Henry the Robot
Prototype Testing and Refinement

Team
Eric Krokos - Facilitator and Cameraman and provider of subjects
Patricia Sazama - Facilitator and provider of subjects
Walter Hutchinson - Observer and Facilitator and provider of subjects
Naveed Khan - Observer and provider of subjects

Problem and Solution Overview
The enrollment rate, along with the number of computer science courses offered in high school, has experienced a decline over the recent years [2]. A survey conducted in 2009 examined several high schools for the number of computer science courses offered by each school over four years. The results revealed that 78% of schools had an introductory course in computer science in 2005, while in 2009 the number had dropped to 65% [2]; leaving a large number of high school and college students, potentially interested in programming, with little or no opportunity for formal experience on the subject. Studies show that programming is notoriously difficult to teach [1]. Our objective is to remove the difficult syntax of programming languages and make simple and easy to understand goals. Our proposed solution is to create a simple programming environment with drag and drop functions and logic to solve various puzzles. These puzzles are AI oriented where the goal is to navigate a character to the goal using a set of given functions and logical commands. Over time the puzzles will grow in complexity, while introducing new commands and concepts to advance the user’s set of logical tools to solve problems.

Prototype Design
We used paper prototyping was used for the all testing purposes. We created a low-fidelity interface of our program using paper and pencils. The overall interface was sketched out displaying each section, button, panel, and command. Paper cutouts were used as clickable buttons and to represent any dynamic functionality. Static buttons were sketched out on our design.

![The program menu and control bar.](image)

The figure above shows the sketch of our menu bar which had all of the buttons that a user needed to interact with the program. All the buttons on this bar were sketched out and designed using affordances. The save and quit button on far left is self-explanatory. The curved arrow button encouraged return functionality. The box shape stop button, a common symbol in movie or music playing devices, was used to stop the program at any time during
execution. Another common symbol, the triangular shape button next to the stop button encourages play/execution action. It is used to run the program after the user has finished putting together all the commands. The sideways triangle with a bar is used for stepping through the program to examine what each command does. The toggle grid is used for enabling/disabling the grid and the familiar "x" button on the top left is used for closing the program.

The command list pane of the program. It hold all the commands a user can add to his program are stored.

The command list pane is placed right below the menu bar. The “move right” buttons were paper cut outs. The idea was to encourage action and show dynamic functionality. The pop up shown in the figure above is revealed after the user hovers over one of these buttons. Paper cutout was also used for displaying these popups. Clicking or dragging on the actual command moves it to the command list pane.

The program list and the worldview of the program.

The figure above shows the gaming/programming aspect of our program. The pane on the left is the user’s program. It has a list of commands that the user puts together. The puzzle like start piece encourages the user to place something under it. Clicking the one of the
commands from the command pane automatically places the clicked command under the last command in the program, locking in with it. The start piece was sketched on the paper to show that it is static and cannot be moved and as the commands are lined up below it, clicking the play button runs them in linear order. A cursor appears next to each command as it executes, to show the user what each command is doing in the context of the whole program.

The grid on the right is not interactive. The robot shown on the grid was also a paper cutout and responds to the commands that are placed on the left pane by the user. For example one “move right” in command pane and clicking the play button will make the robot move one block to the right on the grid. The treasure chest shows that there is an objective that the user needs to accomplishing by putting together the commands in the right order. The wall on the grid shows that there are hurdles that the user might need to overcome. The overall paper prototype gave a good understanding of our interface and the users were able to perform most of the required tasks fairly well.

Testing Method

We had four participants with codenames Justin, JC, Chris, and Lance in the video. Justin is a freshman political science major who we recruited through another friend. JC is the codename for Meg when is a sophomore psychology major who was also recruited through a friend. Chris is the codename for Remington who is a junior criminology/psychology double major and Lance was the codename for Evan who is a senior political science major. Remington and Evan we both recruited through friends as well.

We conducted our study with all four of our team members present. We had each participant come one at a time to ensure none of the trials were biased from seeing previous participants perform the tasks. They were seated at the table with the prototype, with one team member recording, one member observing, and two other team members answering questions and narrating the effects of each action the participant takes. Our prototype was all one screen with different elements that could be moved so we allowed the participant to move those elements freely and we moved other elements (like Henry, after the “play” button was pushed) after the participant did specific actions.

We created our three tasks with the intent to test the intuitivity of the interface and to cover the functions necessary to reaching the goals in our program. The first task was using the program control bar, such as starting, stopping, pausing, and stepping through a program. The second task was editing and creating your own program by clicking or dragging functions from the command list to the program list for a selected entity in the game world. The third task was to find out more information about a function that had not been seen before or that the user did not know the functionality of. By moving the cursor over the function, a pop up window that contains information about the function would appear.

