COMPUTING WITH MINECRAFT: Overview

education.minecraft.net

Overview

This Computing with Minecraft course introduces students to core computer programming concepts and computational thinking skills. It builds on the popularity of Minecraft, featuring Minecraft Education Edition and the Computing with Minecraft dev world, and introduces Microsoft MakeCode, a block- and JavaScript-programming editor in the free Code Connection app.

The course is comprised of five units that combine a variety of computer programming skills in a structured, linear approach to build a city with road networks, buildings, a park, a zoo, and a wind farm in their Minecraft world—all through coding. Most units include three lessons (of approximately 45-60 minutes each) with guided coding activities to gain hands-on coding experience, knowledge check questions to assess their learning, and optional activities for students to apply what they've learned. Teaching all units and lessons will be approximately 19 hours of instruction. Educators who have never taught computer science before are encouraged to incorporate this course into their curriculum, regardless of their subject area. The coded city features can be adapted to cover a broad range of curriculum subjects.

About Minecraft

What is Minecraft? It's a game about placing blocks and going on adventures, set in infinite worlds of wide open terrain. It extends up into the hills and mountains, and down into dark caverns filled with lava and treasure. Just like a sandbox, there are no specific goals in Minecraft, no set objectives to fill. The fun comes from setting your own goals and deciding how you want to play on a given day. Maybe you will head into the jungle in search of a jungle temple. Perhaps you will build your dream tree house with a perfect view of the ocean, or mine deep underground in search of emeralds and diamonds. Whenever one task becomes too difficult, or boring, or when you get stuck and need some help or inspiration, you can just choose to do something else. There is always something engaging to do, and no matter what you choose to do, you will always learn more about the world around you and be thinking about better and more efficient ways to accomplish what you want to do.

Minecraft: Education Edition offers special features for educators such as easy tutorials, different worlds, classroom management tools, secure sign-in, classroom collaboration and tons of sample lessons, plus a global network of mentors and tech support. Educators in grades K-12 are using Minecraft: Education Edition to teach a range of subjects, from history and chemistry to sustainability and foreign languages, and can map lessons directly to specific learning outcomes and curriculum standards. Minecraft: Education Edition helps prepare students for the future workplace, building skills like collaboration, communication, critical thinking, and systems thinking. The open learning environment gives students the freedom to experiment, encouraging creative self-expression and problem-solving. Code Builder is a feature that allows students to learn coding in Minecraft using tools including Tynker, and Microsoft MakeCode. Students can use blocks of code or JavaScript to build and create in Minecraft. Minecraft Hour of Code also offers a free, one-hour introduction to coding basics. Many students are already accustomed to how Minecraft works normally. The prospect of using code to automate tedious tasks such as mining or chopping trees is immensely motivating to Minecraft players and the ability to immediately see the results of the code you wrote in the Minecraft world is incredibly powerful. Students can use a flat or other world to start out with, as a "sandbox" for their code, and as they learn and experiment, they will be able to see the progress of their learning visually in their world as a series of different physical projects and constructions.

About block coding and Microsoft MakeCode

In 1975, Seymour Papert of the MIT Media Lab created a beginners' programming language called LOGO. He developed it based on research that showed that playing with blocks of code was a particularly effective way to teach programming concepts. Papert coined the term "constructionism" to describe the way that learners construct new knowledge by building on established knowledge. The blocks in MakeCode and the blocks in Minecraft are themselves models for the way that new learning happens through the application of concepts in an open-ended learning environment. Block-based programming languages such as Scratch and MakeCode build on Papert's

research and are a great way for students to start learning about coding concepts without having to worry about syntax.

Audience

This course targets students ages 8-11, and can also engage more advanced students of ages 5-7. It is designed to accommodate participants without any Minecraft or coding experience.

Learning goals

By the end of this course, students will be able to:

- Describe what coding is.
- Apply the following programming concepts as block code and/or JavaScript in Microsoft MakeCode to build a city in Minecraft:
    - Event handlers
    - Coordinates, absolute world positions, and relative player positions
    - Loops
- Create multiple lines of code with an intended outcome.
- Merge multiple types of coding features to create one outcome and complete large, varied tasks.
- Understand how code can be used to create large-scale actions that would manually take longer.
- Merge mathematics with code to create calculated in-game actions.
- Recognize more than one way to achieve the same result through code.
- Focus on the design aspect and merging this with code for a unique outcome, showing that code is commanded by design first.
- Use code to design a moving system with animation to a practical, subject-based topic.

