1-1-2010

An Environment for Stroke Therapy Game Authoring

Simon Tam
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science

Department of Computer Science and Engineering

Thesis Examination Committee:
Caitlin Kelleher, Chair
Tao Ju
Cindy Grimm

AN ENVIRONMENT FOR STROKE THERAPY GAME AUTHORING

by

Simon Tam

A thesis presented to the School of Engineering
of Washington University in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

May 2010
Saint Louis, Missouri

ABSTRACT OF THE THESIS

An Environment for Stroke Therapy Game Authoring

by

Simon Tam

Master of Science in Computer Science

Washington University in St. Louis, 2010

Research Advisor: Professor Caitlin Kelleher

Stroke is the leading cause of disability in the industrialized world. The disabilities caused by stroke can significantly impact the quality of daily life for stroke survivors. Studies show that a high number of daily body movement repetitions can help recover lost motor functionality. However, many stroke patients do not perform the home exercises needed to reach this high number of repetitions. Games for rehabilitation have been shown to provide motivation to perform home exercises. However, because stroke can leave survivors with different levels of disability, custom games are required. Creating custom games, though, is time consuming and requires advanced programming knowledge. A tool for quick and easy game development is a solution to this problem. In this master's project, I collaborated with a doctoral occupational therapy student to take steps towards developing a game authoring environment custom to therapists.

Acknowledgements

I would like to thank the members of my research lab: Gazihan Alankus, Matt May, Paul Gross, and Kyle Harms, for all the support and help they have given me. I would like to especially thank Rachel Proffitt for the insight and expertise she provided in the field of Occupational Therapy. Additionally, I thank my Master's advisor Caitlin Kelleher for the guidance she supplied in both my Master's and my undergraduate work.

I would like to dedicate the success of my Master's thesis to my parents. My success as a student is entirely due to their inspiration and support. I also would like to thank Elizabeth Chen for her encouragement and patience.

Special thanks goes to the many students and distinguished faculty within my department who have reviewed this thesis and helped support my academic journey.

Simon Tam

Washington University in St. Louis
May 2010

CONTENTS

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Background

Stroke is the leading cause of disability in the industrialized world [6]. In just the United States, each year approximately 795,000 people are affected by stroke, and 185,000 people who survive a stroke go on to have another [31]. While the risk of stroke increases with age, strokes can occur at any age [6]. Recent data suggests that 45% of people who experience a stroke are younger than 65 (still of working age) and 27% are younger than 55 [39].

The disabilities caused by stroke can significantly affect the quality of daily life for the survivors. Hemiparesis, a partial paralysis of one side of the body, affects 50% of stroke survivors and limits the range of motion, strength, and motor control abilities [18]. Hemiparesis can prevent people who have experienced a stroke from returning to the work force or even living independently. A study found that only 20.64% of patients return to work, but not always to the same job [2]. Some people may further limit their participation in life due to fear of failure or insecurity about having a disability [20]. Because of these disabilities, stroke survivors must often depend on others to assist with the activities of daily living. This burden on the caregivers and the healthcare system is likely to increase with the aging population [38].

Research using animal models suggests that hundreds of daily repetitions of therapeutic motions can help animals recover a significant amount of lost motor functionality [34]. Recent studies suggest that this approach would similarly benefit in human stroke patient rehabilitation [22]. However, such a high number of repetitions cannot be achieved with therapy sessions alone. In typical sessions, patients often only complete less than forty active

or passive motions [22]. To achieve the needed number of repetitions, therapists recommend home exercises for patients but research shows that only 31% of them actually perform these exercises [35].

Early research into using video games for stroke rehabilitation suggests that games can provide a context that motivates patients to complete more therapeutic motions [4, 13, 11, 9]. However, there are a number of barriers that currently prevent large-scale use of games in stroke recovery. Existing commercial games require players to have a normal or close to normal range of motion. Many patients recovering from stroke simply cannot play existing motion games [9]. While some researchers have built games to address particular deficits [15], these games are often valuable to a relatively narrow range of patients. At present, the costs and time necessary to create a game to meet the rehabilitation needs of a particular patient are prohibitively high.

Tools that enable therapists to quickly create or customize games for patients may help to make widespread use of games in stroke rehabilitation possible.

1.2 Master's Project Work

For my master's project, I collaborated with a doctoral student from the Program in Occupational Therapy from the Washington University School of Medicine to work towards a game authoring environment for therapists. The work we did can be broken into a three step process:

1. We jointly conducted a study to analyze therapists' descriptions of existing stroke rehabilitation games and proposed a set of design guidelines for a therapist-centric programming language.
2. I implemented support for motion capturing devices in the Looking Glass IDE.
3. We created initial designs for tailoring the Looking Glass IDE for therapists.

This paper will present the work done in a format mirroring the three step process.

Chapter 2

Related Work

Our related work falls into three areas: 1) work on games for rehabilitation 2) programming environments for game authoring, and 3) programming languages designed for usability.

2.1 Work on Games for Rehabilitation

Games have been used increasingly in an attempt to provide motivating interventions for persons with stroke during therapy. Analysis of the field has shown that stroke therapy using games can be successful [30]. Existing console games with their alternative input devices, such as the Playstation 2 EyeToy [11] and Wii Sports [9], were found to be effective interventions. Unfortunately, they were designed for a population of healthy individuals and exclude persons with stroke who may lack the precision or range of motion necessary to play the games.

To widen the range of potential users, some research groups developed their own games. [7] demonstrated the simplest kind of games, in which the patient tried to move a colored circle from an initial position to a goal position. Simple games developed for data and haptic gloves by [12] and [13] included uncovering an image, scaring away butterflies, mosquitoes and UFOs, catching a ball, playing the piano and squeezing virtual pistons. In [5], games controlled with webcam color tracking included two whack-a-mole type games. Other games they created were controlled with magnetic sensors and included a physics-based orange catching game and a whack-a-mouse game. In addition, they used WiiMotes as pointing devices in a vibraphone music game. [40] included many simple games designed for different input devices where the goal of the game was manipulating objects in a 3-dimensional world. One study used games that were geared towards purposeful exercises including task-oriented

games that simulated everyday tasks using photographs of stoves and tables [33]. Most of these games were very basic and did not include customizations other than difficulty settings.

2.2 Programming Environments for Game Authoring

In 2001, researchers from Carnegie Mellon studied how non-programmers expressed solutions to programming problems [27]. The study was composed of two different parts. The first part examined how 12 fifth graders described scenes from a video game. The second examined how five adults and fourteen fifth graders created and modified a database. In both cases, the participants were asked to write down instructions to the computer that were necessary to accomplish the scenario that was shown. Independent raters then were asked to analyze the responses from the participants using a rating form to classify the answers into different categories. Some of the major conclusions from this study were that novices primarily used event and rule based statements, and that their programming operated on multiple objects in a single statement. These results contrast with the predominant imperative statements and iterative operations in commonly used programming languages. This study was significant in influencing how we designed our own procedures and analyses.

2.3 Programming Tools for Particular Audiences

As a profession, occupational and physical therapy has its own set of guidelines, rules, and most importantly, language. Domain specific languages (DSLs) are programming languages purposed for a particular application domain. DSLs are beneficial for the experts in the problem domain as the programming can be expressed in the idiom and level of abstraction appropriate for understanding, modifying and creating programs [36].

