Games for Therapy: Defining a Grammar and Implementation for the Recognition of Therapeutic Gestures

David Maung
The Ohio State University
CSE Department
395 Dreese Laboratories
2015 Neil Avenue
Columbus, OH 43210-1277
+1 858 882 7211
maung.4@osu.edu

Roger Crawfis, PhD
The Ohio State University
CSE Department
395 Dreese Laboratories
2015 Neil Avenue
Columbus, OH 43210-1277
+1 614 292 2566
crawfis@cse.ohio-state.edu

Lynne V. Gauthier, PhD
The Ohio State University
Dept. of Physical Medicine and Rehabilitation
390 W 9th Avenue
Columbus, OH 43210
lynne.gauthier@osumc.edu

Lise Worthen-Chaudhari, MFA, MS, CCRC
The Ohio State University
Dept. of Physical Medicine and Rehabilitation
390 W 9th Avenue
Columbus, OH 43210
worthen-chaudhari.1@osu.edu

Linda P. Lowes, PT, PhD
Nationwide Children's Hospital
700 Children's Drive
Columbus, OH 43205
linda.lowes@nationwidechildrens.org

Alex Borstad, PhD, PT, NCS
The Ohio State University
Dept. of Physical Medicine and Rehabilitation
390 W 9th Avenue
Columbus, OH 43210
alexandra.borstad@osumc.edu

Ryan J. McPherson
The Ohio State University
ECE Department
205 Dreese Laboratories
2015 Neil Avenue
Columbus, OH 43210-1277
mcpherson.119@osu.edu

ABSTRACT

We introduce a methodology and grammar for the description and development of therapeutic gestures using a natural user interface (NUI) device such as the Microsoft Kinect and provide some implementation guidelines for developing therapeutic games using these gestures. Gesture recognition is a widely studied area and the Kinect SDK provides skeletal tracking to assist in pose determination. However, most gesture recognition systems assume a rather pristine gesture. Noise in the system is filtered, but the underlying gesture is fairly robust. For Physical Therapy (PT) performed for individuals with neuromotor deficits, gestures need to allow for a wider spectrum of possible motion paths. For instance, following a stroke that results in hemiparesis, individuals often display patterns of motor compensation, thus altering the trajectories of many motor movements. This has required a rethinking of how gestures are defined and the development of a more analytical and simplistic formulation of gestures. To this end, this paper discusses our gesture recognition framework and defines a simple grammar used to extract derived measurements and build finite-state machines for gesture recognition that can more easily be controlled by knowledge experts than machine learning techniques. Our target is an in-home alternative of constraint-induced movement therapy (CI Therapy), the current gold standard. The results of CI Therapy are substantial, but the widespread application of the treatment is hindered by the costs and travel difficulties associated with out-patient care.

General Terms
Design.

Keywords
Therapy, Gestures, Kinect, Serious Games.

1. INTRODUCTION

Stroke is a leading cause of disability in the United States, affecting about 700,000 people per year. About 50% of survivors become disabled with regard to arm-hand functionality [8]. A form of motor rehabilitation, termed constraint-induced motion

