Western Engineering

Outreach

Micro:bit Can't Catch Me

Grade 6-8

Meet Today's ENG HERO!

Luiz Capretz - Professor with Western Engineering

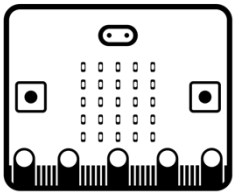
Dr. Capretz has vast experience in Software Engineering as practitioner, manager, and educator. He has worked, taught and done research on the engineering of software in North and South America (Canada, Brazil and Argentina), Europe (U.K. and Italy), Middle East (United Arab Emirates), and Asia (Japan and Malaysia) since 1981. To learn more about his research visit: https://www.eng.uwo.ca/electrical/faculty/capretz_l/index.html

Learning Goal:

- Students will get an introduction to coding and software engineering by creating a fun and interactive game which tests their logical thinking and creativity skills.
- By creating this game, students will have a better understanding of how their favourite games are created. They will learn how to use variables, loops, if-statements and inputs.
- Curriculum Connections: Grade 6-8 - The Four Step Problem-Solving Model

Materials Needed:

- Computer with connection to the internet
- Micro:bit (optional - can be purchased online https://www.amazon.ca/Micro-Micro-Controller-Detection-Compass-Bluetooth/dp/B01G8WUGWU/ref=sr_1_4?keywords=microbit&qid=1585925584&sr=8-4 or https://www.elecfreaks.com/store/elecfreaks-micro-bit-starter-kit.html)



Engineering and Science Connections:

Today, we will be explore coding and software engineering!

What is coding?

Code is a set of instructions that a programmer gives to a computer to tell the computer what they want it to accomplish.

Why do we need this set of instructions?

Computers cannot think for themselves, they can only read what they are being told. We must communicate with computers to be able to make them function.

Can we tell a computer what to do out loud?

No, computers only speak very specific languages, which must be given to them through text.

What are the names of some of the languages that computers understand?

- Drag and Drop
- Scratch
- Hour of Code
- Java
- C++
- C#
- Python
- HTML
- etc.

Today we will be using drag and drop (also known as block) coding. It is a great tool for beginners learning how to code, as it includes a lot of the same terminology used in other coding languages.

When real programmers are designing a code, before they write a single line of code, they come up with a master plan of how they are going to make their code work. This master plan is called a ***Flow Chart.***

The reason flowcharts are so important in programming is because they help us see the entire picture of what we are trying to accomplish. Coding can include very small details that can sometimes be overlooked if you don't have a clear picture of what your code needs to include.

While these flowchart include the big ideas, it is only the main ideas of this process. When we write the flowchart for a computer program, we only include the main ideas of what we need our code to do, then more detailed individual lines of code will be added later when we actually begin to write the program.

There are many different ways to code a program with the same end goal. Often we will find that 2 codes can do entirely the same thing, but look nothing alike.

Coding Vocabulary

- **Variables:** used to store values such as numbers. It is used so that values can easily be changed rather than each value being changed manually one by one. Also can be used to store and change important values that can be used later on.
- **Loops:** used to repeat lines of code rather than writing out those lines of code over and over. The forever loop function repeats code an infinite amount of times.
- **If-statements**: used to execute specific lines of code only if a condition is true.
- **Inputs:** Something that you put into the computer which will most likely be transformed by the system in order to produce an output. Inputs on the micro:bit include the A and B button, and shaking, tilting, and moving the micro:bit in certain ways (among others)
- **Pause:** used to delay code. Can be used in loops to prevent the code from executing too fast. Measured in milliseconds.
- **x,y coordinates**: x coordinate represents horizontal positions in a graph, while y coordinates represent vertical positions in a graph.

How does my program get onto the micro:bit?

For your program to work on the micro:bit, first it has to be compiled. Compiling means to translate a program into a more efficient computer language. When you hit the Download button on the coding interface, your program is compiled into a hex file that contains the machine code and the instruction set used by the ARM processor that is on your micro:bit. Compiling to ARM machine code actually happens in the web browser, where the code from script is joined with the machine code of the micro:bit runtime. Then, all you have to do is plug in your micro:bit and it will pop up like a USB drive. Drag the hex file onto the micro:bit drive. A yellow light on the back of your micro:bit will flash. When it is done flashing, the code is on the micro:bit! Now it should run the code.