A very similar project to our own is the PlayWrite system. PlayWrite is an integrated development environment (IDE) that allows mental healthcare (MHC) professionals with little or no programming experience to create therapeutic games [8]. PlayWrite was designed

by an interdisciplinary group and explores the potential of using games as adolescent mental health interventions. The creators of PlayWrite interviewed MHC professionals to determine their technical expertise and shaped their design of PlayWrite to reflect that. After development of PlayWrite, the system was made available to MHC professionals for four months and game building workshops were held during this time. Reception of the system was largely positive, and the ease of use and flexibility of the program were the strongest aspects. The games produced by these workshops are currently undergoing formal clinical evaluations.

Chapter 3

Study Methods

The design and execution of this study was primarily the work of the occupational therapy student I collaborated with. My involvement is in the analysis and discussion of the results. However, the study is presented here to provide an understanding of the process and how we collected our results.

In order to understand how therapists describe common elements of therapeutic games, we first showed therapists several videos of existing stroke rehabilitation games depicting functionality likely to be needed in future therapeutic games. We then asked them to describe how to solve those problems using their own terms. We chose to present the game functionality using videos to minimize verbal cues that might influence therapists' descriptions.

3.1 Game Component Videos

Based on an analysis of the components that make up a set of games designed for stroke rehabilitation [1], we created 14 video clips. Each video clip illustrates an event or behavior that occurred in a stroke rehabilitation game. These clips can be separated into two categories: 1) relating player motion and object behavior and 2) object motions and behaviors in the game world.

Figure 3.1: "Relating Player Motion and Object Behavior" Screenshot

3.1.1 Relating Player Motion and Object Behavior

These videos were shown to illustrate the relationship between a player's motion and the effect of that motion in the game world. The videos were two synchronized clips of the player moving and the game world reacting (see Figure 3.1). The following videos show different potential relationships between the player motions and behaviors of game world objects:

1. A player controls the position of a baseball glove in a catching game by moving her hand in space.
2. A player raises her shoulder above a certain height to cause a frog to "jump" from one road lane to the one above it.
3. A player rotates her wrist to cause a frog to move right and left along a horizontal road.
4. A player picks up a yellow beanbag (representing dynamite in the game world) and places it down on top of a virtual weed.

5. A player bends her elbow up and down to control the height of a helicopter flying through a city.
6. A player holds a WiiMote like a joystick and points the WiiMote in different directions to control where a fish swims.
7. A player raises and lowers her shoulder to control the height of a snail and defend against a predator fish.

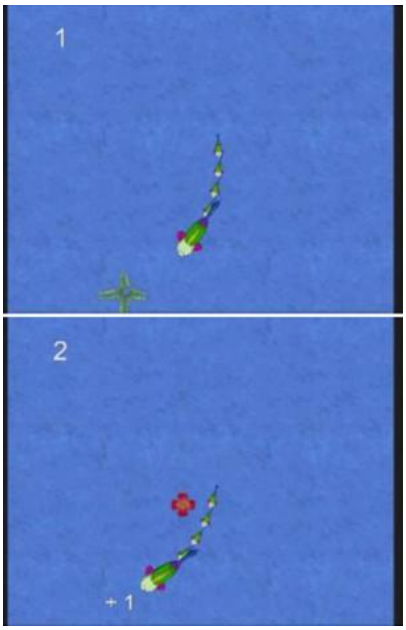

Figure 3.2: "*Object Motions and Behaviors in the Game World*" Screenshots

3.1.2 Object Motions and Behaviors in the Game World

The following videos illustrate a behavior involving one or more objects within the game world. These behaviors include motions of individual objects and groups of objects, collisions, earning points, and conditional actions (see Figure 3.2).

1. A mother fish swims around an ocean scene. Baby fish follow in a line behind her.
2. A helicopter flies through a city. The video includes scenarios illustrating correct behavior (stopping when in front of a building) and incorrect behavior (flying through the building).
3. A video of rotors spinning on a helicopter.
4. A helicopter flies into a target package. The player earns points.

5. A pitcher raises his arm behind his head and throws a baseball.
6. A baseball glove catches a baseball and the player earns points. Then, the baseball glove catches a basketball and the player loses points.
7. A mother fish eats a fern and earns one point. Then, the mother fish eats a flower and earns five points.

For some videos, we found it necessary to direct the participant's attention to particular aspects of the video clip. For the video involving a baseball glove catching both baseballs and basketballs, we were primarily interested in how the therapists choose to discern between the effects of catching the different balls. We were not interested in the motions of the glove or balls on the screen. For these videos, we used intentionally minimal text and colored boxes around relevant objects to direct attention on these actions.

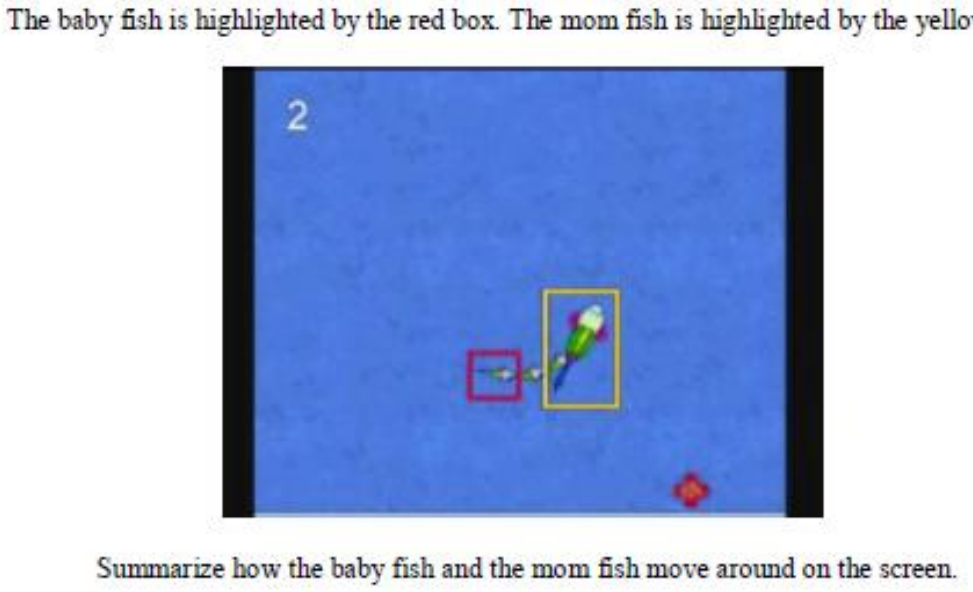

Figure 3.3: Highlighted Focus with Minimal Text

3.2 Participants

The participants involved were ten licensed therapists (eight occupational therapists and two physical therapists) from the Washington University School of Medicine and The Rehabilitation Institute in St. Louis. All ten therapists had experience working with stroke patients. At the time of the study, four of the ten therapists were working directly with persons with stroke in their clinical practices. The other six regularly interact with stroke survivors research studies. All of therapists were under the age of 50. None had prior programming experience.

3.3 Session Procedure

During each session, an experimenter showed the participant the 14 video clips. We randomized the presentation order for each participant to prevent any bias or influence from ordering. Each video clip included minimal textual instructions directing participants to describe how a computer should perform the actions depicted in the clip. We gave participants blank paper on which to write or draw their answers, but most participants gave their answers verbally. If a participant did not fully address all of the actions shown in a given clip, the experimenter prompted the participant for more information using a standardized set of prompts. All of the sessions were videotaped.

