Introduction to Computer Science with MakeCode for Minecraft

Lesson 10: Final Project

In this, the final chapter of MakeCode for Minecraft, we ask students to create a final project that does two things:

1. Show what you know, and
2. Demonstrate something new

If you and your Agent were setting off on a Minecraft adventure, and you could only program four tools to go with you, what would you take? Students will create a Backpack of Tools for use by any Minecraft adventurer setting off on a great journey.

We'll start the chapter by reviewing the concepts we covered in the previous weeks, and provide some ideas for an independent final project that students can focus on in the next several weeks. We will also provide a rubric for keeping students on task and tracking the learning that they are doing as they work on their projects.

What tools will be in your backpack for your Minecraft Adventure?

Course Review

Here is a short review of the topics and concepts we have covered so far in the course:

1. Events
   Code is triggered by events that happen as a result of player actions. An Event Handler listens for different things to happen in the game, and then runs the code associated with it.

The code you write for Minecraft is triggered by events such as 'on player walk', or chat commands to call functions that you've created.

2. Coordinates
   Your position in the Minecraft world is indicated by three coordinates corresponding to your position in three-dimensional space. Coordinate positions can be specified as absolute (independent of your position) or relative (in relation to your current position). Many MakeCode blocks use coordinates to specify a location in Minecraft.

3. Variables
   A variable is used to store information in memory. Variables can be different data types: numbers, text, booleans, or positions. You can change the value of a variable or cause other events to occur based on a variable's value. You can also pass a variable in as a parameter to a chat command.

4. Iteration
   Iteration, or loops tell MakeCode to repeat instructions a certain number of times, or while a certain condition is true. You can nest one loop inside of another loop, to create more complex repetitions. The Agent is your friend who can carry out instructions for you in Minecraft.

5. Conditionals
   A conditional statement only runs if a certain condition or conditions are true. Using 'if… then… else' blocks allow you to check whether a given condition is true, and if so, execute some code, otherwise execute a different set of instructions. You can also check for the opposite of a given condition using the Logical 'Not' block.

6. Functions
   Functions are groups of instructions that accomplish something specific. Once you have defined a function you can refer to that group of instructions simply by its name, and those instructions will run. Just reading the function names makes it easier to see the big picture in a complex program. Hiding the individual instructions in favor of a more general name that describes what they do is an example of abstraction.

7. Arrays
   An array is a convenient way to keep track of objects. You can store things in an array, and retrieve them as needed, in any order. You can store numbers, text, positions, even animals and different Minecraft blocks in an array. When you don't know ahead of time how many

instances of objects you will need to store, arrays are helpful because you can keep adding new objects to the end.

8. Artificial Intelligence
   The field of artificial intelligence, or AI, is growing rapidly and offers lots of opportunities for problem solving. AI can help your Agent adapt to different situations in the Minecraft world, and can allow you to write code for many different possible scenarios. AI can be used to create complex structures, or to navigate existing ones. AI is one very popular way that coding and programming are being used in the world today.

About the Final Project (for Teachers)

The final project is a chance for students to use all the skills they have been learning throughout the semester. Students are asked to create a series of code examples that can be used in the Minecraft world. The time frame is flexible, but you might expect students to take a few days to develop, test, and refine each bit of code. They should end up with three or four separate projects over a three-week period.

The most open-ended projects, the ones that display the most originality and creativity in design, may not be projects that use the most complex code. Some original ideas, for example, are outstanding because of the clever solutions to problems such as bringing water to crops or creating an elaborate trap for the unwary. The code itself might simply be a few blocks attached to an event handler, or code that ignites a Redstone contraption.

Conversely, other projects may not have much Minecraft building involved, or may simply display output to the screen using a Say block, but the students have solved particularly difficult programming problems, such as how to encode a painting into a series of arrays.

Some projects will feature code that has to run in a particular Minecraft world. Other types of code might only run in a flat world, or might require you to find a jungle or other type of terrain in order to run it. This is why we typically ask students to turn in a short video of their code in action so we can see it the way it was meant to run.

It's important to step back and take a holistic view of the entire project and recognize where students did their best work. Where did they struggle the most? What are they most proud of? Then celebrate that.

