Affective Walkthroughs and Heuristics: Evaluating Minecraft Hour of Code

Reza GhasemAghaei, Ali Arya, and Robert Biddle

School of Computer Science,
Carleton University
Ottawa, Canada
{Reza.GhasemAghaei,Ali.Arya,Robert.Biddle}@carleton.ca
https://hotsoft.carleton.ca/

Abstract. This paper presents an evaluation of Code.org's Minecraft Hour of Code that was created to encourage and support people initial learning of computer programming. In particular, this web-based software uses a spatial model world, where the learner's programs manipulate the world. We applied the Affective Walkthrough and the Affective Heuristic Evaluation, proposed evaluation methods for affective learning in multimodal educational software. Our findings provided illumination about the Minecraft Hour of Code approach, highlighting some aspects that are successful, and others where improvement appears necessary. We also gained insight about the evaluation methods and their effectiveness.

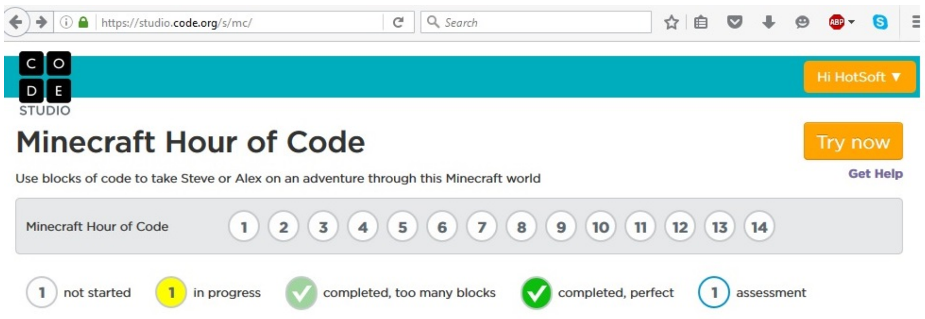
Keywords: education software; evaluation; affective learning; spatial environments

1 Introduction

This paper presents an evaluation of Code.org's Minecraft Hour of Code. The software was created to encourage and support people initial learning of computer programming. In particular, this web-based software uses a spatial model world, where the learner's programs manipulates the world. We applied the *Affective Walkthrough* and the *Affective Heuristic Evaluation* proposed earlier [4] and refined with the previous case studies [5, 6]. Our previous studies were conducted with participants to refine our methods, and inform us about the collaborative processes. In this new study, we simply applied this knowledge and conducted the evaluations ourselves. Our findings provided illumination about the Minecraft Hour of Code approach, highlighting some aspects that are successful, and others where improvement appears necessary. We also gained insight about the evaluation methods and their effectiveness.

Code.org is a non-profit organization that encourages learning of computer science, especially programming and computation thinking. It's "Hour of Code" is an initiative that emphasizes short introductory tutorials aimed at total beginners, and has been widely used, claiming almost 100 million learners, including US President Barack Obama [14].

The Hour of Code approach is strongly visual in nature, both for the program itself, and for what the program does. This is reasonable, as it is established that such visual environments might assist in novice engagement, and also support them demonstrate skills and strategies in a familiar or easily understood context [1].



Minecraft Hour of Code

Use blocks of code to take Steve or Alex on an adventure through this Minecraft world

Minecraft Hour of Code

Fig. 1: The initial interface of the Minecraft Hour of Code tutorials.

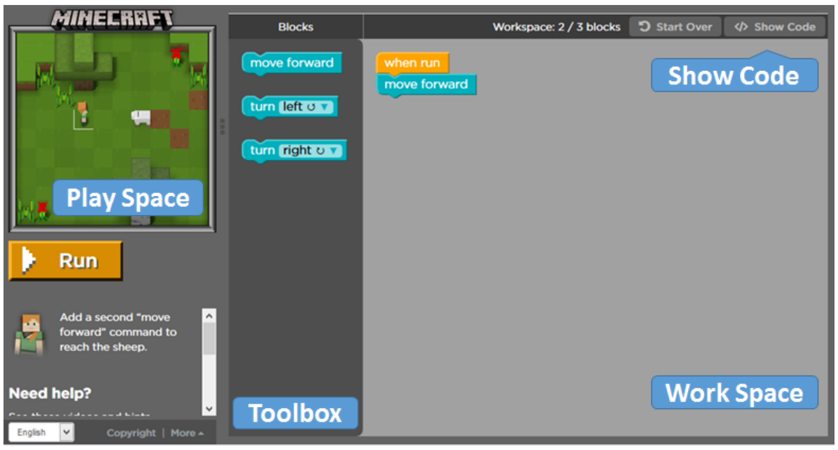
One aspect of Hour of Code's visual nature relates to the program itself, and it uses a variant of the "jigsaw" or "block" approach used in Scratch [11], Alice [2] and other learning systems. The other aspect is the spatial model world, where such worlds include graphical elements that serve as integral parts of a computational context, problem, and solution. The Hour of Code tutorials include many variants of this, for example including ones resembling many popular games, e.g. "Angry Birds", " Plants vs. Zombies". In this paper we focus on the "Minecraft" variant — see Figures 1 and 2. This is of special interest because Minecraft itself involves programming as a core part of the appeal of the "game".