Contributors

Sponsored and published by Microsoft, this curriculum content was authored by Stephen Reid. The course materials were produced by Prime 8 Consulting for Minecraft Education.

About the curriculum author

Stephen Reid is the Mind behind Immersive Minds. He has been pioneering the use of technology as a tool for learning for almost twenty years, and has dedicated his career to exploring, piloting and advocating its use in schools around the world. As Director and Creative Genius, he leads the Immersive Minds team of educators, coders and builders in creating innovative, transformative learning experiences and resources. Stephen is a Microsoft Innovative Educator Expert, and you can follow him on Twitter at @ImmersiveMind.

Bill of Instructional Materials

In addition to this overview guide, the following instructional materials are included with this course.

Document type	Description
Educator guides (5)	A guide is provided for each unit and should be used for preparation and as a reference while delivering content. They include: •A high-level overview of the unit and associated lessons, learning goals, and addressed computer science standards. •Required educator preparation tasks, skills to complete the activities, and resources to master the covered concepts and skills. •Lesson plans, outlines, suggested tasks for before the lesson, lesson details with activity instructions, screenshots, and knowledge check questions, and post-lesson tasks.
Unit presentations (5)	Each unit is supported by its own PowerPoint presentation to provide structure and guide the educator through the lessons and extensions of that unit.
Student workbooks (6)	Each unit includes a workbook that aligns to the unit structure and includes: •Overview, objectives and activities map •Featured coding activity steps, tips, and space to take notes •Unit extension activities •Glossary of terms A workbook of the optional coding challenges is also provided.
MakeCode files (17)	These files include the final solutions for the lesson coding activities in units 1-5 (except unit 1, lesson F, and unit 5, lesson A). The files can be imported to Microsoft MakeCode to see the lesson coding solutions in code blocks or JavaScript.
Minecraft world files (2)	Two Minecraft: Education Edition world files are included for you and your students to play in during this course: •Computing_With_Minecraft_Dev world shows the landscape and signs suggesting the best locations for coding the city structures and features for each lesson. This is the main world you and your students will use to complete the lesson learning experiences. •Computing_With_Minecraft_Full world shows the completed city with all the structures and features from all unit lessons carried out. This world can be used to show your students the what the city could look like after completion of all units.
Standards alignment guide (1)	Summarizes the CSTA K-12 computer science standards addressed in the course and throughout each unit and associated lessons.

Required Educator Preparation

Preparing to teach the course

Educators should aim to meet the following learning objectives before leading the first unit.

Master the course materials

- Begin with a close review of this document.
- Review the Unit 1 materials in detail and look over the remaining units to get a sense of the course.
- Practice the presentations, demonstrations, and coding activities for each unit ahead of time and anticipate learning challenges and opportunities students can encounter.

Understand the coding concepts

This course introduces students to core computer science and coding concepts. Reviewing the instructional materials for each unit (educator guide, presentation, and student workbook) and practicing the lesson activities will explain each unit's concepts. In addition:

- Reference the Glossary and Recommended resources sections of this document, as needed, as well as any unit-specific resources provided in the educator guides.

Know your tools

Throughout the course, the educator and each student will each use: Minecraft: Education Edition, the Computing_With_Minecraft_Dev world. While you don't need to be a master at Minecraft or MakeCode, you do need to know enough to present the content, learn with your students, and help students troubleshoot. The educator will also have the MakeCode files and Computing_With_Minecraft_Full world for the coding solutions and resulting structures and features of the city in Minecraft.

