Coding with Minecraft Education

Summit Middle course handbook

9 course sections
Setup & Installation; Lessons 1-8

A step-by-step block coding course with instructions, worked examples, and code screenshots. Work at your own pace and use the examples to build your own projects.

Contents

Setup & Installation

Minecraft Education Edition Coding - Installation and Preparation Instructions:

Getting started

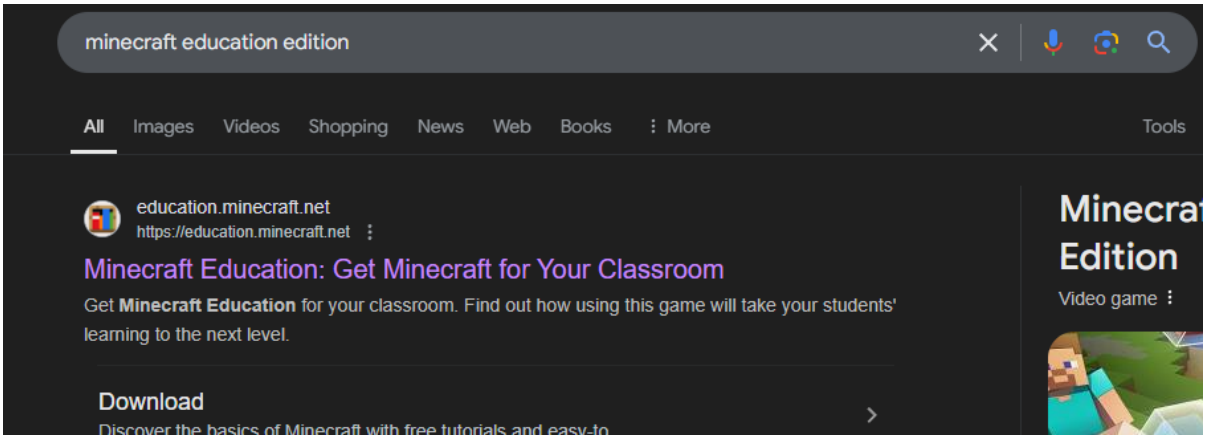
Before beginning the course, set up Minecraft Education on your device.
Follow the installation steps below, then check that the application opens correctly.

Installing Minecraft Education Edition:

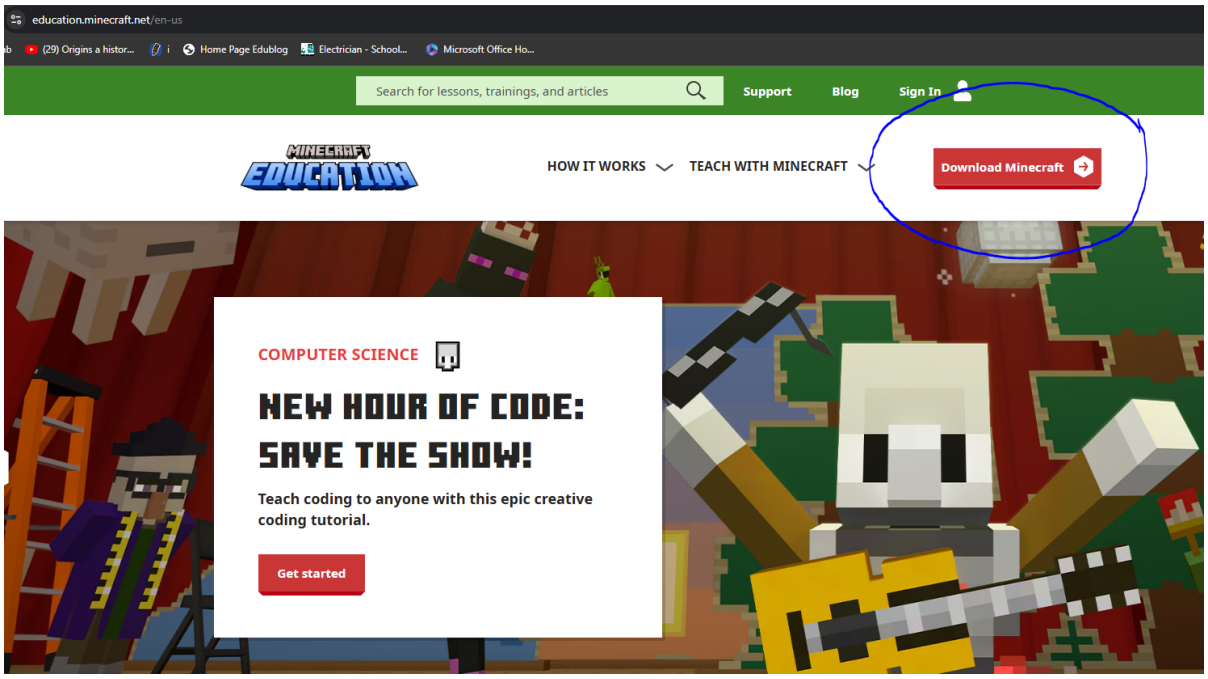
Install Minecraft Education on the device you will use for the lessons.

To install Minecraft:

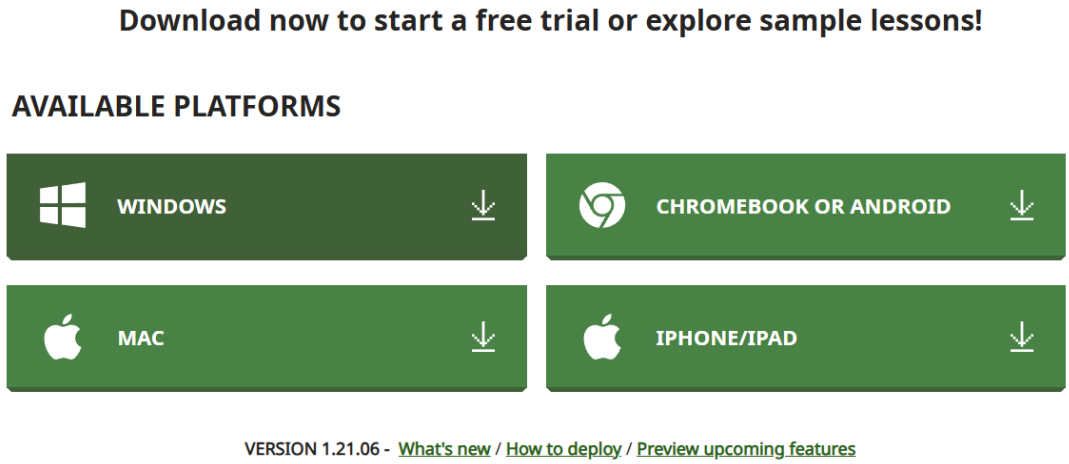
First, open a web browser, and search "Minecraft education edition". The first result should come up looking like this:



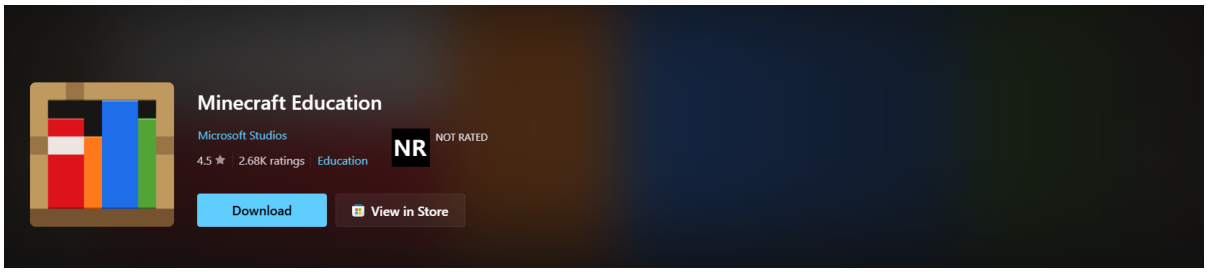
Open "education.minecraft.net", and find the window which reads "Download Minecraft"



Select this, and then select the platform which your device is on. For most students, this will be windows, but all platforms will work.



If on windows, clicking the button will bring you to the Microsoft store, where you can download the app. Once you click download, there are two possible scenarios:



Microsoft Store will open, where you can just click download again and the app will begin to install.

An installer will begin to download onto your computer. If you're on Google or another browser, this may pop up in the top corner of your screen, where you can select to run the application. If this doesn't appear, open File Explorer, and navigate over to the downloads folder where you'll find the installer which you can double click to run. This will download the application.

If these steps don't work and you're unable to download the app, you may also look on the app store of your device. For Microsoft, this is the Microsoft store, for Apple/Mac this is the App Store, and for Google/Chromebook this is the Google Store. Simply search "Minecraft Education" and the app should come up.

Once the download is finished, launch the application.

Once the application opens, follow the on-screen instructions to sign in.
Check that you can open Minecraft Education on your device.
Use the help options in the application if you have trouble getting started.
Make sure you can reach the main menu before beginning the first lesson.

Keep your sign-in details private. If you have trouble accessing the app,
use the help or account recovery options on the sign-in screen.

Once you reach the main menu, you are ready to begin Lesson 1.

Keep your device charged and your charger nearby so you can work through the lessons without interruption.

Save your project and test the code before continuing.

Lesson 1

Summit Middle: Coding with Minecraft - Lesson 1 - Review

Lesson overview

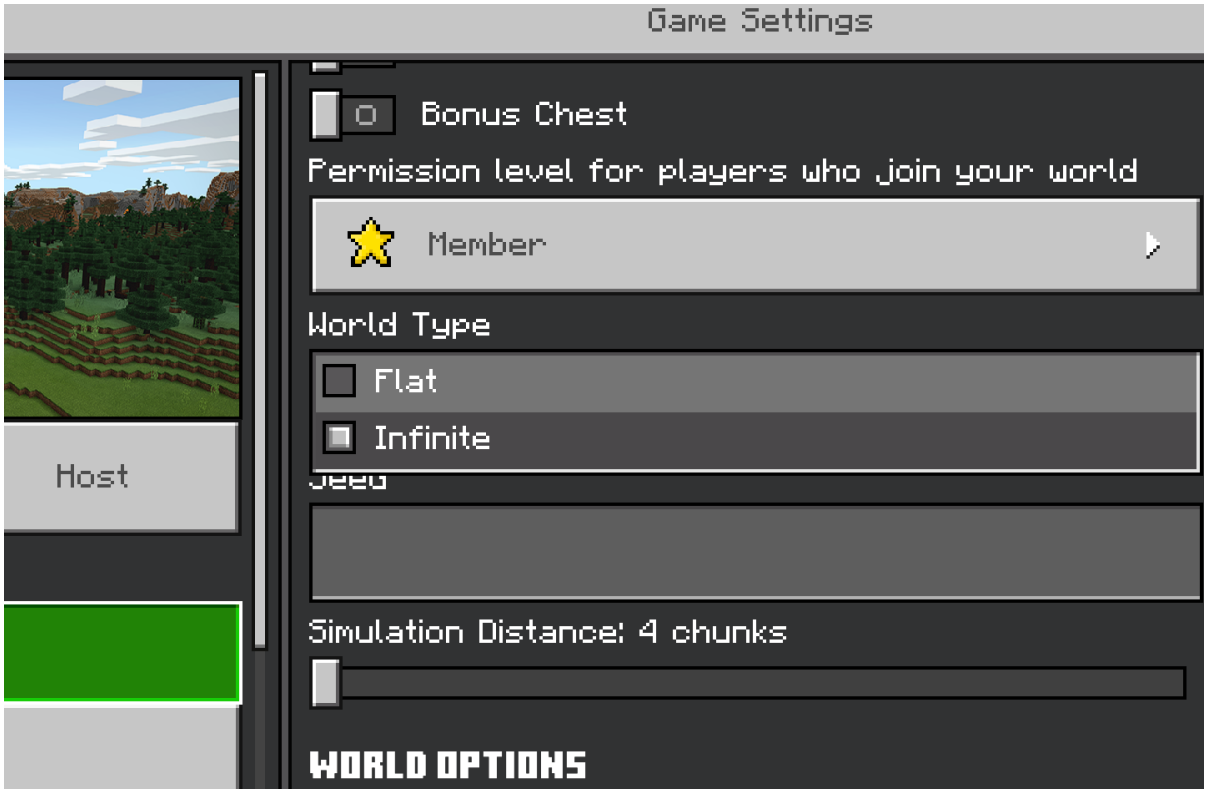
Complete this lesson at your own pace.

Follow this introduction to explore the coding workspace and build your first program.

In this lesson, we focused on familiarizing ourselves with the coding workspace, as well as exploring a few simple blocks.

To begin, we launched Minecraft Education Edition, and created a new world, preferably a flat world.

(for those who don't know how to create, use the following menu option when creating a world) :



Once loaded in, pressing "C" on the keyboard opens the coding tab, and then students must select "Microsoft Make Code", with a purple puzzle-piece icon.

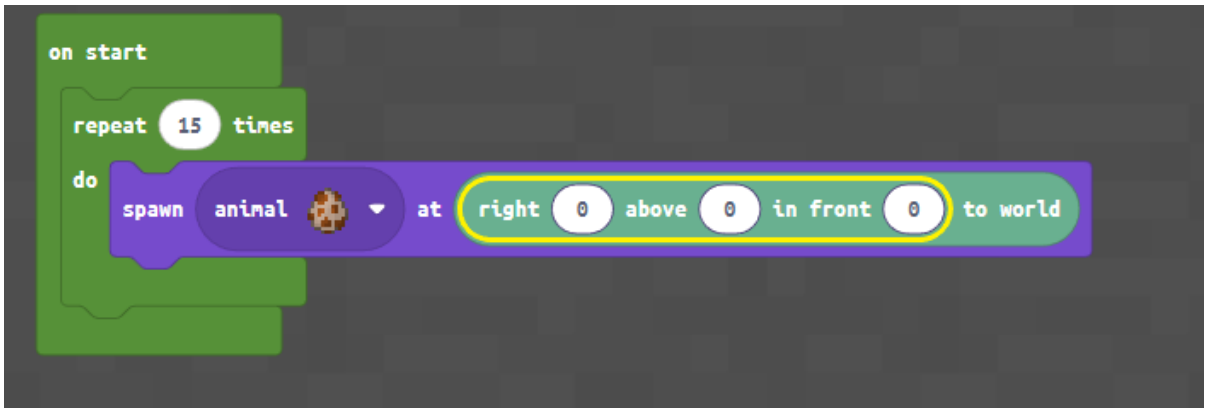
Create a new project using the button and name it "lesson 1".

This opens up the coding space, where we can find blocks, as well as color-coordinated tabs which contain different blocks.

We started our coding by using the existing "on start" block, and going into the Green "Loops" tab to find the "repeat _ times" block. We then dragged this block inside of the "on start" block.

Next, we went to the Mobs tab, and found the block reading "Spawn (animal) at _____", and dragged this inside of the "repeat __ times" block.

Finally, to finish off this chunk of code, we went over to the "Positions" tab, and dragged in the first block, reading "right _ above _ in front _ to world". This block gives a position relative to our player.



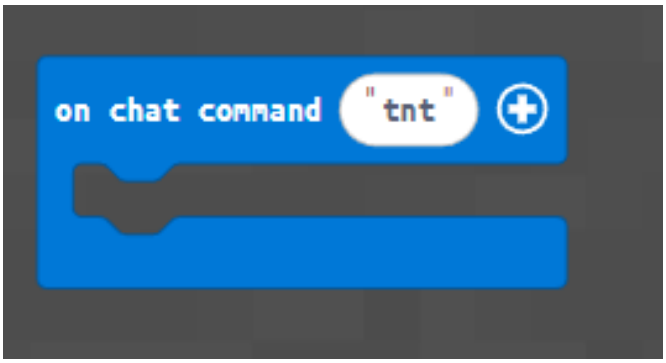
All in all, this block of code runs as follows:

At the start of the game (when this code is started using the big green play button), we spawn an animal at our position, and repeat this 15 times (spawning 15 animals in total).

Next, we explored chat commands. The chat can be opened by clicking "T" on the keyboard, and messages can be sent by clicking the 'enter' key.

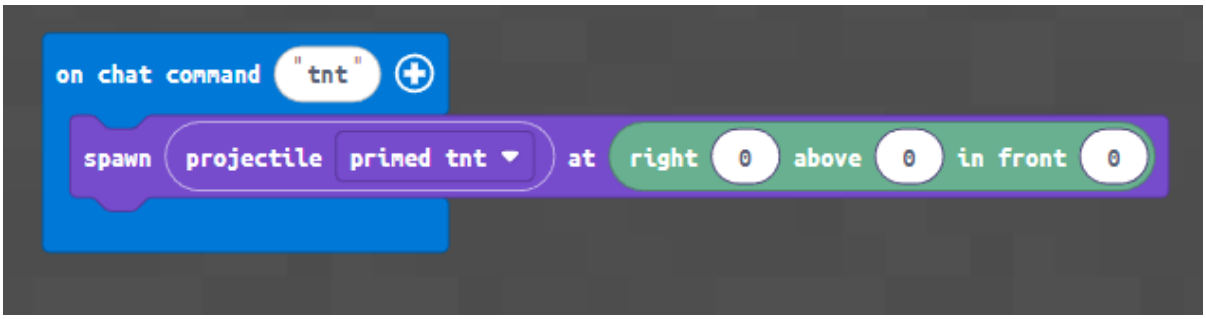
To create commands, we can use the "on chat command ___" block, which is found in the "Player" tab. You can change the name of the command by simply selecting the word in the quotation marks.

With our first "on chat command __" block, we'll first select the quotation marks and then type in "tnt" as our command name.

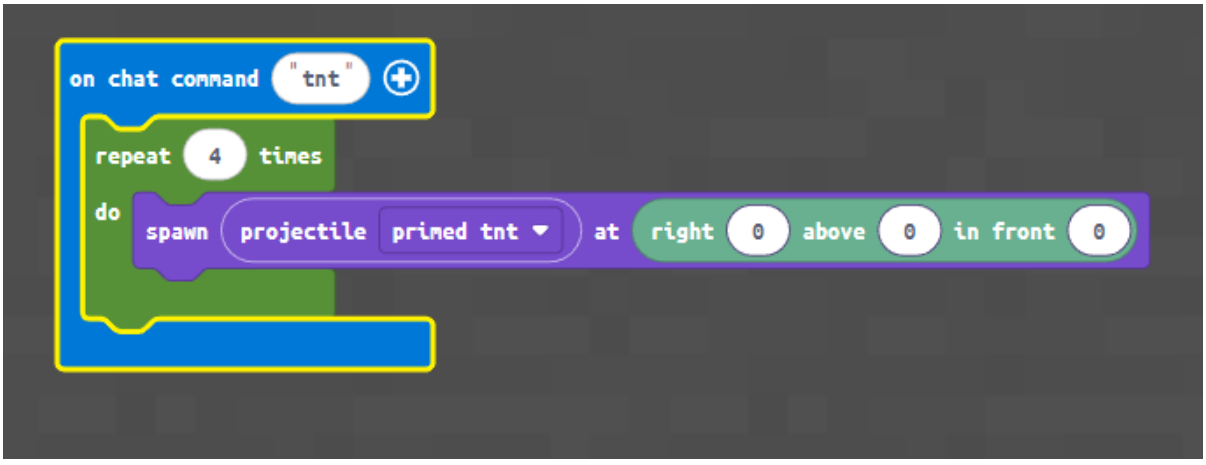


Next, we go over to the "Mobs" tab, and drag in the "spawn ____ at ____" block, and while still in the Mobs tab, drag in "projectile primed tnt" into the first slot.

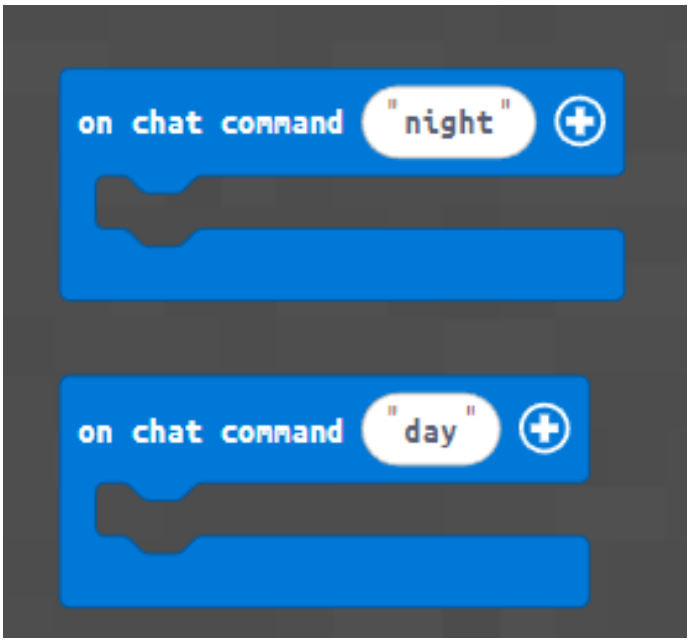
Finally, we went to the Positions tab and dragged in the "right __ above __ in front __" block to make the tnt spawn at the player's position.



Students were tasked as a bonus to figure out how to spawn multiple tnt blocks using the "repeat" block we used earlier, and the solution looks like this:



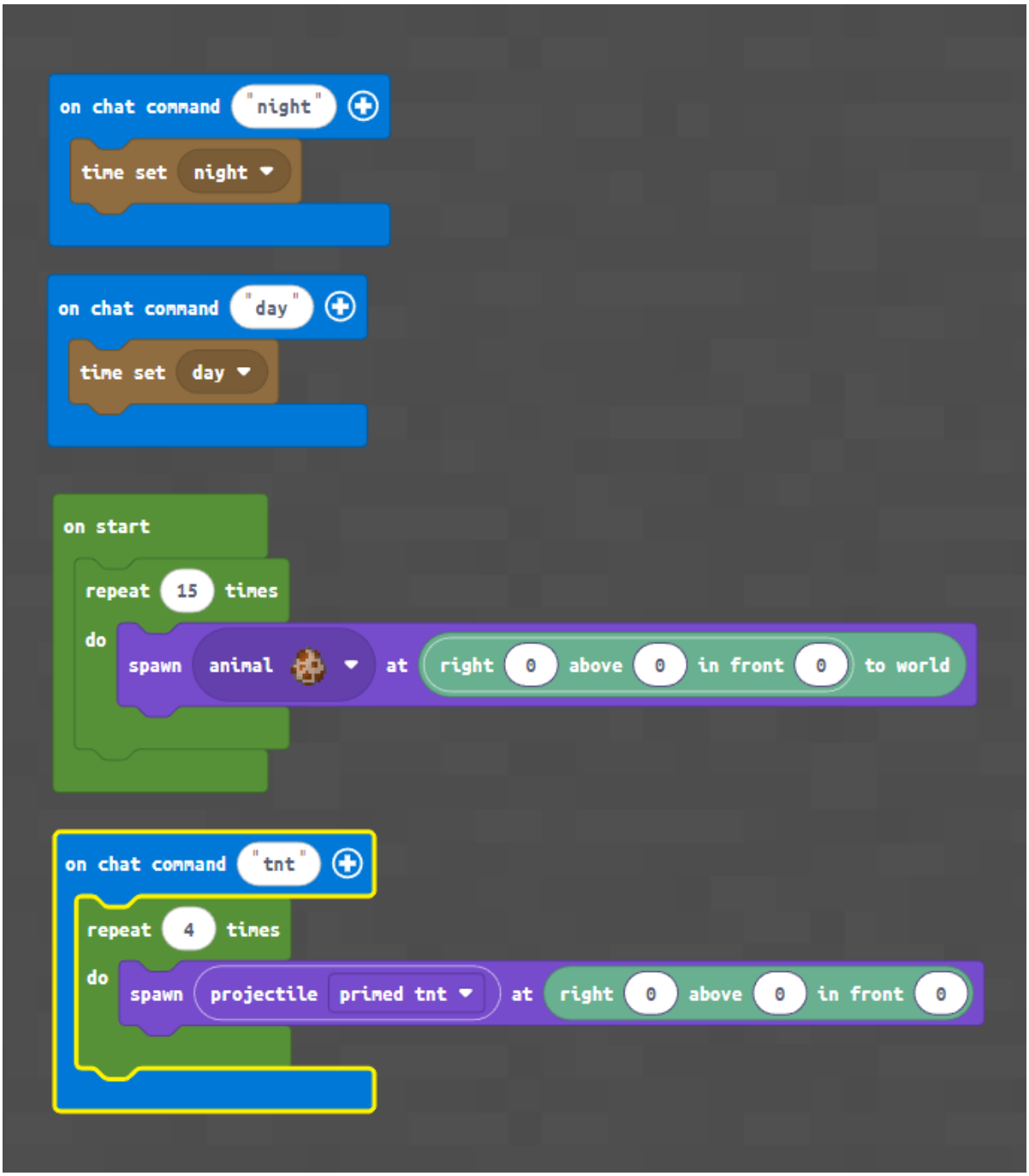
Next, we bring in two more "on chat command" blocks, one that we name "night" and the other "day".



Then, using the "time set __" blocks from the Gameplay tab, students were challenged to make the commands change the time to the correct time of day.

The solution to that looks like this:

Thats all the code for this lesson! Altogether it looks like this:



In the next lesson, build a dungeon game where players fight increasing numbers of enemies as the waves progress.

Complete this lesson at your own pace.

Lesson 2

Lesson overview

Complete this lesson at your own pace.

This lesson, we created a wave-based game where players have to defeat waves of enemies that increase in size the further they go. They were also encouraged to build an arena and see how many waves they could get to.

We explored new blocks, and got more familiar with blocks we've already seen and how we can combine the two.

Some of the new blocks we explored were:

If statements

Variables (change and set)

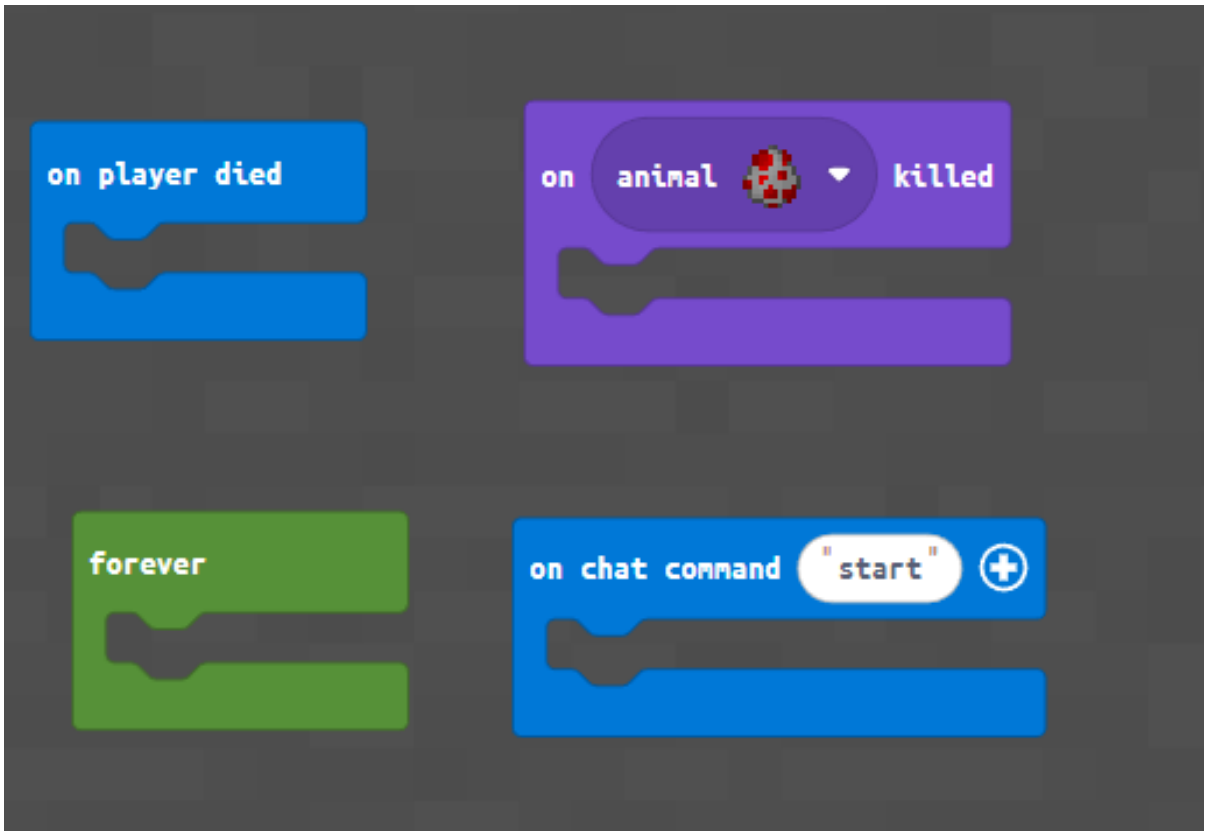
Monster blocks

Change game mode, difficulty

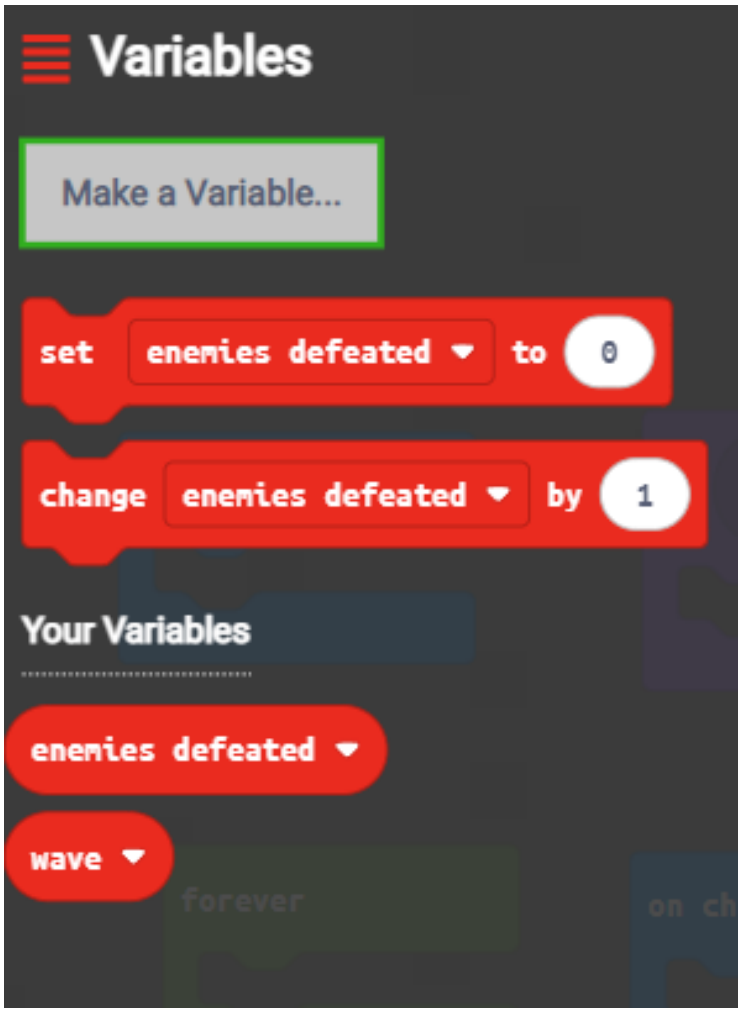
On monster/player defeated

For those who simply want to see the code, just scroll to screenshots at the bottom .

To begin, we dragged in four key blocks: the "on animal killed" from the **Mobs** tab, the forever block from the **Loops** tab, the "on chat command" block from the **Player** tab, and the "on player died" block also from the **Player** tab.



