MCC DOCUMENTATION

An overview of the Youth Competitive Programming Circle's Minecraft Coding Camp, including curriculum and project details

Youth Competitive Programming Circle
staff@ycpc.us

Minecraft Coding Camp

Minecraft Coding Camp: A Virtual World Application

YCPC's Minecraft Coding Camp Program uses ComputerCraft mod for Minecraft to allow students new to coding to learn Lua and use robots to automate various in-game tasks. Cutting out the intensive memorization and training characteristic of traditional pedagogical schemes, this program is designed to spark an interest in coding among high schoolers and advanced middle schoolers.

The Minecraft Coding Camp teaches kids to code using a game modification for Minecraft (referred to as mod for short) by the name of "ComputerCraft". This mod enables students to program robot automation in-game with the Lua programming language, allowing beginners to receive immediate feedback on their programming in the gaming context they are familiar with.

In ComputerCraft, students play on a hand-crafted adventure-style world. They are guided through an in-game tutorial of the basics of coding in Lua: print statements, if-else statements, loops and given documentation on how to use such commands. Then the students are free to choose from a structured curriculum of lesson-projects, each of which is a mini-adventure that increments in difficulty and teaches the student to master built-in robot functions such as .moveForward. This will culminate in the ability to automate their "turtle" robot to complete any task ranging from farming to fighting off zombies.

With a 2:1 student to instructor ratio, no students are left behind and are provided significant assistance with the creation of their own programs.

Minecraft Coding Camp – An Overview

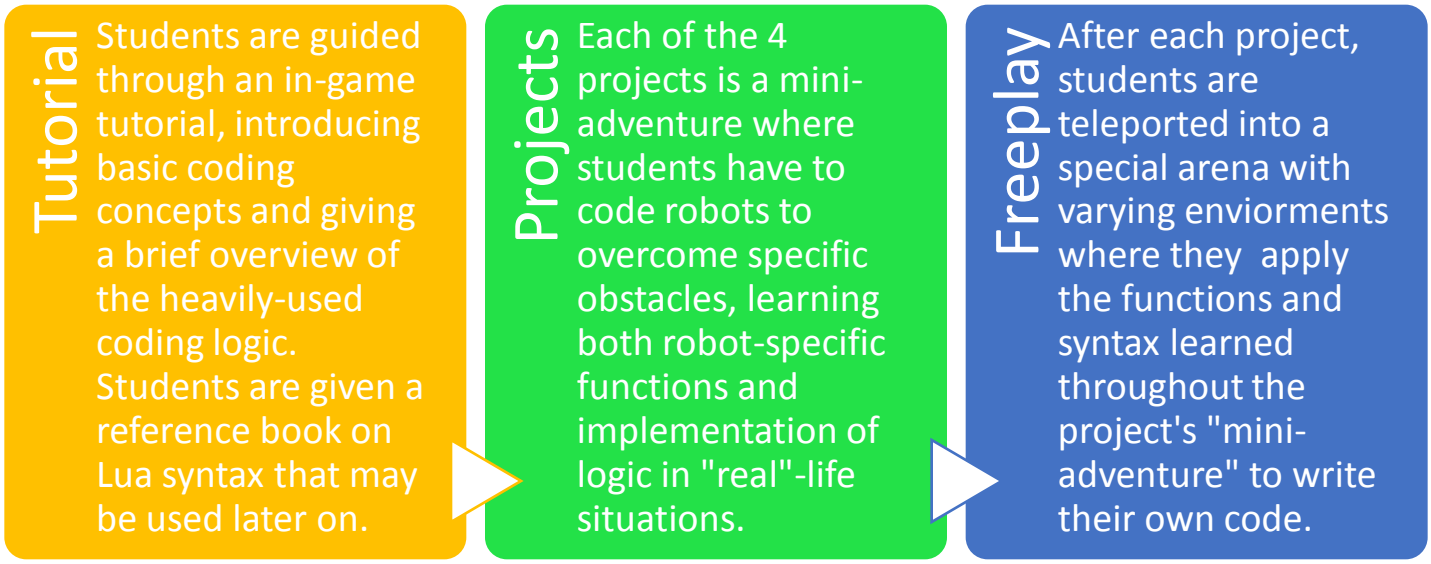
Advantages over Traditional Coding

One of the primary hurdles for beginners to coding is realizing the value of software development. For example, for the Lua programming language, it takes the complex installation of the Lua development kit, runtime environment, development interface, and requisite libraries to even begin coding. Then it takes six lines of syntax-sensitive code and compilation to make the computer write the simple "Hello World".

ComputerCraft, on the other hand, uses virtual robots and their corresponding built-in functions to allow students to program a zombie-fighting machine in those same six lines of code. This makes coding a more meaningful, humble, and entertaining endeavor.

Program Structure

A typical camp consists of two hour sessions split across 2 days. Camps are typically held after school.