Video Recommendation: *What is Coding?*

https://www.youtube.com/watch?v=cKhVupvyhKk

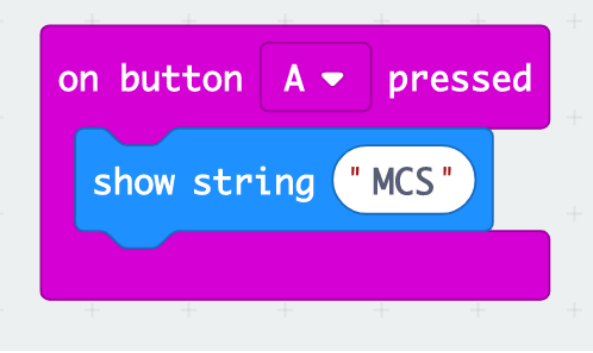
Activity:

- Go to https://makecode.microbit.org/
- Click the purple new project button. It will bring you to a new screen where we will make our code. Notice on the left side there is a picture of a micro:bit. This will show you what the code will look like on the micro:bit. Take some time to explore the blocks on the side. To put them in the coding space, just click and drag. The pieces fit together like puzzle pieces. To delete, click and drag the block back into the side bar. A garbage can will pop up and the block will disappear.

Part 1: Display Your Initials

- To familiarize yourself with the blocks, first you will display your initials. You will need some kind of input block and some way to light up the LEDs on the micro:bit. Check in the basic block section. Test it out!
- One solution for displaying your initials is dragging a "show string" block from Basic into the "on start block" and typing your initials.

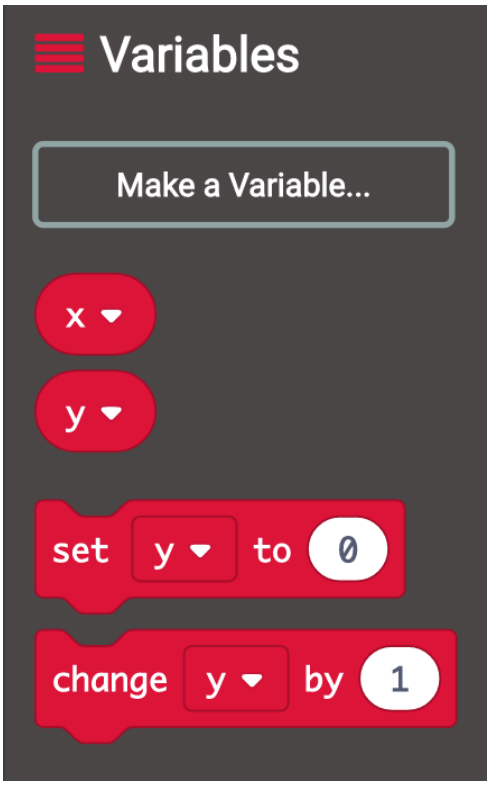
- This will show your initials once when the micro:bit starts.
- Another way is to use a "on button A pressed" block from the input section.



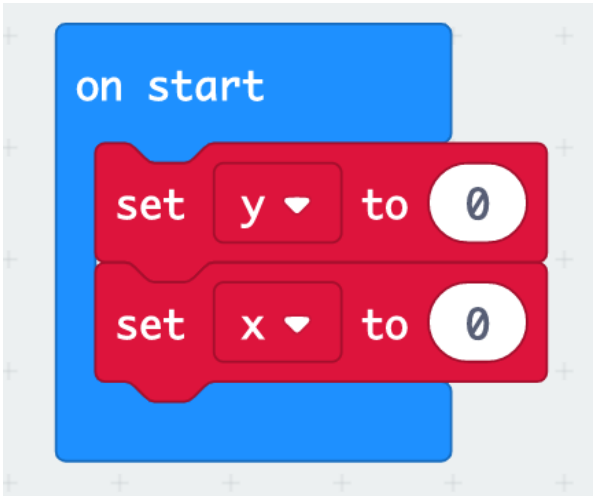
- This will show your initials every time you press button A on the micro:bit. Test it on the micro:bit on your screen by clicking the A button on the micro:bit.
- What other ways are there to display your initials? Try some more.

