Explaining the difference between event-based and procedural programming to non-programmers.

Kendall Park
August 20, 2010

Abstract

Interactive therapy games remain an untapped yet promising resource for therapists. One of the greatest challenges in using games for therapy remains the vast array of unique needs ranging over the patients. Caitlin Keheller's Alice-based software offers to help bridge the gap between therapists and programmer so that therapists can create games uniquely tailored to each patient. There are many challenges to overcoming the coder to non-coder gap such as the distinction between programming topics such as event-based and procedural programming. This paper offers a qualitative analysis of our findings from four user tests and the measures we took to help distinguish the aforementioned topic.

Introduction

Wii-like interactive games have demonstrated great promise in the realm of rehabilitation, creating a fun, motivating environment for patients to perform therapy exercises.[i] Our target audience, stroke survivors, differ dramatically in rehabilitation needs, therefore a generic therapy game cannot hope to envelop such unique needs of stoke survivors. On the flip side, freelancing programmers to create individualized games consumes extensive amounts of time--and therefore money.

What are the alternatives? The average therapist lacks the programming knowledge to create robust rehabilitation games. It is this gap that the Stoke Therapy Games research project attempts to bridge. By creating a free, non-coder-friendly environment in which therapists can rapidly develop therapy games, the large time and monetary costs of game development can be cut. In the future, we hope to achieve an ease of development that allows occupational and physical therapists the freedom to quickly customize their games to the individual case of each stroke survivor.[ii]

The Stroke Therapy Games project functions as an extension of Caitlin Kelleher's Looking Glass, the successor of Storytelling Alice, which aims to teach programming to middle school aged children. Looking Glass creates an environment that prohibits syntax errors and constructs three dimensional representations of objects. A user of Looking Glass will use drag and drop features to “program” actions for their own three dimensional worlds in order to create a story.

Problem

One of the largest problems we encountered during the user test was explaining the difference between the abstract constructs of “actions” and “events.” In the Stroke

Therapy Games project, “actions” represent a singular occurrence of a specific behavior such as “moveForward” or “jump”. Actions are executed in a predetermined order, from the start of a game. Events consisted of an situation-reaction pair that could occur at any time during the game, when certain environmental factors match those of the situation. One could create an event that caused a rabbit character to turn green whenever it collided with a carrot during the game. In programming terms, events are analogous to adding a listener to a program, whereas actions are similar to statements in a method (such as the main method).

However, our target audience of stroke therapists do not posses the computer science background that recognizes the distinction between procedural and event-driven behavior. Finding a way to communicate the difference between those two abstract concepts drove much of user test-driven development in the teaching elements and interface design.

It is important to note that the original Looking Glass program does not draw a distinction between these two concepts. Because Looking Glass intends for the creation of stories as opposed to games, the majority of the user-programmed behavior is procedural. Games, on the other hand, are driven by nondeterministic interaction with the user and therefore require event-based programming. During preliminary user studies, therapists described games in terms of an reactions (“if this, then that”) more than in-order activities (“first this, then that”).[i]

Methodology

Eight graduate students in the Washington University School of Occupational and Physical Therapy were recruited to perform our user-tests (seven female and one male). In each three-hour session a pair of therapists were given a description of a fiction stroke victim as well as blank paper and asked to brainstorm a game for that particular victim. After they decided on the structure for a therapy game they were introduced to our program and given the tutorial (in later sessions they would also be lead through a Stencils-based tutorial). We tried to intervene as little as possible so as to observe the largest points of frustration. Each session was video-taped so we could further review the proceedings and also perform language analysis (what words they often used to describe aspects of the game or program). When the users had successfully created a game or sufficient time had passed, we asked the therapists to comment on their experience and offer suggestions for improvement to our program.