In a typical day, a session begins 15-30 minutes after the end of school and lasts for 90 minutes. The first 30 minutes of the program serve as an orientation where students sign in and are introduced to the Minecraft and ComputerCraft interface. The next 45 minutes are dedicated to students completing the in-game tutorial and trying out the basic syntax and coding logic on the ComputerCraft interface. During the remaining 30 minutes of the first session and the entire second session students can work on any of the four projects. Because a typical project takes around half an hour to complete, students should be able to experience and experiment with all four of the provided projects.

Curriculum

ComputerCraft basics

- ✓ Operating a system console
- ✓ Creating, saving, and running programs
- ✓ Printing text to console
- ✓ ComputerCraft "Turtles" and their special functions

Variables

- ✓ What is a variable
- ✓ How to use a variable
- ✓ Comparing variable values
- ✓ Incrementing variables to control loops

If Statements

- ✓ "if" statements
- ✓ Using functions as conditions
- ✓ Complex/nested "if" statements

Loop Statements

- ✓ While loops
- ✓ Infinite loops
- ✓ Nested loops
- ✓ Using loops to iterate through a 2D grid

Functions

- ✓ Defining and calling local functions
- ✓ Conditional functions
- ✓ Fail-safe movement functions

Prerequisites

On-site Needs

The Minecraft software requires somewhat powerful computers. Also, because our program requires a modded version of Minecraft (to install the ComputerCraft mod), our organization will need access to either the Roaming folder (Windows) or the Library/ApplicationSupport folder (Mac)

Student Experience

Although our program introduces the basic concepts, some students may find the coding logic confusing. Those who need extra assistance should first reference and familiarize themselves with the conditional logic used in coding (such as if-else statements). There are countless resources for this online, including YCPC's personalized online courses at https://www.ycpc.us/courses.

Continued Education

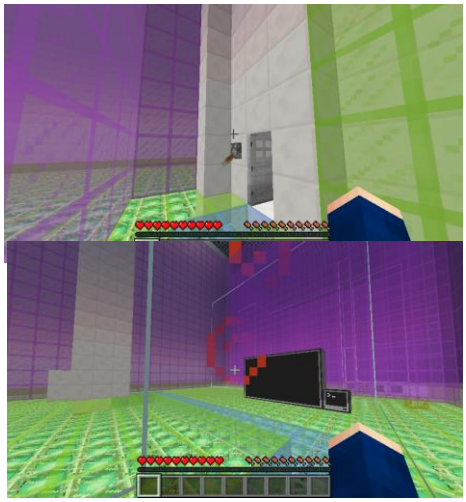
We highly suggest that students continue their coding education after MCC. Along with online resources and our online courses, our organization also offers the Library Coding Initiative and a Chapter program to create a local community of instructors, peers, and teachers that will assist students.

Tutorial

The tutorial for MCC is divided into four rooms that teach the basics of Minecraft to students that are unfamiliar with the game. This tutorial does not require teacher intervention, and has text that automatically pops up on the computer screen to guide the student. After each room, the student's character gets automatically transported to the next room.

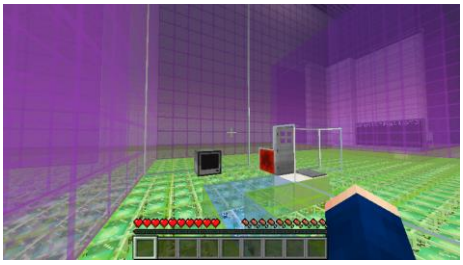
Room One – Basic Controls

This room teaches the fundamentals of playing Minecraft such as movement and inventory access. Conveyor belts are laid around the room to ensure that no student gets lost or has trouble finishing the room.



Room Two – Computers

To run programs in Minecraft one has to utilize computers, which serve as the compilers for coding in Minecraft. This room allows a student to accustom oneself with how to run programs using a Minecraft computer. The 'run' keyword is taught in this room.

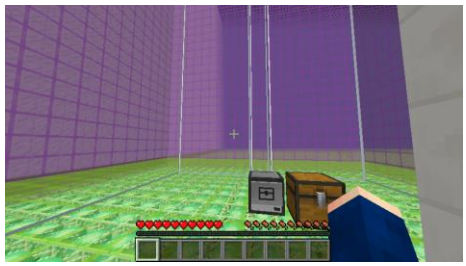


Room Three – Creating Programs

Students in this room are introduced to creating, saving, and editing their own programs using the Minecraft computer. The 'print' keyword is taught in this room.

Room Four – Commands and Turtles

Students in this room delve further into understanding basic commands in Minecraft. Specific syntax commands that may be unfamiliar to students learning programming for the first time are dissected to ensure the student fully understands how the command syntax works. Students are also introduced to turtles, which are the Minecraft robots that will be the subjects of all the coding.



Project 1 - Mining

After the tutorial, students will be able to work on one of four adventure-map styled projects.

In the first lesson students are guided through programming an automated mining "turtle" that will mine for resources and move to find new ores. Using the basic turtle functions .dig, .turn, .refuel and .move, students will learn the basic information needed to develop more complex automatons. Starting out with mining through basic obstacles, students eventually have to develop software that automatically digs through a 2D grid to create a quarry. More specific details are listed below.