Part 2: Can't Catch Me Game

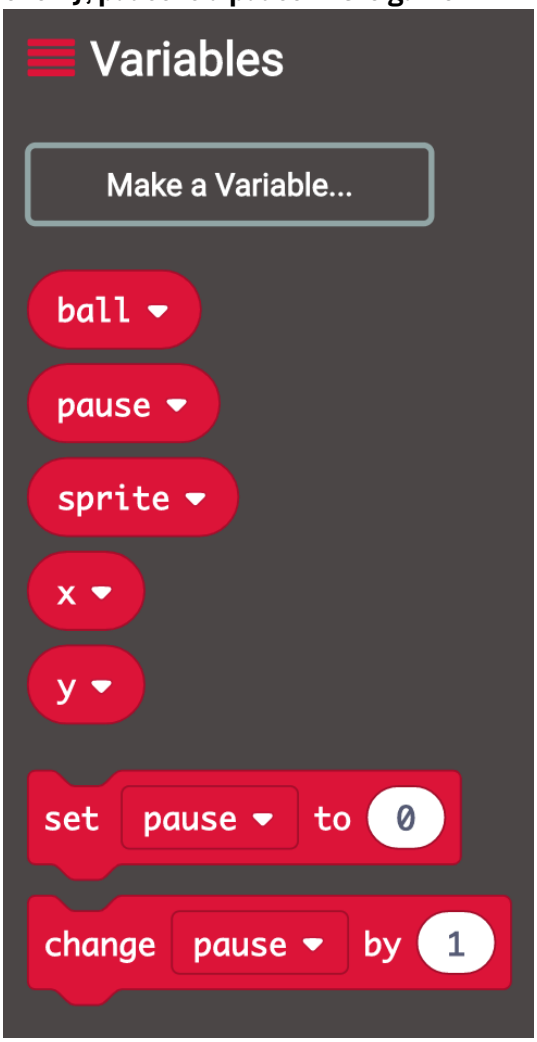
- We are going to create a game where your player has to avoid an enemy. The enemy will randomly appear and we must code the a and b button to move our play around the screen. If the enemy touches our player, the game ends.
- First we are going to create some variables. Click the red variable section. Click "make a variable." Name it x. Click "make a variable" again." Name it y. Now we should have two variables named x and y.
  The variable section should look like this:



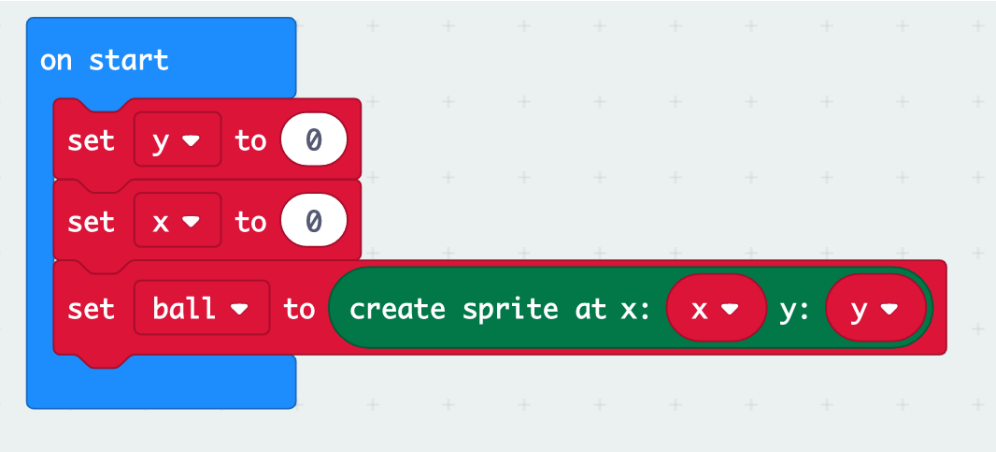
- Drag the "set y to 0" block onto an "on start block". Drag another "set y to 0 block" and place it underneath the first one. Click the little drop down arrow next to y and change it to x. Now we have two blocks that set the initial values of x and y to 0.



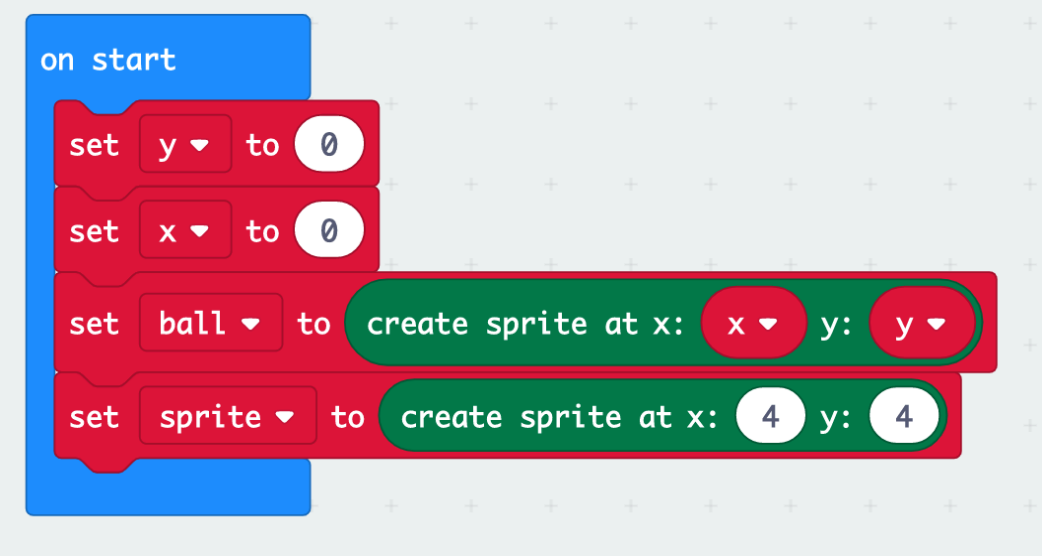
- Now we will make three more variables: ball, sprite, and pause. The ball is our character, the sprite is the enemy, pause is a pause in the game.