3.4 Analysis

We transcribed all of the sessions verbatim, including speech disfluencies and space-fillers.
We performed the analysis in two steps: 1) developing the rating form and 2) classifying participants' game functionality descriptions.

3.4.1 Developing the Rating Form

The rating form includes three basic types of questions: 1) questions designed to examine the similarity between how therapists and other novice programming groups describe programming scenarios and 2) questions designed to help formalize interesting properties in

therapists' descriptions and 3) questions focusing on the relationships between player motions and game actions.

Mentioned in Related Work, [27] performed a study similar to ours in which they asked children and adults to describe events and behaviors in scenes from a video game. We included questions in our rating form similar to those used by [27] to enable a comparison between how therapists, children, and adult non-programmers describe programming solutions.

We also added questions that focused on particular aspects of how therapists describe the videos. These additional questions were spurred by patterns and themes noted when the raters reviewed the therapists' answers. Even more additional questions were included in the rating form to examine the relationship between players' motions and corresponding changes in the game world.

3.4.2 Classifying Game Functionality Descriptions

To ensure rating consistency, we used a two step rating process. First, two raters independently rated each item for two randomly selected participants. The two raters then met and discussed the ratings for these two transcripts to ensure a consistent approach. Following this meeting, the two raters independently rated the remainder of the transcripts.
For all the therapist statements, each rater first broke the statement into individual programming statements (i.e. the shortest statement that expresses a single programming idea). Then, for each question on the rating form, each rater classified the statement to one of the categories suggested by that question (see Figure 3.4 for an example question and its categories). If a statement was deemed to not fit well into any of the suggested categories, the raters were instructed to mark it as "*Other*."

Because the transcripts are a verbatim record of participants' responses, they contain incomplete statements, self corrections and other inconsistencies. Each rater independently determined how to divide a given transcript into programming statements. Due to the variability in how the two raters defined programming statements, we cannot directly calculate an inter-rater reliability. Instead, we calculated the correlation between the percentages of statements in each item category for each of the ten participants. The correlations for each item are shown in the Results section in Table 4.1.

1. Please count the number of times the therapist uses these various methods to assign point values.
   a. 3___ 9___ 11___
      Rule-based, generalized
      Example: Red flowers are worth five points.

   b. 3___ 9___ 11___
      Explicit describing possession by player/avatar- "get(s)", "gain(s)", "lose(s)", "lost", "pick(s) up"
      Example: You get five points.

   c. 3___ 9___ 11___
      Explicit describing possession by score- "Score increased", "Score goes down", "Changing the score", "Subtracts 2 points off score", "Added to the total score"
      Example: The score goes higher.

   d. 3___ 9___ 11___
      Explicit with no possession implied- "Adds a point", "Jumped 10 points", "Added to the previous number", "Deduct a point", "Increase in points", "15 goes to 25"
      Example: The 15 goes to 25 when the helicopter hits the target.

   e. 3___ 9___ 11___
      Other (please specify) _____

Figure 3.4: Example Rating Form Question

Chapter 4

Study Results

From our results, we observed several patterns that we refined into guidelines for developing a therapist friendly programming environment. Our guidelines are:

1. Use event-based language to describe player and object motion.
2. Use rule based language to describe object behaviors.
3. The controller should disappear.
4. Use an object oriented paradigm.
5. Provide automatic high-level behaviors.
6. Use possession language for achieving objectives.
7. Use therapy terminology.

We organized our discussion of results by these guidelines along with their relevant questions.

Table 4.1 Categorization of Study Results

<table>
<tr><td colspan="3">Structure of Programming Statements</td></tr>
<tr><td>Programming Style (r=.94)

32% Constraint-based
29% Observational
24% Event-based
9% Instructional
3% Other
1% Imperative</td><td>Collision Avoidance (r=.73)

64% Event-based from perspective of moving object
16% Constraint from perspective of stationary object
12% Other
8% Constraint on both objects</td><td>Recipients of Rules (r=.61)

50% Program
47% Player
2% Other
1% Self
Cause and Effect Order (r=.93)

59% Cause then effect
41% Effect then cause</td></tr>
<tr><td>Discrete 1-Dimensional Motion (r=.52)

43% Event-based
34% Constraint
21% Negative statement
3% Other</td><td>Continuous 1-Dimensional Motion (r=.77)

44% Complete example, 2 statements
24% Complete example, 1 statement
20% Inclusive term
7% Incomplete example
5% Other</td><td>Continuous 2-Dimensional Motion (r=.83)

32% Incomplete example
27% Inclusive term
19% Controller relationship
10% Implied causation
8% Complete example
3% Other</td></tr>
<tr><td colspan="3">Subjects of Programming Statements</td></tr>
<tr><td>Perspectives (r=.93)

50% Object
28% Player
10% Programmer
12% Other</td><td>Relating Player and Object Motion (r=.94)

39% Third person - player
27% Body part
24% Second person - player
9% Controller</td><td>Avatar/Controller Relationship (r=.98)

73% Implicit body part
23% Implicit
4% Explicit

Use of "They" (r=.94)

63% Avatar
13 % Other objects in the scene
12% Player
11% Computer ("hidden authority")</td></tr>
<tr><td colspan="2">Game Event Descriptions</td><td>Language Choices</td></tr>
<tr><td>Collisions (r=.98)

69% Possession
17% Contact
14% Position</td><td>Assigning Point Values (r=.98)

51% Possession by avatar/player
24% Possession by score
14% Explicit with no possession implied
11% Constraint</td><td>Terminology (r=.87)

71% Event-based
25% Observational
4% Instructional

Motion (r=.98)

51% Common language
49% Therapy language</td></tr>
</table>

* all correlations are significant at the p<.01 level

4.1 Use Event-Based Language to Describe Player and Object Motion

4.1.1 Programming Style

For each statement, we looked at which one of four categories did it classify into. Most therapists used a constraint or rule-based style of programming language that holds true for the program duration. For example, "*He gets points for eating the fern.*" The second most common style of programming was observational which only described what was presently occurring on the screen: "*He's got it on his outer shoulder.*" Closely following the second was production rules or event-based where operations are preceded by "*when,*" "*if,*" "after," etc such as, "*As your arm goes up, the snail goes up.*" The other two categories were instructional and imperative, exampled by "*The helicopter should touch the top part of the building,*" and "*Flexes his right shoulder and extends his back and neck,*" respectively.

