CODING WITH MINECRAFT: Overview

education.minecraft.net

Overview

This Coding with Minecraft course introduces students to core computer programming concepts, such as events, coordinates, variables, conditionals, functions, iteration, arrays and artificial intelligence. It builds on the popularity of Minecraft, featuring Minecraft Education Edition and introduces Microsoft MakeCode, a block- and JavaScript-programming editor.

The course is comprised of 10 units that combine a variety of computer programming skills in a structured, linear approach through coding tutorials. Most units include four 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 30 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.

Audience

This course targets students ages 14-17. It is designed to accommodate participants without any Minecraft or coding experience.

Contributors

Sponsored and published by Microsoft, this curriculum content was authored by Michael Braun, Travis Landers, Douglas Kiang, and Mary Kiang. The course materials were produced by Prime 8 Consulting for Minecraft Education.

About the curriculum authors

Douglas Kiang is a speaker, teacher, and workshop presenter with twenty-seven years of teaching experience in independent schools at every grade level. He currently teaches high school computer science at Punahou School in Honolulu, Hawaii. Douglas holds a master's degree in Technology, Innovation, and Education from Harvard and is a Microsoft Innovative Educator. You can follow him on Twitter at @dkiang.

Mary Kiang has been teaching for over twenty-five years at elementary, middle, and high school levels. She also developed curriculum in the Education Department of the Museum of Science in Boston. She currently teaches 6th grade Math/Science at Punahou School. Mary is a former programmer for Houghton Mifflin and Dun & Bradstreet and holds a Master's degree in Elementary Education from Simmons College. Mary is the founder of GO Code!, an organization that supports girls and young women in exploring coding and STEM.

Michael Braun has a Master of Education (M.Ed.), Educational Leadership and Administration (P–12 Principal Certificate) and is the founder of Makers of STEAM. Michael's innovative approach to teaching has been highlighted by New York Times, Seattle Times, and GeekWire. Washington State's Governor Jay Inslee also recognized Michael's courses in technology. Michael has been a part of the largest physical computing education initiative in the world. Michael has been invited to lead STEAM "Teacher Training" workshops across the United States and abroad. Michael is a Microsoft Innovative Educator. You can follow him on Twitter at @michaelebraun.

Travis Landers has a Master of Education (M.Ed.). Travis' experiences in building websites and using various programming languages have given him the tools to create a learning environment that is fun, applies tested methodologies, and uses an intelligent curriculum design procedure. Travis is always looking for ways to use technology to make teaching more fun and make processes smoother. To name just a few of his many projects, Travis has designed and supported electronic grade systems, android games, websites, video editing, and much more. Recently, he developed a complex architecture that automates video creation for project learning in his school.

Bill of Instructional Materials

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

Document type	Description
Educator guides (10)	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 and any additional materials needed, lesson details with activity instructions and screenshot, and post-lesson assessments and tasks.
Unit presentations (10)	Each unit is supported by its own PowerPoint presentation to provide structure and guide the educator through the lessons of that unit.
Student tutorials and worlds	Each world includes a variety of tutorials for students to follow.
Assessment guide (1)	To communicate expectations and support the educator in evaluating student work and final projects, assessment materials are provided for the course, as well as each unit and its associated lessons: • Formative and summative assessment answer keys and printer-friendly versions to distribute to your students • Project scoring rubrics
Standards alignment guide (1)	Summarizes the CSTA K-12 computer science standards addressed in the course and throughout each unit and associated lessons.
Educator preparation video (1)	A short video is included to provide an overview of the course and assist educator preparation.

Required Educator Preparation

Preparing to teach the course

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

Master the course materials

- Begin with a close review of this document and watch the course overview educator preparation video.
- Review the Delivery options section of this document and consider the classroom time available to determine whether you're teaching all units and lessons, or one of the alternative options.
- Review the unit 1 materials in detail and look over the remaining units to get a sense of the course.
- Review the assessment guide and standards alignment guide.
- Practice the presentations, activities, demonstrations, and tutorials for each unit ahead of time and anticipate learning challenges and opportunities students may encounter.