Our interest is in *affective* learning in multimodal educational software, focusing on the emotional elements in the software. This reflects calls for more attention to emotion in the domain of Human-Computer Interaction (HCI) and education [15]. In this paper we present our application of two such methods, the affective walkthrough and the affective heuristic evaluation proposed earlier to the Hour of Code Minecraft software.

2 Overview of Hour of Code Minecraft

The Hour of Code Minecraft tutorial was developed to engage students in learning programming. It introduces players to basic programming concepts, allowing them to navigate, mine, craft, manipulate and explore in a 2D Minecraft world by plugging together blocks to complete all actions and generate computer code.

The system provides a sequence of tutorials, illustrated by a horizontal line with nodes representing each tutorial step, as shown in Figure 1.



Add a second "move forward" command to reach the sheep.

Fig. 2: The Hour of Code Minecraft interface, with model world on the left, and the program on the right.

For each tutorial, the learner uses an interface such as shown in Figure 2. On the left top is the model world ("play space"), resembling the rectilinear blocks as used in the full Minecraft game, representing objects such as grass, trees, sheep, etc. On the left bottom is a panel with instructions to the learner. When a learner begins the tutorials, they select an avatar, either "Alex" or "Steve". The chosen avatar then appears in the model world, and the program controls their actions within the world.

On the right is an area consisting of a "toolbox" of jigsaw-like pieces, and a "workspace". The pieces in the toolbox are for simple statements or commands, as well as loop ("repeat") and conditional ("if") statements, and the shape of the pieces shows how they connect with other pieces. These can be copied and dragged into the workspace, and assembled in sequence to form programs.

When assembled, the jigsaw pieces do resemble traditional (textual) code, and in most Hour of Code this is the only code shown. In the Minecraft tutorial, however, whenever a tutorial step is completed satisfactorily, a pop-up window shows the equivalent code in JavaScript, which is what is actually used to program Minecraft.

In Hour of Code Minecraft, the tutorial steps reflect teaching and learning goals, in sequence. The sequence is fairly traditional, beginning with sequences of commands, moving on to loops, and then conditional structures. At first, the introduction is strongly guided, and then more freedom allowed to encourage a program solving approach.

In the interface and the tutorial steps, several modalities are involved. Firstly, the interface is graphical and spatial, with the model world and the jigsaw piece program. The character of the graphics is appealing, with a coarse lo-fi and fun game-like appearance that might help engagement. Secondly, the model world suggests a "world of action", like a board game, where the blocks might be moved and interact. Thirdly, and related, is that the tutorial instructions build narrative elements where the player's avatar should accomplish goals by moving and acting in the model world. Over the sequence of the steps, these build into an overall story. This approach would appear to help learner motivation, understanding, and possibly supporting reflection and creativity about what other possibilities might be programmed. Finally, there are audio elements with calm background music and occasional effects to emphasize events.

3 Minecraft Hour of Code and Affective Walkthrough

The affective walkthrough proposed earlier [4] was based on Wharton et al.'s cognitive walkthrough [13], and Dormann and Biddle's affective walkthrough [3]. Following case studies, it was refined [6] to de-emphasize normal usability issues, and to acknowledge the contextual role of teaching goals and potential modality benefits. It follows Kort et al.'s affective model [7]. This identifies four phases of learning and the affective character of each. The first phase is encouraging exploration with positive affect. The second phase introduces challenges, and negative affect is expected. The third phase is to support overcoming challenges and reduce the negative affect, and the fourth phase is to affirm learning and restore positive affect.

3.1 Method

As with the earlier cognitive walkthrough by Wharton et al., the main idea is to select user tasks and then step through the software considering key questions, and making notes and observations, as shown in Table 1. The walkthrough was conducted by us, role-playing evaluators. We used a large screen display to collaborate in the same manner as done by participants described in [6].

The Affective Walkthrough (Version 3)	
1	What is the learning goal of this task?
2	Where in the affective cycle of learning is this task? (i.e. exploring, challenging, overcoming, and affirmation)
3	Is the appropriate affective support provided?
4	Does the affective support work as intended?

Table 1: The Affective Walkthrough (Version 3).

The objective of our study was to evaluate affective learning in Hour of Code Minecraft tutorials, with special attention to the supporting modalities involved. We presented, above, the basic design of the tutorials and the interfaces, and identified the modalities involved and their potential benefits. We then outlined the basis of the affective walkthrough, and the steps involved.

To select the tasks, we simply used the Hour of Code Minecraft tutorial steps, although we grouped the 14 steps into 4 larger tasks, bringing together steps that had strongly related educational purposes.

We then followed the walkthough as shown in Table 1, and considered each question in turn. We made Walkthrough Evaluation Sheets to record our answers and related observations. In the section below, we review the nature of the taskes, and the results of our walkthroughs.

3.2 Findings

First Task To begin, we select a character. We have two choices, e.g. we can pick either Alex or Steve. The first task is about “commands”. The steps are shown in Table 2 and the results of the walkthrough are shown is Table 3.

First Task: Command Sequences	
1	Add a second “move forward” command to reach the sheep.
2	Then walk to the tree and use the “destroy block” command to chop it down
3	Use the “shear” command to gather wool from both sheep.
4	Cut down all trees.

Table 2: First task: walk around.