4.1.2 Cause and Effect Order

For each event statement, we examined the order of cause and effect. The results were closely divided with 18% more statements with cause before the effect such as, "*When you supinate, the fish goes left.*" The opposite choice would be, "*The snail moves up when the arm moves up.*"

4.1.3 Discrete 1-Dimensional Motion

We wanted to know how therapists described discrete 1-dimensional motion (object on screen moves when the patient reaches a threshold in movement). Most therapists used event-based structure such as "*If you raise your arm, the frog moves forward.*" Another category was using a constraint to describe the motion using "*each time*" or "*every time,*" such as, "*Each time he completes full shoulder flexion, it looks like it's bouncing up into the next zone of traffic.*" Interestingly, many therapists also included a negative rule in the description such as, "*He's only going up a level, he doesn't come back down.*"

4.1.4 Continuous 1-Dimensional Motion

This question asked how therapists described continuous 1-dimensional motion (object on screen follows the motion of the player in only one plane). Therapists primarily used two separate statements to describe the motion. For example, "*If you raise up, the snail moves up and as you lower the snail moves down.*" The second most popular choice used a complete single statement, i.e., "*Shoulder flexion makes the snail move up towards the fish and extension comes down*." Some therapists used an inclusive term for description such as "*following,*" "*in conjunction with,*" or "*corresponds with.*" An example would be, "*You have the snail on the screen corresponds with moving your shoulder up and down.*" The final category used an incomplete example in which only one of two directions was stated: "*As your arm goes up, the snail goes up.*"

4.1.5 Continuous 2-Dimensional Motion

This question focused on the ways therapists described continuous 2-dimensional motion (object on screen follows the motion of the player in two planes). Most therapists stated the relationship of the patient with the on-screen object using an incomplete example. For example, "*Moving your hand up will cause the glove to go up,*" specifies only upwards movement but implies a similar downwards movement of the glove. Some also included a term inclusive of all directions, such as "*following*" or "*mirroring*" in statements like, "*The claw is moving according to the motion that the person is doing with his hand.*" To a lesser degree, therapists used the controller to describe motion: "*The fish turns as the WiiMote turns.*" Implied causation was another less popular option, an example being, "*The baseball glove is going in a circular motion because the person is moving his hand in a circular motion.*" A complete, all directions inclusive example was used the least by therapists: "*When you pronate, the fish goes right. And when you supinate, the fish goes left. When it goes up and down, the fish goes up or down.*"

For all the different motions, the therapists used event based and cause-effect structure in their descriptions. These results are similar to the results found in [27] and in [25]. Both this and those studies involved scenes with games so it is appropriate that these event driven games would have event and rule based language defining them. This event based programming paradigm can be found in the HANDS system from the [25] and Visual Basic language. In this type of system, asynchronous actions from outside the program are

dispatched to the appropriate code. The link between an event being received and its redirection to the handlers is hidden from the programmer and is handled by the language. This approach fits these observations as the therapists assumed that the computer would redirect events appropriately and that the only part that concerned them was the implementation of actions upon event signal.

4.2 Use Rule-Based Language to Describe Object Behaviors

4.2.1 Collision Avoidance

This question examined how therapists used rules to describe collision avoidance between objects in the game. 64% of rules were clearly described from the perspective of the object in motion before the collision: "*When the helicopter comes to a building, it should stop right at the edge of the building.*" A much smaller percent, 16%, were from the perspective of the stationary object before the collision: "*The helicopter should be stopped by the building.*" Another option was a rule that places a clear constraint on both entities involved in one collision, such as "*You shouldn't see any helicopter in the grey*." The least popular option was a generalized constraint rule that handles all situations during the program: "*The helicopter should not go into any building.*"

4.2.2 Recipients of Rules

When therapists stated rules, we classified the statement by to what they were directed. When the therapists did include a rule-based statement, they were usually directed to either the player or the program. An example program-directed rule would be, "*The helicopter should be stopped by the building.*" The other option was to direct the rule to the self, as in "*The yellow would mean more caution to me.*"

For object behaviors, most therapists used rule-based language. They expected that once they declared a rule statement, the program would then constrain object behaviors appropriately. Terms such as "*should*" or "*needs to*" were typically used to describe such actions. The therapists' use of rules and constraints is similar to how behaviors are governed

in StarLogo. A programmer can write simple rules for objects and then observe how patterns arise from interactions [29].

4.3 Controller Should Disappear

4.3.1 Relating Player and Object Motion

This question categorized how therapists used various perspectives when describing motion. When relating patient motions to on screen objects, most therapists described the motions from the perspective of the second person player: "*It's moving the same exact way that his hand is moving.*" Many therapists also described the motion from the perspective of the body: "*So I guess the helicopter is governed by the height of, or the flexion and extension of the elbow.*" Less common were descriptions of motion from the second person perspective, "*Moving your hand up will cause the glove to go up*," and perspective from the controller, "*Fish turns as the WiiMote turns.*"

4.3.2 Avatar/Controller Relationship

When the therapist assigned control of the avatar to some external relationship, we categorized these in different relationship types. Several of the games required the therapists to clarify the relationship between the controller (i.e. beanbag, WiiMote) and the on screen objects or avatars. Most therapists noted this relationship implicitly as the body part doing the movement. For example, in one video, the WiiMote is held in the hand and the therapist's statement was, "*The frog moves side to side with pronation and supination of the forearm.*" A slightly different option was implicit assignment, but not to a body part: "*The dynamite moves when the beanbag is picked up.*" Explicit assignment, e.g. "*The dynamite is the yellow beanbag,*" was hardly used.

The player, object, avatar, controller relationships imply that the therapist sees the connection as being from the object/avatar to the player's body part, skipping mention of the controller. This absence of the controller implies an abstraction of code needed for movement. While from the example it should explicitly be, "*The frog moves side to side with the movement of the WiiMote which is moved by pronation and supination of the forearm,*" the therapist hides the information of the WiiMote as middleware between movement and game action.

This is similar to the programmer's use of the turtle in Logo [9]. Instructions are given to the turtle to achieve programming goals such as drawing graphical shapes. While in a general programming language this would be programmed by explicitly changing pixel colors along a line equation, this code is abstracted through providing instructions for turtle movement. In our observations, the therapist only handles the movement of the body part, assuming that the WiiMote will direct the movement to that of the game object. Since the controller will need to be explicitly assigned at some point in the programming process, a prompt to clarify the controller relationship may be necessary.

4.4 Use Object Oriented Paradigm

4.4.1 Perspectives

For each statement, we wanted to know what type of perspective it used. There was a clear use of object perspective throughout the therapists' responses with the player perspective appearing about a quarter of the time. An object perspective statement would be, "*The snail moves up and down.*" A player perspective statement would be, "*To get bonus points, you need to win the game within two minutes.*" Very few included statements from the perspective of the programmer, e.g. "*If the game is won within two minutes, give bonus points to the player.*"

4.4.2 Use of "They"

Each time the term "*they*" was used in a therapist's response, we categorized the usage by what "*they*" was used to describe. The term "*they*" was primarily used by therapists to describe the avatar or object on screen in the game: "*As they turn, they kind of lag behind.*" The other choices were relatively equally distributed as other objects in the scene: "*Basically they [the computer opponents] get points if you catch heavier balls,*" as the player: "*They gain points for catching the baseball,*" and as the computer: "*They deduct a point when you go for the basketball.*"

The perspectives and the usage of "*they*" suggest that the therapists see the actions as being done by the objects. This trait is similar to that of object-oriented programming, where objects execute their own functions and methods.

4.5 Provide Automatic High-Level Behaviors

While no specific questions dealt with this observation, this was a trend we noticed amongst all the therapist statements.

The nature of therapists to use implicit wording suggests that automatic high-level behaviors are necessary to avoid programming omissions. For example, for continuous 2-dimensional motion, the therapists used an incomplete example: "*Moving your hand up will cause the glove to go up.*" They simply assumed that the relationship would hold true for all directions. This illustrates a natural language programming tendency in the therapists. From previous studies, "nonprogrammers omit many actions from natural language problem solutions thus relying on a human-like interpreter to fill in the missing details" [26]. Providing support in these areas, such as a layer of interpretation between their programming and compiled code, will help avoid unnecessary programming errors and frustration.

