Introduction to Computer Science with MakeCode for Minecraft

Lesson 4: Variables

In this lesson, we'll explore the concept of a variable, an important way to store information and make your programs more flexible and adaptable. We'll build on what we have learned about Events by using a variable to pass additional information in when an event occurs. Finally, we'll challenge you to write your own program using variables to customize how it runs.

What is a Variable?
Computer programs process information. Some of the information that is input, stored, and used in a computer program has a value that is *constant*, meaning the value does not change throughout the course of the program. An example of a constant in math is 'pi' because 'pi' has one value that never changes.

Other pieces of information have values that *vary* or change during the running of a program. Programmers create *variables* to hold the value of information that may change. In a game program, a variable may be created to hold the player's current score, since that value would change (hopefully!) during the course of the game.

Ask the students to think of some pieces of information in their daily life that are *constants* and others that are *variables*. What pieces of information have values that don't change during the course of a single day (*constants*)? What pieces of information have values that do change during the course of a single day (*variables*)? Constants and variables can be numbers and/or text.
Examples: In one school day...

- Constants: The day of the week, the year, student's name, the school's address
- Variables: The temperature/weather, the current time, the current class, whether they are standing or sitting

Variables hold a specific type of information. The first time you use a variable, its type is assigned to match whatever type of information the variable is holding. From that point on, you can only change the value of that variable to another value of that same type.

Different types of variables:
A *number* variable holds numerical data.
Examples: a person's age, a player's score, the year.

A *string* variable holds a string of alphanumeric characters.

Examples: a person's name, a password, or the day of the week.

A *boolean* variable has only two possible values: true or false.
Examples: Is it daytime? Is the game over?

In MakeCode, a *Position* variable is a special kind of variable that holds three numbers that describe a specific location in three-dimensional space. We call these numbers X, Y, and Z coordinates.

Variable Name	Value	
myvarA	2	Variables can hold different types of values – numbers, text, coordinates, etc.
myvarB	4	
myvarC	"hello!"	
myvarD	(12, 4, -36)	

myvar A + myvarB = 6 Say myvarC Teleport to myvarD	Variables can be used in place of a value
Set myvarA to myvarB + 2 myvarA = 6	You can change the value of a variable

For the variables they listed earlier, have the students determine what type of data each variable holds.

Unplugged Activity: Slap, Clap, Snap – the Rhythm Robot

This is a simple, fun activity to practice the ideas of using variables to vary the actions of a function.
This function is a simple rhythm function. There are 3 movements to this function.

- Slap (player slaps their thighs or their desk)
- Clap (player claps their hands)

- Snap (player snaps their fingers)

Each of these movements takes a variable that tells the player (rhythm robot) how many times to do each of these movements: SlapNumber, ClapNumber, SnapNumber. The use of variables allows many different rhythms to be produced with just one function.

To play:

- Demo the three different movements in order: Slap, Clap, and Snap
- Each time the function runs, the rhythm robots will run through the sequence of Slap, Clap, and Snap 5 times.
- To determine how many times each of these 3 movements will be performed each time through the sequence pick, or have a student pick a random three digit number. (You could also roll a six-sided die three times to generate a random three digit number.)
- The first digit of the three digit number will be the value of SlapNumber and determines the number of times to slap your thighs.
- The second digit of the three digit number will be the value of ClapNumber and determines the number of times to clap your hands.
- The third digit of the three digit number will be the value of SnapNumber and determines the number of times to snap your fingers.
- 'Run' your function by having the students Slap, Snap, and Clap using the values of the variables SlapNumber, ClapNumber, and SnapNumber.

For example: Given the random number 231, the students would Slap their thighs twice, Clap three times, then Snap their fingers once and repeat that sequence 5 times.

- Pick a new random number to use as values for the variables and run the function again.
- Continue picking random numbers and running the function to see how the different values for the variables affects the results.