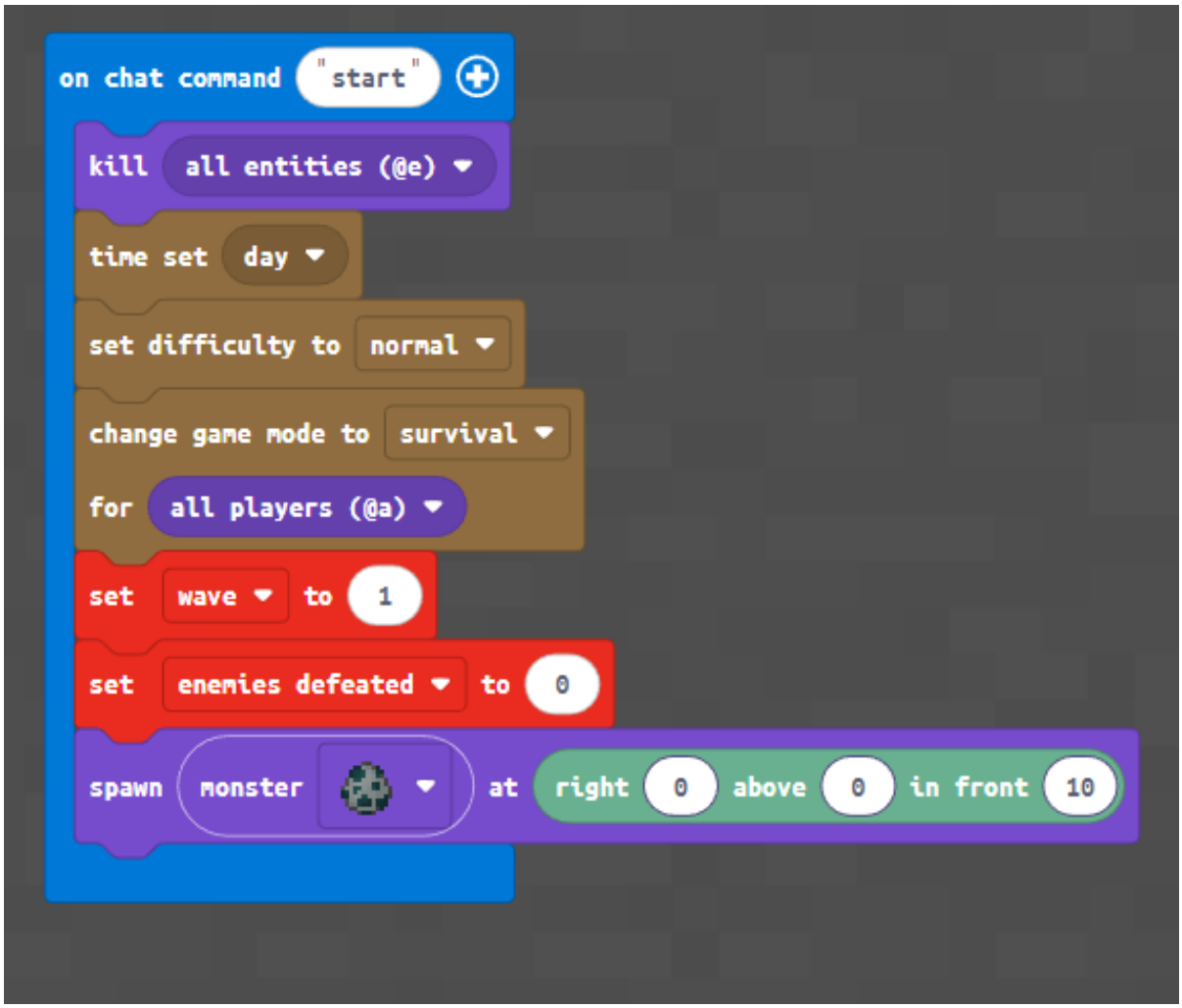
Following this, we need to create two variables. Basically, variables are blocks that we get to create, and they can represent any number. In this case, we'll create two variables: one to keep track of the current wave number, called "wave", and one to keep track of enemies defeated, called "enemies defeated". To create these, go over to the **Variables** tab and select "Make a Variable".



To begin, we'll work on the "on chat command" block. Set the command to be "start", and then begin by opening the **Mobs** tab. Drag in "kill 'nearest player'", and change the box to say "all entities instead". Then, open the **Gameplay** tab, and drag in 3 blocks: "time set __", setting it to be day, "set difficulty to", setting it to be normal, and "change game mode to ___ for __", setting it to be survival for all players @a.

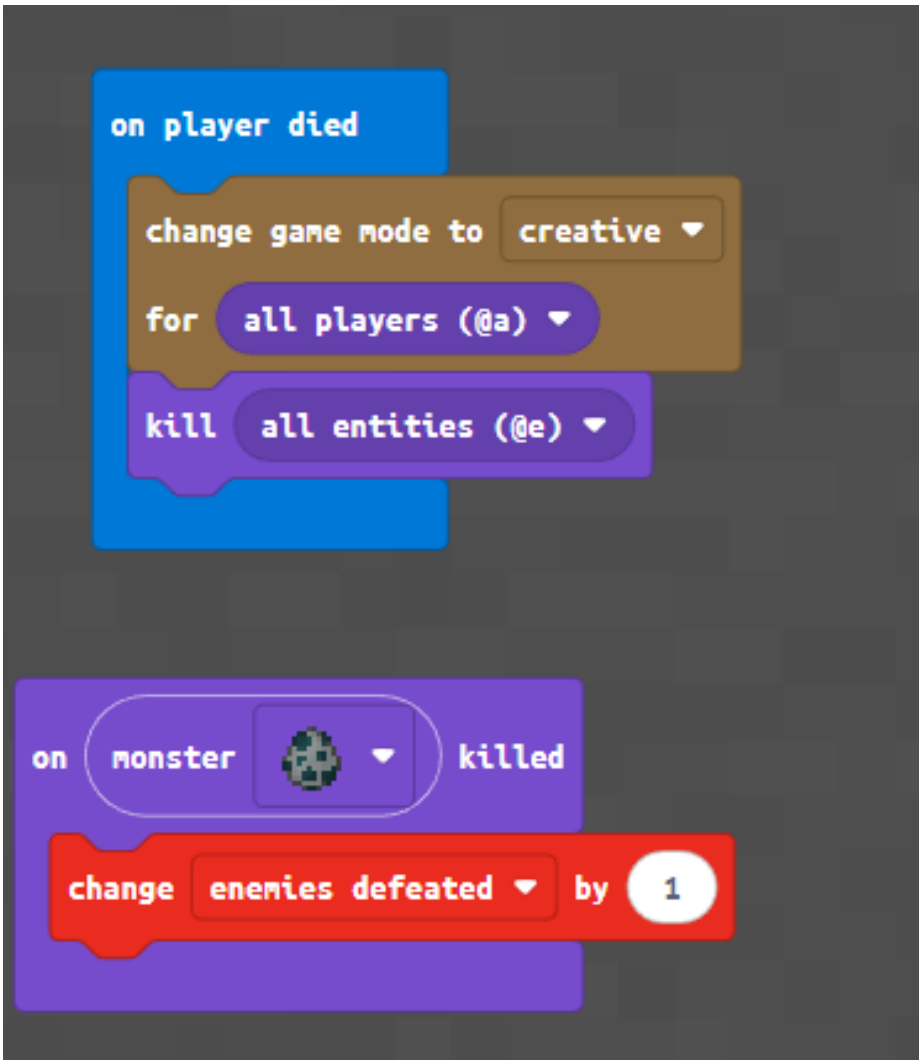
After this, we go over to the **Variables** tab and drag in two blocks that say "set ___". Change the first block so it reads "set wave to 1", and the second block so it reads "set enemies defeated to 0"

Finally, we want to go over to the **Mobs** tab, and drag in the "spawn ____ at _____" block. Then, from the same tab, drag in "monster __" to both this block and the "on animal killed" block from earlier. Finally, to both blocks, go to the Positions tab, and drag in the "right 0 above 0 in front 0" block, setting "in front" to be 10.

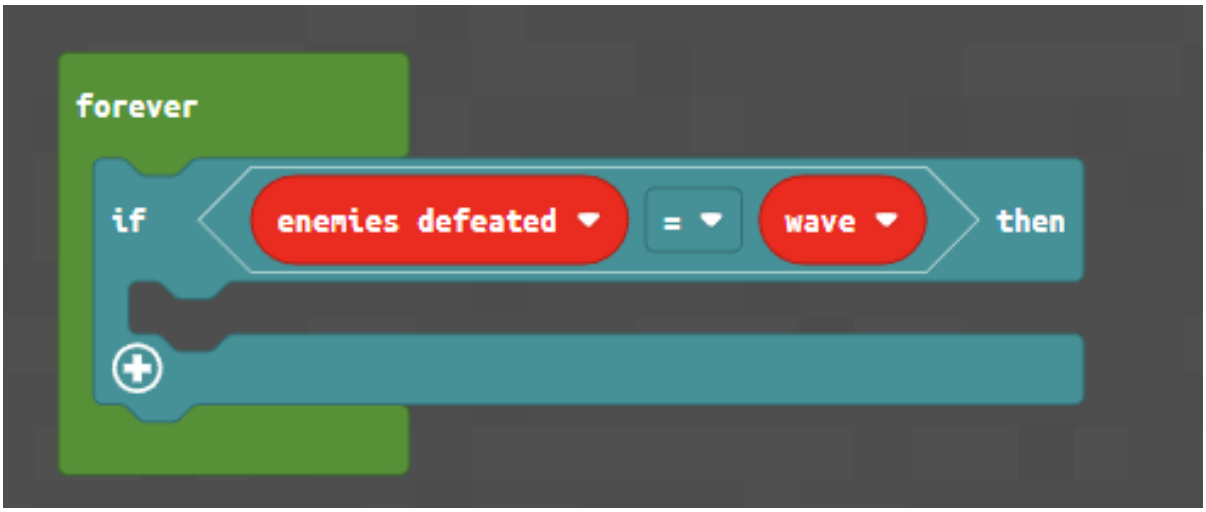


Finally, and most importantly, make sure that the enemy spawn 'egg' that is selected is the 'vindicator' enemy. These steps altogether will ensure that on start, the player will be in the correct game mode, at the right time, that no extra enemies will be present, and that we have the wave number and enemies defeated accurate.

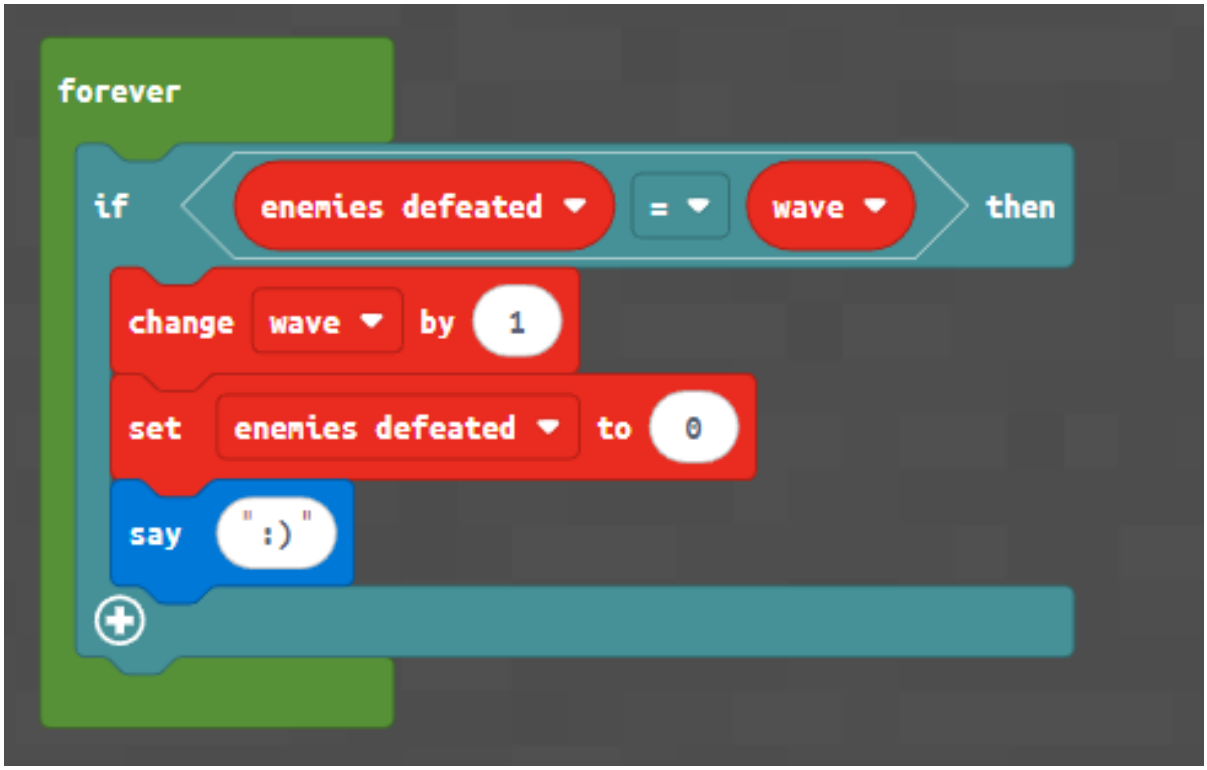
Once this is finished, we'll go over to the "on monster killed" block, and drag in the "change ___ by __" from **Variables** block, and make it say "change enemies defeated by 1". This will ensure that we can track when an enemy has been defeated. After this, to the "on player died" block, drag in a "change game mode to __ for __" from the **Gameplay** tab, making it say "creative" for "all players @a", and then after this, go to the **Mobs** tab, and drag in a "kill ____", making it say "kill all entities @e". This will kill all enemies when the game is over, and will set the player back to creative so that they can gear back up.



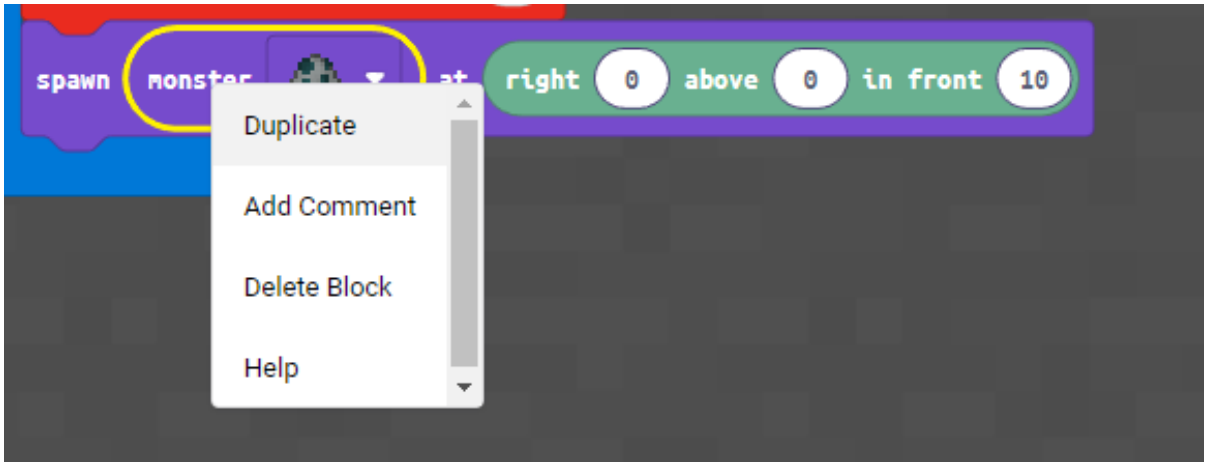
Finally, the bulk of the code; to begin, find the "if ____ then" block from the **Logic** tab. While still in the **Logic** tab, find the "__ = __" block, and drag it in carefully to the gap in the "if" block. Then, go to the **Variables** tab, and drag in both "enemies defeated" and "wave" into either side of the "__ = __" block.

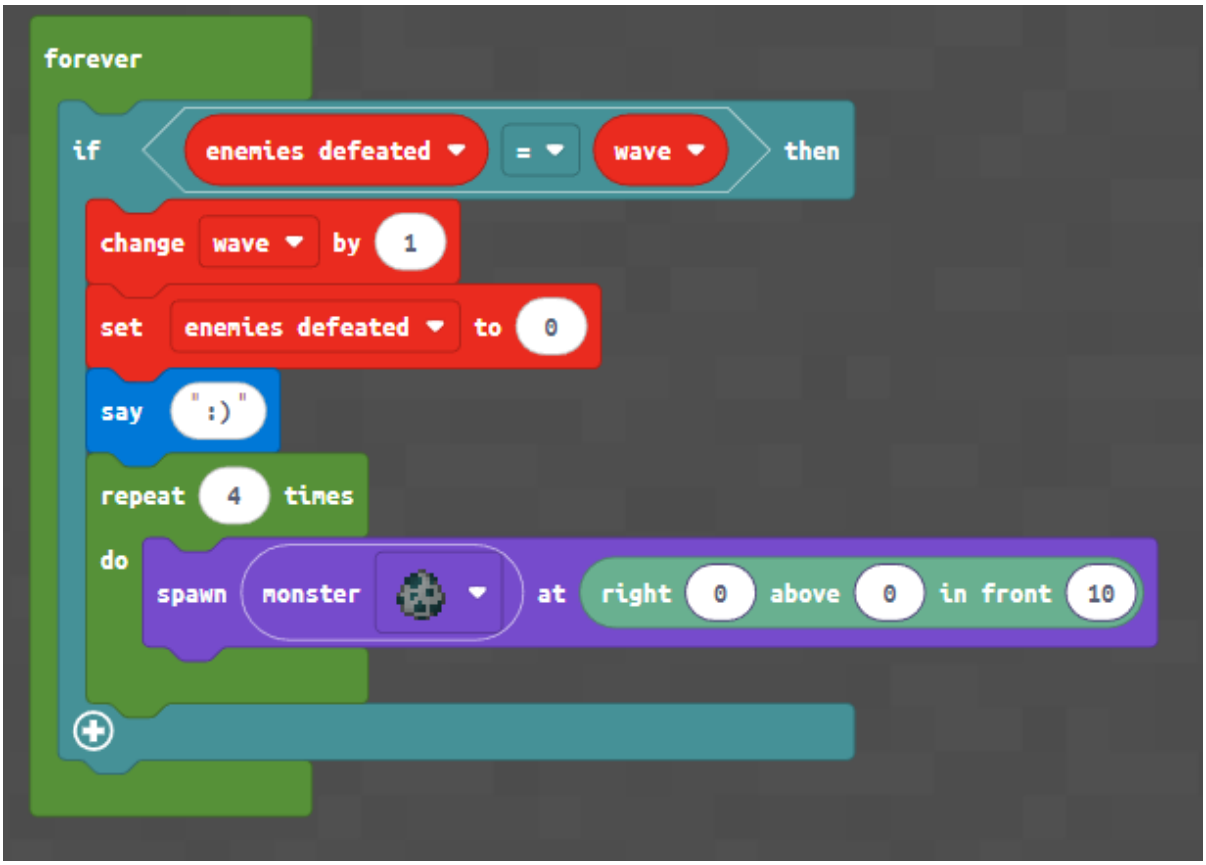


Once this is done, stay in the **Variables** tab, and drag in 1 block that says "set ___ to __", changing it to "set enemies defeated to 0", and then drag in a "change __ by __" block, changing it to "change wave by 1". After this, drag in a "say __" block from the **Player** tab.

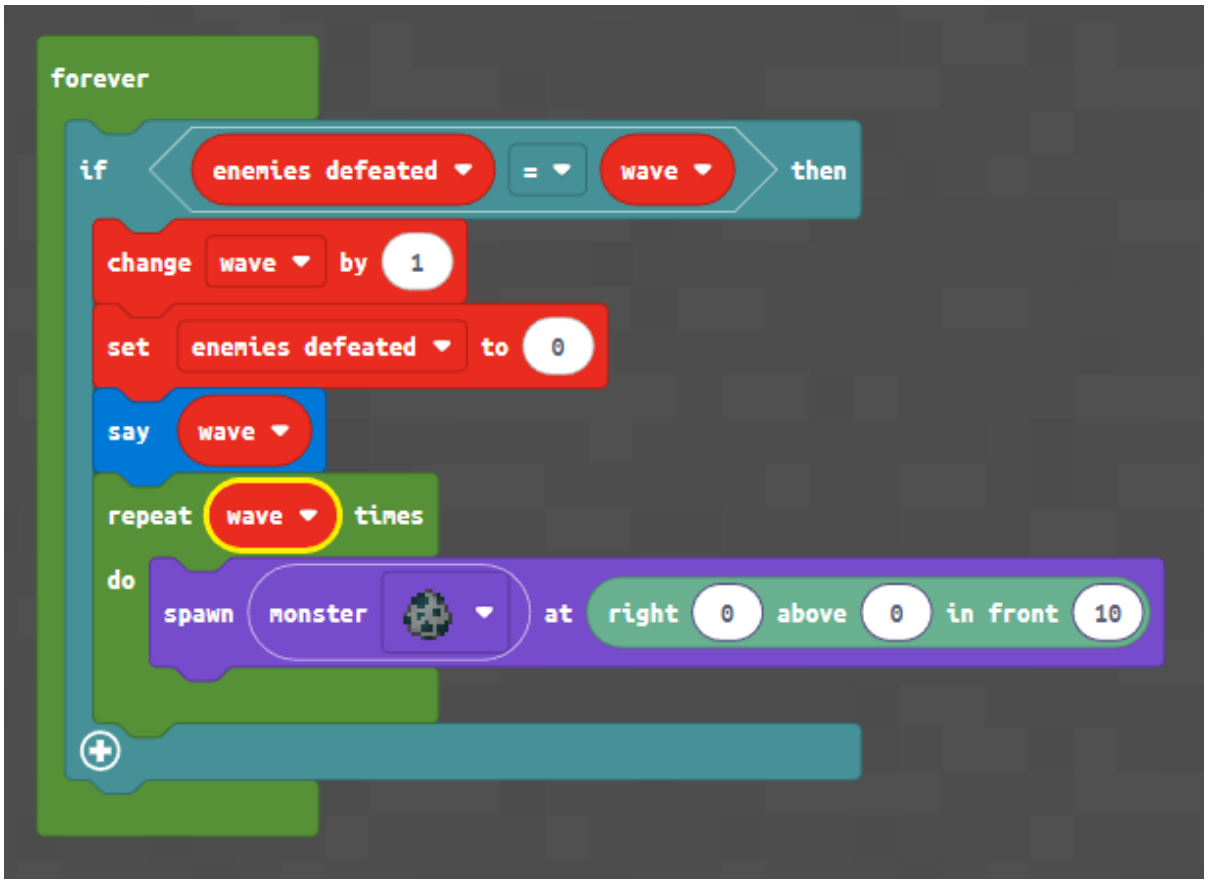


Next, we want to drag in the "repeat __ times" block from the **Loops** tab. After this, already inside of our code, find the "spawn monster block" from our "on chat command start" code. Then, right click on the block and select 'duplicate', and drag the copy into our "repeat __ times" block.





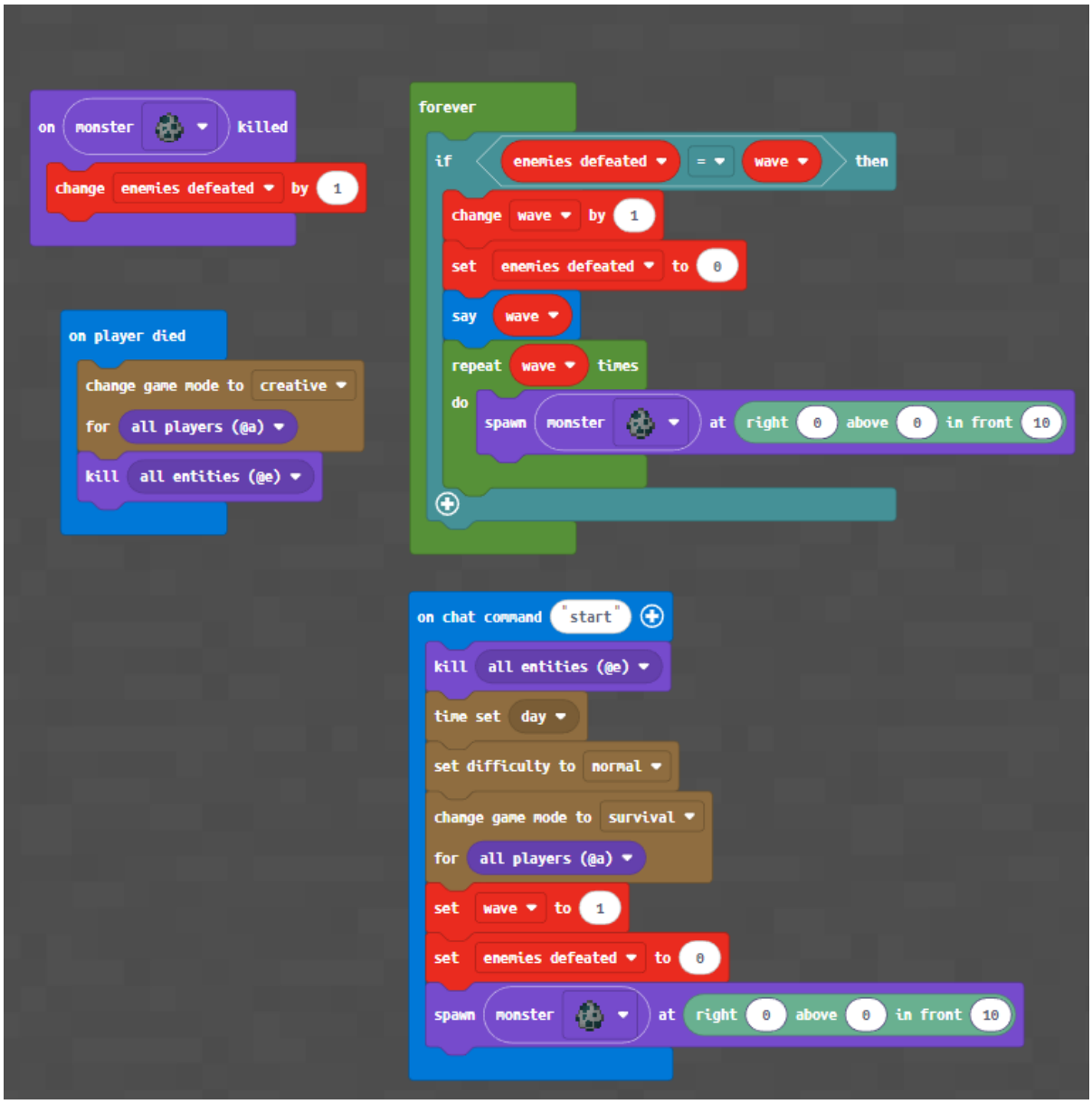
As a final step, go to the **Variables** tab, and drag in the "wave" block into the "say" block and the "repeat __ times" block. After this, the code will be complete!



Students were then encouraged to make an arena, see how far they could get into the game, or create any of their own creative additions to the game.

To begin the arena, simply type in chat "start" once you've geared up .

Code altogether:



Save your project and test the code before continuing.

Continue at your own pace.

Lesson 3

Summit Middle: Coding with Minecraft - Lesson 3 - Review

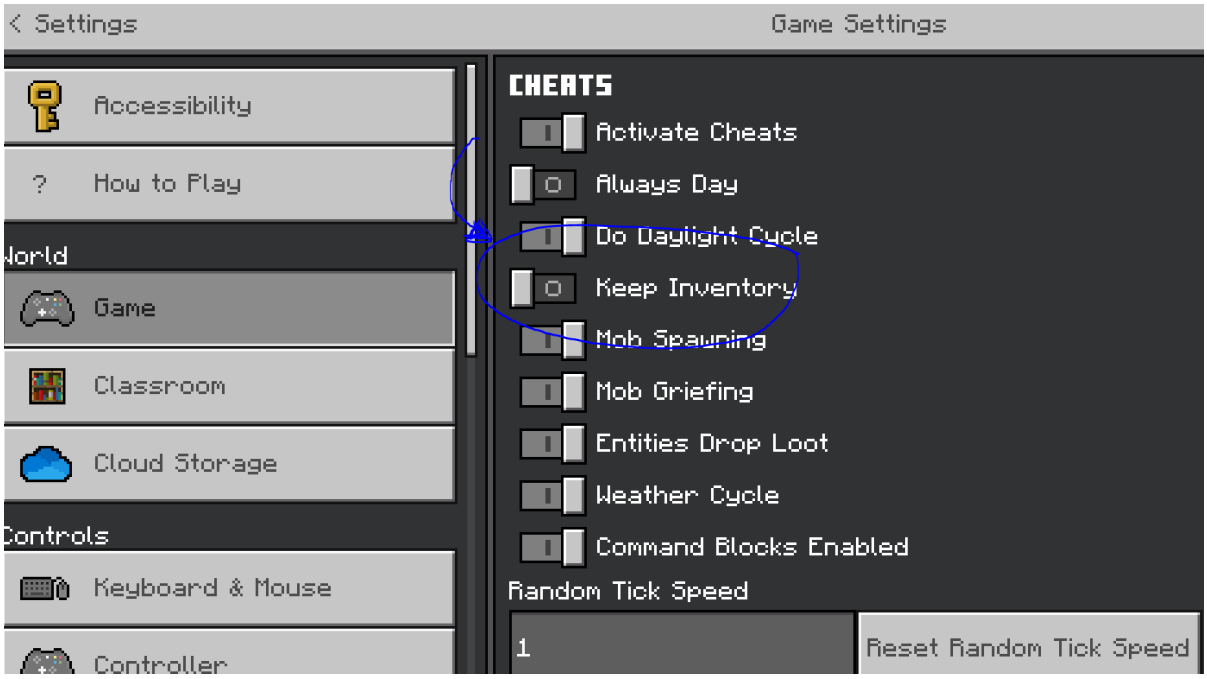
Lesson overview

Continue building the wave game from Lesson 2 using the instructions below.

This lesson builds on the previous project by adding gameplay features and new blocks.
The following lesson begins a separate project.

We explored the **Text** tab, learned some new control with the "pause" block , learned some new ways to use variables, and learned how to give the player starting items.

To begin, we ensured that students who had set the "Keep Inventory" setting to be true changed it to be false again. To do this, go into "settings," then "Game," then scroll down to find "Keep Inventory" and set it to be false.

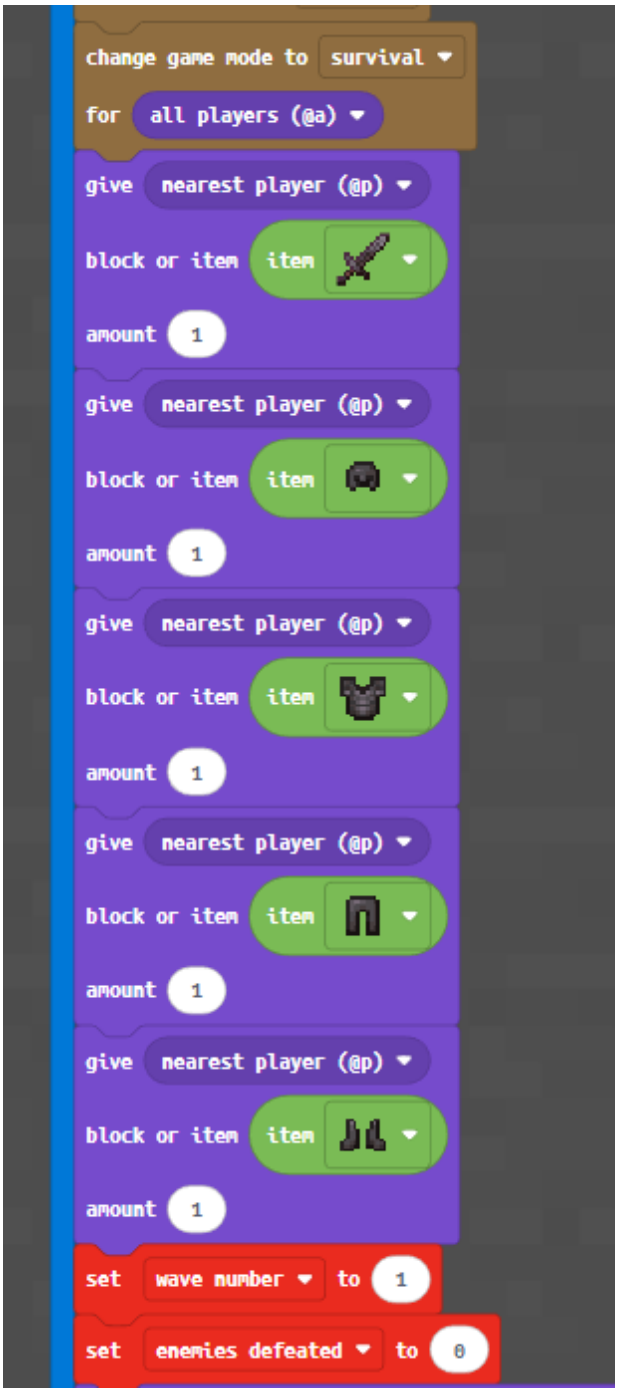


The reason we are doing this is because we are going to be assigning the player some starting weapons and loot so that they can be best prepared for the fight, and we'll be able to set this loot through code.