4.6 Use Possession Language for Achieving Objectives

4.6.1 Collisions

This question asked how the therapists described collisions between objects in the games. We found that most therapists used possessive terms such as "*gets*" or "*catches*" such as in, "*When the helicopter gets the brick, the score goes up.*" Other collision descriptions were either contact based such as, "*As soon as she touches the plant or flower*" or position based, "*The fish goes over the plant.*"

4.6.2 Assigning Point Values

Points were assigned using a similar possessive language. Either the score itself or the player/avatar "*gets points*" when the correct event occurs. If a therapist used statements such as, "*Jumped ten points,*" "*Added to the previous number,*" that would be classified as explicit with

no possession implied. The least used category was point assignment by constraints, demonstrated by, "*Red flowers are worth five points.*"

This possessive-type language for objects lends support for including the score as a characteristic or underlying component of the player. This additionally lends support for object-oriented programming as these are similar to the properties and methods that objects possess.

4.7 Use Therapy Terminology

4.7.1 Terminology

Each time the therapists described events, the event was labeled as one of three categories. 70% of the statements were event-based. For example, "*So when the helicopter his that black box it gets points.*" This contrasts with only 25% as observational statements, which described only the event that was currently occurring: "*The baseball glove is just going exactly where the pink glove goes.*" The remaining category was instructional, e.g. "*Move your shoulder up and down and as you do, the snail will do the same thing.*"

4.7.2 Motion

When therapists described body motion, we classified it as either in therapy-related terms (e.g. flexion, abduction) or in common language terms (i.e. move the shoulder up). Surprisingly, the therapists used an equal number of therapy-related terms and common language terms. Some therapists used either set of terms exclusively while others used therapy-related terms mixed with common language terms.

The use of therapy terminology in their programming reflects that there should be a match between the system and the real world [26]. This reinforces that "the system should speak the user's language, with words, phrases, and concepts familiar to the user, rather than system-oriented terms" [26]. By using therapist terminology, they demonstrate a DSL trait of their programming. This implies that constructing a DSL would be more appropriate for therapist understanding of programming than using a general purpose language.

Chapter 5

Implementation

For developing a game authoring environment, we chose to extend the Looking Glass IDE. Rather than create a new programming tool, we decided to leverage the existing components of the program and build on the already existent ability to easily create programs.

5.1 Introduction to Looking Glass

Looking Glass is a programming environment designed to enable middle school children to create 3-dimensional animated stories [17]. Looking Glass is based on an existing programming environment, Storytelling Alice, which was designed to provide a motivating introduction to computer programming for middle school children [16]. Looking Glass differs from Storytelling Alice by incorporating additional tools designed to enable children to teach themselves new programming skills.

In the IDE, the Action Editor is where the user can edit their program and modify its functionality. The 3D Scene is a visual representation of the objects in the program. Users are able to view the Actions, Questions, and Properties of an object in the left-side pane. These function similar to the methods, functions, and properties of an object in programming language. Tiles from this pane can be dragged and dropped into the Action Editor to add additional lines of code. The Control Flow allows for a user to drag in framework for conditional statements, iterative loops, and other similar programming constructs. When the user wishes to run their program, they can click the Play button located on the 3D Scene window.

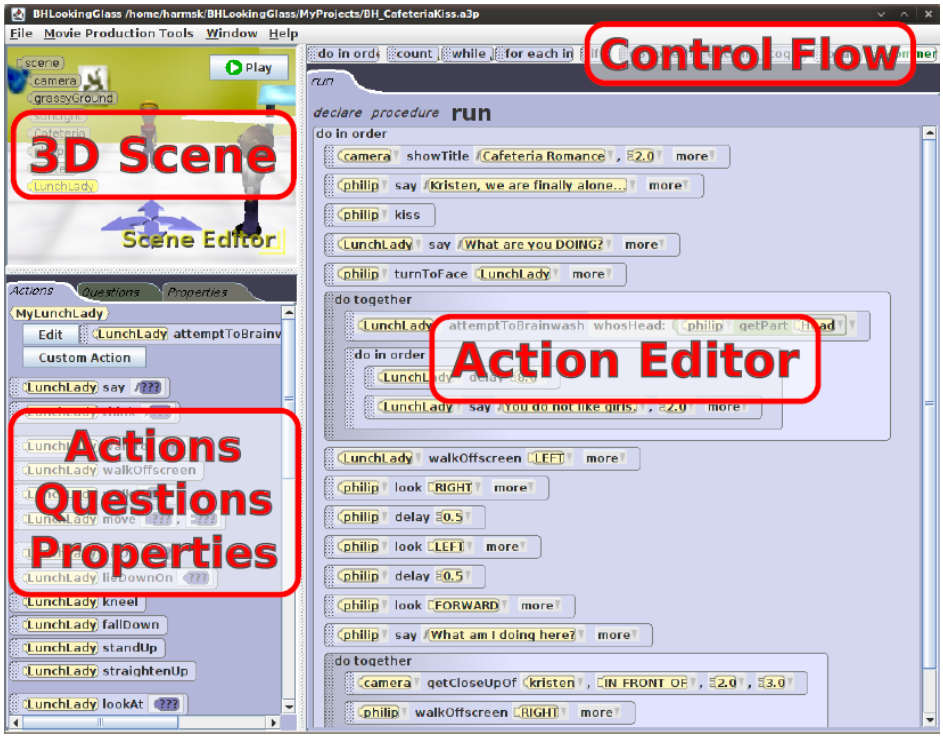

Figure 5.1: Looking Glass IDE

5.2 Proof of Concept

Looking Glass does not natively have support for motion based input devices in the IDE. The only supports for capturing user input were keyboard event listeners and mouse event listeners. I utilized the existing WiiRemoteJ library and libraries that others in the lab developed for WiiMote and webcam input [1]. I modeled these new event listeners similar after the keyboard listeners and mouse listeners that were already in Looking Glass. I then added support for multiple types of input for the WiiMote and webcam (Table 5.1).

Table 5.1: Input Device Capabilities

	Horizontal 1D	Vertical 1D	Tilt Direction, Magnitude	Positional 2D	Button Press, Release
WiiMote	x	x	x		x
Webcam	x	x		x	

The ability to add and utilize the different game inputs devices demonstrated a successful proof of concept for motion games developed in the Looking Glass IDE.