- Upload both Minecraft worlds and the MakeCode files to your educator device, following the instructions in Required device preparation section on the next page.
- Get familiar with Minecraft: Education Edition:
    - If you are not familiar with playing Minecraft, it's highly recommended that you start with My Minecraft Journey, a comprehensive collection of resources meant to get an educator brand new to Minecraft up to speed with playing the game and leveraging it in the classroom. Once you have completed the My Minecraft Journey course, come back and continue with these activities. You can access My Minecraft Journey as a:
    - OneNote notebook on the Minecraft: Education Edition website at https://education.microsoft.com/courses-and-resources/courses/my-minecraft-journey, or
    - Series of courses on the Microsoft Educator Community at education.microsoft.com/courses-and-resources/courses/my-minecraft-journey.
    - See the recommended resources on the following page for additional support
- Get familiar coding in Microsoft MakeCode:
    - Start by watching the video tour of MakeCode for Minecraft (4:54) found at minecraft.makecode.com/about.
    - Then follow the steps for the coding activities for the lessons in Computing with Minecraft 1: The Agency to start coding.
- You can check your work in two ways:

        - Check your code for any lesson against the completed code from the appropriate MakeCode file.
        - Check any structures or features that build when you run the code in the dev world in Minecraft against the already-built structure or feature in the full world.
- To get additional coding practice, consider taking a couple of the tutorials in the home page of MakeCode, such as **Chicken Rain**, **Agent Moves**, or **Agent Build**.

Required device preparation

For all devices

Complete these tasks to set up each device before teaching the first unit:

- Install Minecraft: Education Edition from aka.ms/download.
- Download the **world** file from the website or from the in-game library.
- Ensure you and your students have licenses for Minecraft: Education Edition or use the free trial.
- When prompted, use your Office 365 Education login to sign in.
- Press 'C' to start in-game coding experience.

Recommended resources

If you would like additional support to your initial preparation, please review the following resources.
Minecraft: Education Edition

- Start by searching the knowledge base of the Minecraft: Education Edition website at aka.ms/meekb.
- Contact the support team directly at aka.ms/meesupport.
- Engage with the educator community at education.minecraft.net/conversations/.
- In addition to Minecraft Teacher Academy learning path, you can learn from the Tutorial Worlds found in the in-game library to take a guided tour, learn the controls and crafting necessary to play Minecraft, and experience some of what the game has to offer.

Computer science, coding and Microsoft MakeCode

- For a range of resources about computer science, go to microsoft.com/digital-skills.
- To find additional information on coding concepts, search the Code.org YouTube channel.
- To learn more about the block coding language used in Microsoft MakeCode, see the resources at minecraft.makecode.com/blocks.

Transform learning with Minecraft
Get more ideas to transform learning with Minecraft and optimize technology in your classroom:

- Join the Microsoft Educator Community at education.microsoft.com/.
- Watch the videos at education.minecraft.net/how-it-works/why-minecraft/.
- Find additional training resources for educators at education.minecraft.net/class-resources/trainings/.

Course Delivery

Unit summary and sequencing

The following table summarizes the units and lessons in the suggested teaching sequence.

Unit description	Learning goals	Associated lessons (45-60 minutes each)
Unit 1: The Agency An introduction to the most basic of common coding features and the Agent, an in-game assistant you'll use throughout the course.	Describe what coding is. Launch and connect Microsoft MakeCode to Minecraft: Education Edition. Maneuver their player in Minecraft and use inventory. Use on chat commands. Code their Agent to move in different directions, use their inventory, and build a bridge in Minecraft.	Lesson 1: Get familiar with your Agent Lesson 2: Code your Agent to destroy and collect Lesson 3: Code your Agent to build
Unit 2: City planner A chance to use what you have learned and some new coding tools to build an entire city, one road, building and line of code at a time.	Create multiple lines of code with an intended outcome. Merge multiple types of coding features to create one outcome. Understand how code can be used to create large-scale actions that would manually take longer. Use block code and/or JavaScript as a coding language to change their Minecraft world.	Lesson 1: Code a road network Lesson 2: Code a building Lesson 3: Code a row of houses
Unit 3: Parks and recreation Learn more code and reinforce what you already know to create a small park and fountain in your city.	Use loops to code more efficiently. Describe how loops affect code and in-game actions. Merge mathematics with code to create calculated in-game actions. Use block code and/or JavaScript as a coding language to change their Minecraft world.	Lesson 1: Code a park fence Lesson 2: Code a water feature Lesson 3: Plant some flowers
Unit 4: A zoo Create a zoo for city dwellers to visit using only code, including a gate, signs and animal enclosures.	Recognize more than one way to achieve the same result through code. Focus on the design aspect and merging this with code for a unique outcome, showing that code is commanded by design first. Merge multiple coding types to complete large, varied tasks. Use block code and/or JavaScript as a coding language to change their Minecraft world.	Lesson 1: Code a zoo entrance Lesson 2: Code zoo paths Lesson 3: Code animal enclosures