By completing the first part of Task 1, the software gave emotional feedback such as happy sounds, a jumping character, green highlight on the achievement bar and a pop-up window providing feedback. Figure 3 shows the pop-up window with the achievement bar and our character reaction at the bottom. If we were not able to complete the step with the minimum lines of code, the software will let us know with a challenging part, but it does not help us to complete the step with the minimum lines of code (see Figure 4). But if it was really going to encourage exploration, it should allow the user to see all of the options he/she could select and do, but it does not.

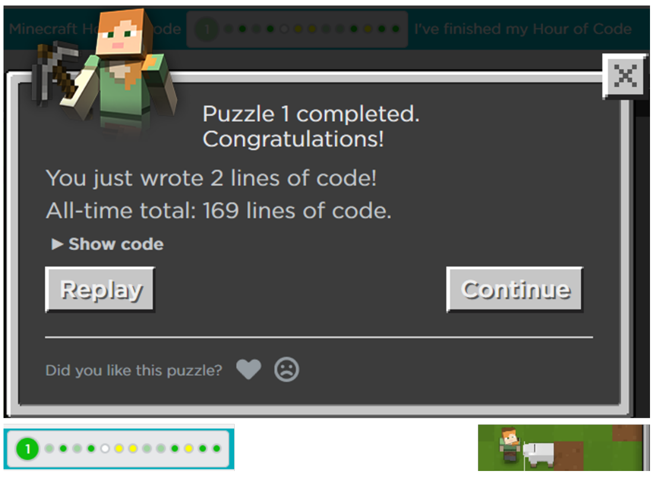
For every task, the software only provides us with the number of blocks we need. Therefore, they are not encouraging exploration all that much. They are not providing us with other alternatives. That is a bit of a criticism if we are encouraging people to explore. If we want to encourage exploration, we do not constrain people. We give them a bunch of different things, so that they explore to see what happens. We are surprised they did not give all the blocks that are relevant to let us explore a bit (e.g. *shear* sheep and *cut down* tree). We were a bit confused that we did not know all the possible things or at least some small subset of the possible things.

In the second part of the first task, the software took away*cut down*, and added *shear*.

It provides the user with more information about the reason of doing the task, e.g.“wood is a very important resource. Many things are made from it.” The third part of the first task uses “shear” command instead of “destroy block”.

Walkthrough— First task: Command Sequences	
1	What is the learning goal of this task? About learning programming and being engaged. In this task we used commands and add them in the work space to help the character move around the screen. we were able to see the code that we created with using blocks. It can be done, but there were no motivation to it, and considering the role of narrative, which could be better. *Summary: The learning goal of this task was understood to learn how to add new commands in the program, but no motivation.*
2	Where in the affective cycle of learning is this task? (i.e. exploring, challenging, overcoming, and affirmation) It cover exploring and gives feedback through sound and text if I did wrong and otherwise an affirmation sound with jumping character and the character being happy. *Summary: The affective cycle of learning was exploring and affirmation.*
3	Is the appropriate affective support provided? Yes, by adding blocks from toolbox in the work space, we can see how our character moves in the play space, and receiving positive feedback or negative that is still encouraging by providing how to try again with character's emotional reaction. *Summary: appropriate affective support is provided with different multimodal support.*
4	Does the affective support work as intended? No, affective support did not work as intended. The emotional support was well provided, but the task did not provide reasons of using the commands. *Summary: even with the affective support provided, the purpose of the task was not clear.*

Table 3: First Task Walkthrough Results: walk around.


Puzzle 1 completed.
Congratulations!

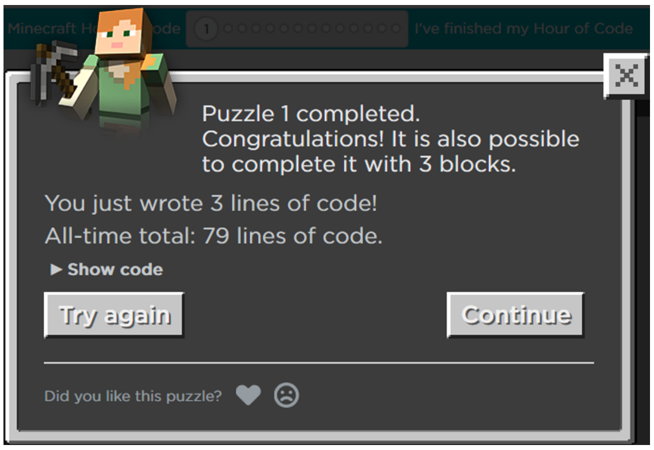
You just wrote 2 lines of code!

All-time total: 169 lines of code.

Fig. 3: Task 1 - Step one.

Second Task: Repeat Loop and Commands — Simple Task	
1	What is the learning goal of this task? The learning goal of this task was to help you learning about repeat loops and commands in programming. *Summary: learning about repeat loops and commands.*
2	Where in the affective cycle of learning is this task? (i.e. exploring, challenging, overcoming, and affirmation) The affective cycle of learning in this task covers all the four cycles. The Minecraft let us explore, and provide us with three different easy to medium, and hard questions covering the challenging cycle. Then it helps us to overcome with popup windows. And at the end it gives us affirmation. *Summary: it has exploring, challenging, overcoming, and affirmation.*
3	Is the appropriate affective support provided? Yes, it starts with a question and a hint, i.e. “We need to build a house before the sun goes down. House requires a lot of wood. Cut down all the 3 trees.” *Summary: affective support was provided*
4	Does the affective support work as intended? No, it does not work as intended. It really does not help us to build a house, just create some walls. Other issue with the software it tells us the task should be done before the sun goes down, but it does not work as intended, and we do not have a time limit or a change in the background. *Summary: the purpose of this task is to build a house, which it did not help us to do so.*

Table 4: Second task walkthrough results: build a house.



