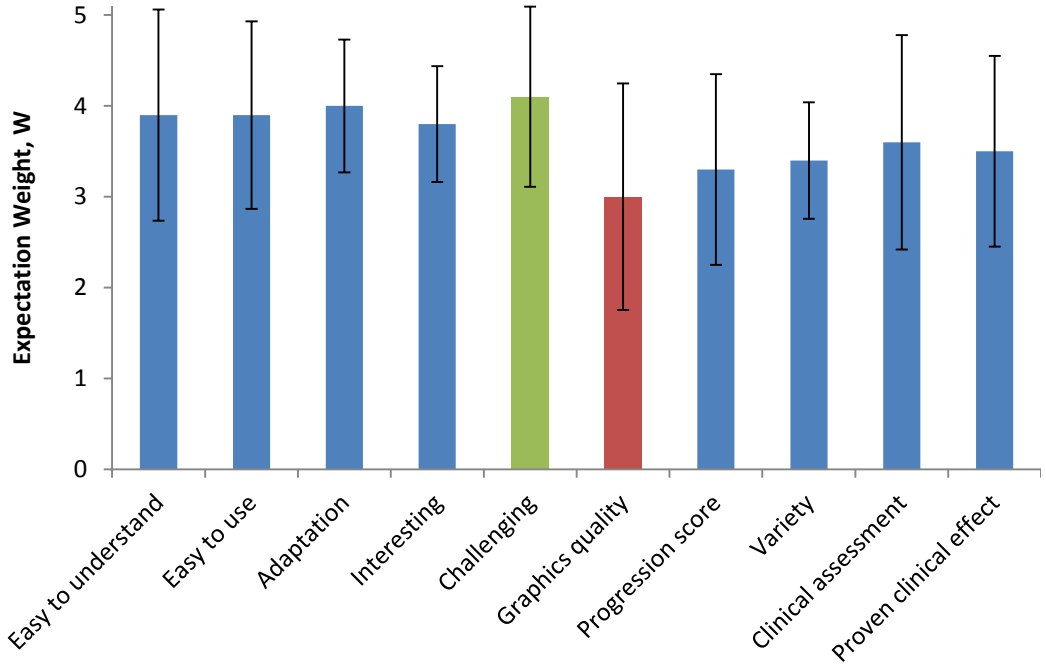


Figure 19: Mean ± standard deviation (SD) of the patients' expectations of virtual rehabilitation games based on the pre-game survey criteria, referred to as the expectation weight, W. The expectation criterion of challenging (green bar) was weighted the highest and the graphics quality (red bar) was the lowest weighted expectation in the pre-game survey.

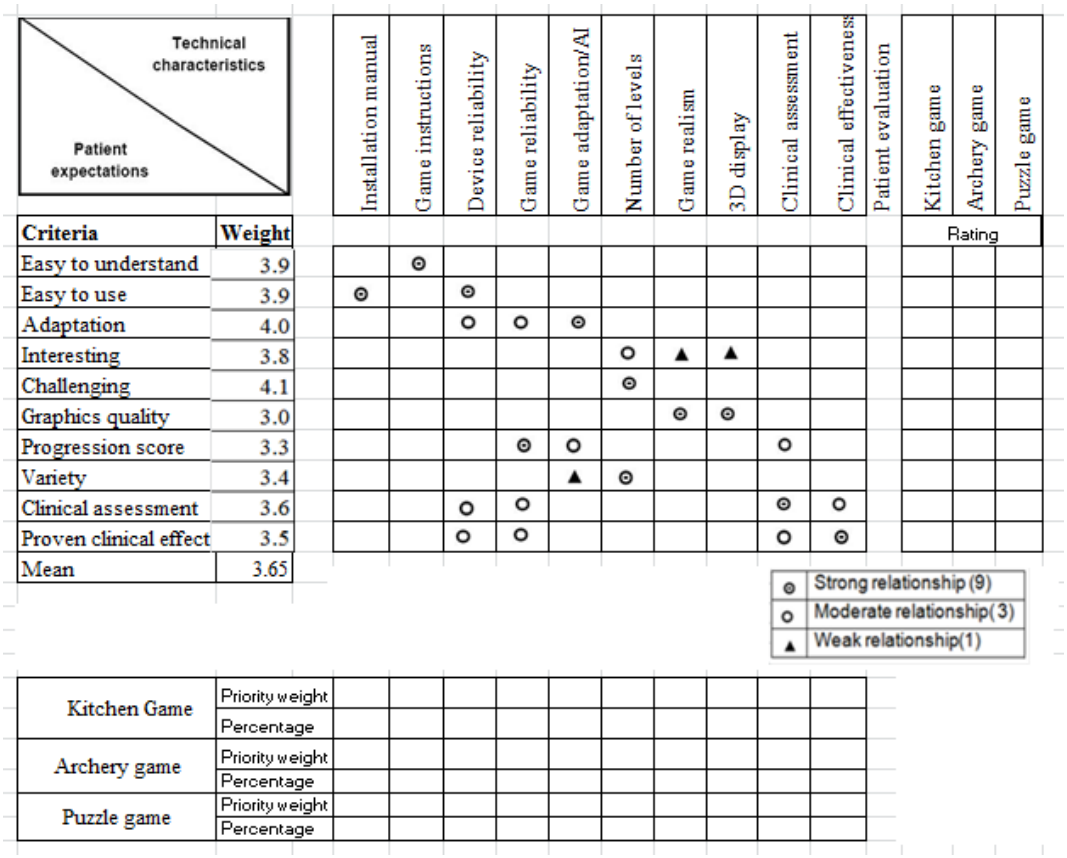
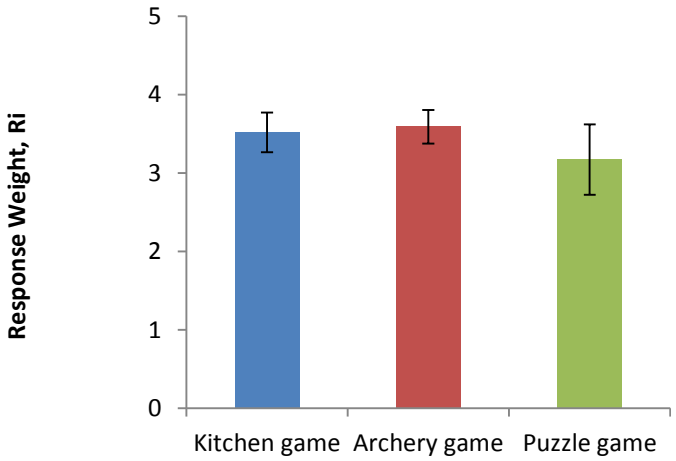
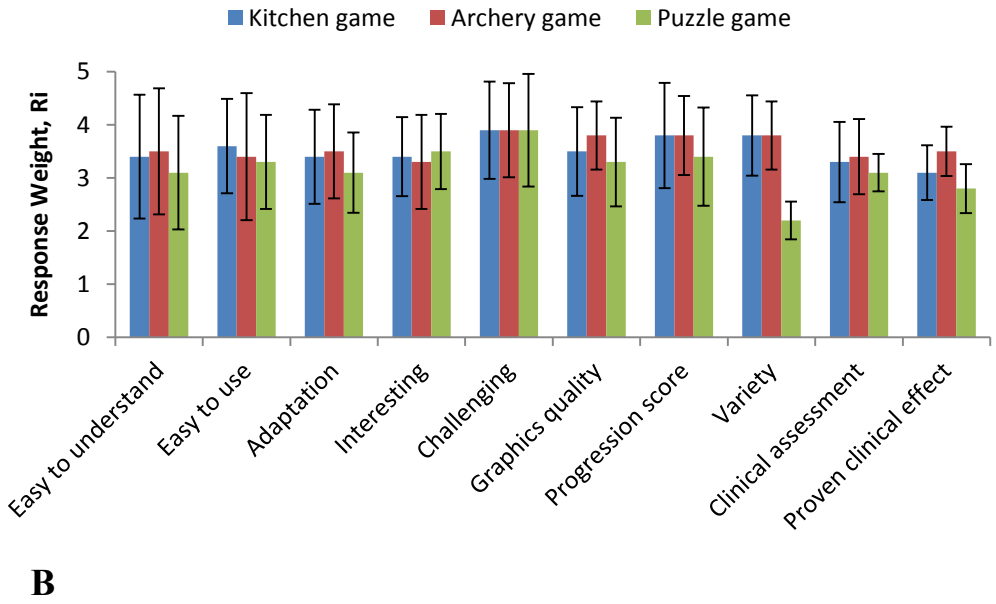


Figure 20: Patients' expectations of virtual rehabilitation games based on the pre-game survey criteria, referred to as the expectation weights, shown in the left column of the HOQ matrix

The post-game survey results showed that the kitchen game and the archery game were similar in patient evaluation with a response weight of 3.5 ± 0.3 and 3.6 ± 0.2 out of a highest score of 5 on a Likert scale, respectively (Fig. 21A). The puzzle game had the lowest overall response weight of the three games with a response weight of 3.2 ± 0.5. When examining individual criteria, all of the games achieved the highest response weight of 3.9 for the challenging criteria (Fig. 21B). These post-game survey results were also provided in the right column of the HOQ matrix in Fig. 22.



A



B

Figure 21: Mean ± SD of the patients' response weight, R_i, for each game for all of the criteria combined (A) and for each criterion (B) based on the post-game survey.

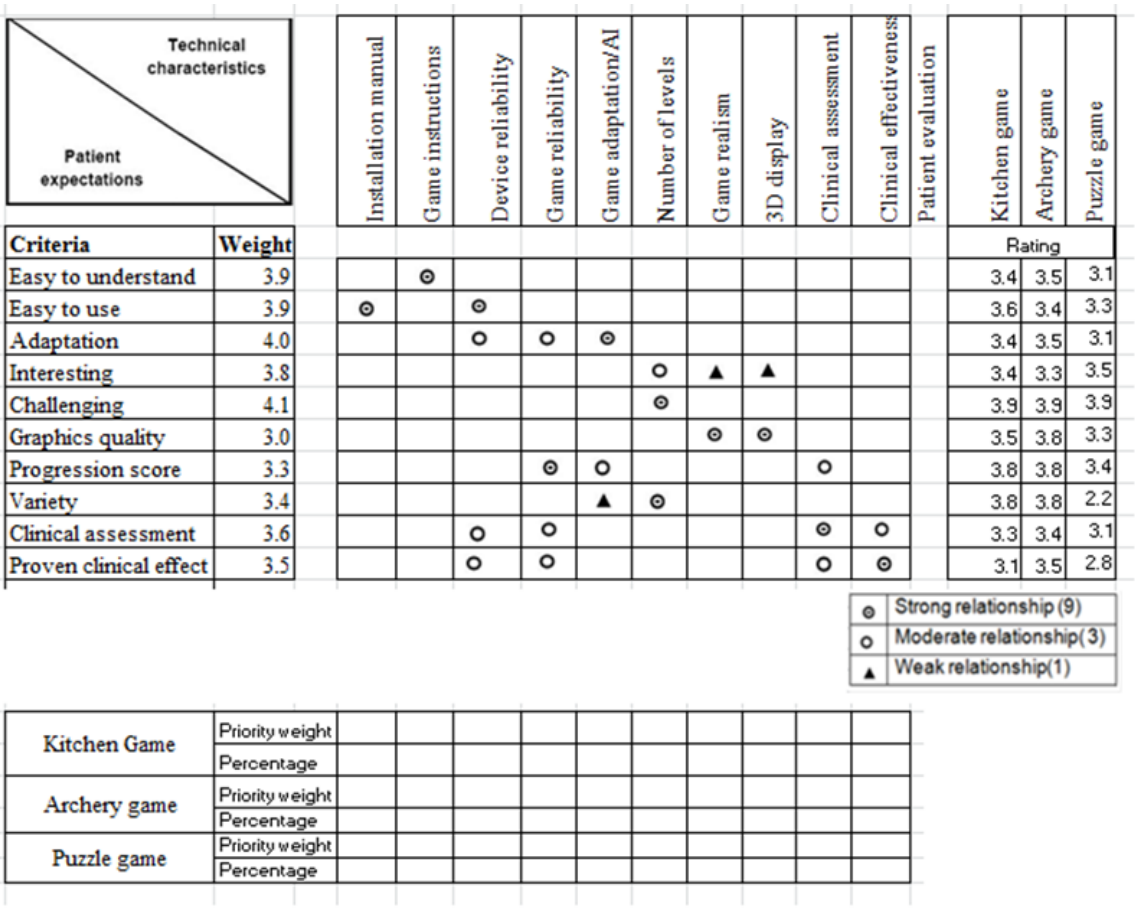


Figure 22: Patients' evaluations of the virtual rehabilitation games based on the post-game survey, referred to as the response weight, are shown in the right column of the HOQ matrix

Priority weight results are shown in Fig. 23. The priority weights were computed using Equation 3, with the expectation and response weights obtained from the pre- and post-game surveys. Device reliability was rated with the highest priority weight, meaning device reliability needs the most improvement according to the HOQ analysis (Fig. 23). It was also noticeable that the priority weights for four technical characteristics (installation manual, game instructions, game realism, and 3-D display) were comparable to each other and were lower than priority weights of other characteristics (Fig. 23). These four characteristic were considered to have the lowest need for improvement in our study.