We actually grade the process of students' learning over the course of the independent project, more than the product itself. We tell our students, "You don't judge a story by its ending," and in

fact, often the story is the most interesting part, not where you ended up (you don't care so much that they lived happily ever after. You want to know all of the trials and tribulations that got them to that point.)

We ask our students, after they finish the project, to go back and look and remind themselves of those "a-ha moments" when they figured something out, or realized a mistake, or otherwise reached a turning point in the plot line of their solving the problem. We ask them to highlight some change in their code that represents this, and to explain briefly what that "epiphany" was. This metacognitive piece is like the "Making Of" director's commentary on a DVD. You have the movie, then you have the commentary. The narrative is their commentary. It doesn't take them too long to do, but it raises awareness of what they are learning while they work on these problems.

This way, we grade what is most important — the process of their learning and their mistakes and corrections, not finished code that is ultimately perfect.

Another significant way to assess whether students are doing what matters is by asking them to design projects that serve a purpose or solve a specific problem in Minecraft. Designing and programming is hard work. However, defining a problem and creating multiple iterations of solutions to test out is good learning, and a task that is accessible to kids even if they are not programmers right from the start. This also encourages students to work together and collaborate in giving each other useful feedback.

Starting to think about projects from a design standpoint can also motivate kids to learn more about programming in order to make the project for themselves. If they don't have the skill set to code the project themselves, they may be able to team up with someone who has coding skills and is looking for a good idea (or help with the Minecraft building aspects of the project.) Learning the value of empathizing with another person or organization, identifying a need, and designing an authentic solution through continuing feedback and iteration is a skill set that is tremendously valuable in many contexts.

Final Project Assignment: Survival Backpack

The final project is a chance for you to use all of the skills you have been learning throughout the semester to create several code projects that will help you in a Minecraft world.

Imagine you are going into a brand new Minecraft world. What are your most immediate needs? Brainstorm with a classmate.

Some examples:

- Collect wood
- Find wool
- Find food
- Build a house
- Mine for coal

Then, choose one or more of those needs that can be solved with code. You will get to take your Agent with you, as well as the Builder and the other powerful tools in your MakeCode inventory, but you are limited to just a single project. What are the top four commands that you would want to have?

In addition, your project code must do both of the following things:

1. Show something you already know
   You should demonstrate your knowledge of one or more concepts we have covered in these lessons.
2. Show something new
   You should demonstrate a technique, efficiency, or code block that you went out and learned how to use on your own, either from the documentation, from experimenting, or from another classmate.

Timeframe: Three weeks of in-class work and activities. Code should be complete by the end of the second week, so students can use the last week for a beta testing period with other students.

Due each week:

- 2–3 work logs
- 1 Record of Thinking

Some possible ideas:

Need	Solution
Entertainment	Create a mini-game
Lots of paper	Create and harvest a sugar cane farm
Better tools	Create an iron-finder

Safety	Create an automatic shelter
Art	Create something beautiful using code

Final Deliverables:

- Work Logs
- Record of Thinking entries
- Final Project Code
- Project Narrative
- Video of your project running
- Final project showcase and celebration at the end

Assessment:

- 50% Process (initial proposal, work logs, records of thinking, final narrative)
- 50% Product (project code component, video of code execution)

Teacher Note: This form of assessment places just as much weight on documenting the process of designing the project, as it does on the finished product itself. This is because in my classroom I want to prioritize "sustained effort over an extended period of time" over a project that might have resulted from three all-nighters in the final week it is due.

However, you may decide to assign more or less weight to each of these pieces, and you should certainly feel free to scale up or down the documentation piece as appropriate for your classroom, grade level, and teaching priorities.

While Working on the Project

The expectation is that students are working steadily on their independent projects for three weeks, testing out ideas, trying different things, getting stuck, and getting unstuck. Because everyone is working on a different project, we can't assign the same homework to everybody, so besides the project work itself, students are also responsible for documenting the work they are doing on the project using work logs, and reflecting on the process of their learning in a record of thinking. Here are more details on these work items:

1. Work Log
   A work log is a short, bullet point list of what you worked on, and how long it took. Stick to just the facts. It shouldn't take more than thirty seconds or so to write up a work log. Students should do one work log entry for every class period, several times a week. A shared Microsoft OneNote notebook is a great way to keep a work log that students can update

regularly. Alternately, you might use a collaborative shared document, or your classroom management system, or even e-mail.

