Storyboarding

Henry the Robot

Eric Krokos
Patricia Sazama
Naveed Khan
Walter Hutchinson
Henry the Robot

Team
Eric Krokos - Actor
Patricia Sazama - Hand Model
Naveed Khan - Stagehand
Walter Hutchinson - Camera man

Problem and Solution
The enrollment rate in computer science courses along with the number of computer science courses offered in high schools have been decreasing in recent years[2]. A poll conducted in 2005 showed that 78% of schools surveyed had an introductory course in computer science, while in 2009 the number had dropped to 65% [2]; leaving a large number of high school and college students interested in programming with little or no formal experience with 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.

Tasks
The three tasks we developed were to test the intuitivity of the interface and to cover the main aspects of creating a program in our software. The first task was using the program control bar, such as starting, stopping, pausing, and stepping through a program. The second task was being able to edit and create your own program by clicking or dragging functions from the command list to the active program list for a selected entity in the game world. The third task was to find out more information about a given 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.

From the start, we wanted to include a familiar program control bar that people without programming knowledge would understand. We used the classic VCR control bar, with the play, stop, pause, and step buttons so that it was clear without having to randomly click on them, what they did. Using this functionality was our easy task. When the program starts up for the first time, a sample tutorial program already completed is provided to you. We want the user to initially click the play button to see how all the world reacts to the given code on the left program list and we want the user to then play with and change the functions. The interface for this did not change from our last iteration we provided in the previous project.

After the user has run the sample program and seen the result, Henry the robot moved to the treasure chest, we wanted and expected them to try to change the functions in the command list by either adding, subtracting, or moving the order around. That is why 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 from the resulting actions when he clicked 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. This is very intuitive for a user because 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 our informative pop up. The functionality for this has not changed since our last project.

**Refined Interface Sketches**

Sketch of one of our concepts. In this case the user would drag functions on the left panel to create the program and the right side would display the actual syntax as the program executes line by line.
Brainstorming sketch of how the program could work. In this case commands are in a separate list, with an API-style description to help users learn.

Another brainstorming sketch, this one included only one button besides the list of drag and drop commands which was “GO” to run the program.
Another one of our sketches. This one used the drag and drop approach where the user would take the commands from the top and place them on the right panel in correct order to accomplish the goal defined on the grid. It also had the feature where “hovering” over a command would provide additional information about that command.
Our Final Design. The commands are puzzle pieces which the user is required place together in correct order to accomplish the goal on the right. The character of “Henry the Robot” and the “Treasure chest” is introduced in this design to define the objective of the gaming aspect. One save and quit button was introduced on the top left instead of having a drop down file menu. The option of toggle grid was added to help the user with more difficult levels.
Selected Interface

The interface above was the final design chosen to use in our video and prototypes. It appeared to be the most intuitive and simple among the ones we had previously designed. The interface clearly separates the different parts of the program from the program list, the command list, the world view, and the command bar. As well as separating the different components, it appears simple and inviting enough to encourage users to dive into the heart of programming and trying out different things.

We tried to keep in mind who our users were going to be when designing this interface. Our target users are those in highschool and college who wanted to learn programming in a fun and friendly environment. The program is geared toward those who have never had any programming experience and therefore has to lay down the fundamental logical concepts needed to further expand their programming horizons. The goal is to teach them these logical concepts without bogging the user down in syntax. We do this by providing them with all the pre-written code they need to complete a series of tasks and challenges, which are solved in ways that teach new concepts. We want the users to have fun while learning how to program and be able to experiment on their own outside the constraints of what is given. The tasks and challenges will gradually get harder and if a user does not correctly write code that solves the current level, the program will either terminate without winning, or create an error that describes what went wrong to the user so they can fix it.

In the interface, we wanted to clearly lay out what things were. In the upper left you see a robot face, that represents the currently selected entity you are editing code for. To change what entity you are writing code for, you would simply double click in the grid world on the entity you want to edit the program for, and the image and code on the left frame would change accordingly. If you select a non-editable object, nothing would happen.

As you are editing code, we wanted to demonstrate that different functions can connect together, and some cannot. To show this we made each function into a block, such as move right for instance. These blocks have different patterns on the top and bottom of them like a puzzle to show that only certain types of blocks can connect to others. Also, given this design, we can limit the number of function blocks the user has access to. For example, we may decide that in order to solve a particular problem the user only needs two move right blocks, so we can force the user to solve the problem only using two right blocks. All the blocks of functions you need to solve the puzzle are given in the command list, located above the grid world and program list. This is a dynamically changing list, so as commands are taken, the command will disappear from the list and the list will adjust as needed.