Understand core concepts and terms

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 and terms. 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

The leader and each student will use Minecraft: Education Edition and Code Builder – Microsoft MakeCode on a laptop, table or PC throughout the course. 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.

- For app installation and setup instructions, see the resources available on the Minecraft: Education Edition website at: https://aka.ms/download ;
- Get familiar with Minecraft: Education Edition through Teacher Academy online training learning path;
- 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 take a couple of the tutorials in the home page of the app to start coding.

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 My Minecraft Journey, you can download the Tutorial World found at education.minecraft.net/worlds/tutorial-world/ 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.

Transforming 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/.

Course Delivery

Watch this video that goes over the course:
https://www.youtube.com/watch?v=p_CooZJYu2s&feature=youtu.be

Activity summary

Each unit includes three types of activities that complement each other by reinforcing different concepts, skills and opportunities for creativity. The activities for each unit are distributed among lessons of approximately 45-60 minutes each.

	Concepts	Skills development	Creativity
Unplugged	High	Medium	Low
Guided or "birdhouse"	Low	High	Medium
Independent project	Medium	Low	High

Type of activity	Description
Unplugged	These activities are a great way to introduce new concepts in a fun way that gets students up and away from their computers and interacting with each other in person. It's important to introduce the concept first so that students have an introduction to the big ideas they will be exploring with Microsoft MakeCode and Minecraft.
Guided or "birdhouse"	These activities provide step-by-step instructions, and every student makes the same thing. Like the birdhouses you may have made in school wood shop, they introduce new skills and provide an easy way to assess student work because you already know what it's supposed to look like at the end. Students can work through these activities at their own pace and implement optional extensions at the end of these activities if they find they have extra time. Guided activities provide high skills development but low creativity because everyone is making the same thing. It's important not to stop after these activities. Go beyond the birdhouse!
Independent project	Each unit culminates in an independent project, an opportunity for each student to "show what you know" by demonstrating the skills they learned in the guided activities. Some of the independent projects are paired independent projects, in which students are encouraged to work with a partner on building something together. Creative projects enable students to apply new skills and concept knowledge in a unique context and provide lots of opportunity for creativity. Try to build in opportunities for students to share their projects with each other, or with the class!

Assessment summary

Learning assessment opportunities are provided for each unit and its associated lessons. For more details about the assessment approach and printer-friendly versions, see the assessment guide.

- Course assessment: Final project
- Unit assessments: Quiz

- Lesson assessments: "Do now," knowledge check questions, and exit ticket

Delivery options

There are several options for delivering this course, depending on the amount of time you have available to incorporate the content into your existing curriculum and the existing skill level and experience of your students. If your students are newer to coding and Minecraft, it's recommended that you start with the first unit as an introduction. Consider one of the following options:

Course content	Approximate hours of instruction	Approximate length of time
All units and lessons	30 hours	One semester
One unit	4-5 hours	One week
Lesson A from each unit	10-12 hours	2-3 weeks
Individual lessons each week	45-60 minutes per lesson	Depends on number of lessons

Suggested unit sequencing

The following table outlines the suggested sequence to teach all units and lessons.

Unit learning goals	Associated lessons (45-60 minutes each)
Unit 1: Introduction • Describe computer science and coding and its importance • Learn to play and maneuver in Minecraft: Education Edition • Understand block programming in Microsoft MakeCode • Change your Minecraft world through coding	Lesson A: Minecraft and Microsoft MakeCode
Unit 2: Events • Describe the different kinds of coding events • Understand the importance of events while playing Minecraft • Understand the importance of events in coding • Learn about real-life events and how they can affect situations • Alter the Minecraft landscape through coding with events • Design an original creative project to change their Minecraft world through coding events	Lesson A: Introduction to events Lesson B: Coding with events Lesson C: Linking events Lesson D: Get creative with events
Unit 3: Coordinates • Describe the difference between relative coordinates and absolute coordinates in real life • Identify your real-world position and calculate the position of a landmark or object relative to your position • Describe the difference between relative player position and absolute world position in Minecraft • Understand the importance of coordinates in coding and while playing Minecraft • Alter the Minecraft landscape by coding with coordinates	Lesson A: Introduction to coordinates Lesson B: Coding with coordinates Lesson C: Automating actions with coordinates Lesson D: Get creative with coordinates