Puzzle 1 completed.
Congratulations! It is also possible to complete it with 3 blocks.

You just wrote 3 lines of code!
All-time total: 79 lines of code.

Fig. 4: Task completion but not with minimum lines of code.

As we are not using repeat loops in this task, the last part of task one needed lots of "moveForward()" blocks. It would be easier if we could just tell the computer to perform the move forward command the number of times we needed. It would be much easier to transition to repeat loops, if we had told the computer to "move forward" a specific number of times. We will need thousands and thousands of blocks to create a new world in Minecraft.

Second Task The second task is about building the rest of a house from the material available. The "repeat" command will come in handy. For the last part of the task we can select "Easy", "Medium" or "Hard". The steps are shown in Table 5 and the results of the walkthrough are shown is Table 4.

Second Task: Repeat Loop and Commands — Simple Task	
1	Build the first part of your house by putting the "place" and "move forward" commands inside the repeat loop.
2	Let's build a house. Choose the floor plan (easy, medium or hard) for your house.

Table 5: Second task: build a house.

The second and third tasks are about how to come up with algorithms, giving a problem and solve it using repeat loops. For the first part of task two, the software provides the reason and time to complete this part by saying: "We need to build a house before the sun goes down. Houses require a lot of wood." For the second part of this task it suggests: "Every house starts with a wall."

This part of the software is not telling the user how much wood he/she gets, and why we need to chop down all the trees. We do not have variables. We do not have any idea how much wood we have. Also we can not really build a house; we can just build some walls, so the story is not correct.

Third Task This task is again about using "Repeat loops" and "commands", but in a more challenging way using some creativity and freedom. It has four steps and each step takes you to different interactive spaces including: 1. Plant crops on both sides of water, 2. Move past the Creepers and reach safety at home, 3. Move underground placing torches and mine coal, 4. Avoid walking into molten and place cobblestone to create a bridge, and mine iron blocks. The steps are shown in Table 6 and the results of the walkthrough are shown is Table 7. This task has gamification steps to bring more engagement to the user.

Third Task: Repeat Loop and Commands — Challenging Task Commands	
1	Plant crops on both sides of the water so you don't get hungry later on. (It's good to plan ahead.)
2	Running into a Creeper is a bad idea. Carefully move past the Creepers and reach the safety of your home.
3	You'll find the most valuable resources underground, but it can get dark. Place at least 2 torches and mine at least 2 coal.
4	Walking into molten lava is a bad idea. Place cobblestone to create a bridge, then mine at least two of the iron blocks.

Table 6: Third task: plan some crops.

Forth Task Finally, the last task is about "commands", "repeat loops", and learning about "if" statements; a fundamental part of learning to program. It introduces us to a concept, which requires more problem solving. "if" statements help all computers make decisions. We are able to use "if" statement in the code to make our character react to what they see in the world, e.g. if there is rock in front of her, she can turn left or right and watch her steps. It has two steps; first step is simple and second step is more challenging, which are including: 1. Lava is hiding beneath some of these blocks, which you'll need to cover up before moving forward, 2. Mine redstone but don't fall in the lava by placing cobblestone over any lava you uncover. The steps are shown in Table 8 and the results of the walkthrough are shown is Table 9. This task has gamification steps to bring more engagement to the user.

3.3 Discussion

Reviewing the results of our walkthroughs, we are able to make several general observations. Most simply, we were pleased with several strengths: good visual effects and artwork, and sensible audio. Beyond those, however, we found more insight, described in sections below.

Third Task: Repeat Loop and Commands — Challenging Task Commands	
1	What is the learning goal of this task? The learning goal of this task was to practice using different commands with repeat loops bringing challenging steps. *Summary: providing challenging steps.*
2	Where in the affective cycle of learning is this task? (i.e. exploring, challenging, overcoming, and affirmation) This task only covers affirmation and somehow challenging cycles, the steps are not encouraging enough and there is no clear help to overcome the challenges. *Summary: it covers challenging and affirmation cycles.*
3	Is the appropriate affective support provided? Yes, there are similar affecitve support provided that are: giving a short story about the task and then emotionally trying to engage us in the space *Summary: it was provided.*
4	Does the affective support work as intended? There was a story at the beginning of each step, but narrative is not considered well to make it an exciting experience. *Summary: even with bringing challenge in this task it was not motivating and good story is missing.*

Table 7: Third Task Walkthrough Results: Plan some crops.

Fourth Task: "if" statement	
1	Lava is hiding beneath some of these blocks, which you'll need to cover up before moving forward. An "if" command will come in handy here. Add a "move forward" command in the correct place to mine these blocks.
2	Now things are getting tricky. Mine 3 redstone, but don't fall in the lava. Use an "If" command to place cobblestone over any lava you uncover.

Table 8: Fourth Task: "if" statement.