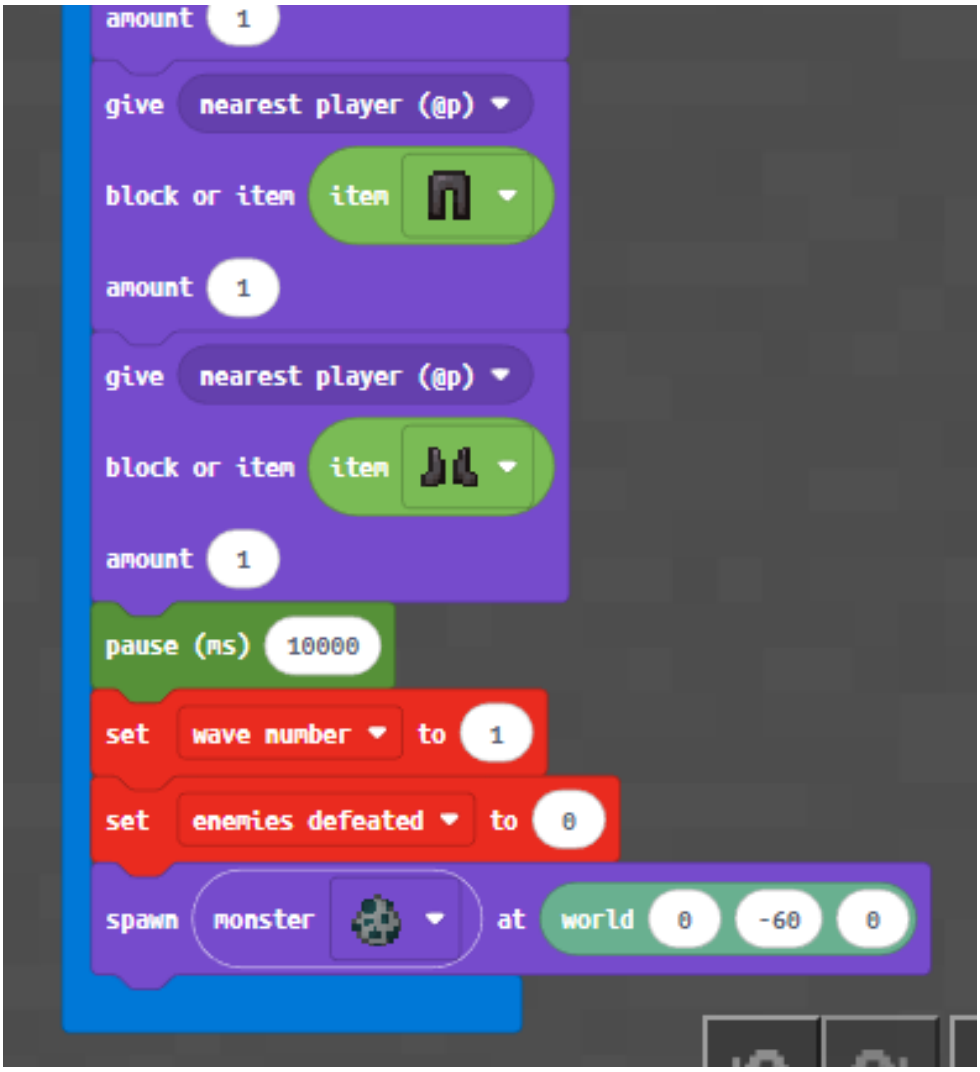
To start, go into the **Mobs** tab and find the block that says "give 'nearest player (@p) block or item __ amount __", and drag it into the coding space. We'll want to place this block inside of our "on chat command" function underneath our "change game mode to survival" block. This block will give an item of our choice to the nearest player.

Next, go over to the **Blocks** tab and drag in the "item" block to the slot after "block or item". Once this is in, select whatever item you want the player to start with. Myself, I'll select a sword.

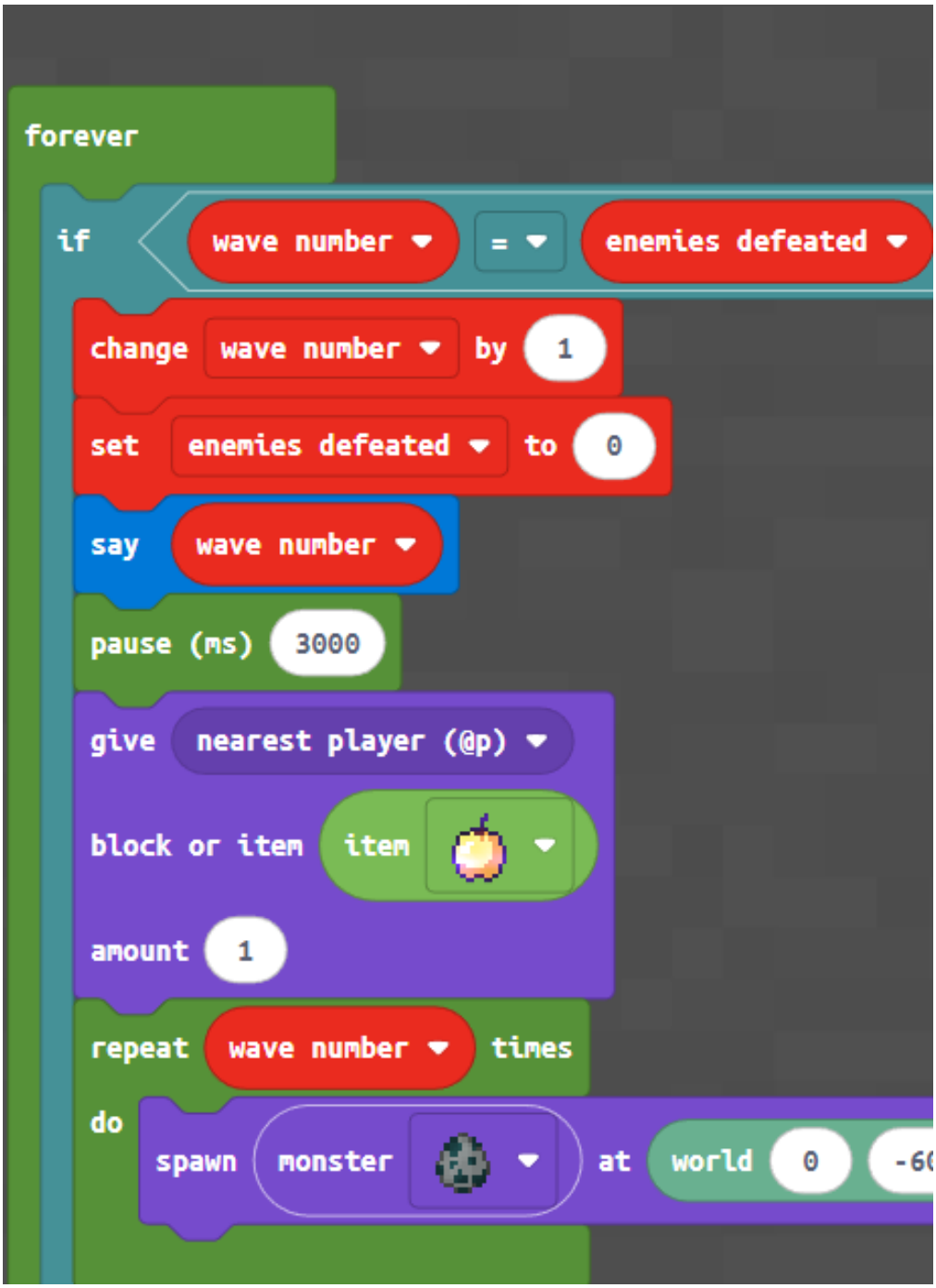
Just a weapon isn't enough though, and we need our character to have some armor as well. To add more items for the player to start with, simply duplicate the block by right clicking on it and selecting "duplicate", and then place the block underneath and select the item to be a piece of armor. Repeat this for all armor pieces, and for any other items you'd like the player to start with.



Next, we learned the "pause (ms) __" block. This block adds a pause in between blocks of code, and has a value in milliseconds that we can modify. 1000ms or milliseconds is equal to 1 second. We found this block in the **Loops** tab and drag it in underneath the "give nearest player" block, setting the value to be 10000ms or 10 seconds. This ensures that the player has time after receiving their items to equip them and prepare.

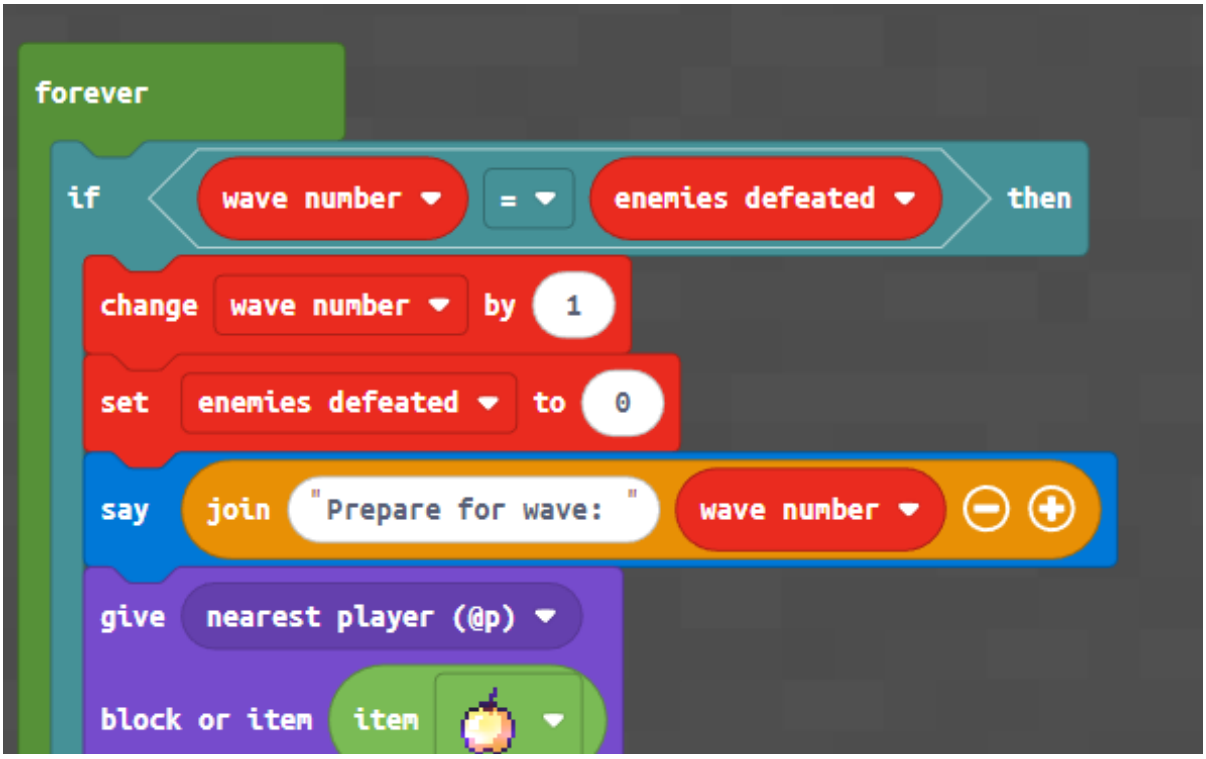


We then implemented the same code in between rounds. To do this, we duplicated the "give item" block again, but this time pasted it after the "say wave" block. Then, we set the item to be a golden apple so that the player can heal between rounds. To give the player some time to catch their breath, we again used the "pause" block from the loops tab, but only set the value to be 3000ms or 3 seconds.



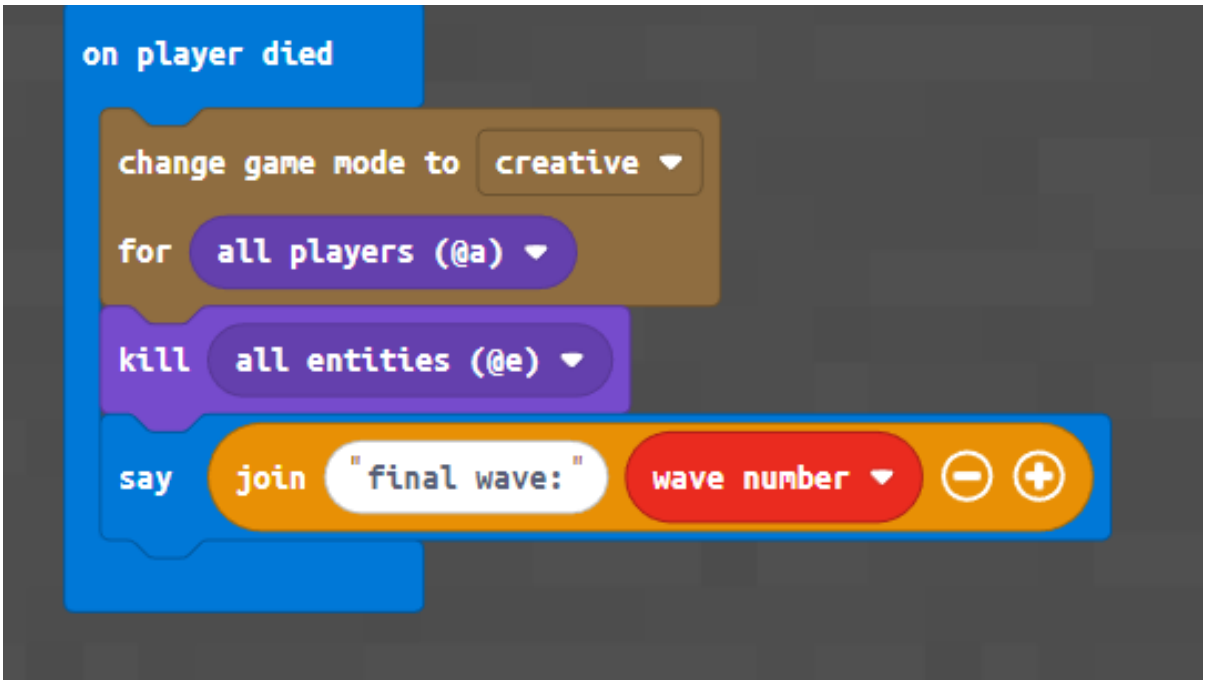
After this, we learned how to use blocks from the **Text** tab. Mainly, we used the "join" block. The "join" block joins the value of a variable or some other block to text. For example, if we wanted to represent some variable that kept track of score in text, we would join the word "Score: " to the score value so that our text would read "Score: 10". In Microsoft Make Code, we use a block called "join".

This block is found inside of the **Text** tab, which is found through the **Advanced** tab at the bottom. Drag the "join" block into the "say" block above where we just placed our "give golden apple" block. Select the first part of the block and type "Prepare for wave: ", and then go to the **Variables** tab and drag in the variable for "wave number" into the second part of the block. This will print out the current wave number before a round starts.



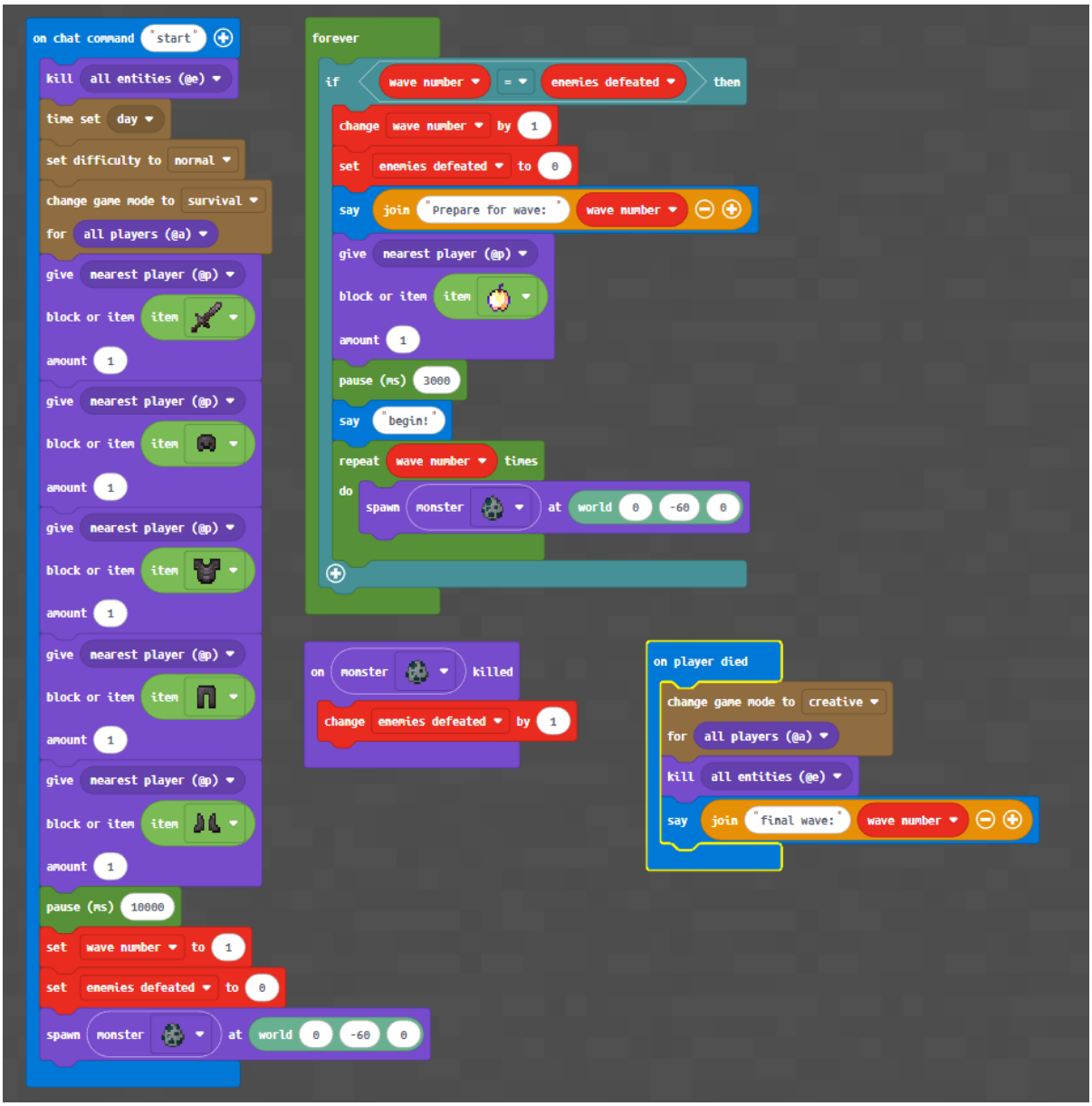
Then, after the "pause", we want to announce when a new round has begun, so find the "say" block from the **Player** tab and make it say "begin!".

Finally, we want to print out the final wave when we lose the game, and to do this, we'll use the same code as before. Find our other "say" block with the "join block" inside of it, and duplicate it. Then, place the duplicated block into the bottom of our "on player died" function. This will print out the final wave at the end of the game.



Thats all of the code!

Here's the final code for this lesson:



Save your project and test the code before continuing.

Continue at your own pace.

Lesson 4

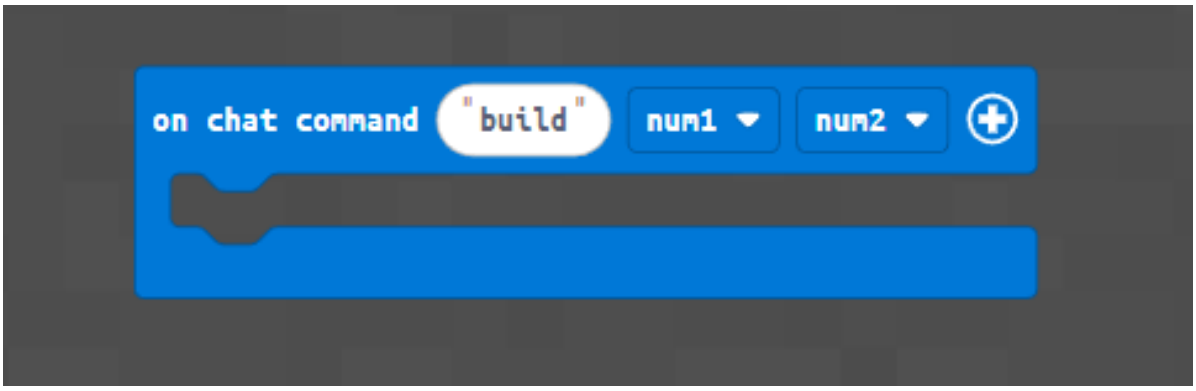
Summit Middle: Coding with Minecraft - Lesson 4 - Review

Lesson overview

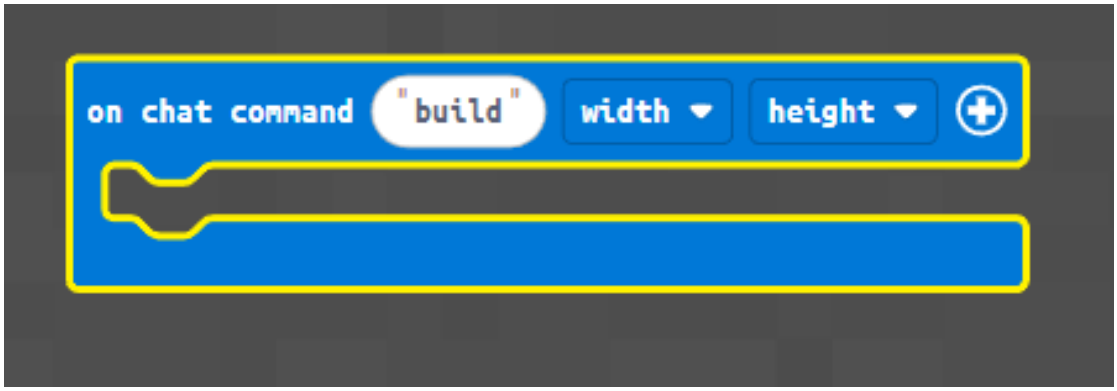
In this lesson we explored the Shapes and Builder tabs, and we learned how to automatically create different structures and shapes while exploring functions that can receive variables.

First, players must make sure that they are playing on a super flat world, which is a setting selected while creating a world that allows a completely flat and clear world.

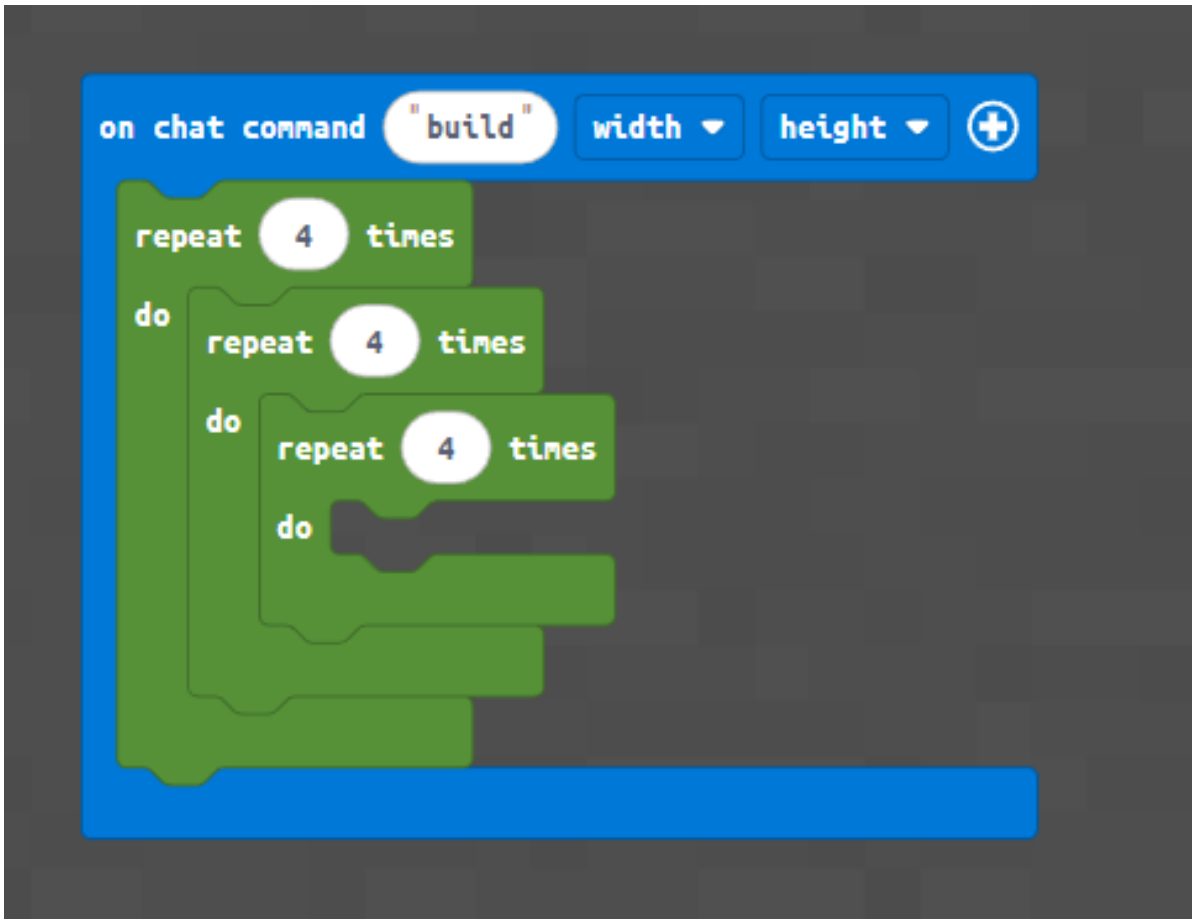
Then, to begin, remove all blocks and drag in a "on chat command" block from the Player tab. Make the text say "build", and then click the plus icon twice to add two variables called "num1" and "num2" or something similar.



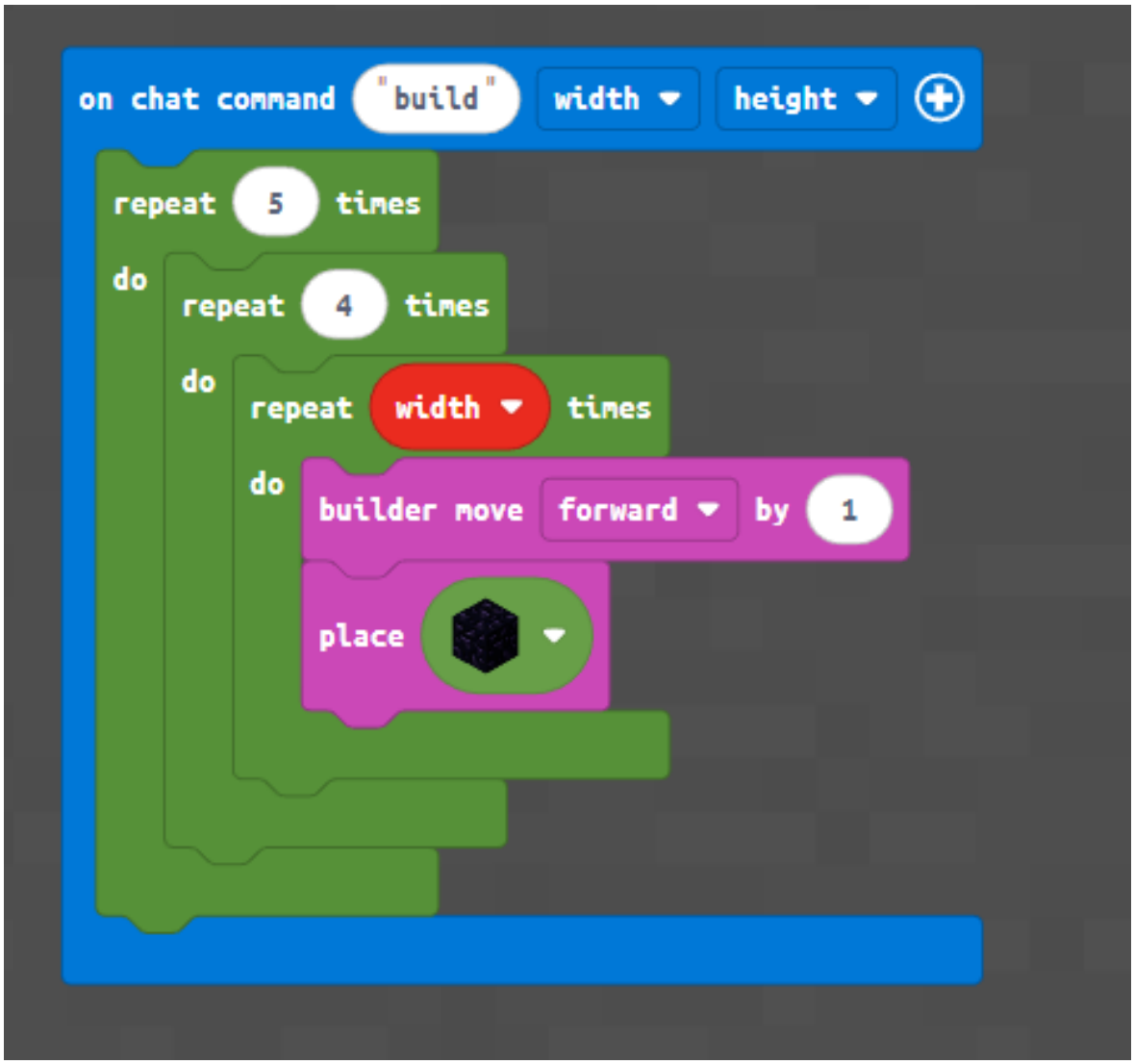
Next, select these variables, and at the bottom of the dropdown is an option saying "Rename Variable...". Select this, and rename the first variable to "width" and the second to "height". What this means is that the chat command will look for the word "build" and then two input numbers which will be the width and height of the function.



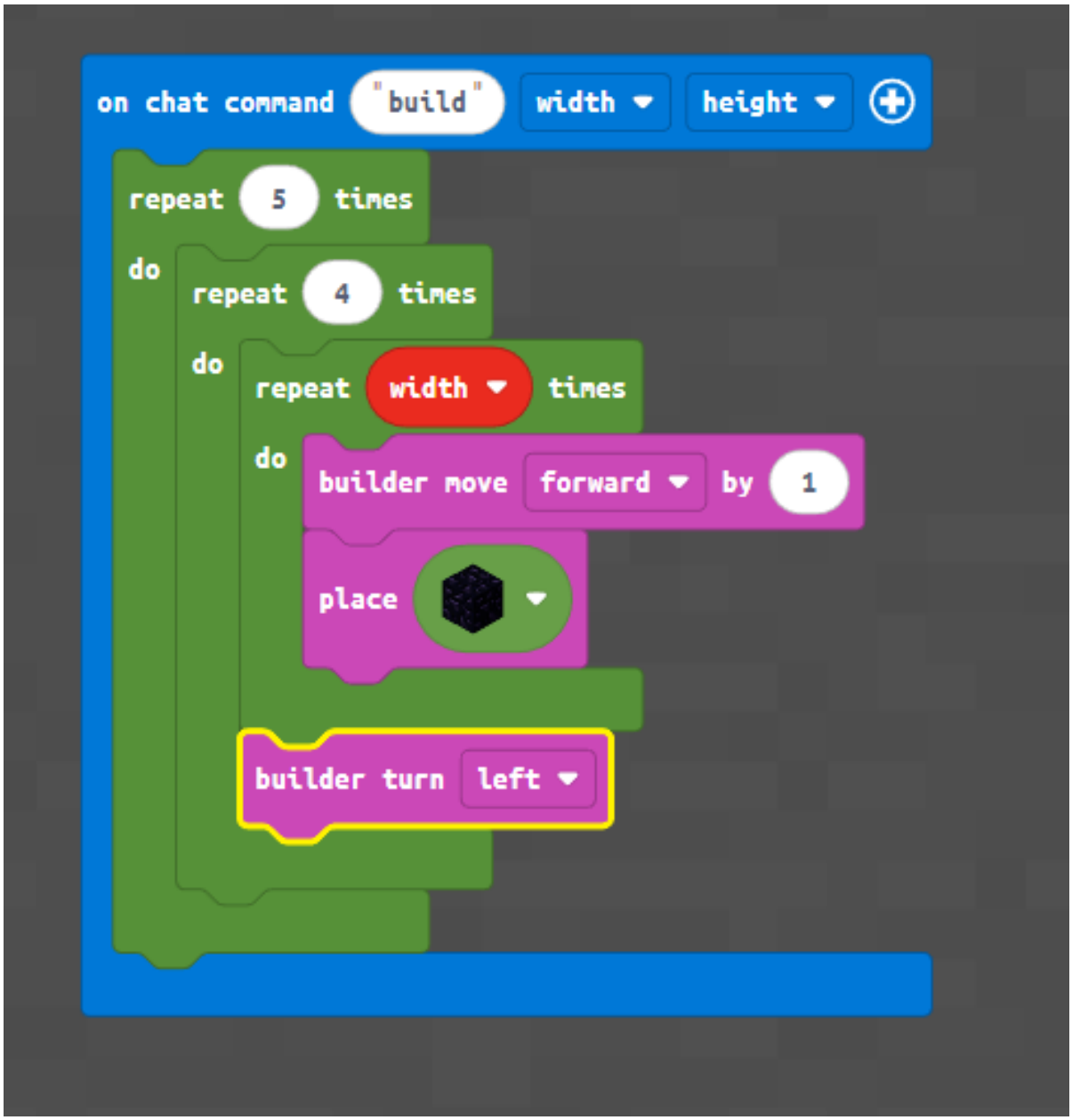
Next, go over to the loops tab and find three "repeat __ times" blocks. Drag them into the code so that they are all nesting within each other.