Rest of the technical characteristics had medium need for improvement. The moderate priority technical characteristics were game reliability, game adaptation/ AI, number of levels, clinical assessment, and clinical effectiveness. These results were also provided in the bottom rows of the HOQ matrix (Fig. 24). The technical characteristic with the highest priority are in red text, followed by moderate priority characteristics in brown, and lastly characteristics with lowest priority need in green. When priority weights for technical characteristics were averaged within a game, HOQ showed that device reliability obtained priority weight of 15% for the kitchen game, 16% for the archery game and 15% for the puzzle game.

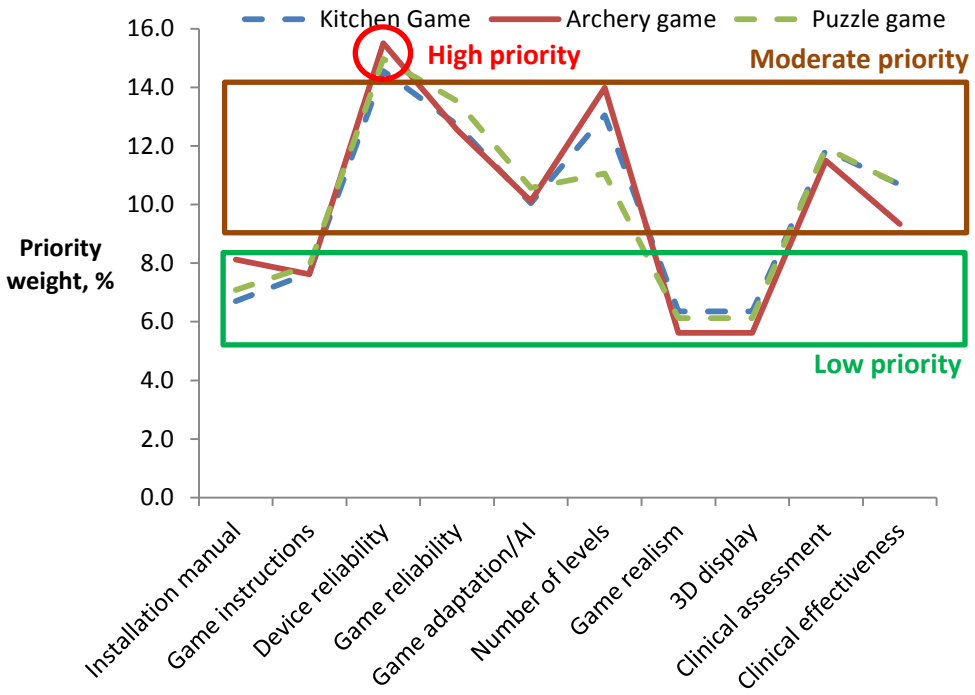


Figure 23: Priority weight for each technical characteristic in each game was divided into highest, moderate, and the lowest priority need. Based on HOQ, device reliability showed the highest priority need for improvement.

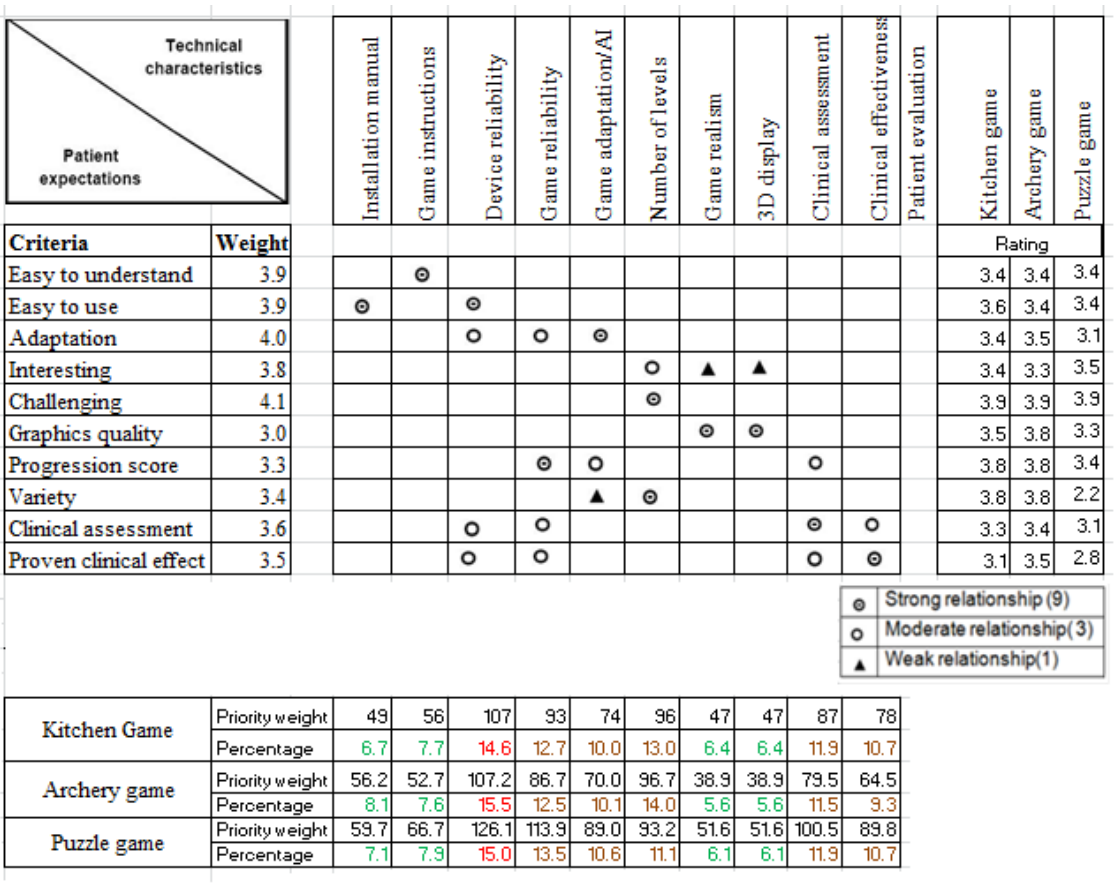


Figure 24: The HOQ matrix for a low-cost VR games identified priority needs as an outcome (bottom row), based on patients' expectation ratings (left column), the game's technical characteristics (top row), interrelationship matrix (center), and patients' evaluation of the game (right column). Red, brown, and green numbers indicate the highest, intermediate, and the lowest technical improvement.

The open-ended feedback collected based on the post-game survey supported HOQ results. Specifically, open-ended feedback showed that patients really enjoyed that the games were challenging and fun. Some of the comments obtained were: (1) "The kitchen game was interesting, the plates fell just like real [plates upon dropping], the tea kettle made real house [looked real], and silverware was challenging to lift".

(2) "The archery game was challenging. The bow showed easy direction to shoot the targets". (3) "The puzzle game was challenging and the states were easy to understand

and I like the color scheme". Open-ended feedback also showed device reliability issues. For instance, one patient commented "Kinect and P5 Glove didn't work all the time. [My] Frustration level was high".

3.3 Discussion and Conclusions

The usability evaluation of the virtual rehabilitation games concluded that there is additional scope of work available in the area of device reliability. Device reliability pertains to the motion tracking devices performing their required function. It was observed during the game play that Kinect and P5 Glove sometimes could not track a patient's arm and finger movements well because the patient's arm was outside the capture range. Therefore, one of the ways to improve device reliability for the virtual rehabilitation games may be to improve the instruction manual to visualize the capture range so that users gain better understanding of the capture volumes of each motion tracking device. Moreover, the games may be modified to generate a warning sign when the movements go out of the capture volume. Filtering of the data could also improve smoothness of virtual arm movements. Advanced movement prediction algorithms to compensate for the device reliability related issues may help usability of the virtual rehabilitation games. In summary, to have the virtual rehabilitation game accepted by patients, improvement on device reliability was identified as a priority development requirement based on the usability evaluation (Fig. 23, Fig. 24).

In addition, the result of the pre-game survey questionnaire clearly indicates (Fig.20) that the patients expect more of challenging games than that of a high quality graphics. The post-game survey results show that the patients' highest expectation

criterion of ‘challenging game’ was met without compromising the graphics quality expected by the patients (Fig. 22, 4.1/5 pre-game expectation weight vs. 3.9/5 post-game response weight for challenging). The resulting mean response weight of 3.5/5 from post-game survey shows that the evaluation results of all three games indicate a good overall rating and likeability (Fig. 22).

4. DISCUSSION AND CONCLUSION

4.1 Low-cost rehabilitation game development

This study used commercially available motion tracking devices of Kinect and P5 Glove and free Blender software for developing a low-cost virtual rehabilitation game that practices arm and finger coordination. While Kinect (Roy, Soni et al. 2013) has been used for rehabilitation games that involve gross upper limb movements of the shoulder and elbow, Kinect alone is unable to measure finger motion. Hence, P5 Glove, which detects finger flexion/extension associated with grasping and releasing of objects, was combined with Kinect in this thesis to complete the motion tracking for the whole upper-limb including fingers. The innovation of this thesis is the combination of Kinect and P5 Glove motion capture systems, because such combination has previously not been used for a virtual rehabilitation game. As mentioned before, most daily living activities involving the upper limb require coordination of the gross and fine motor movements of the arm and fingers such as reaching and grasping of objects with elbow extension and forearm rotation. Thus it becomes of foremost importance to have an effective upper limb rehabilitation approach which involves coordination of the arm and finger movements. The kitchen game developed in this thesis also provides clinically relevant information such as the joint range of motion and time to complete each task, facilitating clinicians' understanding of patients' progress.

The major benefit of Kinect and P5 Glove is that they are less expensive compared to other position tracking devices such as the Optotrak and CyberGlove, respectively (Table 3). In addition, use of the free open-source Blender software helps keeping the total cost down. A cost comparison between the traditional motion capture

and virtual environment systems and the low-cost virtual rehabilitation game developed in this thesis is shown in Table 3. The expected total cost of our low-cost rehabilitation system is $140, which includes both of the motion tracking devices (Kinect and P5 Glove) and the free open-source Blender software (Table 3). On the other hand, a traditional system composed of Optotrak, CyberGlove, and the World ToolKit software (to program the Optotrak and CyberGlove) is estimated to cost approximately $76,000. Thus, the low-cost virtual rehabilitation game in this thesis offers a powerful advantage over the current rehabilitation systems in terms of cost. An additional benefit to the low-cost virtual rehabilitation game in this thesis is that the game can be played using any basic personal computer with graphics and sound capabilities. There is no need to install any specific software to run the game other than the device drivers for Kinect and P5 Glove when they are using the devices for the first time.

Table 3: Virtual rehabilitation game system cost comparison

System component	**Our low-cost virtual rehabilitation game**	**Other rehabilitation game options**
Arm motion capture camera system	Microsoft Kinect ~$100	Optotrak ~$60,000
Finger motion capture system	P5 Glove ~$40	Cyber Glove ~$10,000
Programming Toolkit	Blender (free software)	World ToolKit ~$6,000

4.2 Usability evaluation of the virtual rehabilitation games

Within the House of Quality assessment, the pre-game survey results showed that patients' highest desire in a virtual rehabilitation game is for the games to be challenging. The three games developed in our laboratory were well received in that regard. In addition, our games in general met patients' expectations as evidenced by above-average post-game response weight of 3.5/5 for all games. These findings are promising enough

to lead the researchers to further improve the low-cost virtual rehabilitation games for clinical adoption.

The usability evaluation/HOQ priority weight analysis of the virtual rehabilitation games also show that there is additional work required in the area of device reliability. Device reliability can consist of quality and performance of hardware and set-up and calibration of hardware. The quality and performance of the hardware are linked with the price; therefore there may not be much room for improvement for this category. However, game software may compensate for the hardware quality by utilizing motion prediction algorithm and filtering (Pastor, Hayes et al. 2012). For instance, jittery movements based on Kinect data (Obdrzalek, Kurillo et al. 2012) could be resolved by filtering the noise data by using Kalman filter (Welch and Bishop 1995). Kalman filter is used as a predictor-estimator model and it estimates the output based on the certainty of prior state. Kalman filter is used for estimating the upper limb segment orientation in real time (Yun and Bachmann 2006). Future versions of these low-cost virtual rehabilitation games may integrate this filtering technique for better game performance and user experience.

As for set-up of hardware, the instruction manual described where to locate Kinect and P5 Glove sensor receptor but not in absolute details. As a result, it was observed that many patients placed the sensors such that they were not standing within the capture range. Therefore, the instruction manual may be improved to better explain the appropriate location of the sensors by including more explanatory figures for the capture range. Also, a few modifications to the games to generate a warning sign to indicate the user movements go out of range would prevent the user, falling out of

capture range. In addition, patient-specific calibration procedures may be added as needed to adapt to patients' movement capability.

4.3 Conclusions

We developed a virtual rehabilitation game with free Blender software and affordable motion tracking devices of Kinect and P5 Glove. This virtual rehabilitation game demonstrated strong potential and feasibility for a low-cost rehabilitation game system for home or clinic use with expected cost of $140 for the hardware and free open source software. The low-cost virtual rehabilitation game have potential to significantly facilitate patients' physical rehabilitation for coordinated arm and finger movements because of their feasibility to be available to a wide population, even in chronic stages after stroke. The low-cost virtual rehabilitation game can also provide clinically relevant information such as the joint range of motion and time to complete tasks, thereby facilitating clinicians' understanding of patients' progress.

Our usability evaluation using a well-known method of House of Quality showed that our low-cost virtual rehabilitation games were liked by our patient population. We were able to identify the top priority improvement need of the game system which was device reliability. Future work should develop better data processing algorithms and instructions in order to improve the device reliability.

