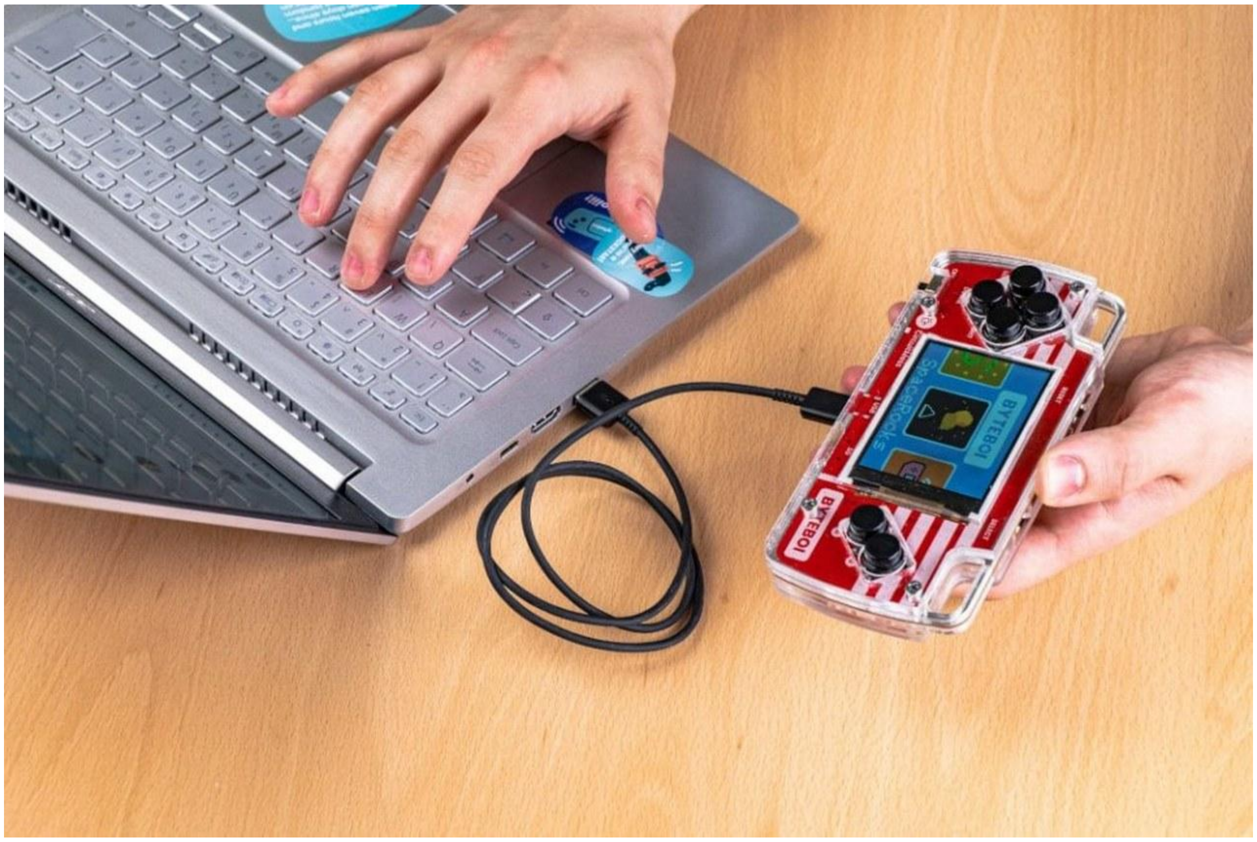
Coding for beginners - how to code your ByteBoi

Let's get down to business!

First, you need to connect your ByteBoi to your computer's USB port and turn it on.



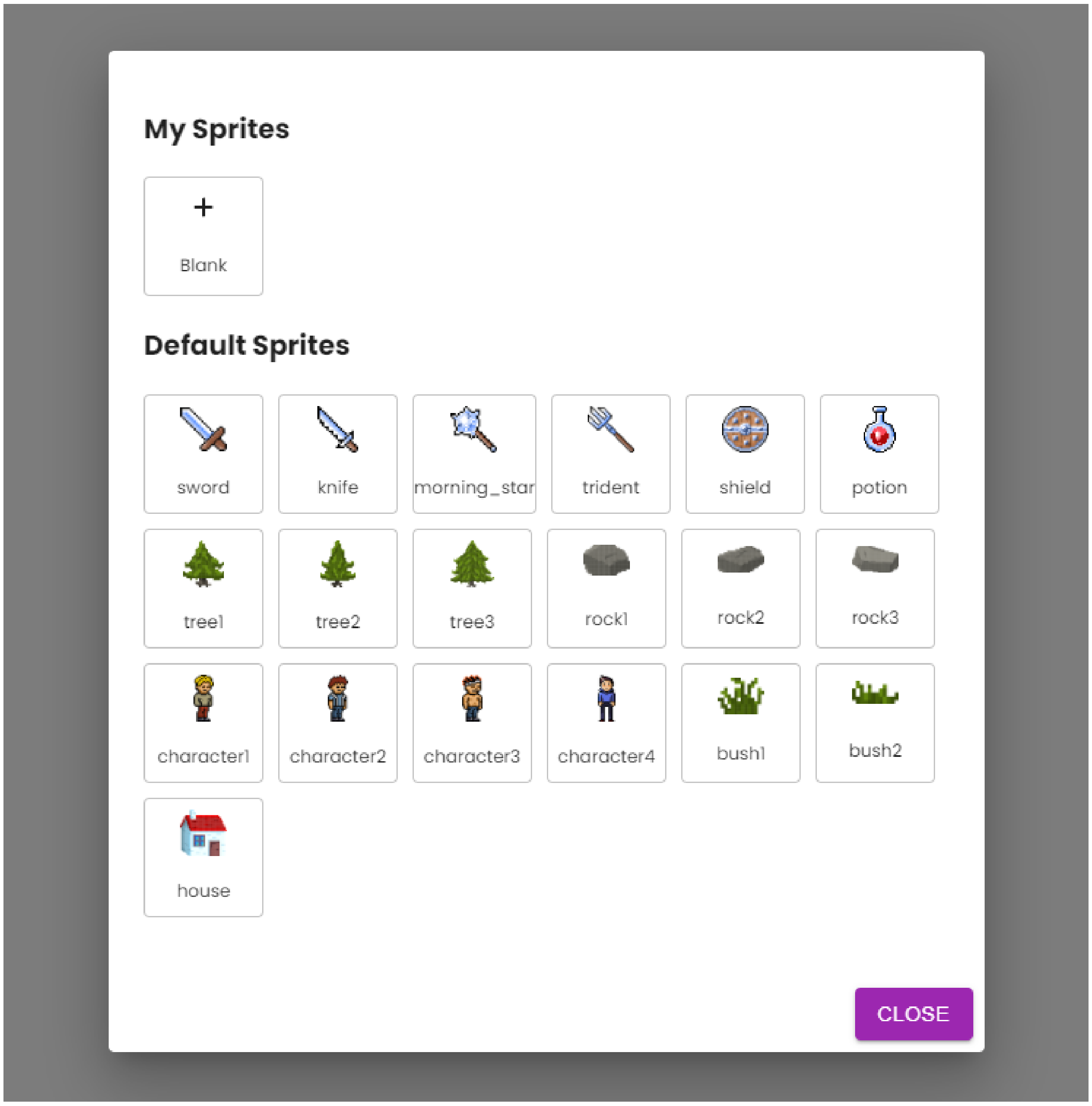
LET'S DRAW SOMETHING!

The first thing we're going to learn is how to use the feature called Sprite editor!

In computer graphics, a sprite is a two-dimensional image or animation that is integrated into a larger scene.

Find the three dots on your Toolbar first, click on it and find the Sprite editor.

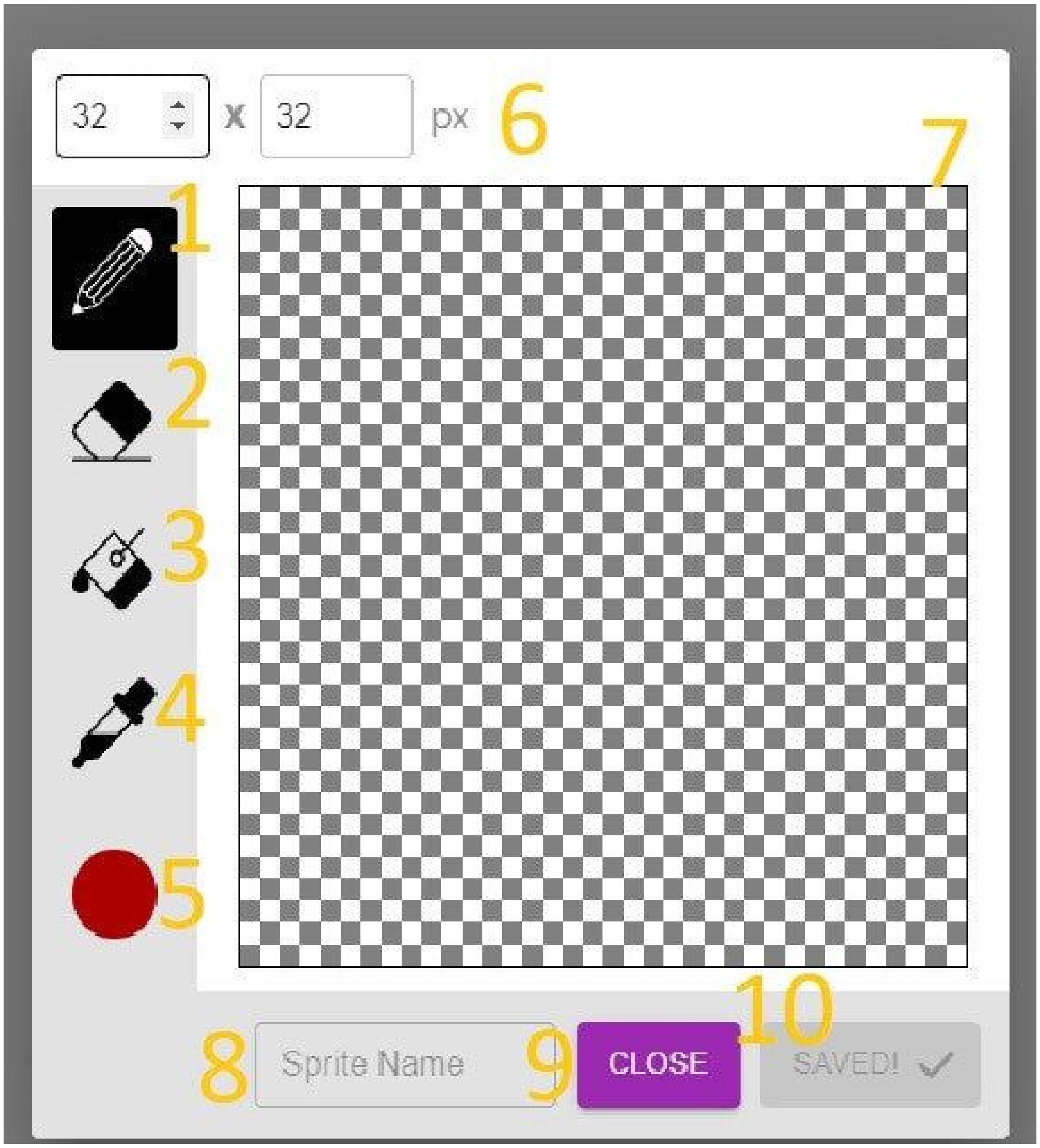
This window will open:



As you can see, you have the option to choose one of the premade sprites or create your own by clicking on "blank".

Let's explore the blank option first.

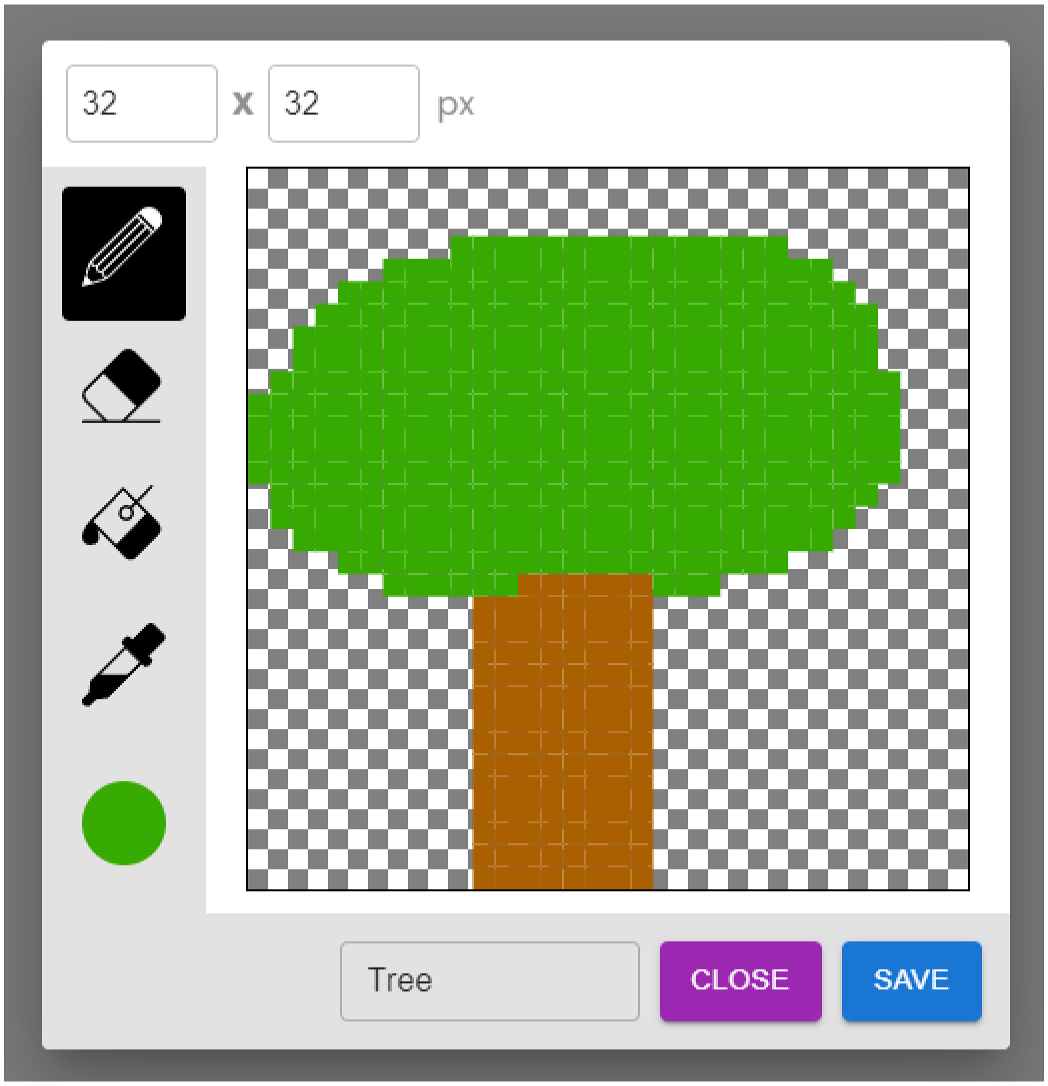
Check what each icon means.



- Paintbrush - you'll use it for painting whatever you want.
- Eraser - you'll use it to erase mistakes you make.

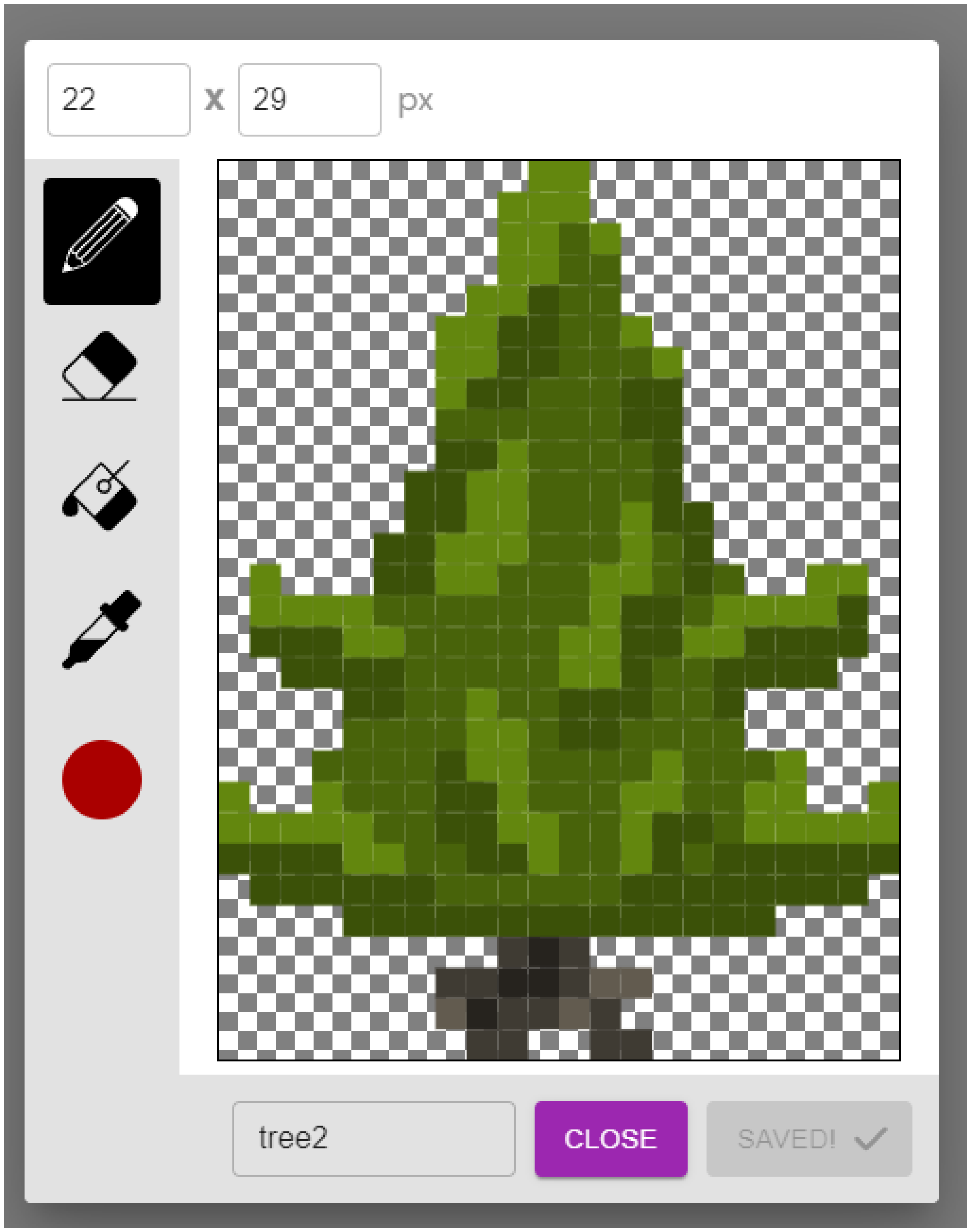
- Paint bucket - choose any color you want, click on the drawing area, and paint the entire background.
- Color dropper - this tool samples colors from anywhere you click and adds them to your sprite.
- Color picker - here, you can choose which color you will use for drawing.
- Dimensions - choose the dimensions of your sprite.
- Drawing area - once you're done with the drawing, press the big blue button saying "Close".
- Sprite name - Write any name you want to give your sprite.
- Close - Close without saving.
- Save - For saving your sprite.

You can paint whatever you want, and later on, you can put it on your ByteBoi's display.



If you don't want to draw, but you'd like to have some cool images on your ByteBoi, you can always choose a pre-made one.

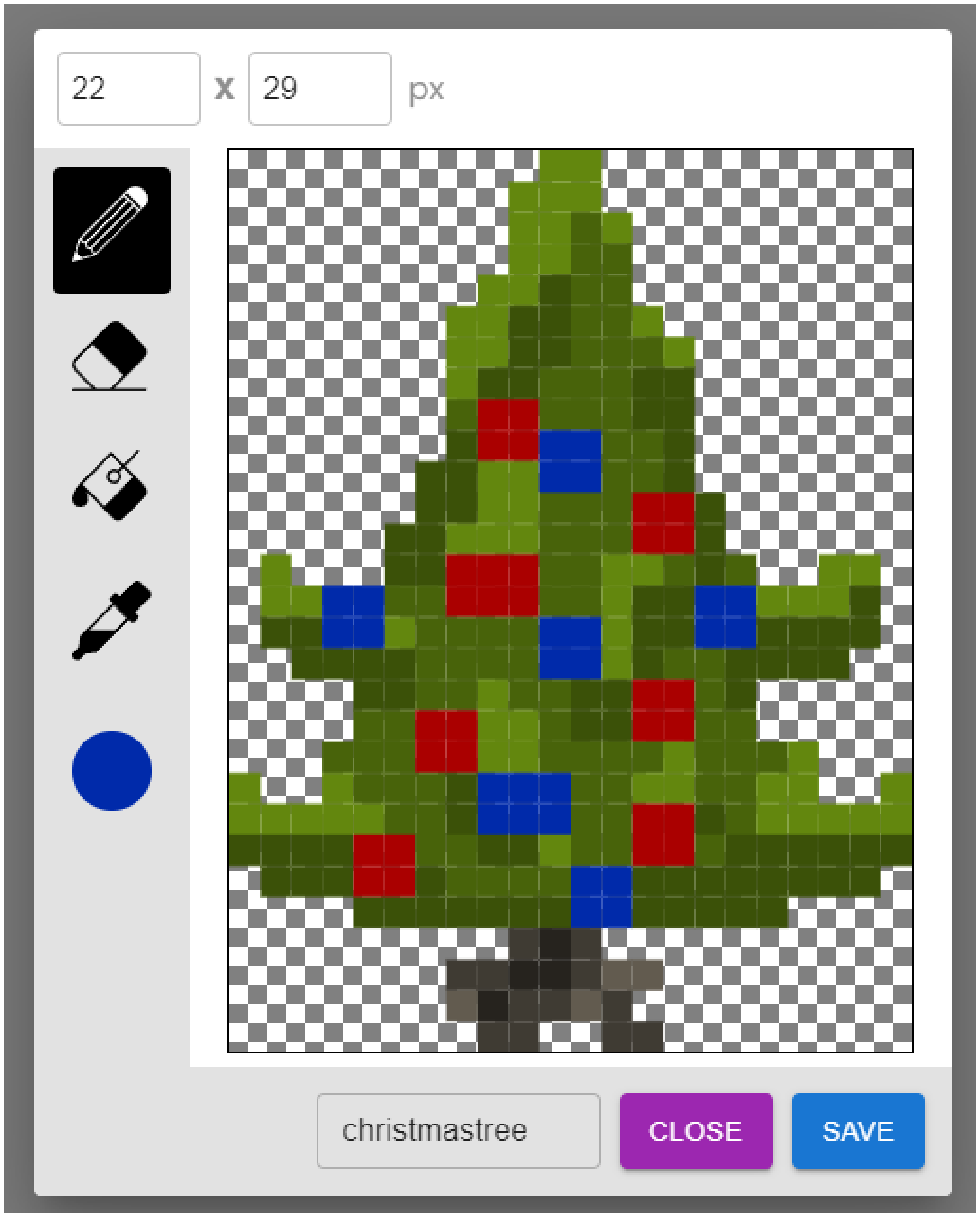
Maybe you want to draw on the pre-made icons and make them better. No problem, just click on the one you want to upgrade and start drawing!



For example, I chose to draw a Christmas tree but didn't want to draw a tree from scratch. So, I took a premade tree and drew Christmas ornaments.

Also, I didn't want my tree to have a generic name, so I renamed it to christmastree.

If you want to rename your drawing, you should know that its name cannot begin with a number, and there can't be any spaces between words.



One more tip is you can change the resolution of your sprite in the upper left corner.

The resolution is the size of a sprite, and it's measured in pixels.

A pixel is the smallest controllable element of a picture represented on the screen.

You can play with this some more, but now we're going to code for real!

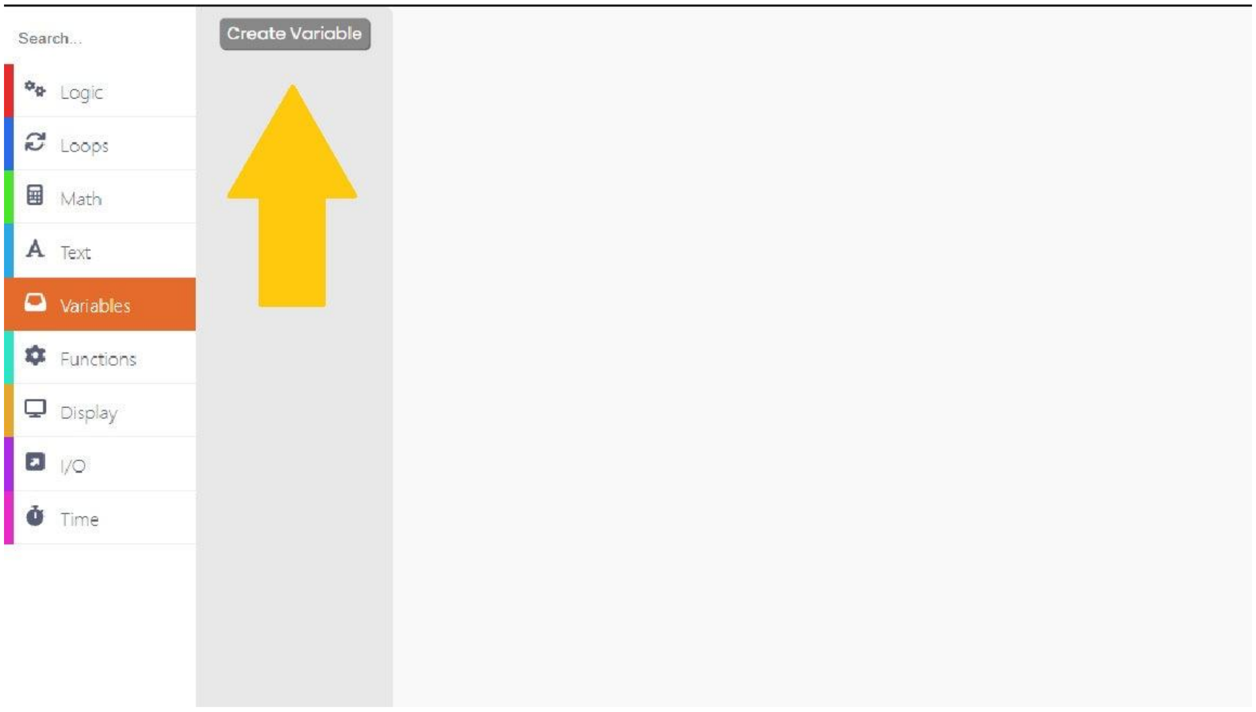
Let's play with the display!