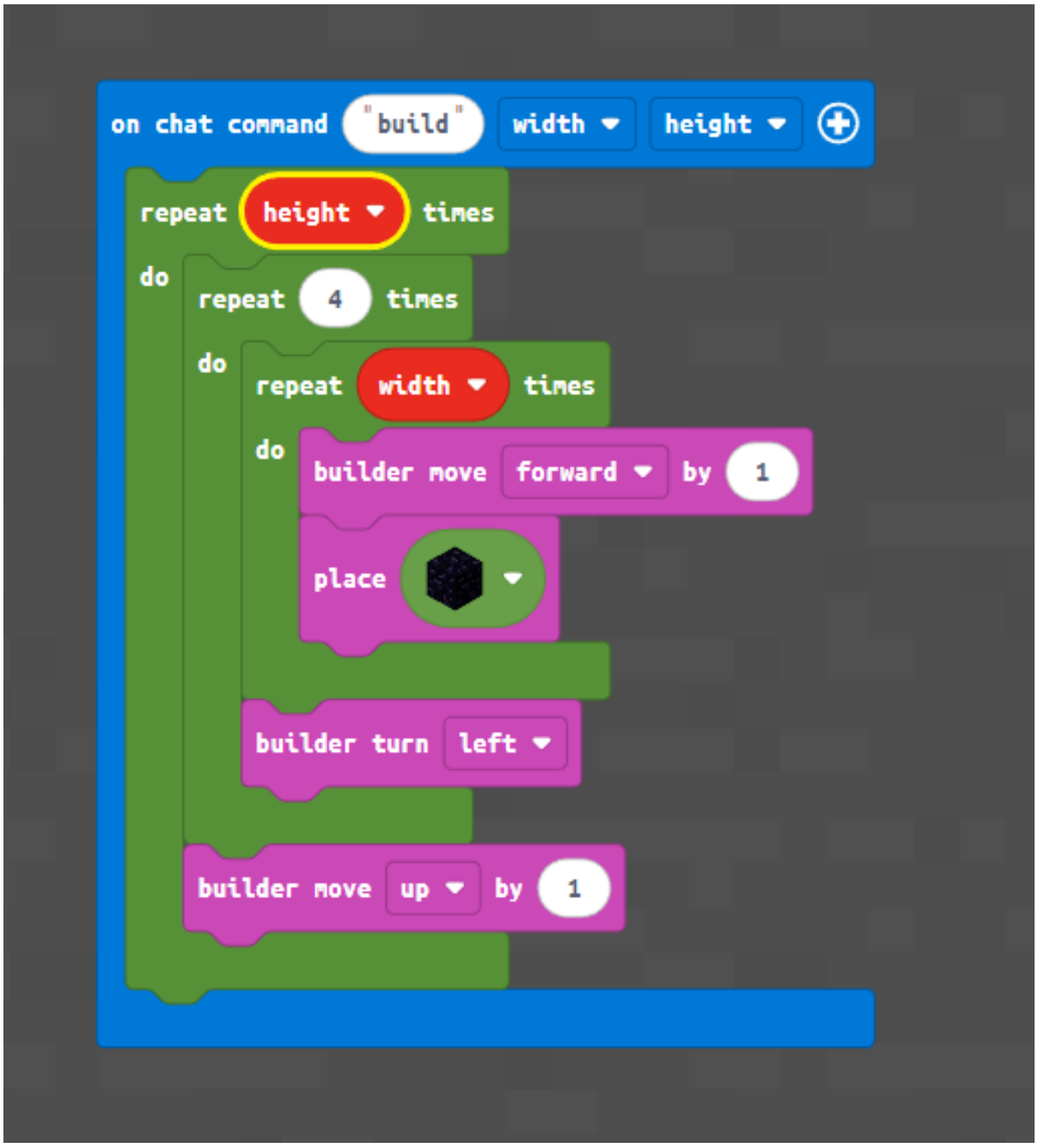
Then, to the innermost block, go over to the Builder tab (found in the Advanced tab) and drag in the "builder move forward by _" block and the "place _" block. After that, drag in the "width" variable from the variables tab. What this will do is move the builder forward and place a block, and it will repeat this 'width' amount of times to create a line.



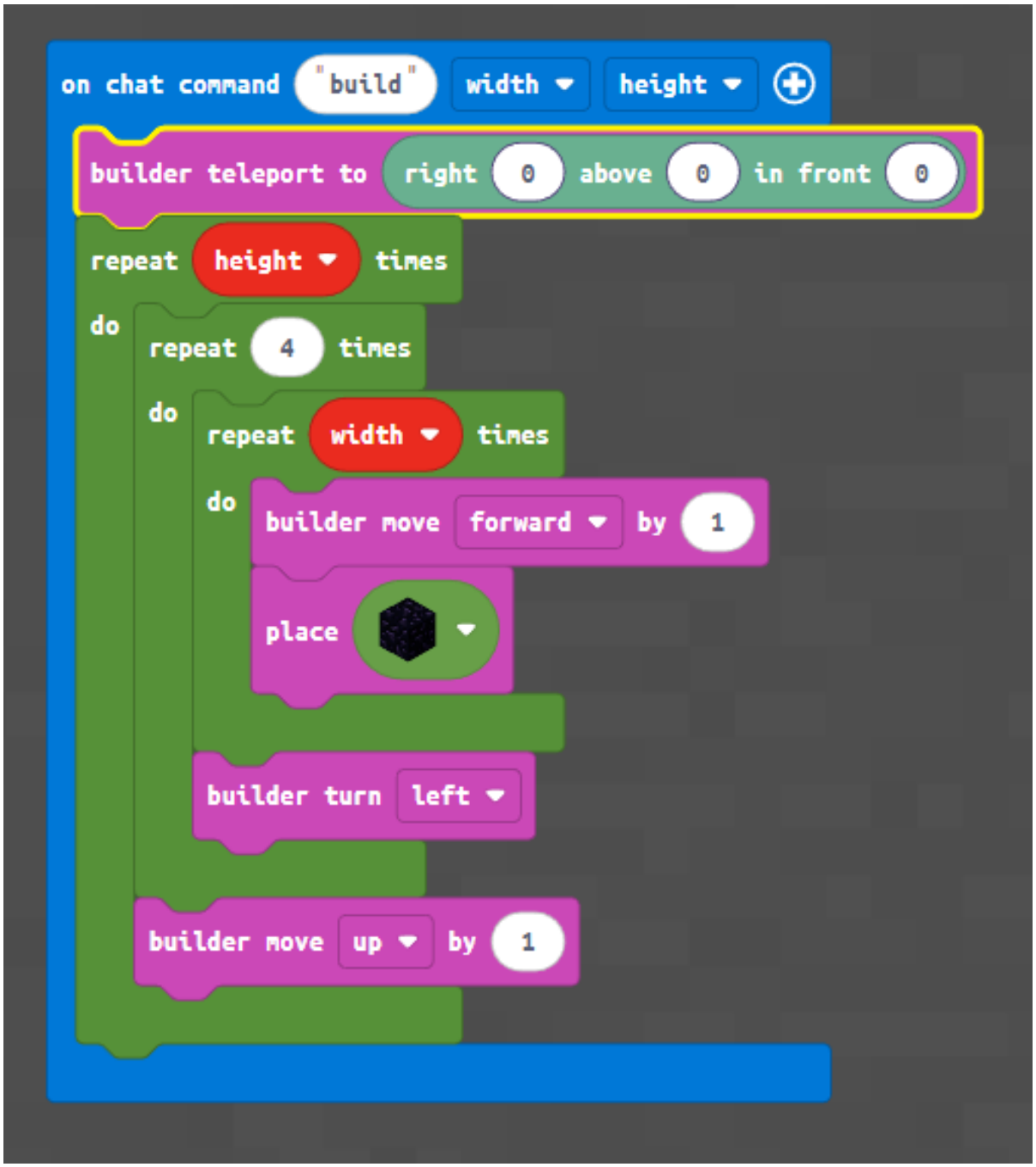
Next, to make multiple lines that can form a square, we need to rotate the builder after a certain amount of blocks. To do this, go underneath the previous "repeat" block, and drag in one more block from the Builder tab: "builder turn __", setting the __ to be left, and making sure that the "repeat" block has "4". This code will now create an empty square.



Finally, to create the height component of this square, go back to the Builder tab one more time, and add in the "builder move __ by 1" to the very bottom "repeat" block, and make sure it says "builder move **up** by 1. Then, drag in the "height" variable from the Variables tab into this top repeat block.



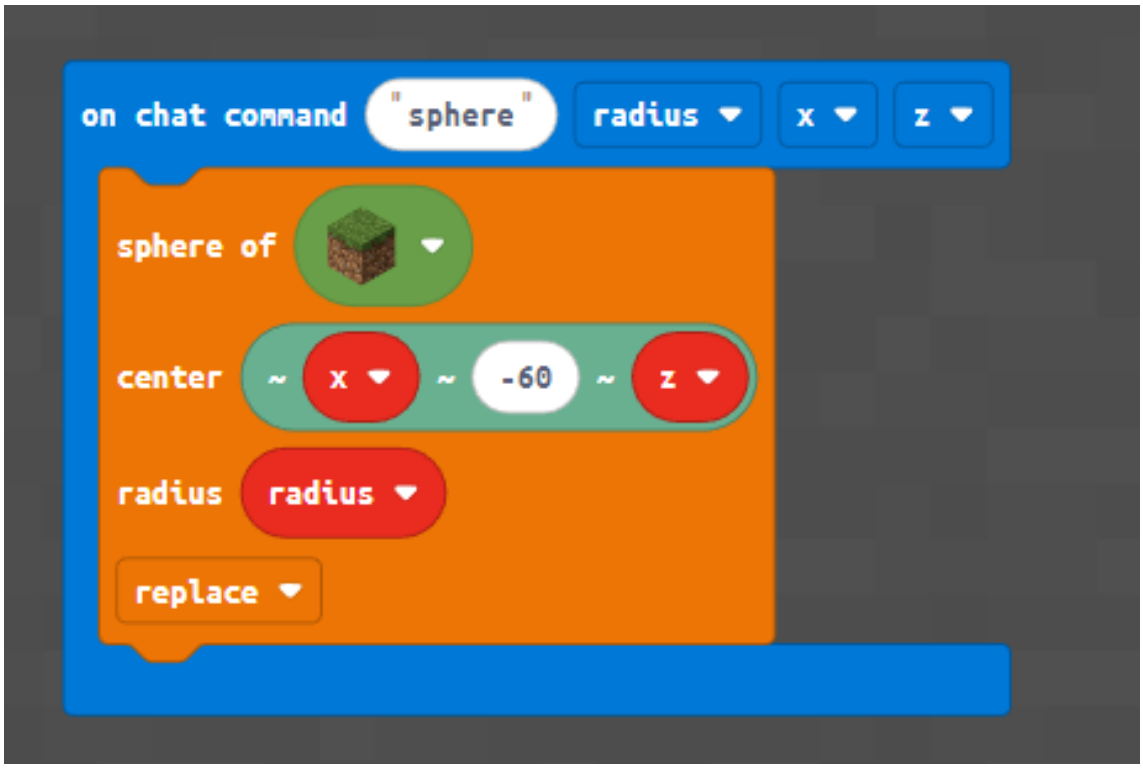
As one final step, we want to make sure this happens at our position, so drag in the "builder teleport to ~0 0 0" block at the very top of the code.



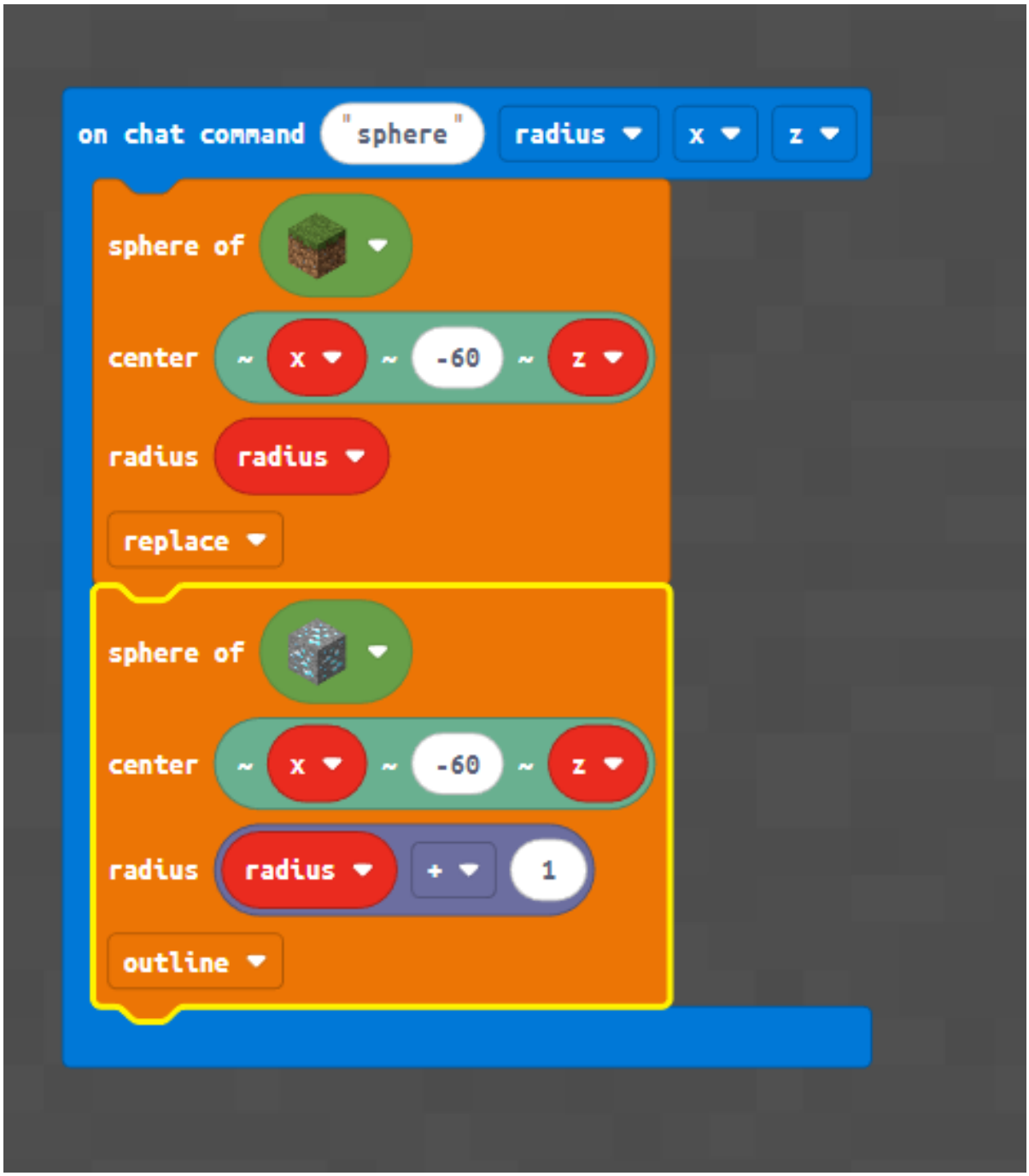
This is our complete code to generate a box! To access it, type in chat "build" followed by 2 numbers for the width and the height.

Next, we wanted to create a sphere. To do this, we made a similar function for the "on chat command" block, so drag another one in. Set the command to be "sphere" and use the "Rename Variable..." option to create 3 variables, "radius", "x", and "z".

Next, go over to the Shapes tab and drag in the "sphere of __" block. Where it says "center," drag in our x and z variables from the Variables tab so that the block reads "x, -60, z". Finally, set the radius to be our "radius" variable from the Variables tab. Make sure the bottom option reads "replace".



Then, we want to create an outline for our sphere, so right click and "duplicate" this sphere block. Then, go over to the "Math" tab and find the block that reads "__ + __" and drag it into the radius space. Then, go to the Variables tab and drag "radius" into the blank space, and type "1" into the other so the block reads "radius + 1". Finally, select where it says "replace" and change it to "outline".



The code is done! To create a sphere, type in chat "sphere" and then your radius, x position, and z position you want to spawn it at. To access the players coordinates (x and z position), go into "settings" and toggle "Show Coordinates" to be true.

Here's the final code:

Save your project and test the code before continuing.

Lesson 5

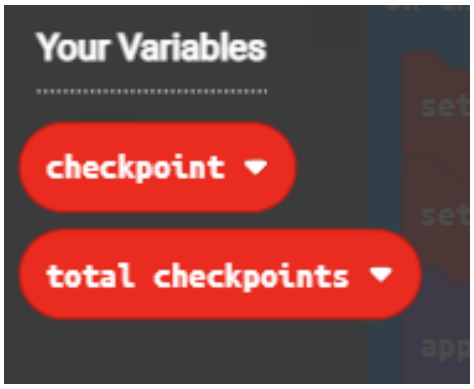
Summit Middle: Coding with Minecraft - Lesson 5 - Review

Lesson overview

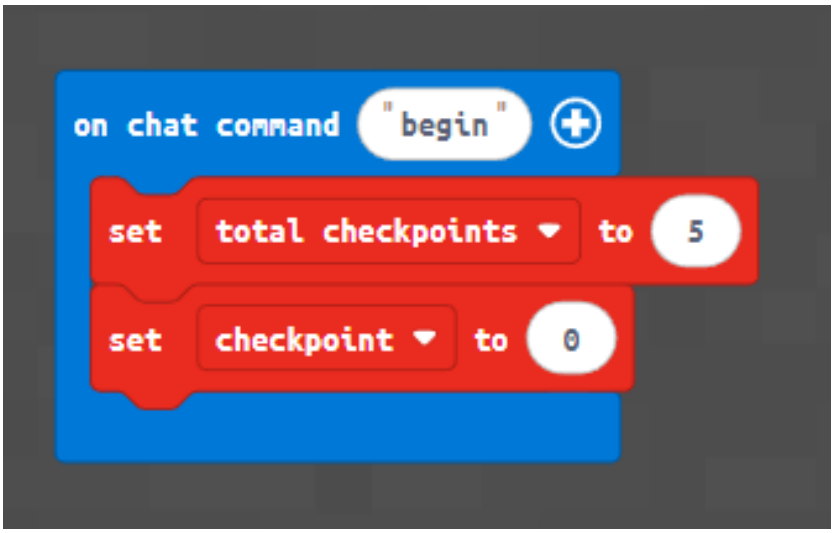
In this lesson we created a racing game with speed boosts and checkpoints!

With this lesson, we further explored the Logic tab and If block, learned the "test for" block, and also improved our knowledge of Text, coordinate, and Math blocks.

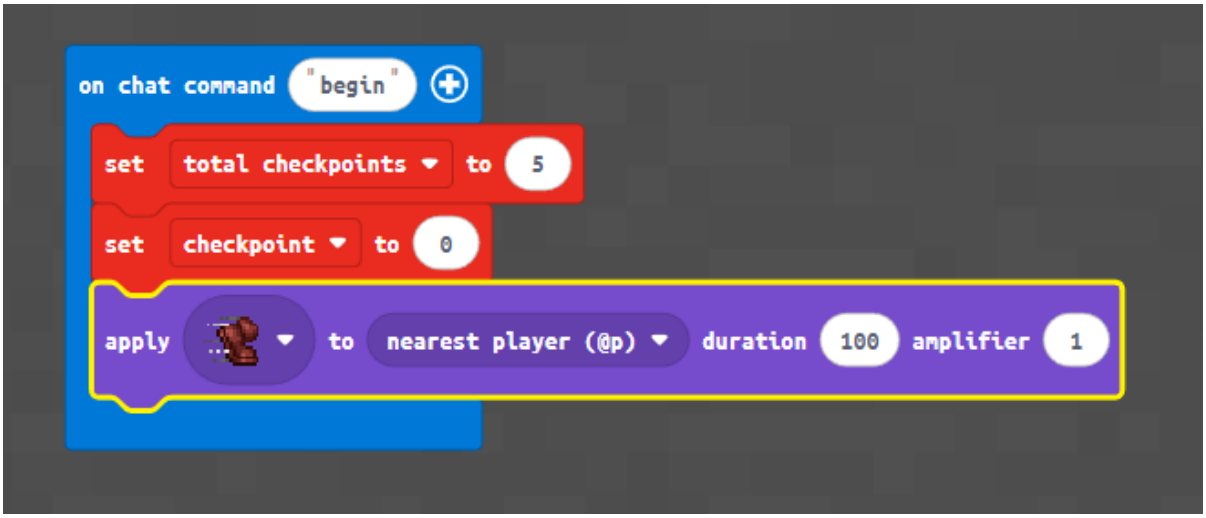
To begin, we created two variables in the red **Variables** tab: "total checkpoints", and "checkpoint". The total checkpoints variable will be used to let our code know how many checkpoints we'll have in our course, and the checkpoint variable will be used to keep track of how many checkpoints the player has reached.



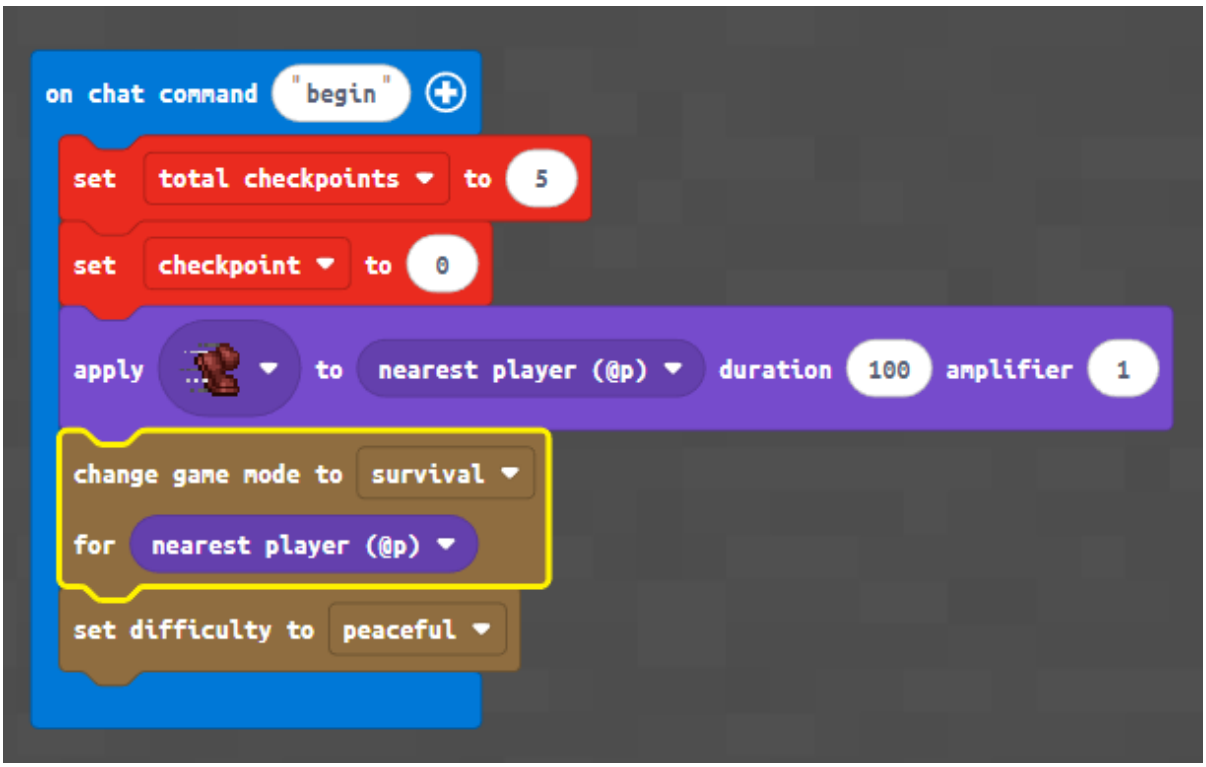
Next, we'll create our "begin" function. First, drag in an "on chat command" block from the blue **Player** tab, and then set the text to be "start". Drag in two of the "set" blocks from the red **Variables** tab, making them read "set total checkpoints to 5" and "set checkpoint to 0".



After this, we'll go into the purple **Mobs** tab and find the "apply __ to __ " block. This block can be used to apply different effects to the player. Select the boot icon (the speed effect) and set the duration to 100 (seconds) and amplifier to 1.

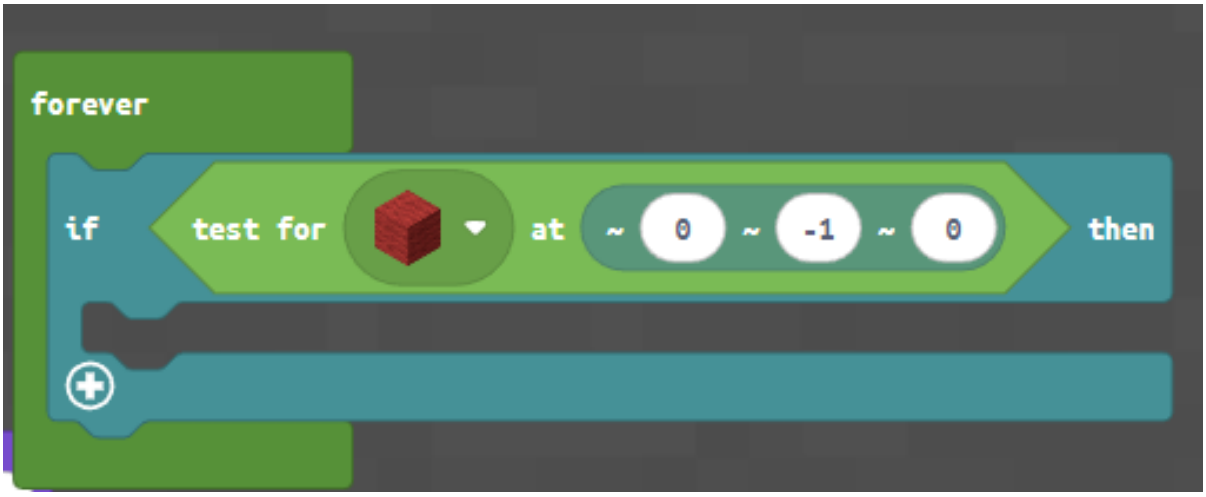


Finally, go to the brown **Gameplay** tab and find two blocks: "change game mode to", and "set difficulty to". We want the first block to change game mode to survival, and the second block to set difficulty to peaceful (so no mobs get in the way of our race). We've now completed the begin function! To start the race, simply type "begin" in chat.

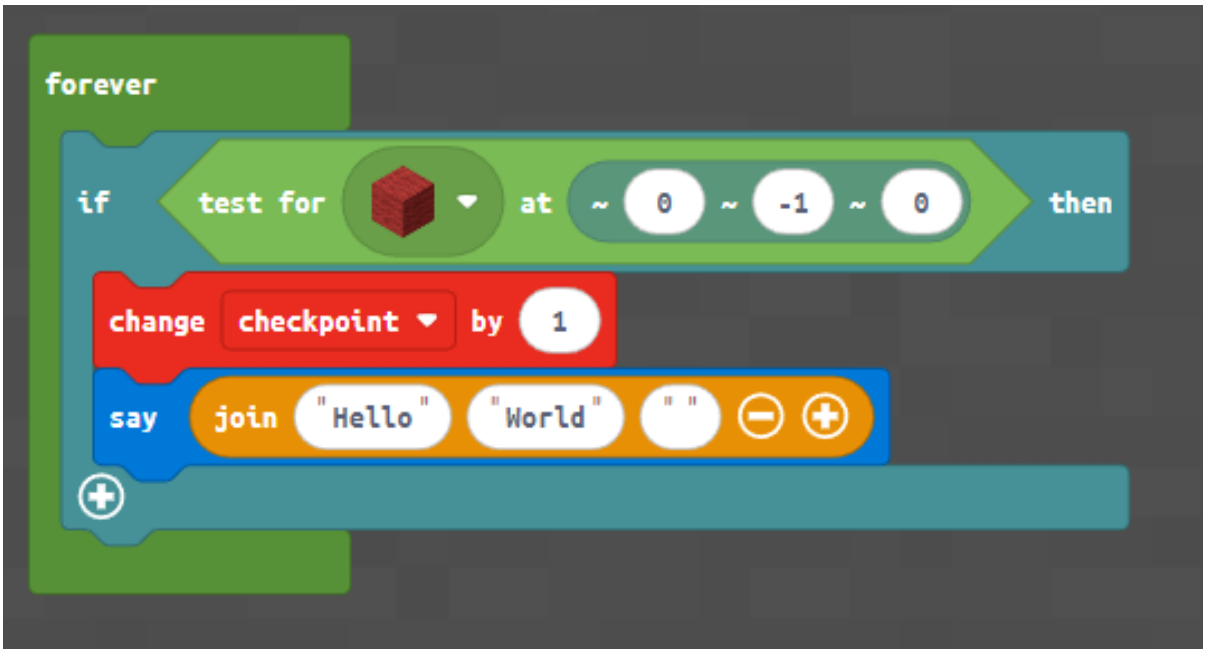


Next, what we need to do is add checkpoints at different sections of the race, and then a checkpoint for the finish line which will check that we haven't skipped any other checkpoints during the race.

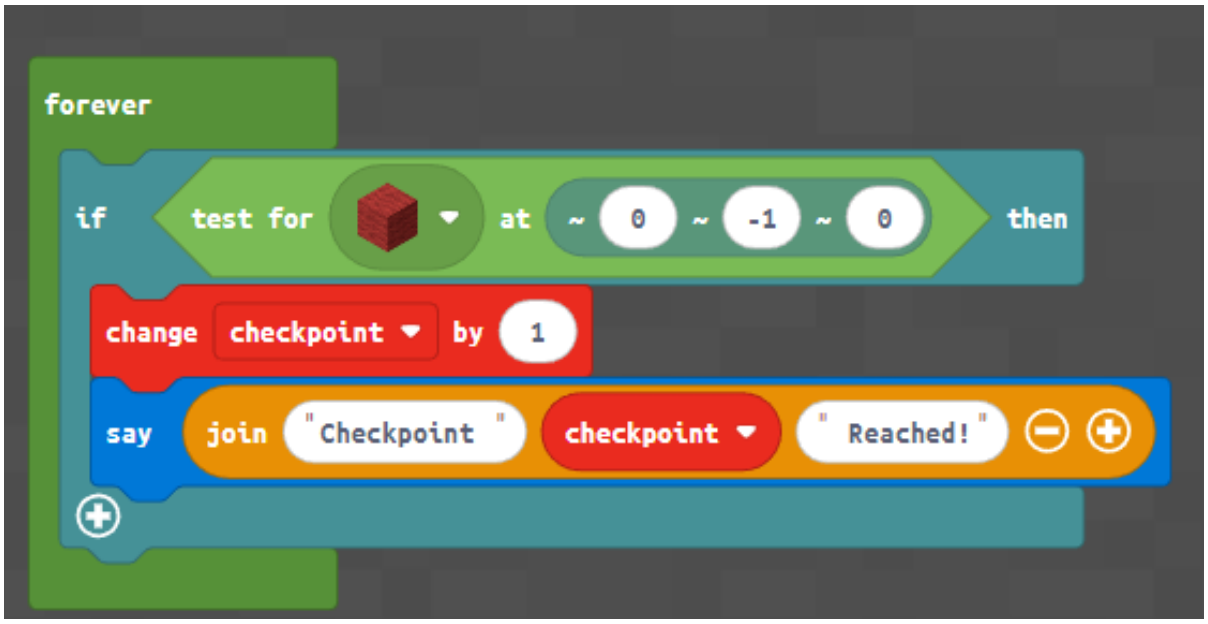
First, we'll create regular checkpoints. To do this, drag in a if block from the blue **Logic** tab, and then a "test for __" from the **Blocks** tab. Select any block that you'd like to use as a checkpoint, and then make sure the coordinates inside this code are "0, -1, 0", as we're checking the block underneath the player.



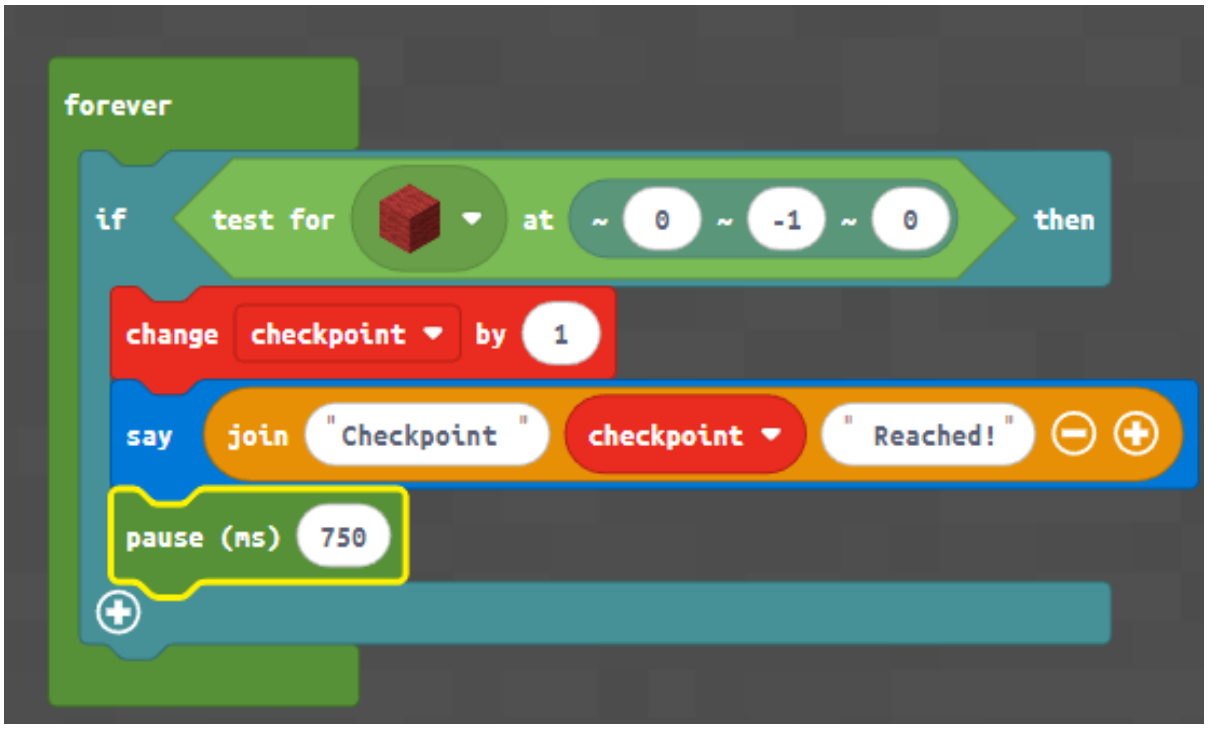
Next, go to the red **Variables** tab, and drag in the "change" block, making the code read "change checkpoint by 1". After this, drag in a "say" block from the blue **Player** tab, then inside of it, drag in a "join" block from the yellow **Text** tab (found by clicking '**Advanced**' on the bottom of the tab list).