This thesis provides evidence that it is possible to develop a low-cost and usable rehabilitation game by using commercially available hardware and free software. Such technical development is expected to be important to motivate and encourage patients to practice movements in various scenarios to result in positive outcomes (Kwakkel, Wagenaar et al. 1997). Upon refining the games per usability evaluation, long-term

clinical studies will be needed to determine the clinical efficacy of low-cost virtual rehabilitation games on patients' physical functions. Furthermore, accessibility for patients and compatibility with conventional rehabilitation programs may be considered to facilitate bench to bed side translation.

References

Billinger, S. A., R. Arena, J. Bernhardt, J. J. Eng, B. A. Franklin, C. M. Johnson, M. MacKay-Lyons, R. F. Macko, G. E. Mead and E. J. Roth (2014). "AHA/ASA Scientific Statement: Physical Activity and Exercise Recommendations for Stroke Survivors: A Statement for Healthcare Professionals From the American Heart Association/American Stroke Association."

Broeks, G., GJ Lankhorst, K. Rumping, AJH Prevo, J (1999). "The long-term outcome of arm function after stroke: results of a follow-up study." Disability & Rehabilitation **21**(8): 357-364.

Burdea, G. (2002). Keynote address: Virtual rehabilitation-benefits and challenges. 1st International Workshop on Virtual Reality Rehabilitation (Mental Health, Neurological, Physical, Vocational) VRMHR, sn.

Burke, J. W., M. McNeill, D. Charles, P. J. Morrow, J. Crosbie and S. McDonough (2009). Serious games for upper limb rehabilitation following stroke. Games and Virtual Worlds for Serious Applications, 2009. VS-GAMES'09. Conference in, IEEE.

Carroll, D. (1965). "A quantitative test of upper extremity function." Journal of chronic diseases **18**(5): 479-491.

Crocher, V., P. Hur and N. J. Seo (2013). Low-cost virtual rehabilitation games: House of quality to meet patient expectations. Virtual Rehabilitation (ICVR), 2013 International Conference on, IEEE.

Crosbie, J., S. Lennon, M. McNeill and S. McDonough (2006). "Virtual reality in the rehabilitation of the upper limb after stroke: the user's perspective." CyberPsychology & Behavior **9**(2): 137-141.

Davison, A. (2007). "The P5 Glove." Pro Java™ 6 3D Game Development: Java 3D™, JOGL, JInput, and JOAL APIs: 349-373.

Duff, M., Y. Chen, S. Attygalle, J. Herman, H. Sundaram, G. Qian, J. He and T. Rikakis (2010). "An adaptive mixed reality training system for stroke rehabilitation." Neural Systems and Rehabilitation Engineering, IEEE Transactions on **18**(5): 531-541.

Duncan, P. W., M. Propst and S. G. Nelson (1983). "Reliability of the Fugl-Meyer assessment of sensorimotor recovery following cerebrovascular accident." Physical therapy **63**(10): 1606-1610.

Fugl-Meyer, A. R., L. Jääskö, I. Leyman, S. Olsson and S. Steglind (1974). "The post-stroke hemiplegic patient. 1. a method for evaluation of physical performance." Scandinavian journal of rehabilitation medicine **7**(1): 13-31.

Glanz, K., A. S. Rizzo and K. Graap (2003). "Virtual reality for psychotherapy: Current reality and future possibilities." Psychotherapy: Theory, Research, Practice, Training **40**(1-2): 55.

Gowland, C., P. Stratford, M. Ward, J. Moreland, W. Torresin, S. Van Hullenaar, J. Sanford, S. Barreca, B. Vanspall and N. Plews (1993). "Measuring physical impairment and disability with the Chedoke-McMaster Stroke Assessment." Stroke **24**(1): 58-63.

Hackett, M. L., J. R. Duncan, C. S. Anderson, J. B. Broad and R. Bonita (2000). "Health-Related Quality of Life Among Long-Term Survivors of Stroke Results From the Auckland Stroke Study, 1991–1992." Stroke **31**(2): 440-447.

Hauser, J. R. (1993). "How Puritan-Bennett used the house of quality." Sloan Management Review **34**(3): 61-70.

Hauser, J. R. and D. Clausing (1988). "The house of quality."
Krakauer, J. W. (2006). "Motor learning: its relevance to stroke recovery and neurorehabilitation." Current opinion in neurology **19**(1): 84-90.

Kwakkel, G., R. C. Wagenaar, T. W. Koelman, G. J. Lankhorst and J. C. Koetsier (1997). "Effects of intensity of rehabilitation after stroke a research synthesis." Stroke **28**(8): 1550-1556.

LaBelle, K. (2011). "Evaluation of Kinect joint tracking for clinical and in-home stroke rehabilitation tools." Undergraduate Thesis, University of Notre Dame.

Lange, B., S. Flynn and A. Rizzo (2009). "Initial usability assessment of off-the-shelf video game consoles for clinical game-based motor rehabilitation." Physical Therapy Reviews **14**(5): 355.

Lange, B., S. Rizzo, C.-Y. Chang, E. A. Suma and M. Bolas (2011). Markerless full body tracking: Depth-sensing technology within virtual environments. The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC), NTSA.

Lloyd-Jones, D., R. J. Adams, T. M. Brown, M. Carnethon, S. Dai, G. De Simone, T. B. Ferguson, E. Ford, K. Furie and C. Gillespie (2010). "Heart disease and stroke statistics—2010 update A report from the American Heart Association." Circulation **121**(7): e46-e215.

Logan, G. D. and D. F. Radcliffe (1997). "Potential for use of a House of Quality matrix technique in rehabilitation engineering [wheelchair customized seating]." Rehabilitation Engineering, IEEE Transactions on **5**(1): 106-115.

Luck, R. (2003). "Dialogue in participatory design." Design Studies **24**(6): 523-535.
Mark, V. W. and E. Taub (2004). "Constraint-induced movement therapy for chronic stroke hemiparesis and other disabilities." Restor Neurol Neurosci **22**(3-5): 317-336.

Miller, I. W., N. B. Epstein, D. S. Bishop and G. I. Keitner (1985). "THE McMASTER FAMILY ASSESSMENT DEVICE: RELIABILITY AND VALIDITY*." Journal of Marital and Family Therapy **11**(4): 345-356.

Morrow, K., C. Docan, G. Burdea and A. Merians (2006). Low-cost virtual rehabilitation of the hand for patients post-stroke. Virtual Rehabilitation, 2006 International Workshop on, IEEE.

O'Sullivan, S. B. and T. J. Schmitz (2007). Physical Rehabilitation, F a Davis Company.
Obdrzalek, S., G. Kurillo, F. Ofli, R. Bajcsy, E. Seto, H. Jimison and M. Pavel (2012). Accuracy and robustness of Kinect pose estimation in the context of coaching of elderly population. Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE, IEEE.

Page, S. J., S. Sisto, P. Levine and R. E. McGrath (2004). "Efficacy of modified constraint-induced movement therapy in chronic stroke: a single-blinded randomized controlled trial." Archives of physical medicine and rehabilitation **85**(1): 14-18.

Pastor, I., H. A. Hayes and S. J. Bamberg (2012). A feasibility study of an upper limb rehabilitation system using kinect and computer games. Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE, IEEE.
Postel, J. (1980). "User datagram protocol." Isi.

Putman, K., L. De Wit, W. Schupp, B. Ilse, P. Berman, L. Connell, E. Dejaeger, A.-M. De Meyer, W. De Weerdt and H. Feys (2006). "Use of time by physiotherapists and occupational therapists in a stroke rehabilitation unit: a comparison between four European rehabilitation centres." Disability & Rehabilitation **28**(22): 1417-1424.

Rizzo, A. A., T. Bowerly, J. G. Buckwalter, M. Schultheis, R. Matheis, C. Shahabi, U. Neumann, L. Kim and M. Sharifzadeh (2002). Virtual environments for the assessment of attention and memory processes: the virtual classroom and office. Proceeding of the 4th International Conference on Disability, Virtual Reality and Associated Technology: University of Reading: Vresprem, Hungary.

Roger, V. L., A. S. Go, D. M. Lloyd-Jones, R. J. Adams, J. D. Berry, T. M. Brown, M. R. Carnethon, S. Dai, G. de Simone and E. S. Ford (2011). "Heart disease and stroke statistics—2011 update a report from the American Heart Association." Circulation **123**(4): e18-e209.

Roy, A. K., Y. Soni and S. Dubey (2013). Enhancing effectiveness of motor rehabilitation using kinect motion sensing technology. Global Humanitarian Technology Conference: South Asia Satellite (GHTC-SAS), 2013 IEEE, IEEE.

Sanford, J., J. Moreland, L. R. Swanson, P. W. Stratford and C. Gowland (1993). "Reliability of the Fugl-Meyer assessment for testing motor performance in patients following stroke." Physical therapy **73**(7): 447-454.

Stanger, C. A., C. Anglin, W. S. Harwin and D. P. Romilly (1994). "Devices for assisting manipulation: a summary of user task priorities." Rehabilitation Engineering, IEEE Transactions on **2**(4): 256-265.

Subramanian, S., L. A. Knaut, C. Beaudoin, B. J. McFadyen, A. G. Feldman and M. F. Levin (2007). "Virtual reality environments for post-stroke arm rehabilitation." Journal of neuroengineering and rehabilitation **4**(1): 20.

Sullivan, L. P. (1986). Quality Function Deployment, Quality Progress:39-50
Sveistrup, H., J. McComas, M. Thornton, S. Marshall, H. Finestone, A. McCormick, K. Babulic and A. Mayhew (2003). "Experimental studies of virtual reality-delivered compared to conventional exercise programs for rehabilitation." CyberPsychology & Behavior **6**(3): 245-249.

Towfighi, A. and J. L. Saver (2011). "Stroke declines from third to fourth leading cause of death in the United States: historical perspective and challenges ahead." Stroke **42**(8): 2351-2355.

Wade, D., R. Langton-Hewer, V. Wood, C. Skilbeck and H. Ismail (1983). "The hemiplegic arm after stroke: measurement and recovery." Journal of Neurology, Neurosurgery & Psychiatry **46**(6): 521-524.

Wang, C., K. S. Phua, K. K. Ang, C. Guan, H. Zhang, R. Lin, K. Sui Geok Chua, B. T. Ang and C. W. K. Kuah (2009). A feasibility study of non-invasive motor-imagery BCI-based robotic rehabilitation for Stroke patients. Neural Engineering, 2009. NER'09. 4th International IEEE/EMBS Conference on, IEEE.

Welch, G. and G. Bishop (1995). An introduction to the Kalman filter.
Yun, X. and E. R. Bachmann (2006). "Design, implementation, and experimental results of a quaternion-based Kalman filter for human body motion tracking." Robotics, IEEE Transactions on **22**(6): 1216-1227.

Zimand, E., P. Anderson, J. Gershon, K. Graap, L. Hodges and B. Rothbaum (2002). "Virtual reality therapy: Innovative treatment for anxiety disorders." Primary Psychiatry **9**(7): 51-54.

Appendix: Abstracts & posters

Abstract 1

LOW-COST VIRTUAL REALITY GAME FOR UPPER LIMB REHABILITATION USING KINECT AND P5 GLOVE

Jayashree Arun Kumar, Pilwon Hur, Binal Motawar and Na Jin Seo

University of Wisconsin, Milwaukee, WI, USA

email: arunkum2@uwm.edu, web: pantherfile.uwm.edu/seon/www/

The paper included in the following pages had submitted for inclusion in the 37th Annual meeting of the American Society of Biomechanics, Omaha, September 2013.

LOW-COST VIRTUAL REALITY GAME FOR UPPER LIMB REHABILITATION USING KINECT AND P5 GLOVE

Jayashree Arunkumar, Pilwon Hur, Binal Motawar and Na Jin Seo

University of Wisconsin, Milwaukee, WI, USA
email: arunkum2@uwm.edu, web: pantherfile.uwm.edu/seon/www/

INTRODUCTION

After stroke, more than 50% of survivors report disability of upper extremity function even after conventional treatments [1]. Continuous intensive rehabilitation therapies in a virtual environment may enhance recovery [2]. However, access to virtual reality rehabilitation systems is currently not optimum due to high costs [3]. The need for low-cost virtual reality rehabilitation systems for use in clinics or at home is clear. The objective of this study was to develop a low-cost virtual reality game to engage patients in interactive kitchen activities for rehabilitation using low-cost commercially available motion tracking devices

METHODS

We have developed a virtual reality kitchen game for upper limb rehabilitation using two low-cost motion tracking devices, Kinect (Microsoft, Redmond, WA, USA) and P5 Glove (Essential Reality, LLC, New York, NY, USA) (Fig. 1). Kinect is used to detect the 3D position of the whole arm. The P5 Glove is used to capture the flexion/extension of individual digits and the 3D position and orientation of the hand.

The 3D virtual kitchen environment was developed using open-source software, Blender (Fig. 2). Users can interact with objects in the kitchen by moving their hand and arm whose motions are detected by the motion tracking devices and mimicked by the virtual hand and arm in the kitchen in real time (Fig. 3). A client program receives the joint position data from the motion tracking devices and transmits it to the server program (game engine in Blender) through a User Datagram Protocol connection. The server program then computes joint angles and maps them to the virtual arm. The virtual arm thus follows the user movement.

Compensatory movements such as reaching by moving the trunk [4] are programmed not to contribute to arm reaching to guide users toward normal reaching motion. To enhance the sense of reality, gravity is simulated in this kitchen such that objects fall to the floor if a user drops or pushes them.

