

Computer Science

6 Lessons

Upper Primary | Grades 3-5 | Ages 8-11

Coding Fundamentals: Part 3 Running Out of Time

EDUCATION.MINECRAFT.NET

Table of Contents

GETTING STARTED WITH MINECRAFT 3

COMPUTER SCIENCE PATHWAY PROGRESSION 4

CURRICULUM SUMMARY 5

INTRODUCTION 7

MINECRAFT: EDUCATION EDITION- TEACHING & LEARNING FRAMEWORK 8

INSTRUCTIONAL SEQUENCE 9

EDUCATIONAL STANDARDS 10

COMPUTER SCIENCE CONCEPTS 12

VISUAL GLOSSARY 13

Getting Started with Minecraft

Install Minecraft: Education Edition

Minecraft: Education Edition can be installed on Chromebook, iPad, Mac, and PC. To ensure your experience with Minecraft: Education Edition is top-notch, make sure your devices meet the minimum system requirements. To check if your device supports Minecraft: Education Edition, see System Requirements.

If you have questions about setting up Minecraft: Education Edition, the following link will provide you with some frequently asked questions and additional information about set-up. On this page, you will find assistance for:

- Get Started
- Purchase Licenses
- Administration and License Management
- Installation
- Troubleshooting

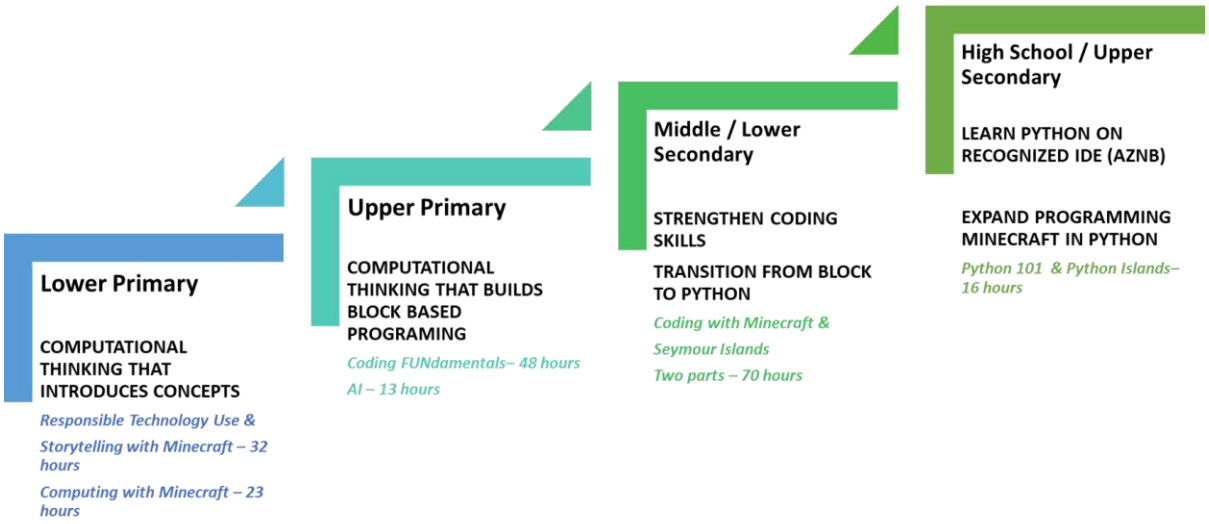
Preparing to Teach with Minecraft: Education Edition

Teachers do not need to have any prior computer science experience; however, they should familiarize themselves with a basic understanding of what is Minecraft: Education Edition. Support for building out teacher knowledge can be found here:

Minecraft: Education Edition: Teacher Academy

Minecraft: Education Edition Webinar Series

Computer Science Pathway Progression



Coding Fundamentals: Part 3 – Running Out of Time is a part of the Upper Primary (Grades 3-5|Ages 8-11) computer science progression. In this part of the progression, students are focusing in on applying coding concepts, such as events, event handlers, variables, loops, and functions in meaningful ways. Students will have multiple opportunities to manipulate and build out their own programs within Code Builder to solve problems and accomplish tasks.

Students will also continue to have opportunities to collaborate during their multiplayer experiences and provide one another with feedback during program development.