	Walkthrough— Fourth task: "if" statement
1	What is the learning goal of this task? The learning goal of this task was to use command, repeat loop, and "if" statement. *Summary: learning about "if" statement.*
2	Where in the affective cycle of learning is this task? (i.e. exploring, challenging, overcoming, and affirmation) This task covers the challenging (providing two steps: simple and challenging) and affirmation, but does not encourage in exploration or help to overcome. *Summary: covers challenging and affirmation, but exploring and overcoming is not very well done.*
3	Is the appropriate affective support provided? Yes, we saw different affective support during the task. *Summary: there is appropriate affective support provided.*
4	Does the affective support work as intended? no, the affective support did not provide a clear narrative to engage us in doing this task. *Summary: no, the affective support did not provide a clear narrative*

Table 9: Fourth Task Walkthrough Results: "if" statement.

World and Story Perhaps our main positive finding was the interplay between the model world of the "playspace" and the narrative aspects of the programming tasks given in the instructions. We felt it was clear and motivating when the learn was asked to create programs to move the avatar, chop down the tree, build a wall, and so on. The world and the story seemed to go together well, almost the way that narratives often work in computer games. This worked very well here, and showed some limitations in tutorials in Scratch, for example, which typically start with a blank canvas, rather than a world ready for a story.

While acclaiming this aspect, we found two weakness. One is that there was often little motivation for the actions requested in the instructions. The learner is asked to chop down trees, for example, before any mention of building walls for a house. Second is that there was no overall story, no eventual goal to accomplish, despite this being so common in computer games. Even in Mario, the player knows they are not done until the princess is saved.

Challenge but not Overcome We very much appreciate the way in which the tutorials helped the learner get started, offering a clear narrative goal, exactly the right tools to accomplish it, and providing affirming feedback when done. In several places, however, it seemed that this limited exploration: it encouraged simple constrained programming, but did not add the context of a large number of possibilities in the "toolbox". Learning to choose a strategy in the presence of many possibilities, with different strengths, is essential to learning about problem-solving.

We also liked the ideas of greater challenge by suggesting it might be done with fewer steps: we felt it really would make the learner try to do better. But whereas a real teacher might monitor and provide hints when the learner gets frustrated, the tutorial offered nothing equivalent. Considering Kort's cycle, there was support to challenge, but not to overcome. As we show in Figure 5, we see the support for the cycle strong in some places (Challenge, Affirm) but weak in others (Explore, Overcome).

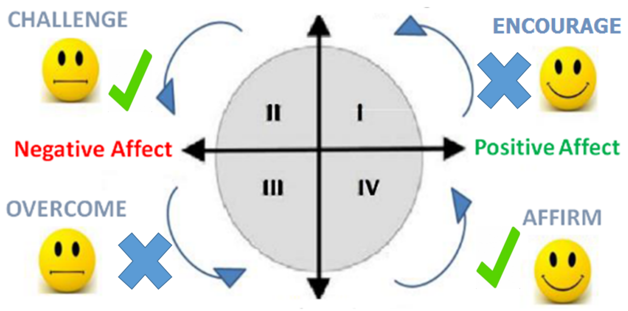


Fig. 5: Minecraft and Kort et al. [7] steps.

Simple Programming As the name "Hour of Code" suggests, the tutorial only addresses the simple beginnings of programming — "First Hour of Code" might be more appropriate. We noticed no real introduction to variables, for example, despite the potential of having them associated with objects in the model world. There was also no lead-in to object orientation, the most common practical paradigm for programming, again despite the potential of objects in a model world — this is exploited well in Alice, for example. In reviewing comments on official videos about the Hour of Code tutorial, this was a common criticism, that it wasn't introducing "real" programming:

> I'm not convinced that this kind of a tutorial could help anybody in understanding of programming paradigms. Modern programming is mostly object-oriented, and this film could at most give a little of vision, how sequential programming should look alike. IMO it's not proper to teach only sequential programming, without any code and without even telling, that another styles of programming exists. Moreover, I think that beginning learning of programming with sequential programming instead of object-oriented programming could make OOP harder to understand
> *Youtube Comment*

However it is still an open question as to whether OOP or sequential programming is best for absolute beginners. Similarly, other programming paradigms are also not addressed. These issues are beyond our limited scope in this study.

One last point is perhaps more relevant. By basing the tutorial on Minecraft, many beginners might hope to get started with actually programming in Minecraft, which is done in JavaScript. The tutorials do show pop-up windows showing JavaScript code equivalent to the jigsaw blocks, but go no further. We understand that the complexities and dependences in real JavaScript would present great difficulties to address within the Hour of Code framework, but we also anticipate that some beginners might find this disappointing.

Affective Heuristics (Version 3)
H1: Design elements and modalities should support the affective learning strategy
H2: Ensure help and documentation is provided where needed but does not distract from affective learning strategy
H3: Maintain visibility of progress, affirming challenges already overcome, and those remaining
H4: Allow the user freedom to explore but also to return to the previous step
H5: Avoid or prevent actions with neither feedback to help overcome, nor affirmation when success
H6: Visualize options clearly to encourage exploration
H7: Tailor actions to be encouraging at first and efficient later, while learners are attempting to overcome challenges
H8: Challenge learners and provide constructive feedback if they fail, and affirming success when they succeed
H9: Match the learners world view in affective strategy and multimodal support
H10: Maintain interface cohesion to support affective strategy

Table 10: The Affective Heuristics (Version 3).