Sample Work Log:

Backpack Project #1: Cave Exploration System
3/31: Worked on getting Builder to navigate uneven hallways (45 min.)
4/1: Added Say blocks at each point so we could stop it from getting lost all the time (30 min.)
4/3: Completed exploring simple caverns, now working on more complex systems (45 min.)
4/4: Coded for multi-level caves, spawner detection system (30 min.)

Teacher Note: We generally don't accept late work logs. If a student simply didn't have time to do any work on their project, he/she should still file a work log, and report that no work got done. Work logs are worth a few points each, so missing one or two isn't a problem, but if it happens a lot it's usually time to do a check-in with that student and see where he/she is with the project.

2. Record of Thinking

   A Record of Thinking is like a journal entry (or like the Minecraft Diary you created for each mini-project) that tells the story of your learning throughout the past week. Go through your work logs for the week and look at what you did, where you got stuck, and how you figured it out. Then write a 150- to 300-word Record of Thinking essay each week of the project addressing the following:

   - Describe something that surprised you this week as you worked on your project.
   - Describe a moment where you go stuck. How did you get unstuck?
   - Did anyone help you this week? Who and how?
   - Choose an adjective that describes how you are feeling about your project this week. Explain why you chose this word.
   - What are you working on next week? (for weeks 1 and 2)
   - If you had more time to work on this project, what would you add? (for week 3)

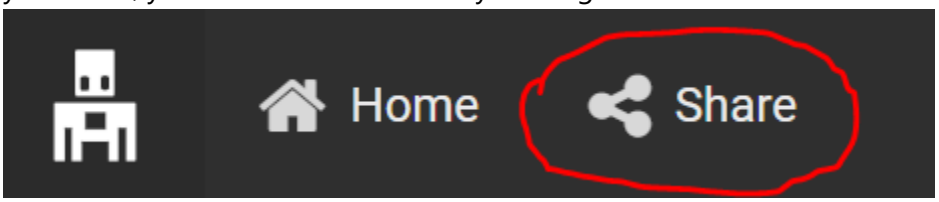
   Sample Record of Thinking Excerpt:

   Week of April 6
   I finally figured out why my code wasn't working. I had the wrong for loop nested inside the other one, and so it was executing way more times than it was supposed to, and that was why the Agent kept going off track. Once I created another variable to keep track of the number of times the Agent came back to the Home row, and checked this variable each time during the loop, everything worked the way it should. I only was able to figure this out because I took each function apart separately and ran it without that function in order to track down where the problem was. That was useful. I have to remember to try this first next time.

Teacher Note: A Record of Thinking is *not* an expanded work log! Students will sometimes just write a more detailed list of all of the tasks they completed over the week, and that's not the point of the Record of Thinking. The Work Logs are to show WHAT you did. The Record of Thinking is to show HOW you learned how to do it. Unlike Work Logs, I will accept late Records of Thinking as long as they come no later than the due date for the next week's Record of Thinking. It is an important form of documentation of the learning process.

Turning in the Final Project

When you turn in the final project, you should turn in your code, and a final narrative. To turn in your code, you can Share the code by clicking the Share button in the top menu bar:



You acknowledge that you consent to share your project by clicking Publish project, and then MakeCode will display a unique URL that you can copy and paste to submit as your project code.

You also need to create a written final narrative to accompany your code. You have worked for the past three weeks to propose, design, and test an original Backpack of Minecraft Code as your final project. I am looking for an honest, accurate assessment of your work over this time.

Please go back and read through all of your Work Logs and Records of Thinking. Then, compose a comprehensive narrative that tells the story of the development of the various projects in your Backpack, and your progress toward your goals along the way. How you tell the story is up to you, but you might consider following most, if not all, of the following questions:

- How did you start the process of designing the project and meeting your goals?
- What did you hope to learn?
- What challenges did you face? How did you overcome them?
- What was the outcome?
- What did you learn in the end?
- Who in the class provided help to you along the way? How?
- What were you proud of?
- What would you do differently next time?

Throughout your narrative, you must cite evidence from your work logs and records of thinking (e.g., Record of Thinking 4/17, Work Log #3, Conference notes, etc.) You may use footnotes for this or add it in parentheses after the material you are citing.