For the next step, let's go a bit further and do something fun on display.

The first thing we are going to use are *variables.*

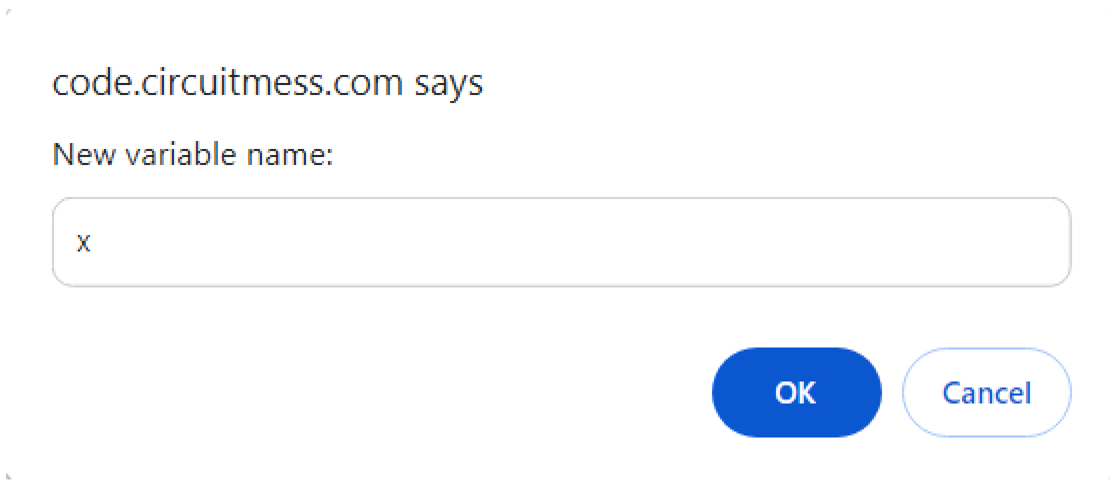
In computer programming, a variable is a storage location that contains a value. Every variable has a specific name. You can store and change the values of a variable.

Firstly, let's create a variable. Find the section named "Variables" and press the "Create variable..." button.



You need to give your variable a name.

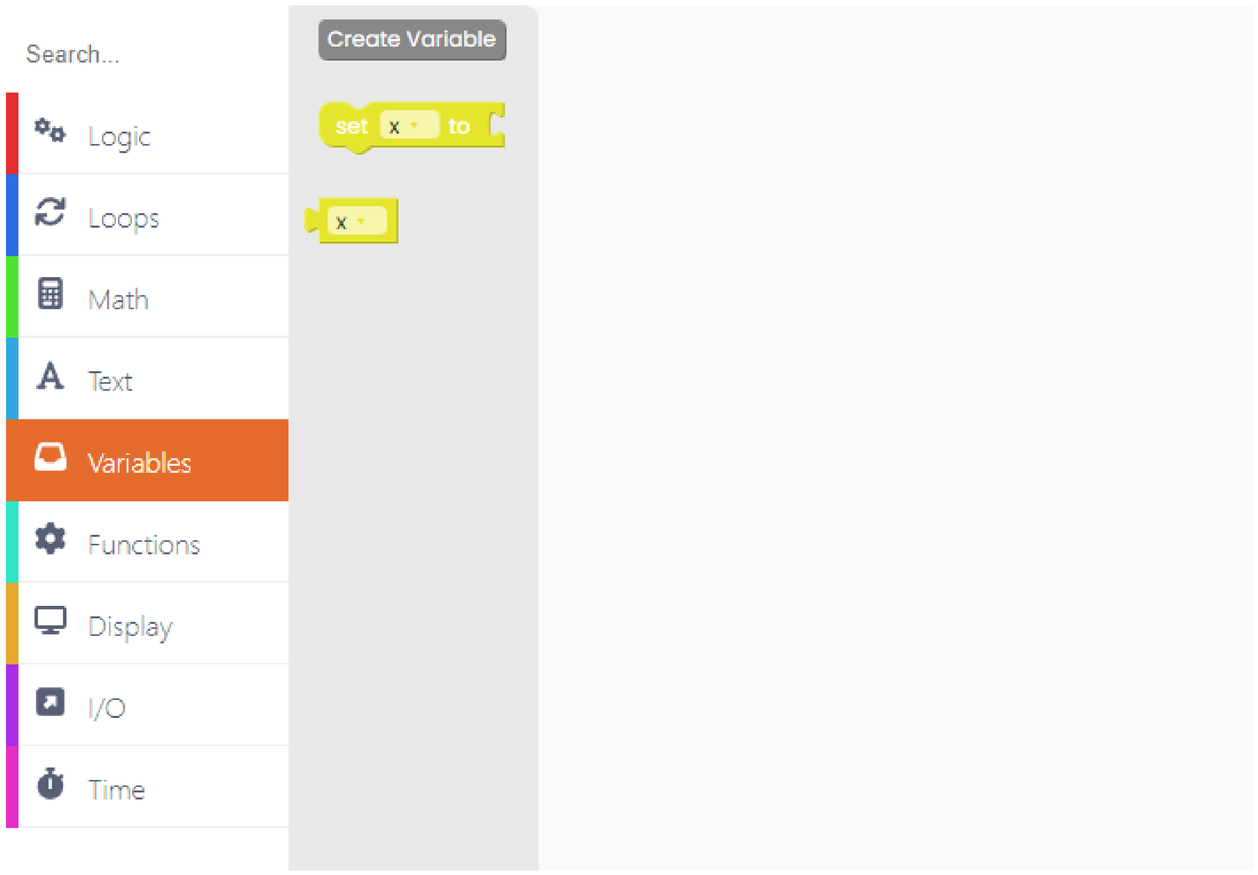
I am not that creative, so I will name my variable "x" (just a single letter x).


code.circuitmess.com says

New variable name:

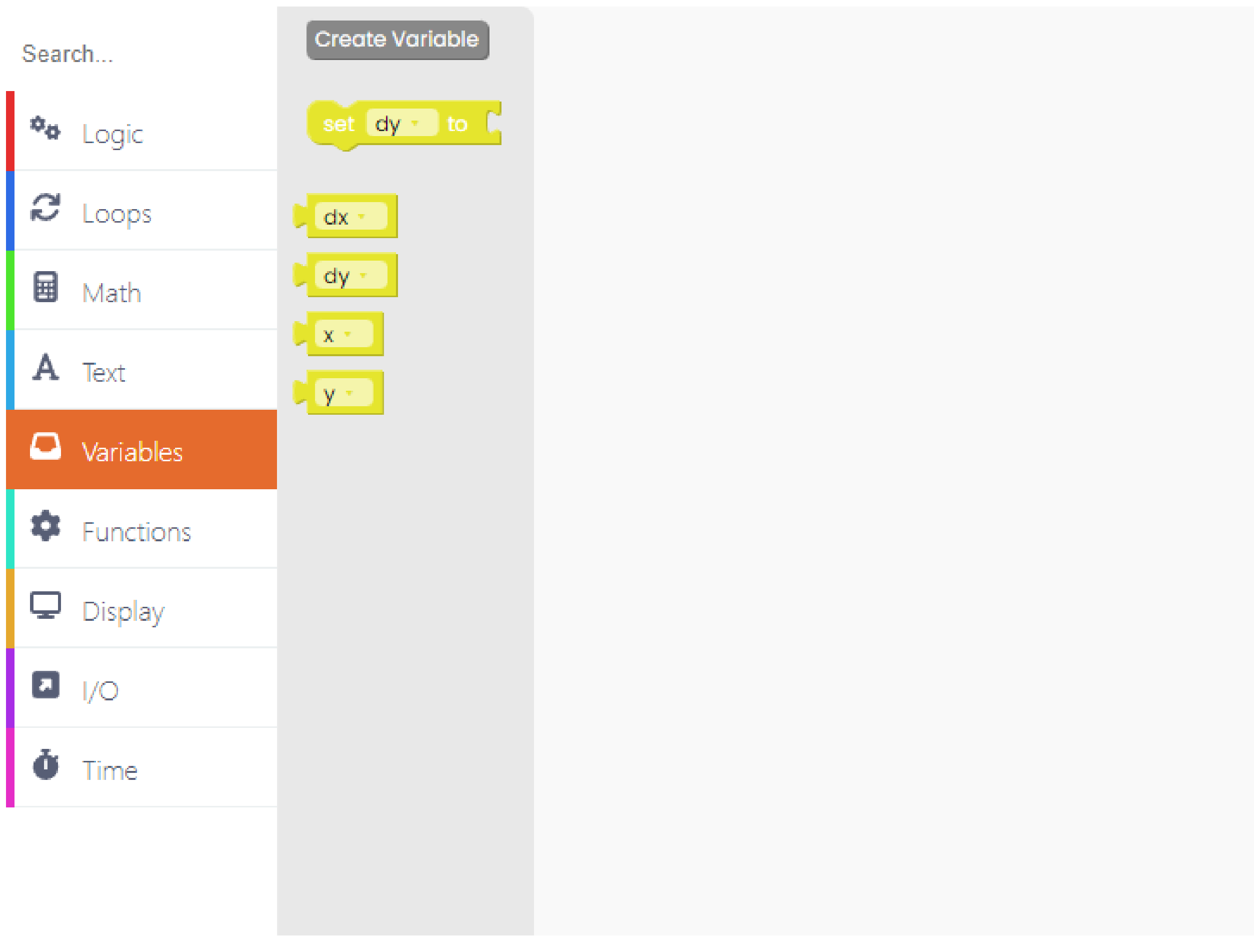
x

We have a variable now. Great!



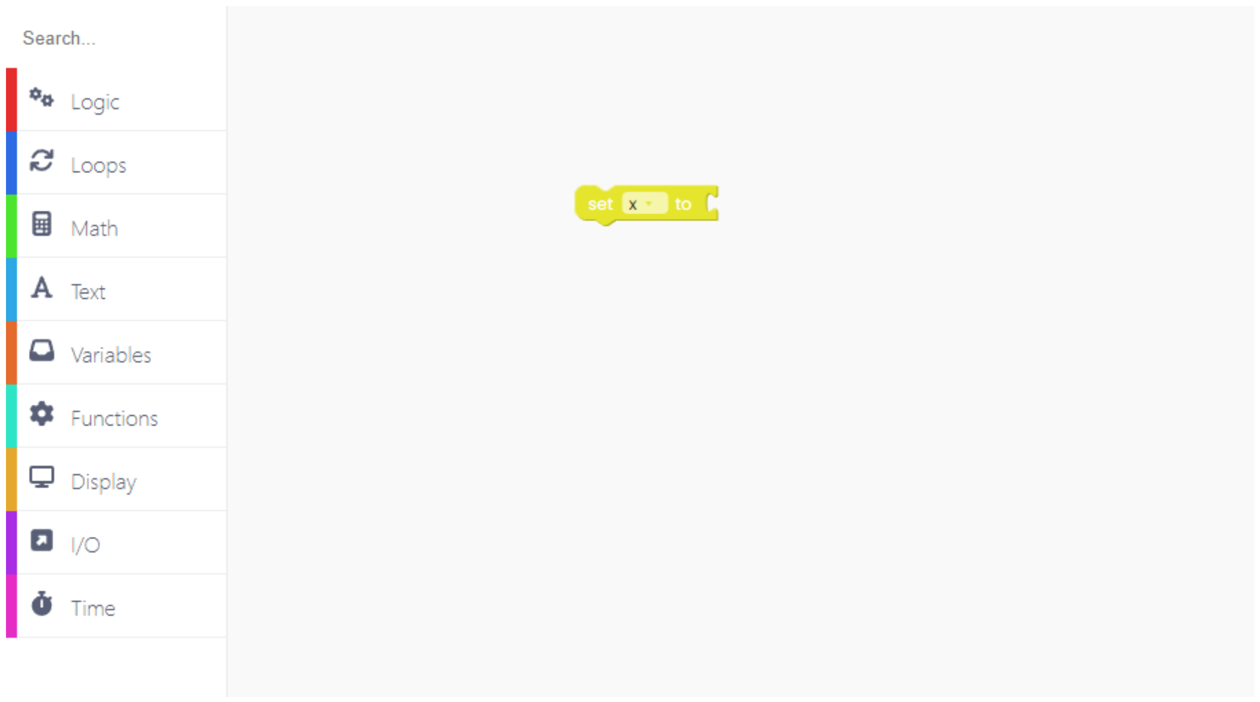
We can make all the variables we will need for this sketch immediately.

Just repeat the same process you did while making the variable "x".



When we create a variable, it's undefined - it has no value. We must set a value for every variable when our computer program starts. That's why you'll need the "set variable" block.

Take the "set x variable" block and drop it into the drawing area.

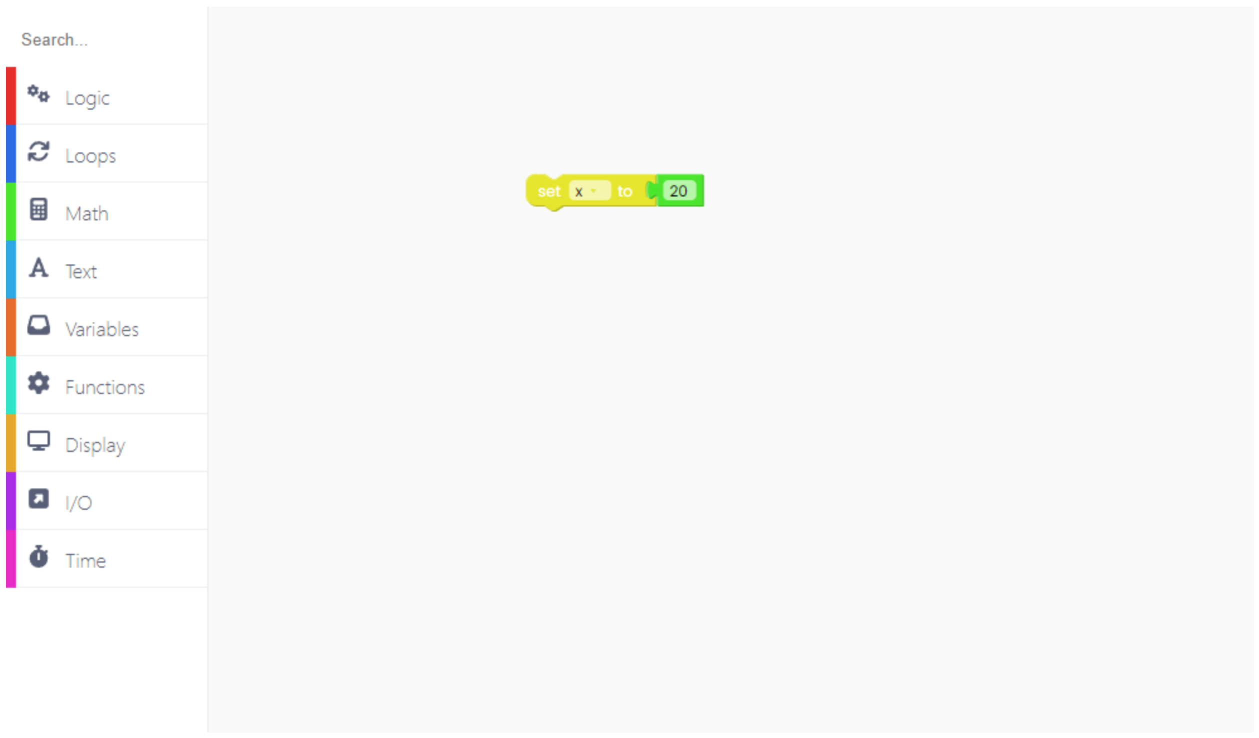


You need to define the value to which you want to set the variable.

Find the block named "123" in the "Math" section. This block is a numerical value block, and you can type in the numerical value you want once you drop it onto the drawing area.

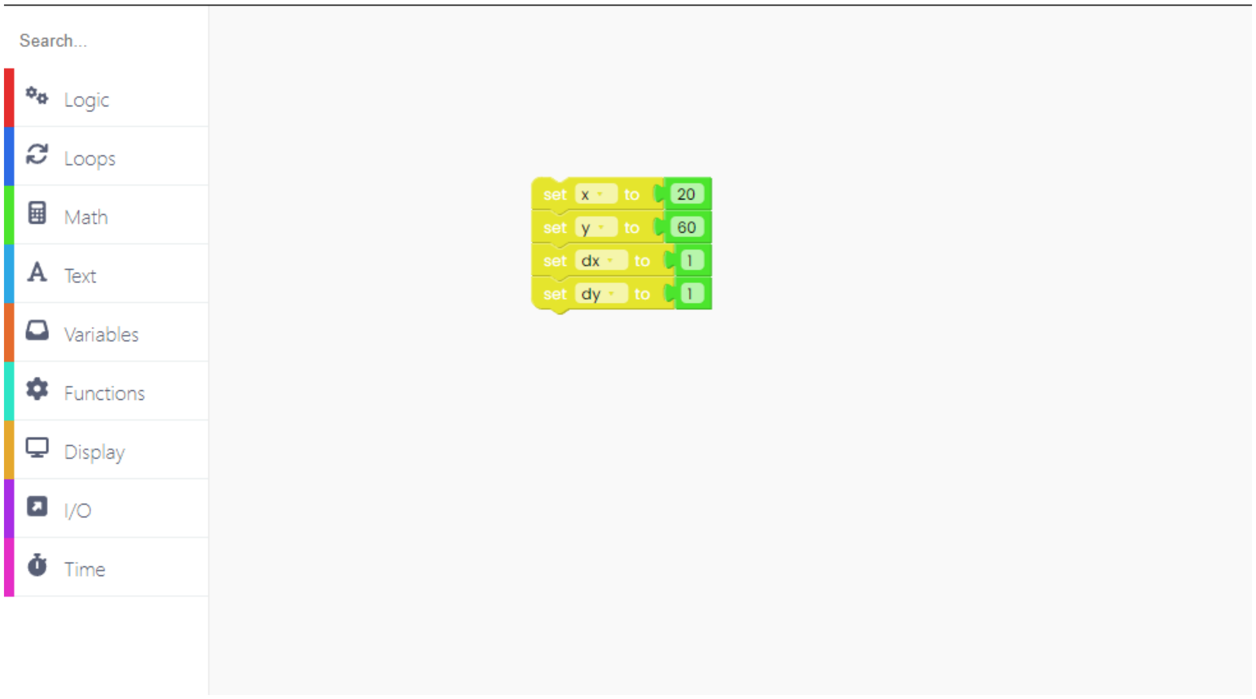
Place the block as shown in the photo below.

Now click on the block and type in the value. Set the numerical value of the block to 20. You can do this by simply typing the number 20 on your keyboard.

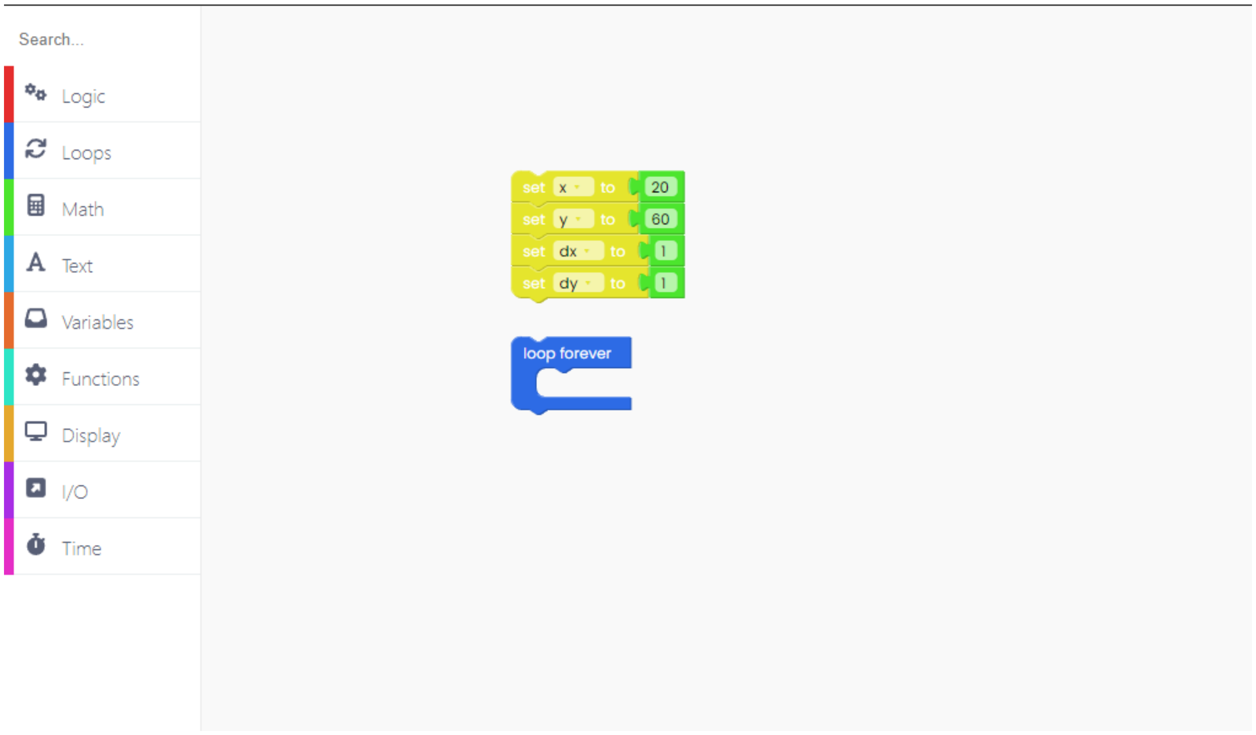


Now, repeat this process for every other variable you make. But make sure to add the correct numerical value for each variable.

By doing this, we defined the position of the ball on the screen.



Look for the "loop forever" block in the "Loops" block section.

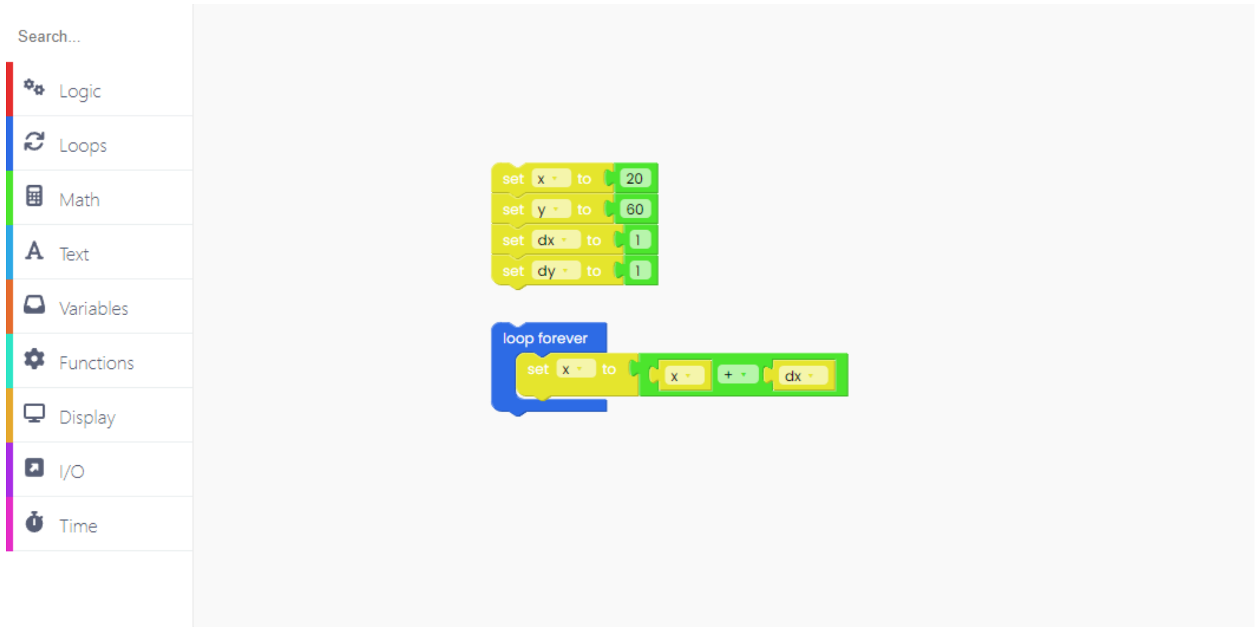


Using this block, we make sure the background of our screen is black once we run the sketch.

Now, we should control the ball's motion.

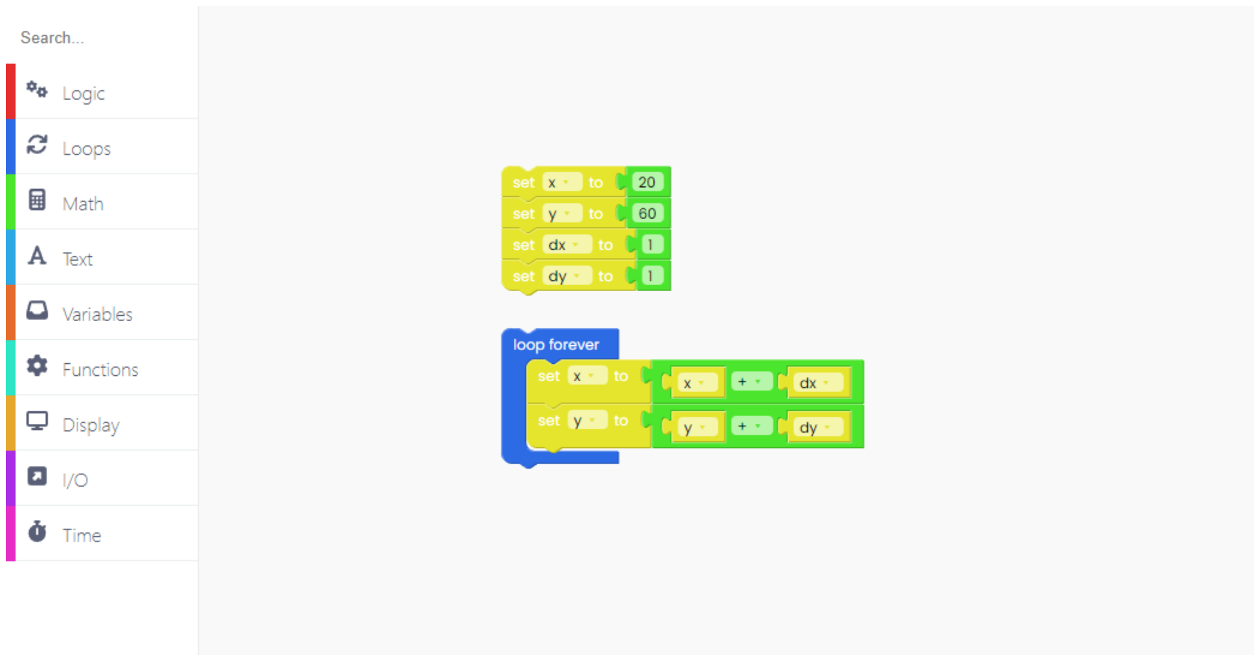
Firstly, go to Variables, and choose a block called "set x to".