Variations:

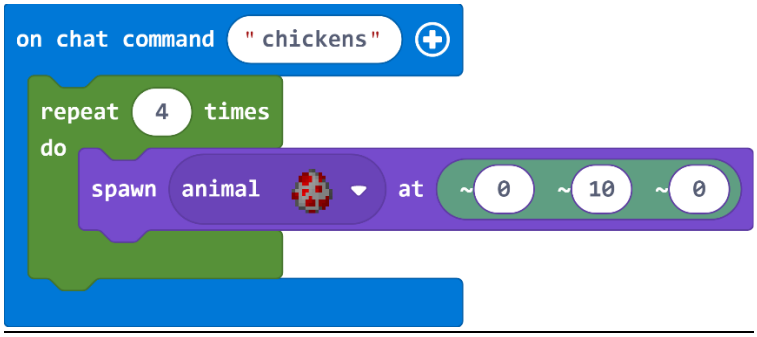
- Let students suggest different movements to add to the Slap, Clap, Snap sequence. They can then add another variable whose value will determine how many times this new movement is performed.
- The number of times to run through the Slap, Snap, Clap sequence can be another variable!

Activity: Chicken Storm

Let's use a variable to determine the number of chickens to spawn in Minecraft, and we'll make these chickens fall from the sky like a storm of chickens!

<u>Steps:</u>

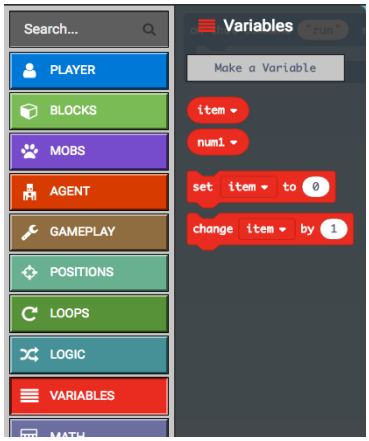
1. Create a new MakeCode project and name it “Chicken Storm”
2. From the Player Toolbox drawer, drag an On chat command block into the coding Workspace and change the command to “chickens”
3. From the Loops Toolbox drawer, drag a Repeat block into the On chat command “chickens” block
4. From the Mobs Toolbox drawer, drag a Spawn animal block into the Repeat block
5. In the Spawn animal block, change the middle value to 10, to make the chickens spawn 10 blocks above your head

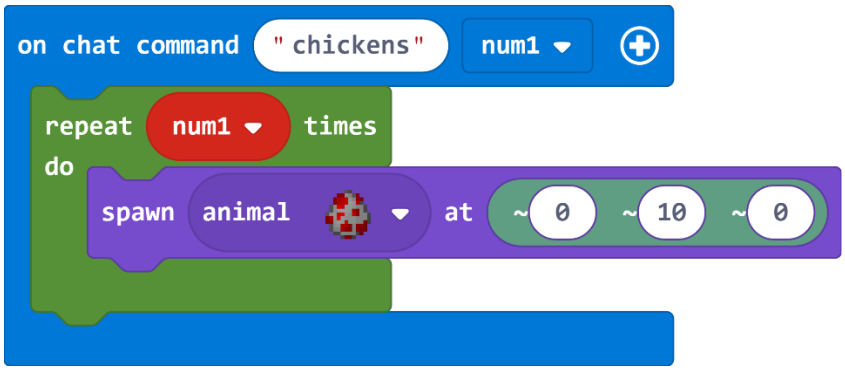


Then, go into Minecraft, press ‘T’ to open the chat window, and type in “chickens” to the chat window. It’s raining hens!

You can make the number in the repeat block higher, to have more than 4 chickens in our chicken storm. Let’s go one step further and allow you to tell the On chat command block exactly how many chickens to spawn when you give the command using a variable.