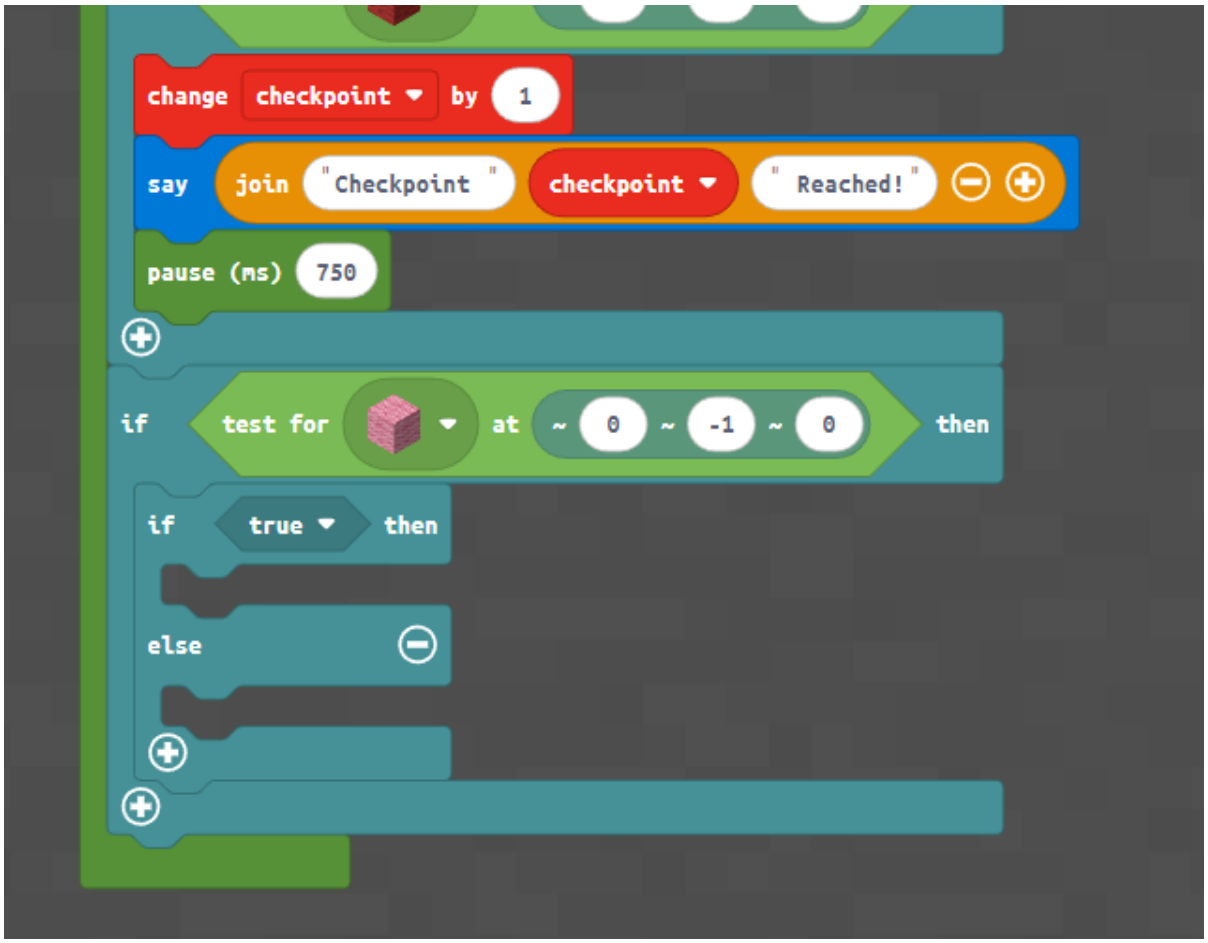
After this, select the plus icon on the join block so that you have three spaces in the block. In the first space, type "checkpoint", making sure you leave a space after the word. In the second, drag in the "checkpoint" variable from the red Variables tab, and in the third space, type "reached!", making sure that there is a space before the word. The reason we do this is because the code doesn't know to place a space in between our words when we're joining them together, so we must do this ourselves.



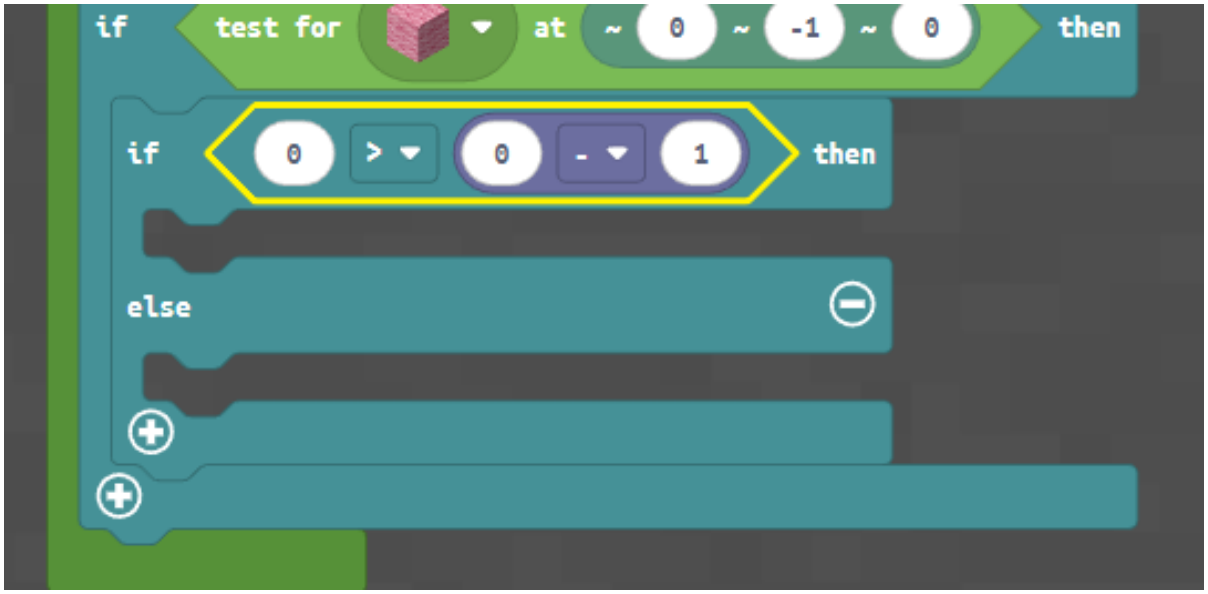
Finally, drag in a "pause (ms)" block from the green **Loops** tab, and set the number inside to be 100. This will ensure that the checkpoints don't instantly repeat if we stand on top of them.



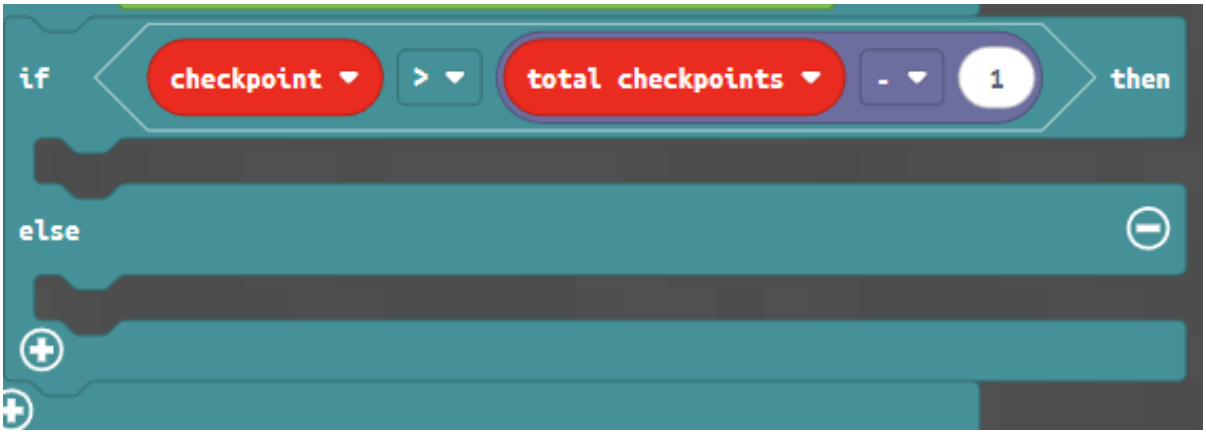
Next, we need to make the code that checks for the finish line block. To do this, simply duplicate (right click + select 'duplicate') the first "if" block, and delete everything inside of the block. Then, change the block in the "test for" block to be something different that you'd like for your finish line. Finally, drag in from the blue **Logic** tab an "if then, else" block".



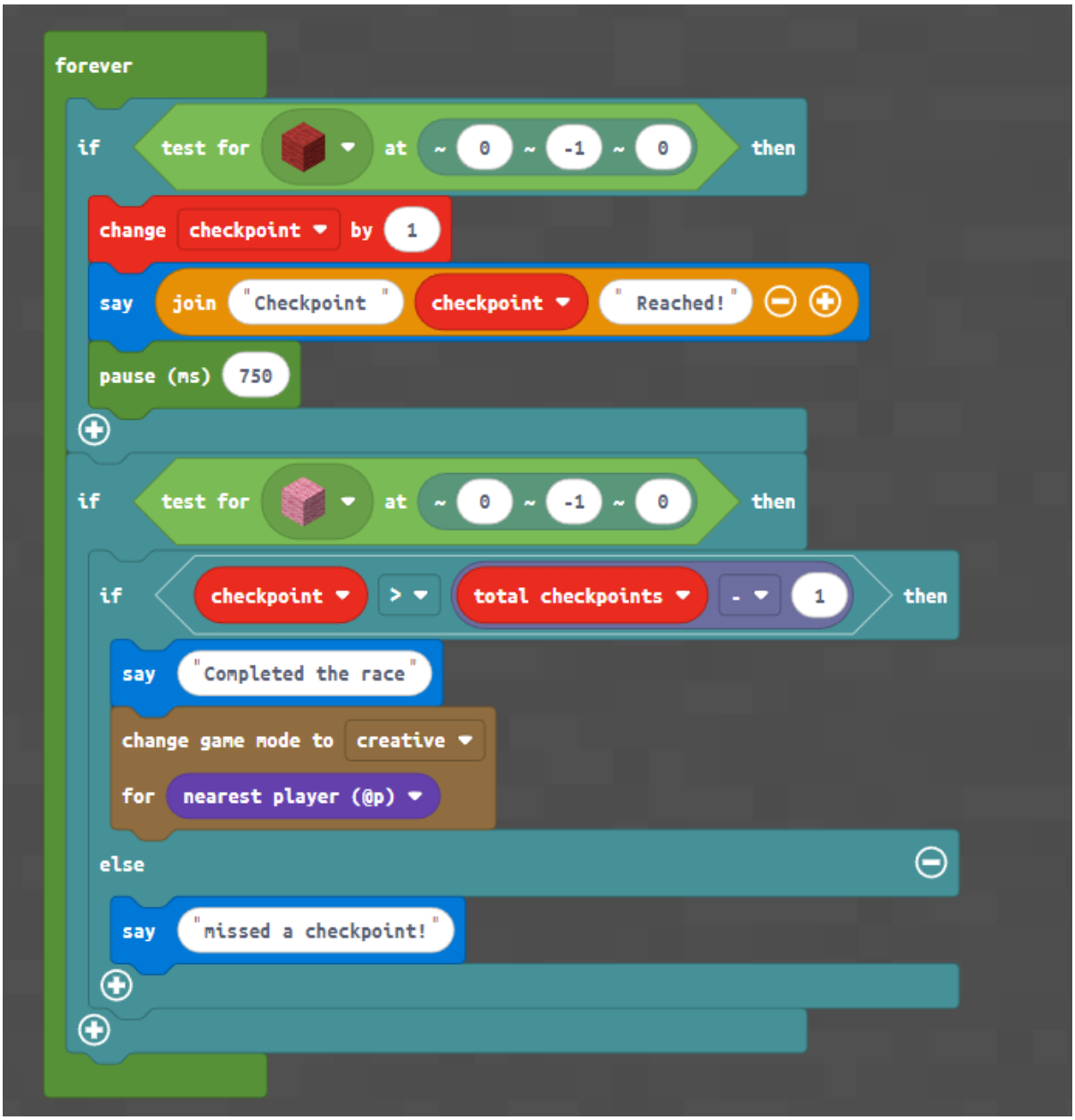
To the if block we just added, go back into the **Logic** tab, and drag in an "(__ > __)" block. Make sure that you have the correct symbol. Then, go over to the blue **Math** tab, and drag in the "(__ - __)" block.



After you've completed this, open the red **Variables** tab, and drag our "checkpoint" variable into the first space, drag our "total checkpoints" variable into the next space, and then set the final space to be 1.



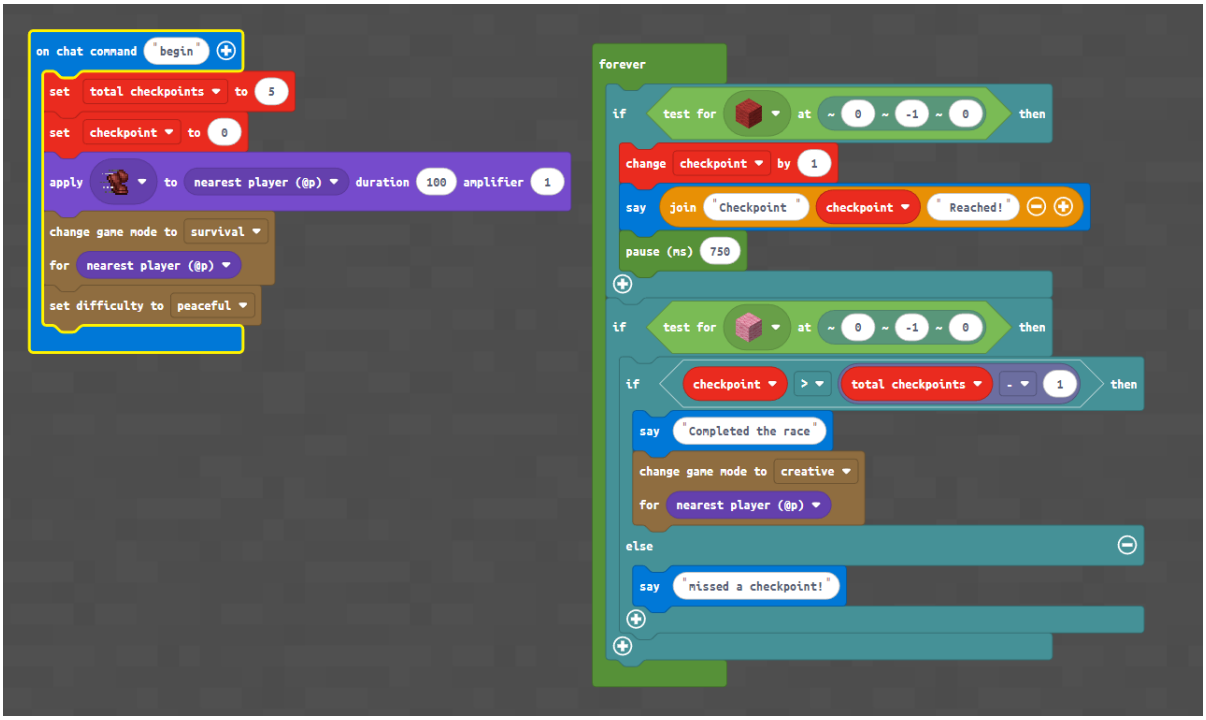
Then, go over to the **Gameplay** tab. Drag in the "change game mode to" block inside the first section of the "if, else" block. Make sure it reads "change game mode to creative". Then, as a final step, add a "say" block from the blue **Player** tab to both sections of the "if, else" block. Make the first one read "completed the race!", and the second read "missed a checkpoint!".



Altogether, this code is going to follow these steps: first, check under the player for a finish line block. If it is detected, we'll check that the player has collected all the checkpoints for the race. If so, then we will congratulate the player, and if not, we will inform the player that they missed a checkpoint.

Now the code is done!

Here it is:



After the code is created, students must make a race track for the player. This means creating 5 checkpoints (or whatever students set "total checkpoints" to be) by placing their checkpoint blocks in the racetrack, and then creating a finish line using the block they selected for their finish line.

Continue at your own pace.

Save your project and test the code before continuing.

Lesson 6

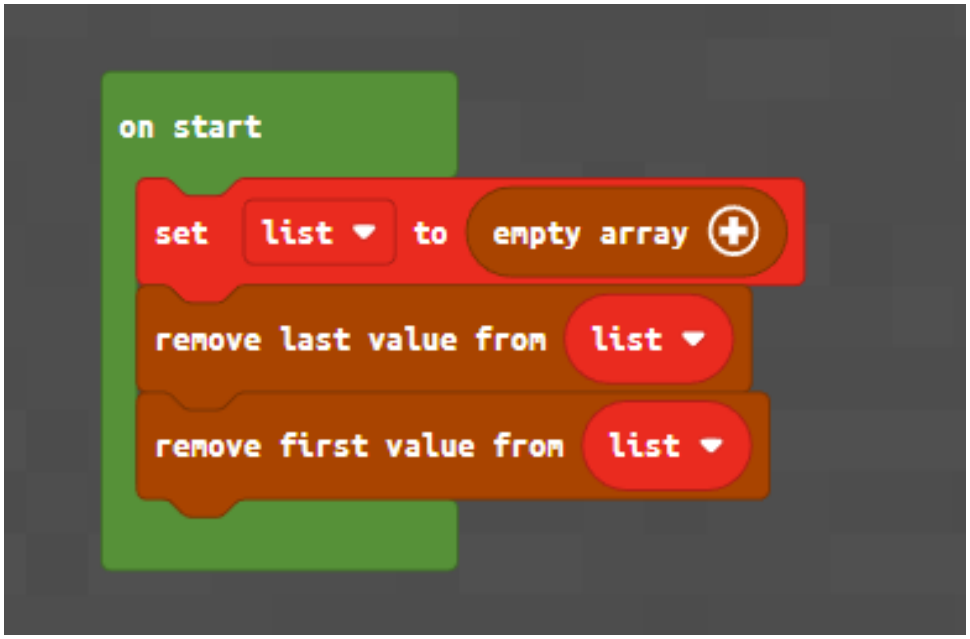
Summit Middle: Coding with Minecraft - Lesson 6 - Review

Lesson overview

In this lesson we learned a few new key concepts that are vital to coding, especially if students want to continue their programming journey.

We learned about for loops (blocks), arrays, and functions. We combined these concepts to understand how we can create neater, more efficient, and more powerful code.

To begin, we first ensured that we had an "on start" block in our space. Then, we went over to the red **Arrays** tab in the coding space, which can be found after clicking the **Advanced** tab. Drag in the first block, which reads "set list to array of _ _", and then underneath, drag in the "remove last value from list" and "remove first value from list". Finally, click the minus button on the "array of" block. This will initialize our array as being empty, and since Minecraft code isn't perfect, we also include the remove first and last blocks to make sure that the array/list starts as clear.



This project is a inventory sort of system, where we'll be able to keep a list of blocks by their "block ID", and add them to our inventory and check on them.

Here's a list of blocks and their block ID's:

https://www.digminecraft.com/lists/item_id_list_edu.php

To understand how this project works, we also learned how an array works. Think of an array as a grocery list that stores items in a numbered list. Every item in the list has a order number, or "index" associated with it. For example, think of the following:

Grocery List:

Bananas

Apples

Mangoes

Granola Bars

It is convenient and organized to store items like this, and doing so lets us make changes to or keep track of many items at once.

We then combine this list with a "for" loop. The name "for" doesn't really explain what the block/loop does, so we'll provide a simpler explanation: a for loop is essentially a "repeat" block where we keep track of which repetition we are on. We store the number of repetitions at any time in the "index" variable of a "for" block.

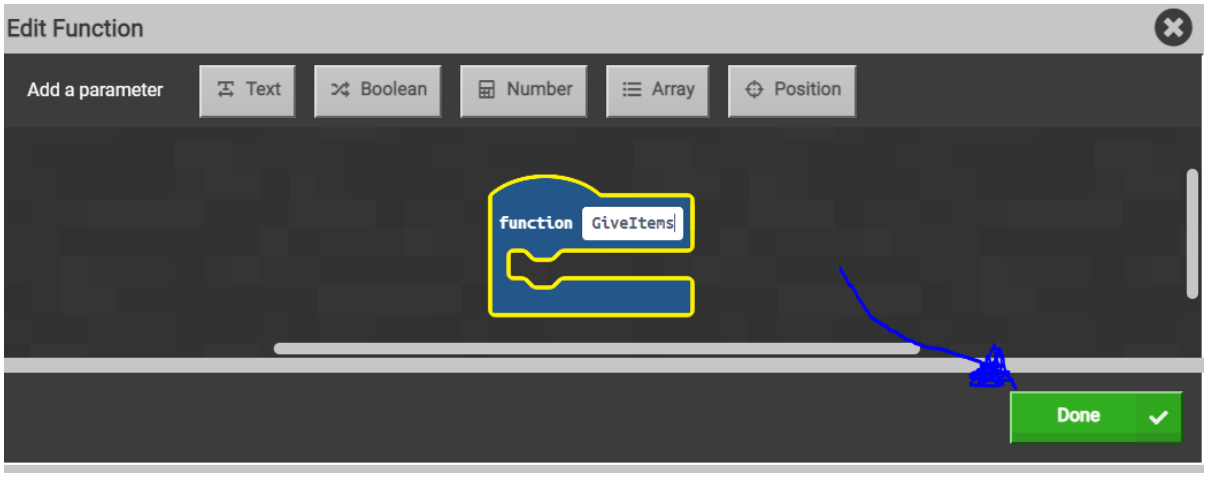
When we combine these two blocks together, we can create a "for" block which acts on the "index" item on the list. For example, on the first repeat of a for block, we might look at Bananas. Then, on the second repeat, we'll look at Apples, and so on. We can use for loops in combination with arrays to store and recall large groups of information easily.

Next, we learned another key concept in coding: functions.

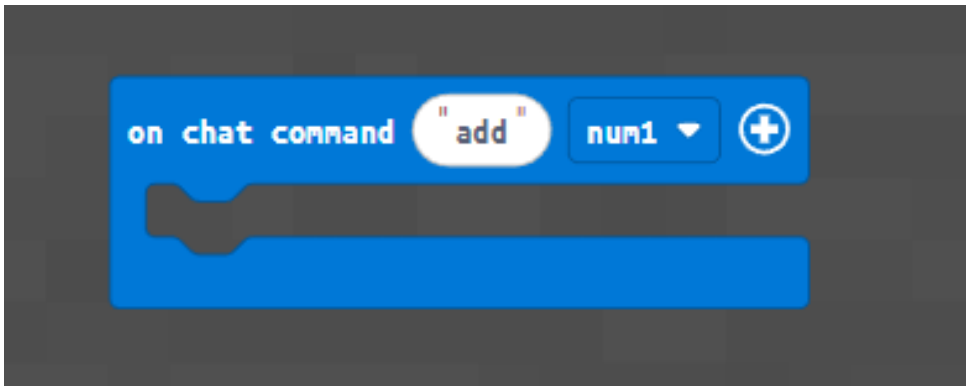
Functions provide a way to group and repeat complex tasks.

If we had code that, for example, moved a player 2 steps forwards, then placed a block, then jumped, then placed another block, that's already 4 blocks of code. If we were to put this code into a function, all we would need to do if we wanted to repeat this code is "call" the function, which would repeat the task.

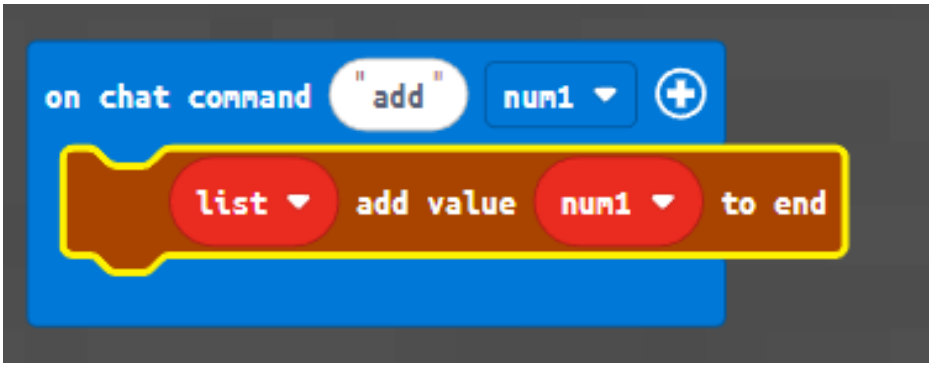
Now that we know this, we can head over to the blue **Functions** tab, also found in the **Advanced** tab, and select "Make a Function". Make sure not to click any plusses or minuses, and create two different functions, using the text boxes to name them "GiveItems" and "CheckItems". As general coding etiquette, we like to capitalize the first letter of all words in a function.



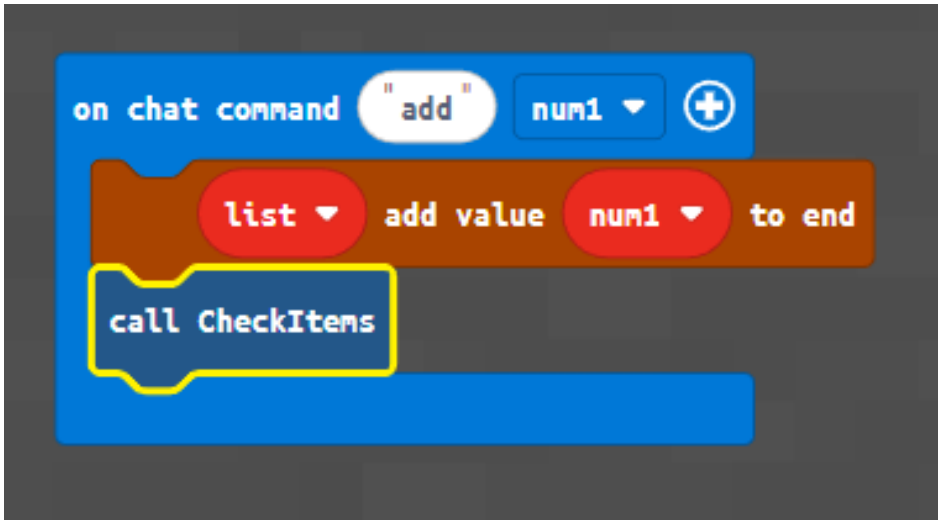
Now, drag in a "on chat command" block from the blue **Player** tab. Click the "+" so that there is a variable for the chat command block to take in after the function named "num1".



Next, go back into the red **Arrays** tab and find the "list add value __ to end" block, and then go to the **Variables** tab and drag in "num1" into the last space in that block. This will add whatever value we choose to the list of blocks. Remember, we're keeping track of the items/blocks in our list by using "block ID's".

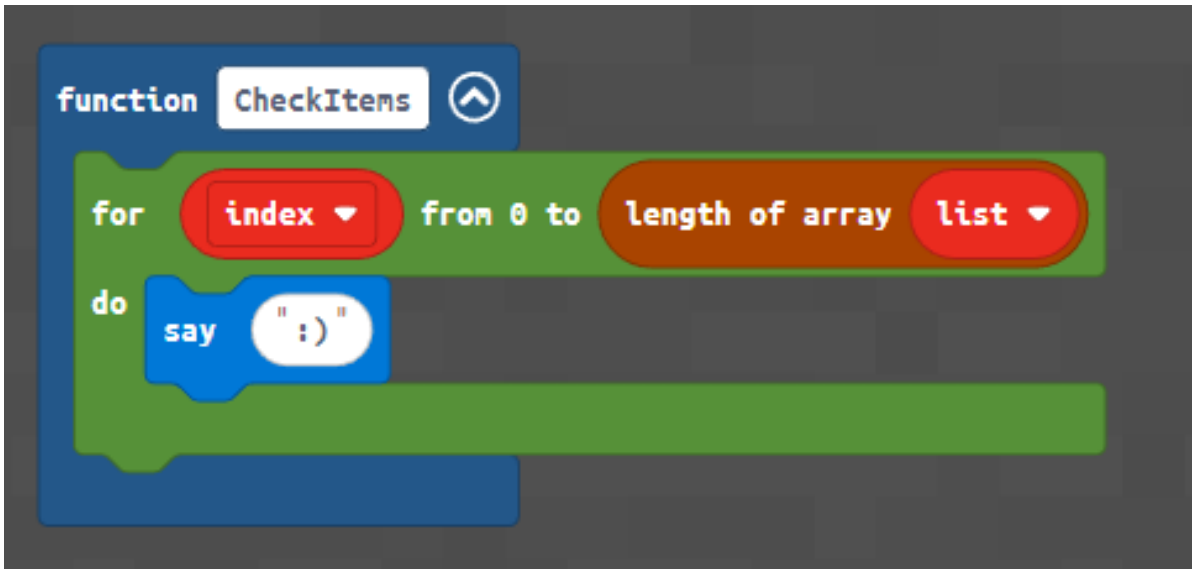


Then, after this, go into the **Functions** tab and drag in the "call CheckItems" block underneath. This will call whatever blocks we have in our CheckItems function.

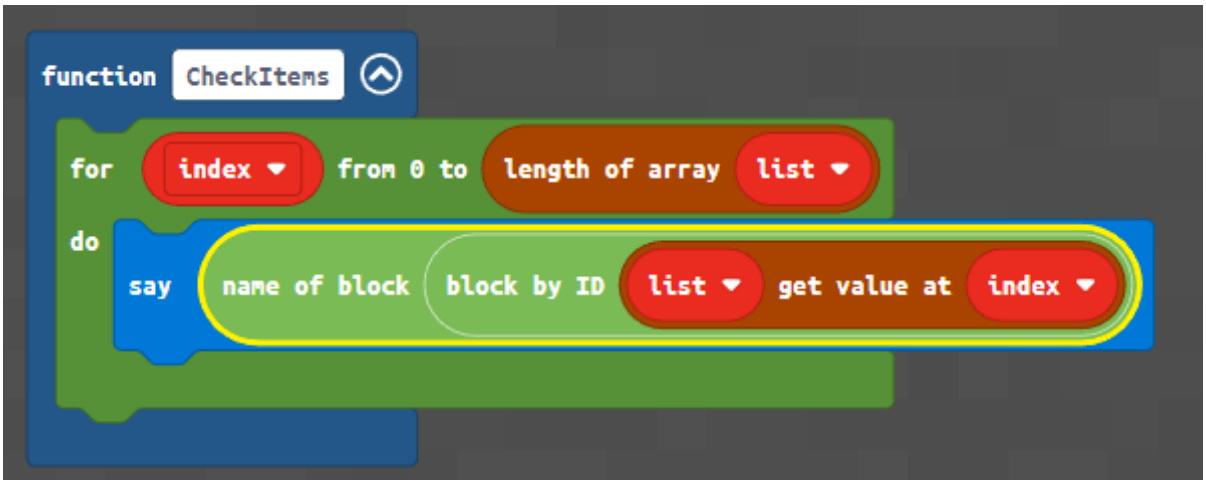


Find the "function CheckItems" block. Next, go into the green **Loops** tab and find the "for index from 0 to _" block. Into the slot after "0 to", drag in the **Array** tab's "length of array list" block. This means that

the loop will run for every item in the list. After this, take a "say: " block from the Player tab and drag it inside.



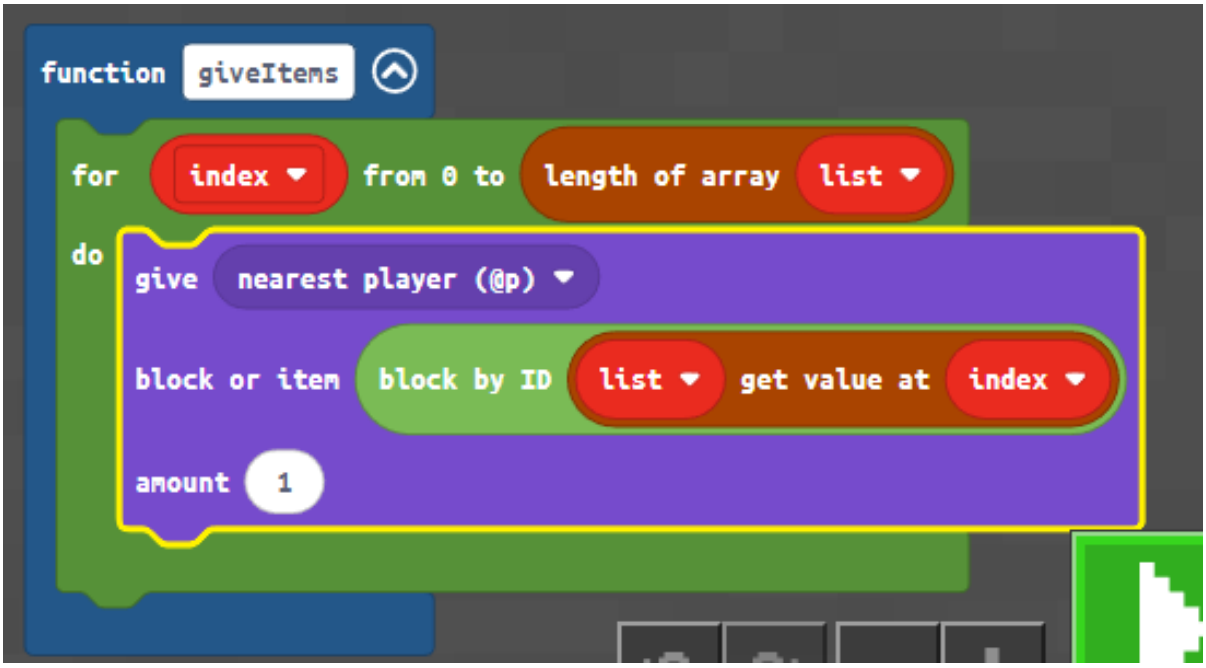
Go to the green **Blocks** tab and near the bottom of the tab, find the "name of block" block and "block by ID" block. Drag the "block by ID" block inside of the "name of block" block. Then, return to the red **Arrays** tab and find the "list get value at __" block, and drag in "index" from the red Variables tab. Drag this inside of the "block by ID" block, and then drag this entire thing into the "say" block.