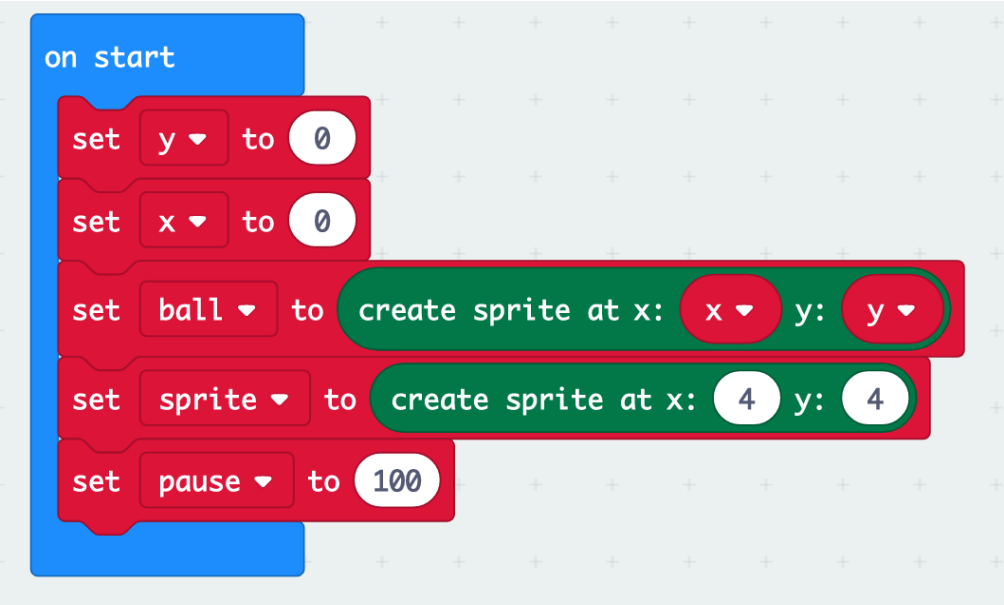
- Drag another "set variable to 0" block onto the "on start block". Use the drop down arrow to select ball. Next, we will drag a block into where it says 0. In the block menu, click on advanced. More blocks will appear. Select the dark green game section and drag a "create sprite at x y" block into the "set ball to 0". We want this variable to create a player where the variables of x and y are. So, from the variable section grab the block that says "x" and drag it into the first spot on the "create sprite at x y" block. Drag a "y" variable into the second spot. It should look like this:



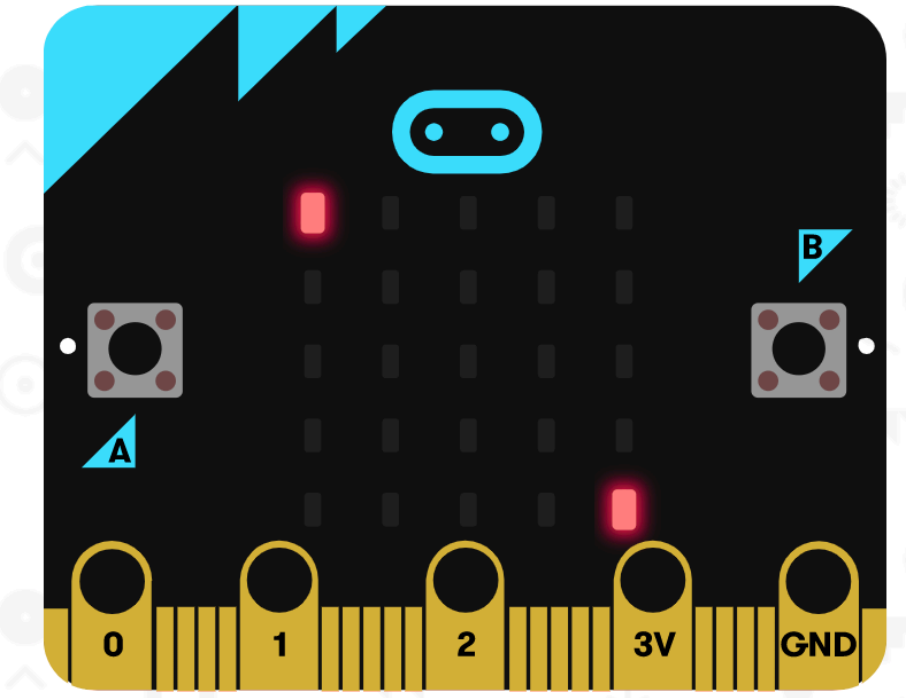
- Now we will drag in another "set variable to 0" block. Click the drop down and change it to "sprite." Drag in another "create sprite at x y" from the game section. Change the numbers to 4 for both x and y.