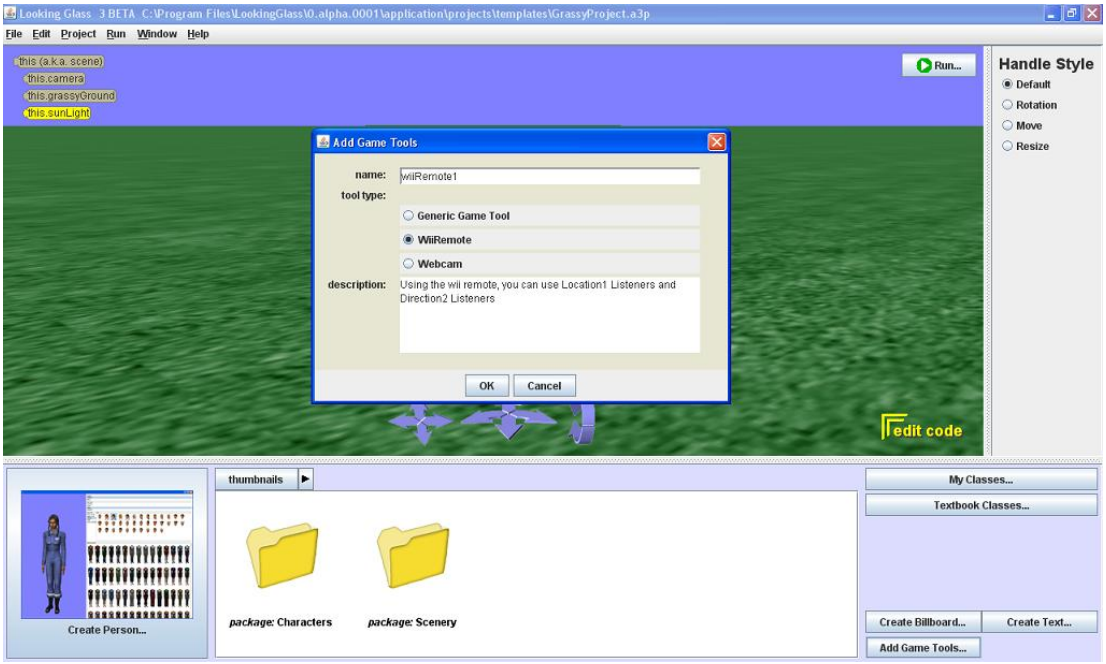

Figure 5.2: Adding a WiiMote

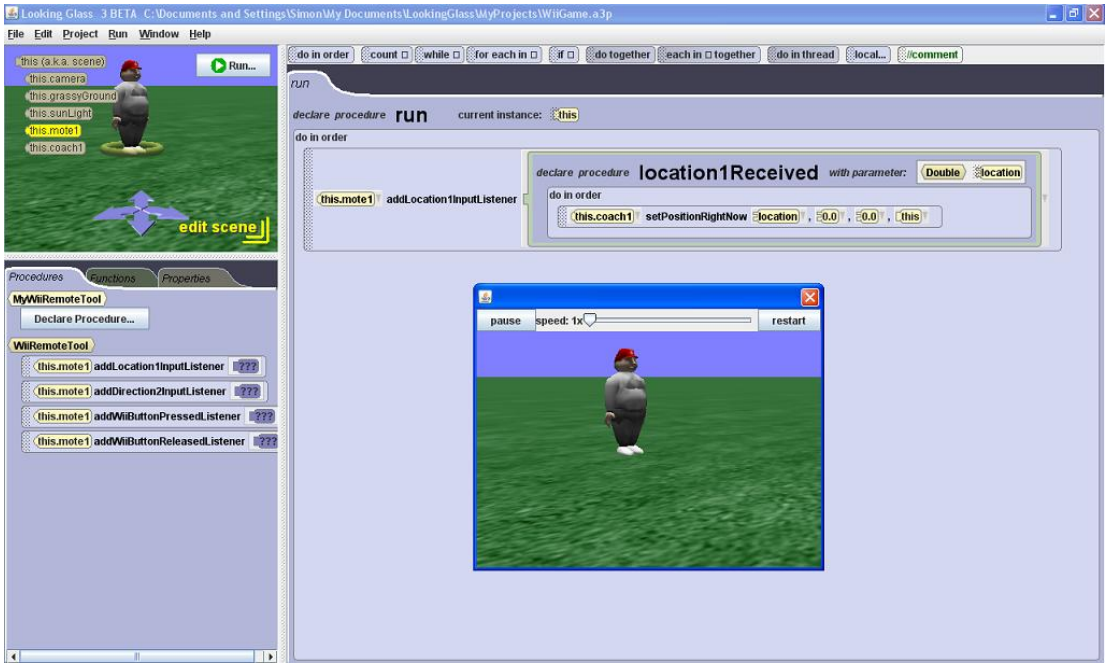

Figure 5.3: WiiMote Code Execution

Chapter 6

Tailoring Design

While Looking Glass is an environment designed for ease of use when creating virtual worlds, the program is designed for middle school students and not therapists. From looking at Related Work we concluded that Looking Glass needed to be tailored to the thought process and language of therapists in order to leverage the benefits of a DSL. We used the lessons from our study to drive the design process. While we did not cover all aspects of redesigning the IDE, the major design decisions shall be presented below.

6.1 Major Design Decisions

6.1.1 Player as an Object

From the study, therapists often referred to the player in third-person or second-person perspective. From their description of score and from other answers, they treated the player as an object in which they could manipulate the score and perform actions. Because of this, we added the player as an object that could be inserted into the program. This allows for games with multiple players.

6.1.2 Addition of Event-Based Methods

Previously the only tabs available for objects were their Actions, Questions, and Properties. Because the game videos were described with primarily event-based language, we added in another tab that handled an object's Events. The events that the therapists described for a player were the motion events. A full list of the player object's events is shown in Figure 6.1.

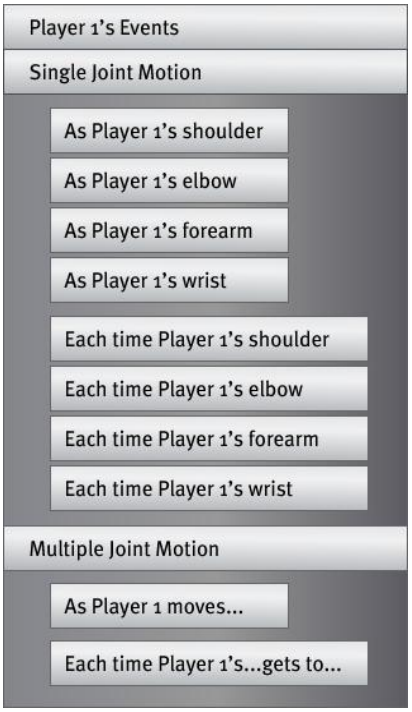

Figure 6.1: Player's Event Tab

6.1.3 Removal of Controllers

One of the results we concluded from the study was that the therapists rarely mentioned the controller and assumed that the player movements would automatically correspond with the objects in the program. Our authoring environment follows suit and has no mention of game input devices or event listeners when programming the game. Instead, the IDE relies on a layer of interpretation code that scans the programmer code for any event methods and dynamically allocates the appropriate input devices and event listeners according to the type of motion and body part that was specified in the line of code. For example, if the therapist only had one event method in the game that had the helicopter move up and down with the movement of the player's shoulder, the interpretation code would create a WiiMote object, attach a vertical 1-dimensional listener, and change the helicopter's y-position as the listener's output changes. The only time a game controller would appear would be when running the program, when the player connects and calibrates their controller.

6.1.4 Natural Language Wording

In the therapists' statements, they used natural language, i.e. plain English statements, as opposed to writing pseudocode or drawing pictures. This suggests a preference to see the programming language to be as close to natural language as possible. In addition, the terminology used for describing motions is done with therapeutic language, as that was another result from our study. If a programmer wished to have the helicopter move up and down as the player moves their shoulder up and down, the line of code that corresponds would be Figure 6.2.