Unit 5: Wind power Code a wind farm, including animated turbines and an electrical resource to power your city.	Use code to design a moving system. Make multiple moving parts to create an animated feature. Apply the code for animation to a practical, subject-based topic Use block code and/or JavaScript as a coding language to change their Minecraft world.	Lesson 1: Code a wind farm Lesson 2: Animate the wind farm Lesson 3: Code a lighting system

Activity summary

Three types of activities are included in the curriculum to reinforce different concepts, skills, and opportunities for creativity.

Featured coding activities

Each lesson revolves around a guided coding activity to build or enhance a city structure or feature in the world. Each unit educator guide includes step-by-step instructions and screenshots, and world includes tutorials that students can access and follow.

Unit extensions

Each unit includes four optional coding activities to ramp up the difficulty of any lessons, as needed. These activities are tailored to the unit's theme but do not include coding steps, screenshots or solutions as they are intended to help students apply what they've learned in the featured coding activities in new ways. The activities are described at the end of each unit educator guide, student workbook, and in the last four slides of the PowerPoint presentation.

Consider using the extensions in one of the following ways:

- If you finish a lesson early, have the class do an extension coding activity as a group.
- Alternatively, you could assign an extension to individuals or small groups to work on during class. Reserve time before the wrap-up section for students or groups to share their coding progress with the rest of the class.
- If you have students who are excelling with the coding skills of the featured coding activity and seem disengaged during a featured coding activity, assign one of the extensions to challenge them. Reserve time at the end of class for them to share their coding progress with the rest of the class.
- Assign to students as a homework or computer lab assignment before starting the next unit.

Coding challenges

Four optional, open-ended coding challenges are included in the curriculum to challenge the students to apply their learning on a larger scale and encourage them to explore other Microsoft MakeCode functionality. These activities are not unit nor lesson specific and don't include coding steps, screenshots or solutions. The challenges are described in Appendix A of this document and within the Coding challenges workbook for distribution to the students.

Consider using the challenges in one of the following ways:

- If you have students who are excelling with the featured and extension coding activities, assign one of the challenges to re-engage them.
- Assign to students as a special project, homework, or computer lab assignment after completion of Units 4 or 5. Students could do presentations of their coding solutions and execution in Minecraft.
- After completion of the five units, you could do each challenge as supplemental lessons. The activity could be done as a group, depending on the number of students, or assigned to an individual or small

groups. Again, students or small groups could present their coding solutions and execution in Minecraft.

Assessment summary

Learning assessment questions are provided for each unit and its associated lessons. These are provided with the curriculum for your convenience. Feel free to use those that are most helpful to you, or to tweak them for your class.

- Warm-up: Prompts to write on the board for students to think about as they arrive and get settled before class starts. They are intended to reinforce concepts, encourage students to link prior knowledge to the day's concepts, and/or preview the day's lesson.
- Knowledge check: Unit specific discussion questions to quickly assess students' learning after relevant activities during the lesson. They are also included in slides in each unit PowerPoint presentation.

CSTA Standards

The following table lists all standards addressed in the course by age group. For more detailed information about the standards addressed in each unit, please see the Standards alignment guide.

Ages 8-11 (grades 3-5)
1B-AP-08 Compare and refine multiple algorithms for the same task and determine which is the most appropriate. 1B-AP-10 Create programs that include sequences, events, loops, and conditionals. 1B-AP-11 Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process. 1B-AP-12 Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more features that are advanced. 1B-AP-15 Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended. 1B-AP-17 Describe choices made during program development using code comments, presentations, and demonstrations.
Ages 5-7 (grades K-2)
1A-AP-10 Develop programs with sequences and simple loops, to express ideas or address a problem. 1A-AP-11 Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions. 1A-AP-14 Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops. 1A-AP-15 Using correct terminology, describe steps taken and choices made during the iterative process of program development.