6. Click the plus sign (+) just to the right of the On chat command “chickens” block. You should see another drop-down menu appear, with the name “num1” at the top of it. num1 is a number variable that you can use in your code.
7. From the Variables Toolbox drawer, drag the num1 block into the Repeat block replacing the number 4

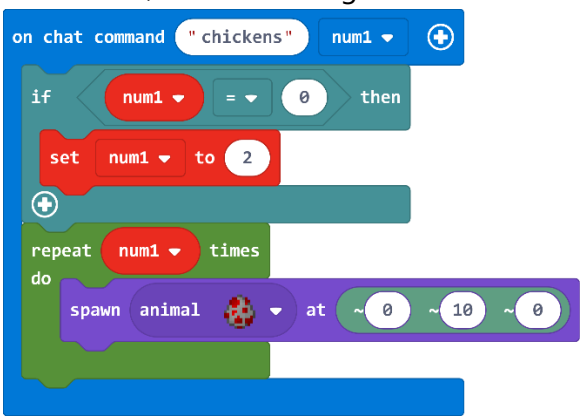




Now, when you press 'T' in Minecraft, if you type in "chickens 15" then the variable num1 will take the value of 15. If you type "chickens 25" then num1 takes the value of 25. The number you pass in after the command "chickens" is called a parameter, or an argument. A variable is like a container that holds the value of any parameter you pass in.

Optional Extension: Create a Default Chicken Value

Now that we have created a chicken command that takes a number variable and produces that many chickens, it is good programming practice to make sure to handle cases where "chickens" isn't given a number. By default, num1 has a value of zero. You can use a conditional statement to check if num1 has a value of zero, and set it to a default value of 2 chickens. That way if you type the "chickens" command but forget to type a number after it, you will at least get a couple of chickens, which isn't as good as 10 chickens, but it's better than no chickens at all!



Shared Program: https://makecode.com/_08WV7RPa53qw

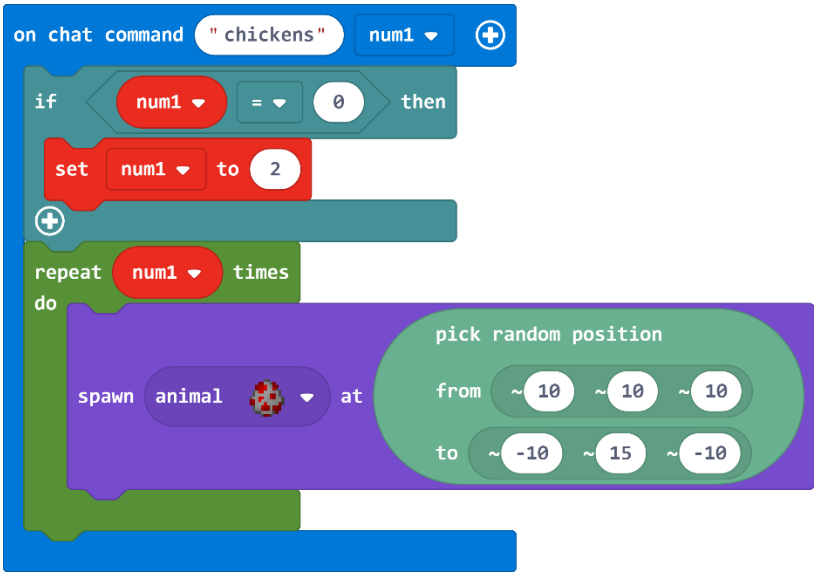
JavaScript:

```
player.onChat("chickens", function (num1) {
    if (num1 == 0) {
        num1 = 2
```

```
    }
    for (let i = 0; i < num1; i++) {
        mobs.spawn(mobs.animal(AnimalMob.Chicken),
positions.create(0, 10, 0))
    }
})
```

Optional Extension: Chicken Storm

You can further improve the realism of the chickens command by using the Pick random position block from the Positions Toolbox drawer. This block will scatter the chickens randomly around the area described by the two coordinates. You can also vary the height of the drop, so the chickens will land at different times. Now it is truly a storm of chickens!