Exploring the Solutions

In our Stroke Therapy Games project, we identified two primary methods to aid users in discriminating between actions and events: the learning materials provided to the therapists with the program, and also the interface of the program itself--what is explained to the user, and what is self-explanatory. This section summarizes the iterative changes that took place as we continually adapted the software in response to the user-tests conducted. The data gathered was primarily observation and qualitative analysis of the users responses. Not enough user-tests were conducted to produce enough data for quantitative analysis, nor was program presented to the users identical between each study, as we were constantly developing and refining features based on the previous test.

Tutorial

Our first version of the tutorial consisted of a stapled packet of paper that led the user through the general process of creating a game through text and illustrations. The tutorial presented actions before events, with the rationale that actions are inherently simpler to comprehend than events and therefore should be introduced first.

In the first user-study, it became clear that introducing actions before events only caused confusion to the therapist as the user transitioned from actions to events. The tutorial was modified for the subsequent studies, drawing out the actions section of the tutorial and placing it in an appendix. It became clear that it was possible to create a game based entirely on events, and the procedural actions would only be used for setting up the initial conditions of the game.

A crucial aspect of explaining actions and events rested in the terminology used to describe events. We experimented with terms such as “action-reaction pairs,” “triggers and consequences,” or “situations and reactions.” We decided that “situations and reactions” best conveyed the setting up an event, but left the subject open to the results of our experimentation. (See figures 1 and 2 for further explanation.)

The major tutorial revision was made when we started switching over to a Stencils-based tutorial. The Stencils-based tutorials are an in game tutorial first engineered by Caitlin Kelleher to explain the concepts of Storytelling Alice to the user.[iii] Stencils is an virtual tutorial that creates an overlay over the program, guiding the user through a series of tasks used to introduce the program's features. Because the stroke therapy branch of Looking Glass required the use of many new interface features, the version of Stencils used with Looking Glass was not entirely compatible with our program. Some version of Stencils was used for the last three user-tests, slowly replacing the paper tutorial as we adapted more our program's features to support Stencils. By the last two tests, Stencils had become the primary source of instruction, with the paper tutorial used only as a reference if needed.

Interface

Deviating from the Looking Glass structure, we created a separate Events Tab next to the Run Tab in the user interface. This was where most of the programming would take place. Our biggest issue initially was drawing attention to this tab. When the program was loaded, the Run Tab, like in Looking Glass, loaded out as the primary open tab. Users had difficulty finding the Events-tab, even when aided by the tutorial.

We decided that the interface needed an overhaul to better support the new features such as the Events Tab. To explore new interface designs we created several paper versions of interface, and experimented with their usability. Paper prototyping offers an time-efficient, inexpensive, and tangible way to determine the practicality of different interface designs.[iv] From these prototypes we opted to distinguish the Events Tab by changing its color from that of the Run Tab. We also altered the interior to reserve a space for each in-game object's events. Actions and Events commands were also separated into their own private sections (as they used to be together).

From later tests we noticed that when the users added many objects to the game, the Events Tab would become unnecessarily crowded (as most of those objects did not have any events associated with them). In response, we changed the Events Tab so that it did not automatically include sections for each object's events but only added them when an object specific “Add [Object Name]'s Events” button was pressed. This cleared up the Events Tab so that the user would not get bogged down searched for a particular object's event section.

Conclusions

From our preliminary user tests, many alterations were made to increase the responsiveness of non-programmers to highly-abstracted concepts of programming. One of the major challenges was creating a distinction between event-based and procedural programming, something that coders, after years of experience, often take for granted. Our tutorial evolved from a paper-based instruction manual to an interactive Stencils-based tutorial, focused primarily on events instead of actions. The interface, especially the Events Tab, was modified to draw attention to differences between events and actions through color and layout. Because these were user-tests early in the decision, the program was rapidly altered from one test to another based on the response of each pair of therapists.

Future Work

My summer's research illustrates positive progress towards the use of video-games for stroke rehabilitation. Game-based therapy is a vast, virtually untapped resource for therapists, but bridging the gap between programmer and therapists offers the greatest challenges before widespread implementation of programs such as our Looking Glass-based game creation software. Further research topics could include the necessity of therapists as programmers, the effectiveness of 3D environments for stroke therapy games (as opposed to 2D environments), and the level of abstraction necessary to guise programming topics such as events and actions into modules understandable to the non-coder.