In addition to that block, add the math block in which we'll add the value of 'dx' to the value of 'x'.



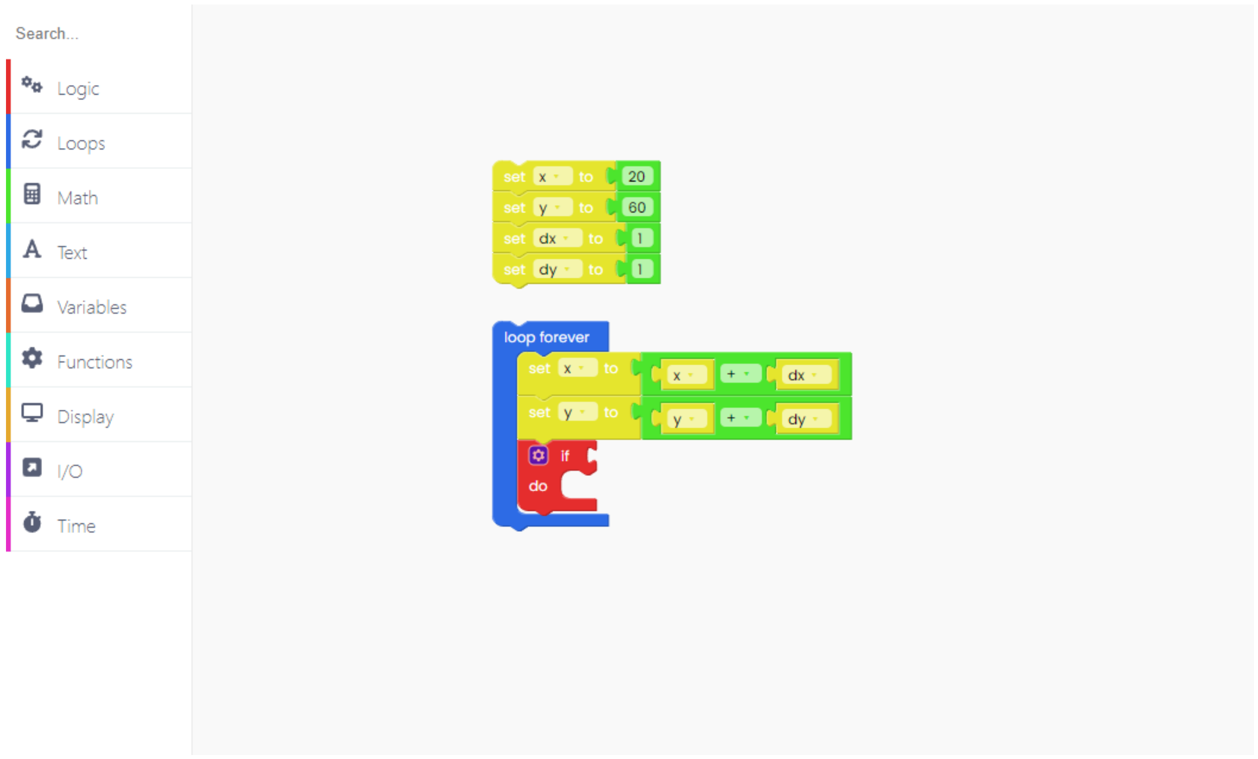
```
set x to 20
set y to 60
set dx to 1
set dy to 1

loop forever
    set x to x + dx
```

Repeat this for the variable 'y' as well.



Let's move on to the "Logic" section!

Drag the if-do block below.

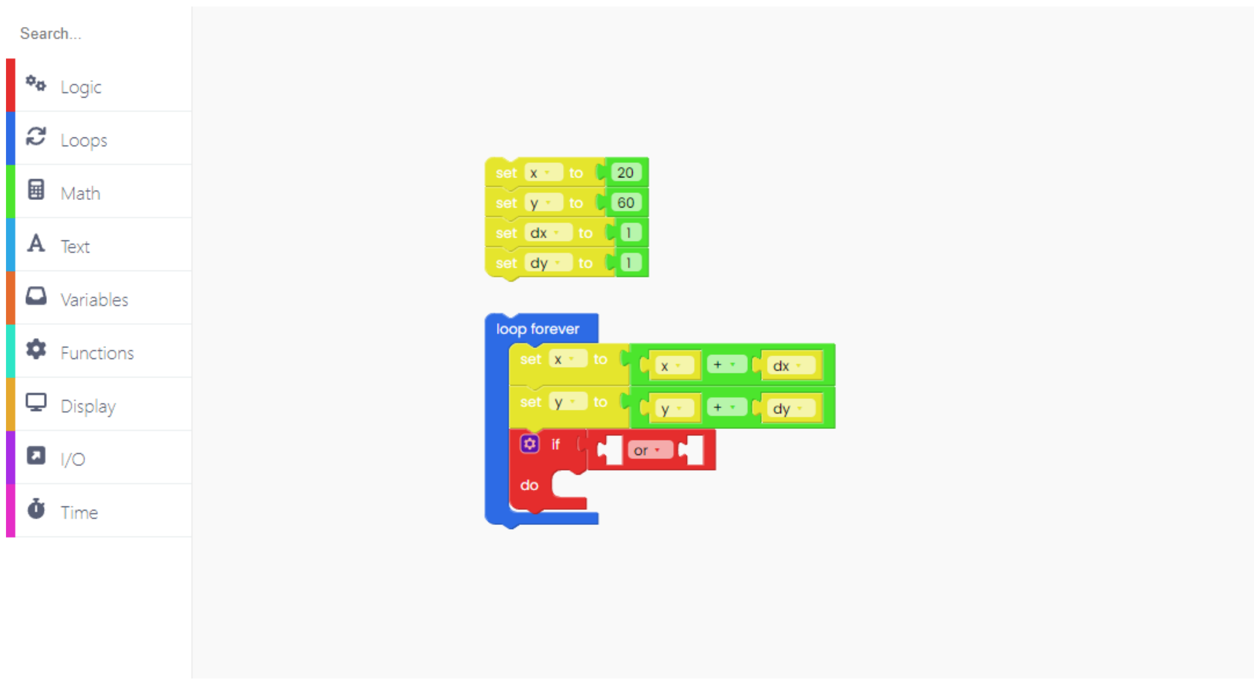


set x to 20
set y to 60
set dx to 1
set dy to 1

loop forever
set x to x + dx
set y to y + dy
if
do

Up next, we'll need a Boolean block - the one that says "or".

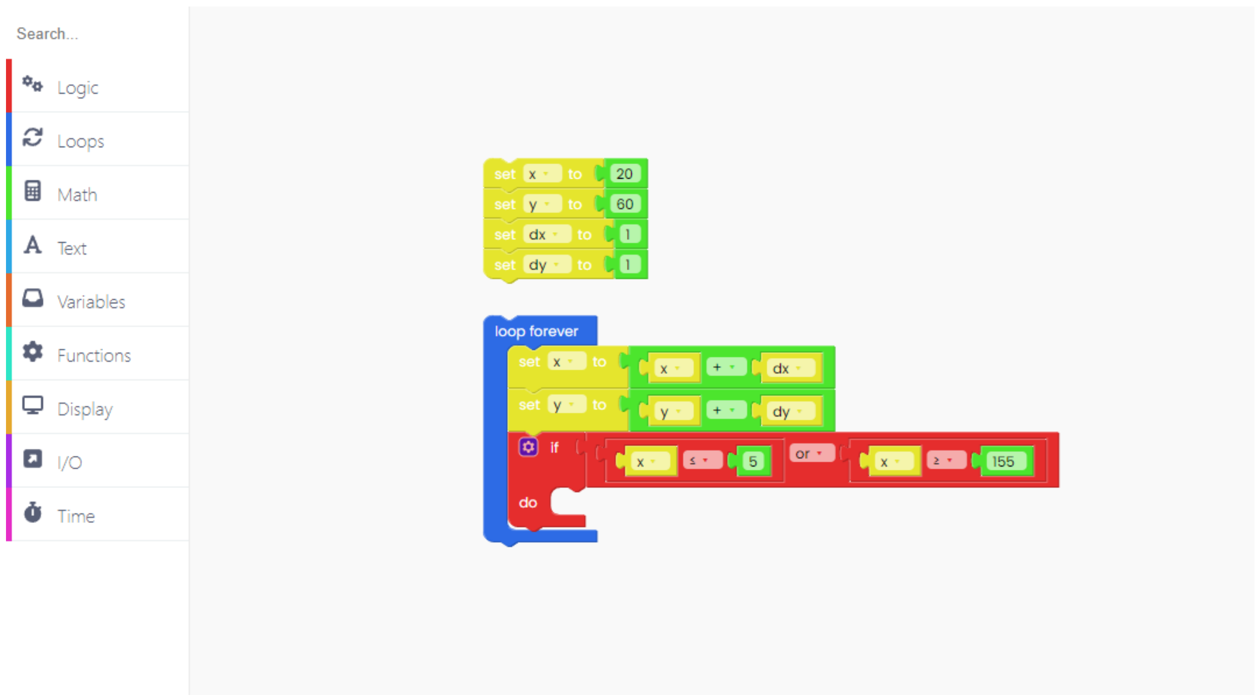
Boolean is a binary variable that can have one of two possible values, 0 (false) or 1 (true).



Now we'll need a comparison block. This block is usually used for comparing the value of a variable to a fixed value (i.e., let's see if the variable x is less than or equal to number 5).

Place the variable we've created on the left side of the comparison block.

Take the numerical value block from the math section and place it on the right side of the comparison block.

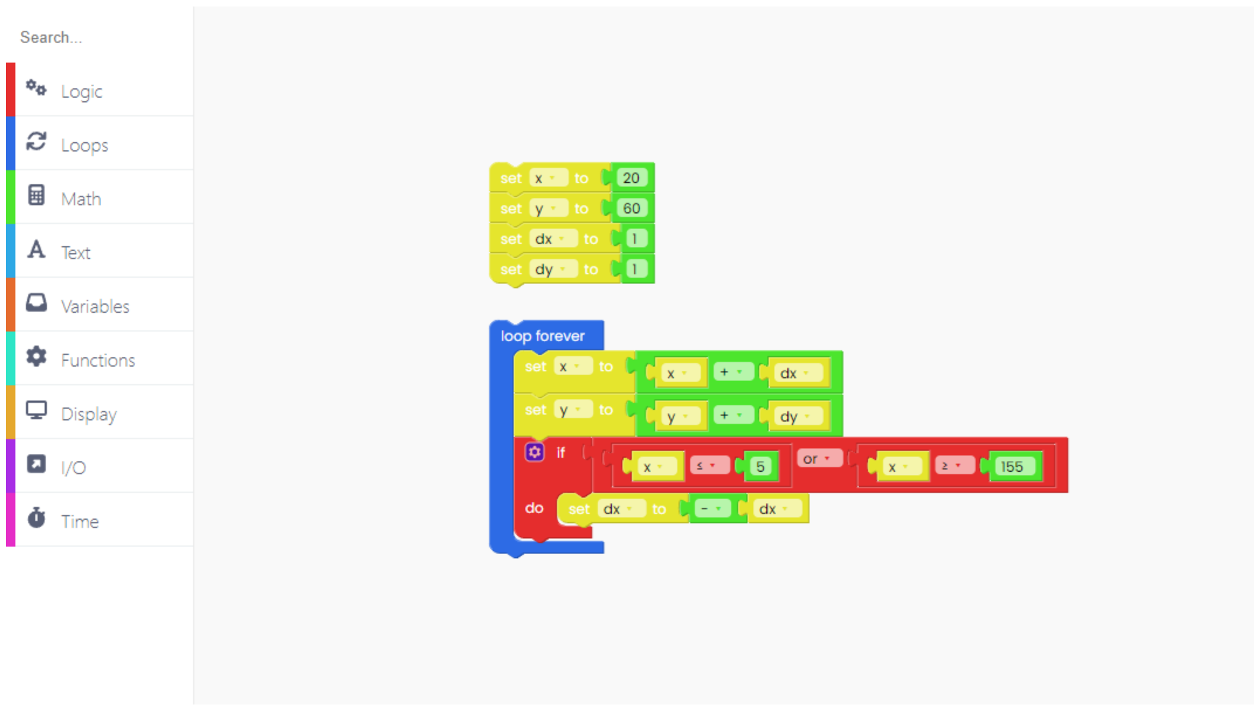


```
set x to 20
set y to 60
set dx to 1
set dy to 1

loop forever
    set x to x + dx
    set y to y + dy
    if x ≤ 5 or x ≥ 155
    do
```

Also, we have to define a numerical value of the "dx" variable, so we need to take a "set dx to" block.

Find single operand blocks in the Math section.

An operand is part of a computer instruction that specifies what data is to be manipulated or operated on.

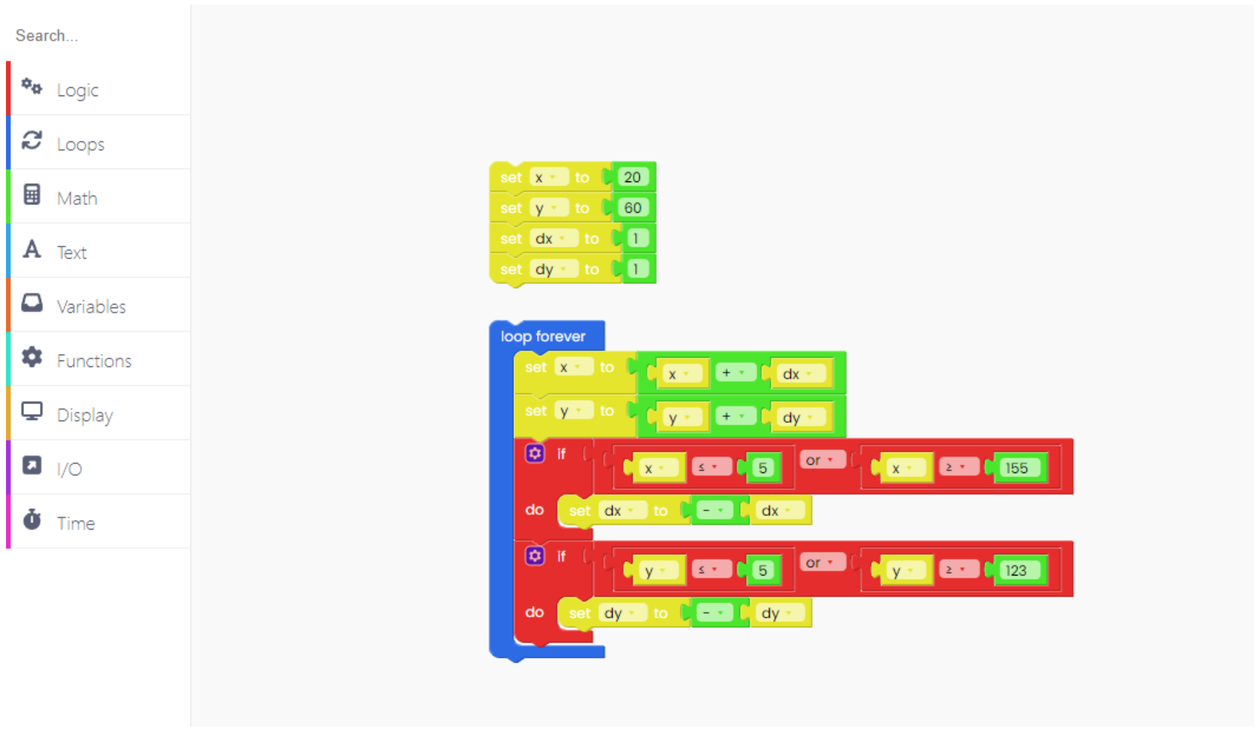


```
set x to 20
set y to 60
set dx to 1
set dy to 1

loop forever
  set x to x + dx
  set y to y + dy
  if x ≤ 5 or x ≥ 155
  do set dx to - dx
```

Now you'll have to duplicate the last couple of blocks. You can do that by clicking on the right button on your mouse and choosing "duplicate".

We'll need the second group of blocks for the "y" variable.

Check out the values we put for the "y" variable:

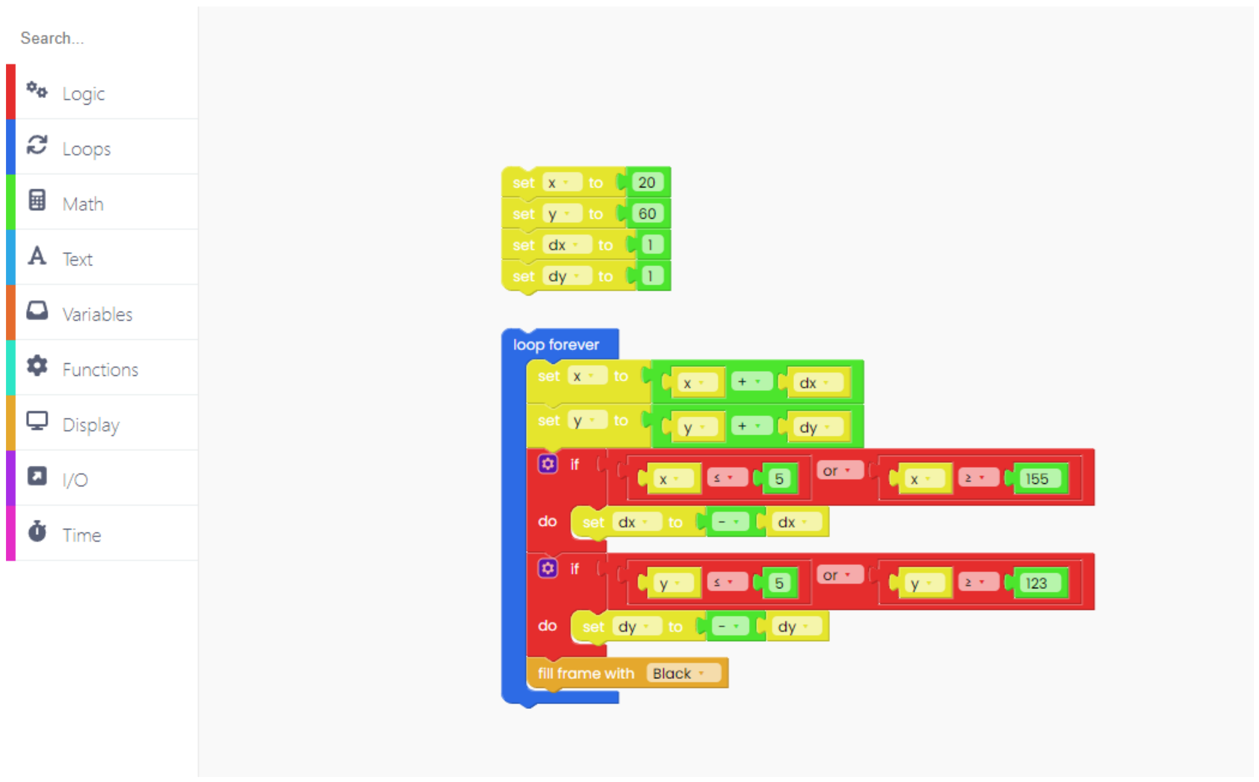


```
set x to 20
set y to 60
set dx to 1
set dy to 1

loop forever
    set x to x + dx
    set y to y + dy
    if x ≤ 5 or x ≥ 155
    do set dx to - dx
    if y ≤ 5 or y ≥ 123
    do set dy to - dy
```

The hard part is done!

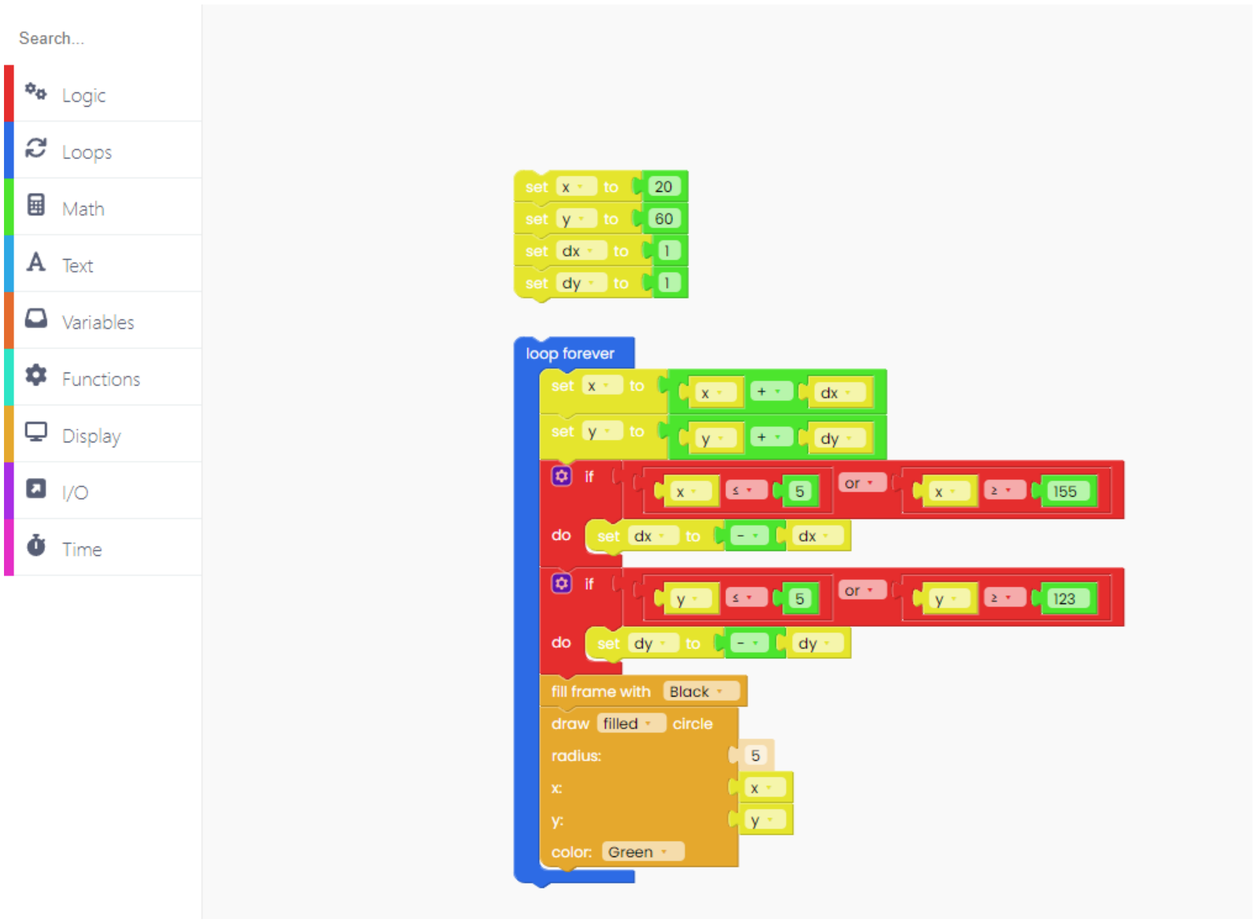
Let's draw the ball that will appear on the screen.