- For the last block in this section, drag another "Set variable to 0" block. Select "pause". Make the number 100.
- The on start block should look like this now:

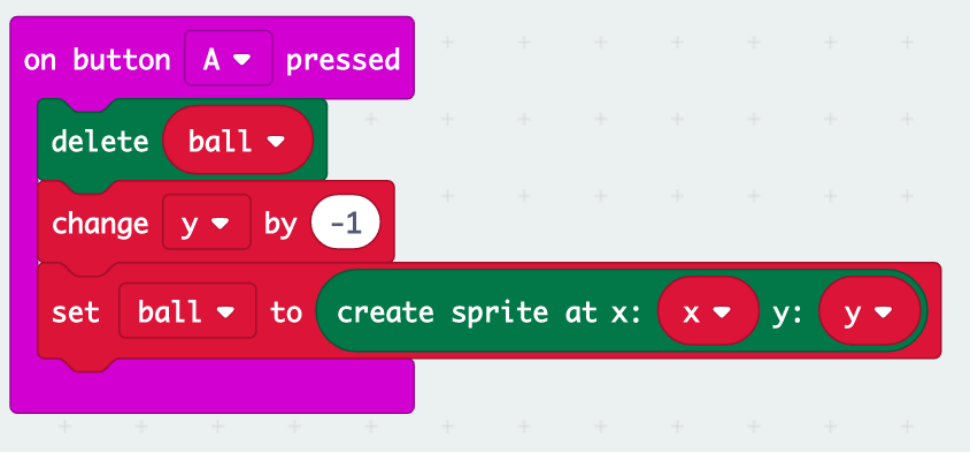


- In this section, we have defined our variables x and y. Then the variable ball will create a sprite at the x and y coordinates. For now, that will be at (0,0). This will appear on the micro:bit as an LED lit up. The variable sprite will create an enemy sprite at location(4,4). The pause variable is set to equal 100. Your on screen micro:bit will look like this:



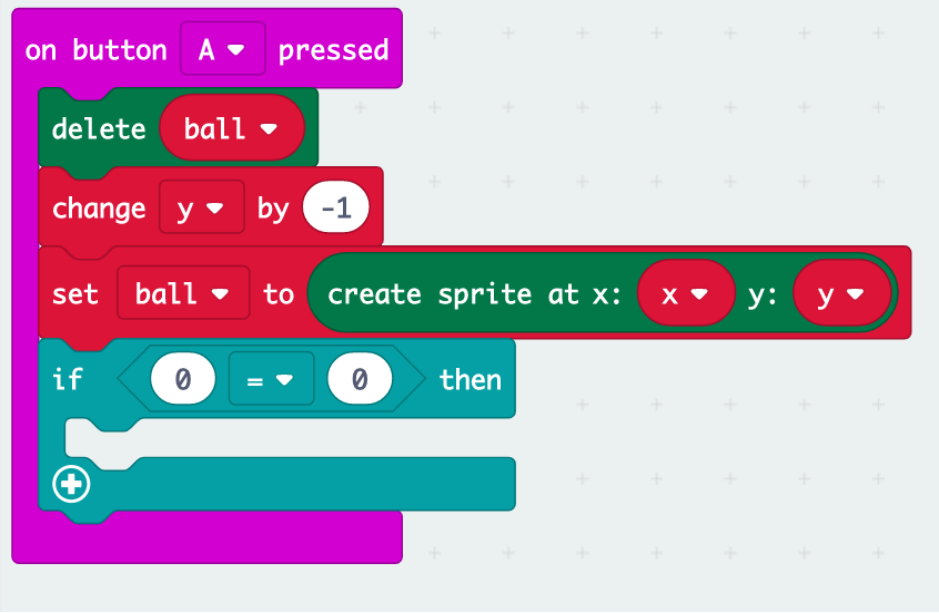
- Next, we will code button A and button B to move our character (the ball variable) around the screen. Each click of the A button will move our character up. Each click of the B button will move our character right.
- Drag an "on button A pressed" block from the pink input section. From the dark green game section drag a "delete variable" block. Click the drop down menu and select ball. This will delete the location the ball is currently in (turning off the LED at (0,0))