Shared Program: https://makecode.com/_HpD1L6LemCVu

JavaScript:

```
player.onChat("chickens", function (num1) {
    if (num1 == 0) {
        num1 = 2
    }
    for (let i = 0; i < num1; i++) {
        mobs.spawn(mobs.animal(AnimalMob.Chicken),
positions.random(
```

```
        positions.create(10, 10, 10),
        positions.create(-10, 15, -10)
        ))
    }
})
```

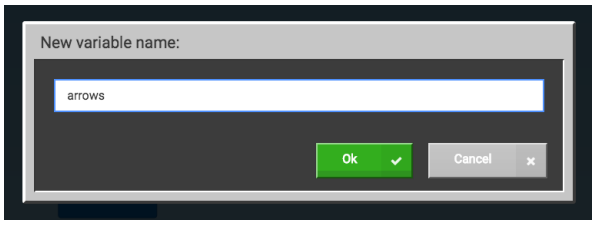
For more fun with commands and variables, try this interactive tutorial that makes a Jump command that sends you as high as you want!
Mega Jump – https://minecraft.makecode.com/tutorials/mega-jump

Activity: Arrow Counter

You can use variables to count the number of times something happens in your world. You can use them to keep score, or you can use a conditional statement to cause something to happen when the counter gets to a certain number. Let's look at an example that uses a counter to keep track of the number of arrows we shoot.

Steps:

1. Create a new MakeCode project and name it "Arrow Counter"
2. In the Variables Toolbox drawer click the "Make a Variable" button
3. Name the new variable "arrows"



Now this new arrows variable will be available to you in the drop-down menus of many of the blocks that use variables.

4. From the Player Toolbox drawer, drag an On arrow shot block into the coding Workspace
5. From the Variables drawer, drag a Change item block into the On arrow shot block
6. In the Change item block, use the drop-down menu to select the variable name arrows

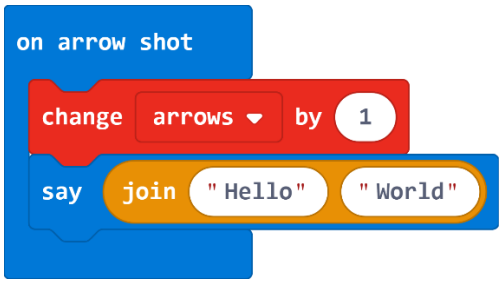
Now, every time you shoot an arrow, the counter will increase by 1 (if you want to decrease the value of a variable, use a negative number.)

Let's create a way to actually see the value of the variable. The Say block prints a line of text to the chat window at the top of the Minecraft game screen.

7. From the Player Toolbox drawer, drag a Say block under the Change arrows block

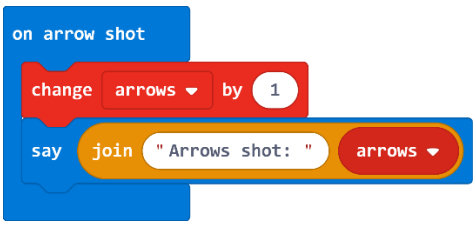
Let's join the number variable arrows with some text so we can get a running totals of arrows shot.

8. Click the Advanced category in the Toolbox to display the Text Toolbox drawer
9. From the Text Toolbox drawer, drag a Join block into the Say block replacing the "Hi" string



10. In the first slot of the Join block, type in "Arrows Shot: "
11. From the Variables Toolbox drawer, drag the arrows variable block into the second slot of the Join block, replacing string "World"

This block will join the string label with the number variable and output a string that the Say block will print at the top of the screen every time you shoot an arrow.