Before drawing anything on a screen, we want to fill the screen with one particular color, and for that, we'll need this block:



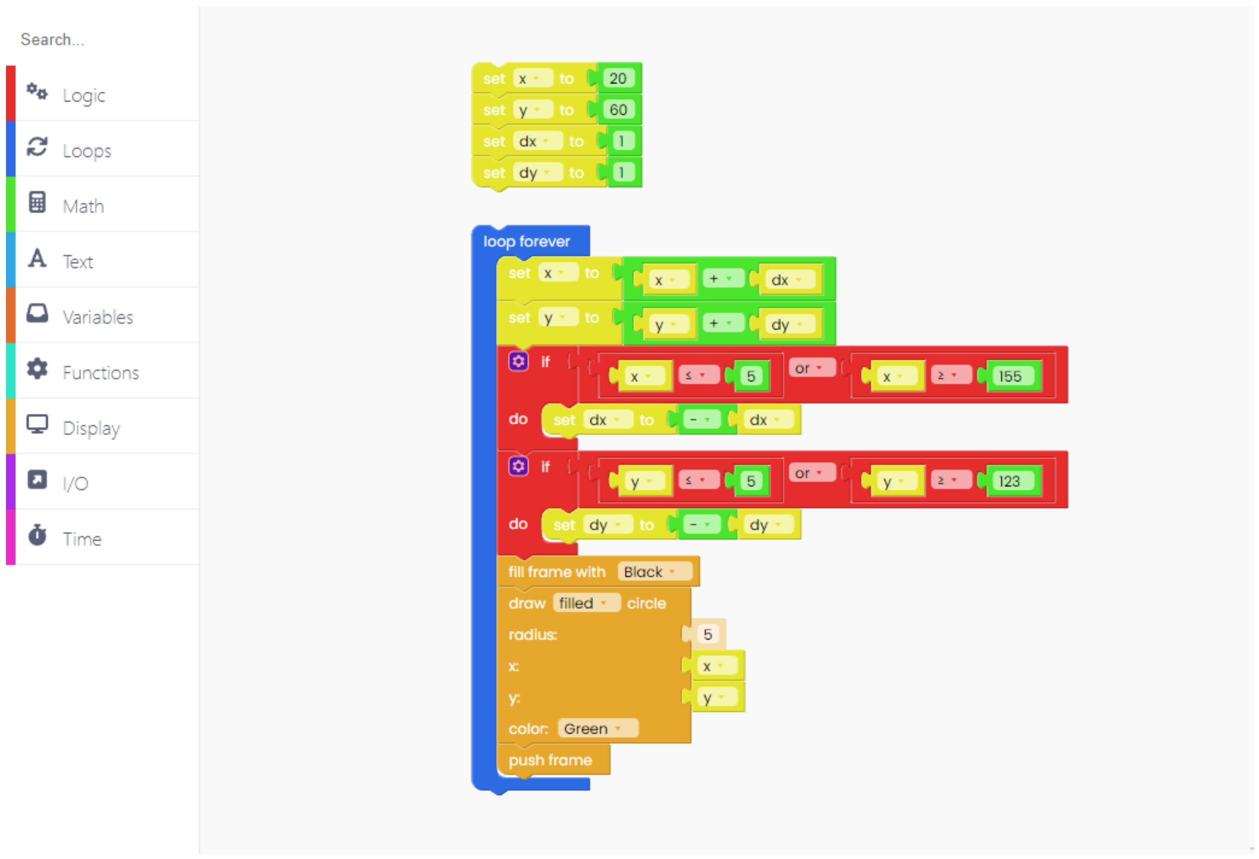
```
set x to 20
set y to 60
set dx to 1
set dy to 1

loop forever
    set x to x + dx
    set y to y + dy
    if x ≤ 5 or x ≥ 155
    do set dx to - dx
    if y ≤ 5 or y ≥ 123
    do set dy to - dy
    fill frame with Black
```

Now, let's draw the ball.



```
set x to 20
set y to 60
set dx to 1
set dy to 1

loop forever
    set x to x + dx
    set y to y + dy
    if x ≤ 5 or x ≥ 155
    do set dx to - dx
    if y ≤ 5 or y ≥ 123
    do set dy to - dy
    fill frame with Black
    draw filled circle
    radius: 5
    x: x
    y: y
    color: Green
```

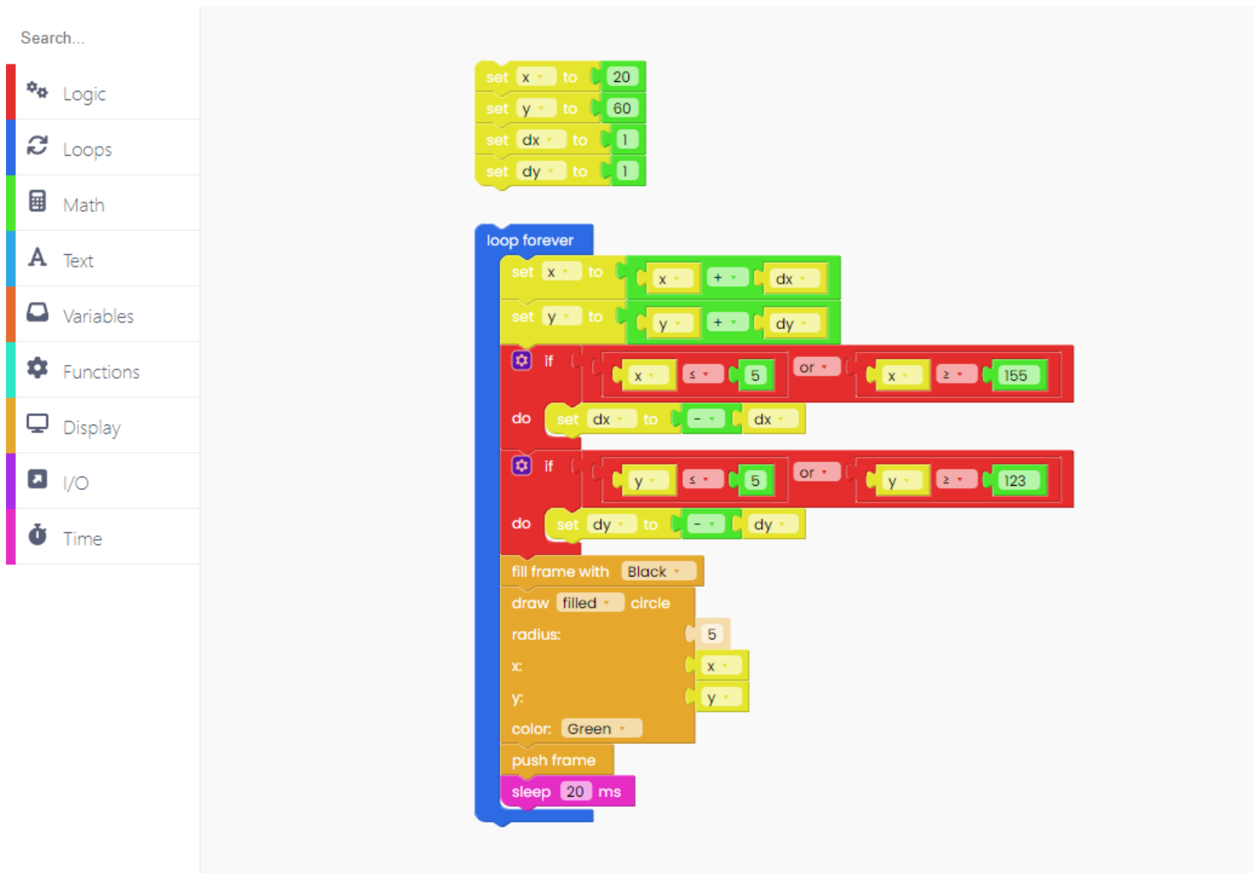
For the circle to appear on a screen, we have to add the "push frame" block at the end.



```
set x to 20
set y to 60
set dx to 1
set dy to 1

loop forever
    set x to x + dx
    set y to y + dy
    if x ≤ 5 or x ≥ 155
    do  set dx to - dx
    if y ≤ 5 or y ≥ 123
    do  set dy to - dy
    fill frame with Black
    draw filled circle
        radius: 5
        x: x
        y: y
        color: Green
    push frame
```

Finally, add the time block labeled "sleep 0 ms". Change the zero into 20.

This block determines the amount of time that'll pass before the code runs again.



```
set x to 20
set y to 60
set dx to 1
set dy to 1

loop forever
    set x to x + dx
    set y to y + dy
    if x ≤ 5 or x ≥ 155
    do set dx to - dx
    if y ≤ 5 or y ≥ 123
    do set dy to - dy
    fill frame with Black
    draw filled circle
    radius: 5
    x: x
    y: y
    color: Green
    push frame
    sleep 20 ms
```

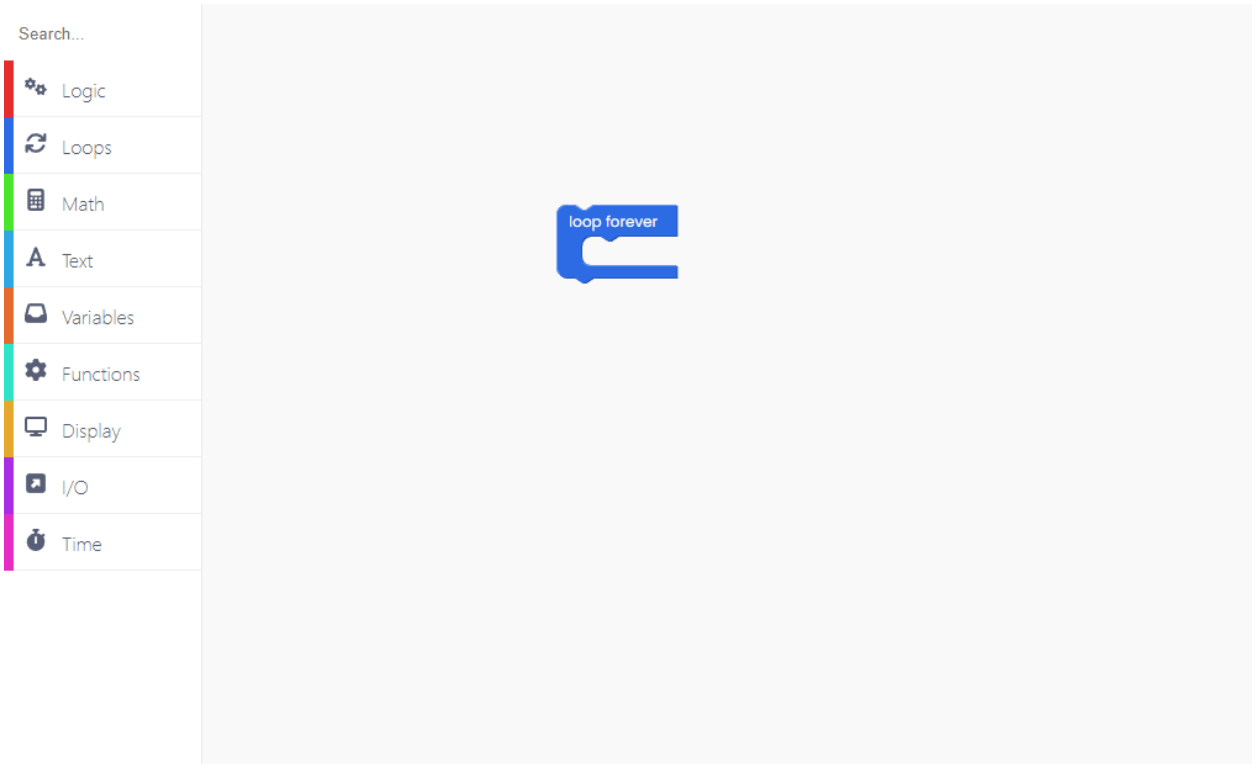
Great job!

Click on the Run button and check the code.

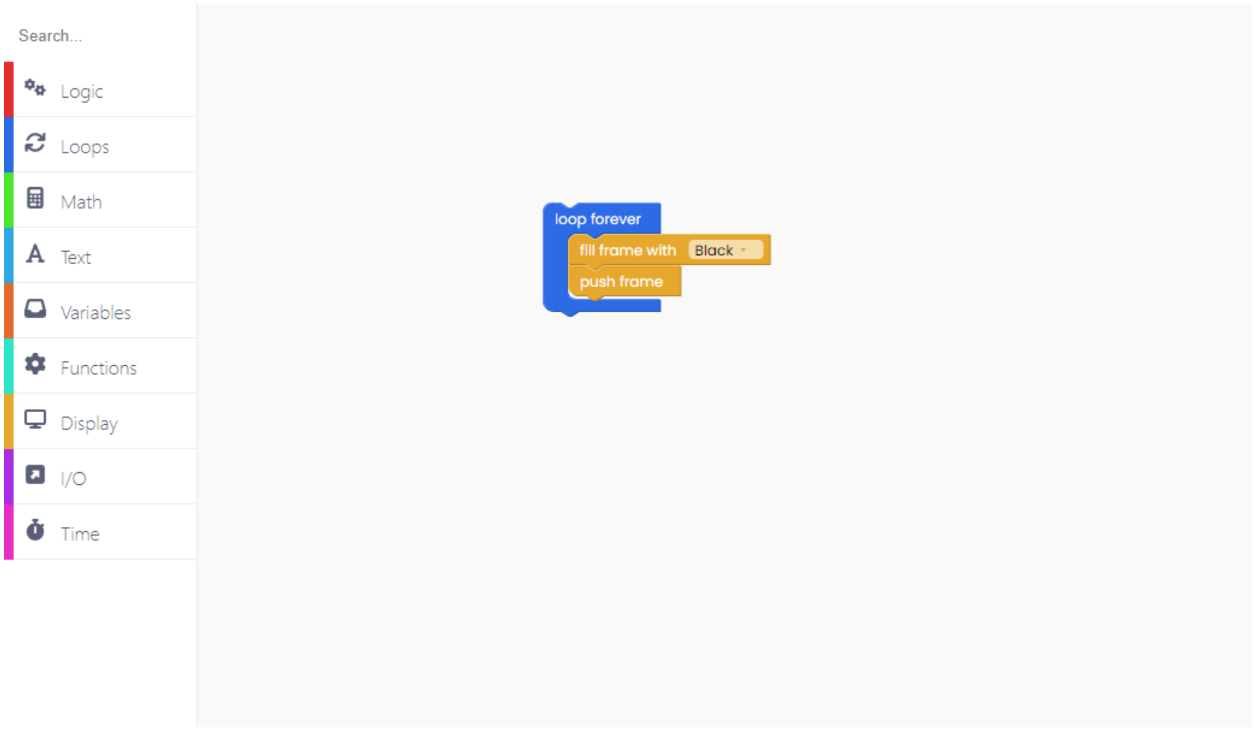
Simple timer

This chapter will teach us how to change the color of the display every second.

Begin by looking for the "loop forever" block in the "Loops" block section.

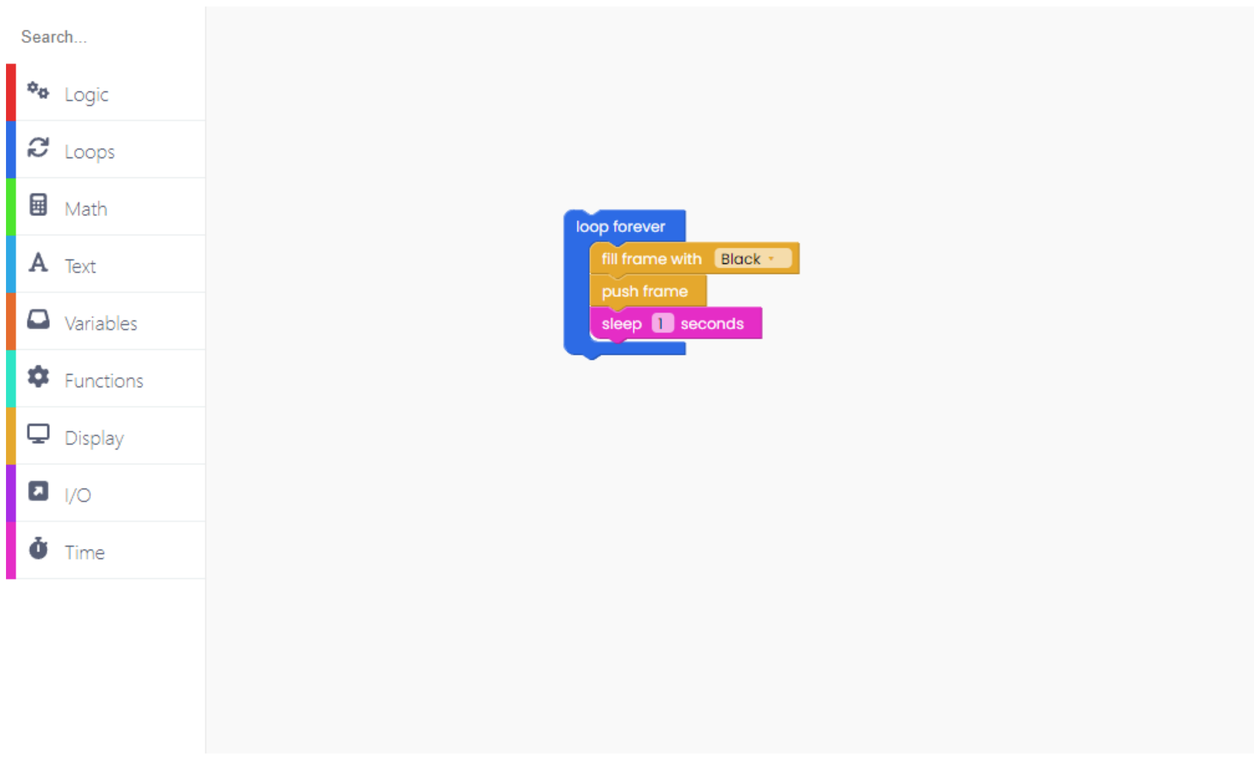


To change the color of the screen, we'll need the "fill frame with..." and "push frame" blocks from the "Display" block section.

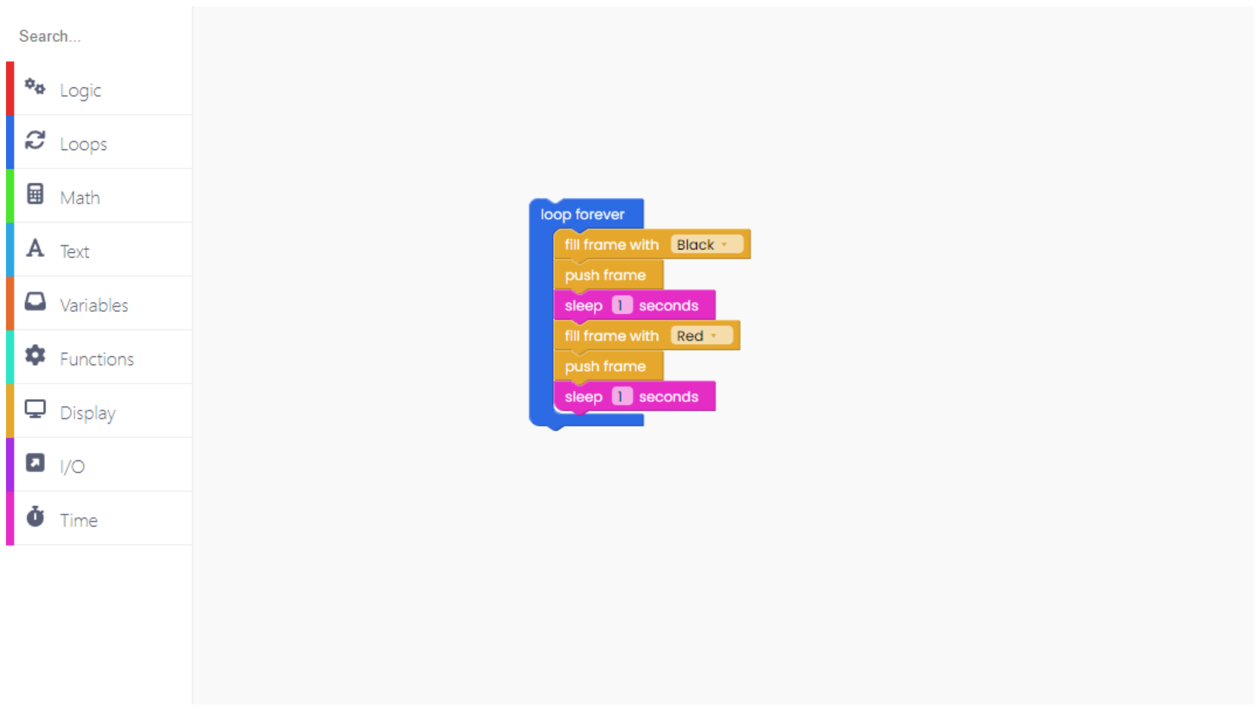


Let's pause for 1 second and then change the color of the screen again.

For the pause, look for the time block labeled "sleep 1 seconds".

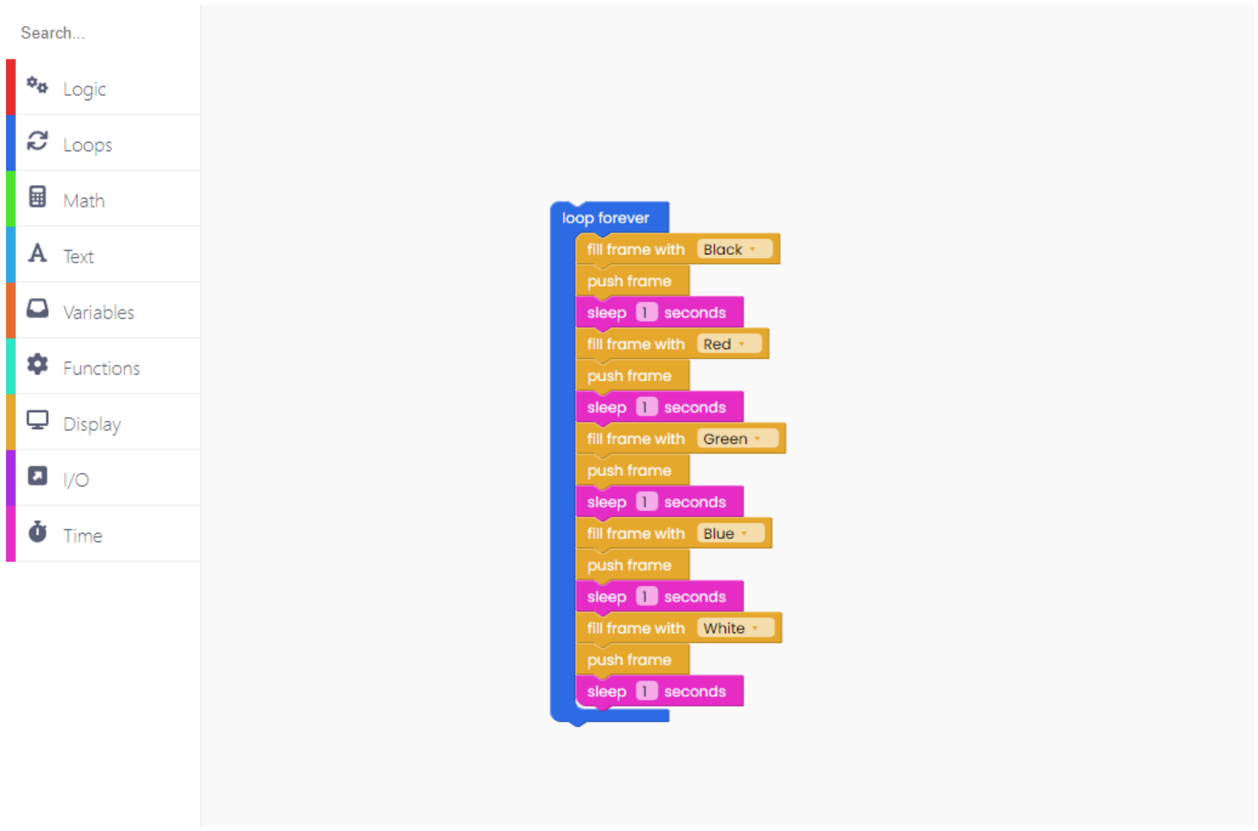


Now, we'll change the color of the screen into red.



```
loop forever
    fill frame with Black
    push frame
    sleep 1 seconds
    fill frame with Red
    push frame
    sleep 1 seconds
```

You can play with as many colors as you want!



```
loop forever
    fill frame with Black
    push frame
    sleep 1 seconds
    fill frame with Red
    push frame
    sleep 1 seconds
    fill frame with Green
    push frame
    sleep 1 seconds
    fill frame with Blue
    push frame
    sleep 1 seconds
    fill frame with White
    push frame
    sleep 1 seconds
```

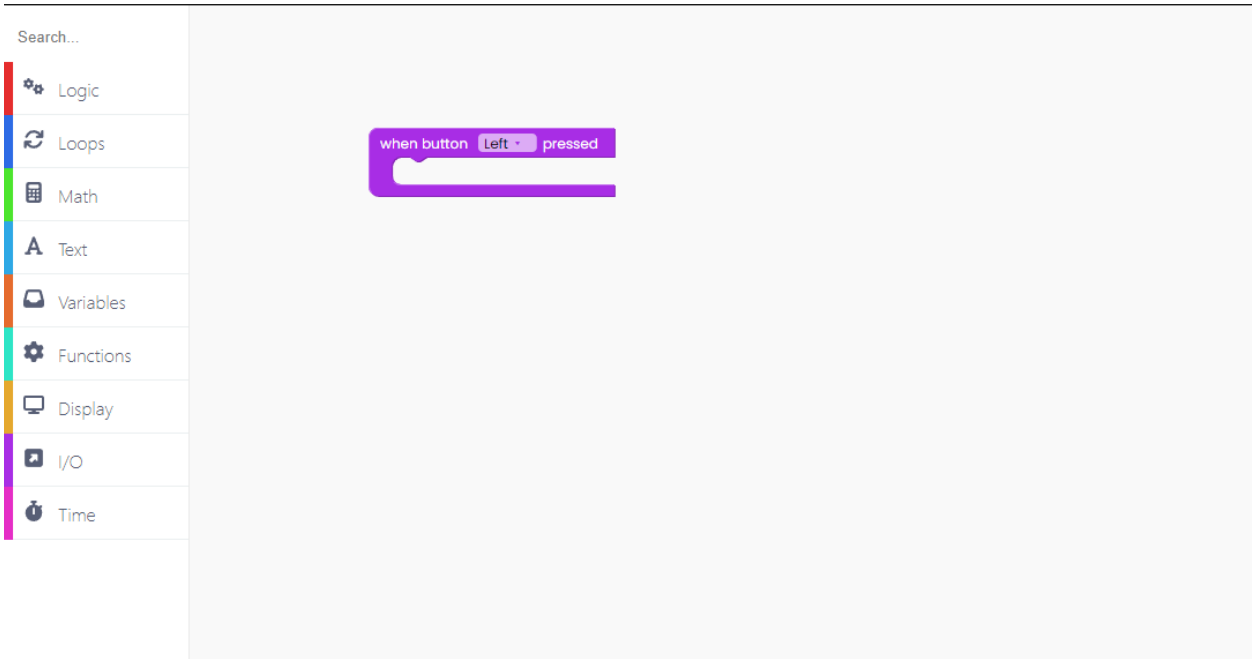
After finishing the code, click on the Run button, and check the colors changing.

Buttons

In this chapter, we'll learn how to change the color of the screen and write different words by clicking on different buttons.

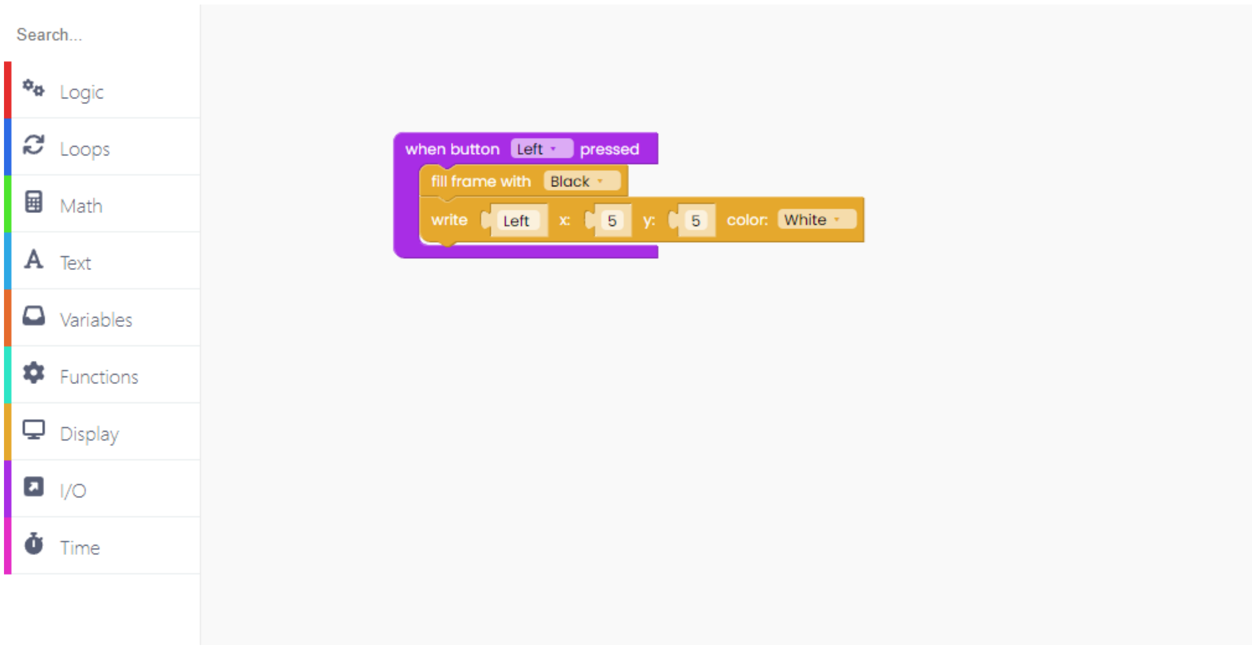
For coding anything with buttons, you need an I/O block section.