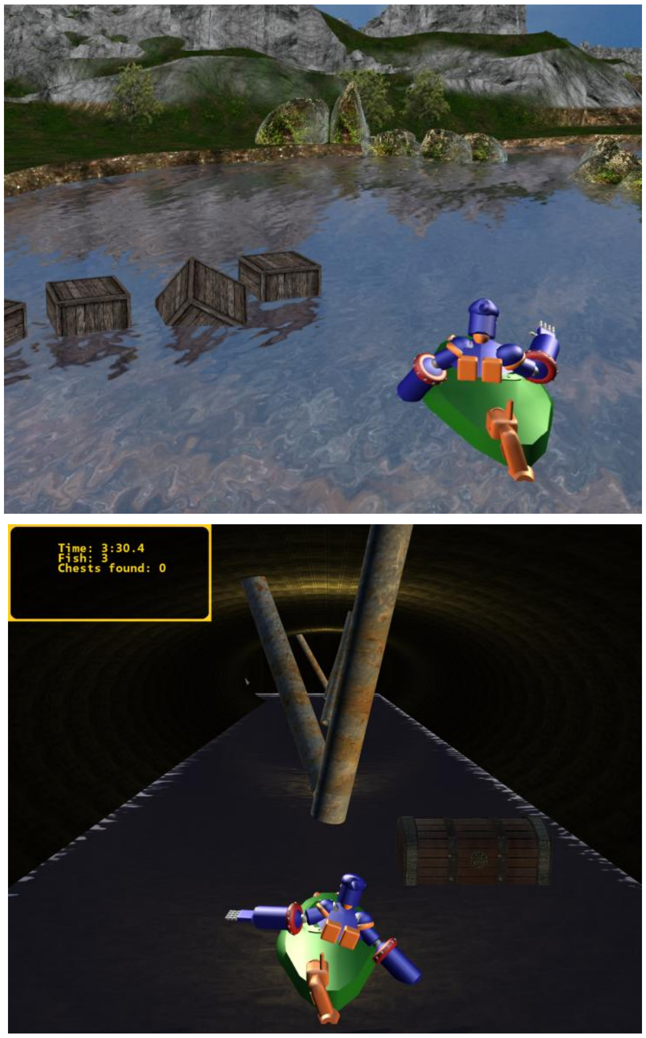

Figure 1: Screenshots from the Canyon Adventure game. Players must navigate around obstacles in the river, traverse rapids, and explore an underground maze. Other game activities include fishing and collecting treasure chests.

therapy (CI Therapy), has been shown to provide improvements in paretic arm function and frequency of use [12], and to promote structural and functional brain plasticity [4]. Although viewed as the "gold standard" intervention by many practitioners, CI therapy is unavailable to most stroke survivors due to its cost, travel/scheduling demands, and dearth of trained providers.

To overcome these accessibility challenges, our team is developing a game using the Microsoft Kinect and XNA game studio environment which is designed to provide a similar therapy in the home setting. See Figure 1. To the best of our knowledge, this is the first attempt to create a stand-alone home-based rehabilitation program for upper extremity hemiparesis based on an empirically-validated treatment. Although a few avatar-based virtual reality games and programs [5] have been designed for rehabilitation, they have not promoted critical motor learning elements, such as intensity of practice and carry-over of therapy gains to daily activities, that are being implemented here and have typically been of low complexity (prompting the person to reach out and touch virtual objects in repetitive patterns). Furthermore, other robotic technologies offered to facilitate delivery of motor therapy post-stroke treatment have been criticized for not incorporating critical motor learning elements or delivering empirically-validated treatments [2]. Existing stroke rehabilitation technologies may supplement therapy effectively, but are thus unlikely to be successful stand-alone interventions that can be implemented in a home setting.

In keeping with standard CI therapy protocol, game content is being designed to support three hours of game play per day for ten consecutive weekdays. Our game, called Canyon Adventure, implements a procedurally generated river environment and a procedurally generated underground waterway system to provide many hours of playable content. Therapy is provided by mapping therapeutic gestures to in-game motions and requiring these motions to be performed repetitively throughout the course of game play.

The definition of these therapeutic gestures is a major component of the development effort. We needed to work closely and iteratively with colleagues from other disciplines, who do not code, but whose expertise is critical in order to design and implement clinically relevant movement within the context of a computer game. We report our solution: detection of state sequences describing pathologic gestures - gestures that we expected to change substantially over an intensive two week course of game play. This paper describes our programming framework and an XML grammar for the formulation of complex gestures that are suitable for therapy. We describe the implementation of this framework and grammar in our Canyon Adventure game and report initial results from play testing among individuals who have had a stroke. Finally, we provide insights to lessons learned when attempting to define and implement a set of therapeutic gestures into a game.