- From the variable section, add a "change variable by 0" block. From the drop down menu select y. Change the number to -1. From the variable section again, drag a "set variable to 0 block." From the dropdown menu select ball. Drag a "create sprite at x y" block from the game section. From the variable section drag an x variable and put it in the first spot. Put a y variable in the second spot. It should look like this:

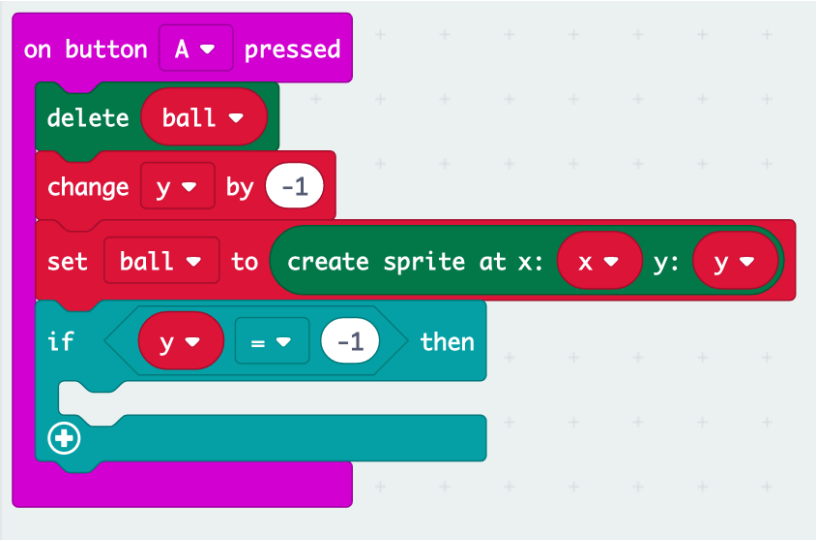


This section changes the y variable, then makes the ball appear at the new spot it created by changing the y variable.

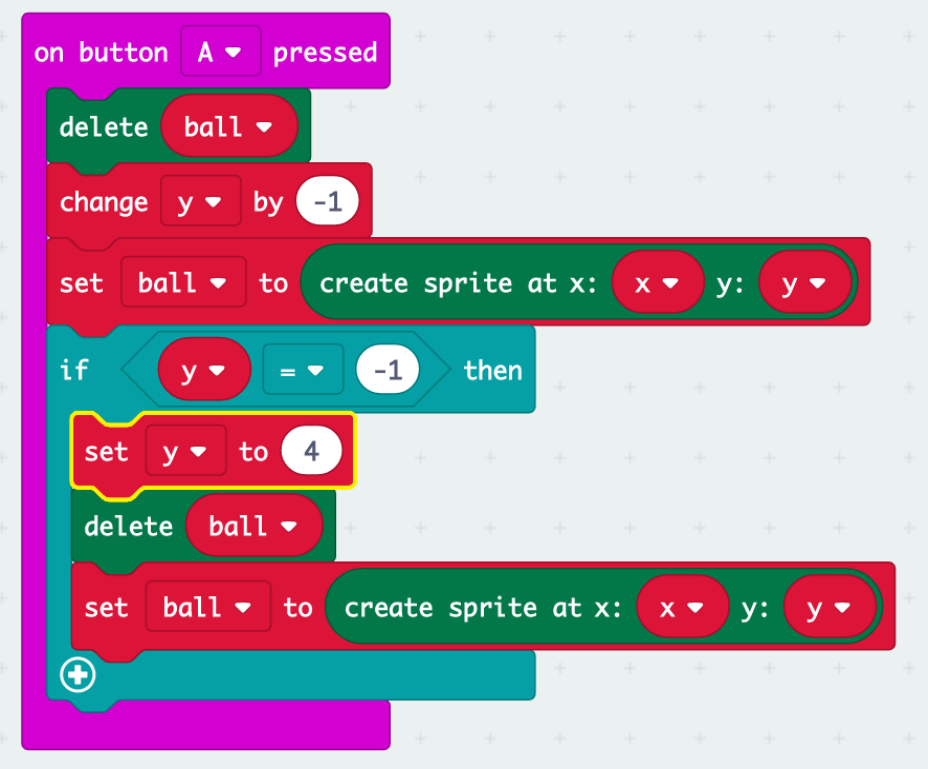
- Next, we will include our first logic block. From the logic section, drag an "if true then" block and add it to the on button a pressed. From the logic section, drag a comparison block "0 = 0". Place this where it says true in the "if true then" block.



- Next from the variable section, drag a y variable and put it where the first 0 is. On the other side type -1.