I will read this carefully and grade it along with your final project code and average it with the total of your work logs and records of thinking to come up with your final grade for the project.

Sample Final Narrative:

It's clear to me now that in the second week, I was a little lost and confused with the direction my project was taking. I can see now that in my chats with Mr. Kiang and with classmates (Conference Notes 4/3) I was not being very precise in my questions, and I didn't totally understand what he was explaining.

Looking back, one of my goals was to meet more regularly with my table mates. Hopefully I would be a lot less confused and at least this time when I got stuck, we would be able to solve it together. I wrote about this in my Record of Thinking (Record of Thinking #2) but I am surprised that things cleared up for me so quickly once we did start meeting together. This allowed me to get past something that was really bothering me, specifically adding and removing things from an array, and I was able to complete that in less than a day after having been stuck for more than a week (Work Log #4).

Once I started to get a little more clear on what to do, I was able to get more effective help from my classmates. Specifically, Jordan helped me a lot with figuring out how to use the different mask options with the clone block . He also showed me some helpful MakeCode sample projects and examples. I think if I could do this over again, I would have scheduled more time earlier to meet with Mr. Kiang and/or found a better way to share the different bits of code I was working on with my table mates because we all were trying to figure out basically the same things, only in different projects. I didn't even find out until the end that you could jump into JavaScript to make changes to the code, and it makes it all with the right blocks when you go back! (Beta Testing notes) That would have saved me a lot of time.

In addition to the project narrative, please provide a short (less than 2 minutes) video of your code in action.

Beta Testing and Final Showcase

Beta testing is an important part of testing the final projects to uncover bugs or design issues that could make the projects difficult to use. One way to test the projects is to ask all students to come in to class on a specific day with the URLs of their projects ready to test, and give their

classmates a chance to test the code in their own Minecraft worlds. This is not the final deadline, but projects should be "feature-complete" i.e., all features need to be incorporated into the code, and the construction of any Minecraft elements of the project needs to be done or almost done.

Students can take turns presenting their projects to the entire class, or they can work in pairs to take turns trying their partner's project out and offering feedback. Students who are being critiqued should take beta testing feedback notes and turn them in as part of their final project narrative.

Final Showcase

Have a celebration of your students' hard work and hold an event at your school for parents, administrators, and other community members to appreciate all of the hard work that went into making each of the final projects.

We have found that a "science fair" format works nicely, with students sitting at tables where they can demonstrate their projects, show the videos, and answer questions. You might also consider setting up a YouTube channel where your students can publish walkthroughs of the various MakeCode for Minecraft code examples they have come up with, and publish URLs to the shared code, as well. Either way, a little public recognition of all of your students' hard work goes a long way!

Assessment

	1	2	3	4
Code (Show what you know)	Code doesn't demonstrate use of previous concepts, and/or variable names are unclear or ineffective, a number of places where code could be improved.	Code mostly ineffectively demonstrates the use of previous concept(s). Variable names may be unclear. Code could be more efficient.	Code somewhat effectively demonstrates the use of previous concept(s). Variable names are generally unique and clearly describe what information values the variables hold. Code is	Code very effectively demonstrates the use of previous concept(s). Variable names are unique and clearly describe what information values the variables hold. Code is highly efficient.

			reasonably efficient.	
Code (Show something new)	Code doesn't demonstrate any new concepts, and/or variable names are unclear or ineffective, a number of places where code could be improved.	Code mostly ineffectively demonstrates the use of new concept(s). Variable names may be unclear. Code could be more efficient.	Code somewhat effectively demonstrates the use of previous concept(s). Variable names are generally unique and clearly describe what information values the variables hold. Code is reasonably efficient.	Code very effectively demonstrates the use of new concept(s). Variable names are unique and clearly describe what information values the variables hold. Code is highly efficient.
Work Logs	More than two late or missing work logs and/or not accurate nor sufficiently detailed.	Two late or missing work logs and/or work logs not accurate nor sufficiently detailed.	One late or missing work log and/or work logs not accurate nor sufficiently detailed.	All work logs submitted on time, and accurate
Reflection	Reflection piece lacks 3 of the required elements.	Reflection piece lacks 2 of the required elements.	Reflection piece lacks 1 of the required elements.	Reflection piece describes: • Development Process • Something new • Something proud of • Future mods