2. BACKGROUND

Microsoft provides the Kinect for Windows SDK [10] as a programming toolkit for working with the Microsoft Kinect. The SDK provides the ability to view video camera and depth camera images as well as skeletal data and provides a starting point for application development. Skeletal data is represented as a set of 20 points referred to as joints. Of the joints tracked by the SDK we used those we could track reliably for a person who is seated: bilateral pelvis, shoulders, elbows, wrists, and hands as well as the unilateral center points for head, center shoulder (approximately cervical spine), spine (approximately thoracic spine) and hip center (approximately lumbar spine). For each joint, the Kinect SDK provides orientation in world space, hierarchical orientation, and position.

To this point, gesture recognition has largely been left as an exercise for the programmer. The Microsoft Kinect SDK provides some guidelines on the design of gestures [9]; however, these guidelines do not necessarily support communication about or formulation of complex gestures such as the therapeutic gestures we develop in our Canyon Adventures game. Echtler attempts to describe a unified gesture description language [3]; however, this language only considers a position in 2D or 3D space and was not designed with a complete skeletal frame in mind. Lai discusses a mathematical model for the recognition of gestures without regard to their definition or implementation [7]. Bleiweiss discusses a blending of pregenerated animations with player movement and gestures [1]. While there have been some attempts to recognize gestures by machine learning [6][11], we believe a heuristic method allows knowledge experts to directly contribute to the development of gestures. For our team, we found that creating a language definition and corresponding XML definition, of therapeutic gestures and margins of error, served to support our development work.

The goal of rehabilitative therapy is to improve the ability to perform activities of daily living, or daily tasks (e.g., opening a door, holding a cup of water and drinking from it, putting on socks and shoes). Thus, we focused on detecting functional reaching movement, more specifically on upper extremity reaching and trunk gestures. We identified at least two main challenges for defining gestures with this rehabilitation framework in mind: 1) gestures performed during daily tasks range from very simple to very complex, and 2) motor function varies widely across the population of candidates for CI therapy. For example, some individuals may have limited voluntary control of their shoulder, elbow, and hand (e.g., are unable to straighten their fingers or elbow), whereas others may only lack fine-motor skills (i.e., finger dexterity). This suggests a need for a broad definition of gestures that can account for varying degrees of motor impairment. Our approach to gesture definition begins by defining a gesture as a set of postures or positions, referred to as states. Each state is defined as one or more criterion and relevant parameters. For a gesture to be recognized as completed, a user must pass through each state in succession from beginning to end. When a gesture is completed, a game mechanic is triggered.

3. DEFINING A THERAPEUTIC GESTURE

The priority movements were defined first in natural language: twist the body, raise the arm, tilt the head, reach across and behind. Then, these movements were defined in anatomic language: torso rotation (bilateral), shoulder abduction, shoulder flexion/extension, head tilt toward shoulder, elbow extension, elbow raised in global space, wrist raised in global space, wrist cross the midline of body, wrist cross behind frontal plane of body. Each priority movement was then defined in biomechanical language that specified

1) the reference frames (RFs) to consider,
2) the skeletal markers to consider.,
3) the global markers to consider..
4) the criterion and associated parameters for initial and end states per gesture.

We considered two potential RFs: the global RF (GRF), provided by the camera with origin at the camera lens, and the body RF (BRF) with origin located within the detected body. We define the body RF as follows. The positive X axis is defined as parallel to the vector originating at the left shoulder joint and continuing through the right shoulder joint. The positive Y axis is defined as the vector originating at the hip center joint and passing through the shoulder center joint. The positive Z axis is defined as parallel to the cross product of these two vectors ($X \times Y$). The hip center joint is the origin of the body coordinate frame.

At its simplest form, a gesture may consist of a single state and would be achieved when the criterion for that state is met. For example, a therapist might say, "Beginning with your arm at your side, move your arm sideways away from your hip, while keeping your arm straight." For the individual to accomplish this gesture, the individual must achieve a single state whose criterion is: The distance between the hand and the hip is greater than 10cm. More complex gestures might contain states that take many more criteria to define. An example of a more complex gesture is stirring, as in stirring a cauldron. To determine that a person is stirring, one could establish that the hand passed through a number of positions over time such as shown in Figure 2.