Go into Minecraft and try it out! You can give yourself a bow and arrows by typing the following commands into the Chat window:

/give @p bow
/give @p arrows 64

Shared Program: https://makecode.com/_1DPAgv2iX9s4

Activity: Fall is in the Air

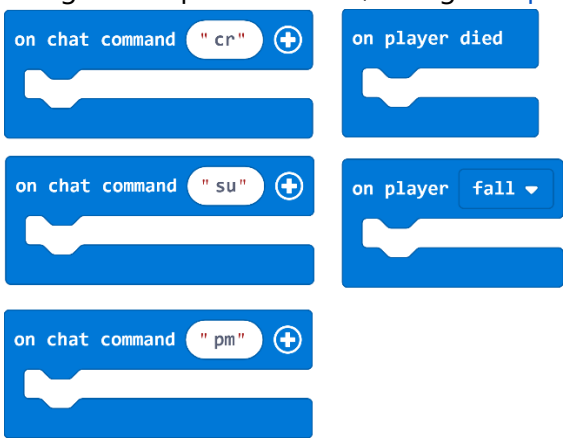
A Minecraft Survival world is full of high places. Let's use an Event Handler and a counter to come up with a useful post-mortem report after you have fallen from a high place.

For this project, you will either want to be in a Survival world next to a large cliff, or you can toggle between Creative mode and Survival mode and fly up to a high point for testing.

We will create a variable that keeps track of the distance you have fallen, and report that distance in meters after you respawn.

Steps:

1. Create a new MakeCode project and name it "Falling"
2. From the Player Toolbox drawer, drag three On chat command blocks to the coding Workspace
3. Name these commands to "cr" (creative), "su" (survival), and "pm" (post-mortem)
4. From the Player Toolbox drawer, drag out two event blocks: On player walk, and On player died
5. Using the drop-down menu, change On player walk to On player fall



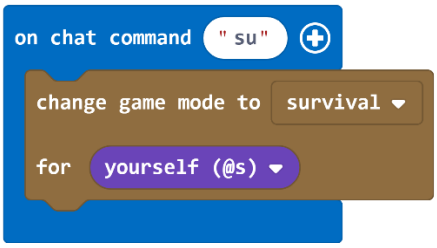
The first thing we'll do is use the On chat command "cr" to change the game mode to Creative mode. You use this to fly.

6. From the Gameplay Toolbox drawer, drag out a Change game mode block and drop it in the On chat command cr block
7. In the Change game mode block, use the drop-down menu to select 'creative' mode, and click on the Player Selector block to select 'yourself @s'

Now, to switch to Creative mode, all you need to do is issue the chat command "cr" in the game, and you will be able to double-tap the space bar to fly straight up, as high as you want.

We'll want to use the same block to switch into Survival mode.

8. From the Gameplay Toolbox drawer, drag out a Change game mode block and drop it in the On chat command su block
9. In the Change game mode block, click on the Player Selector block to select 'yourself @s'



10. In the Variables Toolbox drawer, click the 'Make a Variable' button to create a new variable, and name it "fall"

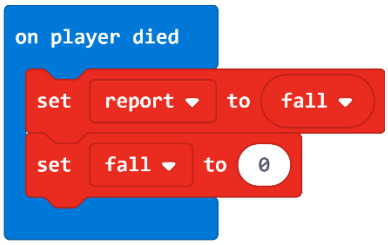
When the player falls, this variable will start incrementing while the player is falling to keep track of the number of blocks the player fell. We will also use another variable to store the final number of blocks the player fell.

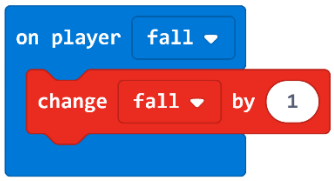
11. In the Variables Toolbox drawer, click the 'Make a Variable' button to create a new variable, and name it "report"
12. From the Variables drawer, drag the Change item block into the On player fall block in the coding Workspace
13. In the Change item block, use the drop-down menu to select the variable fall instead of item
14. From the Variables Toolbox drawer, drag the Set item block into the On player died block
15. In the Set item block, use the drop-down menu to select the variable report instead of item

16. From the Variables Toolbox drawer, drag the fall block into the second slot of the Set block replacing the '0'

This will save the value of fall in the report variable so you can then reset fall to zero.