4 Minecraft Hour of Code and Affective Heuristic Evaluation

This section addresses the use of the affective heuristic evaluation [5] to evaluate affect in the user interface, educational design, and content of Minecraft Hour of Code to see if the software supports the educational objectives, narrative and persuasion to make the learners more engaged in learning programming. The affective heuristic evaluation is based on Nielsen and Molich [9], and adapted with affect supported by Norman [10] and Kort et al.'s emotional cycle of learning model [7] as well as multimodal design based on Sankey [12]. We have explained the full rational in more detail earlier [4].

4.1 Method

The study was performed in the same laboratory as earlier. In this study, we applied the evaluation method as discussed earlier. We reviewed the heuristics, explored, and then reflected on the software. We then wrote comments about the environment filling in a table with respect to the different heuristics. For heuristic severity we adapted Nielsen's severity ratings for usability problems [8], changing their emphasis to reflect emotional educational impact. We explicitly emphasized on the learning objective and the modality that was employed.

For the learning objective we considered the following. This system was designed to be fun for a student working alone or in a classroom where the instructor has minimal preparation or computer science background. It has fourteen tutorials to learn the basics of JavaScript programming concepts that are: use of commands, repeat loops and if statements. It aims to create a quick and enjoyable experience for students and instructors who are new to computer science.

The modality involved a 2D block world with programming using jigsaw pieces. There is also an avatar to represent the user, animation of the block world, and engaging audio samples. Moreover, it used a narrative and persuasion quasi-modalities.

We went through the interface systematically. For example we added blocks to the workspace, trying to think of the intended user. We checked in each step if the system state to consider the learning objective and the modality employed. At the end of the suggested tasks, we filled in a table with issues based on the ten heuristics. Table 10 shows the affective heuristic evaluation. This is the set of heuristics following revisions discussed in [5].

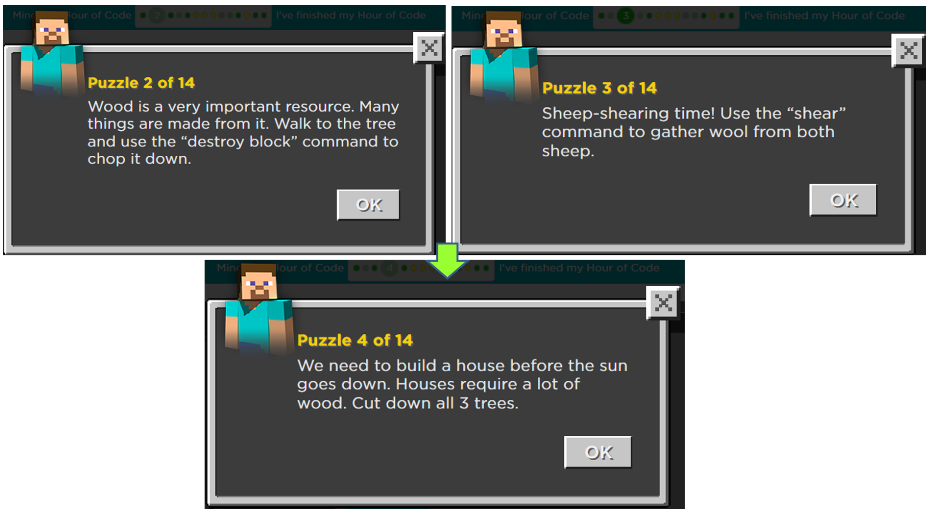
4.2 Findings

In this study, the evaluators read the heuristics, explored, and then reflected on the environment. They then wrote comments including the interface element that was violated, problems that illustrated poor considering of affect and modality, suggestions and recommendations for solutions, and the severity based on Jakob Nielsen's five-step rating scale [8]. Tables 11 and 12 show the affective heuristic evaluation results. Each of the ten heuristics led to a useful comment at least once.

4.3 Discussion

The heuristics invited reflection on modality as well as affect. Our findings show that the affective heuristic evaluation led to identification of many problems and potential solutions as shown in the tables. By reviewing these results, we can make some general observations.

Narrative and continuity There was no clear continuity between the steps. One of the tutorial steps was to chop down trees (Figure 6 top left), and the next one was to *shear* sheep (Figure 6 top right). The wool that resulted from shearing was never used or mentioned again. It did say that wood was useful for building and later steps did build walls for a house, but there was no indication of the wood came from chopping down the trees. It could have been made more continuous making it clear that the trees were used to build the house and also by using the wool to make carpets etc. They are building a house without using the wood or the wool, and therefore, there is no strong continuity and storytelling between the steps (Figure 6 bottom). This might also have been a useful opportunity to introduce quantities of wood or wool as variables.


Puzzle 2 of 14
Wood is a very important resource. Many things are made from it. Walk to the tree and use the "destroy block" command to chop it down.
OK

Puzzle 3 of 14
Sheep-shearing time! Use the "shear" command to gather wool from both sheep.
OK

Puzzle 4 of 14
We need to build a house before the sun goes down. Houses require a lot of wood. Cut down all 3 trees.
OK

Fig. 6: Narrative and continuity.