Find the "when button left pressed" in the I/O block section and drop it into the drawing area.



When we press this particular button, we want the screen to turn black, and for the word "left" to appear on it.

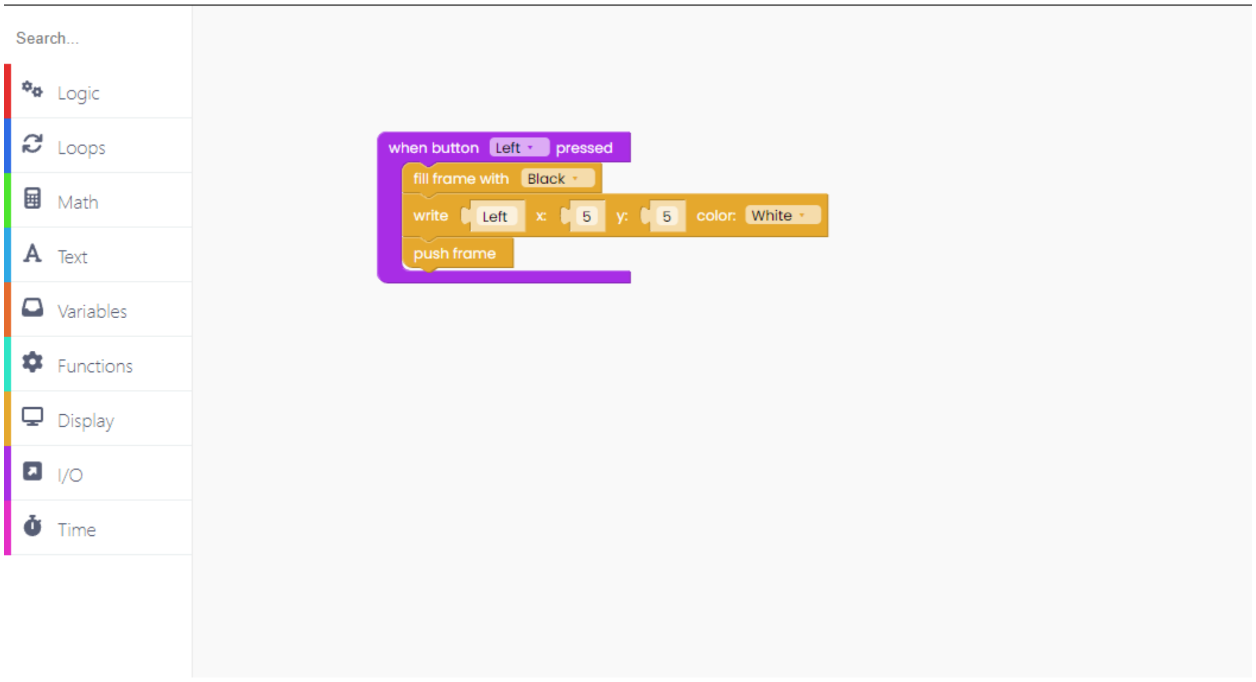
To do so, create this block:



The 'X' and 'Y' coordinates in the block indicate where the word will show on the screen.

Feel free to choose any color you want for the background of the screen or for the word. We'll use black for the background and white for the word in this example.

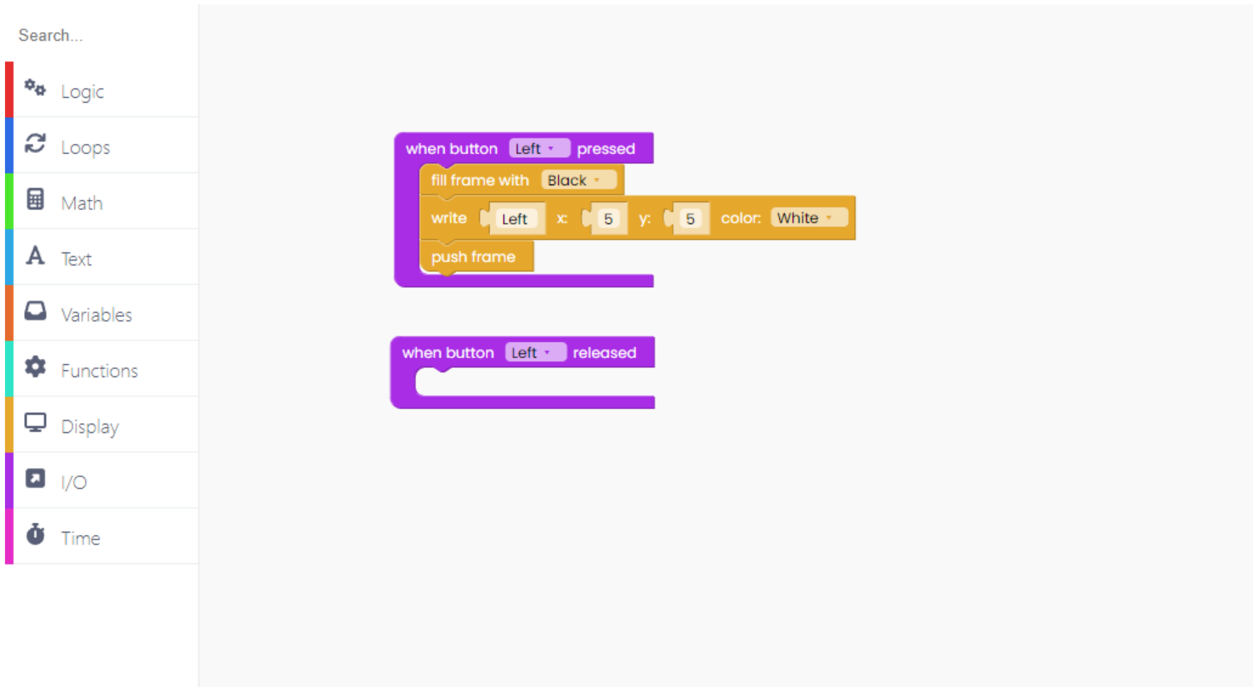
Don't forget the "push frame" block at the end.



```
when button Left pressed
  fill frame with Black
  write Left x: 5 y: 5 color: White
  push frame
```

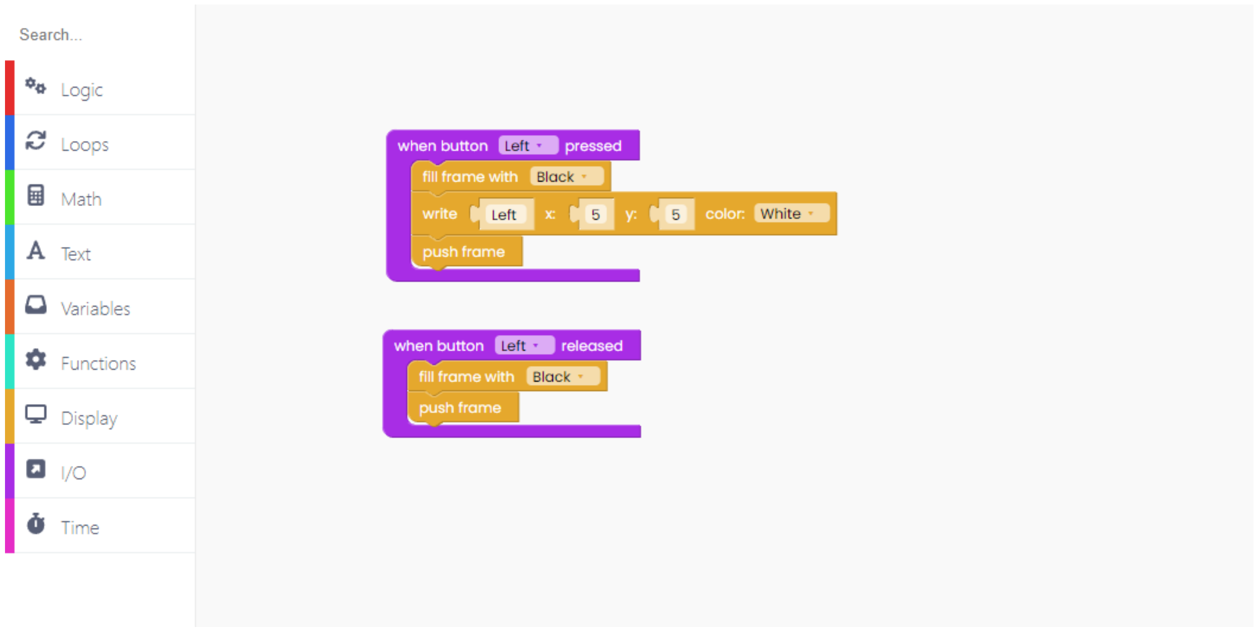
Now we'll write the code for what happens when the pushbutton is released.

Find this block in the I/O block section:



```
when button Left pressed
    fill frame with Black
    write Left x: 5 y: 5 color: White
    push frame

when button Left released
```

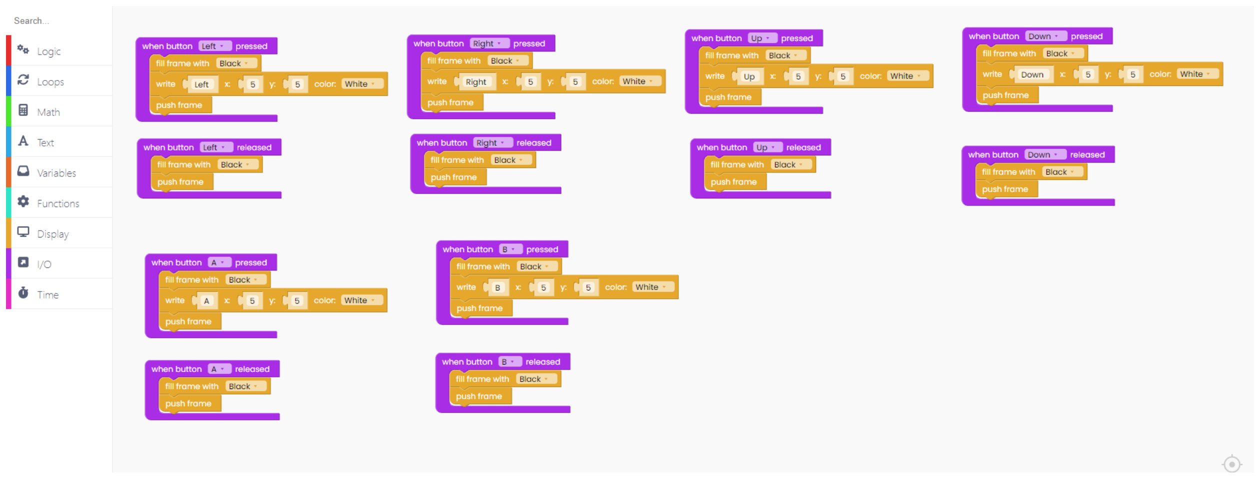
We want the screen to turn black and for the word to disappear.



```
when button Left pressed
    fill frame with Black
    write Left x: 5 y: 5 color: White
    push frame

when button Left released
    fill frame with Black
    push frame
```

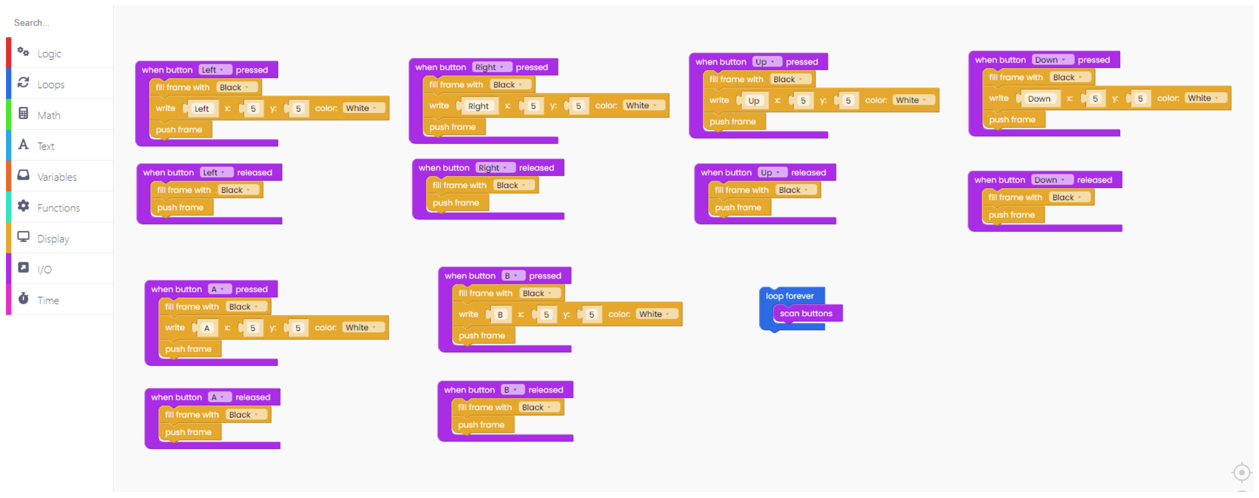
Let's repeat this for each button.

This is what your code should look like at the end:



```
when button Left pressed
    fill frame with Black
    write Left x: 5 y: 5 color: White
    push frame

when button Left released
    fill frame with Black
    push frame

when button Right pressed
    fill frame with Black
    write Right x: 5 y: 5 color: White
    push frame

when button Right released
    fill frame with Black
    push frame

when button Up pressed
    fill frame with Black
    write Up x: 5 y: 5 color: White
    push frame

when button Up released
    fill frame with Black
    push frame

when button Down pressed
    fill frame with Black
    write Down x: 5 y: 5 color: White
    push frame

when button Down released
    fill frame with Black
    push frame

when button A pressed
    fill frame with Black
    write A x: 5 y: 5 color: White
    push frame

when button A released
    fill frame with Black
    push frame

when button B pressed
    fill frame with Black
    write B x: 5 y: 5 color: White
    push frame

when button B released
    fill frame with Black
    push frame
```

Don't forget the "loop forever" and "scan buttons" blocks at the end.



```
when button Left pressed
    fill frame with Black
    write Left x: 5 y: 5 color: White
    push frame

when button Left released
    fill frame with Black
    push frame

when button Right pressed
    fill frame with Black
    write Right x: 5 y: 5 color: White
    push frame

when button Right released
    fill frame with Black
    push frame

when button Up pressed
    fill frame with Black
    write Up x: 5 y: 5 color: White
    push frame

when button Up released
    fill frame with Black
    push frame

when button Down pressed
    fill frame with Black
    write Down x: 5 y: 5 color: White
    push frame

when button Down released
    fill frame with Black
    push frame

when button A pressed
    fill frame with Black
    write A x: 5 y: 5 color: White
    push frame

when button A released
    fill frame with Black
    push frame

when button B pressed
    fill frame with Black
    write B x: 5 y: 5 color: White
    push frame

when button B released
    fill frame with Black
    push frame

loop forever
    scan buttons
```

Click on the Run button, and start pressing the pushbuttons on your gaming console.

If the color of the screen changes and the words appear, your code is successful.

Create your first game!

Let's try something more advanced now!

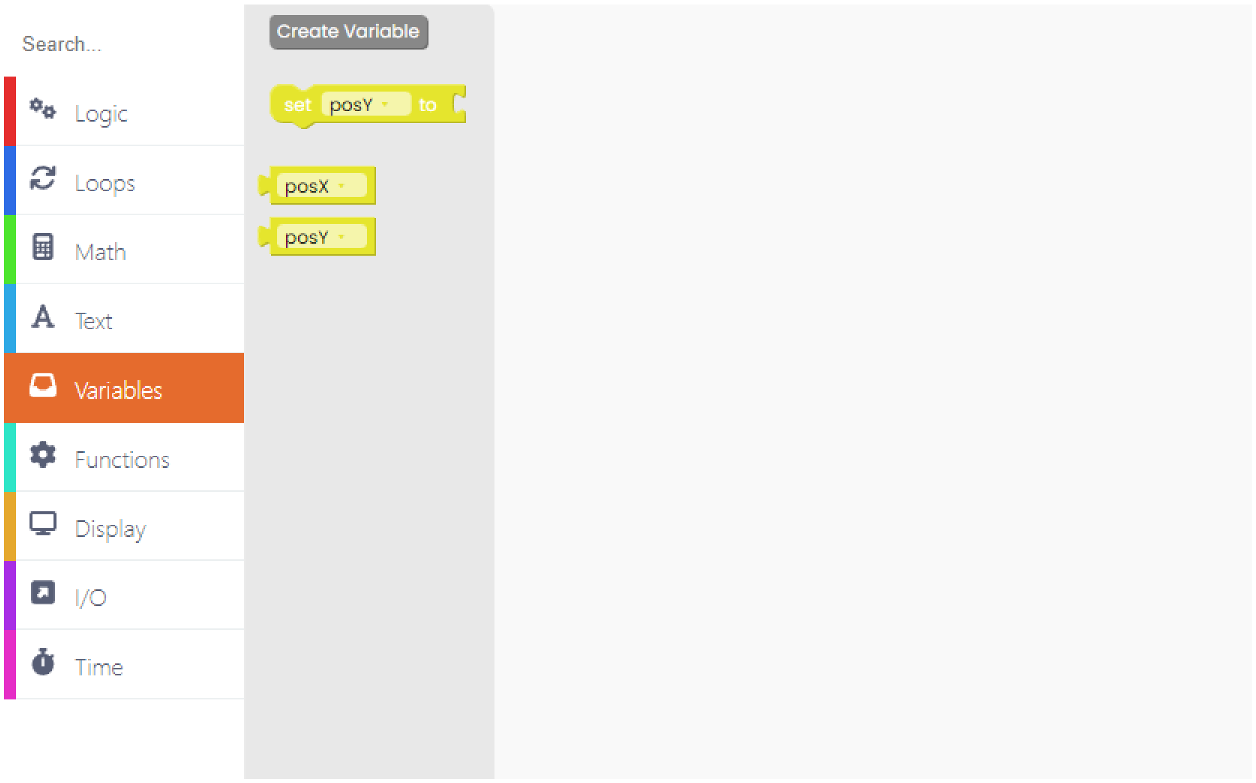
Creating a whole game is not an easy task. There are a lot of little details that need to be considered to make the game work as well as possible. When creating your own game, always set the goals at the beginning so that you can work from the bottom up with some structure in mind. Writing code without a bigger picture in mind can be a big problem later in code development.

The game that we'll create will showcase pretty much every feature we've gone through so far.

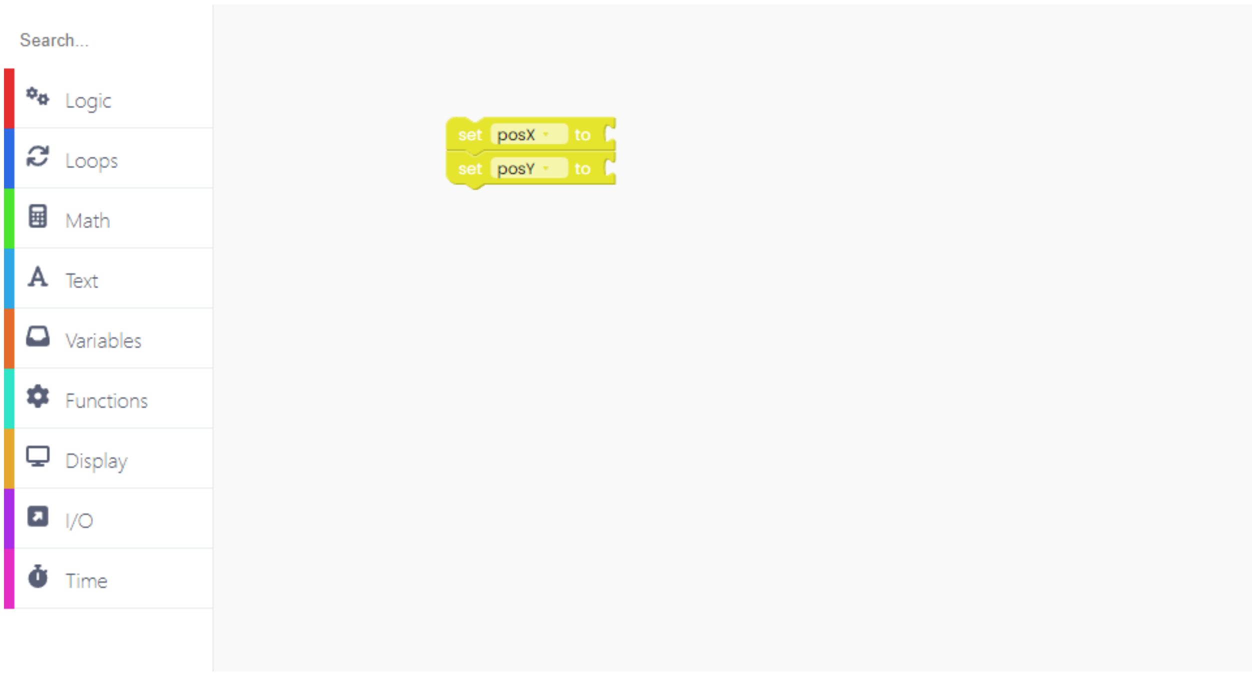
It will look like a simple video game, where you'll move a character that collects objects while counting a score.

Let's begin!

Now, let's make some variables and name them "posX" and "posY". Those will define your (player's) position on the screen.

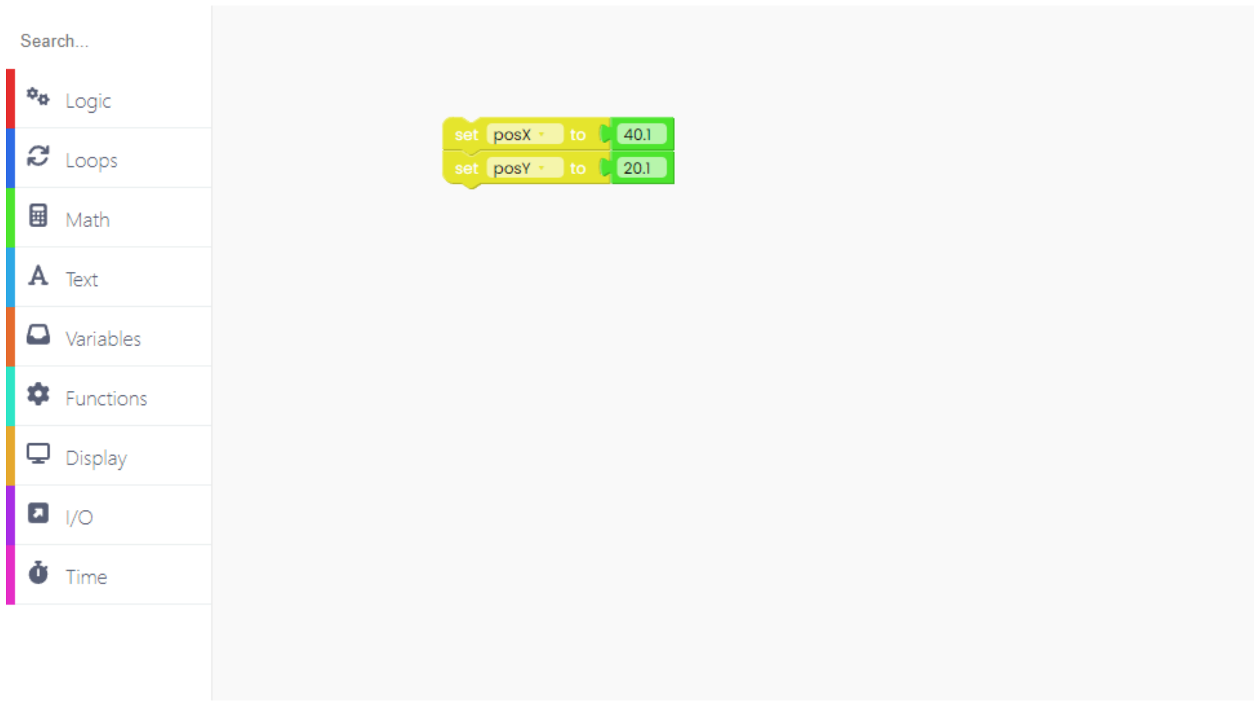


Drag and drop them onto the drawing area.



Find this block named "123" in the "Math" section. Type in the numerical value you want once you drop it onto the drawing area.