Curriculum Summary

Coding Fundamentals: Part 3 – Running Out of Time is specially designed computer science content for students in Upper Primary/Grades 3-5/Ages 8-11. Students will learn, practice, and apply relevant computer science skills and concepts as well as literacy skills in both unplugged and digital experiences. The lessons are designed to provide students with an opportunity to build knowledge of the concept(s) in an unplugged version (i.e., demonstrate the concept on paper), practice the concept with the direct support of their teacher in the Minecraft world, and then finally by completing the task independently.

Lesson Design

Each EDU guide contains multiple activities that are intended to be taught over the specified amount of sessions (explained in the **Instructional Sequence** part of this document); however, you should use discretion and modify/adapt the lesson activities based on your students' needs and abilities. Within the lessons, the instructional sequence will contain three parts correlating with the gradual release model:

Direct Instruction—Teacher-Directed, "I Do"	In the first step, the teacher introduces and models the appropriate way of performing the skills included in the new concept being taught.
Guided Instruction— Teacher Modeling, "We Do"	After the teacher models the correct way to understand or perform the new concept being taught, teacher will guide the students as they work through some examples together.
Independent Practice—Teacher Support, "You Do"	This step is where students demonstrate their initial level of understanding of the new concept being taught through independent practice.

Instructional Materials

Curriculum Overview	That is this document you are reading now! This will provide you with insight about the curriculum and what is taught within the curriculum.
Educator's Guides (EDU Guides)	An educator's guide is provided for each of the lessons. The guide provides a high-level overview of the lesson, learning goals, standards addressed, required preparation for the activities, the lesson plans for the activities, and any additional materials needed.
Classroom Presentations	Each unit is supported by its own PowerPoint presentation to provide structure and guide the educator through the activities for the lesson.

Formative Assessments	After each lesson in the EDU guide, there is an opportunity to check for student understanding of the concept taught within the lesson. These formative assessments are typically comprised of 2-4 questions directly related to the learning that just took place.
Summative Assessment	At the end of the entire lesson sequence, students will be provided with a performance-based task to demonstrate their new knowledge and skills learned throughout the computer science unit, Coding Fundamentals Part 3: Running Out of Time. This performance-based task can be assessed using the provided rubric.
Minecraft World Files	The specific world files needed to experience the instructional activities have been linked directly within the Educator's Guides.

Introduction

Level: Upper Primary | Grades 3-5 | Ages 8-11

Essential Question: How can computers help us experiment and develop innovative concepts and ideas?

Overview

OH NO! The experiment has gone terribly wrong...
While testing our latest experiment, the time traveling device, we have become stuck in a time loop! The time loop is sending us to random times and places throughout history. As students spawn in game, you will be see Dr. Rubin. She informs the students that they must get back to the Time Gate! In order to navigate back to present day, you must use your coding knowledge to overcome the obstacles and find your way to the time gate.

- Prehistoric Times with the Dinosaurs
- Ancient Times at the Roman Coliseum
- The 1850s Gold Rush in the Wild West
- Modern Day Beet Farm for the Future

After arriving back successfully through the Time Gate, you will take everything you've learned about coding concepts and create a challenge for your peers to solve!

Pacing: 12 hours (could be more if tutorials and/or supplemental lessons are included)

Materials

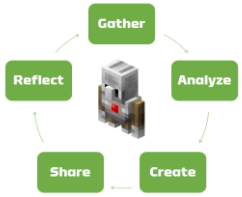
Hardware	• The teacher will need a laptop or tablet with a projector for the plugged-in coding lessons. • Internet access will be required for a portion of the lessons and activities. • Each student will need a device to complete the plugged-in coding activities.
Software	• Minecraft: Education Edition needs to be deployed on the devices utilized within these lessons. Use this link to find information about Deploying Minecraft: Education Edition. • The teacher and students will need the relevant MCworld files downloaded for each lesson (or they can be accessed from the in-game library).
Other Materials	• Grid Paper (for sketch plans) • Any additional handouts needed will be included within the individual EDU guides

Minecraft: Education Edition Teaching and Learning Framework

<table>
<tr><th colspan="2">GOAL:
Provide students with the necessary skills to become creative coders and content creators</th></tr>
<tr><td>Coding Mindset
Although Minecraft: Education Edition provides rigorous and engaging academics, we also deliver a holistic education- providing instruction for the social, emotional, and physical needs for our students. We want to foster a distinctive set of attributes. These qualities prepare our students to make exceptional contributions both in school and outside of school.