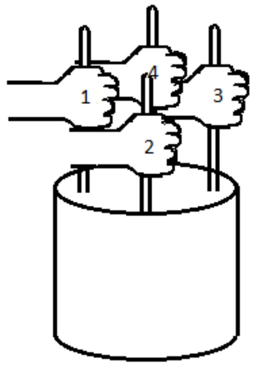

Figure 2: Stirring gesture and the states to recognize it: 1) hand in front of body close, 2) hand further out and right of center, 3) hand further out and center, and 4) hand closer and left of center.

The therapist designing the stirring gesture may decide to constrain the gesture further. For example, the therapist may decide to constrain the elbow position in each of the above states to be close to the center line of the body. Although the latter gesture definition may be most appropriate for some higher-functioning individuals, people with more motor dysfunction may not be able to accomplish the stirring gesture when stricter criteria are enforced. Our gesture definition accounts for this by allowing a state to consist of one or more criteria as necessary.

4. DEFINING STATES

A gesture state is considered to be a set of one or more criteria. Each criterion defines a condition which must be met. Criteria are defined in detail below. For example, if a therapist wants to define a gesture for elbow extension (e.g., straightening the arm), the major criterion for this gesture would be elbow angle. The specific measurements for this criterion would be the range of angles that would qualify as meeting that state. An example definition of this gesture could be as follows:

```
Definition of elbow extension
    State 1: Arm contracted
        Criterion 1: Elbow angle greater than 0 degrees.
        Criterion 2: Elbow angle less than 70 degrees.
    State 2: Arm extended
        Criterion 1: Elbow angle greater than 130 degrees.
        Criterion 2: Elbow angle less than 180 degrees.
```

5. CRITERIA FOR GESTURES

The need for a specific mathematical definition for criteria is clear, but there is also a desire for simplicity. In a perfect world, our gesture language would be broad enough to encompass the necessary gestures for therapy and no broader. In this respect we have attempted to maintain a minimal definition of criteria which handles all of the important gesture requirement cases that the therapists on our team have given us to date.

We will build our definition of criteria from the skeletal data provided by the Kinect for Windows SDK. Specifically we use the provided joint position information, with the fundamental states in our grammar being based on one or more of the following three elements: 1) the position of a skeletal joint, 2) a vector between two joints, and 3) an angle defined as the angular measure between two vectors.

The position of a skeletal joint may be used, for example, to detect that a given position is held for a set amount of time. To implement this we added the hold criterion which specifies a joint, a margin of error, and a time duration. To satisfy this criterion, the specified joint must not deviate from its average position further than the margin of error within the specified time.

A vector between two joints is used, for example, to detect the arm raise gesture wherein the goal is to measure the distance the person has moved the hand from the hip. Thus we measure the distance between the two skeletal joints as the length of the resultant vector between the hip joint and the hand joint. This specification can be constrained (or projected) to only measure the distance in a single coordinate axis of a coordinate frame. For example, the therapist may wish to measure how far above the shoulder the players are able to raise their affected hands. In this case we would use the y component of the vector from the shoulder joint to the wrist joint.

The angle between two vectors is the basis for much of our therapeutic gesture recognition. The vectors comprising an angle of interest may or may not involve consecutive joints. An example of an angle between consecutive joints would be the elbow angle defined as the angle between the elbow-to-shoulder vector and the elbow-to-wrist vector. An example of an angle not involving two consecutive joints could be an angle between the shoulder-center-to-hip-center vector and the shoulder-to-wrist vector. We use this specific angle in our rowing gesture described in more detail below. This functionality requires the ability to query the angle between any two vectors, regardless of whether these are contiguous.