Write these numerical values in the circles:



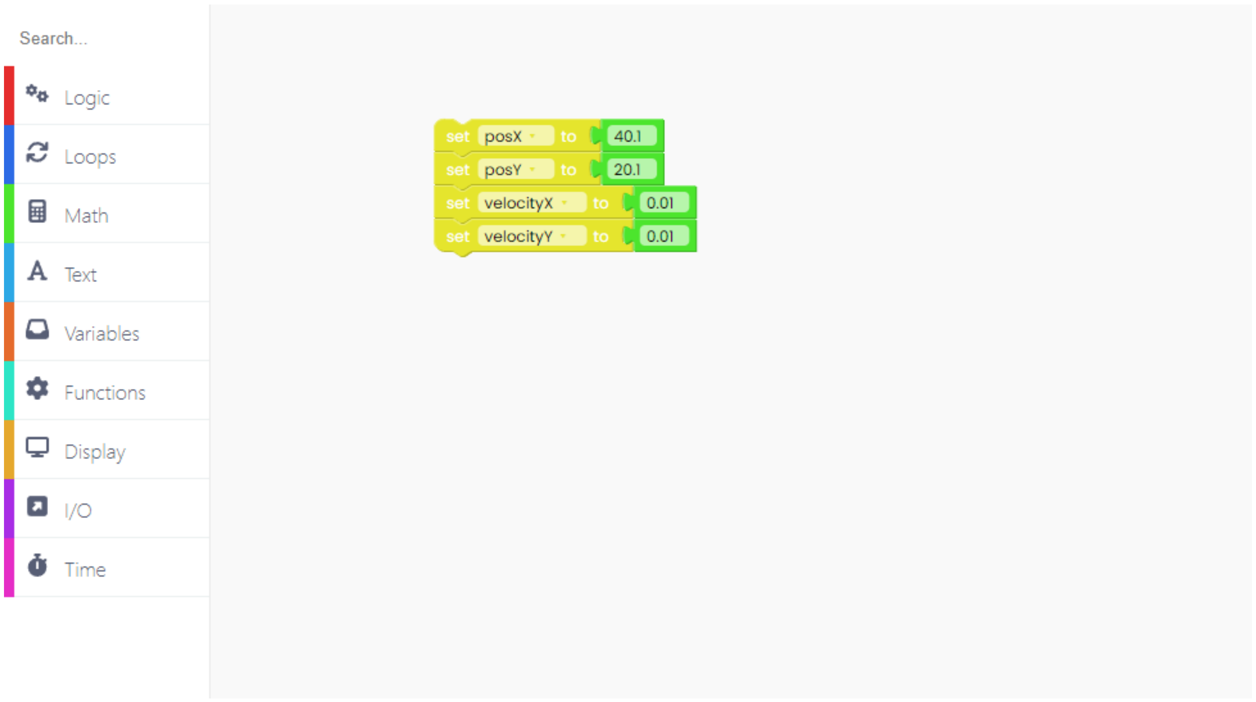
set posX to 40.1
set posY to 20.1

Create another set of variables that'll determine the speed of your ball.

Let's call them "velocityX" and "velocityY". Drag and drop the new variables under the variables we used for determining position.

"VelocityX" means that this will be the velocity of the ball in the x-axis direction, and "velocityY" determines the velocity of the ball in the y-axis direction.

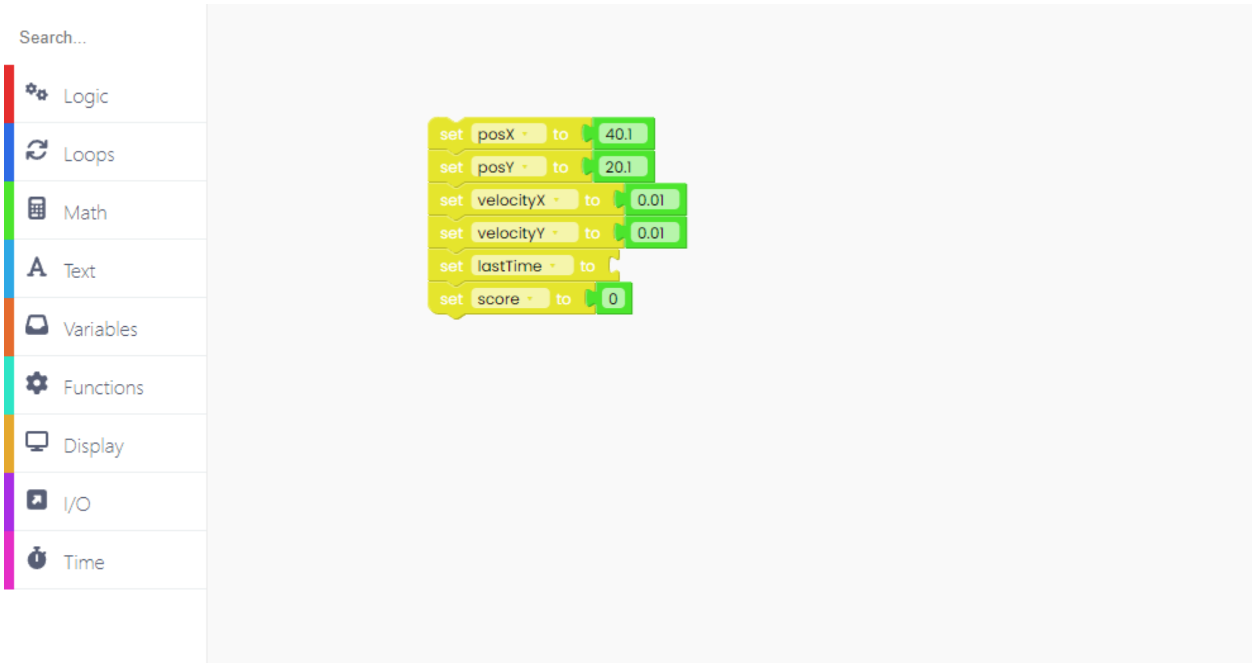
Make the numerical value of the velocity 0.01 for both X and Y.



```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
```

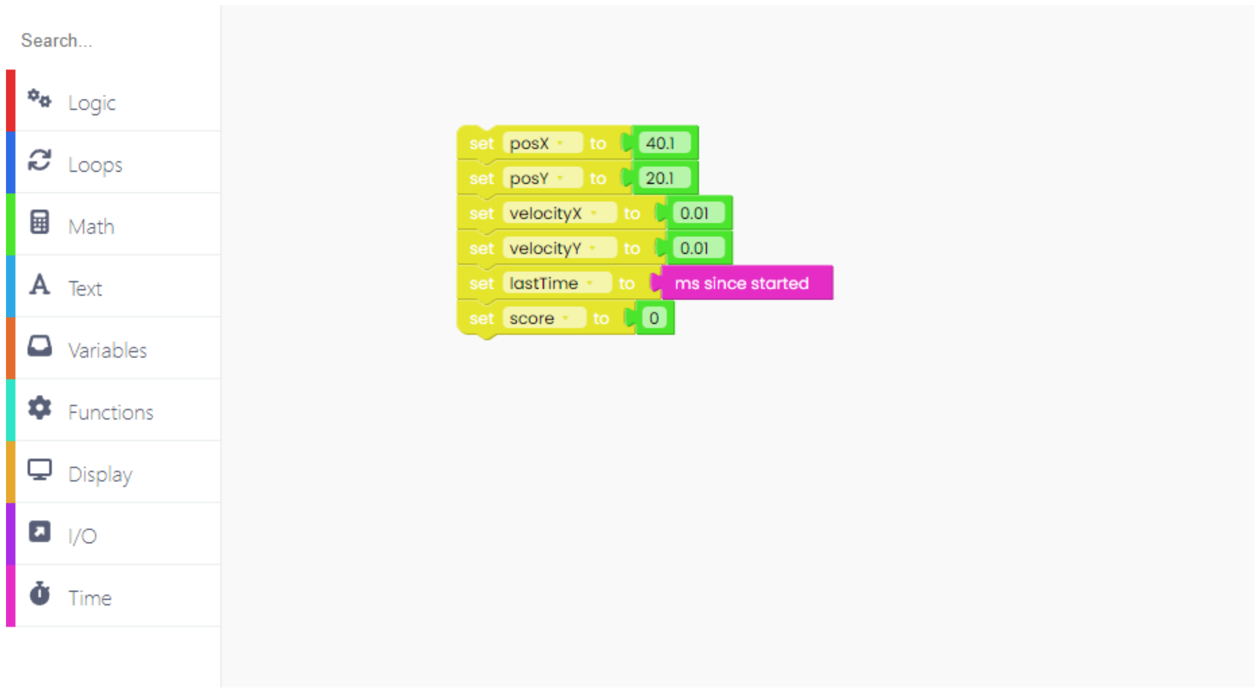
Let's make a few more variables while setting up our game.

You can name them "lastTime" and "score". Of course, you should put the value of the "score" to zero (0) at the beginning of the code.



```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to
set score to 0
```

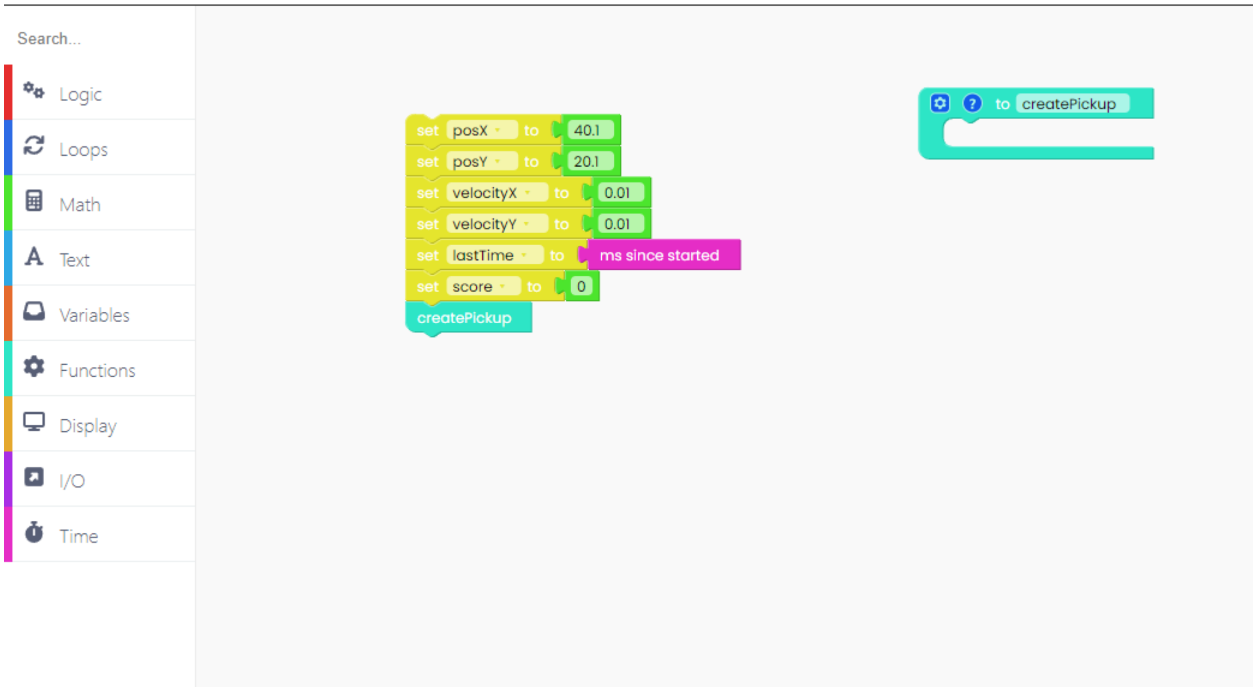
Now go to the Time block section and look for the "ms since started" block and add it to the "lastTime" variable.



```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
```

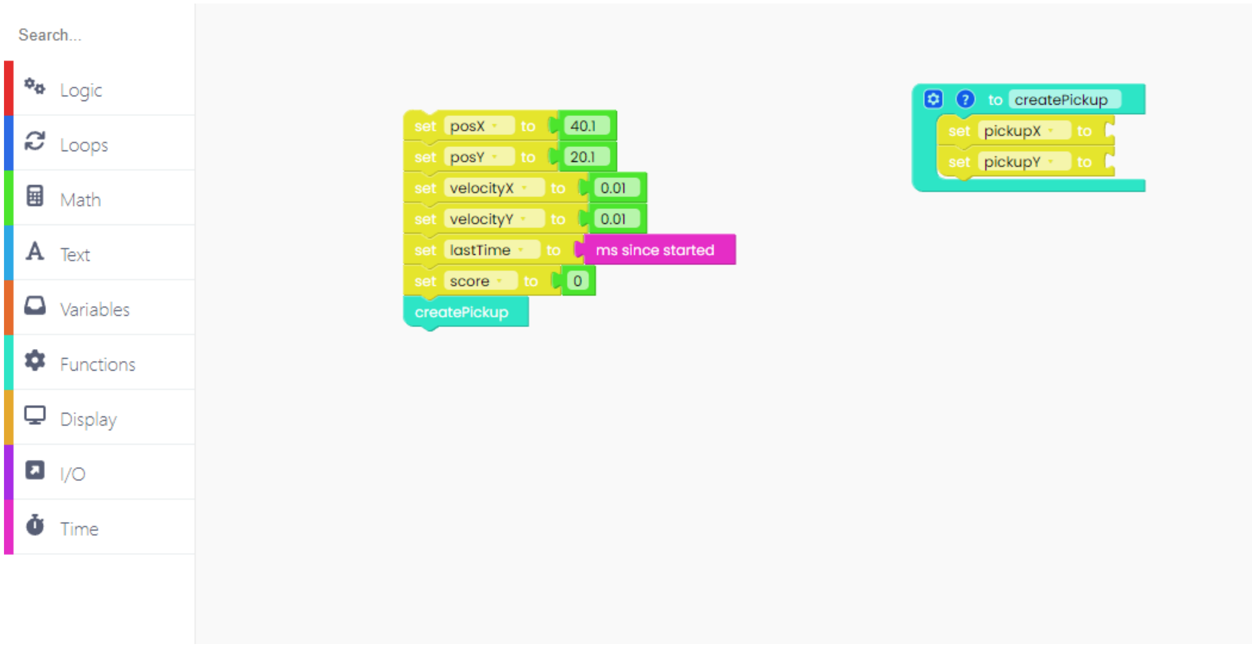
We have to create a function if we want to execute some code.

Let's name our function "createPickup" and place it under the "set score" variable.



You probably guessed the next step - creating variables for the newly made function.

Let's call them "pickupX" and "pickupY" as they'll define the position of the pickup (the smaller ball you'll have to catch).

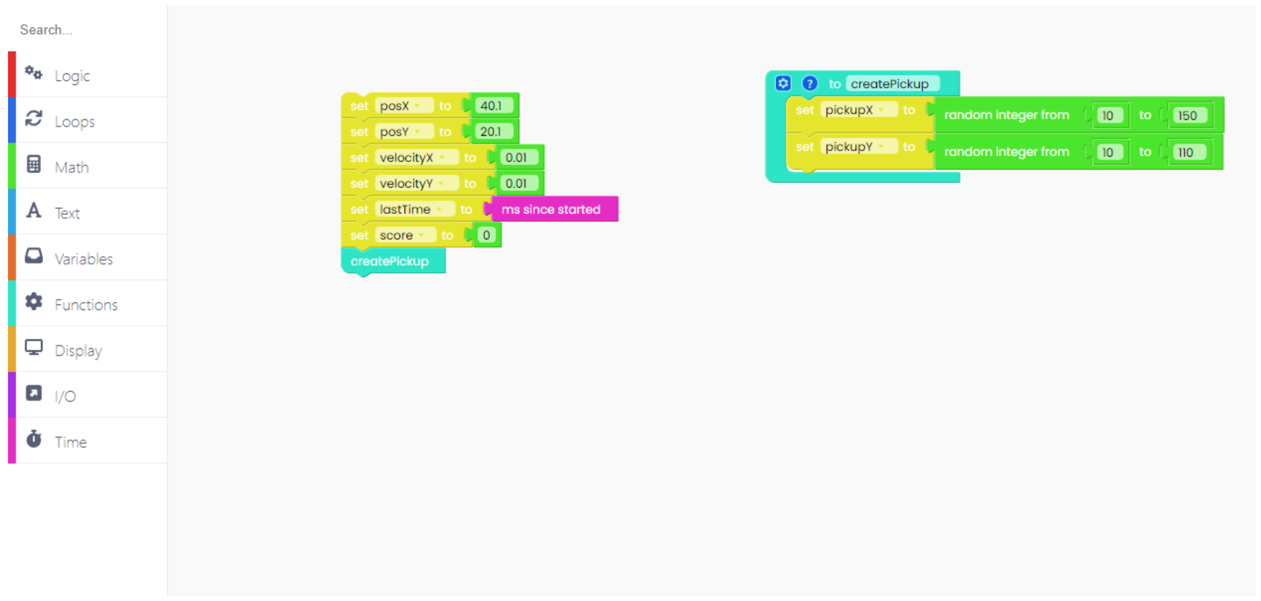


We'll set those variables to a random integer, and you'll find that block in the Math section.

An integer is a whole number that can be positive, negative, or zero. Examples of the integer are: '5, 1, 5, 8, 97.

Place those blocks in the pickup variables, and write the numerical value.

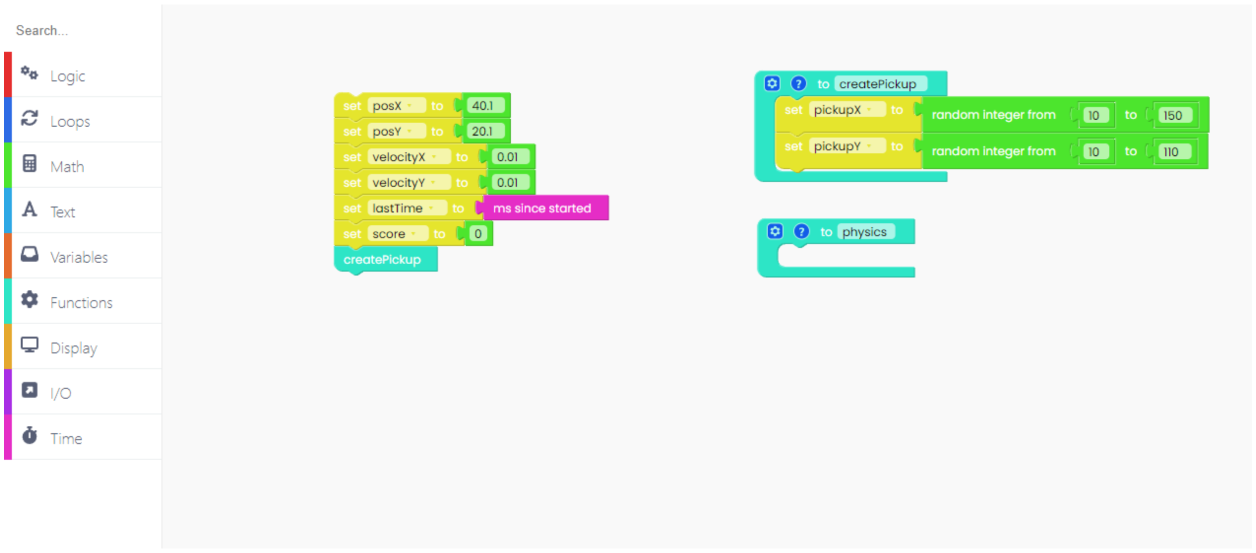
For "pickups" and "pickupY" you have to use a random number from 0 to 150 because the display resolution is 160*120 pixels.



```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup
```

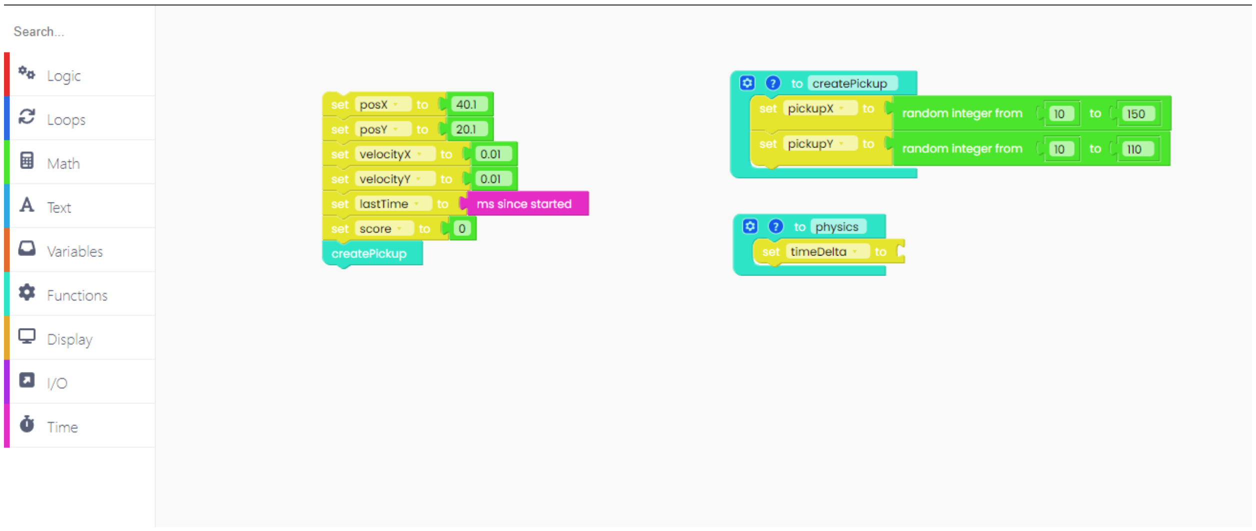
```
to createPickup
    set pickupX to random integer from 10 to 150
    set pickupY to random integer from 10 to 110
```

Firstly, we'll create a new function and call it "physics".



```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup

to createPickup
    set pickupX to random integer from 10 to 150
    set pickupY to random integer from 10 to 110

to physics
```

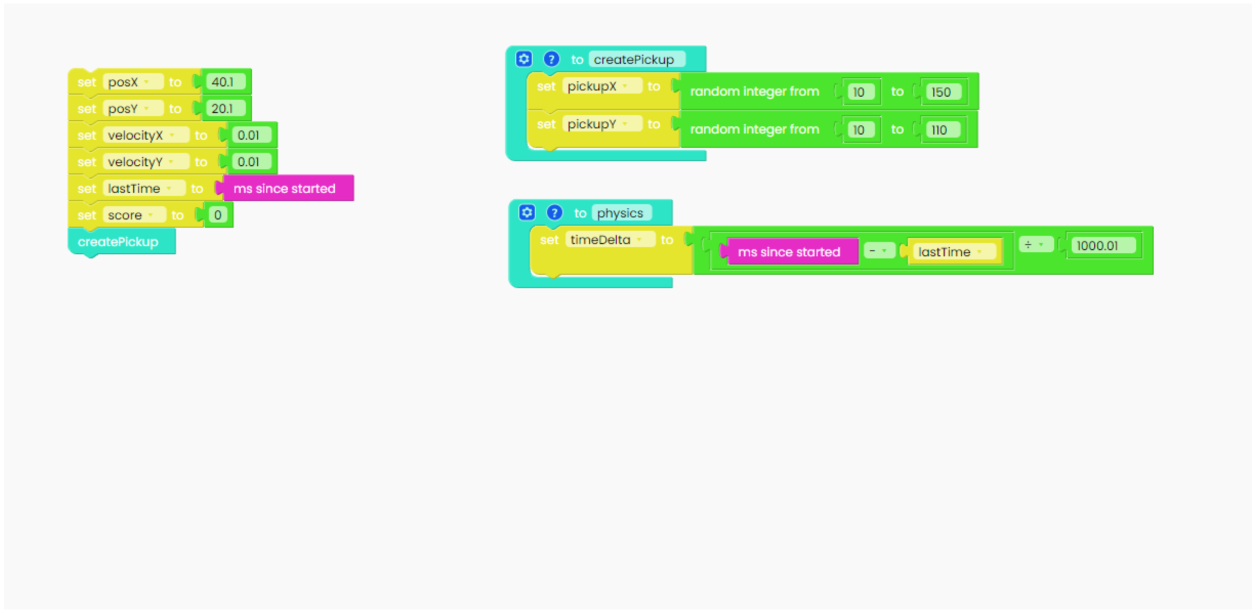
The first thing we have to do in the physics function is create a new variable called "timeDelta".



```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup

to createPickup
    set pickupX to random integer from 10 to 150
    set pickupY to random integer from 10 to 110

to physics
    set timeDelta to
```

A Greek symbol delta is used to note the difference between the two things. We created the variable "timeDelta" to note the elapsed and measured time difference.

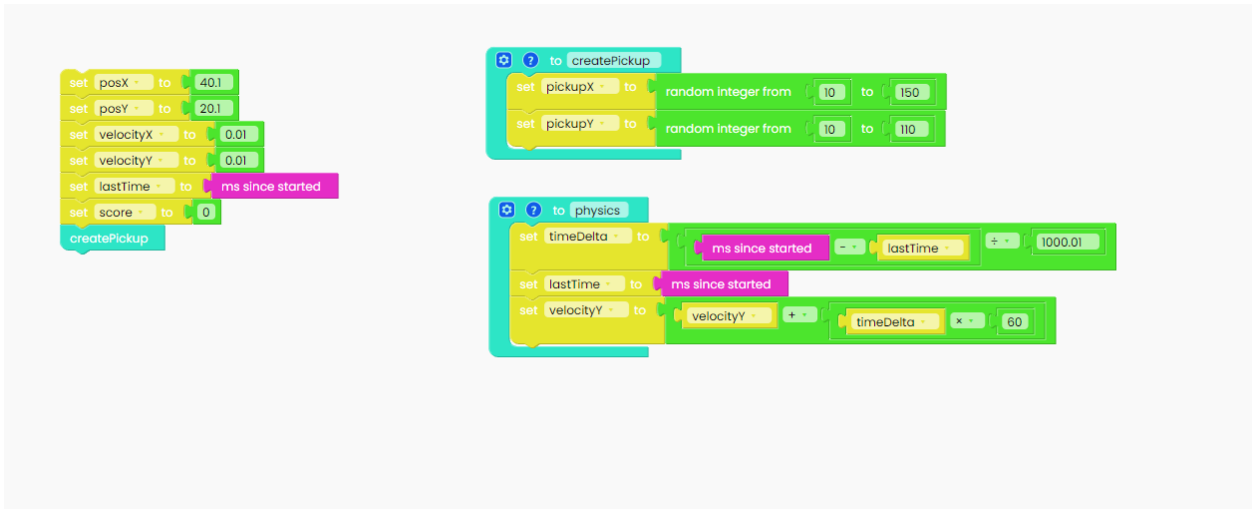
We will calculate "timeDelta" by subtracting from the current time the last time the function was executed. Also, we'll have to divide it by 1000.01 to get the time in seconds.



```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup

to createPickup
    set pickupX to random integer from 10 to 150
    set pickupY to random integer from 10 to 110

to physics
    set timeDelta to ms since started - lastTime ÷ 1000.01
```

Now, let's play with the physics formulas a little bit.