Figure 1: Motion tracking devices used in the game (www.microsoftstore.com, vrealities.com)

Figure 2: 3D kitchen environment developed

Figure 3: Virtual arm reaching to grasp a glass

The game requires a user to perform a variety of functional tasks that involve grasping, moving and releasing of kitchen items, inspired by the clinical test, Fugl-Meyer Assessment. These tasks focus on individual and mass flexion/extension of the digits, grasping objects in different sizes and shapes, forearm pronation/supination, elbow extension, and shoulder abduction. For instance, the user is asked to move different items (plates, glasses, soda cans, soup cans) from the counter to an overhead shelf or from the overhead compartment to the cabinet under the counter to practice reaching and grasping motions.

In another task, the user has to pick up assorted utensils such as spoons, forks and knives one by one from the countertop and place them inside a drawer to practice precision grasps. The user is also asked to put toppings on a pizza or prepare salad using multiple ingredients to practice hand and arm coordination. The ingredients are in various shapes and sizes (e.g., pepperoni slices, salt shaker, dressing bottle, lettuce) to expose the user to various grip postures. In addition, the user is asked to flip an hour glass, dial a rotary phone, and wipe the table with a kitchen towel to practice other functional movements with the arm and hand.

To motivate the user, points are awarded per good performance by the user. If the user picks up a wrong item, places an item in an incorrect location, or performs with undesirable hand/arm postures, partial credits will be given. These points are stored to track and display to the user their own progress over repeated game plays.

RESULTS AND DISCUSSIONS

We have developed a virtual kitchen rehabilitation game containing a dynamic virtual arm. The virtual arm follows the user movement in real time well enough for playing the game. However, the need to improve accuracy and remove occasional jittering for the virtual arm movement exists for aesthetics and sense of reality, possibly by enhancing filtering and motion description algorithm for the data obtained from the low-cost devices. Additional future work includes user–specific calibration to accommodate patients with different functional capacities or ranges of motion and refinement of instructions for users.

This kitchen game with Kinect and P5 Glove demonstrates strong potential and feasibility for low-cost rehabilitation game systems for home or clinic use. The expected cost for this system is $310 for the hardware, as can be seen in Table 1. The game setup and content is designed to provide entertainment along with physical activity needed for rehabilitation.

CONCLUSIONS

We have developed a low-cost virtual game for patients to perform kitchen activities as a game with the virtual arm and hand. This kitchen game has the potential to complement occupational therapy and improve sensorimotor function of impaired hand and arms. Upon further technical and instructional improvements of the game, evaluation of the game for patients with hand/arm impairments will be performed to determine clinical efficacy and define future improvement needs.

REFERENCES

1. Peter RW, et al. *Stroke*, **28**, 507-512, 1997.
2. Henderson A, et al. *Topics in Stroke Rehabilitation*, **14**, 52-61, 2007.
3. Deutsch JE, et al. *Physical Therapy. J*, **88,** 1196-1207, 2008.
4. Cirstea MC, et al. *Brain*, **123**, 940-953, 2000.

Table 1: System cost analysis

System component	The current system	Other options
Arm motion detection system	Microsoft Kinect ~ $250	Optotrack ~$60,000
Finger motion detection	P5 Glove ~ $60	CyberGlove ~$10,000
Programming Toolkit	Blender (free software)	WorldToolKit ~$6,000

Abstract 2

USABILITY EVALUATION OF A LOW-COST VIRTUAL REALITY REHABILITATION GAME FOR STROKE PATIENTS WITH UPPER LIMB IMPAIRMENT USING KINECT AND P5 GLOVE

Jayashree Arun Kumar, Pilwon Hur, Kishor Lakshminarayanan and Na Jin Seo

University of Wisconsin, Milwaukee, WI, USA

email: arunkum2@uwm.edu, web: pantherfile.uwm.edu/seon/www/

The paper included in the following pages had been submitted for inclusion in the 38th Annual meeting of the World Congress of Biomechanics, Massachusetts, July 2014.

Usability evaluation of a low-cost virtual reality rehabilitation game for stroke patients with upper limb impairment using Kinect and P5 Glove

Jayashree Arunkumar, Pilwon Hur, Kishor Lakshminarayanan, Na Jin Seo
University of Wisconsin-Milwaukee

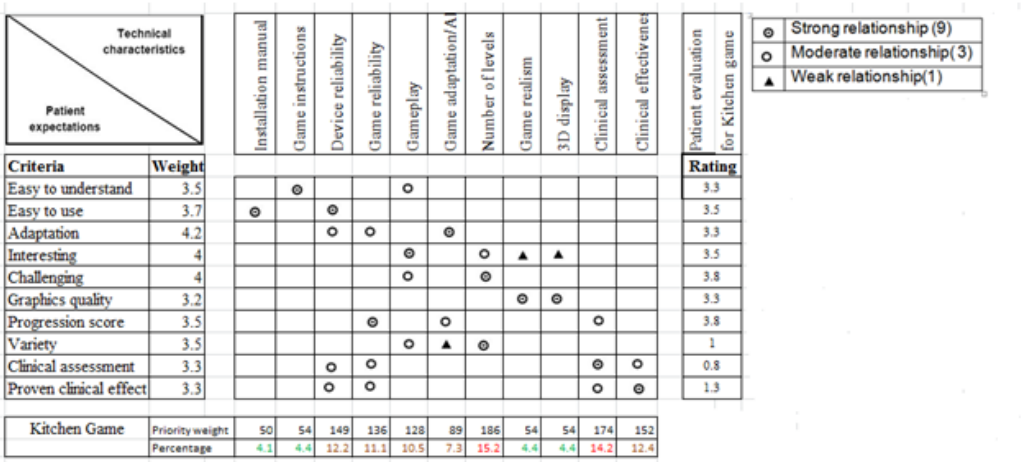
Recent emergence of low-cost motion tracking systems such as Kinect and P5 Glove enables development of low-cost virtual reality (VR) games. Low-cost VR games have tremendous potential to assist with physical rehabilitation for patients with paralysis by motivating patients to be engaged in repetitive movements for an extended period of time. However, wide usage of such rehabilitation games would not occur unless the games are patient-friendly and liked by patients. The objective of this study was to systematically evaluate usability of a low-cost VR kitchen game utilizing Kinect and P5 Glove using the House of Quality technique [1].

The House of Quality matrix was constructed based on patients' general expectation on VR rehabilitation games (perceived importance of each criterion, left column of Fig.1) and their evaluation of the kitchen game (rating per criterion, right column of Fig.1) which were obtained from six stroke patients on a Likert scale using questionnaires before and after they experienced the kitchen game on their own, respectively. As an outcome, priority weights were computed based on these weights and ratings as well as the interrelationship matrix (shown in the middle of Fig.1).

The results indicated that the top priority improvement needs are to introduce various difficulty levels in the game and to add clinical assessments (red priority weights on the bottom row of Fig.1). Based on the observation during the gaming experience and focus group discussion that took place after the questionnaires, it was found that patients would like the game to accommodate individuals with different motor functions by introducing multiple challenging levels in the game. In addition, the patients would like the game to periodically assess their motor function based on known clinical assessments to track their improvement over time.

In conclusion, introduction of difficulty levels per patient's current motor functions and addition of clinical assessment to track clinical benefits from using this game are expected to substantially increase usability of this VR kitchen game. This study demonstrates a feasibility of the House of Quality technique to

systematically evaluate usability of a VR rehabilitation game and identify the top priority improvement needs.



Technical characteristics / Patient expectations	Weight	Installation manual	Game instructions	Device reliability	Game reliability	Gameplay	Game adaptation/AI	Number of levels	Game realism	3D display	Clinical assessment	Clinical effectiveness	Patient evaluation for Kitchen game
Criteria	**Weight**												**Rating**
Easy to understand	3.5		◎			○							3.3
Easy to use	3.7	◎		◎									3.5
Adaptation	4.2			○	○		◎						3.3
Interesting	4					◎		○	▲	▲			3.5
Challenging	4					○		◎					3.8
Graphics quality	3.2								◎	◎			3.3
Progression score	3.5				◎		○				○		3.8
Variety	3.5					○	▲	◎					1
Clinical assessment	3.3			○	○						◎	○	0.8
Proven clinical effect	3.3			○	○						○	◎	1.3
Kitchen Game	Priority weight	50	54	149	136	128	89	186	54	54	174	152	
	Percentage	4.1	4.4	12.2	11.1	10.5	7.3	15.2	4.4	4.4	14.2	12.4	

◎ Strong relationship (9)
○ Moderate relationship (3)
▲ Weak relationship (1)

Figure 1: The House of Quality matrix for a low-cost VR Kitchen game identified top priority technical improvement needs as an outcome (bottom row), based on patients' expectation ratings (left column), the game's technical characteristics (top row), interrelationship matrix (center), and patients' evaluation of the game (right column). Red, brown, and green numbers indicate the highest (>14%), intermediate, and the lowest (<5%) technical improvement priorities for the game, respectively.

Reference

Crocher V, Hur P, Seo NJ. "Low-cost virtual rehabilitation games: House of Quality to meet patient expectations", *International Conference on Virtual Rehabilitation*, 2013, Philadelphia, PA.

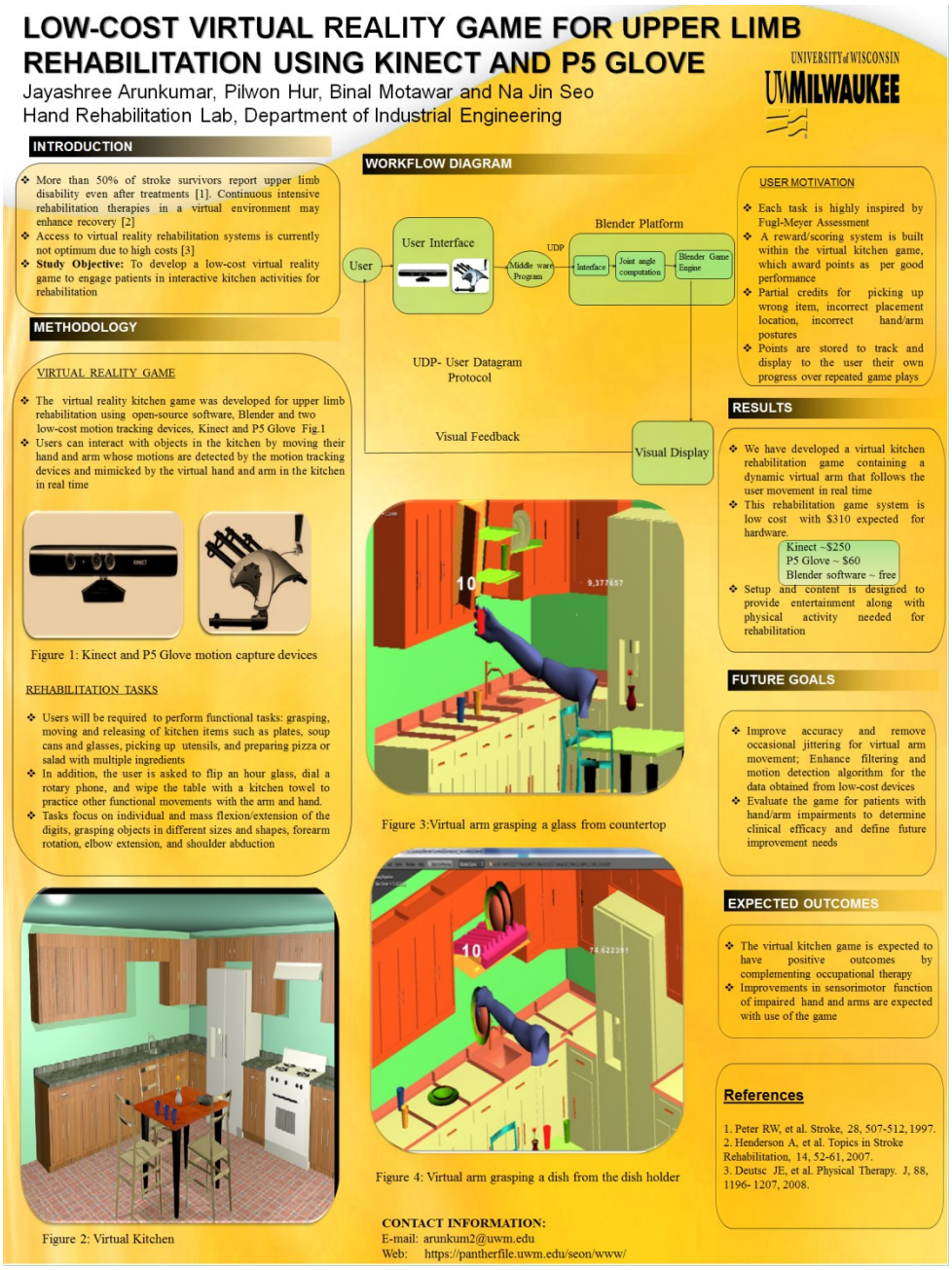


Figure 25: Poster presentation - College of Engineering and Applied science (CEAS 2013)