Encouraging Exploration There was not enough encouraging to explore at each step, and no clear story for exploration. The commands provided were specific to each step, meaning there were not all commands provided in each tutorial e.g. *shear* was provided in the sheep-shearing task but not in other tasks. The commands were changed for every tutorial, which is a real limit to learning as it restricts exploration, and people would expect specific commands for each programming situation. Figure 7 shows the command blocks for tutorials two and three. They can use "destroy block" in the second tutorial step but it is not seen in the third tutorial step where they have to use "shear" instead.

JavaScript Code As we mentioned, the real Minecraft software is programmed using JavaScript. Minecraft Hour of Code uses the jigsaw command blocks. But when the task is complete, is shows the real JavaScript code. But a learner would not know what they have to do with this code; there is a poor connection between

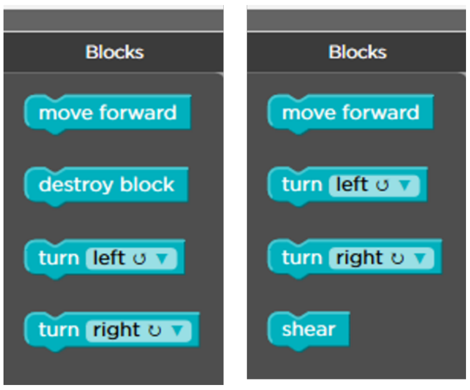
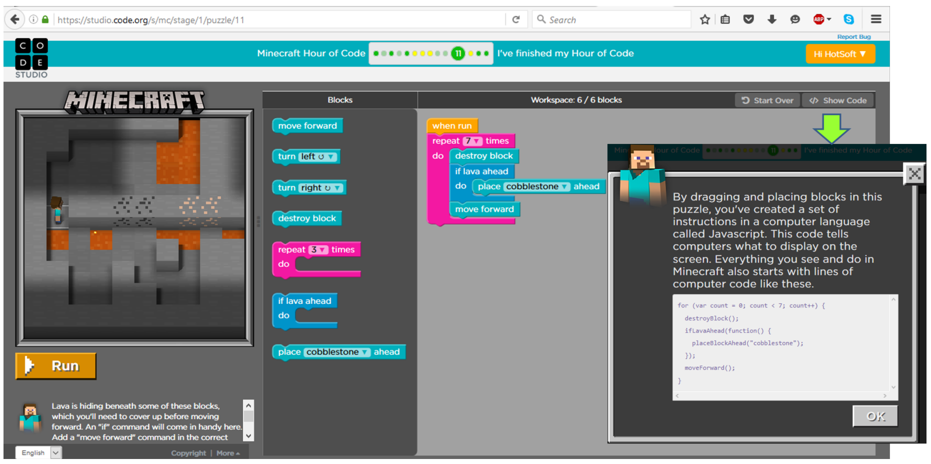


Fig. 7: Encouraging Exploration.

the JavaScript code and the block commands that represents that code, and there is no affective feedback provided to the learners. Therefore, the connection between them is not made; we can not edit or change the JavaScript code. They are not really learning the JavaScript language. The JavaScript code is not even nicely formatted, and there is no interaction with the learners, which can lead to disappointment for them (see Figure 8).



Lava is hiding beneath some of these blocks, which you'll need to cover up before moving forward. An "if" command will come in handy here. Add a "move forward" command in the correct

By dragging and placing blocks in this puzzle, you've created a set of instructions in a computer language called Javascript. This code tells computers what to display on the screen. Everything you see and do in Minecraft also starts with lines of computer code like these.

```
for (var count = 0; count < 7; count++) {
  destroyBlock();
  ifLavaAhead(function() {
    placeBlockAhead("cobblestone");
  });
  moveForward();
}
```

Fig. 8: Blocks with the generated code.

5 Conclusions

In this paper we presented our evaluation of the multimodal affective learning support in Code.org's Hour of Code tutorial set based on the popular game *Minecraft*. We applied our two proposed evaluation methods called the affective walkthrough and the affective heuristic evaluation, modeled on the widely used cognitive walkthrough and heuristic evaluation. The new methods keep the procedural framework of the cognitive walkthrough and heuristic evaluation, but as the evaluator steps through, the questions and heuristics are about emotional support in education.

The Hour of Code Minecraft software applies various modalities to support learning, talking a visual approach with a engaging game-like graphics, a model world, and a programming language using jigsaw-like pieces (as do Scratch and Alice). The tutorial instructions add a narrative aspect, whereby the programming tasks involve acting out a story in the model world.

Our experience in conducting the evaluation was positive and enlightening. We recognized elements in the Hour of Code design that we would not otherwise have noticed, and also identified ways in which the design might be improved. We see as especially valuable the interplay between the spatial model world, and the narrative stemming from the instructions. On the other hand, we felt that some extra freedom in command choice might in several places be helpful. Also, where challenges were provided by suggesting shorter solutions were possible, it would be helpful to add a capability for the learner get hints so they can overcome the challenges if they are stuck. We also recognized the limits of the tutorials, which really only learners with the very early steps in learning to program.

We appreciated the focus of the affective walkthrough and affective heuristic evaluation focus on modality benefits and teaching goals, and most importantly its use of Kort's model of the affective cycle in learning. We feel they offer a helpful and systematic approach to evaluating affective learning in multimodal software.