Acknowledgements

I would like to thank Tracy Camp and the CRA-W for this DREU experience, Dr. Caitlin Keheller for her mentoring, Matt May for being such a great partner, and the rest of my labmates for their support: Paul Gross, Kyle Harms, Jordana Hodges, Jennifer Yang.

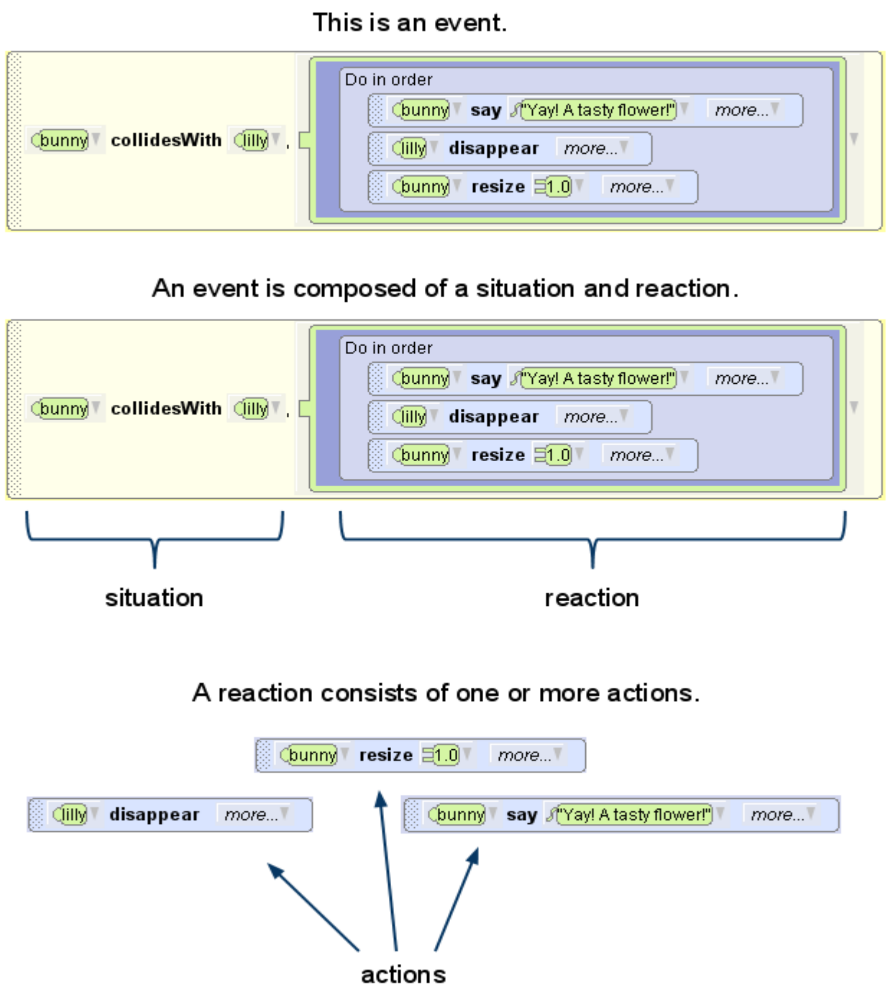

Figure 1. Composition of an event.