Glossary

he following table includes coding and Minecraft terms used in the course, listed alphabetically.

Absolute Player Position	A position or coordinate (X, Y, Z) that represents the distance from the Minecraft world origin of (0, 0, 0). This type of position is fixed and does not change, regardless of where the player currently is in Minecraft.
Agent	A robot character that helps the player learn coding skills and follows commands written in Code Connection.
Algorithm	A set of (often repeated) steps used to solve a problem. The set of steps for doing long division of numbers is an algorithm.
Array	A series or collection of places to store things, similar to many variables in one place. An object in an array collection is referred to as an **element** or item. An individual element can be referenced by its **index.**
Array Length	The total number of items in the array collection.
Array Sort	The order of items in the array collection (i.e., by date, price, name, color, etc.)
Array Type	The type of item being stored in the array collection (i.e., comics, coins, cards, etc.)
Assign	To set the value of a **variable**.
Artificial Intelligence	Intelligence demonstrated by machines. Also known as AI.
Biome	Refers to different geographic regions within Minecraft. Each has its own climate and mobs.
Block programming	A programming language found in coding editors—such as Microsoft MakeCode and Scratch—that uses different colored and shaped blocks that connect together in specific order to allow beginners to learn about coding concepts without having to worry about syntax.
Boolean	A **variable type** that can be either true or false. A **Boolean condition** is a condition that evaluates to either true or false.
Builder	The Builder is a tool that makes it easier to build complex structures in the game. It's an invisible cursor (or marker) that a user can be moved around in the Minecraft world and place multiple blocks at once in the game world, based on the builder's trajectory and marks. Also see **marker**.
Classroom Mode	Companion app for Minecraft: Education Edition that gives educators additional abilities to manage their students within the game.
Code Connection	An extension for Minecraft: Education Edition that allows educators and students to explore, create, and play in Minecraft all by writing code.
Computer Program	A set of instructions that a computer can follow. Also called 'code'. Apps and games are examples of computer programs.
Conditional Statement	Also known as an IF THEN or IF THEN ELSE statement. The part of a computer program or code that tells a computer **when** to perform an action.

Coordinate(s)	A coordinate represents a position or location. Coordinates tell a computer program **where** an action should take place by providing the location for the action. Also see **position**, **X position or coordinate**, **Y position or coordinate**, and **Z position or coordinate**.
Creeper	Green-skinned mob in Minecraft that explodes if it gets close to the player.
Debug	The process of correcting errors within a program. i.e. The process of removing "bugs" from a program.
Declare	To create a **variable**. Also see **initialize**.
Element	An individual item in an **array**.
Event	Something that happens outside a program (like a screen tap or mouse click) that the program can respond to.
Event Handler	Part of a program that runs when a specific event happens (it "handles" the event). In MakeCode, an event handler block looks like a square with a gap in the middle and usually starts with the word "on."
for Loop	A programming construct that allows for a block of code to be executed a specified number of times.
Function	A self-contained set of instructions for performing a specific task within a **computer program**. Most objects have multiple functions associated with them.
Gamemode	Refers to the type of gameplay in Minecraft. In creative mode, players have unlimited access to all of the blocks in the game. In survival mode, players must search the world for resources.
Index	A numerical value that corresponds to a unique element in an array. Index values start at zero, so the first element in an array has an index value of 0.
Initialize	To set the value of a **variable** for the first time.
Iteration	Code that tells a computer to repeat sets of instructions under different conditions.
Helper Function	A function that performs part of the computation of another function. It usually has a specific purpose and can be/is used in multiple places within a **computer program**.
JavaScript	A text-based programming language using letters, numbers, and symbols. It's one of the most popular programming languages in the world.
Loop	In general, a programming construct that allows for a block of code to be repeated multiple times. See **for loop** and **iteration**.
Loop Counter	The variable used in a **for loop** to determine the number of times the loop will execute.
Marker	Another term for the invisible cursor or pointer of the **Builder** tool. The user sets the marker coordinates or position in the Builder coding block.
Mob	Short for "mobile," mob refers to creatures in Minecraft.
Nether	Accessed via a nether portal, this alternate dimension is full of lava and unique mobs.
NPC	Non-Player Character. Can be used to dispense information, run commands, or direct students to outside web links.