References

1. Chao, P.Y.: Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. Computers & Education 95, 202–215 (2016)
2. Dann, W.P., Cooper, S., Pausch, R.: Learning to Program with Alice (w/CD ROM). Prentice Hall Press (2011)
3. Dormann, C., Biddle, R.: Understanding game design for affective learning. In: Proceedings of the 2008 Conference on Future Play: Research, Play, Share. pp. 41–48. ACM (2008)
4. GhasemAghaei, R., Arya, A., Biddle, R.: Multimodal software for affective education: Ui evaluation. In: EdMedia: World Conference on Educational Media and Technology. vol. 2015, pp. 1851–1860 (2015)
5. GhasemAghaei, R., Arya, A., Biddle, R.: Evaluating software for affective education: A case study of affective heuristics. In: EdMedia: World Conference on Educational Media and Technology. vol. 2016, pp. 573–580 (2016)

6. GhasemAghaei, R., Arya, A., Biddle, R.: Evaluating software for affective education: A case study of the affective walkthrough. In: International Conference on Human-Computer Interaction. pp. 226–231. Springer (2016)
7. Kort, B., Reilly, R., Picard, R.W.: An affective model of interplay between emotions and learning: Reengineering educational pedagogy-building a learning companion. In: icalt. p. 0043. IEEE (2001)
8. Nielsen, J.: Severity ratings for usability problems. Papers and Essays 54, 1–2 (1995)
9. Nielsen, J., Molich, R.: Heuristic evaluation of user interfaces. In: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 249–256. ACM (1990)
10. Norman, D.A.: Emotion design: Why we love (or hate) everyday things (2004)
11. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., et al.: Scratch: programming for all. Communications of the ACM 52(11), 60–67 (2009)
12. Sankey, M.D.: How to develop 15 multimodal design heuristics in 3 easy (not) lessons. International journal of Pedagogies and Learning 3(2), 60–73 (2007)
13. Wharton, C., Rieman, J., Lewis, C., Polson, P.: The cognitive walkthrough method: A practitioner's guide. In: Usability inspection methods. pp. 105–140. John Wiley & Sons, Inc. (1994)
14. Wilson, C.: Hour of code—a record year for computer science. ACM Inroads 6(1), 22–22 (2015)
15. Woolf, B., Burleson, W., Arroyo, I., Dragon, T., Cooper, D., Picard, R.: Affect-aware tutors: recognising and responding to student affect. International Journal of Learning Technology 4(3-4), 129–164 (2009)

Affective Heuristic Evaluation Involved	Interface Element	Problems not considering affect and modality	Suggestions for Solution/ Comments/ Recommendations	Sev (0 to 4)
H1: Design elements and modalities should support the affective learning strategy	Show code	The JavaScript code is provided but without affective support	Have it be more interactive Some kind of interaction to get them more engaged Use more physical/material design elements	4
H2: Ensure help and documentation is provided where needed but does not distract from affective learning strategy	Affirmation pop-up window	It does not provide all the possible things or at least some small subset of possible things	Give hints and use affective persuasion	3
	Tasks, command sequences	Reasons of using commands not provided	There can be a multimedia content e.g. video and providing the purpose and goals of using commands	3
H3: Maintain visibility of progress, affirming challenges already overcome, and those remaining	Entire environment	Pretty good job for this part	Progress timeline can also show the steps	0
H4: Allow the user freedom to explore but also to return to the previous step	Tool Box	Software only provides us with number of blocks we need, and it is not encouraging exploration all that much	It should provide us with other alternatives	3
	Minecraft Hour of Code tutorials list	No label next to each task in the timeline	Better to have label	4
H5: Avoid or prevent actions with neither feedback to help overcome, nor affirmation when success	Affirmation pop-up window	To overcome the challenge with minimum lines of code there is no feedback to help overcome	Provide some feedback to how to be able to get to minimum lines of code	3

Table 11: The Affective Heuristic Evaluation H1 to H5.

Affective Heuristic Evaluation Involved	Interface Element	Problems not considering affect and modality	Suggestions for Solution/ Comments/ Recommendations	Sev (0 to 4)
H6: Visualize options clearly to encourage exploration	Tasks, command sequences	No motivation and role of narrative could be better	Provide a good narrative about using command sequences in this task	2
	Tasks, repeat loops	Narrative is not considered well to make it an exciting experience	Provide a good narrative about using repeat loops in this task	2
	Tasks, “if” statement	No clear narrative.	Provide a good narrative about using "if" statement in this task	2
H7: Tailor actions to be encouraging at first and efficient later, while learners are attempting to overcome challenges	All tasks	No clear narrative and connection between the steps	Easier tasks are earlier but there is no clear connection between the steps	2
H8: Challenge learners and provide constructive feedback if they fail, and affirming success when they succeed	Puzzle 2 to 3	No challenge	add some challenge to puzzle 3	0
H9: Match the learners world view in affective strategy and multimodal support	Puzzle 4	Starts well, but then give examples, and then does not continue well, e.g. not building a house. Sun does not go down. Story is not correct	It is not complete but it is pretty good at creating a child world.	1
H10: Maintain interface cohesion to support affective strategy	Tool box	No learning to choose a strategy in presence of many possibilities with different strengths	Add more affective strategies	3
	Show Code	Also JavaScript not bringing engagement and motivation	Using affective strategies to have interaction with users	4
	All steps	Narrative is not well done	Have a better storytelling and continuity	2

Table 12: The Affective Heuristic Evaluation H6 to H10.