Now our CheckItems is complete, and after you use the "add" command to add blocks to our list, the CheckItems function will read the items on the list.

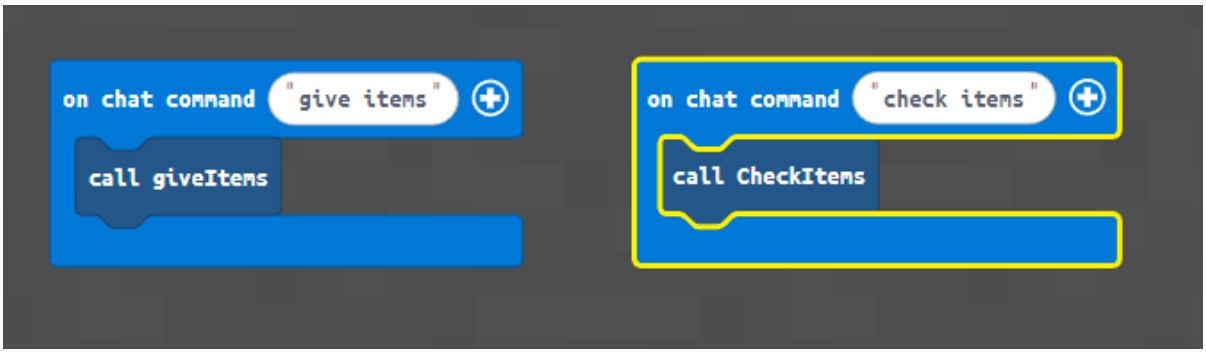
Next, we need to make our GiveItems function. We already have the block, and to begin, simply duplicate the "for" block from the other function and drag it in, but remove all the other blocks inside of the block.

Next, go into the purple **Mobs** tab and drag in the "give nearest player block or item" block. After this, select the "block by ID" block from the other function and duplicate it, then drag it inside of the slot next to "block or item: ". Now we have two working functions, one that checks our list, and one that gives the player the items on the list.



Now all we need is a way to execute these functions: and here we'll see the simplicity that comes with using functions.

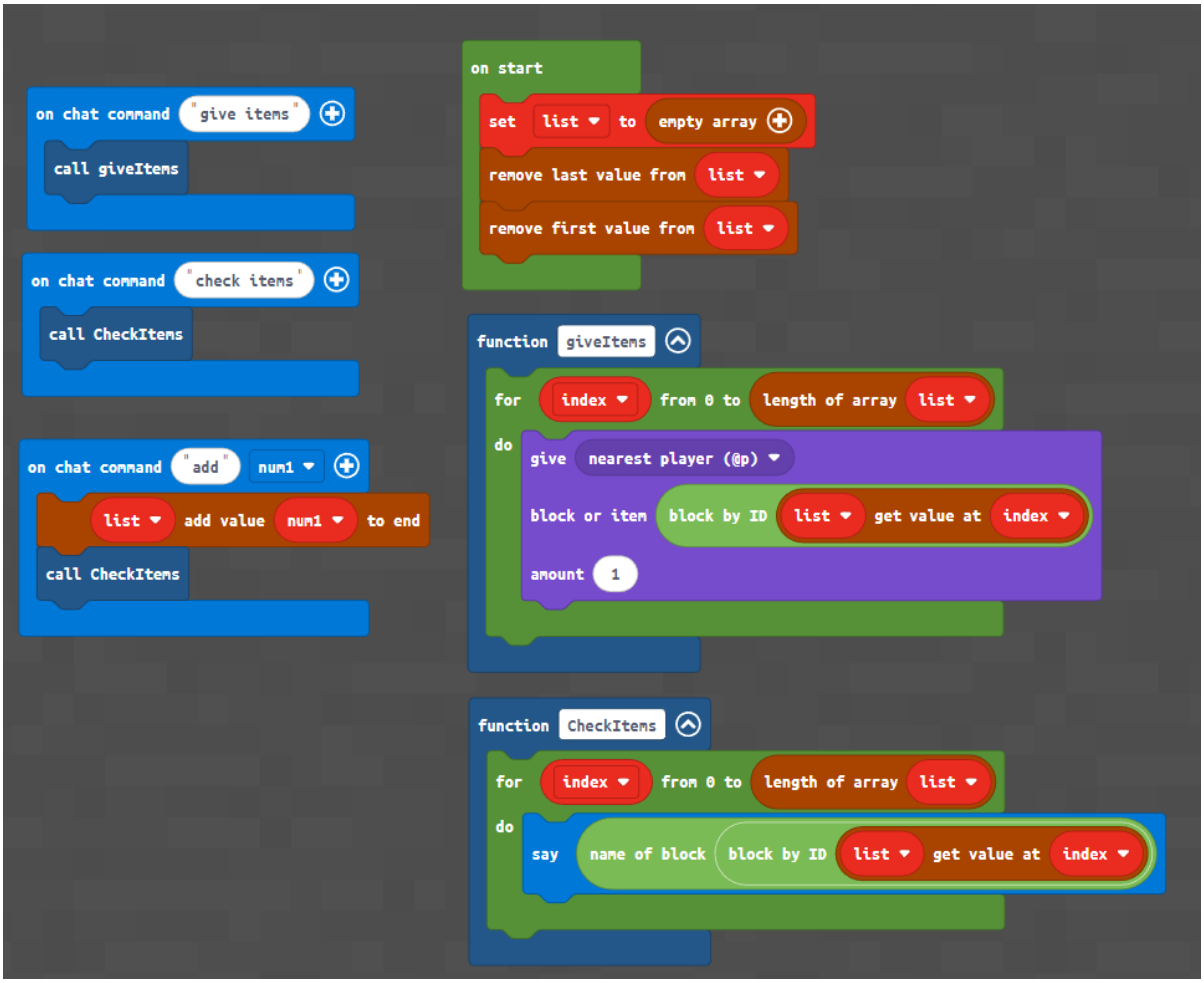
Grab two "on chat command" blocks from the blue **Player** tab, and drag them in, making one's command "check blocks" and the other's "give blocks". Then, go to the **Functions** tab, and drag in the corresponding "call __" block. Its that simple to repeat our functions!



To use the code, simply type "add" and then the block ID of the block you'd like to add to your list. Type "check items" to check the items in the list, and "give items" to give items on the list.

That's it for the code!

Here it is:



Save your project and test the code before continuing.

Continue at your own pace.

Lesson 7

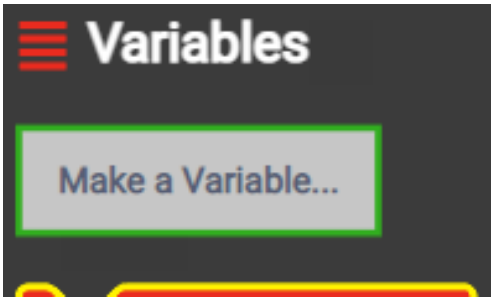
Summit Middle: Coding with Minecraft - Lesson 7 - Review

Lesson overview

In this lesson we created an infinite parkour course, improving our understanding of "random" blocks, functions, and the duplicate function.

To begin, we created a new project, and ensured that we were on a "superflat" world.

Next, we went over to the red Variables tab and created 3 variables: floor height, current height, and score.



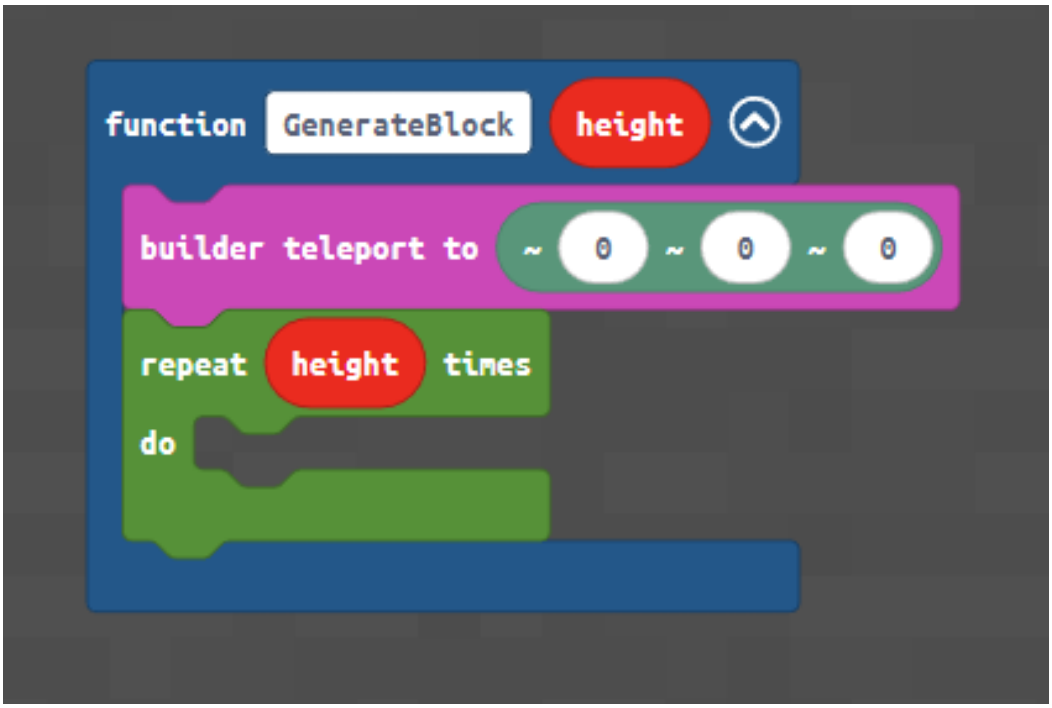
We then made our initiation code. We first made sure we had a "on chat command" block, and then made the command "start" and clicked the plus icon so that we had a variable after our text. Then, we selected this variable, selected "Rename Variable...", and renamed it to "max height". This will allow the player to input a length for the parkour course.

Next, we added some general blocks for start: from the Gameplay tab, the "change game mode to survival" and "set difficulty to peaceful" blocks. This ensures that our player won't run out of stamina, but also can't just fly through the course.

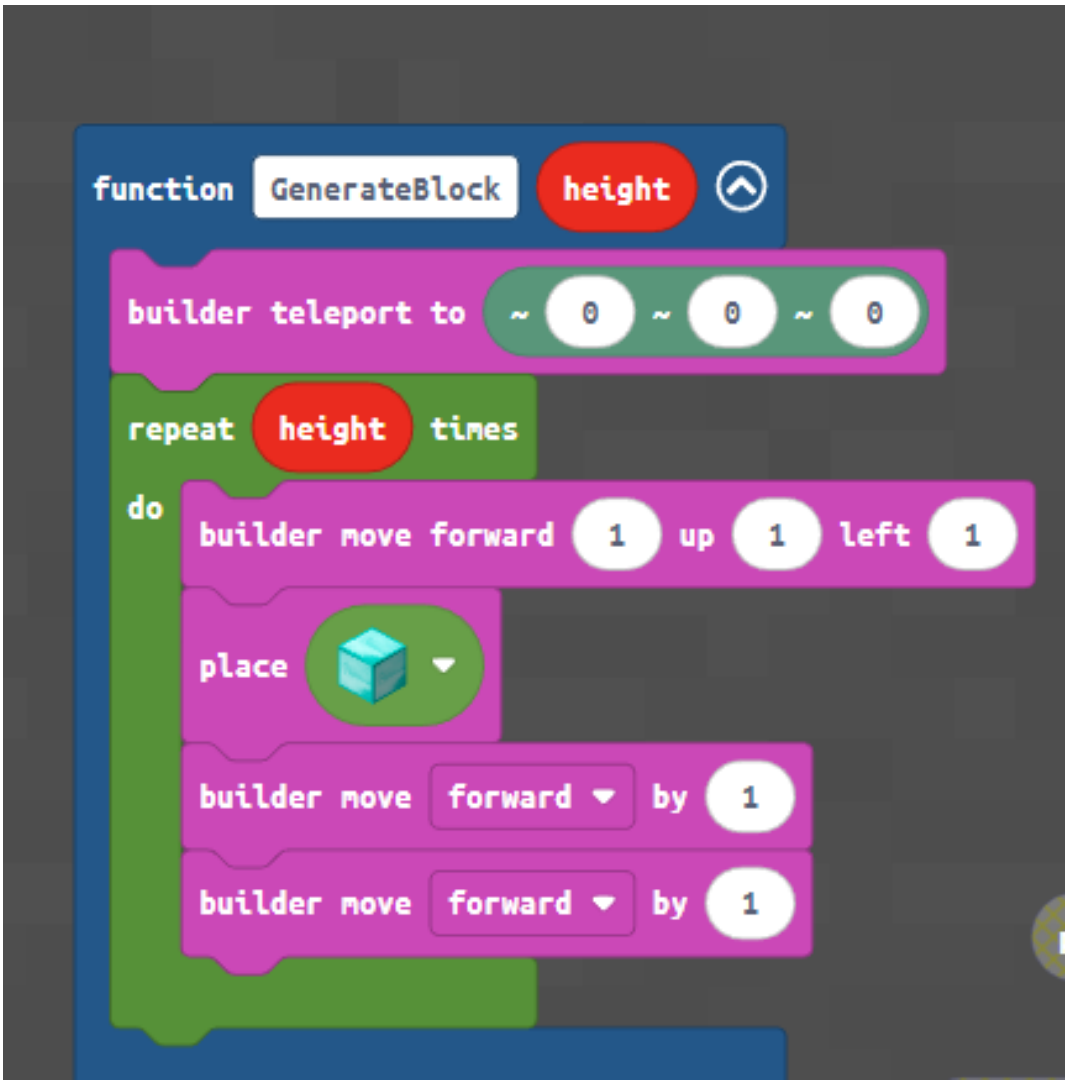
After this, we went into the blue Functions tab, found by selecting "Advanced", and selected "Make a Function". We named this function "GenerateBlock", and then selected "number" at the top, naming the new variable "height".

What we do here is give a variable, "height", for the function to "take in". This essentially means that whenever we want to "call" or run the Generate Block function, we also need to give it a number, in this case, the height that the course generated will be.

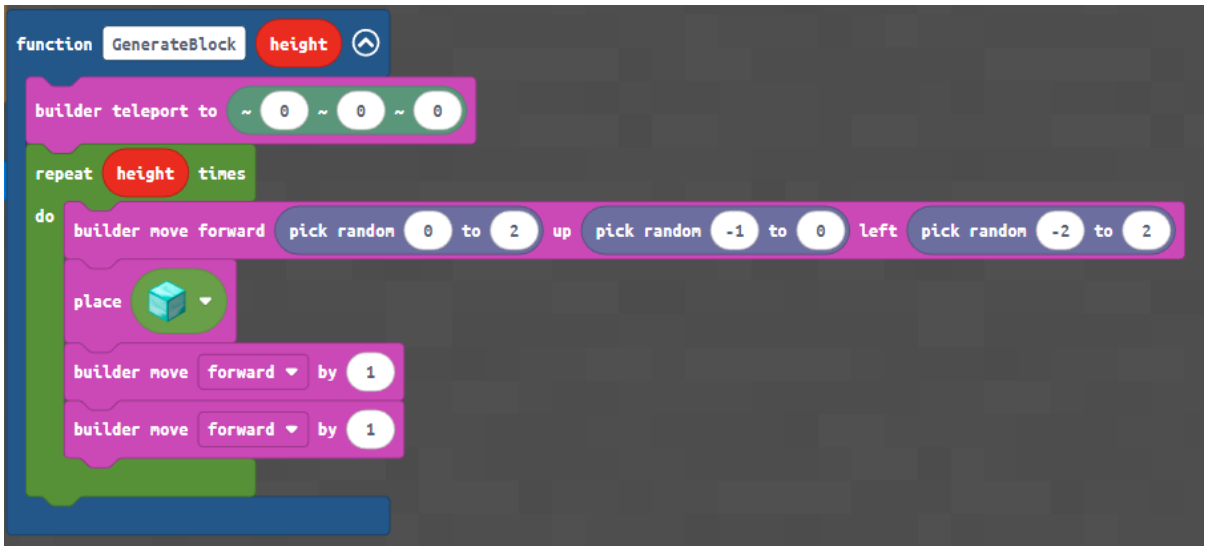
Once the function is created, go over to the pink Builder tab, also found in the Advanced tabs. Drag in the "builder teleport to ~0 ~0 ~0" block. Next, go into the green Loops tab, and drag in the "repeat __ times" block. This next step is very important: select and duplicate the "height" variable from the function, and drag it into the repeat block.



After this is done, go back into the Builder tab. Drag in the "builder move forward _ up _ left _" block, the "place _" block, and two of the "builder move forward by 1" blocks.



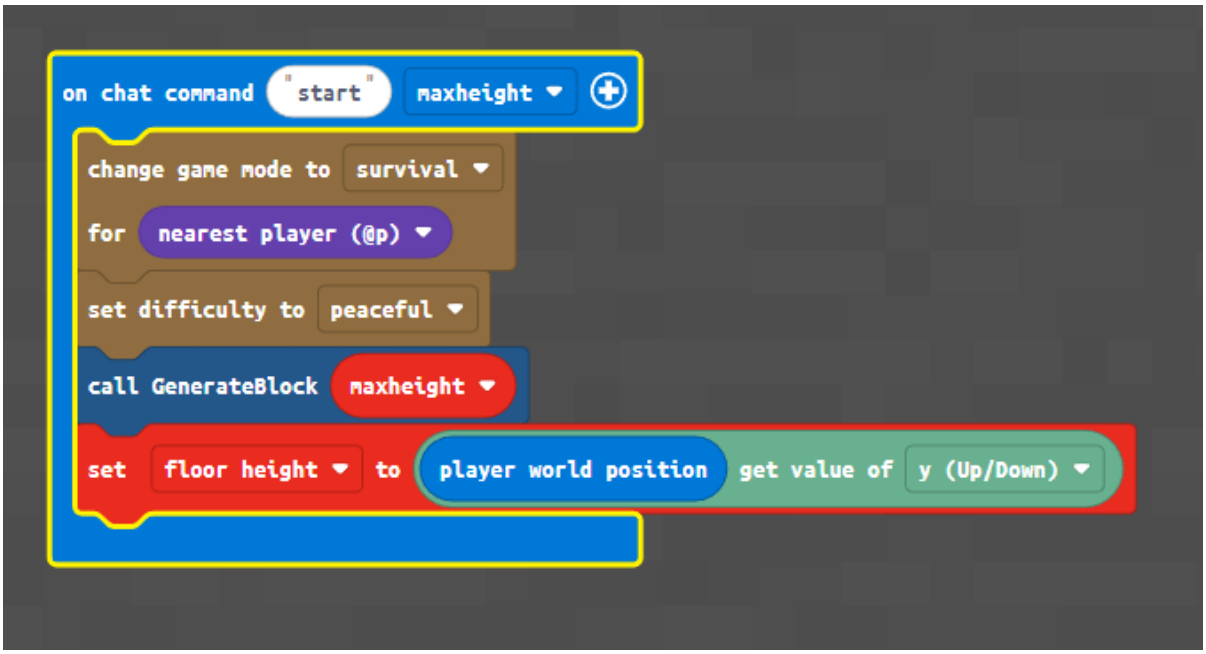
Then, go over to the Math tab, and find the "pick random _ to _" block. Drag it into all 3 slots of the "builder move forward" block. This will ensure that the builder has some side-to-side randomness so that the parkour course isn't so linear. Then, ensure the values read "0 to 2", "-1 to 0", and "-2 to 2".



After this, select whatever block you want the course to be from the "place __" block, and then make sure that one of the "builder move __ by 1" blocks at the bottom is "forward", and one is "up". The final step for our generation is to go into the Functions tab again, and drag the "call GenerateBlock" block to the bottom of our "start" code. Then, drag in "max height" from the variables tab into the slot.

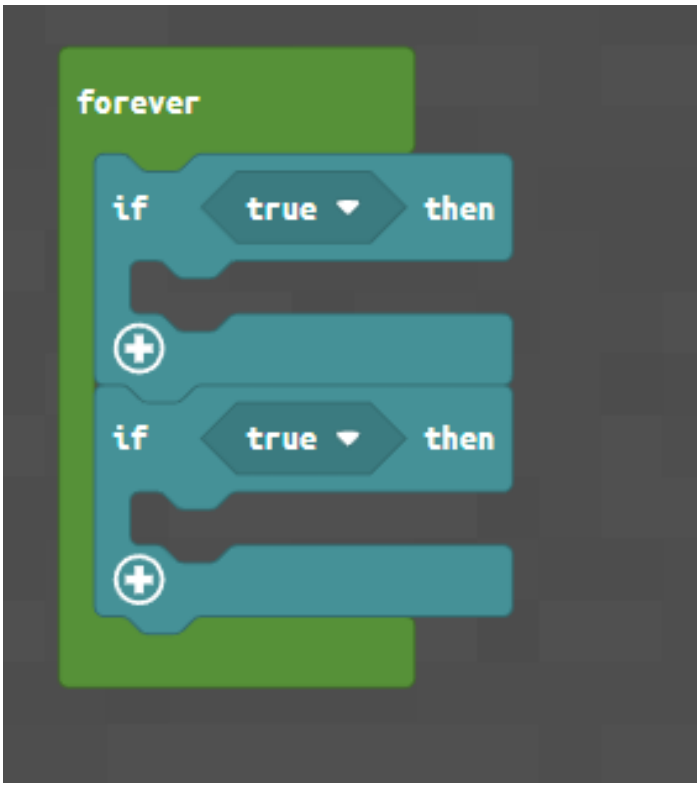
In our parkour game we will need to keep track of the player's height in order to give them a score and also to detect when they have fallen. To do this, we use a "floor height" variable to track the initial height, a "current height" variable to track the player's current height, and a "score variable" that will determine the difference between the player's current height and the initial height. Note that the "current height" variable only updates when the player gets higher up, not lower, and we can use this to detect when the player has fallen.

To begin this functionality, find the "set __ to __" from the Variables tab. Then, go into the Positions tab, and drag in the "position get value of x" block close to the bottom of the tab into the "set" block. Then, go into the Player tab, and drag in the "player world position" block in place of the "position" variable. Finally, select the second part of the block and make it say "y (Up/Down)" This combination of two blocks is one that we duplicate many times for our code, as it gathers the player's height coordinate.

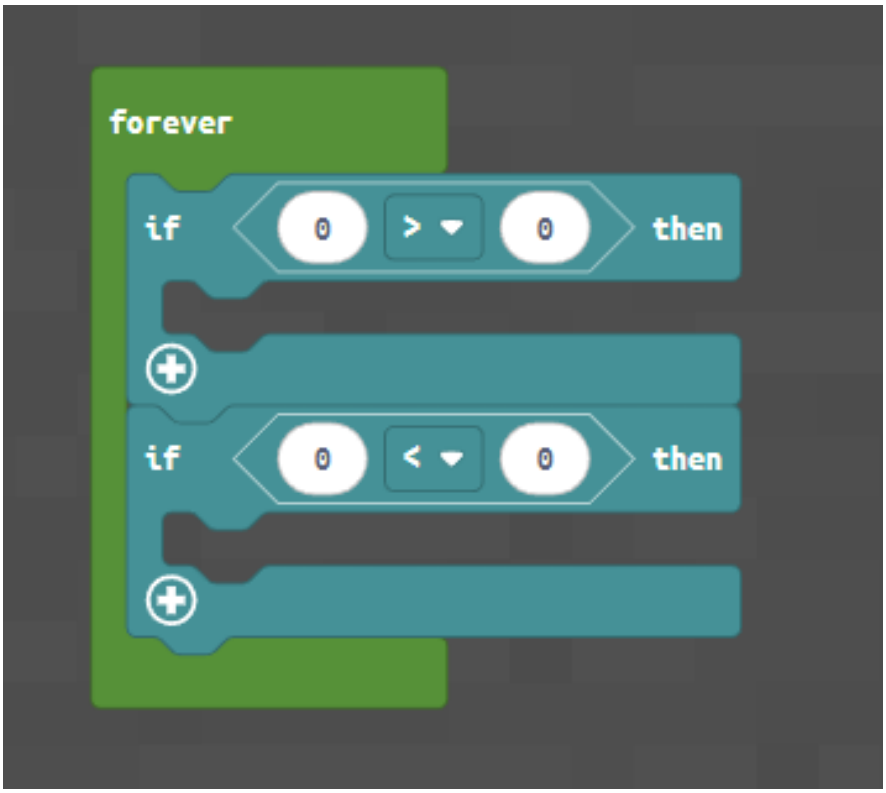


Next, make that "set" block read "set floor height to __", and make the second "set" block read "set current height to floor height".

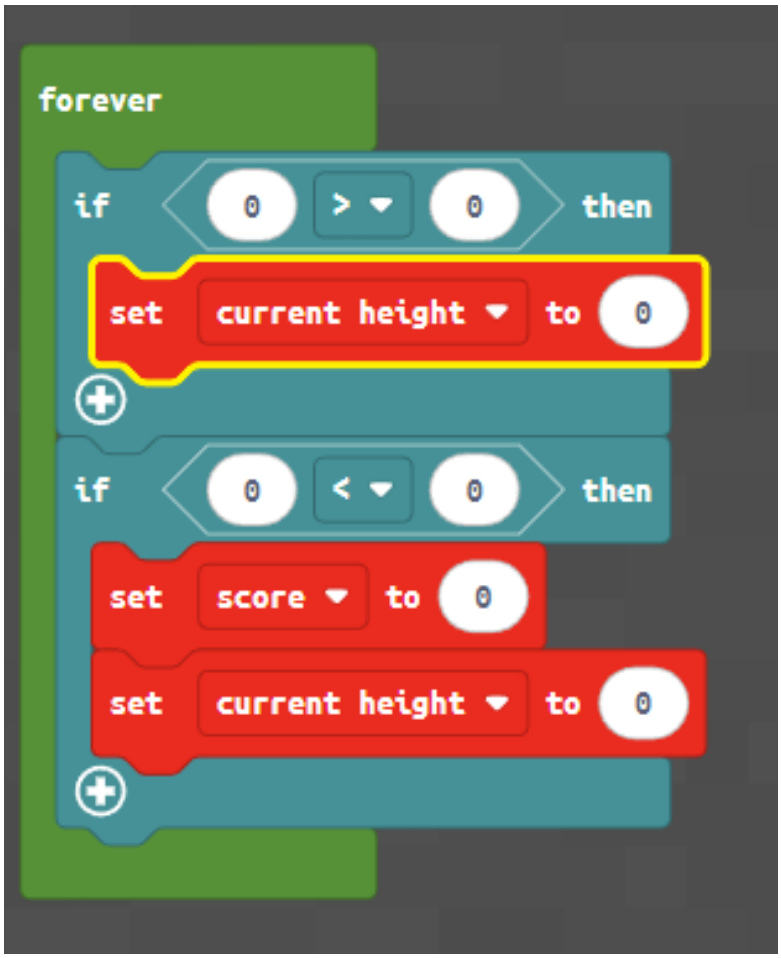
Then, we'll do our height-checking code. Go into the Loops tab and find the "forever" block. Then, inside, go into the Logic tab and drag in two "if __ then" blocks.



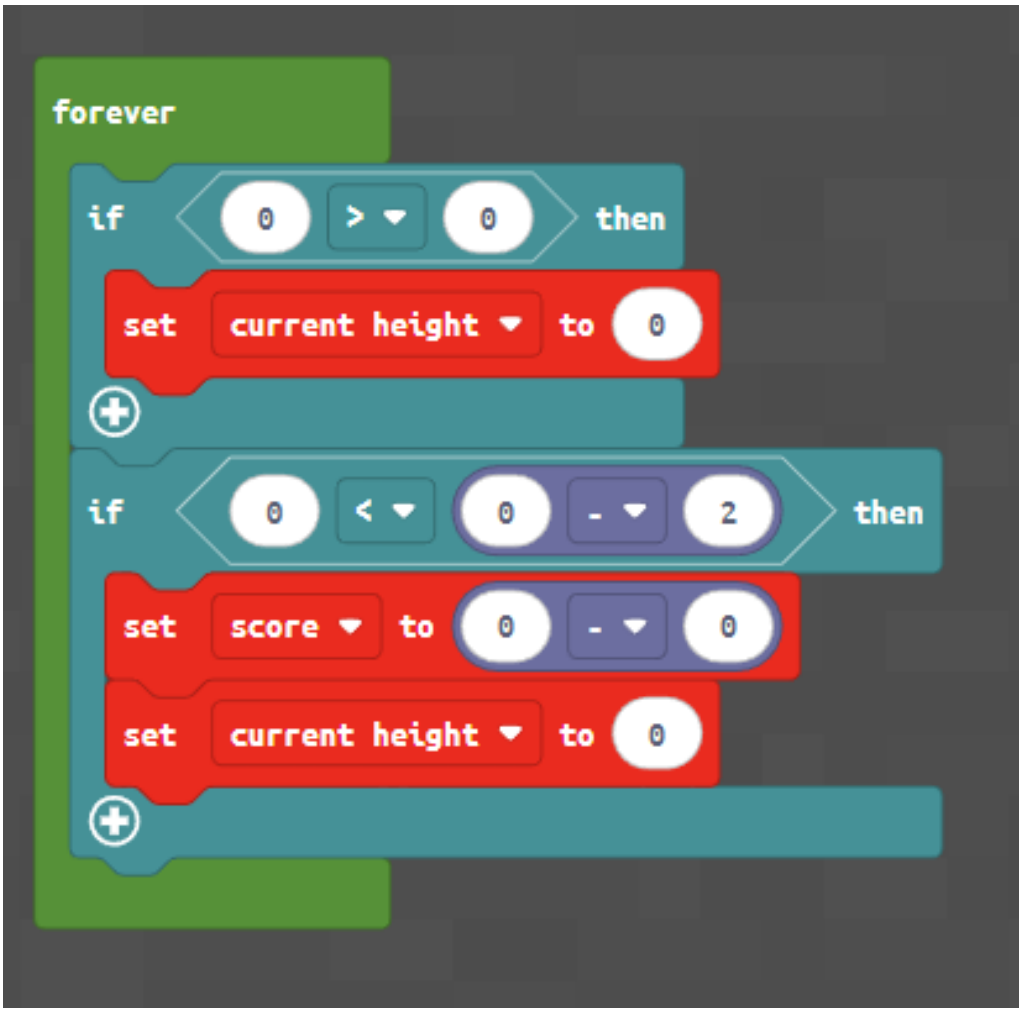
Into both of these blocks, drag in the "__ > __" operator from the Logic tab. Make sure that you change the sign so that both blocks are different, and they look like this:



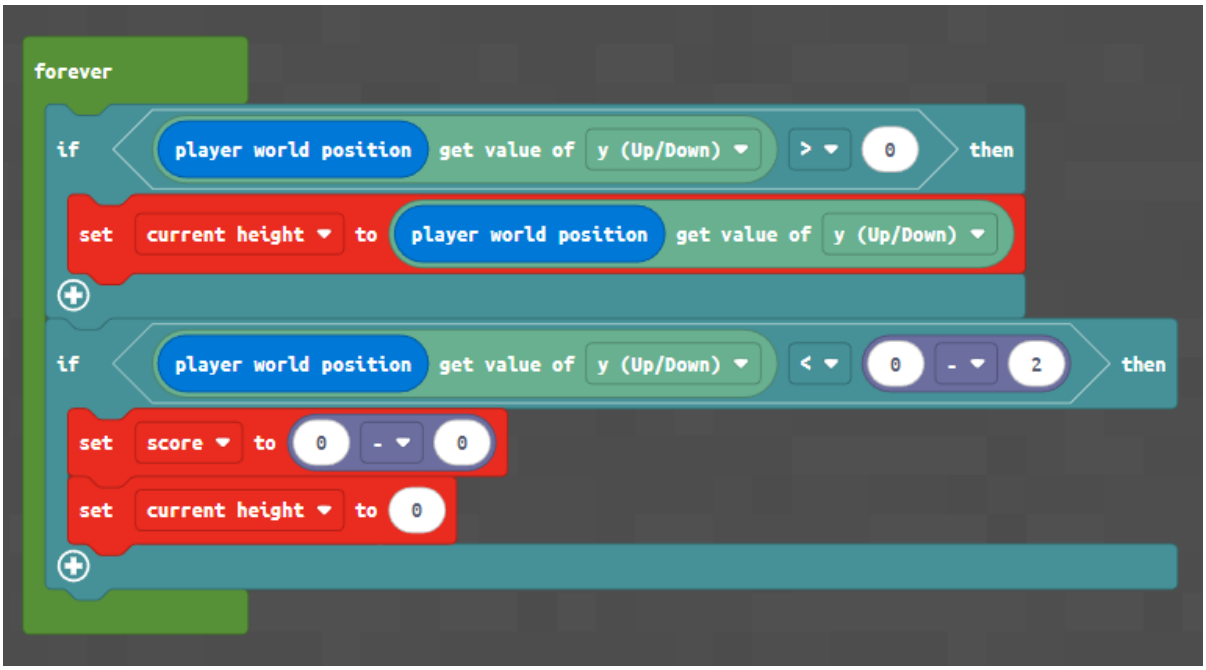
Then, drag in three "set" blocks from the Variables tab into these "if" blocks: one in the first, and two in the second.