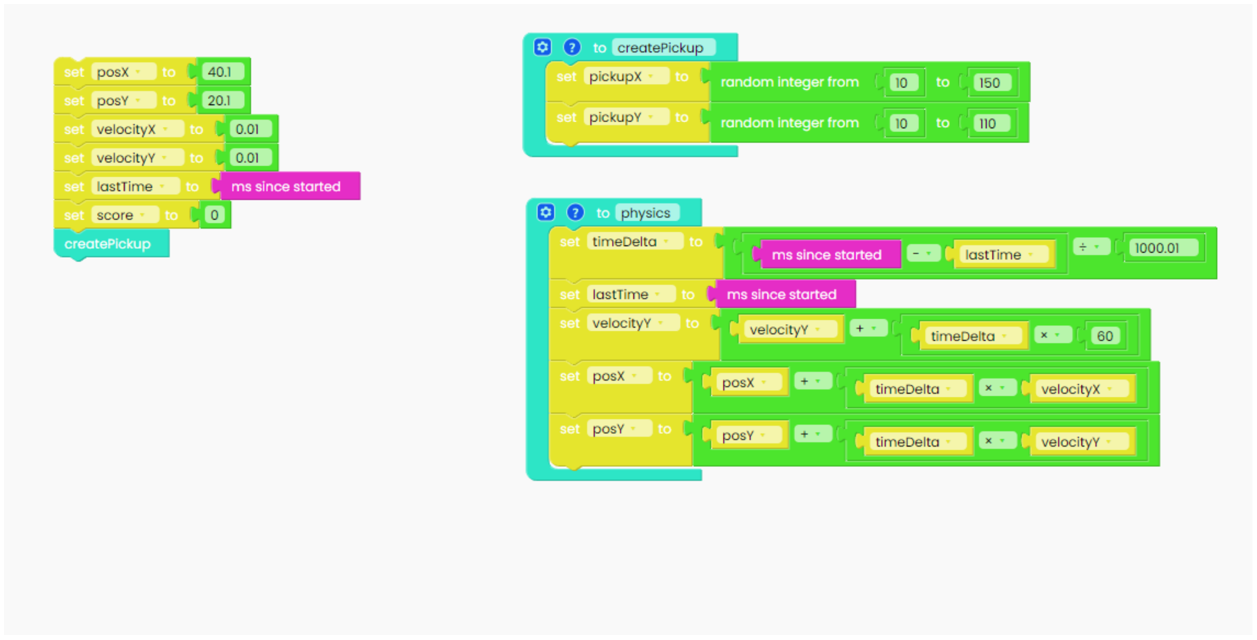
"VelocityY" will increase by multiplication of "timeDelta" and 60.



```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup

to createPickup
    set pickupX to random integer from 10 to 150
    set pickupY to random integer from 10 to 110

to physics
    set timeDelta to ms since started - lastTime ÷ 1000.01
    set lastTime to ms since started
    set velocityY to velocityY + timeDelta × 60
```

We have to update position X and position Y based on velocity.

You can do that by using "change posX/posY" blocks and placing them under the "change velocityY" block.



```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup
```

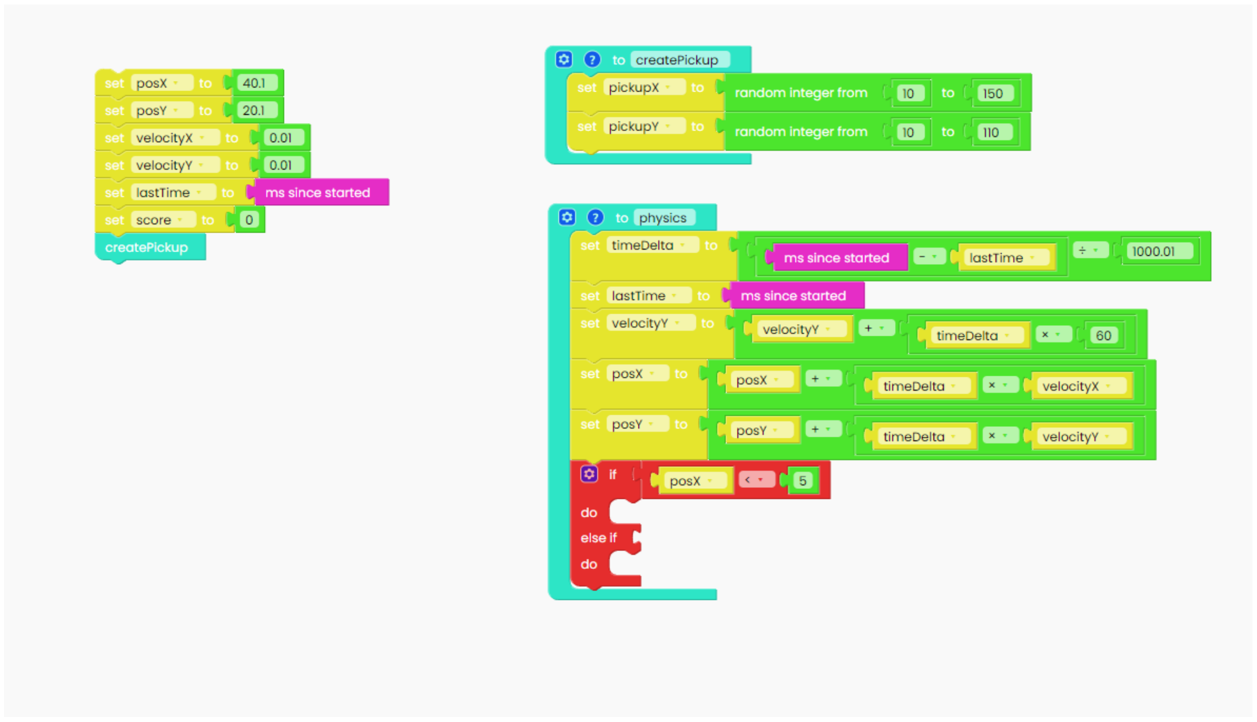
```
to createPickup
    set pickupX to random integer from 10 to 150
    set pickupY to random integer from 10 to 110
```

```
to physics
    set timeDelta to ms since started - lastTime ÷ 1000.01
    set lastTime to ms since started
    set velocityY to velocityY + timeDelta × 60
    set posX to posX + timeDelta × velocityX
    set posY to posY + timeDelta × velocityY
```

Now, let's see what will happen if the ball hits one of the edges of the display.

We'll use the "if-do-else" block from the Logic section and place the comparison block in it.

Your code will check if the ball's position on the x-axis is under five (5). As we mentioned before, the x-axis goes from 0 to 150.



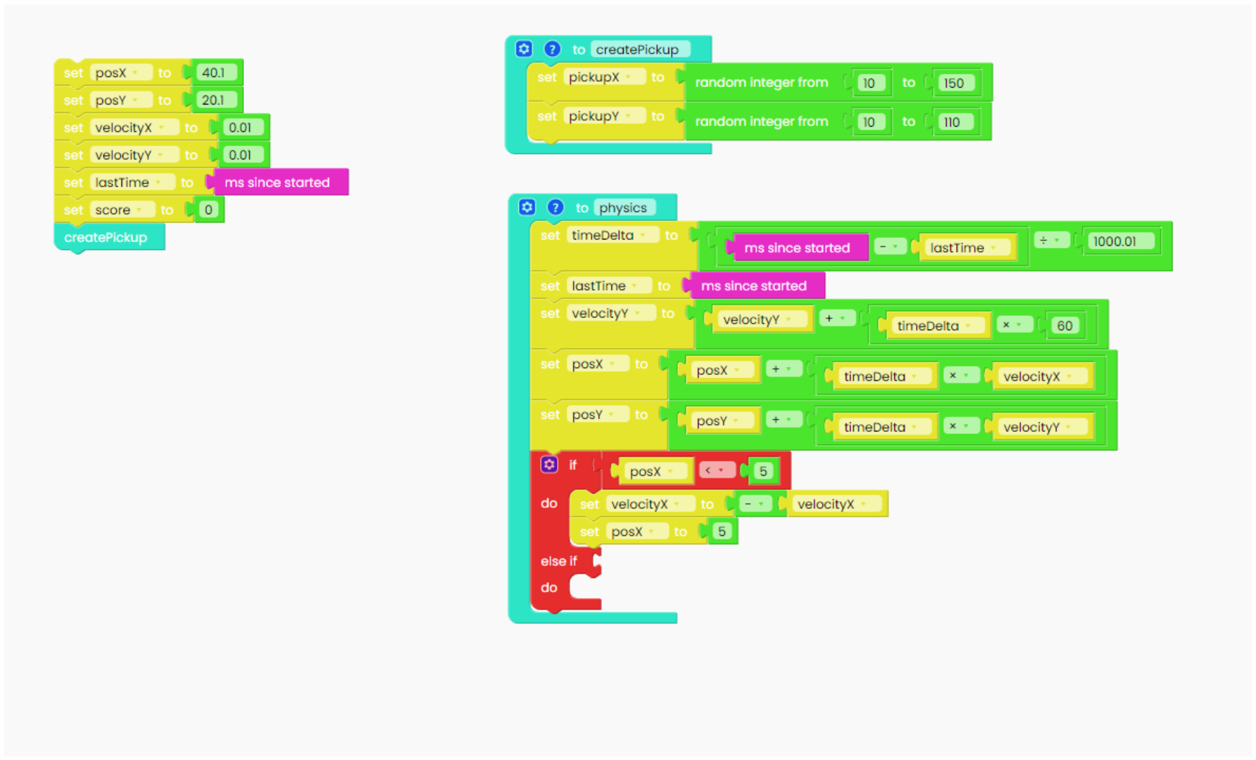
```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup
```

```
to createPickup
    set pickupX to random integer from 10 to 150
    set pickupY to random integer from 10 to 110
```

```
to physics
    set timeDelta to ms since started - lastTime ÷ 1000.01
    set lastTime to ms since started
    set velocityY to velocityY + timeDelta × 60
    set posX to posX + timeDelta × velocityX
    set posY to posY + timeDelta × velocityY
    if posX < 5
    do
    else if
    do
```

If position X is under the numerical value we gave it, which is five, velocityX will change into -velocityX (negative on the axis).

In other words, the ball will change its direction when it hits the edge.



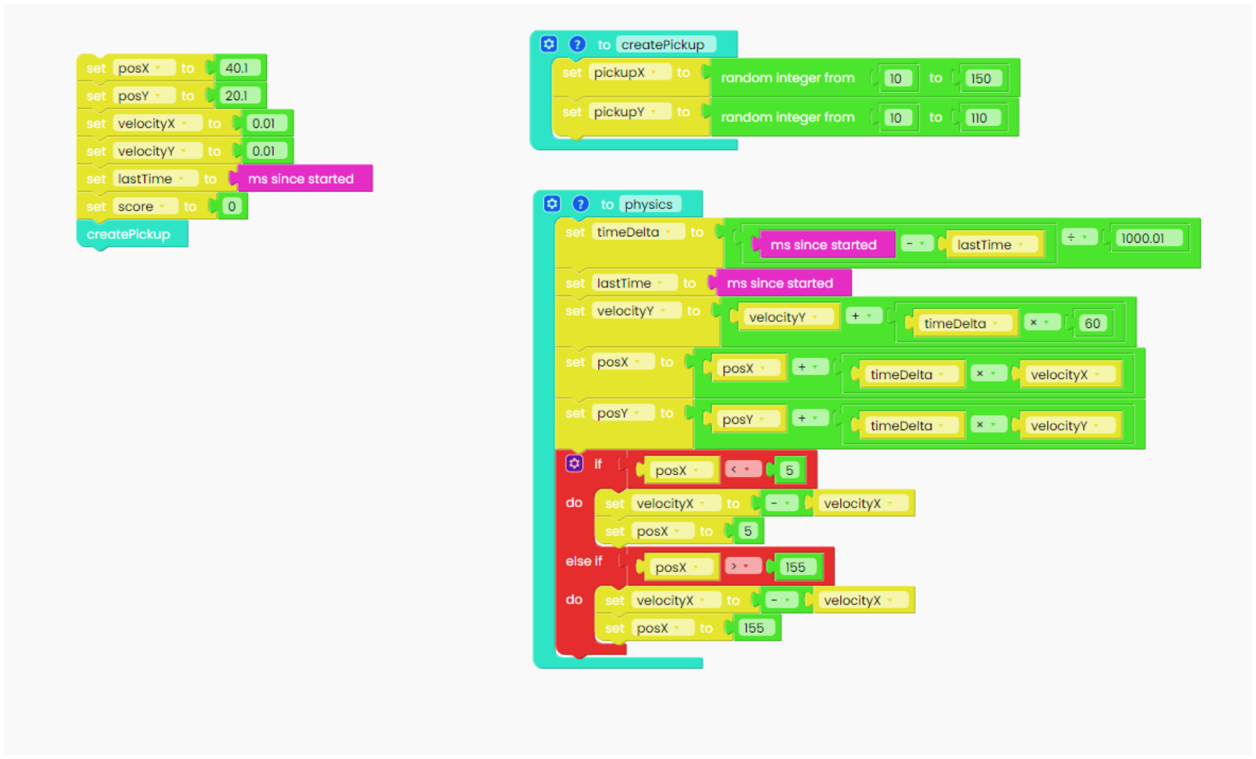
```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup
```

```
to createPickup
    set pickupX to random integer from 10 to 150
    set pickupY to random integer from 10 to 110
```

```
to physics
    set timeDelta to ( ms since started - lastTime ) ÷ 1000.01
    set lastTime to ms since started
    set velocityY to velocityY + ( timeDelta × 60 )
    set posX to posX + ( timeDelta × velocityX )
    set posY to posY + ( timeDelta × velocityY )
    if posX < 5
    do  set velocityX to - velocityX
        set posX to 5
    else if
    do
```

Now, let's see what happens if the ball is in position over 155.

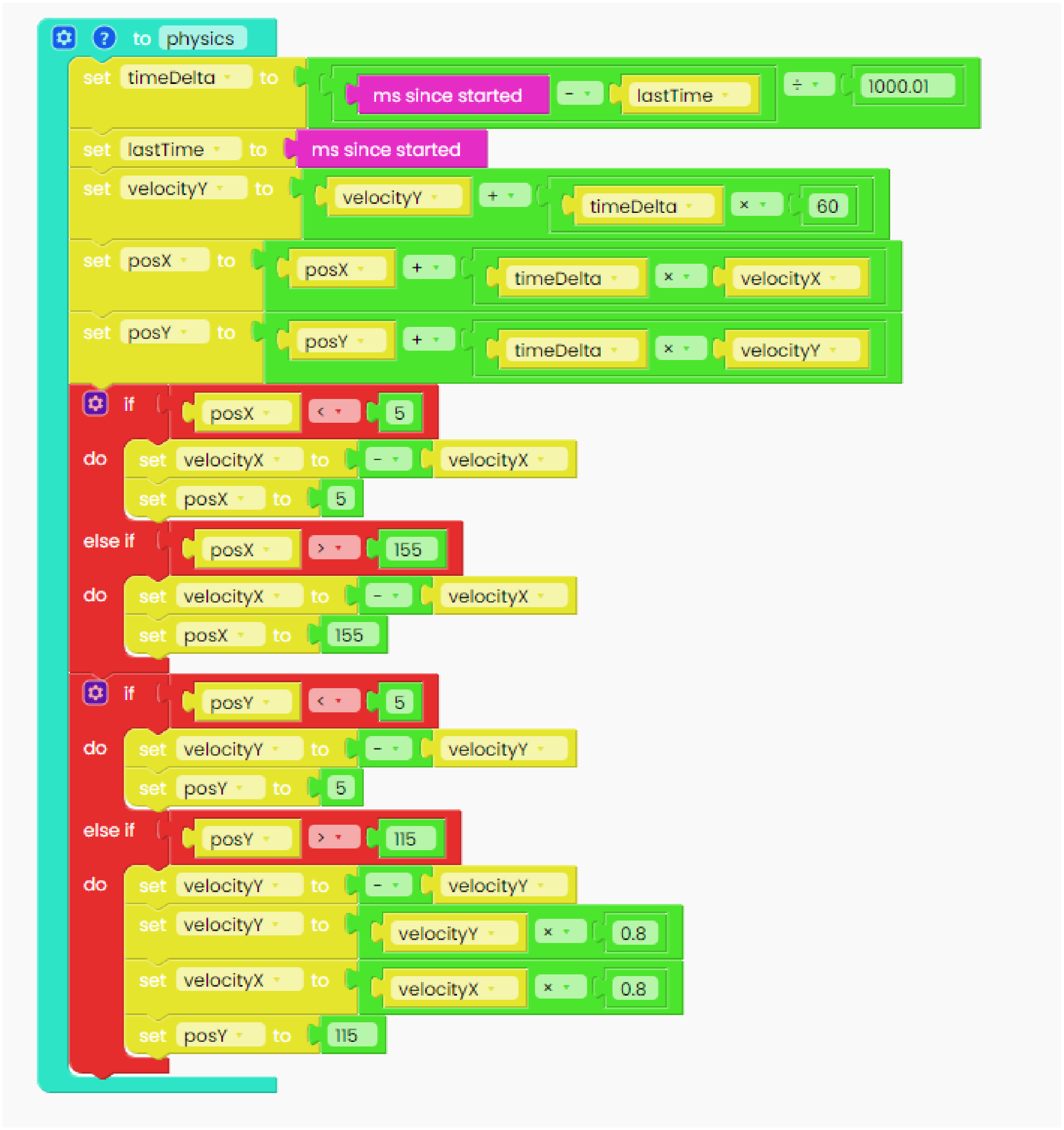
Once again, we have decided that the posX is set to 155. So, if the velocityX is bigger than 155, the ball will change its direction.



```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup

to createPickup
    set pickupX to random integer from 10 to 150
    set pickupY to random integer from 10 to 110

to physics
    set timeDelta to ( ms since started - lastTime ) ÷ 1000.01
    set lastTime to ms since started
    set velocityY to velocityY + ( timeDelta × 60 )
    set posX to posX + ( timeDelta × velocityX )
    set posY to posY + ( timeDelta × velocityY )
    if posX < 5
    do  set velocityX to - velocityX
        set posX to 5
    else if posX > 155
    do  set velocityX to - velocityX
        set posX to 155
```

Now, let's repeat the procedure for the Y position.

The physics function should look like this:



```
to physics
  set timeDelta to ( ( ms since started - lastTime ) ÷ 1000.01 )
  set lastTime to ms since started
  set velocityY to ( velocityY + ( timeDelta × 60 ) )
  set posX to ( posX + ( timeDelta × velocityX ) )
  set posY to ( posY + ( timeDelta × velocityY ) )
  if posX < 5
  do
    set velocityX to - velocityX
    set posX to 5
  else if posX > 155
  do
    set velocityX to - velocityX
    set posX to 155
  if posY < 5
  do
    set velocityY to - velocityY
    set posY to 5
  else if posY > 115
  do
    set velocityY to - velocityY
    set velocityY to ( velocityY × 0.8 )
    set velocityX to ( velocityX × 0.8 )
    set posY to 115
```

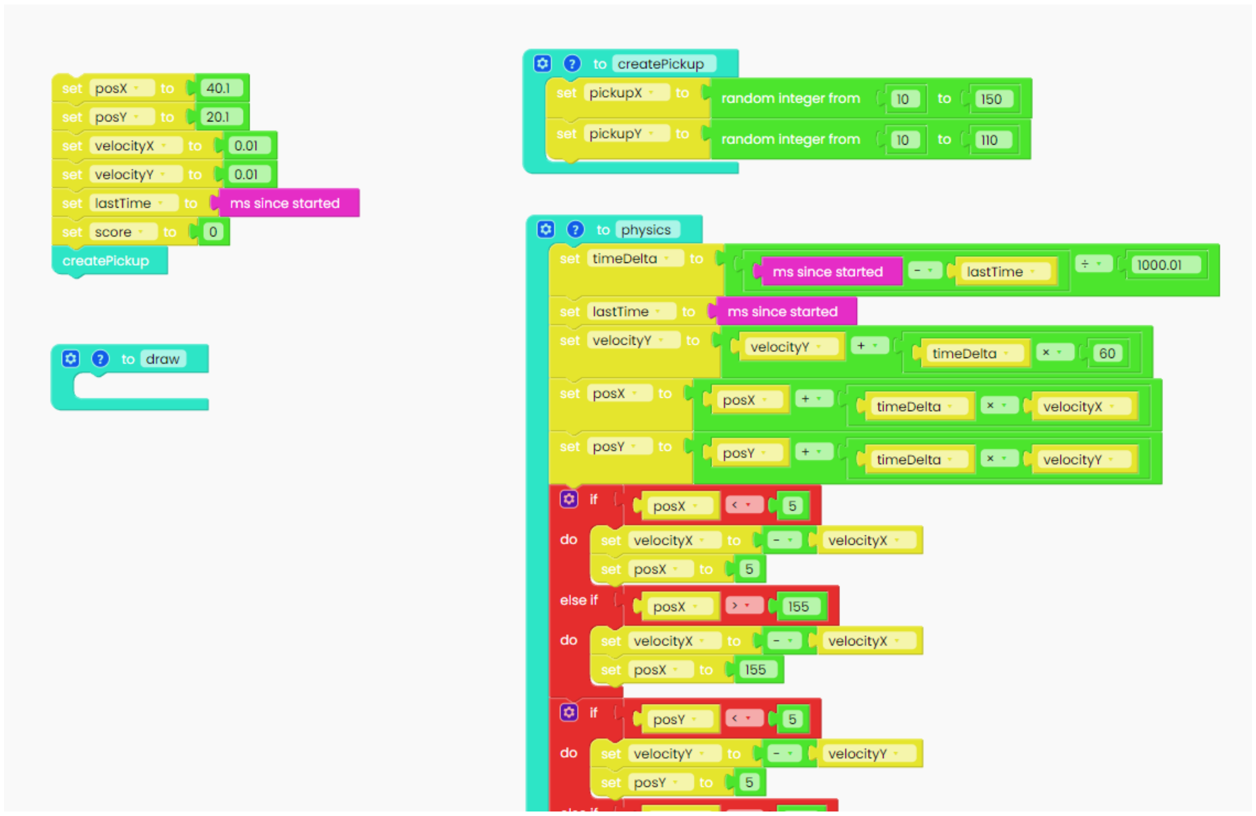
As you can see, we did the same thing as we did with position X.

We've done the hard part! Congratulations!

Now, let's do some drawing.

Can you guess what the next step is?

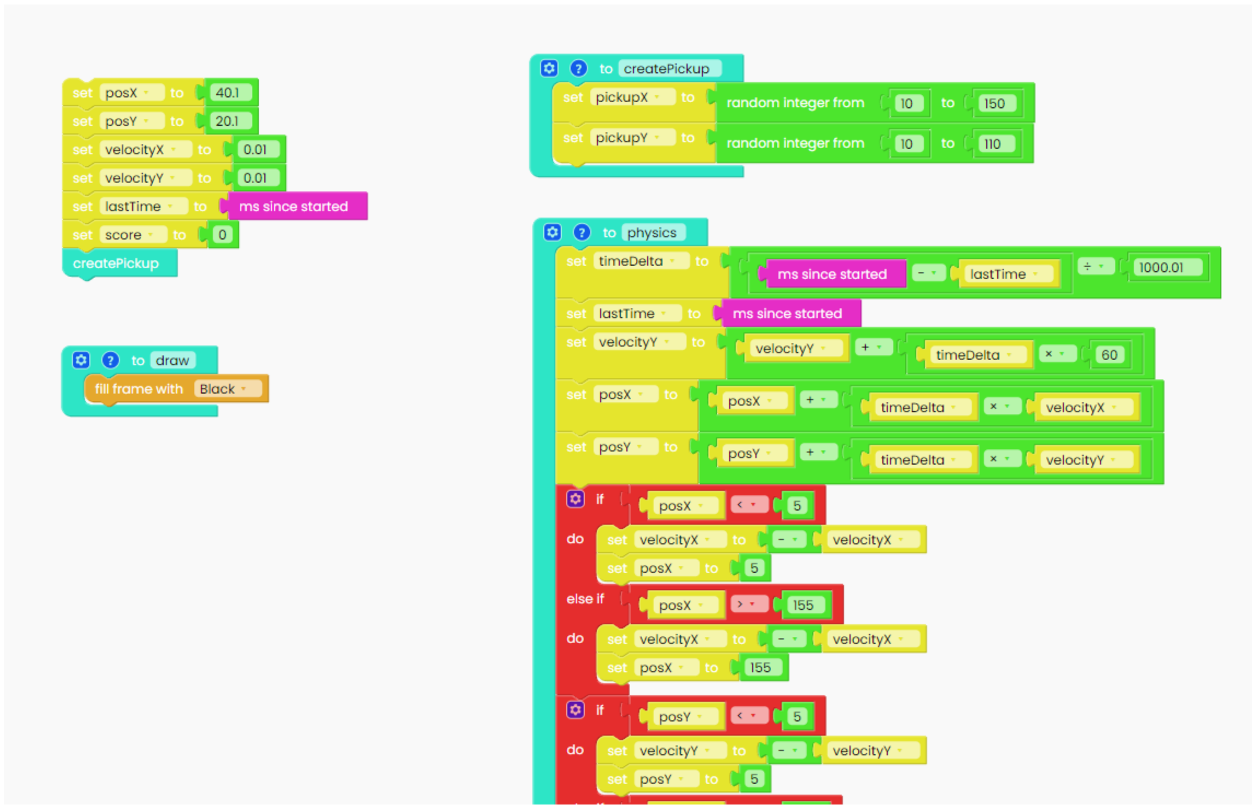
That's right! Creating a new function - let's call it draw!



```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup

to draw

to createPickup
    set pickupX to random integer from 10 to 150
    set pickupY to random integer from 10 to 110

to physics
    set timeDelta to ms since started - lastTime ÷ 1000.01
    set lastTime to ms since started
    set velocityY to velocityY + timeDelta × 60
    set posX to posX + timeDelta × velocityX
    set posY to posY + timeDelta × velocityY
    if posX < 5
    do  set velocityX to - velocityX
        set posX to 5
    else if posX > 155
    do  set velocityX to - velocityX
        set posX to 155
    if posY < 5
    do  set velocityY to - velocityY
        set posY to 5
```

Every time you draw something on your display, the first thing you want to do is fill the display with some color. We chose black.

You can find that particular block in the Display section.