To find out more information about a block, you can either hover your mouse over a block and get a brief summary of what that function is, or you can go to the formal help menu. In the help menu there will be a list of functions on the left, and a information box containing detailed information about that function on the right. This gives the user two ways of getting information and the user can determine on their own if they need a little or a lot of information regarding a command.
Storyboards

The video starts with Eric playing a PC game called “Rome Total War.” He starts getting really excited about the game and then starts wondering how a game like this is made. He asks himself questions like how does the game work, what’s goes on in the background, and how did the developers come up with idea for it. All these thoughts raise his curiosity about programming. He is inspired to learn more about the programming that goes into the game and decides to open the source code for the game.

When he opens the file, he’s bombarded with symbols, syntax, and words that he’s never seen before. Even though he has no idea about how the code worked, he proceeds to make some changes. He had no idea how to compile the code. After browsing around for sometime, he finds the build command and tries using that. The result is hundreds of errors and warning just because of the small change he made to the source code. Frustrated and confused, he starts to use google, asking questions like “How to program?” and “Why is programming so hard?” The results lead him to explore different websites. He finally ends up googling “Programming made easy” which direct him to our program called “Henry the Robot.” He downloads the program and starts it.

He is greeted with a welcome screen which had a demo program. He notices that there’s a command list on top and two move right commands placed in order on the left side of the screen. After some observation, Eric clicks the play button and watches the robot on the grid move two spaces to the right towards the treasure chest. A pop-up shows up with the words “Level 1 Complete,” and two buttons, “This Level” and “Next Level.” Because he’s still unsure about what exactly the program on this level did, he clicks the “This Level” button which returns him to the demo program again.

He looks at the level more carefully this time, knowing that the premade program gets the robot to the treasure chest. He notices that the robot on the grid was two spaces away from the treasure chest. After some observation, he decides to click one of the move right commands in the program list, which moves the command up to the command list. This leaves only one move right in the program list. He then clicks the play button which causes the robot on the screen to move one step to the right, the program to terminate, and the grid world to go back to its original position. He starts wondering why the program didn’t work and decides to do some further exploration. He thinks that it would be very helpful if there was some way of knowing what the commands actually do. He goes up to the move right command in the command list and pauses at it, trying to think of how he would get more information on it. As he hovers over it, a bubble describing what the command does pops up. It explains that the move right command “moves the robot one place to the right.” From this, he realizes that he would have to use both of the move right commands to accomplish the goal. So he clicks the move right command in the command list pane, causing it to move to the left pane and attach itself to the end of the program. He then proceeds to click the start button which causes the robot to move two steps towards the treasure chest, accomplishing the goal. A dialog box pops up showing “Level 1 Complete” along with “This Level” and “Next Level” showing him that he has accomplished the goal for that level. He clicks “Next Level” to continue learning with Henry the Robot.

After using the program for a some time and completing the various levels, Eric starts to get the hang of programming. He’s able to quickly clear the easier levels and use multiple
commands in different ways. He now understands concepts related to object oriented programming, efficient algorithms and code optimization. It wasn’t long before he was able to write full programs using actual syntax.

**Creation of an Epic**

Before we all met to start filming and writing the paper we had all agreed to start off with a couple of storyboard ideas on how the sequence of events should occur. We came together and discussed what we liked about certain storyboards and what we didn’t like about others and how they would play into the actual film. We agreed we wanted the prototype to act out naturally by letting the main character in the film explore and try different things freely while still completing the three tasks.

Before we started filming and creating the final prototype for the tasks, we first created the storyboard for the entire film. We drew up each scene as a box in the storyboard and represented what we wanted to accomplish in that scene. For example, we wanted to start the movie off that the main character was a video gamer and had thoughts about trying to create his own games. This was represented in multiple boxes and we drew the scene in such a way as to suggest the camera angles for shooting a lot like Steven Spielberg did for his movies as shown in class.
The storyboard for the whole film, particularly the first half with the live footage. (Reads left to right, top to bottom)

For the prototype, we created the scenario using powerpoint, so we could make it look somewhat pretty and control all the aspects of how it would look without the error-lending filming it in real time. We then used CamStudio to record the prototype and then edited that film into the live action film we recorded earlier.

We started off filming the real life scenes first to get a better grasp of what the user would be like when using the prototype. We tried to make the film seem somewhat professional and took multiple shots of each scene and clipped them together to form a continuous movie. To keep the viewer from forgetting that the prototype is supposed to be occurring in real life and not just a demo, we edit back to real life at the end to see the result of the main character’s adventure using the program.

What we found difficult was making the camera seem to flow naturally from each clip to the other. Figuring out the best way to get the prototype in the video was also a challenge; we bounced between several different ways of making the prototype, such as paper, Balsamiq, and Powerpoint (which we ended up using). Trying to tell the story and include the prototype in under 4 minutes also proved difficult, we were trying to find the sweet spot of having real film with the Powerpoint footage and keeping it engaging while making it clear what the tasks were.
References