Using the program control bar functionality was our easy task. When the program starts up for the first time, a sample tutorial program already completed is provided to you. The user, not knowing entirely how to interact with the product would see the familiar VCR-style control bar and press the play button.

The second task was creating and editing a program. To edit the program list, the user can either click and drag a function in and out of the program or command list at the top of the screen, or they can double click on a function in the command list to have it automatically go to the end of the program. As the user is editing the program list and using commands he will eventually generate a new program and learn how it works from how Henry moves when the user presses play.

As the user edits the program list, he will see functions that he has never seen before.
Rather than requiring the user to put the command in to see what it does, or just ignoring it, we wanted the user to hover over the command with the cursor and find out more information about it. When the mouse is over the function block for a given amount of time a pop up bubble will appear and given basic information about the function. Our goal was to make this very intuitive for a user, if they see an element in a program they are unsure about their instinct will be to point at it and pause while trying to decide what the element does, which will then cue the informative pop up.

After each participant sat down at the prototype we gave them a brief introduction to our testing procedure and the tasks. We instructed them to use their finger as the cursor and interact with the prototype as if it were a program on a computer screen, and to interact with the elements on it like they were pieces of the program.

We first gave them the simple instruction “run the program.” We asked them to think out-loud and narrate their thought process, explaining what was confusing or what seemed like the correct action to them. This enabled us to see common pitfalls users would fall into with our current prototype. If the participants had questions that were about how they could interact with the prototype that were unclear simply because it was a paper prototype (ie. “can I move this?”) we answered, but if they had questions about how to do a specific task we asked them how they thought they might do it. We stayed silent unless they specifically asked questions so that we didn’t bias their behaviors. After they completed this task we gave them the “success” screen and moved on.

After they completed the first task we gave them the second task which was simply “create a program that achieves the goal.” We wanted to see if they could figure out the goal of the program (getting Henry to the treasure chest) without being explicitly told what to do. We tried to give them very general instructions (ie. only calling it “the goal,” not “get Henry to the treasure chest.”). Once they completed that, we moved on to the next task.

We then finally gave them the third task which was “to learn more about a command.” At the beginning of this task we tried to reiterate that the participant’s finger was a cursor and they would have to interact with the prototype with that in mind, otherwise they might not intuitively hover over a feature when they weren’t sure what it does. After they completed this task we informed them they were done and thanked them for their time.

During the trials the observer noted when a participant failed to do a task because they tried to complete the task the wrong way. These will be used to determine unclear parts of the prototype design that we need to change to make the functionality more evident.

Testing Results

We were really surprised how the users interacted with our software and how they tried to do actions that either were not possible or did nothing. Overall the testers were able to figure out and solve most of the tasks, but it either took them many tries to get the task right or they got it right away, there seemed to be no middle ground.

The first task was having been given a program, run the program. The correct action was to press the arrow shape at the top of the program that looked like a VCR button. What we had not expected was on the program list, the entry point of the program or its “main” was called start, and they had started clicking it expecting to start the program. Only one person had gotten this right on the first try. However, once the tester had realized that the top bar was like a VCR they agreed on its simplicity and that they were just nervous from testing for the first time and not being familiar with the interface.
The second task was to have the user write a program, given a list of functions and commands, to solve the puzzle. This was probably the hardest task given. The first problem was that people did not understand where to write the script and how it interacted with the world to the right of the screen. Second, it was not obvious that you could click and drag the functions, for our test “Move Right” to the script. Instead the test subjects would repeatedly click “Move Right”, while it was in the command list bar, expecting the character in the world to move right. The problem was the user had forgotten they were writing a program instead of just making the character move to the goal.

The third task was to find out more information about a command. In programs such as Microsoft word or most commercial programs, you hover over a button to figure out more information about what that button does. We had done the same for our program, the idea was you would hover your cursor over the function such as “Move Right” and a little bubble would appear giving a short description of that command. This was not what happened during our testing, the users would keep clicking the function, expecting something to happen and were really lost on how to find out information. Some of them had also tried right clicking the button to get information on the function expecting a drop down menu to offer more options on what can be done with the command.

Another problem we found was that because we were using paper prototyping, our subjects didn’t intuitively interact with the prototype as if it was a computer screen. Some would pick up pieces that they could interact with, turning them over to check the back or other actions that someone wouldn’t do on a computer. Certain things that are natural when interacting with a computer (acts like hovering over an icon or dragging an object) weren’t as natural when participants used their fingers as a cursor on a paper prototype.

Interface Revisions

While the testers were interacting with our programs, we were keeping record of what problems they were having and in between testers we briefly discussed alternative ideas or changed for the interface.