CURIOUS
COMMITTED
COOPERATIVE
CONSIDERATE
CONFIDENT</td><td>Computational Thinking Skills
Computational thinking provides a vital skill set in which students must possess in order to fulfill the industry's needs in the jobs of tomorrow.
Our ever-changing workforce creates a critical need for innovation. Our students need computational thinking skills not just to solve the problems within their educational journey, but to also meet the challenges of adapting to our constantly changing workforce.

DECOMPOSITION
PATTERN RECOGNITION
ABSTRACTION
ALGORITHMS</td></tr>
<tr><td>Computer Science Units of Study
Minecraft: Education Edition provides meaningful, relevant, and engaging units of study. The units of study will possess a conceptual lens to allow for depth and complexity to develop conceptual understanding—knowledge which transfers through time, across cultures, and across situations.

DIGITAL CITIZENSHIP
PROGRAMMING
CYBERSECURITY
IMPACTS OF COMPUTING</td><td>Community
As students gain and possess new knowledge and skills, we strive for them to find a greater purpose of "why do I need to know this" or more importantly, "how can I use this information?".
We aim to empower students, develop confidence and self-efficacy into a commitment to serve the community in which we live in and beyond.

</td></tr>
</table>

This unit will focus on **programming**, as students will learn and apply important coding concepts that will allow them to create powerful code for their Agent to complete tasks and solve problems.

This unit will focus on the coding mindset of being **confident**, as they will be using all of the coding concepts learned to create a coding challenge for their peers.

This unit will focus on **algorithms** as students explore the necessary and individual steps needed to create effective code.

This unit will provide students with the opportunity to consider how they can rely on their own **community** for relevant and meaningful feedback. Students will give and receive feedback to their peers throughout program development.

Instructional Sequence

This next section will provide you with an overview of the activities included in this lesson sequence. The lesson sequence is presented in chronological order—we suggest working in order, as the content will build upon skills presented in the previous session. A session is equivalent to one class period, or a 45-60 minute session. However, educators should feel empowered to modify and adapt the lesson sequence to best meet the needs of their students.

Lesson Sequence Overview

Session	Objectives	Teacher will	Students will	Resources
*1**	*Students will learn to develop algorithms or programs to complete a task*	*Explain the lesson challenges for the day: player validated and world validated activities*	*Students will complete their way through five obstacles to make it back to Time Gate*	*The Experiment*
*2**	*Students will learn how to utilize and combine different events*	*Review the concept of events and event handlers*	*Students will combine different event blocks to sneak past the dinosaur, navigate through the temple, and use the Time Gate to teleport back*	*Dinosaur Run*
*3**	*Students will create an app to using variables to increase efficiency in coding*	*Explain the concept of variables and how they can be used in coding to be more efficient*	*Students will use create an app using variables to program the Agent to help build the Roman Coliseum*	*Rome Sweet Rome*
*4**	*Students will continue to learn about using events, event handlers, and nested loops*	*Review how using events, event handlers, and nested loops can help complete tasks more efficiently*	*Students will solve coding tasks involving variables and nested loops for the miners during the Gold Rush of the 1850s*	*Wild West*
*5**	*Students will be introduced to the concepts of functions.*	*Model and explain what are functions and the purpose of functions in coding*	*Students will program the Agent to farm the beets using functions in their code*	*We Need Beets!*
*6-11**	*Students will collaborate with peers to create a computational artifact.*	*Explain the summative task and provide success criteria and guidance to collaborative groups.*	*Students will work together to plan, design, test, and present their creative coding challenge.*	*Coding Challenge*

*Educators should modify and adapt the amount of sessions based on students' needs.

Educational Standards

CSTA Standards

- **1B-AP-11** Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
- **1B-AP-10** Create programs that include sequences, events, loops, and conditionals.
- **1B-AP-08** Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
- **1B-AP-15** Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.
- **1B-AP-09** Work respectfully and responsibly with others online.
- **1B-AP-16** Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development.
- **1B-AP-17** Describe choices made during program development using code comments, presentations, and demonstrations.

ISTE Standards

- **1.5.c** Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.
- **1.5.b** Students collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making.
- **1.4.a** Students know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems.
- **1.5.d** Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.
- **1.7.c** Students contribute constructively to project teams, assuming various roles and responsibilities to work effectively toward a common goal.
- **1.6.c** Students communicate complex ideas clearly and effectively by creating or using a variety of digital objects such as visualizations, models, or simulations.