• Design an original creative project to apply your coding skills in new ways	
Unit 4: Variables • Describe the different kinds of coding variables • Understand the importance of variables while playing Minecraft • Understand the importance of variables in coding • Learn about real-life variables and how they can affect situations • Alter the Minecraft landscape through coding with variables • Design an original creative project to change their Minecraft world through coding an automated solution with variables	Lesson A: Introduction to variables Lesson B: Coding with variables Lesson C: Combining variables Lesson D: Get creative with variables
Unit 5: Conditionals • Describe the importance of conditionals in coding • Create IF THEN and IF THEN ELSE conditional statements • Code with a variety of conditional blocks to automate their agent to find and collect important resources • Evaluate code to identify problems like infinite loops and debug the code with conditionals • Add a Say block inside I f then blocks to help debug problem code • Work collaboratively to design an original creative project to apply their coding skills in new ways	Lesson A: Introduction to conditionals Lesson B: Coding with conditionals Lesson C: Debug problem code with conditionals Lesson D: Get creative with conditionals
Unit 6: Functions • Describe the different kinds of coding functions • Understand the importance of functions while playing Minecraft • Understand the importance of functions in coding • Learn about real-life functions and how they can affect situations • Alter the Minecraft landscape through coding Design an original creative project to change their Minecraft world through coding an automated solution with functions	Lesson A: Introduction to functions Lesson B: Coding with functions Lesson C: Building on functions Lesson D: Get creative with functions
Unit 7: Iteration • Understand the use of iteration in coding and synonymous terms used by programmers • List examples of iteration in daily life • Describe the different types of loops in MakeCode • Use different types of loops to debug and code more efficiently • Design an original creative project to automate a solution with iteration	Lesson A: Introduction to iteration Lesson B: Coding with iteration Lesson C: Debugging with iteration Lesson D: Get creative with iteration
Unit 8: Arrays • Understand the usefulness of arrays in coding as a collection of related items • List examples of arrays in real life • Use the array operations to add and arrange items • Describe array indexes and elements in MakeCode • Sort array values with basic sorting algorithms • Design an original creative project that uses arrays to create a piece of artwork in Minecraft	Lesson A: Introduction to arrays Lesson B: Coding with arrays Lesson C: Building on arrays Lesson D: Get creative with arrays

Unit 9: Artificial intelligence • Identify factors that distinguish humans from machines • Recognize that computers model intelligent behavior • Understand the importance of artificial intelligence and explore some of the ethics and fears relating to artificial intelligence • Find opportunities to code your agent to behave intelligently in Minecraft • Design an original creative project to teach your agent to intelligently adapt to the Minecraft environment	Lesson A: Introduction to artificial intelligence Lesson B: Coding an intelligent agent Lesson C: Get creative with artificial intelligence
Unit 10: Final independent project • Design an original creative project to program four tools for a survival backpack to help you in a Minecraft world • Demonstrate your learned coding skills and apply them in a new way • Validate your approach to the project, including beta testing and analysis of code to debug and problem solve	Lesson A: Introduction to the project Lesson B: Coding the project Lesson C: Beta test and finalize the project