For the first task, the goal was to start the a program given that it was written for you to familiarize yourself with the control bar. As a result of the failure of the test subjects to being able to start the program, we’ve decided to remove the entry point name from the script and leave it blank to keep users from clicking it. The reason being was that during the tests, many of the testers saw the word “start” and assumed that would start the program, where it was actually the starting point in the program they were going to write. Second, we made the top row of buttons more button like, rather than just appearing as static images. This behavior may have been caused by a flaw in the paper prototyping rather than in the design, because on the paper design, it was not very clear at showing what is a button and what is not.

The second task was to create or edit a program by moving commands around. The problem was, in the command bar, users kept clicking on the functions rather than clicking and dragging them. To correct for this, we are making it so the cursor will change to a hand icon when you hover over a function. If you double click the function, the function will automatically move to the current program and vise versa, if you double click the function in the program, it will move back up to the command list.

The third task was to find out more information about a function by hovering over it. We
observed that the tester was impatient and would just right click the function looking for help rather than hovering over it. We are going to add support for right clicking, where right clicking will give a list of options that you can do on that function or command, and one option will be to give help and information on that command.

Summary and Lessons Learned

Over the course of doing our paper prototype we learned many things, that many users are impatient on using a program and learning how to use it, although this could be partly due to the testing environment, and that the expected affordances that we thought were in place were not so obvious to others.

Programming often requires a lot of patience for organizing and developing your thoughts and then fixing and finding bugs in your code. The people we had tested did not have much patience for learning how to use an interface and using the program. For example, when we wanted the user to move a function, they would just double click it rather than clicking and dragging, which takes more time than just double clicking. Another example was when finding out more information about a command. We had thought that the user would hover their finger (cursor) over the function for a second or two and then we would give them the information bubble. Instead we found they wanted immediate feedback and information and would right click the function to try to speed things up.

We tried to organize the interface to make the expected use of it as obvious and intuitive as possible. This had failed primarily when the user had tried to start the program and again when moving functions around. The main problem with the start button was we had what looked like two start buttons rather than one. So we realized we have to make the interface very specific and make each function as independent from each other as possible. Lastly, we had put the functions in a list above the program list and figured that they would try to drag them to the editing program. Instead they had just tried to click them instead and we would have to say that that did nothing.

Video Report

The videotaping was done in an apartment with just the interviewer and the person filming. We tried to make the participant as comfortable as possible by offering refreshments and trying to calm them down before testing the prototype to alleviate any anxiety about being with new people in a new environment. While one person was testing we had the others waiting in the living room and keeping it so they could not hear or see what was going on with the current tester to reduce bias or prior knowledge.

The videotaping I felt was not disruptive and most of the testers were comfortable with filming. We had told them that the video was only going to be used for the purposes of this project and was not for any profit.

Being able to later observe the participants actions in video, rather than just by written observations, we were able to see how long it took the user to solve a task and to get a grasp of any stress or anxiety in their voice. Also, filming allows for minimal use of just writing down observations and allows for a unrestricted question and answering sessions after each session. We were able to write most of this paper by rewatching all of the interviews and listening to the testers feedback again rather than just using what we had written down during testing.

The main thing we learned about videotaping studies was to try to keep the camera out of the user's vision and to make it very passive. Some of the testers were a little anxious about being on film, just as most people are, and by not having the camera in their face it keeps their anxiety down and allows them to focus more on the tasks rather than "playing for the camera".
References


Consent for Videotaping

As part of a human computer interaction study, video surveillance is required.

I, ________________________________, hereby authorize the use of video surveillance for the purpose of “Henry The Robot” prototype testing.

Signature:

_________________________________________________________________________

Date:

_________________________________________________________________________
Consent for Video Taping

As part of a human computer interaction study, video surveillance is required.

I, Justin Dietrich, hereby authorize the use of video surveillance for the purpose of "Henry The Robot" prototype testing.

Signature: 

Date: 09/24/12
Consent for Video Taping

As part of a human computer interaction study, video surveillance is required.

I, [Participant Name], hereby authorize the use of video surveillance for the purpose of "Henry The Robot" prototype testing.

Signature: [Signature]

Date: 4/19/12
Consent for Video Taping

As part of a human computer interaction study, video surveillance is required.

I, __________, hereby authorize the use of video surveillance for the purpose of "Henry the Robot" prototype testing.

Signature: __________________________
Date: 4/22/12

Evan Gany
Consent for Video Taping

As part of a human computer interaction study, video surveillance is required.

I, ________________________________, hereby authorize the use of video surveillance for the purpose of "Henry The Robot" prototype testing.

Signature: _____________________________

Date: ________________________________

21 April 12