For convenience we have also added all of the axes of both our world and body coordinate system as vectors which can be used in angle measurements. These are:

- ±world-x-axis
- ±world-y-axis
- ±world-z-axis
- ±body-x-axis
- ±body-y-axis
- ±body-z-axis

There are specific cases, such as in our rowing gesture, where we found it convenient to constrain an angle to a plane. In this specific example, the therapists were not concerned with how close to the body the arm was while the person was rowing. We provided this functionality by projecting the arm onto the yz plane and measuring the angle with respect to the y axis. In general, to provide for this functionality, we have added the following constraints which can be used with the angle criteria: xy, xz, and yz.

It is desirable for these gestures to operate for either the left or right side of the body, given that motor training within the game is primarily administered to the hemiparetic side, which varies across participants. Also, it is sometimes desirable to constrain or require motion in the opposite arm as well as the more affected arm. Instead of explicitly labeling joints left or right, we use the labels primary (for the more affected arm) or secondary (for the less affected arm) and use a game option to select whether primary is the left or right side.

6. GRAMMAR DEFINITION

We define our grammar in XML as it is an easily human readable form that can be understood and manipulated by all participants across disciplines involved in the project. The XML example in Appendix A is provided to aid in the discussion of the following grammar.

6.1 <measurements> element

The measurements element contains all the named measurements which are of type <angle> or <vector>. These measurements will be referred to in gesture states.

6.1.1 Attributes

none

6.2 <gestures> element

The gestures element is a simple container contains all the named <gesture> elements for a given program.

6.2.1 Attributes

none

6.3 <angle> element

Specifies an angle measurement which contains exactly two vector elements and measures the angle between those vectors.

6.3.1 Attributes

name	Specifies the name of the angle criterion

6.4 <vector> element

There are three types of vectors:

1. A vector specified by a pair of joints.
2. A special axis vector specified using the type attribute.
3. The position vector which contains one joint and is the vector from the origin to that joint.

6.4.1 Attributes

name	The name of the vector.
type	Specifies the type of vector. Valid types are: default, world-x-axis, world-y-axis, world-z-axis, body-x-axis, body-y-axis, body-z-axis, and position.

6.5 <joint> element

Provides a reference to a specific skeletal joint. If the joint does not lie along the middle of the skeleton, you can specify whether it is the primary (affected side of the body) joint or secondary (unaffected side of the body). You must also specify the order of joints in a vector, with 0 being the origin and 1 being the endpoint.

6.5.1 Attributes

skeletal-joint	Specifies the Kinect skeletal joint to use. Valid values are ankle, knee, hip, hip-center, shoulder, shoulder-center, elbow, wrist, and head.
side	The side of the body. Valid values are primary or secondary.
order	The order of the joint. Valid values are 0 and 1.

6.6 <gesture> element

Specifies a single named gesture which can be recognized. A gesture must contain a set of one or more states. States contain criteria.

6.6.1 Attributes

name	The name for this gesture
type	The type of the gesture. Valid values are normal and measurement.

6.7 <state> element

Specifies a single state within a gesture which must be satisfied. A state contains one or more criteria.

6.7.1 Attributes

name	The name for this state (which can be used for display or debugging).
order	A numerical ordering of the gesture states. Valid values are non-negative integers.

6.8 <criterion> element

Specifies a single criterion which must be satisfied for a gesture state to be recognized.

6.8.1 Attributes

ref	Name of the criteria definition which this criteria uses.
component	Specifies what to measure and must match the type of criterion being referred to. Valid values are distance, x, y, or z. Note that Measure is not valid for an analog vector gesture. All components of the vector are returned to the subscriber for their use.
plane	Constrains an angle criterion to a plane. Valid values are xy, xz, yz, and none.
compare	Specifies how to compare the measure to satisfy this criterion. Valid comparisons are lt (for less than), gt (for greater then), lte (for less than or equal to) and gte (for greater than or equal to), and hold.
value	Specifies the value for the comparison or the allowable margin of error in the case of a "hold" criterion. This is a floating point number.
time	The amount of time for a hold gesture to be satisfied. This is a floating point number.