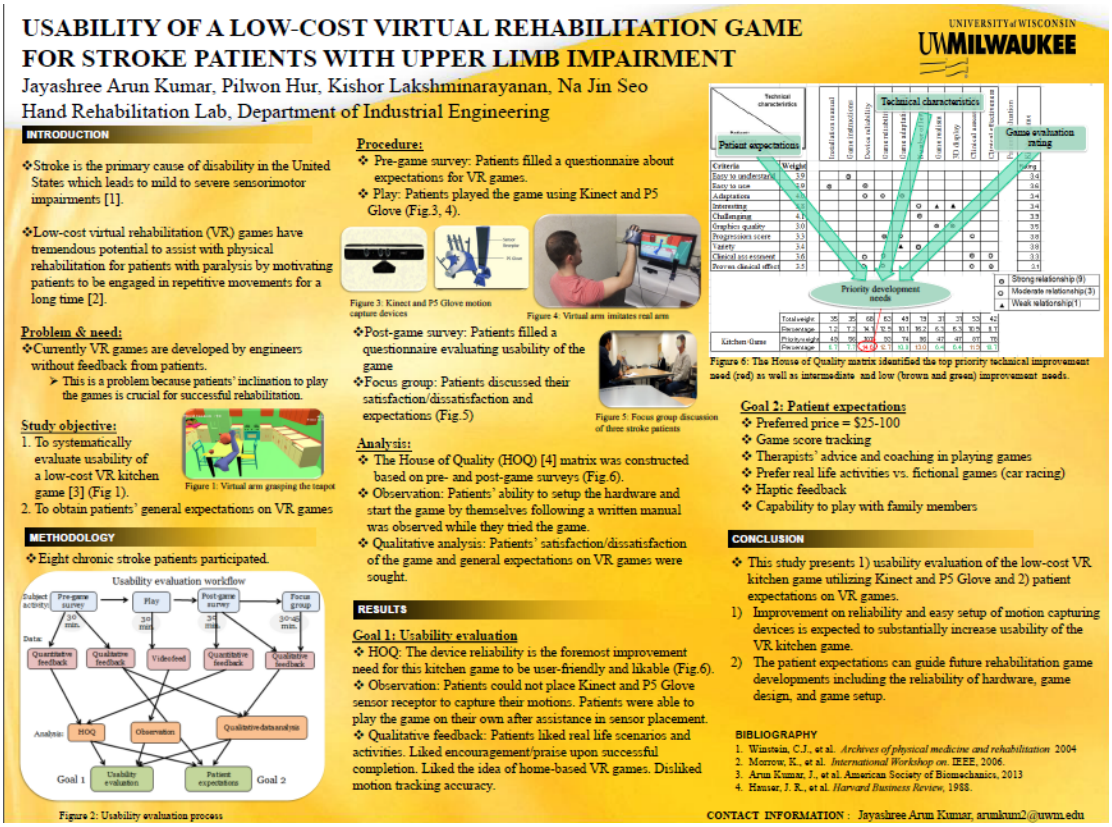


Figure 26: Poster presentation - College of Engineering and Applied science (CEAS 2014)

Appendix: Software source code

Python software source code

Joint angle computation

```
import GameLogic
import bge
import mathutils
import math
from math import radians
from math import sin
import time

try:
   bge.logic.globalDict['start_time']
except KeyError:
   bge.logic.globalDict['start_time']=10000
   bge.logic.globalDict['elapsed_time']=0
   bge.logic.globalDict['Text']=0
   bge.logic.globalDict['Text']=bge.logic.globalDict['Text']+ 0.0165
   print("**********not defined**********")


bge.logic.globalDict['start_time'] = timer()
print("time.time and time_clock()", time.time(),time.clock())

#start_time=bge.logic.globalDict['start_time']
print("Start time",bge.logic.globalDict['start_time'])

distance=0.15
distance1=0.25
try:
   bge.logic.globalDict['theta_min']
except KeyError:
   bge.logic.globalDict['theta_min']=10000
   bge.logic.globalDict['theta_max']=-10000
   bge.logic.globalDict['theta_prev']=0
   print("**********not defined**********")

theta_prev=bge.logic.globalDict['theta_prev']
theta_max=bge.logic.globalDict['theta_max']
theta_min=bge.logic.globalDict['theta_min']
```

```
try:
   bge.logic.globalDict['alpha_min']
except KeyError:
   bge.logic.globalDict['alpha_min']=10000
   bge.logic.globalDict['alpha_max']=-10000
   bge.logic.globalDict['alpha_prev']=0
   print("**********not defined**********")

alpha_prev=bge.logic.globalDict['alpha_prev']
alpha_max=bge.logic.globalDict['alpha_max']
alpha_min=bge.logic.globalDict['alpha_min']


try:
   bge.logic.globalDict['beta_min']
except KeyError:
   bge.logic.globalDict['beta_min']=10000
   bge.logic.globalDict['beta_max']=-10000
   bge.logic.globalDict['beta_prev']=0
   print("**********not defined**********")

beta_prev=bge.logic.globalDict['beta_prev']
beta_max=bge.logic.globalDict['beta_max']
beta_min=bge.logic.globalDict['beta_min']

try:
   bge.logic.globalDict['gamma_min']
except KeyError:
   bge.logic.globalDict['gamma_min']=10000
   bge.logic.globalDict['gamma_max']=-10000
   bge.logic.globalDict['gamma_prev']=0
   print("**********not defined**********")

gamma_prev=bge.logic.globalDict['gamma_prev']
gamma_max=bge.logic.globalDict['gamma_max']
gamma_min=bge.logic.globalDict['gamma_min']

cont = GameLogic.getCurrentController()
objects= GameLogic.getCurrentScene().objects
owner = cont.owner


x=1.5
y=1
z=2.0
armature = cont.owner
```

```
bone1 = armature.channels["Bone.forarm"]
bl=bone1.joint_rotation

bone2 = armature.channels["Bone.bicept"]
Bone = armature.channels["Bone"]
bone3 = armature.channels["Bone.003"]
Bone_012 = armature.channels["Bone.012"]
Bone_009 = armature.channels["Bone.009"]
Bone_014 = armature.channels["Bone.014"]
Bone_001 = armature.channels["Bone.001"]
Bone_010= armature.channels["Bone.010"]
Bone_013 = armature.channels["Bone.013"]
Bone_015 = armature.channels["Bone.015"]
bone4 = armature.channels["Bone.004"]
bone5 = armature.channels["Bone.005"]
Bone_002 = armature.channels["Bone.002"]
Bone_011 = armature.channels["Bone.011"]
bone5 = armature.channels["Bone.014"]
Bone_016 = armature.channels["Bone.016"]

bone1.rotation_mode = 5
bone2.rotation_mode = 5
bone3.rotation_mode = 5
bone4.rotation_mode = 5
bone5.rotation_mode = 5
Bone .rotation_mode = 5
Bone_001.rotation_mode = 5
Bone_002.rotation_mode = 5
Bone_015.rotation_mode = 5
Bone_016.rotation_mode = 5
Bone_009.rotation_mode = 5
Bone_010.rotation_mode = 5
Bone_011.rotation_mode = 5
Bone_012.rotation_mode = 5
Bone_013.rotation_mode = 5
Bone_014.rotation_mode = 5

euler = bone1.rotation_euler
euler = bone2.rotation_euler
euler = bone3.rotation_euler
euler = bone4.rotation_euler
euler = bone5.rotation_euler
euler = Bone.rotation_euler
euler = Bone_001.rotation_euler
euler = Bone_002.rotation_euler
```

```
euler = Bone_015.rotation_euler
euler = Bone_016.rotation_euler
euler = Bone_009.rotation_euler
euler = Bone_010.rotation_euler
euler = Bone_011.rotation_euler
euler = Bone_012.rotation_euler
euler = Bone_013.rotation_euler
euler = Bone_014.rotation_euler

x = euler.z + 0.01


tempstr1=bge.logic.globalDict['Data']
tempstr2=tempstr1.replace("b","")
tempstr=tempstr2.replace("'","")
bge.logic.globalDict['tempdata']= tempstr.split()

#Joint angles computation

if len(bge.logic.globalDict['tempdata'])>9:
   Shoulder = []
   for i in range(3):
      Shoulder.append(bge.logic.globalDict['tempdata'][i+1])
      print(Shoulder)
   Elbow = []
   for i in range(3):
      Elbow.append(bge.logic.globalDict['tempdata'][i+4])

   Wrist = []
   for i in range(3):
      Wrist.append(bge.logic.globalDict['tempdata'][i+7])

#Elbow angle

   ES = []
   EW = []# used for elbow rotation
   WE = []

   for i in range(3):
      ES.append(float(Shoulder[i])-float(Elbow[i]))
      EW.append(float(Wrist[i])-float(Elbow[i]))
      WE.append(float(Elbow[i])-float(Wrist[i]))


   ESNorm=math.sqrt(ES[0]*ES[0]+ES[1]*ES[1]+ES[2]*ES[2])
   EWNorm=math.sqrt(EW[0]*EW[0]+EW[1]*EW[1]+EW[2]*EW[2])
```

```
    WENorm=math.sqrt(WE[0]*WE[0]+WE[1]*WE[1]+WE[2]*WE[2])
    for i in range(3):
        ES[i]=ES[i]/ESNorm
        EW[i]=EW[i]/EWNorm
        WE[i]=WE[i]/WENorm

    dotprod=ES[0]*EW[0]+ES[1]*EW[1]+ES[2]*EW[2]
    el=bone1.channel_matrix
    theta = math.acos(dotprod)
    print("Elbow angle",theta)
    if(theta-theta_prev)>math.pi:
        theta=(theta-(2*math.pi))
    elif(theta-theta_prev)<-math.pi:
        theta=(theta+(2*math.pi))

    if theta<theta_min:
        theta_min=theta
    if theta>theta_max:
        theta_max=theta
    theta_prev=theta

    print("theta_max",theta_max)
    print("theta_min",theta_min)

    bge.logic.globalDict['theta_max']=theta_max
    bge.logic.globalDict['theta_min']=theta_min

#Shoulder angles

#Alpha angle

    ES = []
    X=[1,0,0]

    for i in range(3):
        ES.append(float(Elbow[i])-float(Shoulder[i]))
    print(ES)
    ESNorm=math.sqrt(ES[0]*ES[0]+ES[1]*ES[1]+ES[2]*ES[2])
    for i in range(3):
        ES[i]=ES[i]/ESNorm

    #dot_alpha= X[0]*ES[0]+X[1]*ES[1]+X[2]*ES[2]
    alpha= math.atan2(-ES[2],ES[0])
    print("Alpha angle",alpha)

    if(alpha-alpha_prev)>math.pi:
```

```
        alpha=(alpha-(2*math.pi))
    elif(alpha-alpha_prev)<-math.pi:
        alpha=(alpha+(2*math.pi))

    if alpha<alpha_min:
        alpha_min=alpha
    if alpha>alpha_max:
        alpha_max=alpha
    alpha_prev=alpha

    print("alpha_max",alpha_max)
    print("alpha_min",alpha_min)

    bge.logic.globalDict['alpha_max']=alpha_max
    bge.logic.globalDict['alpha_min']=alpha_min
#Beta angle

    beta= math.asin(ES[1])
    print("Beta angle ",beta)

    if(beta-beta_prev)>math.pi:
        beta=(beta-(2*math.pi))
    elif(beta-beta_prev)<-math.pi:
        beta=(beta+(2*math.pi))

    if beta<beta_min:
        beta_min=beta
    if beta>beta_max:
        beta_max=beta
    beta_prev=beta

    print("beta_max",beta_max)
    print("beta_min",beta_min)

    bge.logic.globalDict['beta_max']=beta_max
    bge.logic.globalDict['beta_min']=beta_min

#Gamma Angle

    ES = []
    WE = []
    #print(dir(ES))
    for i in range(3):
        ES.append(float(Shoulder[i])-float(Elbow[i]))
        WE.append(float(Elbow[i])-float(Wrist[i]))
```

```
    ESNorm=math.sqrt(ES[0]*ES[0]+ES[1]*ES[1]+ES[2]*ES[2])
    WENorm=math.sqrt(WE[0]*WE[0]+WE[1]*WE[1]+WE[2]*WE[2])
    for i in range(3):
        ES[i]=ES[i]/ESNorm
        WE[i]=WE[i]/WENorm

    import mathutils
    vec_a = mathutils.Vector((WE[0],WE[1],WE[2]))
    vec_b = mathutils.Vector((ES[0],ES[1],ES[2]))
    cross_prod = vec_a.cross(vec_b)
    print("Cross Product of WE Vector and ES Vector ",cross_prod)
    cross_prod_Norm =
math.sqrt(cross_prod[0]*cross_prod[0]+cross_prod[1]*cross_prod[1]+cross_prod[2]*cro
ss_prod[2])
    for i in range(3):
        cross_prod[i]= cross_prod[i]/cross_prod_Norm

    XZ =[cross_prod[0],0,cross_prod[2]]


    from math import sin
    #Y=cross_prod[1]
    #print("Y value is",Y)
    #print("Opposite side of the sin",A)
    #Hyp=cross_prod[0]+cross_prod[1]+cross_prod[2]
    gam= cross_prod[0]*-(math.sin(alpha))+cross_prod[1]*0+cross_prod[2]*-
(math.cos(alpha))
    gamma=math.asin(gam)
    print("Gamma angle",gamma)

    if(gamma-gamma_prev)>math.pi:
        gamma=(gamma-(2*math.pi))
    elif(gamma-gamma_prev)<-math.pi:
        gamma=(gamma+(2*math.pi))

    if gamma<gamma_min:
        gamma_min=gamma
    if gamma>gamma_max:
        gamma_max=gamma
    gamma_prev=gamma

    print("gamma_max",gamma_max)
    print("gamma_min",gamma_min)

    bge.logic.globalDict['gamma_max']=gamma_max
    bge.logic.globalDict['gamma_min']=gamma_min
```

```
bone1.rotation_euler = [0,-gamma, math.pi-theta]
bone3.rotation_euler=[0,0,-float(bge.logic.globalDict['tempdata'][0])/40.0]
bone4.rotation_euler=[0,0,float(bge.logic.globalDict['tempdata'][0])/40.0]
bone5.rotation_euler=[0,float(bge.logic.globalDict['tempdata'][0])/40.0,0]

Bone.rotation_euler=[0,0,-float(bge.logic.globalDict['tempdata'][0])/40.0]
Bone_001.rotation_euler=[0,0,float(bge.logic.globalDict['tempdata'][0])/40.0]
Bone_002.rotation_euler=[0,-float(bge.logic.globalDict['tempdata'][0])/40.0,0]
Bone_015.rotation_euler=[0,0,-float(bge.logic.globalDict['tempdata'][0])/40.0]
Bone_016.rotation_euler=[0,0,float(bge.logic.globalDict['tempdata'][0])/40.0]

Bone_009.rotation_euler=[0,0,-float(bge.logic.globalDict['tempdata'][0])/40.0]
Bone_010.rotation_euler=[0,0,float(bge.logic.globalDict['tempdata'][0])/40.0]
Bone_011.rotation_euler=[0,-float(bge.logic.globalDict['tempdata'][0])/40.0,0]
Bone_012.rotation_euler=[0,0,-float(bge.logic.globalDict['tempdata'][0])/40.0]
Bone_013.rotation_euler=[0,0,float(bge.logic.globalDict['tempdata'][0])/40.0]
Bone_014.rotation_euler=[0,-float(bge.logic.globalDict['tempdata'][0])/40.0,0]

bone2.rotation_euler = [alpha-math.pi/2,(gamma)*0.1,beta]
#fingers = Bone_002.rotation_euler

armature.update()
bge.logic.globalDict['theta_ROM'] = (bge.logic.globalDict['theta_max']-
bge.logic.globalDict['theta_min'])
print("ROM_theta",bge.logic.globalDict['theta_ROM'])
bge.logic.globalDict['alpha_ROM'] = (bge.logic.globalDict['alpha_max']-
bge.logic.globalDict['alpha_min'])
print("ROM_alpha",bge.logic.globalDict['alpha_ROM'])
bge.logic.globalDict['beta_ROM'] = (bge.logic.globalDict['beta_max']-
bge.logic.globalDict['beta_min'])
print("ROM_beta",bge.logic.globalDict['beta_ROM'])
bge.logic.globalDict['gamma_ROM'] = (bge.logic.globalDict['gamma_max']-
bge.logic.globalDict['gamma_min'])
print("ROM_gamma",bge.logic.globalDict['gamma_ROM'])

bge.logic.globalDict['elapsed_time']= (timer() - bge.logic.globalDict['start_time'])

bge.logic.globalDict['elapsed_time'] = bge.logic.globalDict['elapsed_time'] * 1000

print ("elapsed_time",bge.logic.globalDict['elapsed_time'])
print("Global_dict.time",bge.logic.globalDict['Text'])
```