CSTA Standards

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

Unit 1: Introduction and unit 2: Events
CPP.L1:6-05 Construct a program as a set of step-by-step instructions to be acted out. CPP.L1:6-06 Implement problem solutions using a block-based visual programming language.
Unit 3: Coordinates
3A-IC-26 Demonstrate ways a given algorithm applies to problems across disciplines. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 3B-AP-14 Construct solutions to problems using student-created components, such as procedures, modules and/or objects. CT.L2-12 Use abstraction to decompose a problem into sub problems. CPP.L1:6-05 Construct a program as a set of step-by-step instructions to be acted out. CPP.L1:6-06 Implement problem solutions using a block-based visual programming language.
Unit 4: Variables
CL.L2-03 Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities. CT.L1:6-01 Understand and use the basic steps in algorithmic problem-solving. CT.L1:6-02 Develop a simple understanding of an algorithm using computer-free exercises. CPP.L1:6-05 Construct a program as a set of step-by-step instructions to be acted out. 2-A-5-7 Create variables that represent different types of data and manipulate their values.
Unit 5: Conditionals
CL.L2-05 Implement problem solutions using a programming language, including: looping behavior, conditional statements logic, expressions, variables, and functions. CL.L2-03 Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities. CL.L2-04 Exhibit dispositions necessary for collaboration: providing useful feedback, integrating feedback, understanding and accepting multiple perspectives, socialization. CL.L3A-01 Work in a team to design and develop a software artifact. K-12 Computer Science Framework Core Concept: Control Structures.
Unit 6: Functions
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-14 Create procedures with parameters to organize code and make it easier to reuse. 3A-CS-01 Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects. 3A-AP-13 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
Unit 7: Iteration

CL.L2-05 Implement problem solutions using a programming language, including: looping behavior, conditional statements logic, expressions, variables, and functions.
CL.L3A-03 Explain how sequence, selection, iteration, and recursion are building blocks of algorithms.

Unit 8: Arrays

3A-DA-09 Translate between different bit representations of real-world phenomena, such as characters, numbers, and images.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms.
2-AP-11 Create clearly named variables that represent different data types and perform operations on their values.
2-AP-12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
2-AP-14 Create procedures with parameters to organize code and make it easier to reuse.
K-12 Computer Science Framework Core Concept: Control Structures.
CT.L2-12 Use abstraction to decompose a problem into sub problems.
CPP.L1:6-05 Construct a program as a set of step-by-step instructions to be acted out.
CPP.L1:6-06 Implement problem solutions using a block-based visual programming language.
NGSS 3-5-ETS1-2 Generate and compare multiple possible solutions to a problem based on how well each is likely to meet the criteria and constraints of the problem.

Unit 9: Artificial intelligence

CL.L2-03 Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities.
CL.L2-04 Exhibit dispositions necessary for collaboration: providing useful feedback, integrating feedback, understanding and accepting multiple perspectives, socialization.
CL.L3A-01 Work in a team to design and develop a software artifact.
K-12 Computer Science Framework Core Concept: Control Structures.
CT.L2-12 Use abstraction to decompose a problem into sub problems.
CPP.L1:6-05 Construct a program as a set of step-by-step instructions to be acted out.
CPP.L1:6-06 Implement problem solutions using a block-based visual programming language.
NGSS 3-5-ETS1-2 Generate and compare multiple possible solutions to a problem based on how well each is likely to meet the criteria and constraints of the problem.

Glossary

The 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 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 number is an algorithm.
array	A group of objects, all of the same **variable type**, that can be referenced as a single variable. Objects in a collection are referred to as **elements** or items. An individual element can be referenced by its **index.** Also known as a **collection** in other programming languages.
assign	To set the value of a **variable**.
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.
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.
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 a **collection**.
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.
for each loop	A programming construct that allows for a block of code to be executed one time for each object in a collection, table, or other data structure with multiple elements. For example, a **for each** loop allows for a block of code to be executed for each element in a collection or for each row in a table.
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 an element in a collection or table. 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.
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** or **for each loop**.
loop counter	The variable used in a **for loop** to determine the number of times the loop will execute.
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.

program	See **computer program**.
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, 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 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