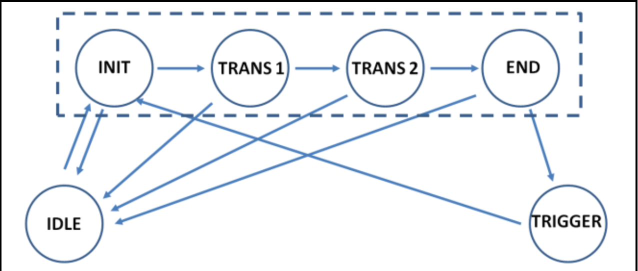

Figure 3. State machine for gesture recognition. States bordered by the dashed-line are part of the gesture.

7. IMPLEMENTATION

The grammar defined above is well suited to processing with a recursive descent parser. An object tree is created from the XML definition where each node implements an interface which we call IRecognizable. A gesture is then recognized by calling Recognize(SkeletalFrame skeleton) at the gesture level.

All gestures implement the *observable* design pattern. Measurement gestures notify their subscribers whenever the measurement they are tracking changes by some tolerance (which might be every update), while other gestures only notify their observers when the gesture is recognized.

Using this framework, gesture recognition is performed using a finite state machine (FSM) as shown in figure 3. The FSM starts in the idle state when it is inactive and transitions to the initial state when it becomes active. The input is a series of skeleton frames containing joint positions. Each state of the gesture specifies the criteria for the state to be recognized. If the state is recognized, the gesture is advanced to the next gesture state. When the final state in the gesture is recognized, an in-game action is triggered, and the state returns to the initial state. Other actions in the game may deactivate this gesture or take it to the idle state.

To limit the amount of time allowed for a gesture to be recognized, a ring buffer which stores a sliding history of skeletal frames is used and each gesture is checked against the contents of the ring buffer each update cycle. When a gesture is recognized, the contents of the buffer can be emptied to prevent overlapping gestures. Each gesture has its own finite state machine. During a particular segment of the game, several gestures can be active. Each of these runs their state machines in parallel.

The gesture manager class is a container class for all gestures. In an event driven game model the gesture manager performs the following operations on each update cycle.

```
foreach gesture in the set of all currently active gestures
{
    gesture.recognize( skeletalframe )
}
```

In the CI Therapy initial design, we specified the following 19 gestures:

- Arm to Side
- Bend Down
- Bilateral Shoulder Flexion
- Bilateral Wave
- Elbow Extension
- Elbow Raise
- Forward Arm Raise
- Head Side to Side
- Reach Across
- Reach Forward
- Reach Up
- Reach Up and Forward
- Rowing
- Saturday Night Fever (raise arm above head and reach for floor on opposite side of body)
- Scoop
- Shoulder Turn
- Stirring
- Sweep Left (starting with hand in lap, reach arm forward and to the right and sweep arm across body)
- Sweep Right (starting with hand in lap, reach arm forward and to the left and sweep arm across body)

8. RESULTS

The grammar above was able to handle the definition of all of the gestures which we initially specified in our Canyon Adventure game. Furthermore, the grammar proved adaptable in allowing us to modify the gestures as we tuned them for implementation.

To date, our gestures have been tested by several relatively high-functioning individuals who have experienced a stroke. These individuals were able to successfully navigate the Canyon Adventures game and their gestures were successfully recognized on the majority of attempts. Feedback from participants indicated that the gesture navigation was easy to learn.

Incorporating input from stakeholders (e.g. people who have experienced a stroke) into the design process has offered valuable insight into the definition and placement of gestures in games. By doing so, we were able to identify several situations in which gesture recognition would fail in the context of game play: namely when contiguous gestures had overlapping states. Examples include: 1) gestures for which the end state of one gesture overlapped with the beginning state of a subsequent gesture, 2) gestures which were the inverse of each other (e.g., for gestures consisting of only two states, the beginning and end states were reversed), and 3) gestures which share a start state but diverge.

Secondly, we learned that simple gesture definitions are better than the more complex, both in the ability to tune and the ability to distinguish and differentiate. For example, if the elbow position is irrelevant in a raise hand gesture, it is better to code it as a single measurement of hand position relative to the shoulder without regard to the elbow position.