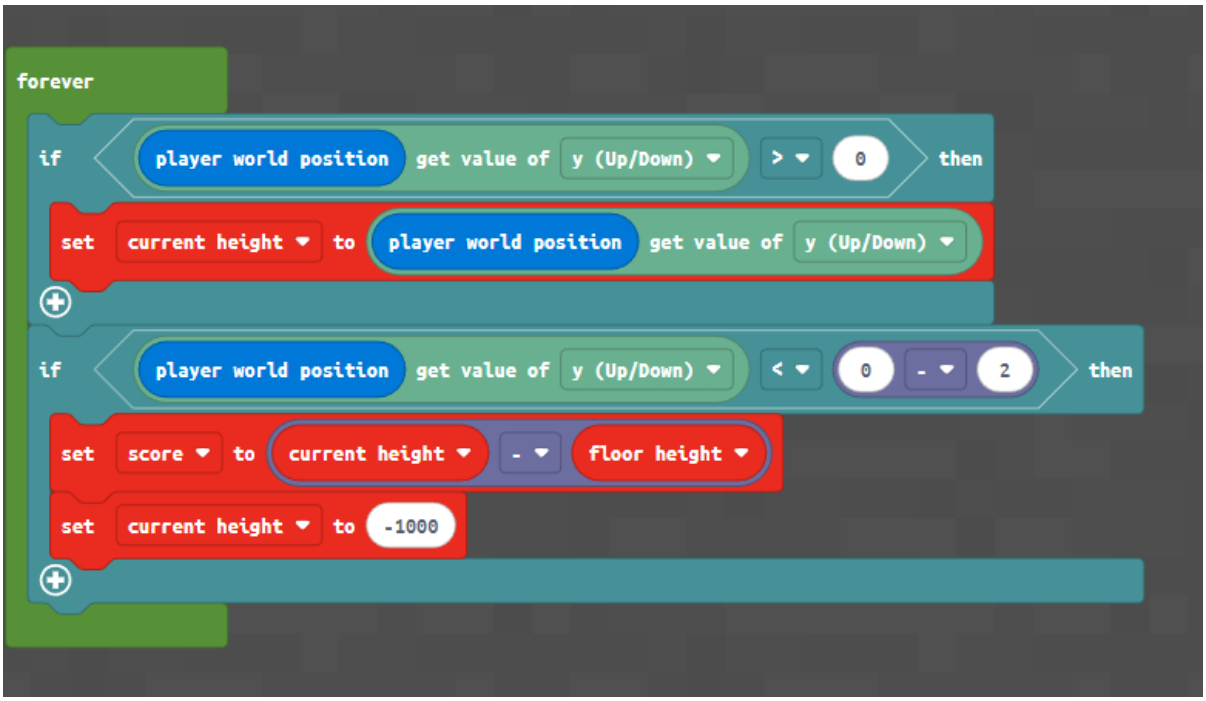
Take the "__ - __" operator from the Math tab, and drag it into the second space in the second "if" block, and then also into the first "set" block directly underneath.



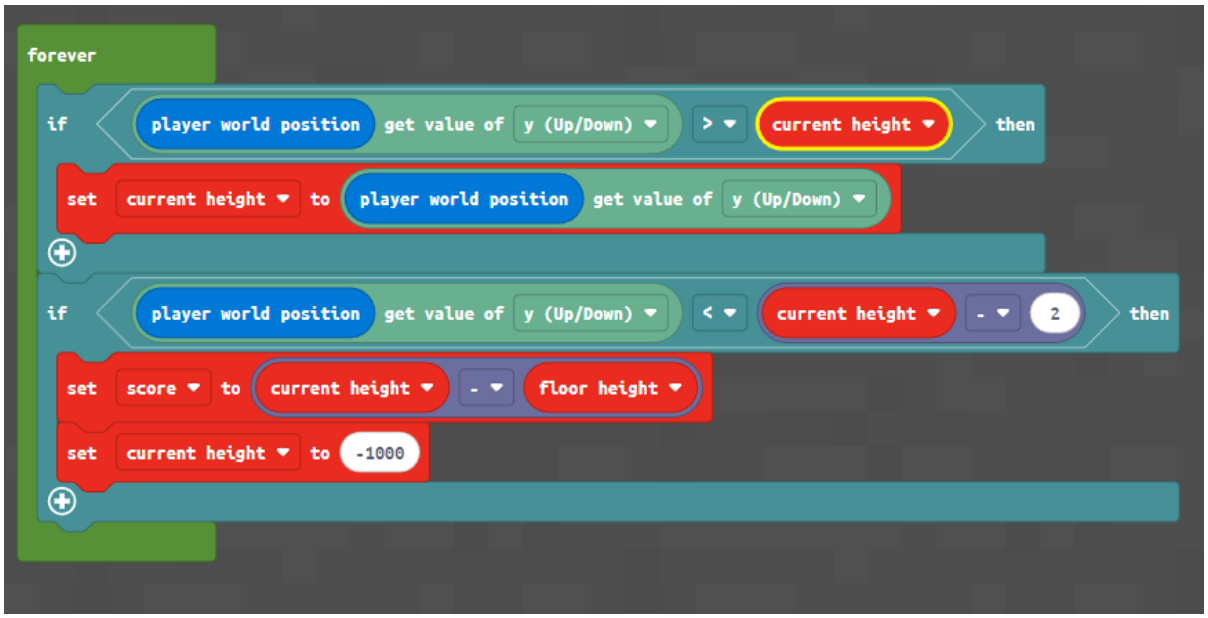
Then, duplicate the "player world position get value of..." block 3 times. First, drag one into the first slot of the first "if" block. Then, drag another into the "set" block underneath, and finally, drag one into the first slot of the second "if" block.



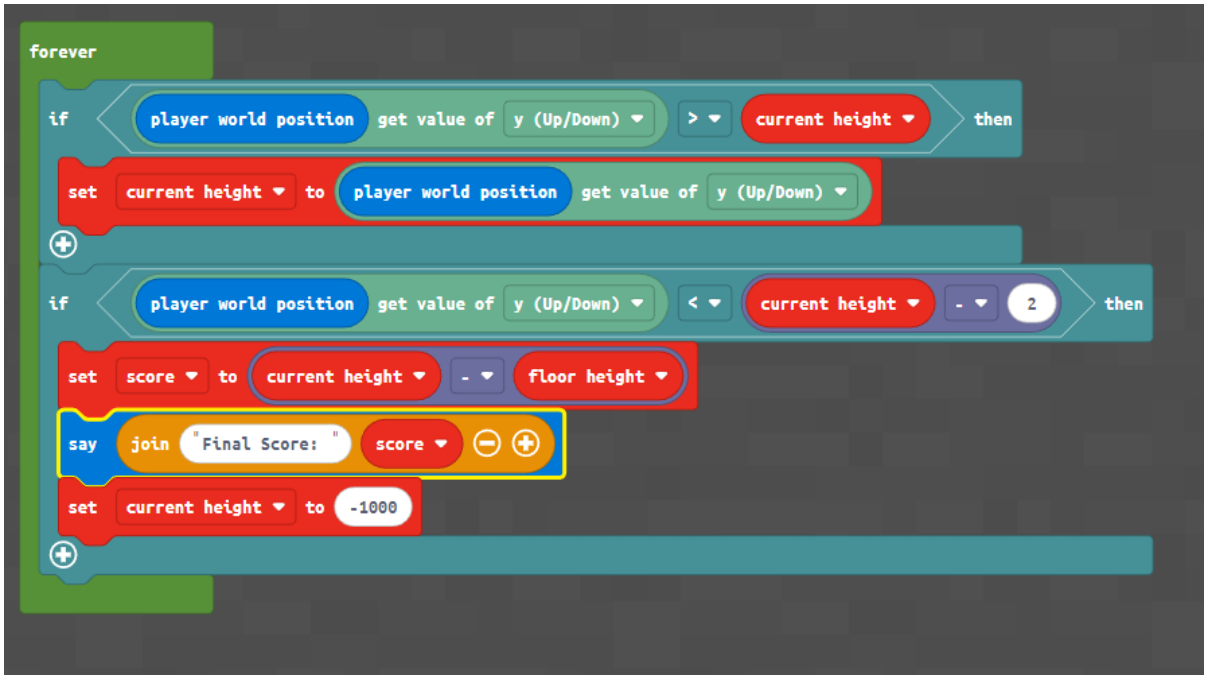
Finally, drag in the "current height" and "floor height" variables from the Variables tab into the first "set" block in the second "if" block. Make that "set" block read "set score to current height - floor height". Make the other "set" block underneath read "set current height to -1000", and finally, make sure the top "set" block from the first "if" block is setting the "current height" variable.



Also, drag in the "current height" variable into the second slot of the first "if" block. After this, drag in the "current height" variable into the first part of the " - " block, and make sure the second part reads "2".



Now, for the last step, we just need to print out the score when the player loses. Find the "say" block from the Player tab, and then afterwards, go to the Text (Advanced) tab. Get the "join" block and drag it inside, and then drag in the "score" variable into the second space. Finally, edit the first space so that it says "Final Score: ".



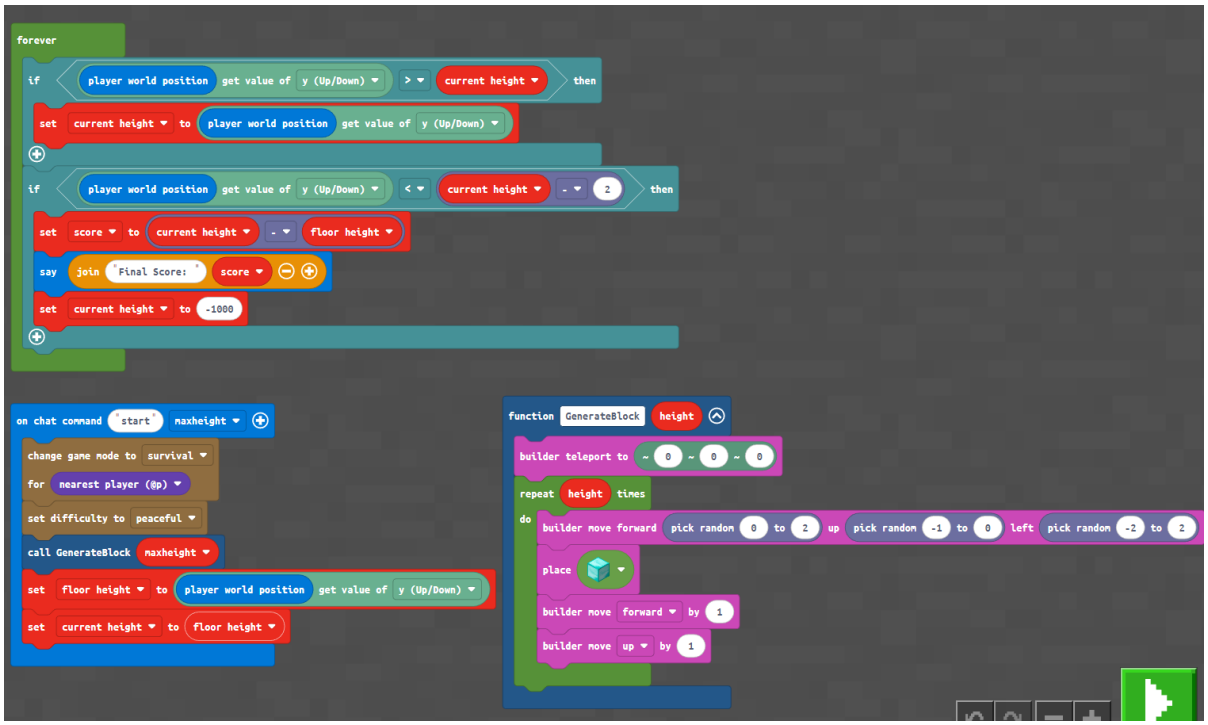
Altogether, the function works like this:

The first "if" block checks if the player has gone higher than the currently recorded highest point, and if it has, the new highest point is recorded with our "current height" variable.

Then, the second "if" block checks if the player has fallen more than 2 blocks below their highest point, and if they have, then the code has considered the player to have fallen, where it calculates their score and prints it out.

Note that every time a player begins the parkour course, they need to use the "start" chat command followed by the length of the course if they want the score to reset. This will generate a parkour course for the player.

Here's the code:



Save your project and test the code before continuing.

Continue at your own pace.

Lesson 8

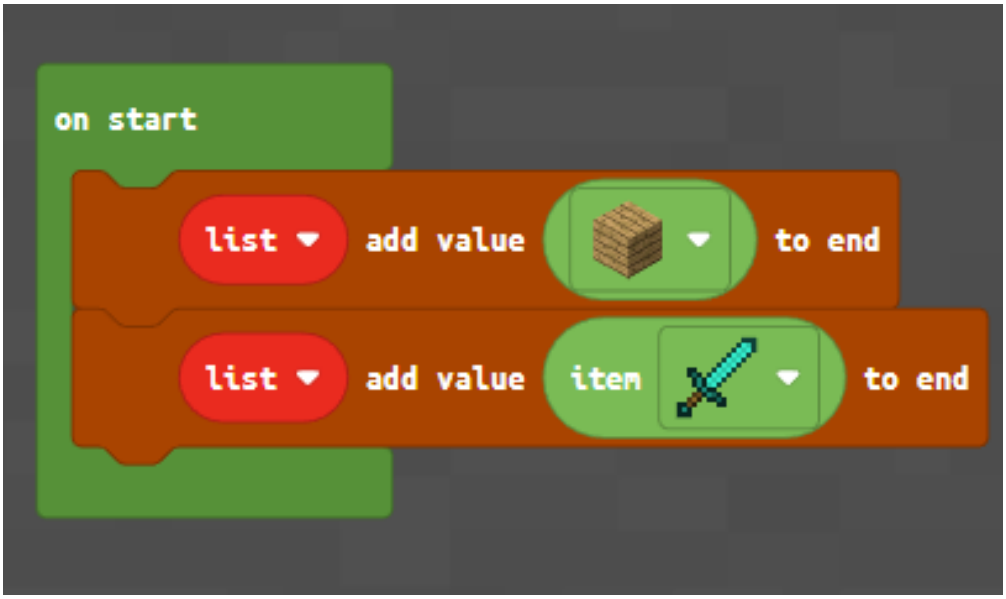
Summit Middle: Coding with Minecraft - Lesson 8 - Review

This final lesson adds Lucky Blocks to an earlier project: the wave-based game.

Lucky Blocks are a Minecraft add-on that many players know about - they're blocks that the player can destroy that can give them one of many different items. For our purposes, we will give the player their gear for the wave games through lucky blocks, which will be supplied at the beginning of the game and in-between rounds.

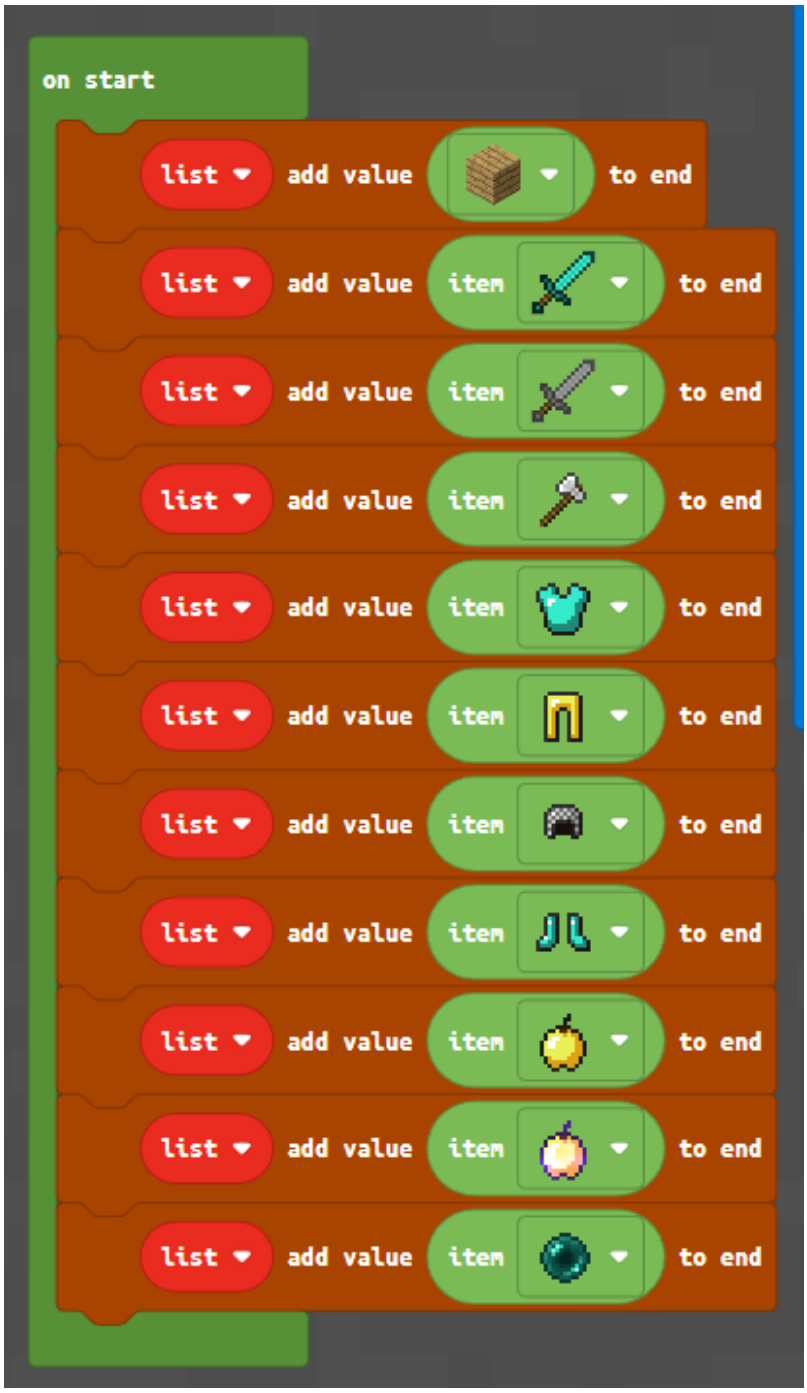
To accomplish this, we used arrays/lists to store all possible items, and then the "pick random" blocks to pick a random index from our list to give the player. We also learned how to use the Agent in order to give the player blocks.

To begin, drag in the "on start" block from the Loops tab. Then, find the **Arrays** tab (under Advanced) and drag in the "list add value __ to end". Then, go to the blocks tab, and drag in either an "item" or "block" block into the open space.



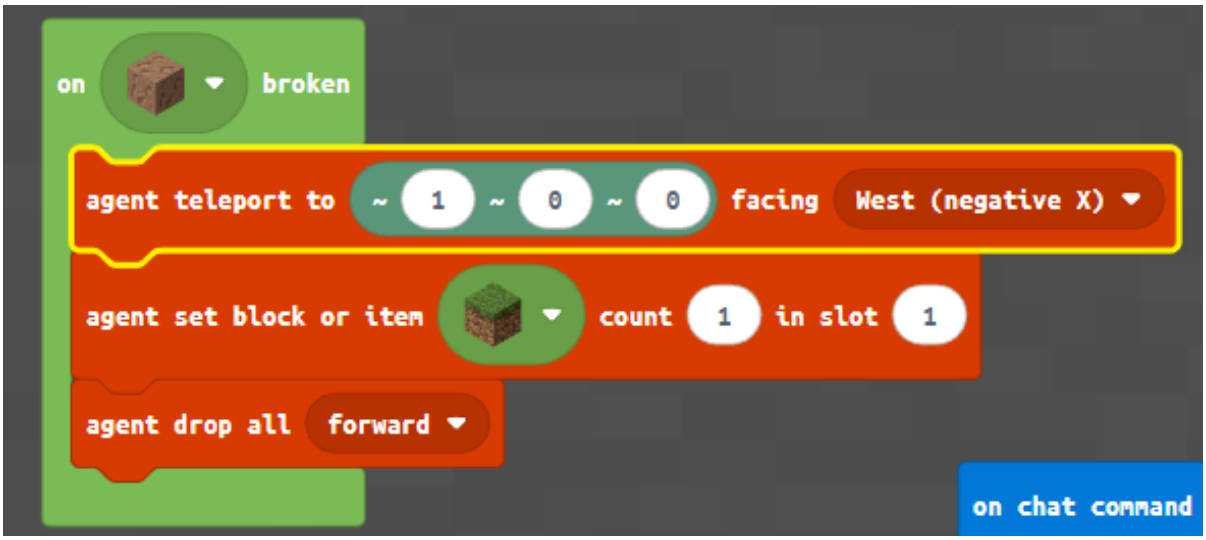
With this chunk of code, we're going to assign all of the items that will be in the player's starting inventory. Right click and select "duplicate" to create multiples of this block, where you're encouraged to add a variety of different loot that the player could gain during their battle.

(You DO NOT need to copy the following code, this is just an example of what students could add):

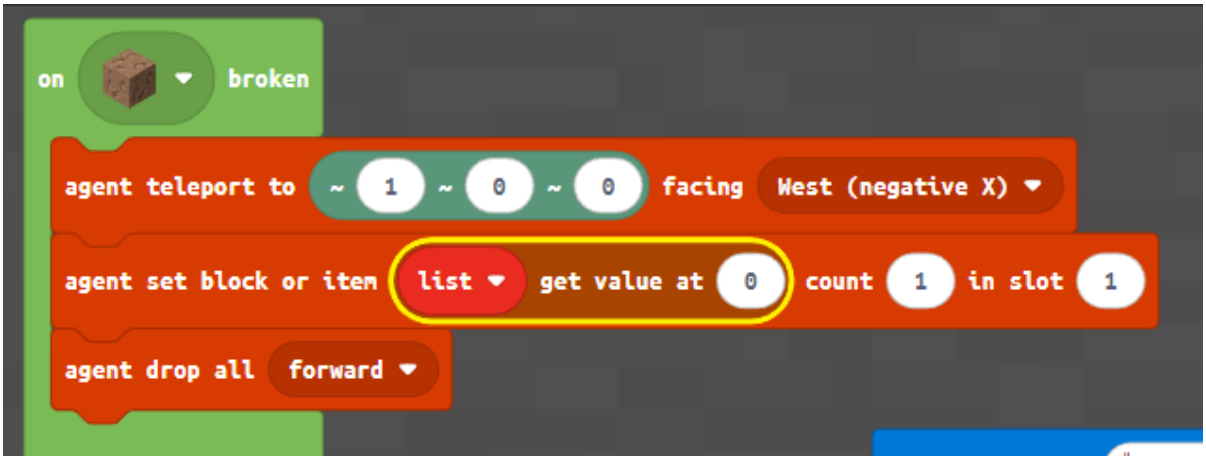


After this, find the "on (block) broken" block from the **Blocks** tab, and set the block to be "mushroom". Make sure that you select the FIRST mushroom block, as the code won't work if you select different blocks in the later sections of code.

Next, enter the **Agent** tab, and drag in three blocks: "agent teleport to ___ facing __", "agent set block or item __ count _ in slot _", and "agent drop all ___".

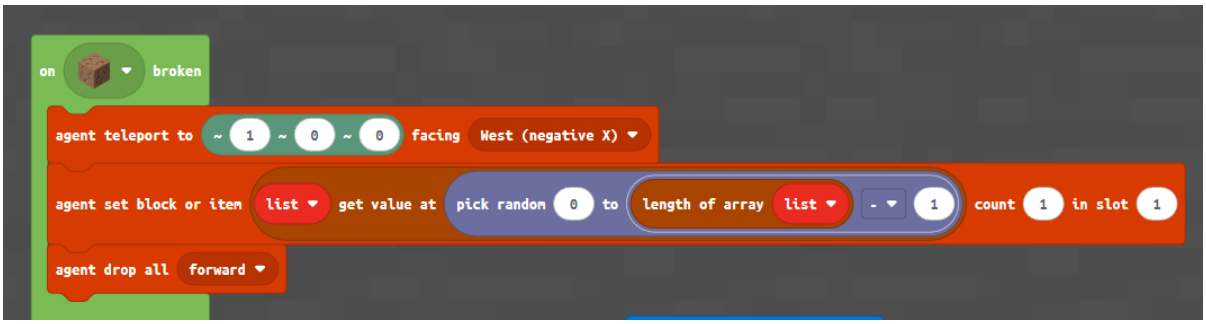


We'll want to modify the first block so that it reads "~ 1 ~ 0 ~ 0 facing West (negative X)". Then, the third block should read "agent drop all forward". Finally, we'll make go into the **Arrays** tab again, and drag in the "list get value at __" block.



After this, go into the **Math** tab, and after "get value at" drag in a "pick random 0 to __" block, and then in the second blank after the 0, drag in a "(__) - (__)" block also from the **Math** tab. Then, drag in the

"length of array list" from the Arrays tab into the first slot of the subtraction block, and set the second number to be 1.



The reason that we pick a random index from the list with a number from 0 - (length - 1), is that arrays start with a 0th item.

Looking back on our shopping list example from previous lessons, while a regular shopping list would look like this:

Carrots

Beans

Apples

etc.

An array shopping list would look more like this:

0. Carrots

Beans

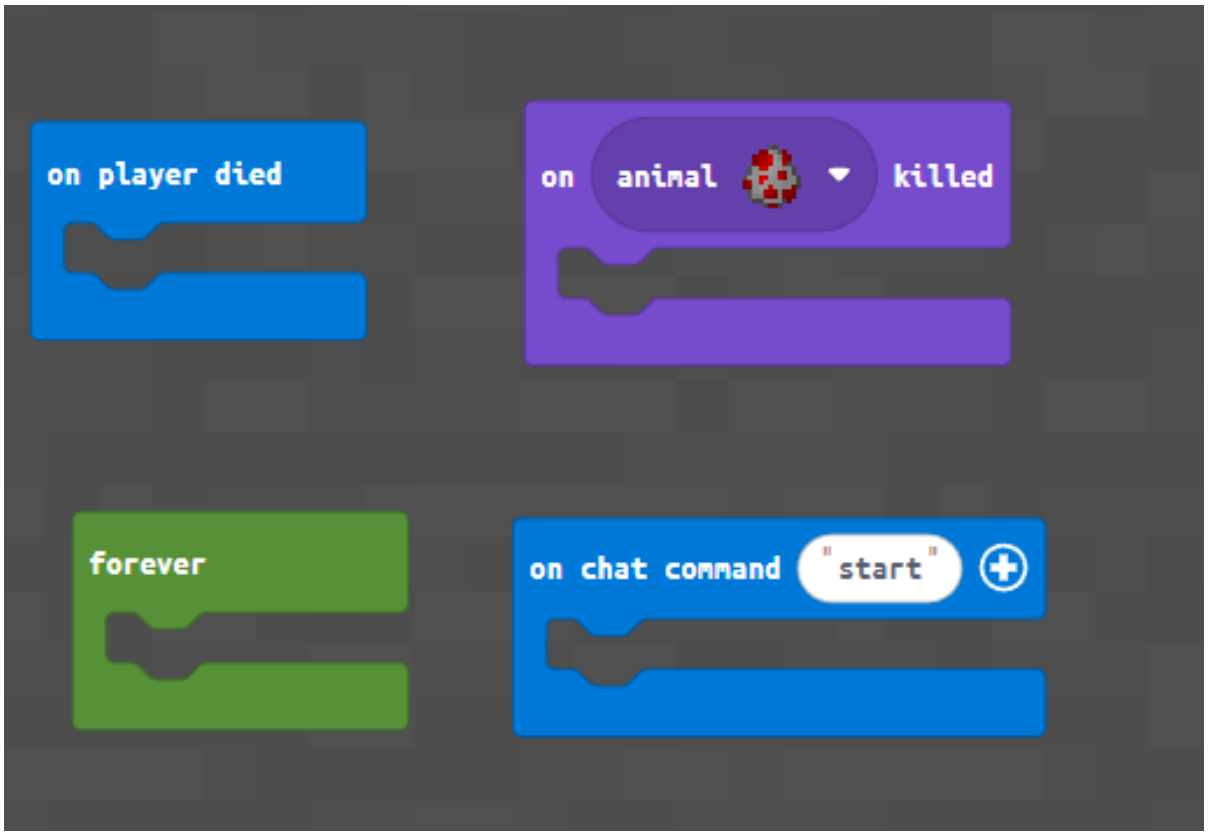
Apples

etc.

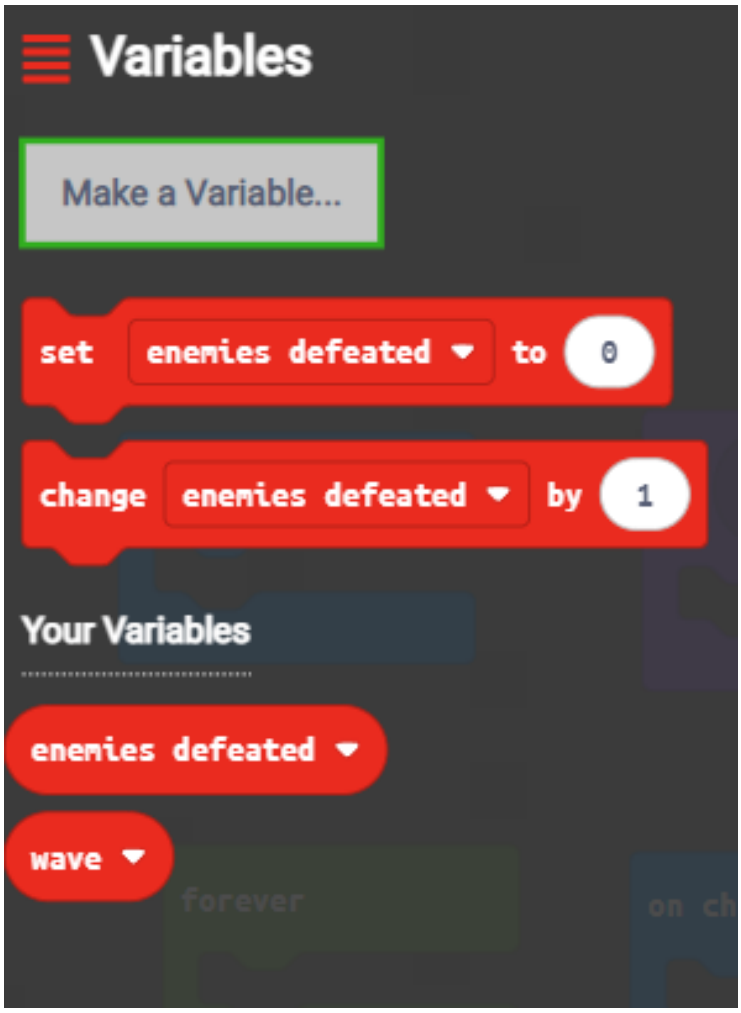
So choosing a random item from that list means we have to pick a list number between 0 and 3 (the total length minus one), not 1 and 4.

After this, students simply had to follow same instructions as lesson 2 for the original wave game:

First, we dragged in four key blocks: the "on animal killed" from the **Mobs** tab, the forever block from the **Loops** tab, the "on chat command" block from the **Player** tab, and the "on player died" block also from the **Player** tab.