National Curriculum: Computing - Key Stage 2

- Use technology safely, respectfully and responsibly; recognise acceptable/unacceptable behaviour; identify a range of ways to report concerns about content and contact.
- Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts
- Use sequence, selection, and repetition in programs; work with variables and various forms of input and output
- Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs
- Select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information

Australian F-10 Curriculum: Digital Technologies - Year 3 and Year 4

- Using a range of communication tools to share ideas and information, for example participating in online environments (ACTDIP022)
- Acquire, store and validate different types of data, and use a range of software to interpret and visualise data to create information (ACTDIP016)
- Following, modifying and describing the design of a game involving simple algorithms represented diagrammatically or in English (ACTDIP019)
- Experimenting with different ways of representing an instruction to make a repetition, for example loops in a flowchart diagram or using a 'REPEAT' statement (ACTDIP018)
- Planning and implementing a solution using a visual programming language, for example designing and creating a simple computer game involving decisions and repetitions, suitable for younger children, that requires user input to make selections, taking into account user responses (ACTDIP020)
- Experimenting with different ways of instructing to make choices and repeat instructions (ACTDIP020)
- Describing in simple terms the nature of a problem and what a solution needs to achieve, for example what need the problem is associated with, who the solution is needed for, what data are needed and what features the solution would need to include (ACTDIP017)
- Using a range of communication tools to share ideas and information, for example participating in collaborative online environments (ACTDIP022)

Computer Science Concepts: Coding Fundamentals Part 3 - Running Out of Time

Lesson	Concept(s)
The Experiment	Events + Event Handlers Repeat Loops Variables
Dinosaur Run	Events + Event Handlers Repeat Loops Variables
Rome Sweet Rome	Variables Pattern Recognition Conditionals Sequencing Nested Loops
Wild West	Events Variables Nested Loops
We Need Beets!	Functions Documentation (Code Comments) Testing Debugging
Coding Challenge (Summative Task)	Program Development Peer Feedback Collaboration Debugging

MINECRAFT VISUAL GLOSSARY

Agent

personal robot in Minecraft



Blocks

the basic units of structure in Minecraft that make up the game's world



Book & Quill

an item used to create written books in Minecraft



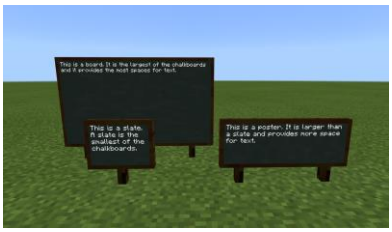
Camera

allows you to take screenshots and selfies in Minecraft: Education Edition



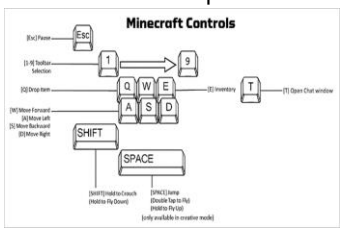
Chalkboards

special blocks that allow you to write and display text in Minecraft



Controls

(keyboard)
keyboard buttons that help you move around and complete tasks



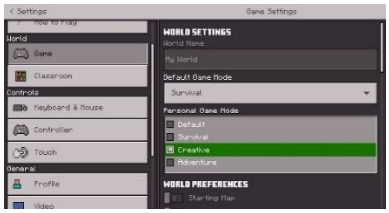
Controls

(touch)
the touch pad that helps you move around and complete tasks



Creative

game mode that gives you unlimited resources, ability to fly, and lets you destroy blocks instantly when mining



Hotbar

selection bar that appears on the bottom of the screen



Inventory

pop-up menu the player (or Agent) uses to manage the items they carry



Minecraft: Education Edition

a game-based learning platform



Mob

game character resembling a living creature



Portfolio

saves all of the photos that you have taken with a camera; allows you to add captions



NPC

non-player character



Radio

device used to reset the coding activity



Spawn Point

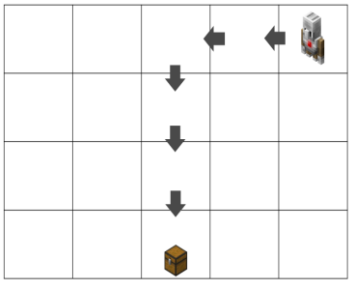
the location where a player begins game play