Finally, participants reported that gesture-based feedback facilitated their successful implementation of the gestures during game play. We implemented this feedback by requiring that each gesture expose its total number of states and current state. Our current design is a feedback icon which shows a rotating dial which varies from zero to 100 percent in 360 degrees. We find this presents the correct information for a single gesture; however many sections of game content allow the player to choose from multiple gestures. Overcoming the challenge of providing accurate feedback in these sections of the game is a focus of future work.

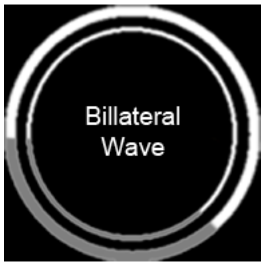

Figure 4. Feedback icon for a bilateral wave gesture that is 70% complete.

9. FUTURE WORK

Our interdisciplinary team will continue to define the gestures that are optimal for stroke rehabilitation and to develop content and in-game actions to promote successful implementation of these gestures by stakeholders. We are also working with individuals who have experienced a stroke to identify the most user-friendly methods of providing gesture feedback within the game. The end goal of this project is to provide an intuitive and engaging therapy for stroke survivors in a cost-effective manner. Home delivery trials are expected within the next six months.

10. ACKNOWLEDGEMENTS

I would like to acknowledge the contributions of the Kinetic Therapy team from the CSE 5912 Game Capstone class in the autumn 2012 semester at The Ohio State University. All of these undergraduate students provided significant contributions to the Canyon Adventure game content: Juan Roman, Daniel Bybik, and Tristan Reichardt. This work was supported by NSF PCORI grant number 60034417: “Designing a low-cost virtual reality gaming platform for neurorehabilitation of hemiparesis.”

11. REFERENCES

[1] Amit Bleiweiss, Dagan Eshar, Gershom Kutliroff, Alon Lerner, Yinon Oshrat, and Yaron Yanai. 2010. Enhanced interactive gaming by blending full-body tracking and gesture animation. In *ACM SIGGRAPH ASIA 2010 Sketches* (SA '10). ACM, New York, NY, USA, , Article 34 , 2 pages. DOI=10.1145/1899950.1899984 http://doi.acm.org/10.1145/1899950.1899984

[2] Brewer B, McDowell SK, Worthen-Chaudhari L. Poststroke Upper Extremity Rehabilitation: A Review of Robotic Systems and Clinical Results. Top Stroke Rehabil;14(6):22-44.

[3] Echtler, Florian, Gudrun Klinker, and Andreas Butz. "Towards a unified gesture description language." *Proceedings of the 13th International Conference on Humans and Computers*. University of Aizu Press, 2010.

[4] Gauthier LV, Taub E, Perkins C, Ortmann M, Mark VW, Uswatte G. Remodeling the brain: plastic structural brain

changes produced by different motor therapies after stroke. Stroke. 2008;39:1520-5

[5] Jun-Da Huang. 2011. Kinerehab: a kinect-based system for physical rehabilitation: a pilot study for young adults with motor disabilities. In The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '11). ACM, New York, NY, USA, 319-320. DOI=10.1145/2049536.2049627 http://doi.acm.org/10.1145/2049536.2049627

[6] Itauma, I.I.; Kivrak, H.; Kose, H.; , "Gesture imitation using machine learning techniques," *Signal Processing and Communications Applications Conference (SIU), 2012 20th* , vol., no., pp.1-4, 18-20 April 2012 doi: 10.1109/SIU.2012.6204822

[7] Kam Lai; Konrad, J.; Ishwar, P.; , "A gesture-driven computer interface using Kinect," *Image Analysis and Interpretation (SSIAI), 2012 IEEE Southwest Symposium on* , vol., no., pp.185-188, 22-24 April 2012 doi: 10.1109/SSIAI.2012.6202484

[8] Kelly-Hayes M, Beiser A, Kase CS, Scaramucci A, D'Agostino RB, Wolf PA. The influence of gender and age on disability following ischemic stroke: the Framingham study. J Stroke Cerebrovasc Dis. 2003;12:119-126.