Server program

```
import bge, socket, GameLogic
```

```
import sys
import time

def main():

#    print('AAAA')
    cont = bge.logic.getCurrentController()
    own = cont.owner
#    print('BBB')

    if not 'init' in own:
        own['init']=1
        own['UDP_IP'] = "127.0.0.1"
        own['UDP_PORT'] = 8778

        address =(own['UDP_IP'],own['UDP_PORT'])
        print('Begin.')
        GameLogic.globalDict['sock'] = socket.socket( socket.AF_INET,
socket.SOCK_DGRAM )
        GameLogic.globalDict['sock'].setsockopt(socket.SOL_SOCKET,
socket.SO_REUSEADDR,3)
        ##GameLogic.globalDict['sock'].serve_forever()
        GameLogic.globalDict['sock'].bind(address)
        ##GameLogic.globalDict['sock'].listen(200)
        GameLogic.globalDict['sock'].setblocking(0)
        GameLogic.globalDict["EncoderReceiverState"] = 1
        bge.logic.globalDict['Data']='1'
        print('Socket created')

try:
        data,addr = GameLogic.globalDict['sock'].recvfrom(256)
        data = data.decode("utf-8")
        #print(data)
        bge.logic.globalDict['Data']=data

#        print(GameLogic.globalDict['Data'])

#        data = GameLogic.globalDict['Data'][0]
        #addr = GameLogic.globalDict['Data'][1]
#        new = data.split()
#        GameLogic.globalDict['new_array']= new
#        for i in new:print(i)
#        print(GameLogic.globalDict['new_array'][0])

        #print(new)
```

```
except:
   pass
#   if not data:
#       break

#       reply = 'OK...' + data

#       GameLogic.globalDict['sock'].sendto(reply ,addr)
#       print 'Message[' + addr[0] + ':' + str(addr[1]) + '] - ' + data.strip()

 #      GameLogic.globalDict['sock'].close()
main()
```