17. From the Variables Toolbox drawer, drag another Set item block into the On player died block and use the drop-down menu to change the variable to fall



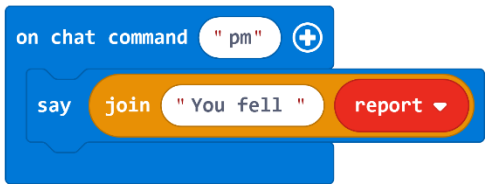


Finally, we need to report the height of your fall with a little post-mortem report. Remember that variables have a specific data type and the Say block expects a string, not a number. So we can't just Say the variable report. We will have to convert a number into a string.

18. From the Player Toolbox drawer, drag a Say block into the On chat command pm block
19. Click the Advanced tab in the Toolbox to see the Text Toolbox category
20. From the Text Toolbox drawer, drag a Join block into the Say block replacing "Hi!"

The Join block will put two variables or literal strings together, and it is treated as a string. So, even though report is a number variable, when we join it together with another string, it becomes a string.

21. In the Join block, type "You fell " in the first slot replacing "Hello"
22. From the Variables Toolbox drawer, drag the report variable block into the second Join block slot replacing "World"

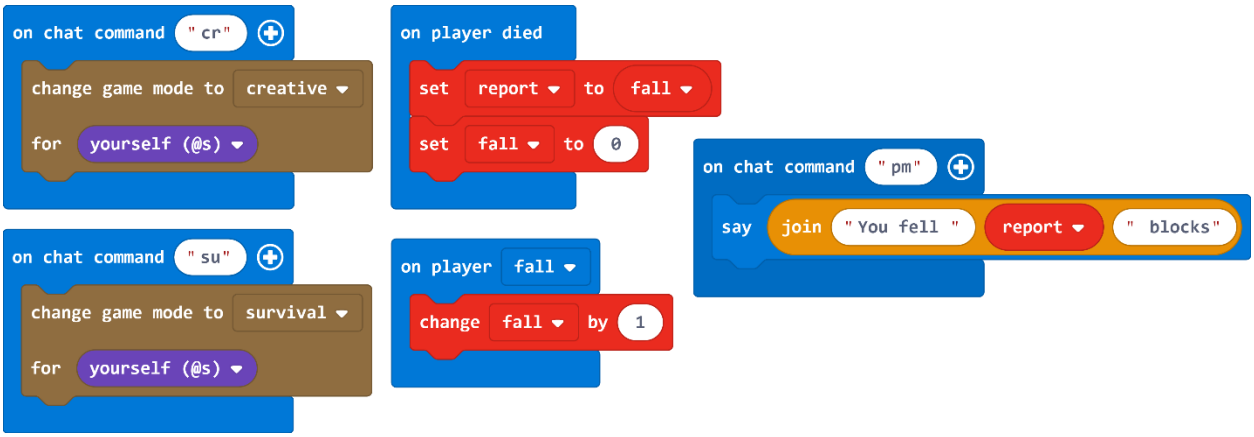


This will work, but it will look something like "You fell 43". Well, 43 what? We should add some units on the other side of that variable. In order to do this, we will need to go into JavaScript.

23. Click on the JavaScript toggle button at the top of the screen to switch into the MakeCode JavaScript editor
24. Find this line of code: player.say("You fell " + report)
25. Add the string " blocks" to the end of the line so it looks like this: player.say("You fell " + report + " blocks")
26. Click on the Blocks toggle button at the top of the screen to switch back into the MakeCode Blocks editor

That's it! Now, type "cr" in the chat window to enter Creative mode, double-tap the space bar and then press up until you are high in the sky, then enter "su" in the chat window to fall back to earth. After you die, click "Respawn" and then enter "pm" in chat window to see a report of how far you fell!