[9] Kinect for Windows Human Interface Guidelines v1.5.0. Microsoft Corporation. http://msdn.microsoft.com/en-us/library/jj663791.aspx

[10] Microsoft Kinect for Windows SDK. Microsoft Corporation. http://msdn.microsoft.com/en-us/library/hh855347.aspx

[11] Yale Song, David Demirdjian, and Randall Davis. 2012. Continuous body and hand gesture recognition for natural human-computer interaction. *ACM Trans. Interact. Intell. Syst.* 2, 1, Article 5 (March 2012), 28 pages. DOI=10.1145/2133366.2133371 http://doi.acm.org/10.1145/2133366.2133371

[12] Taub E, Miller NE, Novack TA, Cook EW,3rd, Fleming WC, Nepomuceno CS, Connell JS, Crago JE. Technique to improve chronic motor deficit after stroke. Arch Phys Med Rehabil. 1993;74:347-54.

APPENDIX A: XML EXAMPLE

```xml
<root>
  <measurements>
    <angle name="shoulder-hand-angle">
      <vector type="default">
        <joint skeletal-joint="shoulder"
         side="primary" order="0"/>
        <joint skeletal-joint="wrist"
         side="primary" order="1"/>
      </vector>
      <vector type=”-body-y-axis>
    </angle>
    <vector name="hip-hand-vector">
      <joint skeletal-joint="hip" side="primary"
       order="0"/>
      <joint skeletal-joint="wrist" side="primary"
       order="1"/>
    </vector>
    <vector name="shoulder-hand-vector">
      <joint skeletal-joint="shoulder"
       side="primary" order="0"/>
      <joint skeletal-joint="wrist" side="primary"
       order="1"/>
    </vector>
    <vector name="wrist-position" type="position">
      <joint skeletal-joint="wrist"
       side="primary"/>
    </vector>
  </measurements>
  <gestures>
    <gesture name="rowing">
      <state name="initial" order="0">
        <criterion ref="shoulder-hand" plane="yz"
         compare="gt" value="45.0"/>
        <criterion ref="shoulder-hand" plane="yz"
         compare="lt" value="90.0"/>
      </state>
      <state name="middle" order="1">
        <criterion ref="shoulder-hand" plane="yz"
         compare="gt" value="25.0"/>
        <criterion ref="shoulder-hand" plane="yz"
         compare="lt" value="45.0"/>
      </state>
      <state name="final" order="2">
        <criterion ref="shoulder-hand" plane="yz"
         compare="gt" value="0.0"/>
        <criterion ref="shoulder-hand" plane="yz"
         compare="lt" value="25.0"/>
      </state>
    </gesture>
    <gesture name="raise-arm-to-side">
      <state name="initial" order="0">
        <criterion ref="shoulder-hand" plane="xy"
         compare="gt" value="0.0"/>
        <criterion ref="shoulder-hand" plane="xy"
         compare="lt" value="25.0"/>
      </state>
      <state name="middle" order="1">
        <criterion ref="shoulder-hand" plane="xy"
         compare="gt" value="25.0"/>
        <criterion ref="shoulder-hand" plane="yz"
         compare="lt" value="45.0"/>
            </state>
      <state name="final" order="2">
        <criterion ref="shoulder-hand" plane="xy"
         compare="gt" value="45.0"/>
        <criterion ref="shoulder-hand" plane="xy"
         compare="lt" value="90.0"/>
      </state>
    </gesture>
    <gesture name="hold-hand-out">
      <state name="final" order="2">
        <criterion ref="hip-hand-vector"
         component="y" compare="gt" value="5.0"/>
        <criterion ref="shoulder-hand-vector"
         component="y" compare="lt" value="-5.0"/>
        <criterion ref="wrist-position"
         compare="hold" value="2.5" time="3.0"/>
      </state>
    </gesture>
  </gestures>
</root>
```