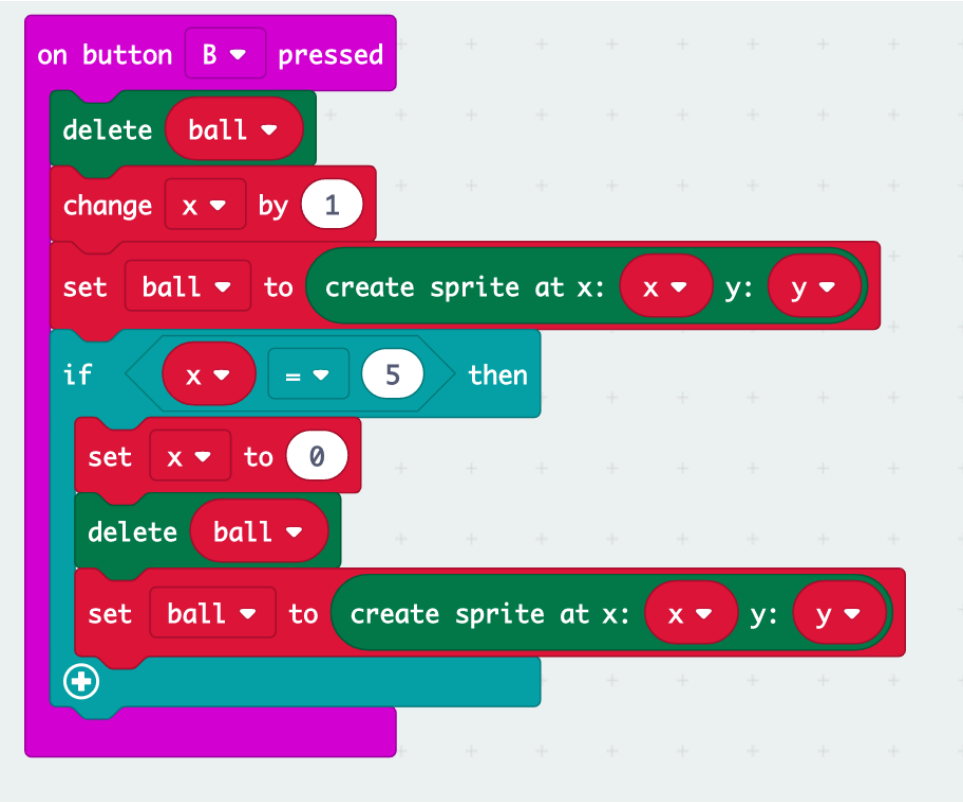


- Inside the "if then" statement, place a "Set variable to 0" block. From the drop down, select y. Change the 0 to 4. Then drag a "delete ball" from the game section. Then another "set variable to" block. Select the ball variable. Then drag a "create sprite at x y" block from the game section and add the x and y variable just like above. It should look like this

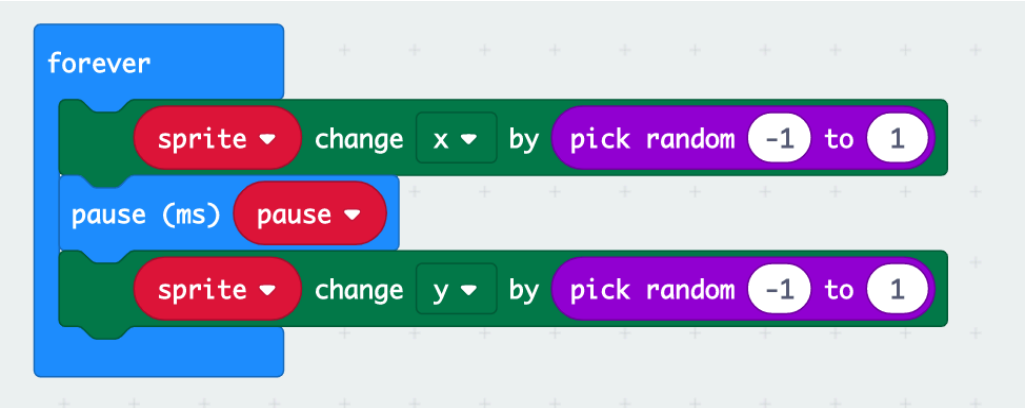


- The if statement checks to see if y=-1. If this is the case, it sets y to 4, deletes the ball, then creates a new one at the new x,y coordinates. The reason why we need to do this is because if y=-1 that means it is no longer on the screen, so we reset it to the bottom of the screen and it can continue to move its way up.
- Next, we will code the B button. Right click on the "on button a pressed" block and select duplicate. This will copy the whole thing!