Figure 6.2: Natural Language Programming

6.1.5 Events Tab

In our game authoring environment the therapists emphasized events as the primary structure for their programming. We created an Events tab in the Actions Editor to clearly specify where event methods would be placed. These events are organized in sections by the object that the event is acted upon. These sections are collapsible and expandable for further organization and focus of code. New event sections are created as events from new objects are dragged and dropped in. There is an additional tab, the Game Intro tab, which provides a place for programmers to place code that occurs at the beginning of game execution. This is demonstrated in Figure 6.3.

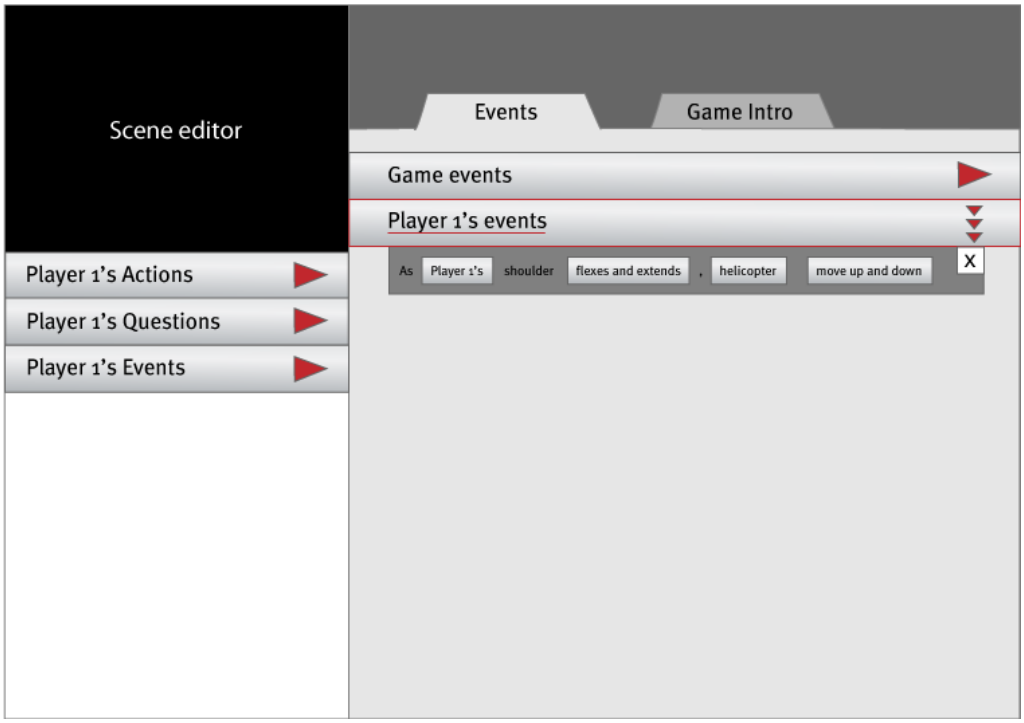

Figure 6.3: Proposed Game Authoring IDE

6.1.6 Collisions as Possessive Language

Another observation pulled from the therapist results was the use of possessive language to describe collisions. The use of verbs such as, "*catches*" or "*eats*" was employed the most. We simplified these verbs to "*gets*" and used that to show collision. We also structured these methods by using the event-based style mentioned before. An example of this is Figure 6.4.

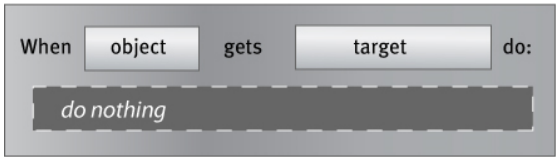

Figure 6.4: Collision Method

6.2 Design Limitations

While we propose many changes to the Looking Glass IDE to adapt for a stroke game authoring environment, there are many aspects of games that we did not account for and will require additional studies to observe how therapists describe those. Subjects such as game world set up and camera motion are among the ones that require further studies.

Chapter 7

Conclusion

Games have been proven as an effective tool for stroke rehabilitation. Currently, therapists lack the tools to provide these games for their patients. Through our project, we have observed how therapists would describe stroke rehabilitation games in their own language. We have proven that motion capture devices can be added into Looking Glass through the IDE. We have also begun work on tailoring the Looking Glass IDE to specifically fit therapists for game authoring. However, there is still work to be done in creating a complete environment. We have all the basic motions and collisions in games but through the study, we realized that there still an issue with setting up game worlds. This problem is especially significant when trying to set up semi-random game environments. We believe that there needs to be additional studies to resolve this. Also, while some of the tailored designs are implemented, the goal is for other members of our research lab to finish implementation this summer and begin testing with people in the therapy community.

Yet, the work and initial steps accomplished represent definite progress towards creating a therapy-centric authoring environment. The environment design's largest advantage is its ability to wrap traditionally complex programming behaviors (e.g. adding in motion listeners, correlating input to object properties), into very simple packages. We feel that these proposed constructs will decrease the number of lines of code necessary to create typical therapeutic programs. Additionally, by placing the tools directly into therapists' hands allows for the therapists to be the ones responsible for crafting game therapies. The communication and turnaround time issues between designing and game therapy and handing the details to a programmer have been negated through this direct process. This allows for efficient use of time in game authoring.

With this initial work and future plans, we feel that we can successfully create a programming environment to allow therapists to quickly and easily create customized, effective therapy games for persons with stroke.

References

[1] Alankus, G., Lazar, A., May, M., & Kelleher, C. (2010). Towards customizable games for stroke rehabilitation. *Proceedings of the 28th international conference extended abstracts on Human factors in computer systems,* 2113-2122.

[2] Angeleri, F., Angeleri, V.A., Foschi, N., Giaquinto, S., & Nolfe, G. (1993). The influence of depression, social activity, and family stress on functional outcome after stroke. *Stroke*, *24*, 1478 - 1483.

[3] American Heart Association. Heart Disease and Stroke Statistics – 2004 Update. Dallas, Texas: American Heart Association, 2003.

[4] Bach-y-Rita, P., Wood, S., Leder, R., Paredes, O., Bahr, D., Bach-y-Rita, E.W., & Murillo, N. (2002). Computer-Assisted Motivating Rehabilitation (CAMR) for Institutional, Home, and Educational Late Stroke Programs. *Topics in Stroke Rehabilitation*, *8*(4), 1-10.

[5] Burke, J., McNeill, M., Charles, D., Morrow, P., Crosbie, J., & McDonough, S. (2009). Optimising engagement for stroke rehabilitation using serious games. *The Visual Computer, 25*(12), 1085-1099.

[6] Center for Disease Control and Prevention. (2010, Feb 1). *Stroke - Home.* Retrieved from http://www.cdc.gov/Stroke/index.htm

[7] Colombo, R., Pisano, F., Mazzone, A., et al. (2007). Design strategies to improve patient motivation during robot-aided rehabilitation. *Journal of Neuroengineering and Rehabilitation 4(*3).

[8] Coyle, D., Doherty, G., & Sharry, J. (2010). PlayWrite: end-user adaptable games to support adolescent health. *Proceedings of the 28th international conference extended abstracts on Human factors in computer systems,* 3889-3894.