Complete program



JavaScript:

```
let fall = 0
let report = 0
player.onChat("cr", function () {
    gameplay.setGameMode(
    GameMode.Creative,
    mobs.target(TargetSelectorKind.LocalPlayer)
```

```
    )
})
player.onChat("su", function () {
    gameplay.setGameMode(
        GameMode.Survival,
        mobs.target(TargetSelectorKind.LocalPlayer)
    )
})
player.onDied(function () {
    report = fall
    fall = 0
})
player.onTravelled(TravelMethod.Fall, function () {
    fall += 1
})
player.onChat("pm", function () {
    player.say("You fell " + report + " blocks")
})
```

Shared Program: https://makecode.com/_MoyCau9H9VvP

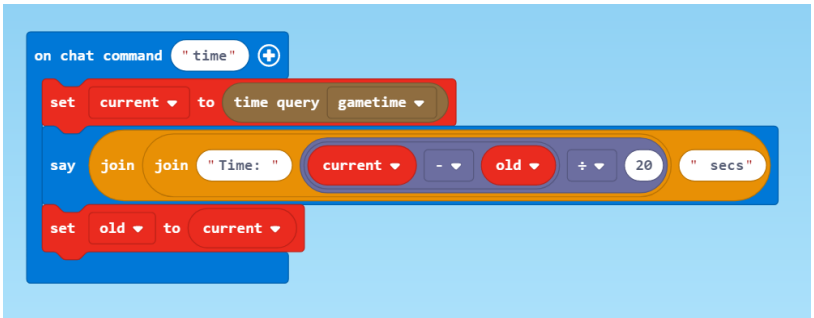
Independent Project

In this lesson, we've looked at how to create a new variable, and how to change the value of an existing variable. We learned how to pass the value of a variable into the On chat command block as a parameter, and we also looked at one way to use a variable to keep track of events that happen in the game.

For this project, create an original MakeCode project that uses multiple variables of at least two different types, to keep track of information in Minecraft. Also find a way to use a chat command with a parameter (this can be in the same project, or a different project.)

Here is an example:
Alpine Race

Take turns jumping up a mountain. Keep track of the elapsed time as well as the number of jumps it took each person from start to finish. Here is some starter code that you can use to do the timing.



This code creates two variables, one to hold the current game time in ticks, and the other to hold the old value. When you type "time" in the chat window, it will display the number of seconds that have elapsed since the last time you called Time.

To time the race, you might want to set one value to the current game time when the race starts, and then set the second variable to the game time when the race ends. Then you can just do a subtraction operation.

Minecraft Diary

Compose a diary entry addressing the following:

- What type of information did you choose to keep track of? How?
- What problems did you encounter? How did you solve them?
- How did you use variables in your project and what were their types?
- What was something new that you learned for this project? Describe how you figured it out.
- Include at least one screenshot of your project.
- Share your project to the web and include the URL here.

Assessment

	1	2	3	4

Diary	Minecraft Diary entry is missing 4 or more of the required prompts.	Minecraft Diary entry is missing 2 or 3 of the required prompts.	Minecraft Diary entry is missing 1 of the required prompts.	Minecraft Diary addresses all prompts.
Variables	No variables are implemented.	At least 1 variable is implemented in a meaningful way OR Variables are all of same type	At least 2 different variables are implemented in a meaningful way. Variables are of different types (text, number, boolean, and/or Position)	At least 3 different variables are implemented in a meaningful way. Variables are of different types (text, number, boolean, and/or Position)
Chat / Parameter	Project uses no chat command at all.	Project uses a chat command but does not implement parameters.	Project uses a chat command with one or more parameters that are not used by the code.	Project incorporates a chat command that uses one or more parameters in the enclosing code.

CSTA K-12 Computer Science Standards

- CL.L2-03 Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities

- CT.L1:6-01 Understand and use the basic steps in algorithmic problem-solving
- CT.L1:6-02 Develop a simple understanding of an algorithm using computer-free exercises
- CPP.L1:6-05 Construct a program as a set of step-by-step instructions to be acted out
- 2-A-5-7 Create variables that represent different types of data and manipulate their values.