- Change on button A to on button b pressed. Then change “change y by -1” to “change x by 1”. In the if then statement, change y to x and -1 to 5. Change “set y to 4” to “set x to 5”. It should look like this:

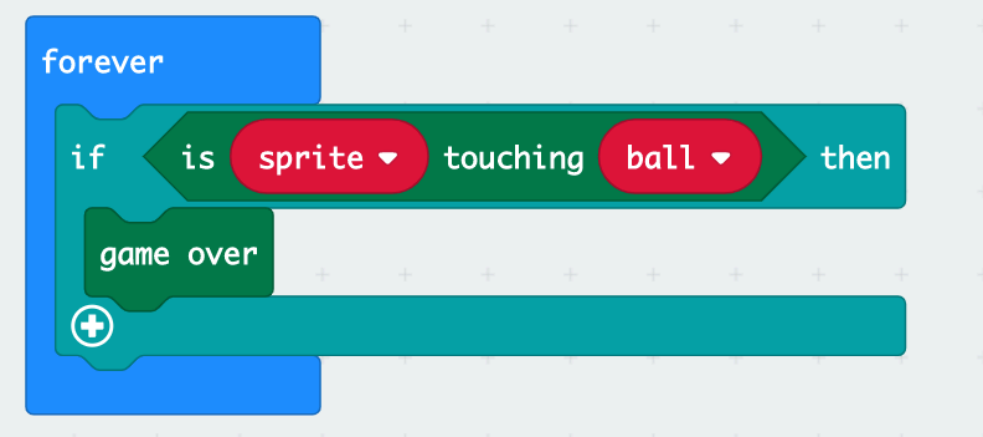


- Similar to on button a pressed, this section moves our character. It moves it to the right. When x=5 it is at the edge of the screen, so we rest it to the left side in the if then statement.
- Now we will code our sprite to spawn in random locations. This will move the enemy around random. Drag in a forever block from the basic section. Add a “sprite change x by 1” block from the game section.
- Then, from the purple math section add a “pick random number 0 to 10” block and put it in the 1 of the previous block. Change the numbers to -1 and 1.
- Add a pause button from the basic section. Drag a “pause” variable into the pause button space.
- Then right click the green block, duplicate it and add it below the pause block. Change the x to y. It should look like this



This randomly changes the location of the sprite in the x axis, followed by the y access, moving it one space at a time, depending on the random number picked. Because it is in the forever block, it will do this forever.

- Lastly, we will code the game over.
- Drag another forever block from the basic section.
- Add an "if true then" block from the logic section. Then add a "if sprite is touching" block from the game section to the true spot in the statement. Drag a ball variable into the blank spot.
- Inside the if then statement, place a game over block from the game section. It should look like this:



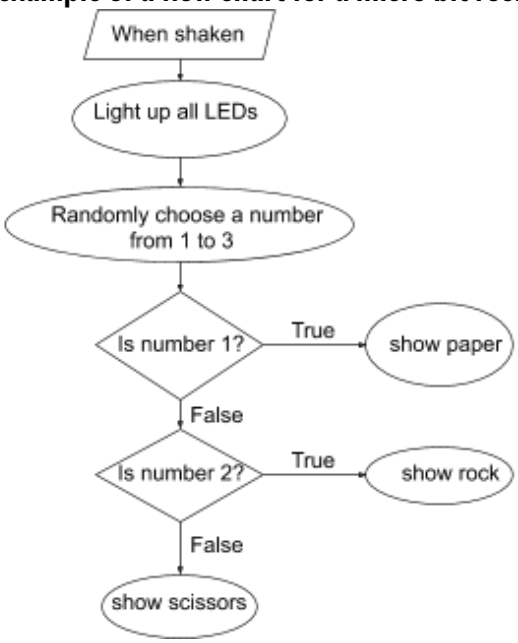
- This section means that once the sprite and ball come into contact, the game ends and it runs the game over sequence.
- That's it! Your game is all done. Download it onto the micro:bit and test it out and play a few times. If you don't have a micro:bit, play it on the example micro:bit on your screen.

What Did You Learn?

- What are some coding languages?
- What role does logic play in code?
- Why are flow charts important in large coding projects?
- What does a forever loop do?

Future Learning

- Here is an example of a flow chart for a micro:bit rock, paper, scissors game:



Based on the flow chart, try to create the code.

- Block coding is a stepping stone to other coding languages. It helps teach you about what code does in an easy-to-use way. At the top of your micro:bit code, there is a button you can click to see your code in JavaScript. Take a look to see everything we just did in a written coding language. Can you identify the things we used? Try changing some of the variables in JavaScript to make your character or the enemy move differently, more quickly, or in larger steps. Try coding your initials to display in JavaScript.

Share your creations!

We would love to see what you made. Email as at discover@uwo.ca or tag us on social media.

Instagram: @westernueng
Twitter: @westernueng
Facebook: @westernueng

Thanks for discovering with us!