Number	A **variable type** that holds numerical data.
Object	A fundamental building block for any **computer program**, designed to hold data and allow for manipulation of that data through **functions** and **properties**.
On Chat Command	Code that is triggered or happens when you type the appropriate command in the chat window of Minecraft.
Parameter	Data passed to a **function**.
Position	Coordinates in Minecraft and Microsoft MakeCode that identify a three-dimensional location in Minecraft as **(X, Y, Z)**. In Minecraft, **a position** is the entire 1 x 1 x 1 space that a block occupies. Also see **absolute player position** and **relative player position**.
Program	See **computer program**.
Project Malmo	A Minecraft mod that allows computer scientists to use the Minecraft worlds as a testing ground to improve their artificial intelligence. Part of a Microsoft AI research project.
Recursion	A special form of iteration in which a function calls itself during its execution. This enables the function to repeat itself several times, outputting the result at the end of each iteration.
Redstone	Mined from Redstone ore, its dust is used to power circuits and machinery in Minecraft.
Relative Player Position	A position or coordinate (~X, ~Y, ~Z) that represents the distance from the player's current position as the origin of (~0, ~0, ~0). This type of position changes as the player moves around in Minecraft.
Skin	The appearance of a player's avatar in Minecraft. Can be selected from the main menu, and customized by players using digital art tools. Steve and Alex are the default skins.
Slash command	Entered in the game's chat window by typing **T** on the keyboard, these cheat commands allow for the control of game features such as time of day, weather and giving out blocks.
String	A **variable type** that holds a sequence of alphanumeric characters and/or symbols.
User Input	The information or data given to the computer by the user, typically with a keyboard, mouse, or gamepad that is used in the program.
Variable	A container for data. Every variable has a name that is used to reference the data that it contains. Every variable also has a **variable type**.
Variable Type	The type of data that a **variable** can contain. Examples of variable types are *number* and *string*.
Variable Scope	The part of a program where a variable can be read. For example, a variable declared in one function is said to be a *local* variable and cannot be read from other functions. However, variables can be declared with a *global* scope, making them readable in all functions of a program.
WASD	Common control scheme for games on Query keyboards that allows the right hand to be used to control the mouse.
X position or coordinate	The distance along the horizontal plane from east to west in Minecraft. Analogous to longitude values.

Y position or coordinate	The distance along the vertical plane up and down in Minecraft. Analogous to altitude values.
Z position or coordinate	The distance along the horizontal plane from south to north in Minecraft. Analogous to latitude values.

Copyright

Appendix A: Coding Challenges

These challenges are included in the Coding challenges workbook for distribution to students.

Challenge 1: Providing hydropower

Starting coordinates: **43, 80, -374**

By order of The Agency, this land has been designated for a hydropower dam. An expansion of these premises is needed to provide power to the city.

Your challenge is to code a working dam.

- Use your knowledge from Units 1-5 and explore other functionality in MakeCode to create your own hydropower dam.
- Think about how to block the water flow and use **Redstone** to turn it on and off.

Challenge 2: Feed the world

Starting coordinates: **39, 70, -455**

By order of The Agency, this land has been designated Food Development and Supply. An expansion of these premises is needed.

Your challenge is to code an agricultural dome.

- Use your knowledge from Units 1-5 and explore other functionality in MakeCode to create your own food supply system as part of the needed Agency expansion.

Challenge 3: Save the trees

Starting coordinates: **-50, 65, -510**

By order of The Agency, this land has been designated a sustainable forestry area.

Your challenge is to code a sustainable forestry system.

- Use your knowledge from Units 1-5 and explore other functionality in MakeCode to create your own sustainable system in which a new tree is replanted in place of any tree that is cut down.
- Try to maintain a balance and do not cut down more than you plant. Remember, trees take time to grow.

Challenge 4: Control the flood

Starting coordinates: **117, 64, -505**

By notification from The Agency, this area is known to be a flood risk due to rising tides.

Your challenge is to code a flood, as well as the cleanup of the damage.

- Use your knowledge from Units 1-5 and explore other functionality in MakeCode to create your own flood of the beach and run the code.
- The higher the flood, the farther back it will flow and the more damage it will cause.

Then code the cleanup of the damage and run that code.