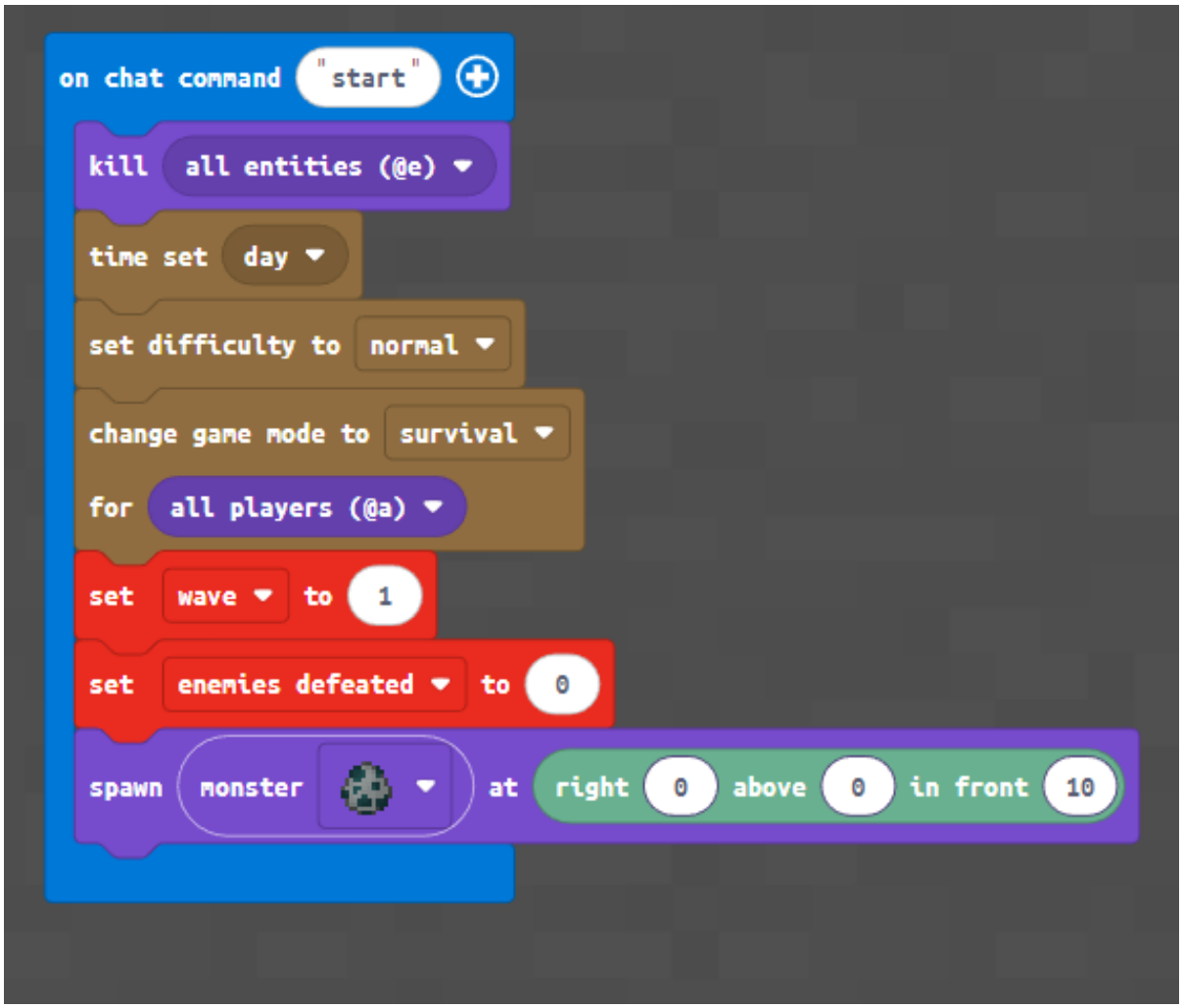
Following this, we need to create two variables. Basically, variables are blocks that we get to create, and they can represent any number. In this case, we'll create two variables: one to keep track of the current wave number, called "wave", and one to keep track of enemies defeated, called "enemies defeated". To create these, go over to the **Variables** tab and select "Make a Variable".



To begin, we'll work on the "on chat command" block. Set the command to be "start", and then begin by opening the **Mobs** tab. Drag in "kill 'nearest player'", and change the box to say "all entities instead". Then, open the **Gameplay** tab, and drag in 3 blocks: "time set __", setting it to be day, "set difficulty to", setting it to be normal, and "change game mode to ___ for __", setting it to be survival for all players @a.

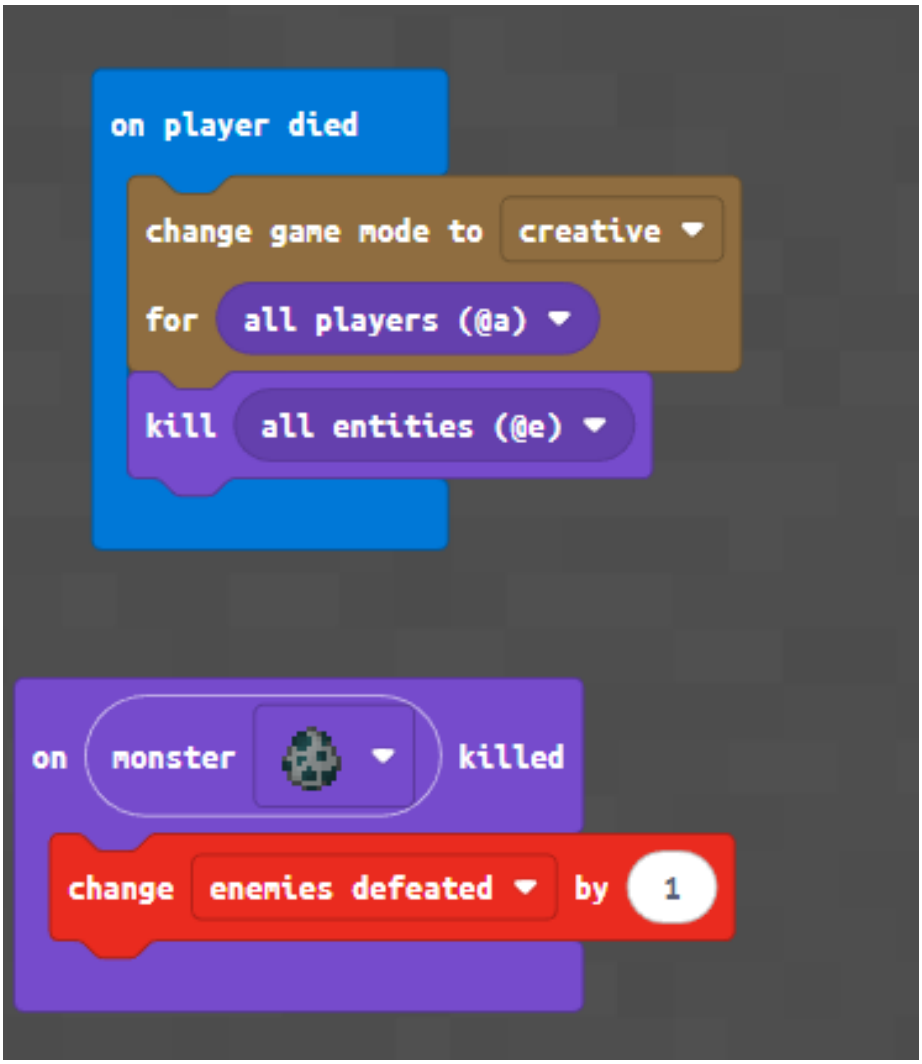
After this, we go over to the **Variables** tab and drag in two blocks that say "set ___". Change the first block so it reads "set wave to 1", and the second block so it reads "set enemies defeated to 0"

Finally, we want to go over to the **Mobs** tab, and drag in the "spawn ____ at _____" block. Then, from the same tab, drag in "monster __" to both this block and the "on animal killed" block from earlier. Finally, to both blocks, go to the Positions tab, and drag in the "right 0 above 0 in front 0" block, setting "in front" to be 10.

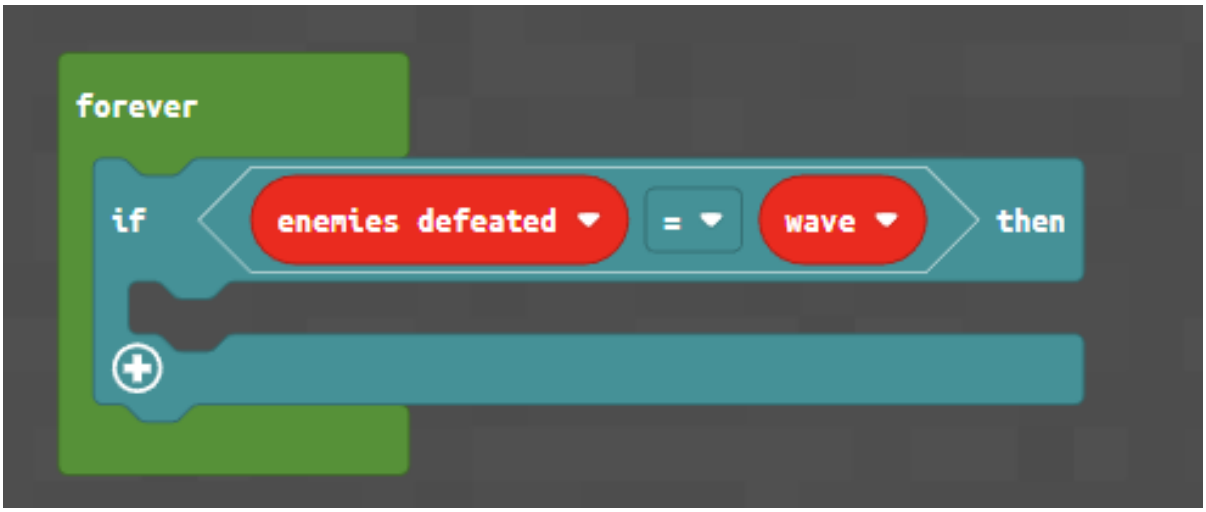


Finally, and most importantly, make sure that the enemy spawn 'egg' that is selected is the 'vindicator' enemy. These steps altogether will ensure that on start, the player will be in the correct game mode, at the right time, that no extra enemies will be present, and that we have the wave number and enemies defeated accurate.

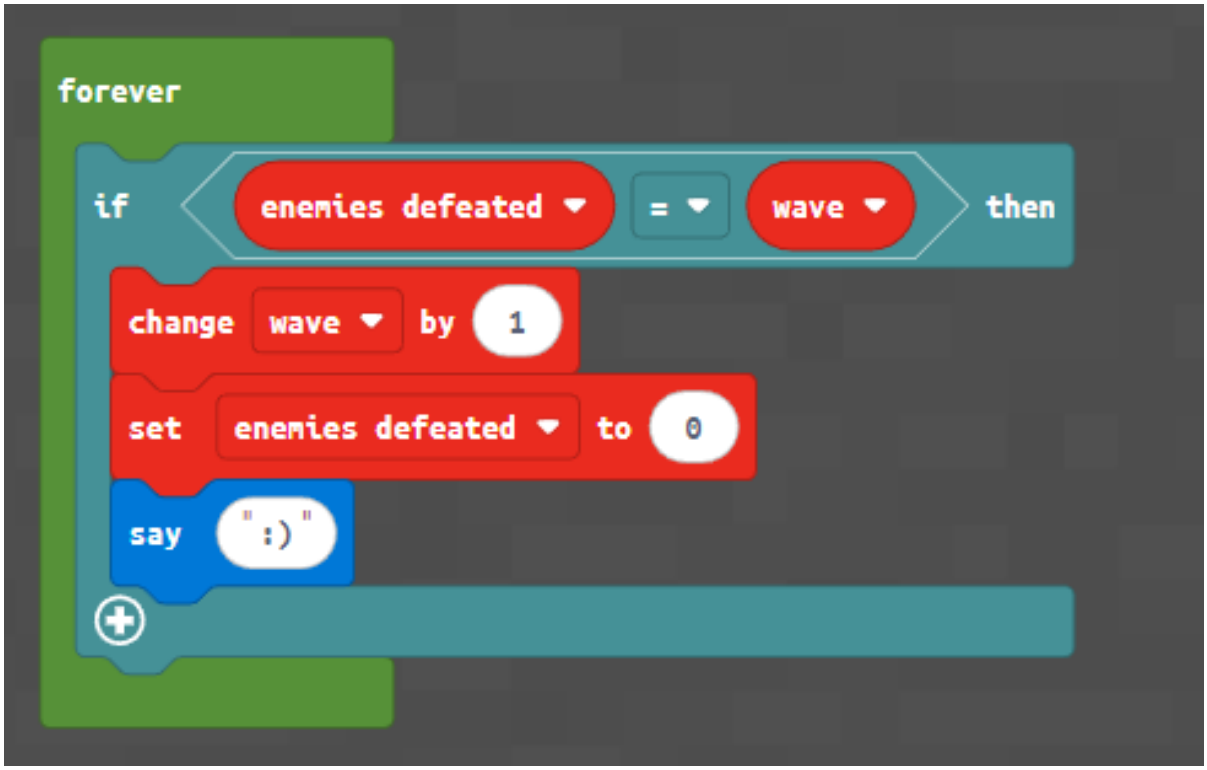
Once this is finished, we'll go over to the "on monster killed" block, and drag in the "change ___ by __" from **Variables** block, and make it say "change enemies defeated by 1". This will ensure that we can track when an enemy has been defeated. After this, to the "on player died" block, drag in a "change game mode to __ for __" from the **Gameplay** tab, making it say "creative" for "all players @a", and then after this, go to the **Mobs** tab, and drag in a "kill ____", making it say "kill all entities @e". This will kill all enemies when the game is over, and will set the player back to creative so that they can gear back up.



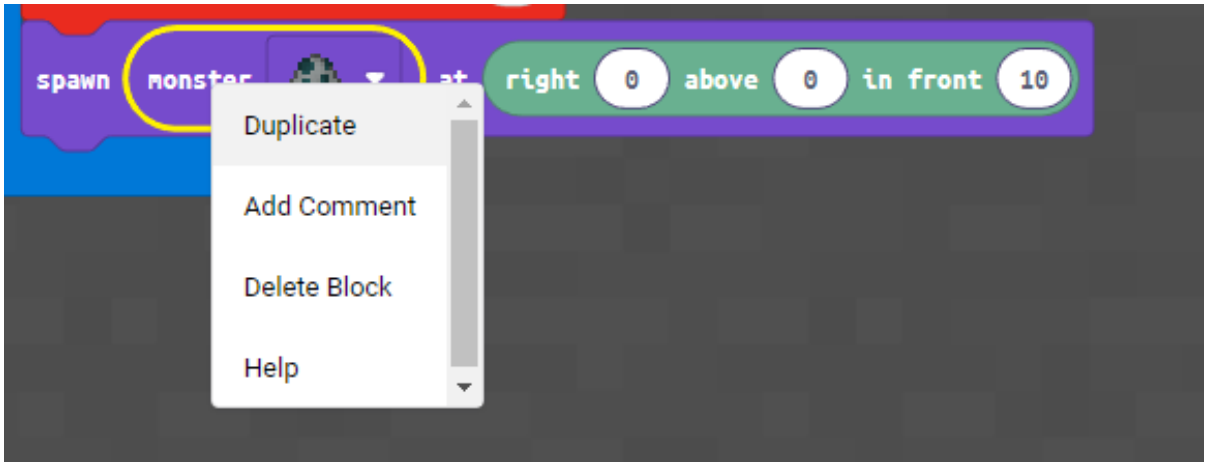
Finally, the bulk of the code; to begin, find the "if ____ then" block from the **Logic** tab. While still in the **Logic** tab, find the "__ = __" block, and drag it in carefully to the gap in the "if" block. Then, go to the **Variables** tab, and drag in both "enemies defeated" and "wave" into either side of the "__ = __" block.

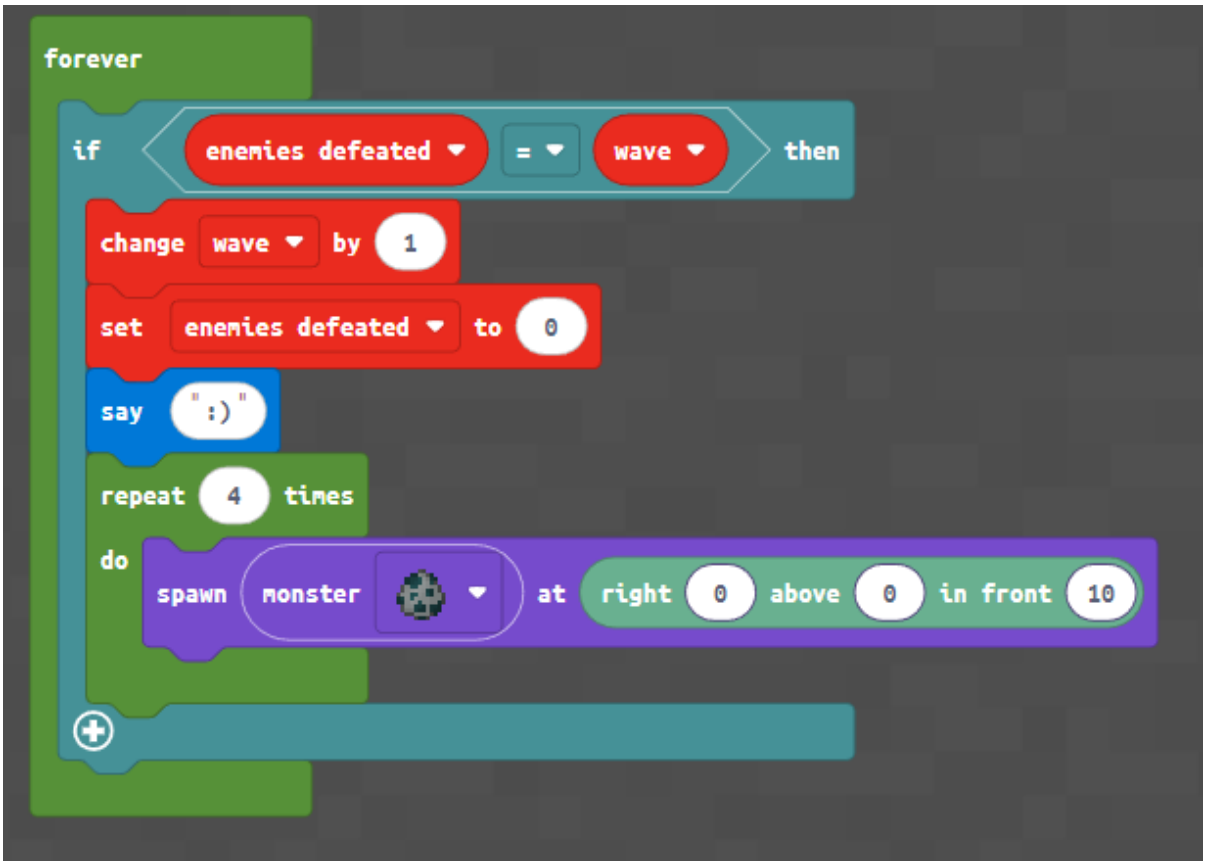


Once this is done, stay in the **Variables** tab, and drag in 1 block that says "set ___ to __", changing it to "set enemies defeated to 0", and then drag in a "change __ by __" block, changing it to "change wave by 1". After this, drag in a "say __" block from the **Player** tab.

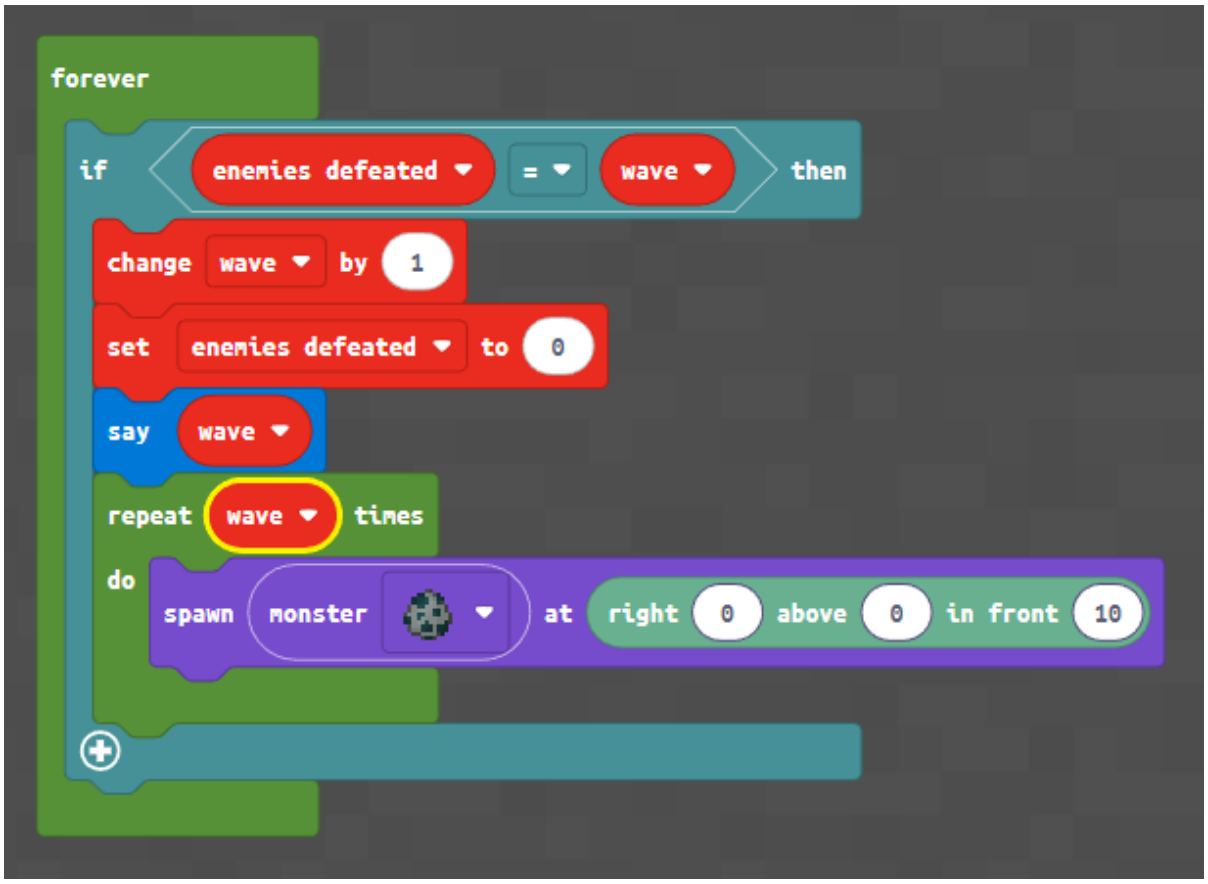


Next, we want to drag in the "repeat __ times" block from the **Loops** tab. After this, already inside of our code, find the "spawn monster block" from our "on chat command start" code. Then, right click on the block and select 'duplicate', and drag the copy into our "repeat __ times" block.



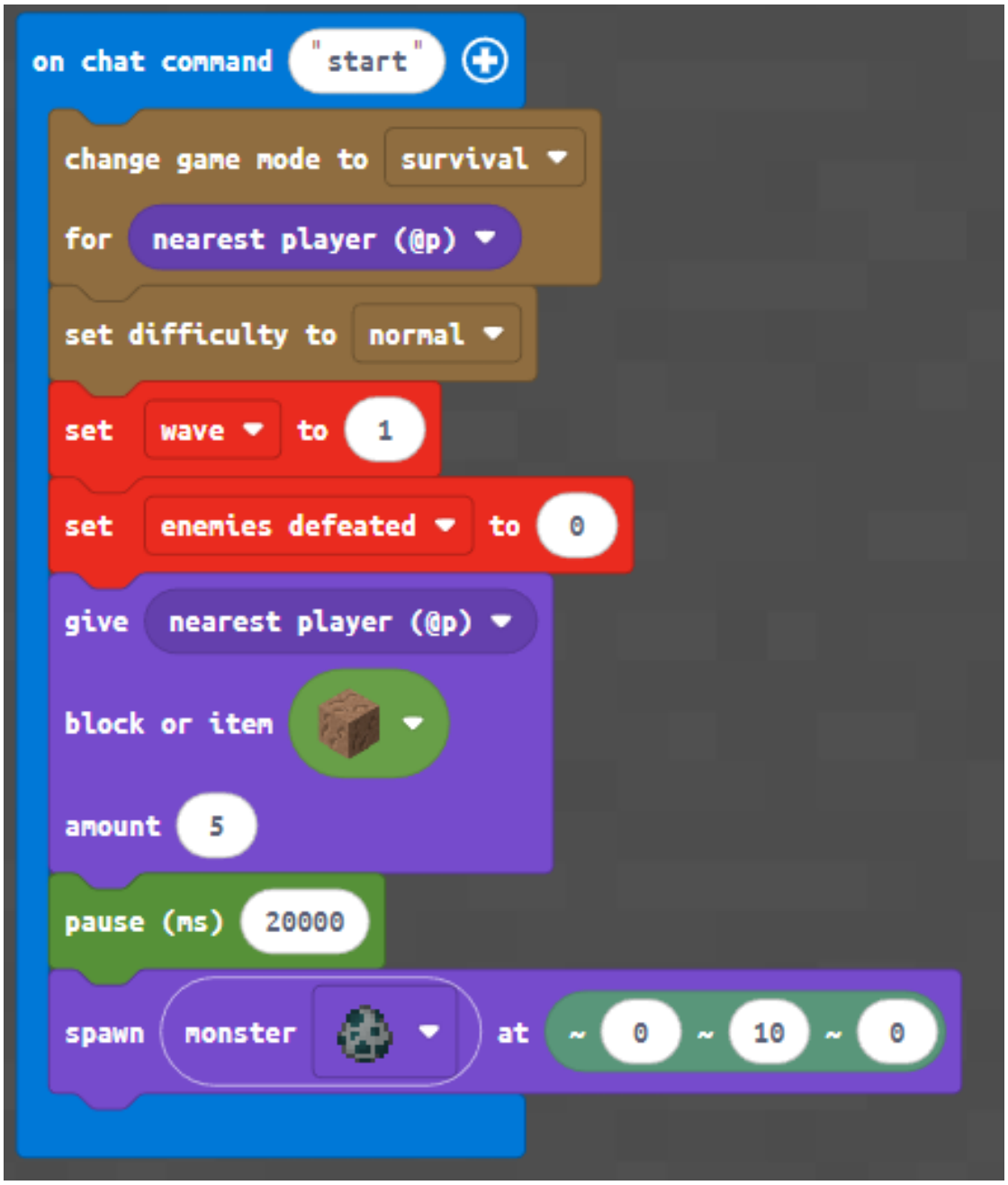


As a final step, go to the **Variables** tab, and drag in the "wave" block into the "say" block and the "repeat __ times" block. After this, the code will be complete!

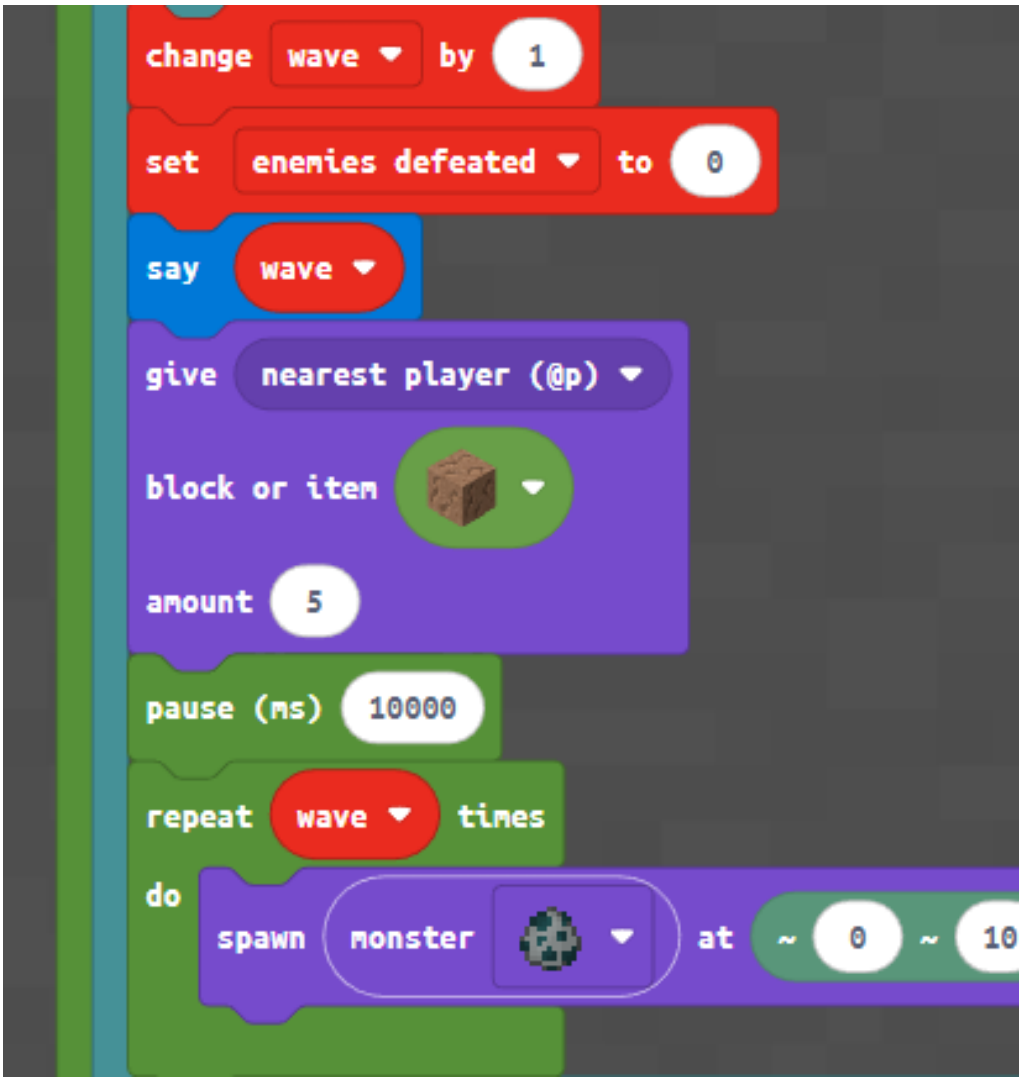


After this, there are only a few more steps to add our lucky blocks into the game:

First, in the "on chat command start" block, drag in a "give nearest player block or item" block from the **Mobs** tab, followed by a "pause (ms)" block from the Loops tab right above the "spawn monster" block. Select the same "mushroom" block from earlier after "block or item", and make the "amount" 5. After this, make sure that the "pause" is set to 15000. This will give the player 15 seconds to break their lucky blocks and equip their loot before the round starts.

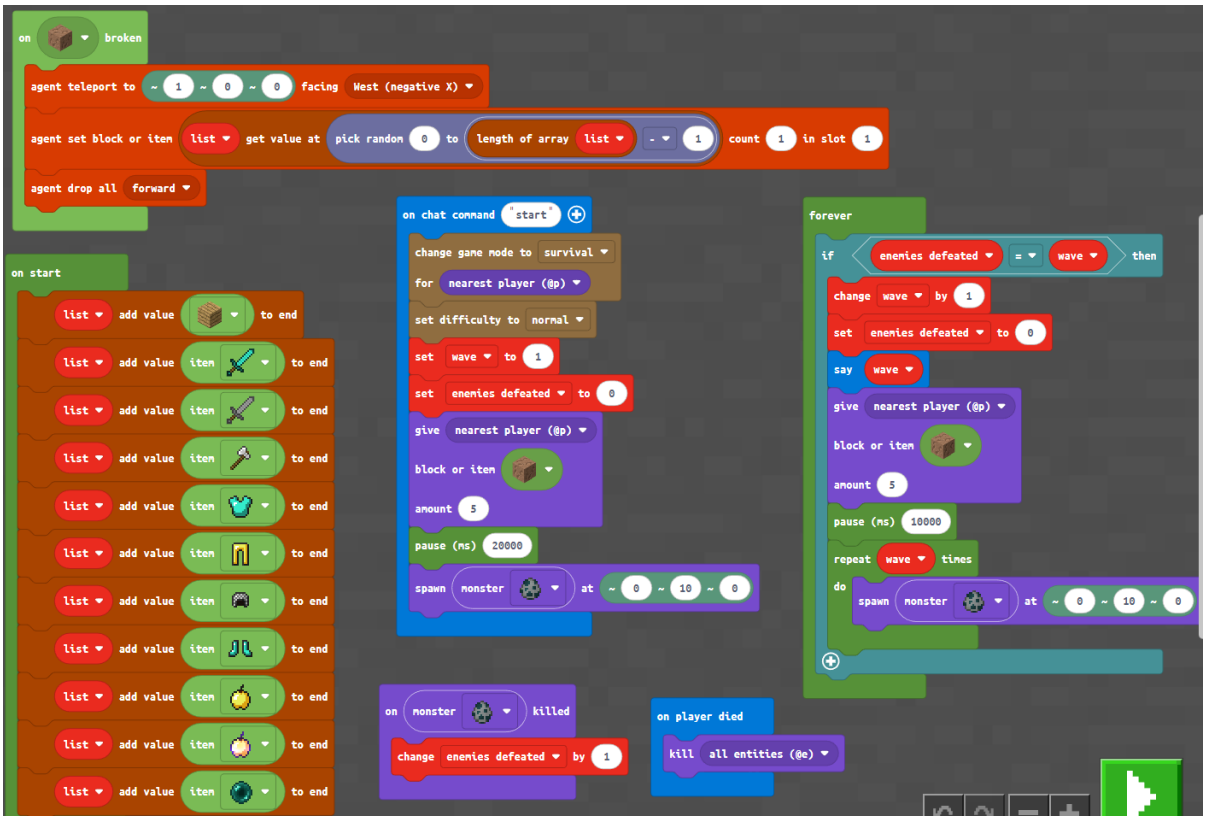


As a final step, right click and "duplicate" this give block, and place it after the "say wave" block in the forever block.



All done!

Here's the final code:



To begin a new round, simply type "start" in chat .

Save your project and test the code before continuing.

You have reached the end of the course. Revisit any lesson to practise a concept, or combine ideas from several projects to build a game of your own.

Save your project and test the code before continuing.