C# software source code

Client program

```
//------------------------------------------------------------------------------
// <copyright file="MainWindow.xaml.cs" company="Microsoft">
//     Copyright (c) Microsoft Corporation.  All rights reserved.
// </copyright>
//------------------------------------------------------------------------------

namespace Microsoft.Samples.Kinect.SkeletonBasics
{
   using System;
   using System.IO;
   using System.Windows;
   using System.Windows.Media;
   using Microsoft.Kinect;
   using System.Net;
   using System.Net.Sockets;
   using System.Text;
   using System.Threading;
   using Zion.Input;

   /// <summary>
   /// Interaction logic for MainWindow.xaml
   /// </summary>
   ///
   public partial class MainWindow : Window
   {
      P5State p5state = new P5State();
      P5Dll p5 = new P5Dll();
      /// <summary>
```

```
        /// Width of output drawing
        /// </summary>
        private const float RenderWidth = 640.0f;

        /// <summary>
        /// Height of our output drawing
        /// </summary>
        private const float RenderHeight = 480.0f;

        /// <summary>
        /// Thickness of drawn joint lines
        /// </summary>
        private const double JointThickness = 3;

        /// <summary>
        /// Thickness of body center ellipse
        /// </summary>
        private const double BodyCenterThickness = 10;

        /// <summary>
        /// Thickness of clip edge rectangles
        /// </summary>
        private const double ClipBoundsThickness = 10;

        /// <summary>
        /// Brush used to draw skeleton center point
        /// </summary>
        private readonly Brush centerPointBrush = Brushes.Blue;

        /// <summary>
        /// Brush used for drawing joints that are currently tracked
        /// </summary>
        private readonly Brush trackedJointBrush = new
SolidColorBrush(Color.FromArgb(255, 68, 192, 68));

        /// <summary>
        /// Brush used for drawing joints that are currently inferred
        /// </summary>
        private readonly Brush inferredJointBrush = Brushes.Yellow;

        /// <summary>
        /// Pen used for drawing bones that are currently tracked
        /// </summary>
        private readonly Pen trackedBonePen = new Pen(Brushes.Green, 6);

        /// <summary>
```

```
        /// Pen used for drawing bones that are currently inferred
        /// </summary>
        private readonly Pen inferredBonePen = new Pen(Brushes.Gray, 1);

        /// <summary>
        /// Active Kinect sensor
        /// </summary>
        private KinectSensor sensor;

        /// <summary>
        /// Drawing group for skeleton rendering output
        /// </summary>
        private DrawingGroup drawingGroup;

        /// <summary>
        /// Drawing image that we will display
        /// </summary>
        private DrawingImage imageSource;

        /// <summary>
        /// Initializes a new instance of the MainWindow class.
        /// </summary>
        public MainWindow()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Draws indicators to show which edges are clipping skeleton data
        /// </summary>
        /// <param name="skeleton">skeleton to draw clipping information for</param>
        /// <param name="drawingContext">drawing context to draw to</param>
        private static void RenderClippedEdges(Skeleton skeleton, DrawingContext
drawingContext)
        {
            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Bottom))
            {
                drawingContext.DrawRectangle(
                    Brushes.Red,
                    null,
                    new Rect(0, RenderHeight - ClipBoundsThickness, RenderWidth,
ClipBoundsThickness));
            }

            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Top))
            {
```

```
            drawingContext.DrawRectangle(
                Brushes.Red,
                null,
                new Rect(0, 0, RenderWidth, ClipBoundsThickness));
        }

        if (skeleton.ClippedEdges.HasFlag(FrameEdges.Left))
        {
            drawingContext.DrawRectangle(
                Brushes.Red,
                null,
                new Rect(0, 0, ClipBoundsThickness, RenderHeight));
        }

        if (skeleton.ClippedEdges.HasFlag(FrameEdges.Right))
        {
            drawingContext.DrawRectangle(
                Brushes.Red,
                null,
                new Rect(RenderWidth - ClipBoundsThickness, 0, ClipBoundsThickness,
RenderHeight));
        }
    }


    /// <summary>
    /// Execute startup tasks
    /// </summary>
    /// <param name="sender">object sending the event</param>
    /// <param name="e">event arguments</param>
    private void WindowLoaded(object sender, RoutedEventArgs e)
    {
        // Create the drawing group we'll use for drawing
        this.drawingGroup = new DrawingGroup();

        // Create an image source that we can use in our image control
        this.imageSource = new DrawingImage(this.drawingGroup);

        // Display the drawing using our image control
        Image.Source = this.imageSource;

        // Look through all sensors and start the first connected one.
        // This requires that a Kinect is connected at the time of app startup.
        // To make your app robust against plug/unplug,
        // it is recommended to use KinectSensorChooser provided in
Microsoft.Kinect.Toolkit
```

```
    foreach (var potentialSensor in KinectSensor.KinectSensors)
    {
        if (potentialSensor.Status == KinectStatus.Connected)
        {
            this.sensor = potentialSensor;
            break;
        }
    }

    if (null != this.sensor)
    {
        // Turn on the skeleton stream to receive skeleton frames
        this.sensor.SkeletonStream.Enable();

        // Add an event handler to be called whenever there is new color frame data
        this.sensor.SkeletonFrameReady += this.SensorSkeletonFrameReady;

        // Start the sensor!
        try
        {
            this.sensor.Start();
        }
        catch (IOException)
        {
            this.sensor = null;
        }
    }

    if (null == this.sensor)
    {
        this.statusBarText.Text = Properties.Resources.NoKinectReady;
    }


    if (p5.Connect())
    {
        p5.SetMouseState(false);
    }

    //while (!messageReceived)
    //{
    //    Thread.Sleep(100);
    //}
}

/// <summary>
```

```
        /// Execute shutdown tasks
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
        private void WindowClosing(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            try
            {
                p5.Close();
                p5 = null;
            }
            catch
            { }

            if (null != this.sensor)
            {
                this.sensor.Stop();
            }

        }

        public bool udpDefined = false;
        Socket udpSocket2;
        EndPoint local2EP;
        EndPoint remote2EP;
        byte[] sendBuffer;

        /// <summary>
        /// Event handler for Kinect sensor's SkeletonFrameReady event
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
        private void SensorSkeletonFrameReady(object sender,
SkeletonFrameReadyEventArgs e)
        {
            Skeleton[] skeletons = new Skeleton[0];

            using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
            {
                if (skeletonFrame != null)
                {
                    skeletons = new Skeleton[skeletonFrame.SkeletonArrayLength];
                    skeletonFrame.CopySkeletonDataTo(skeletons);
                }
            }
```

```
        using (DrawingContext dc = this.drawingGroup.Open())
        {
            // Draw a transparent background to set the render size
            dc.DrawRectangle(Brushes.Black, null, new Rect(0.0, 0.0, RenderWidth,
RenderHeight));

            if (skeletons.Length != 0)
            {
                Skeleton skel = skeletons[0];

                double hipZ=0.0;
                double hipZ_prev=10000.0;
                foreach (Skeleton skel1 in skeletons)
                {
                    if (skel1.TrackingState == SkeletonTrackingState.Tracked)
                    {
                        hipZ = skel1.Joints[JointType.HipCenter].Position.Z;
                        if (hipZ < hipZ_prev)
                        {
                            hipZ_prev = hipZ;
                            skel = skel1;
                        }
                    }
                }


                //Skeleton skel = skeletons[0];
                //SkeletonTrackingState.Tracked
                //skeletons.GetLength
                //foreach (Skeleton skel in skeletons)


                //{
                    RenderClippedEdges(skel, dc);

                    if (skel.TrackingState == SkeletonTrackingState.Tracked)
                    {
                        this.DrawBonesAndJoints(skel, dc);
                    }
                    else if (skel.TrackingState == SkeletonTrackingState.PositionOnly)
                    {
                        dc.DrawEllipse(
                        this.centerPointBrush,
                        null,
                        this.SkeletonPointToScreen(skel.Position),
```

```
                    BodyCenterThickness,
                    BodyCenterThickness);
                }
              //}
            }

            // prevent drawing outside of our render area
            this.drawingGroup.ClipGeometry = new RectangleGeometry(new Rect(0.0,
0.0, RenderWidth, RenderHeight));
        }
    }

    /// <summary>
    /// Draws a skeleton's bones and joints
    /// </summary>
    /// <param name="skeleton">skeleton to draw</param>
    /// <param name="drawingContext">drawing context to draw to</param>
    private void DrawBonesAndJoints(Skeleton skeleton, DrawingContext
drawingContext)
    {
        if (!udpDefined)
        {
            udpDefined = true;
            udpSocket2 = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
ProtocolType.Udp);
            local2EP = new IPEndPoint(IPAddress.Any, 0);
            remote2EP = new IPEndPoint(IPAddress.Loopback, 8778);
            udpSocket2.ExclusiveAddressUse = false;
            udpSocket2.SetSocketOption(SocketOptionLevel.Socket,
SocketOptionName.ReuseAddress, true);
            udpSocket2.Bind(local2EP);
            udpSocket2.Blocking = false;
        }

        // Update p5 state
        //P5Dll.P5_GetState(0, 0, ref p5state);
        p5state=p5.GetState();

        float temp_sum1 = 0.0f;
        for (int i = 0; i < 5; i++)
        {
            temp_sum1 += (float)p5state.Finger[i]; // get finger bending info
        }
        int temp_sum = (int)Math.Round(temp_sum1 / 5.0);

        // Render Torso
```

```
            this.DrawBone(skeleton, drawingContext, JointType.Head,
JointType.ShoulderCenter);
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.ShoulderLeft);
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.ShoulderRight);
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.Spine);
            this.DrawBone(skeleton, drawingContext, JointType.Spine,
JointType.HipCenter);
            this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipLeft);
            this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipRight);

            // Left Arm
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderLeft,
JointType.ElbowLeft);
            this.DrawBone(skeleton, drawingContext, JointType.ElbowLeft,
JointType.WristLeft);
            this.DrawBone(skeleton, drawingContext, JointType.WristLeft,
JointType.HandLeft);

            // Right Arm
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderRight,
JointType.ElbowRight);
            this.DrawBone(skeleton, drawingContext, JointType.ElbowRight,
JointType.WristRight);
            this.DrawBone(skeleton, drawingContext, JointType.WristRight,
JointType.HandRight);

            // Left Leg
            this.DrawBone(skeleton, drawingContext, JointType.HipLeft,
JointType.KneeLeft);
            this.DrawBone(skeleton, drawingContext, JointType.KneeLeft,
JointType.AnkleLeft);
            this.DrawBone(skeleton, drawingContext, JointType.AnkleLeft,
JointType.FootLeft);

            // Right Leg
            this.DrawBone(skeleton, drawingContext, JointType.HipRight,
JointType.KneeRight);
            this.DrawBone(skeleton, drawingContext, JointType.KneeRight,
JointType.AnkleRight);
            this.DrawBone(skeleton, drawingContext, JointType.AnkleRight,
JointType.FootRight);
```

```
            Joint joint2 = skeleton.Joints[JointType.ShoulderRight];
            Joint joint8 = skeleton.Joints[JointType.ElbowRight];
            Joint joint10 = skeleton.Joints[JointType.WristRight];

            string udpString = temp_sum.ToString() + " " + joint2.Position.X.ToString("F4")
+ " " + joint2.Position.Y.ToString("F4") + " " + joint2.Position.Z.ToString("F4") + " ";
            udpString += joint8.Position.X.ToString("F4") + " " +
joint8.Position.Y.ToString("F4") + " " + joint8.Position.Z.ToString("F4") + " ";
            udpString += joint10.Position.X.ToString("F4") + " " +
joint10.Position.Y.ToString("F4") + " " + joint10.Position.Z.ToString("F4") + " ";

            sendBuffer = Encoding.ASCII.GetBytes(udpString);
            //sendBuffer = Encoding.ASCII.GetBytes(temp_sum.ToString());
            udpSocket2.SendTo(sendBuffer, remote2EP);
            this.Title = udpString;



            //Declares all 20 joint positons
            /*
            Joint joint0 = skeleton.Joints[JointType.Head];
            Joint joint1 = skeleton.Joints[JointType.ShoulderCenter];
            Joint joint2 = skeleton.Joints[JointType.ShoulderRight];
            Joint joint3 = skeleton.Joints[JointType.ShoulderLeft];
            Joint joint4 = skeleton.Joints[JointType.Spine];
            Joint joint5 = skeleton.Joints[JointType.HipCenter];
            Joint joint6 = skeleton.Joints[JointType.HipRight];
            Joint joint7 = skeleton.Joints[JointType.HipLeft];
            Joint joint8 = skeleton.Joints[JointType.ElbowRight];
            Joint joint9 = skeleton.Joints[JointType.ElbowLeft];
            Joint joint10 = skeleton.Joints[JointType.WristRight];
            Joint joint11 = skeleton.Joints[JointType.WristLeft];
            Joint joint12 = skeleton.Joints[JointType.HandRight];
            Joint joint13 = skeleton.Joints[JointType.HandLeft];
            Joint joint14 = skeleton.Joints[JointType.KneeRight];
            Joint joint15 = skeleton.Joints[JointType.KneeLeft];
            Joint joint16 = skeleton.Joints[JointType.AnkleRight];
            Joint joint17 = skeleton.Joints[JointType.AnkleLeft];
            Joint joint18 = skeleton.Joints[JointType.FootRight];
            Joint joint19 = skeleton.Joints[JointType.FootLeft];
            */
            //Displays X,Y,and Z in window title for the specified marker position
            //this.Title = "(" + joint19.Position.X.ToString() + " , " +
joint19.Position.Y.ToString() + " , " + joint19.Position.Z.ToString() + ")";
```

```
        //Records all of the data for the session
        /*
        writer.Write("{0} \t {1} \t {2} \t", joint0.Position.X.ToString(),
joint0.Position.Y.ToString(), joint0.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint1.Position.X.ToString(),
joint1.Position.Y.ToString(), joint1.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint2.Position.X.ToString(),
joint2.Position.Y.ToString(), joint2.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint3.Position.X.ToString(),
joint3.Position.Y.ToString(), joint3.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint4.Position.X.ToString(),
joint4.Position.Y.ToString(), joint4.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint5.Position.X.ToString(),
joint5.Position.Y.ToString(), joint5.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint6.Position.X.ToString(),
joint6.Position.Y.ToString(), joint6.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint7.Position.X.ToString(),
joint7.Position.Y.ToString(), joint7.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint8.Position.X.ToString(),
joint8.Position.Y.ToString(), joint8.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint9.Position.X.ToString(),
joint9.Position.Y.ToString(), joint9.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint10.Position.X.ToString(),
joint10.Position.Y.ToString(), joint10.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint11.Position.X.ToString(),
joint11.Position.Y.ToString(), joint11.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint12.Position.X.ToString(),
joint12.Position.Y.ToString(), joint12.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint13.Position.X.ToString(),
joint13.Position.Y.ToString(), joint13.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint14.Position.X.ToString(),
joint14.Position.Y.ToString(), joint14.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint15.Position.X.ToString(),
joint15.Position.Y.ToString(), joint15.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint16.Position.X.ToString(),
joint16.Position.Y.ToString(), joint16.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint17.Position.X.ToString(),
joint17.Position.Y.ToString(), joint17.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \t", joint18.Position.X.ToString(),
joint18.Position.Y.ToString(), joint18.Position.Z.ToString());
        writer.Write("{0} \t {1} \t {2} \n", joint19.Position.X.ToString(),
joint19.Position.Y.ToString(), joint19.Position.Z.ToString());
        */

        // Render Joints
        foreach (Joint joint in skeleton.Joints)
```

```
            {
                Brush drawBrush = null;

                if (joint.TrackingState == JointTrackingState.Tracked)
                {
                    drawBrush = this.trackedJointBrush;
                }
                else if (joint.TrackingState == JointTrackingState.Inferred)
                {
                    drawBrush = this.inferredJointBrush;
                }

                if (drawBrush != null)
                {
                    drawingContext.DrawEllipse(drawBrush, null,
this.SkeletonPointToScreen(joint.Position), JointThickness, JointThickness);
                }
            }
        }

        public double getVerticalAngle(Joint shoulder, Joint wrist)
        {
            float diffx=wrist.Position.X-shoulder.Position.X;
            float diffy=wrist.Position.Y-shoulder.Position.Y;
            float diffz=wrist.Position.Z-shoulder.Position.Z;
            double
mag=Math.Sqrt(Math.Pow(diffx,2)+Math.Pow(diffy,2)+Math.Pow(diffz,2));
            return Math.Asin(diffy / mag) * 180 / Math.PI;
        }

        public double getHorizontalAngle(Joint shoulder, Joint wrist)
        {
            float diffx = wrist.Position.X - shoulder.Position.X;
            float diffz = wrist.Position.Z - shoulder.Position.Z;
            double mag = Math.Sqrt(Math.Pow(diffx, 2) + Math.Pow(diffz, 2));
            return Math.Asin(-diffx / mag)*180/Math.PI;
            //return Math.Atan2(diffx, diffz);
        }

        public double maxArmLength = 0;

        public double getBowStrength(Joint shoulder, Joint wrist)
        {
            float diffx = wrist.Position.X - shoulder.Position.X;
            float diffy = wrist.Position.Y - shoulder.Position.Y;
            float diffz = wrist.Position.Z - shoulder.Position.Z;
```

```
            double mag = Math.Sqrt(Math.Pow(diffx, 2) + Math.Pow(diffy, 2) +
Math.Pow(diffz, 2));
            maxArmLength = (maxArmLength < mag) ? mag : maxArmLength;
            double temp_strength=(maxArmLength - mag) / maxArmLength * 60.0;
            if (temp_strength<0.0)
               temp_strength=0;
            else if (temp_strength>40.0)
               temp_strength=40.0;
            return temp_strength;
        }

        /// <summary>
        /// Maps a SkeletonPoint to lie within our render space and converts to Point
        /// </summary>
        /// <param name="skelpoint">point to map</param>
        /// <returns>mapped point</returns>
        private Point SkeletonPointToScreen(SkeletonPoint skelpoint)
        {
            // Convert point to depth space.
            // We are not using depth directly, but we do want the points in our 640x480
output resolution.
            DepthImagePoint depthPoint = this.sensor.MapSkeletonPointToDepth(skelpoint,

 DepthImageFormat.Resolution640x480Fps30);

            return new Point(depthPoint.X, depthPoint.Y);
        }

        /// <summary>
        /// Draws a bone line between two joints
        /// </summary>
        /// <param name="skeleton">skeleton to draw bones from</param>
        /// <param name="drawingContext">drawing context to draw to</param>
        /// <param name="jointType0">joint to start drawing from</param>
        /// <param name="jointType1">joint to end drawing at</param>
        private void DrawBone(Skeleton skeleton, DrawingContext drawingContext,
JointType jointType0, JointType jointType1)
        {
            Joint joint0 = skeleton.Joints[jointType0];
            Joint joint1 = skeleton.Joints[jointType1];

            // If we can't find either of these joints, exit
            if (joint0.TrackingState == JointTrackingState.NotTracked ||
               joint1.TrackingState == JointTrackingState.NotTracked)
            {
               return;
```

```
            }

            // Don't draw if both points are inferred
            if (joint0.TrackingState == JointTrackingState.Inferred &&
                joint1.TrackingState == JointTrackingState.Inferred)
            {
                return;
            }

            // We assume all drawn bones are inferred unless BOTH joints are tracked
            Pen drawPen = this.inferredBonePen;
            if (joint0.TrackingState == JointTrackingState.Tracked && joint1.TrackingState
== JointTrackingState.Tracked)
            {
                drawPen = this.trackedBonePen;
            }

            drawingContext.DrawLine(drawPen, this.SkeletonPointToScreen(joint0.Position),
this.SkeletonPointToScreen(joint1.Position));
        }

        /// <summary>
        /// Handles the checking or unchecking of the seated mode combo box
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
        private void CheckBoxSeatedModeChanged(object sender, RoutedEventArgs e)
        {
            if (null != this.sensor)
            {
                if (this.checkBoxSeatedMode.IsChecked.GetValueOrDefault())
                {
                    this.sensor.SkeletonStream.TrackingMode = SkeletonTrackingMode.Seated;
                }
                else
                {
                    this.sensor.SkeletonStream.TrackingMode =
SkeletonTrackingMode.Default;
                }
            }
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            try
            {
```

```
                this.Close();
            }
            catch
            { }
        }
    }

    public class UdpRecvData
    {
        public string recvData;
    }
}
```

Appendix: IRB approval

UNIVERSITY of WISCONSIN
UWMILWAUKEE

Department of University Safety & Assurances

Jessica Rice
IRB Administrator
Institutional Review Board
Engelmann 270
P. O. Box 413
Milwaukee, WI 53201-0413
(414) 229-3182 *phone*
(414) 229-6729 *fax*

http://www.irb.uwm.edu
ricej@uwm.edu

Modification/Amendment - IRB Expedited Approval

Date: January 28, 2014

To: Na Jin Seo, PhD
Dept: Industrial and Manufacturing Engineering

Cc: Jayashree Arunkumar

IRB#: 12.406
Title: Low-Cost Wireless Virtual Reality System to Improve Stroke Survivor's Upper Limb Function

After review of your research protocol by the University of Wisconsin – Milwaukee Institutional Review Board, your protocol has received modification/amendment approval for:

- Addition of questions to the questionnaire

IRB approval will expire on **August 6, 2014**. If you plan to continue any research related activities (e.g., enrollment of subjects, study interventions, data analysis, etc.) past the date of IRB expiration, a Continuation for IRB Approval must be filed by the submission deadline. If the study is closed or completed before the IRB expiration date, please notify the IRB by completing and submitting the Continuing Review form found on the IRB website.

Unless specifically where the change is necessary to eliminate apparent immediate hazards to the subjects, any proposed changes to the protocol must be reviewed by the Institutional Review Board before implementation.

Please note that it is the principal investigator's responsibility to adhere to the policies and guidelines set forth by the University of Wisconsin – Milwaukee and its Institutional Review Board. It is the principal investigator's responsibility to maintain proper documentation of its records and promptly report to the Institutional Review Board any adverse events which require reporting.