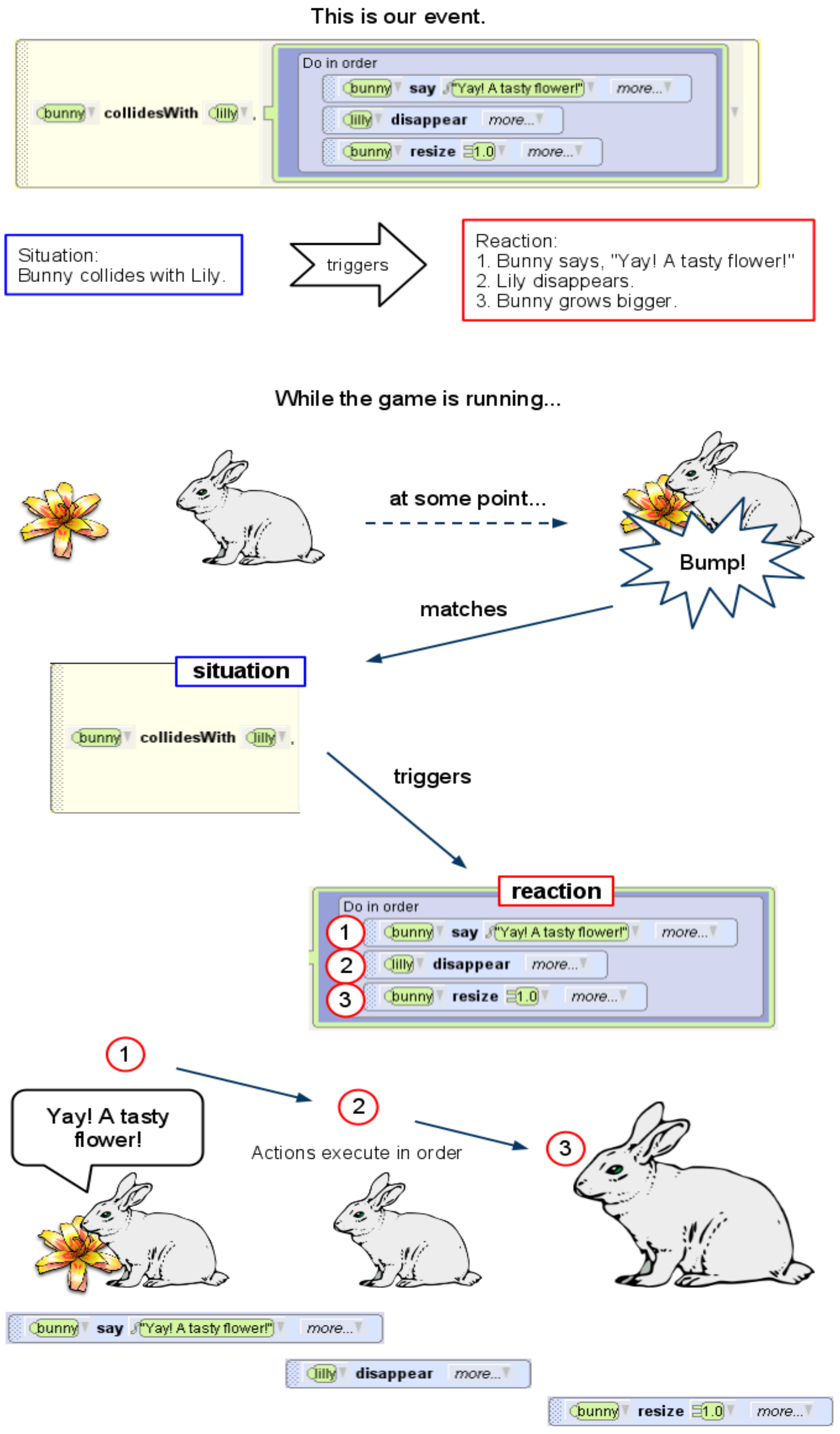

Figure 2. Situation and reaction.

i Alankus, G. R. Proffitt, J. Engsberg, C. Kelleher. Stroke Therapy through Motion-Based Games: A Case Study, Proceedings of ASSETS 2010

ii Alankus, G. M. May, A. Lazar, C. Kelleher. Towards Customizable Games for Stroke Rehabilitation. Proceedings of CHI 2010, pages 2113-2122

iii Kelleher, C. and R. Pausch. Stencils-based tutorials: design and evaluation. 2005 Conference on Human Factors in Computing Systems, pages 541-550

iv Snyde, C. Paper prototyping: The fast and easy way to design and refine user interfaces. 2003 Morgan Kaufmann Pub