[9] Deutsch, J.E., Borbely, M., Filler, J., Huhn, K., & Guarrera-Bowlby, P. (2008). Use of a low-cost, commercially available gaming console (Wii) for rehabilitation of an adolescent with cerebral palsy. *Physical Therapy, 88*(10), 1196—1207.

[10]Emihovich, C. & Miller, G.E. (1988). Talking to the turtle: A discourse analysis of logo instruction. *Discourse Processes*, *11*(2), 183-201.

[11]Flynn, S., Palma, P., & Bender, A. (2007). Feasibility of Using the Sony PlayStation 2 Gaming Platform for an Individual Poststroke: A Case Report. *Journal of Neurologic Physical Therapy, 31*(4), 180-189.

[12]Huber, M., Rabin, B., Docan, C., et al. (2008). PlayStation 3-based tele-rehabilitation for children with hemiplegia. *Virtual Rehabilitation*, *2008*, 105-112.

[13]Jack, D., Boian, R., Merrians, A. S., Tremaine, M., Burdea, G. C., Adamovich, S. V., Recce, M., & Poizner, H. (2001). Virtual reality-enhanced stroke rehabilitation. *IEEE Transaction on Neural Systems and Rehabilitation Engineering, 9*(3), 308-318.

[14]Johnson, M.J., Feng, X., Johnson, L.M., & Winters, J.M. (2007). Potential for a suite of robot/computer-assisted motivating systems for personalized, home-based, stroke rehabilitation. *Journal of NeuroEngineering and Rehabilitation*, *4*(6)

[15]Jung, Y., Yeh, S., & Stewart, J. (2006). Tailoring virtual reality for stroke rehabilitation: a human factors design. *CHI '06 extended abstracts on Human factors in computing systems,* 929-934.

[16]Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys, 37(*2), 83-137.

[17]Kelleher, C., & Pausch, R. (2007). Using Storytelling to Motivate Programming. *Communications of the ACM, 50*(7), 58-64

[18]Kelly-Hayes, M., Beiser, A., Kase, C. S., Scaramucci, A., D'Agostino, R. B., & Wolf, P. A. (2003). The influence of gender and age on disability following ischemic stroke: The Framingham study. *Journal of Stroke and Cerebrovascular Diseases, 12*(3), 119-126.

[19]Kleim, J. A., Jones, T. A., & Schallert, T. (2003). Motor enrichment and the induction of plasticity before or after brain injury. *Neurochemical Research, 28*(11), 1757-1769.

[20]Lai, S. M., Studenski, S., Duncan, P. W., & Perera, S. (2002). Persisting consequences of stroke measured by the Stroke Impact Scale. *Stroke, 33,* 1840-1844.

[21]Lang, C.E., MacDonald, J.R., and Gnip, C. Counting repetitions: an observational study of outpatient therapy for people with hemiparesis post-stroke. *Journal Neuro. Phys. Therapy 31*, 1 (2007), 3-10.

[22]Langhorne, P., Coupar, F., and Pollock, A. Motor recovery after stroke: a systematic review. *The Lancet Neurology 8,* 8 (2009), 741-754.

[23]Lotze, M., & Cohen, L. G. (2006). Volition and imagery in neurorehabilitation. *Cognitive Behavioral Neurology, 19*, 135-140.

[24]Nudo, R. J. (1996). *Functional remodeling of motor cortex: Implications for stroke rehabilitation.* McDonnell Conference, Washington University, St. Louis, Missouri.

[25]Pane, J (2002). A programming system for children that is designed for usability. Ph.D. thesis. Carnegie Mellon University, Pittsburgh, PA; www.cs.cmu.edu/~pane/thesis/

[26]Pane, J., & Myers, B. (1996). Usability issues in the design of novice programming systems. *School of Computer Science Technical Reports*, Carnegie Mellon University, CMU-CS-96-132.

[27]Pane, J. F., Ratanamahatana C.A.N.N., & Myers, B.A. (2001). Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies*, *54*(2), 237-264.

[28]Mayer, R.E. (1979). A psychology of learning BASIC. *Communications of the ACM*, *22*(11), 589-593.

[29]Resnick, M. (1996). StarLogo: an environment for decentralized modeling and decentralized thinking. *Conference companion on Human factors in computing systems: common ground*, 11-12.

[30]Rizzo, A. & Kim, G.J. (2005). A SWOT Analysis of the Field of Virtual Reality Rehabilitation and Therapy. *Presence: Teleoperators & VE,* 14(2), 119-146.

[31]Rosamond, W., Flegal, K., Furie, K., Go, A., Greenlund, K., Haase, N., Hailpern, S. M., Ho, M., Howard, V., Kissela, B, Kittner, S., Lloyd-Jones, D., McDermott, M., Meigs, J., Moy, C., Nichol, G., O'Donnell, C., Roger, C., Sorlie, P., Steinberger, J., Thom, T., Wilson, M., Hong, Y. (2008). Heart disease and stroke statistics—2008 update: A report from the American Heart Association statistics committee and stroke statistics subcommittee. *Circulation, 117*, e26-e146.

[32]Saeki S. (2000). Disability management after stroke: Its medical aspects for workplace accommodation. Disability Rehabilitation, 22(13–14), 578 – 582.

[33]Sanchez, R., Jiayin Liu, Rao, S., et al. (2006). Automating Arm Movement Training Following Severe Stroke: Functional Exercises With Quantitative Feedback in a Gravity-Reduced Environment. *Neur. Sys. and Rehab. Engr., IEEE Tr.* 14(3), 378-389.

[34]Selzer, M., Clarke, S., Cohen, L., Duncan, P., and Gage, F. *Textbook of Neural Repair and Rehabilitation: Volume 2, Medical Neurorehabilitation*. Cambridge Uni. Press, 2006.

[35]Shaughnessy, M., Resnick, B.M., and Macko, R.F. Testing a model of post-stroke exercise behavior. *Rehabilitation Nursing: The Official Journal of the Assoc. of Rehabilitation. Nurses 31*, 1 (2006), 15-21.

[36]van Deurson, A., Klint, P., & Visser, J. (2000). Domain-specific languages: An annotated bibliography. *ACM SIGPLAN Notices*, *35*(6), 26-36.

[37]Wagner, J. M., Lang, C. E., Sahrmann, S. A., Edwards, D. F., & Dromerick, A. W. (2006). Sensorimotor impairments and reaching performance in subjects with poststroke hemiparesis during the first few months of recovery. *Physical Therapy, 87*(6), 751-765.

[38]Warlow, C., Sandercock, P., Hankey, G., et al. *Stroke: Practical Management*. Blackwell Publishing, 2008.

[39]Wolf, T., Baum, C., & Connor, L. (2009). The changing face of stroke: Implications for occupational therapy practice. *American Journal of Occupational Therapy*, *63*(5), 621-625.

[40]Yeh, S.C., Rizzo, A., Zhu, W., Stewart, J., McLaughlin, M., Cohen, I., Jung, Y., Peng W. (2005). An integrated system: virtual reality, haptics and modern sensing technique (VHS) for post-stroke rehabilitation. *Proceedings of the ACM symposium on Virtual reality software and technology*

Vita

Simon Tam

Date of Birth	September 16, 1987
Place of Birth	State College, Pennsylvania
Degrees	B.S., Computer Science, May 2010 M.S. Computer Science, May 2010