The mining project starts off with accessing a program called 'mine', which is located in a mining turtle. After understanding the digging code, the student is asked to run that program and see the turtle mine one stone block.

Then the student proceeds to the next part, where a program is already precoded to mine two stone blocks stacked on another.

The first task the student must complete independently comes in the third room. The student has to code a program that digs a 1x1x2 hole through a stone wall in order to progress. Using the refuel, forward, dig and up commands utilized in the previous 'mine' programs, students must figure out how to proceed.

The tunnel that the student made will lead to another requiring a student to get across a pool of lava using a turtle's place function. Many new concepts are required for this task, including assigning variables, checking variables (== sign), detect functions, if then statements, and place functions.

After getting past the treacherous lava, the student must dig through another stone wall before getting to the next challenge, where the student must dig a redstone block located on top of lava using turtle.forward() and turtle.dig().

Finally, the student has to try to get as many emeralds as possible using only a mining turtle in the 'arena' below. This task encompasses all the commands learned thus far, like refuel, forward, dig, digUp, digDown, etc.

Project 2 - lumberjack

In this project, students automate a robot to cut trees of varying heights and replant saplings. The lumberjack automation uses functions from the mining tutorial and also introduces slightly more sophisticated utilities such as .detect and .place.

Loops and arithmetic operators are introduced at the start of this project, where students get familiar with code for cutting down trees of any height using logging turtles.

Then the student is asked to obtain a redstone torch located at the top of the tree. He or she will have to use while loops and dig, digUp, and place functions.

Project 3 - Melee

Students synthesize previously learned concepts and ideas to create automated guards that fight off zombies and hunt down monsters. Students learn to automate melee turtles to complete various tasks such as defending the player or moving in a snake pattern to kill all mobs in a 2D grid.

Students start off by learning the code for killing zombies in a confined 2x2 space. The code needed involves while loops and attack functions. Local functions are also introduced in order to reduce code so that students don't repeat the same code over and over. Local functions and while loops are used to move the turtle around in a circle to kill all the zombies.

To proceed, the student must create a program that kills all the zombies within a 2x5 grid. Students will have to use some ingenuity as well as the previous code given to complete this task. After killing the zombies, the student can then obtain the redstone block in the chest that the zombies were guarding, and proceed to the next challenge by placing the redstone block in the designated spot, which opens the door to the next area.

The last challenge in this project is to kill 10 zombies that spawn inside an arena. Students will not be able to attack them head-on, so they will use the two turtles at their disposal and create a program so that they will go around the arena and kill the zombies. Student knowledge will be put to the test as students need to figure out how to utilize while loops and local functions to program turtles that will kill the zombies sufficiently. There are many different ways and strategies for completing this task.

Project 4 - Farming

Here students automate a turtle to suck seeds from a chest, till dirt, plant seeds, harvest crops, and place the yield into a chest. To do so, students utilize advanced features such as .suck, .drop and .sleep. This farming project is by far the hardest project, and is designed for the most motivated of students.

The first part of the project demonstrates a basic tilling system using the farming turtle. The program shown uses variables, if/then statements, modulus, and moving functions. The turtle snakes through the plot of land, tilling the land as it goes by. The program itself keeps track of the current row and current column the turtle is in, and implements the necessary movements to make sure every tile of land is tilled. There are four 'demonstration' turtles, each of which introduce more and more features, like seed planting, harvesting and storage of wheat, and the bonemealing of wheat. However the basic movement code remains the same for all four turtles.

Then the student comes to a small plot of land, where he or she can put all their knowledge about farming in Minecraft to use. The student will need to utilize the movement code demonstrated in the four turtles' code beforehand, as well as implement various other features like seed planting, and wheat harvesting, storage and bonemealing. This task is very open-ended and will be very challenging.

If the student has completed all the projects in the specified order, after this project, the student has completed the camp!

Bring Your Own Device:

At some sessions, computers may be provided; however, if YCPC computers are unavailable, we do ask that students bring their own devices. In such a situation we ask that they bring a computer capable of running relatively demanding software. Also, because Minecraft burns through battery rapidly, we highly recommend the presence of a compatible charger. Wi-fi will be provided at events unless otherwise noted.

Set Up Process

If students use their own computers for the program, some software will need to be installed.

Required Components:

- Minecraft Launcher:
  - Minecraft.exe for Windows
  - Minecraft.dmg for Linux
- Minecraft Forge Version 10.13 (To run Minecraft Mods)
- Computer Craft Mod
- Custom MCC World
- Minecraft Account

Installation Process:

The standard process for installing the required components

i. Place our custom Minecraft folder with Forge, the Computer Craft mod, and our custom world preinstalled. In Mac, replace the Minecraft folder in Library/Application_Support. In Windows, replace the .minecraft folder in Roaming (can be found by searching %appdata%)

ii. Run the Minecraft launcher for your specific operating system, which can be found at https://minecraft.net/download or provided on-site

iii. Students may use their own Minecraft account or an organization provided account.