COMPUTER SCIENCE GLOSSARY

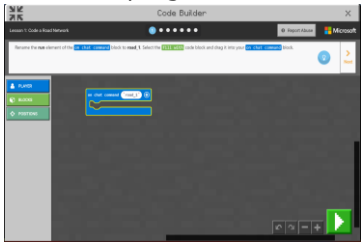
Algorithm

a sequence of defined steps



Code Builder

coding editor used to create programs



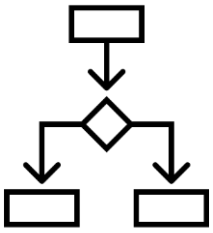
Code (Coding)

the method of giving a computer instructions to perform a specific task



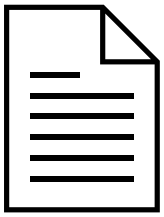
Conditional

an action that occurs if something specific happens



Copyright

the act of giving, receiving, and sharing information (talking, writing, listening, or reading)



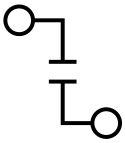
Debugging

process of finding and resolving errors within computer programs



Decomposition

breaking down a complex problem into smaller, manageable, and easy-to-understand parts



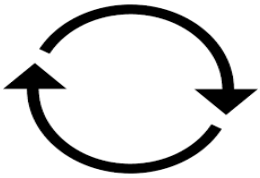
Intellectual Property

the original expression of an idea that later becomes something tangible



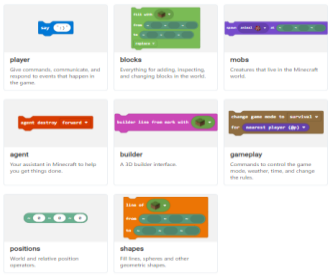
Loop

sequence that repeats a portion of code a set number of times until the desired task is complete



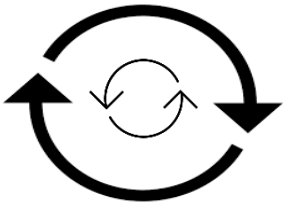
MakeCode Blocks

blocks used to create code



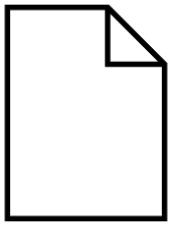
Nested Loop

a loop inside a loop



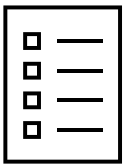
Pseudocode

informal description of a computer program or algorithm



Program Development

process of creating a program with a description of the sequence of events, goals, and expected outcomes



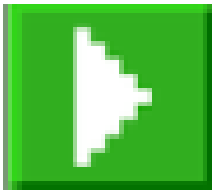
Sequence

instructions presented in a specific, correct order to a computer



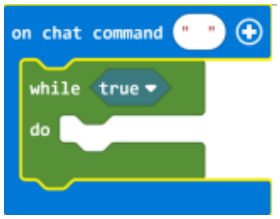
Testing

the process of running the program to see if runs as intended



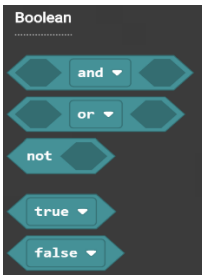
While Loop

repeats the code if a defined condition is met



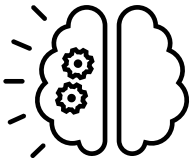
Boolean

a value that is either true or false



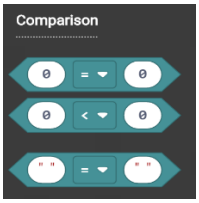
Persistence

not giving up



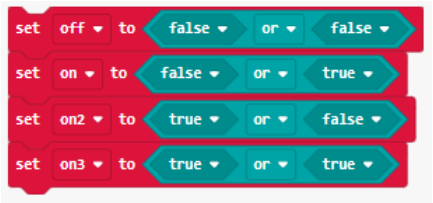
Comparison Operator

compares two values against one another; that comparison gives us a Boolean result



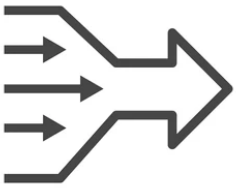
Disjunction

A or B evaluates to true if either A is true or B is true or if both A and B are true



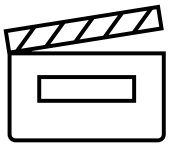
Function

a named section of a program that performs a specific task



Event

the change in the state of the object; this change is brought as a result of some sort of user interaction



Event Handler

the part of the program that runs when a specific event happens (i.e., it "handles" the event)



Variable

the places in which computers store values in a computer program