Contact the IRB office if you have any further questions. Thank you for your cooperation and best wishes for a successful project

Respectfully,

Jessica Rice
IRB Administrator

Appendix: Questionnaires & Instruction manual

Questionnaires:

Questionnaire I: Pre-game survey : User expectation

Subject #: Date:

Virtual rehabilitation description: games or interactive activities where movements are required from the user to perform the game/activity. Kinect, Wii etc. are examples of interactive games or virtual reality games. Here we deal with games specifically dedicated to upper-limb rehabilitation, designed for hemiparetic people recovery and that people could use at home.

This questionnaire has been designed to better understand your expectations about virtual rehabilitation systems. This first questionnaire is divided in three different parts and consists of 27 questions.

Part I –A) Rating:

You are going to rate the importance of the following criteria for a rehabilitation game. According to you, how important is each element for a rehabilitation game? For each question rank the importance from 0 (not important) to 5 (essential).

1. How important is it for the rehabilitation gaming equipment to be **easy to put on and use**?

1	2	3	4	5
(Not important)				(Very important)

2. How important is it for any rules and goals of the game to be **easy to understand**?

1	2	3	4	5
(Not important)				(Very important)

3. How important is it for the game to be **interesting**?

1	2	3	4	5
(Not important)				(Very important)

4. How important is it for the game to be **challenging/motivating**?

1	2	3	4	5
(Not important)				(Very important)

5. How important are the **fancy graphics**, or the display and pictures provided on the screen?

1	2	3	4	5
(Not important)				(Very important)

6. How important is it to keep a track of your **score related to your progression** in the game?

1	2	3	4	5
(Not important)				(Very important)

7. How important is it to have a **variety of different scenes, activities and games**?

1	2	3	4	5
(Not important)				(Very important)

8. How important is it for a rehabilitation game to **adapt** to your progress and movement ability during use?

1	2	3	4	5
(Not important)				(Very important)

9. How important is it for a virtual rehabilitation system to provide a **clinical functional score** of your movement ability that therapists usually use?

1	2	3	4	5
(Not important)				(Very important)

10. How important is it for a rehabilitation game to **have a proven clinical benefit**?

1	2	3	4	5
(Not important)				(Very important)

Questionnaire II: Post-game survey : Games rating

Subject #: Date:

You have now tested two games dedicated to rehabilitation. You played the archery game and the USA-map puzzle game. In this questionnaire you are going to evaluate these two games according to several criteria and give your remarks and suggestions.

Part I- Rating:

In this part you are going to rate each game regarding several criteria. For each question, give your opinion from 0 to 5 regarding to the proposed criteria and the specified game.

A- For the puzzle game (USA map):

1. Do you think that the game equipment was **easy to put on, start and use**?

1	2	3	4	5
(Impossibly difficult)	(Somewhat difficult)	(OK)	(Simple enough)	(Very easy)

2. Were the provided **instructions helpful** to install and play the game?

1	2	3	4	5
(Completely useless)	(Useless)	(Helpful)	(Simple and helpful)	(Very simple and helpful)

3. Was it **easy to understand** the goal and the rules?

1	2	3	4	5
(Impossibly difficult)	(Somewhat difficult)	(OK)	(Simple enough)	(Very easy)

4. Were the **fancy graphics** and the display satisfactory?

1	2	3	4	5
(Very unsatisfactory)	(Unsatisfactory)	(OK)	(Satisfactory)	(Exceeded my expectation)

5. Was the **scoring system** appropriate?

1	2	3	4	5
(Completely inadequate)	(Inadequate)	(OK)	(adequate)	(Completely adequate)

6. Was it **interesting** to play?

1	2	3	4	5
(Very boring)	(Not interesting)	(OK)	(Interesting)	(Fascinating)

7. Was the game **challenging/motivating** to play?

1	2	3	4	5
(Not-challenging at all)	(Not challenging)	(OK)	(Challenging)	(Very challenging)

8. Was the game **difficult** to play?

1	2	3	4	5
(Really too easy)		(Just difficult enough)		(Impossible to realize)

9. Were the **shoulder and elbow movements** requested to play too difficult/too easy to realize?

1	2	3	4	5
(Really too easy)		(Just difficult enough)		(Impossible to realize)

10. Were the **hand (fingers) movements** requested to play too difficult/too easy to realize?

1	2	3	4	5
(Really too easy)		(Just difficult enough)		(Impossible to realize)

11. Did the game have **various scenes and activities**?

1	2	3	4	5
(Really too simple)		(Various enough)		(Too much variety)

12. Did the game provide **clinical feedback** about your game performance?

1	2	3	4	5
(Not at all)		(Appropriate amount of feedback)		(Too much to understand)

13. Did the game seem to have **a proven clinical benefit**?

1	2	3	4	5
(Not at all complicated)		(Clinically beneficial)		(Too complicated)

14. Give your opinion and remarks about the game you have tested:

_____
_____
_____
_____
_____
_____
_____

B- For the archery game:

1. Do you think that the game equipment was **easy to put on, start and use**?

1	2	3	4	5
(Impossibly difficult)	(Somewhat difficult)	(OK)	(Simple enough)	(Very easy)

2. Were the provided **instructions helpful** to install and play the game?

1	2	3	4	5
(Completely useless) and helpful)	(Useless)	(Helpful)	(Simple and helpful)	(Very simple

3. Was it **easy to understand** the goal and the rules?

1	2	3	4	5
(Impossibly difficult)	(Somewhat difficult)	(OK)	(Simple enough)	(Very easy)

4. Were the **fancy graphics** and the display satisfactory?

1	2	3	4	5
(Very unsatisfactory)	(Unsatisfactory)	(OK)	(Satisfactory)	(Exceeded my expectation)

5. Was the **scoring system** appropriate?

1	2	3	4	5
(Completely inadequate)	(Inadequate)	(OK)	(adequate)	(Completely adequate)

6. Was it **interesting** to play?

1	2	3	4	5
(Very boring)	(Not interesting)	(OK)	(Interesting)	(Fascinating)

7. Was the game **challenging/motivating** to play?

1	2	3	4	5
(Not-challenging at all)	(Not challenging)	(OK)	(Challenging)	(Very challenging)

8. Was the game **difficult** to play?

1	2	3	4	5
(Really too easy)		(Just difficult enough)		(Impossible to realize)

9. Were the **shoulder and elbow movements** requested to play too difficult/too easy to realize?

1	2	3	4	5
				(Impossible to realize)

(Really too easy) (Just difficult enough)

10. Were the **hand (fingers) movements** requested to play too difficult/too easy to realize?

1	2	3	4	5
(Really too easy)		(Just difficult enough)		(Impossible to realize)

11. Did the game have **various scenes and activities**?

1	2	3	4	5
(Really too simple)		(Various enough)		(Too much variety)

12. Did the game provide **clinical feedback** about your game performance?

1	2	3	4	5
(Not at all)		(Appropriate amount of feedback)		(Too much to understand)

13. Did the game seem to have **a proven clinical benefit**?

1	2	3	4	5
(Not at all complicated)		(Clinically beneficial)		(Too complicated)

14. Give your opinion and remarks about the game you have tested:

_____
_____
_____
_____
_____
_____
_____

C- <u>For the Kitchen game</u>:

1. Do you think that the game equipment was **easy to put on, start and use**?

1	2	3	4	5
(Impossibly difficult)	(Somewhat difficult)	(OK)	(Simple enough)	(Very easy)

2. Were the provided **instructions helpful** to install and play the game?

1	2	3	4	5
(Completely useless)	(Useless)	(Helpful)	(Simple and helpful)	(Very simple and helpful)

3. Was it **easy to understand** the goal and the rules?

1	2	3	4	5

(Impossibly difficult)	(Somewhat difficult)	(OK)	(Simple enough)	(Very easy)

4. Were the **fancy graphics** and the display satisfactory?

1	2	3	4	5
(Very unsatisfactory)	(Unsatisfactory)	(OK)	(Satisfactory)	(Exceeded my expectation)

5. Was the **scoring system** appropriate?

1	2	3	4	5
(Completely inadequate)	(Inadequate)	(OK)	(adequate)	(Completely adequate)

6. Was it **interesting** to play?

1	2	3	4	5
(Very boring)	(Not interesting)	(OK)	(Interesting)	(Fascinating)

7. Was the game **challenging/motivating** to play?

1	2	3	4	5
(Not-challenging at all)	(Not challenging)	(OK)	(Challenging)	(Very challenging)

8. Was the game **difficult** to play?

1	2	3	4	5
(Really too easy)		(Just difficult enough)		(Impossible to realize)

9. Were the **shoulder and elbow movements** requested to play too difficult/too easy to realize?

1	2	3	4	5
(Really too easy)		(Just difficult enough)		(Impossible to realize)

10. Were the **hand (fingers) movements** requested to play too difficult/too easy to realize?

1	2	3	4	5
(Really too easy)		(Just difficult enough)		(Impossible to realize)

11. Did the game have **various scenes and activities**?

1	2	3	4	5
(Really too simple)		(Various enough)		(Too much variety)

12. Did the game provide **clinical feedback** about your game performance?

1	2	3	4	5
(Not at all)		(Appropriate amount of feedback)		(Too much to understand)

13. Did the game seem to have **a proven clinical benefit**?

1	2	3	4	5
(Not at all complicated)		(Clinically beneficial)		(Too complicated)

14. Give your opinion and remarks about the game you have tested:

_____
_____
_____
_____
_____
_____
_____

Instruction manual

Virtual Kitchen Game Instructions

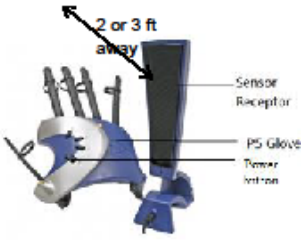
Step 1:

Place the P5 Glove on your right hand with your fingers in the rings. Pull the palm strap across the palm to secure it to the hook. Turn the glove on by pressing the large button on the back of the hand.

The glove is on if you can see a red LED light at the bottom of the sensor receptor.

Step 2:

Position the P5 sensor receptor on your right side facing towards you. Make sure the sensor is unobstructed.



Step 3:

Position the Kinect sensor against the monitor on the left side facing towards you (see figure). Make sure the sensor is unobstructed.

Step 4:

Using the mouse, double click the KitchenGame icon on the desktop and follow the on-screen instructions. Please stand 4-5 ft away from the monitor while playing the game.

Step 5:

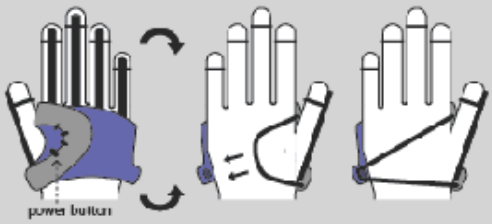
Once you have completed the game, press the escape key to close the program.

To play the game again restart the game.

Archery Game Instruction

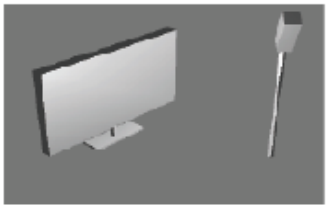
Step 1

Place the glove on your right hand with your fingers in the rings. Place your thumb inside the palm strap and secure it to the hook by the pinky. Turn the glove on by pressing the large button on the back.



Step 2

Position the P5 sensor tower on your right side facing towards you (see figure).
Make sure the sensor is unobstructed, and the LED light is on.

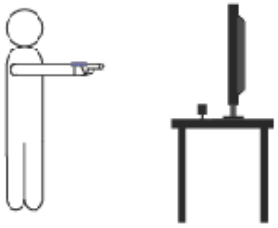


Step 3

Position the Kinect sensor against the screen on the right side facing towards you. Make sure the sensor is unobstructed.

Step 4

Using the mouse, double click the *Archery Game* icon on the desktop and follow the on screen instructions. While playing the game, please stand 4-5 ft away from the screen.
To aim the bow, outstretch your arm to the target.
To load the arrow, close your fingers.
To fire the arrow, completely open your fingers.



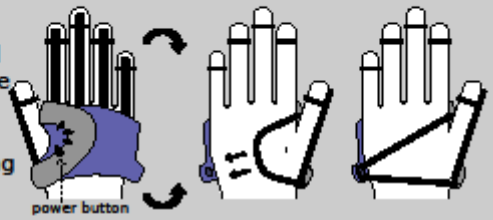
Step 5

Once you have completed the game, press the escape key to close the program.

To run the first game

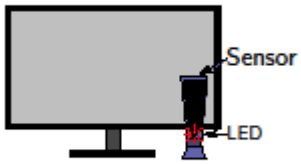
Step 1

Place the glove on your right hand with your fingers in the rings. Place your thumb inside the palm strap and secure it to the hook by the pinky. Turn the glove on by pressing the large button on the back.



Step 2

Position the sensor against the screen on the right sidefacing towards you. Make sure it the sensor unobstructed,the glove is plugged in, and the LED light is on (press the power button if not).

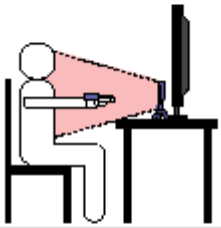


Step 3

Sit down approximately 2-3 feet from the sensor.

Step 4

Using the mouse, double click the Puzzle Game icon on the desktop and follow the on screen instructions. Be sure to keep the glove within the view of the sensor while completing the instructions.



Step 5

Once you have completed the task using the P5 Glove, press the escape key to close the program.