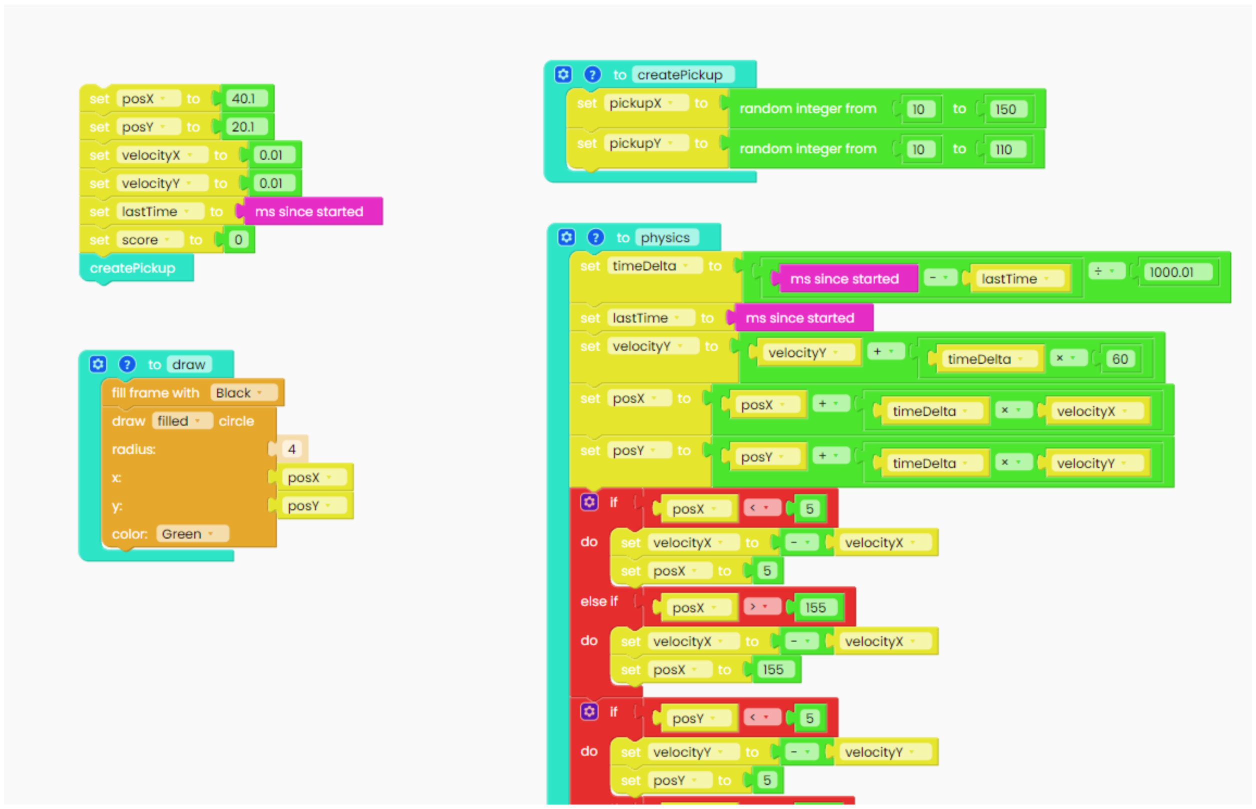
```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup

to draw
  fill frame with Black

to createPickup
  set pickupX to random integer from 10 to 150
  set pickupY to random integer from 10 to 110

to physics
  set timeDelta to ms since started - lastTime ÷ 1000.01
  set lastTime to ms since started
  set velocityY to velocityY + timeDelta × 60
  set posX to posX + timeDelta × velocityX
  set posY to posY + timeDelta × velocityY
  if posX < 5
  do  set velocityX to - velocityX
      set posX to 5
  else if posX > 155
  do  set velocityX to - velocityX
      set posX to 155
  if posY < 5
  do  set velocityY to - velocityY
      set posY to 5
```

Once we color our screen black, we can start to draw.

As you already know, the game consists of a player represented by a ball and the other smaller balls that a player needs to catch.

Once again, let's go back to the Display section and choose one of the blocks.

This is the one you'll need:



```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup
```

```
to draw
  fill frame with Black
  draw filled circle
    radius: 4
    x: posX
    y: posY
    color: Green
```

```
to createPickup
  set pickupX to random integer from 10 to 150
  set pickupY to random integer from 10 to 110
```

```
to physics
  set timeDelta to ((ms since started - lastTime) ÷ 1000.01)
  set lastTime to ms since started
  set velocityY to velocityY + (timeDelta × 60)
  set posX to posX + (timeDelta × velocityX)
  set posY to posY + (timeDelta × velocityY)
  if posX < 5
  do  set velocityX to - velocityX
      set posX to 5
  else if posX > 155
  do  set velocityX to - velocityX
      set posX to 155
  if posY < 5
  do  set velocityY to - velocityY
      set posY to 5
```

The x coordinate will be the variable "posX", and y coordinate will be the variable "posY".

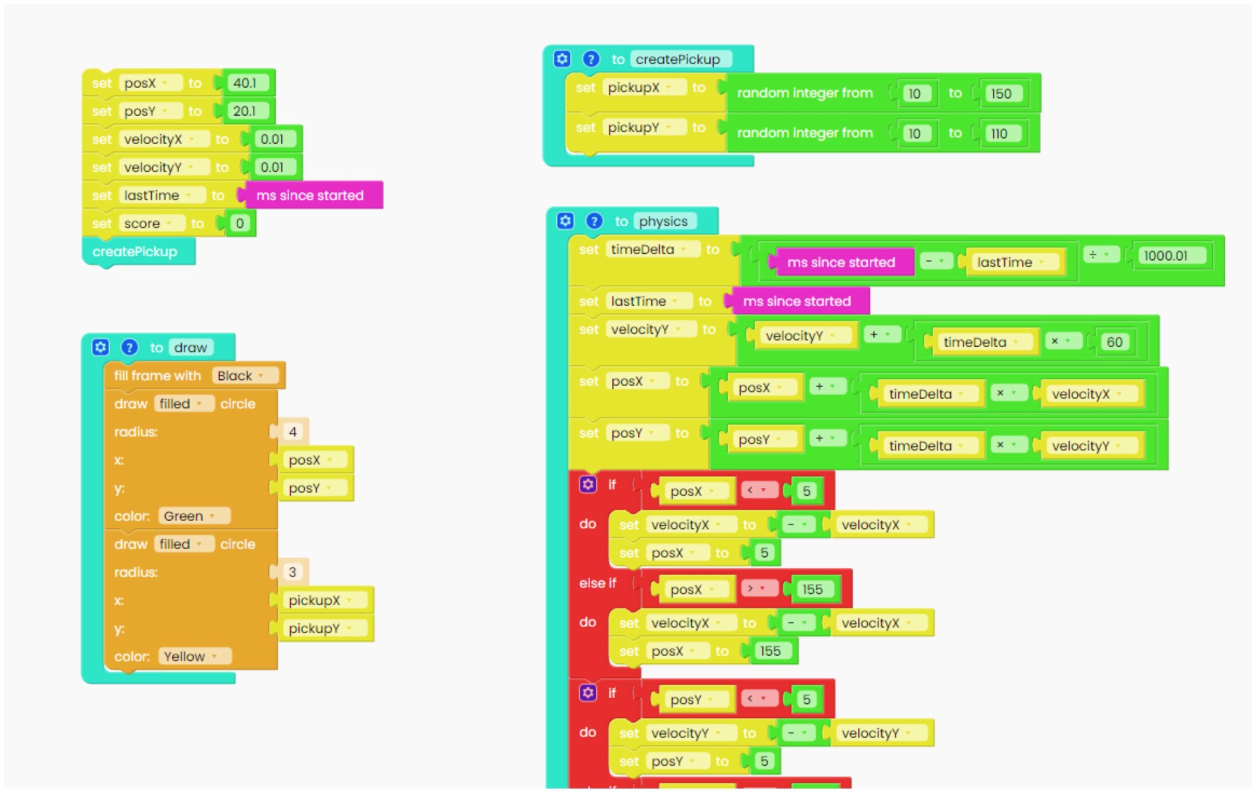
Radius is the size of the player ball, and we decided the size to be 4.

Now, let's draw the smaller balls you'll need to catch. The procedure is the same as we used to make a player ball.

Since we have to recognize two balls, we need to change the color of the other ball. You can put any color you want in your game.

Also, the x coordinate of the drawing function is variable "pickupX" and the y coordinate is the variable"pickupY".

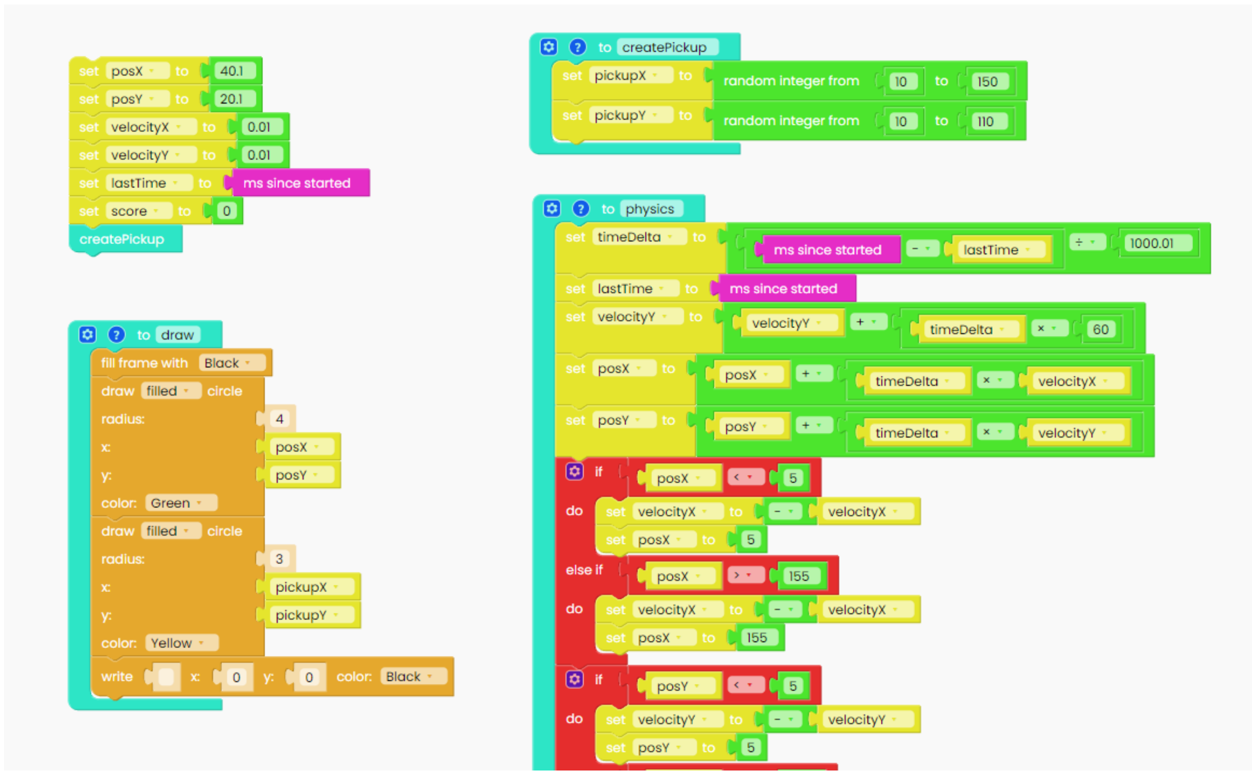
As this is a smaller ball, its radius will be 3.



```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup

to draw
  fill frame with Black
  draw filled circle
    radius: 4
    x: posX
    y: posY
    color: Green
  draw filled circle
    radius: 3
    x: pickupX
    y: pickupY
    color: Yellow

to createPickup
  set pickupX to random integer from 10 to 150
  set pickupY to random integer from 10 to 110

to physics
  set timeDelta to ((ms since started - lastTime) ÷ 1000.01)
  set lastTime to ms since started
  set velocityY to (velocityY + (timeDelta × 60))
  set posX to (posX + (timeDelta × velocityX))
  set posY to (posY + (timeDelta × velocityY))
  if posX < 5
  do  set velocityX to - velocityX
      set posX to 5
  else if posX > 155
  do  set velocityX to - velocityX
      set posX to 155
  if posY < 5
  do  set velocityY to - velocityY
      set posY to 5
```

Only one thing is left for us to draw, and that's the score. Making a game without keeping the score wouldn't make any sense.

You'll do that by dragging and dropping this particular block from the Display section:



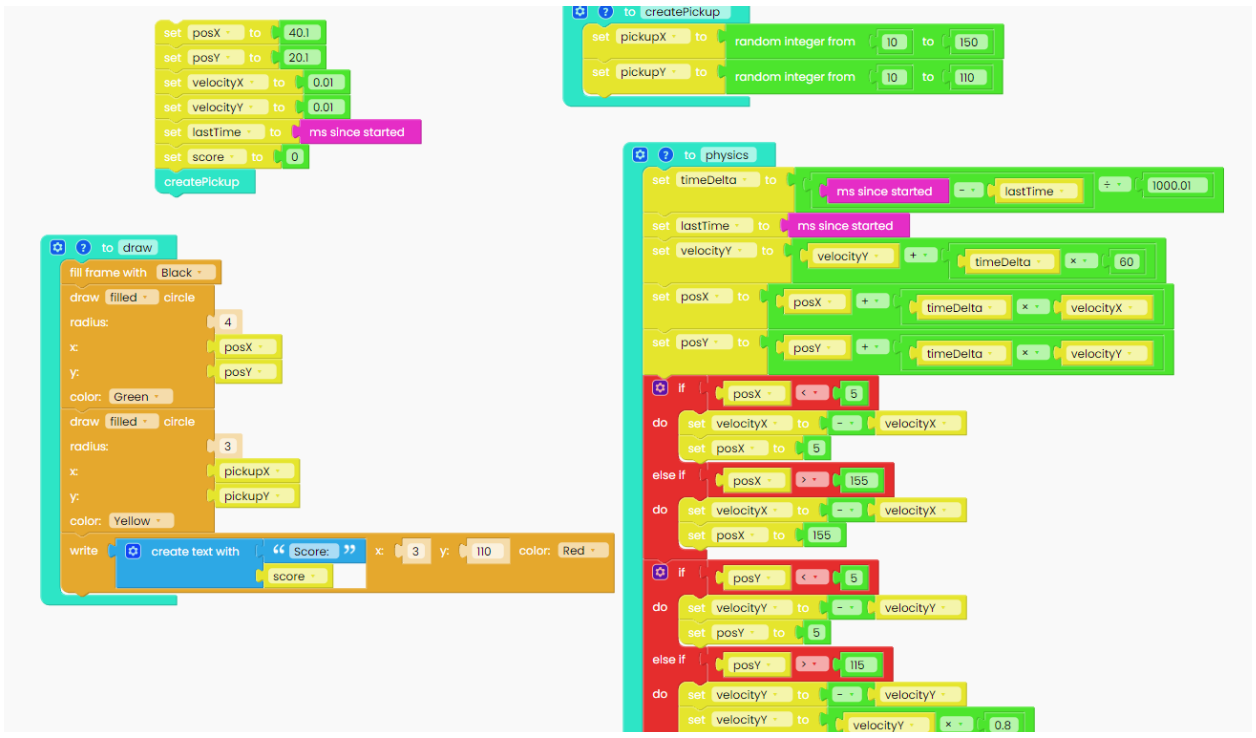
```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup

to draw
    fill frame with Black
    draw filled circle
    radius: 4
    x: posX
    y: posY
    color: Green
    draw filled circle
    radius: 3
    x: pickupX
    y: pickupY
    color: Yellow
    write x: 0 y: 0 color: Black

to createPickup
    set pickupX to random integer from 10 to 150
    set pickupY to random integer from 10 to 110

to physics
    set timeDelta to ms since started - lastTime ÷ 1000.01
    set lastTime to ms since started
    set velocityY to velocityY + timeDelta × 60
    set posX to posX + timeDelta × velocityX
    set posY to posY + timeDelta × velocityY
    if posX < 5
    do  set velocityX to - velocityX
        set posX to 5
    else if posX > 155
    do  set velocityX to - velocityX
        set posX to 155
    if posY < 5
    do  set velocityY to - velocityY
        set posY to 5
```

In order to display the score on the screen, we need to find the "create text" blue block.

We'll put the word "Score:" in the left circle, and for that, we need the first block from the Text section.

On the right side of the block, we'll put a variable called "score".

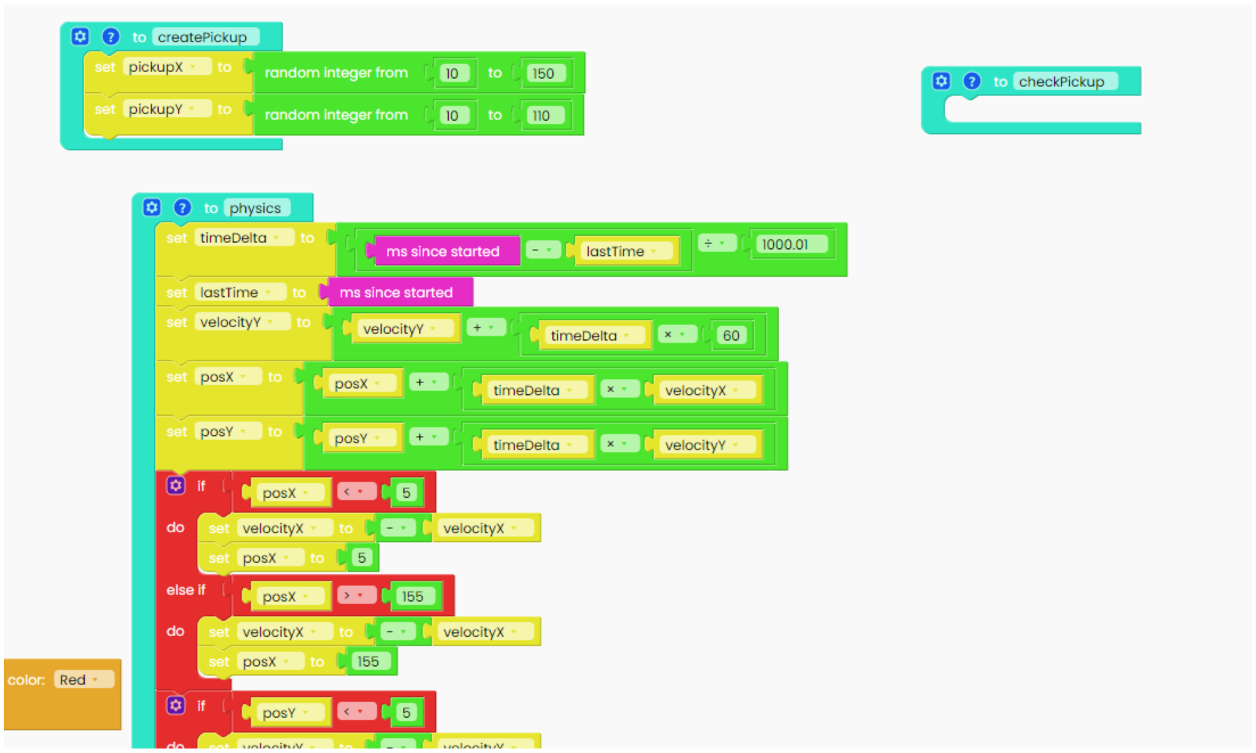
In x and y, you'll write the desired position where the score will be written on the screen.



```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup

to createPickup
    set pickupX to random integer from 10 to 150
    set pickupY to random integer from 10 to 110

to draw
    fill frame with Black
    draw filled circle
        radius: 4
        x: posX
        y: posY
        color: Green
    draw filled circle
        radius: 3
        x: pickupX
        y: pickupY
        color: Yellow
    write create text with " Score: "
                           score
        x: 3  y: 110  color: Red

to physics
    set timeDelta to ( ms since started - lastTime ) ÷ 1000.01
    set lastTime to ms since started
    set velocityY to velocityY + ( timeDelta × 60 )
    set posX to posX + ( timeDelta × velocityX )
    set posY to posY + ( timeDelta × velocityY )
    if posX < 5
    do  set velocityX to - velocityX
        set posX to 5
    else if posX > 155
    do  set velocityX to - velocityX
        set posX to 155
    if posY < 5
    do  set velocityY to - velocityY
        set posY to 5
    else if posY > 115
    do  set velocityY to - velocityY
        set velocityY to velocityY × 0.8
```

Good job, we're done with drawing!

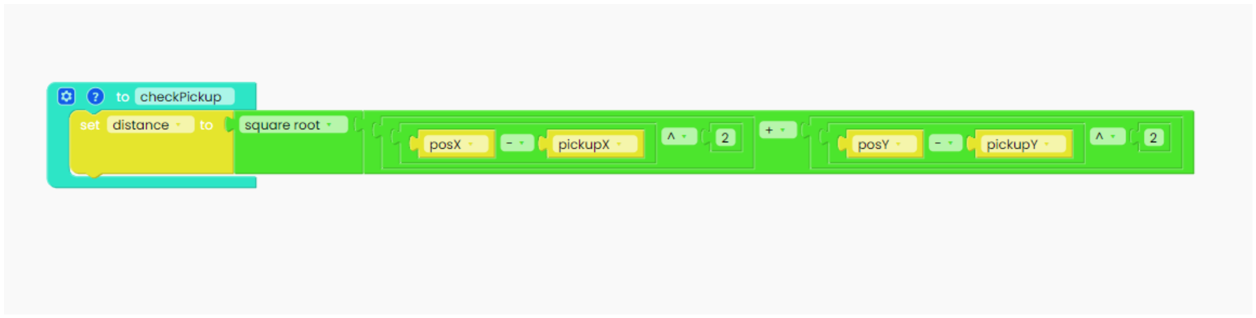
Create a new function, and call it "checkPickup". This function will detect whether the player has collected a ball.



```
to createPickup
    set pickupX to random integer from 10 to 150
    set pickupY to random integer from 10 to 110

to checkPickup

to physics
    set timeDelta to ((ms since started - lastTime) ÷ 1000.01)
    set lastTime to ms since started
    set velocityY to (velocityY + (timeDelta × 60))
    set posX to (posX + (timeDelta × velocityX))
    set posY to (posY + (timeDelta × velocityY))
    if posX < 5
    do  set velocityX to - velocityX
        set posX to 5
    else if posX > 155
    do  set velocityX to - velocityX
        set posX to 155
    if posY < 5
    do  set velocityX to - velocityX

color: Red
```

The first thing we'll have to do with this function is to calculate the distance between the player and the pickup ball. We used the formula you see below.

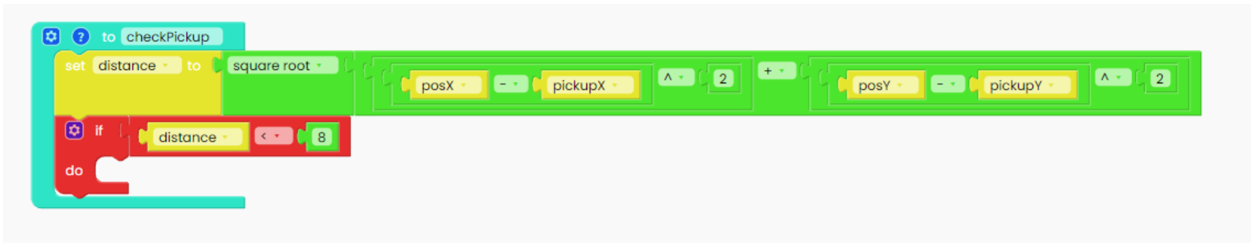
That's the formula you can use for calculating the distance between two points in a 2D space.



```
to checkPickup
    set distance to square root (((posX - pickupX) ^ 2) + ((posY - pickupY) ^ 2))
```

We need to use the "if" block from the Logic section to check if the distance we mentioned is less than 8 pixels.

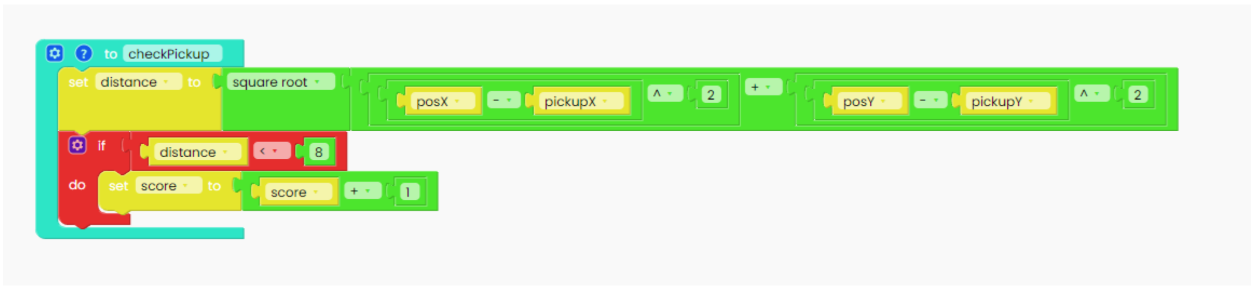
We'll use a comparison block to check that.



```
to checkPickup
  set distance to square root ( posX - pickupX ^ 2 + posY - pickupY ^ 2 )
  if distance < 8
  do
```

If the distance is less than 8 pixels, that means that the player caught the ball, and your scoreboard increases by one point.

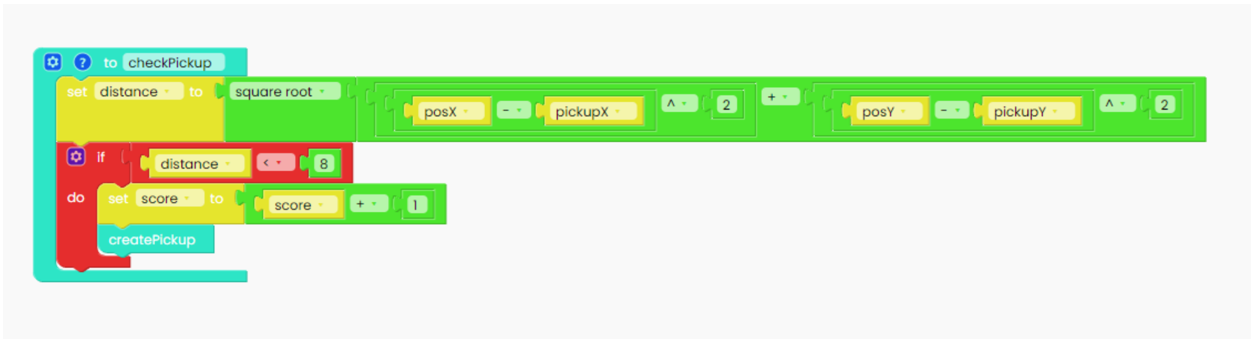
To keep the score, you'll have to take this block:



```
to checkPickup
  set distance to square root ( posX - pickupX ^ 2 + posY - pickupY ^ 2 )
  if distance < 8
  do set score to score + 1
```

So anytime you pick a ball, you'll get one point!

If you want to put your function in action, you need to call it. So, duplicate the "createPickup" block and drag it below.



```
to checkPickup
  set distance to square root ( posX - pickupX ^ 2 + posY - pickupY ^ 2 )
  if distance < 8
  do set score to score + 1
     createPickup
```

The very last thing we'll do for our game is something we have already learned.

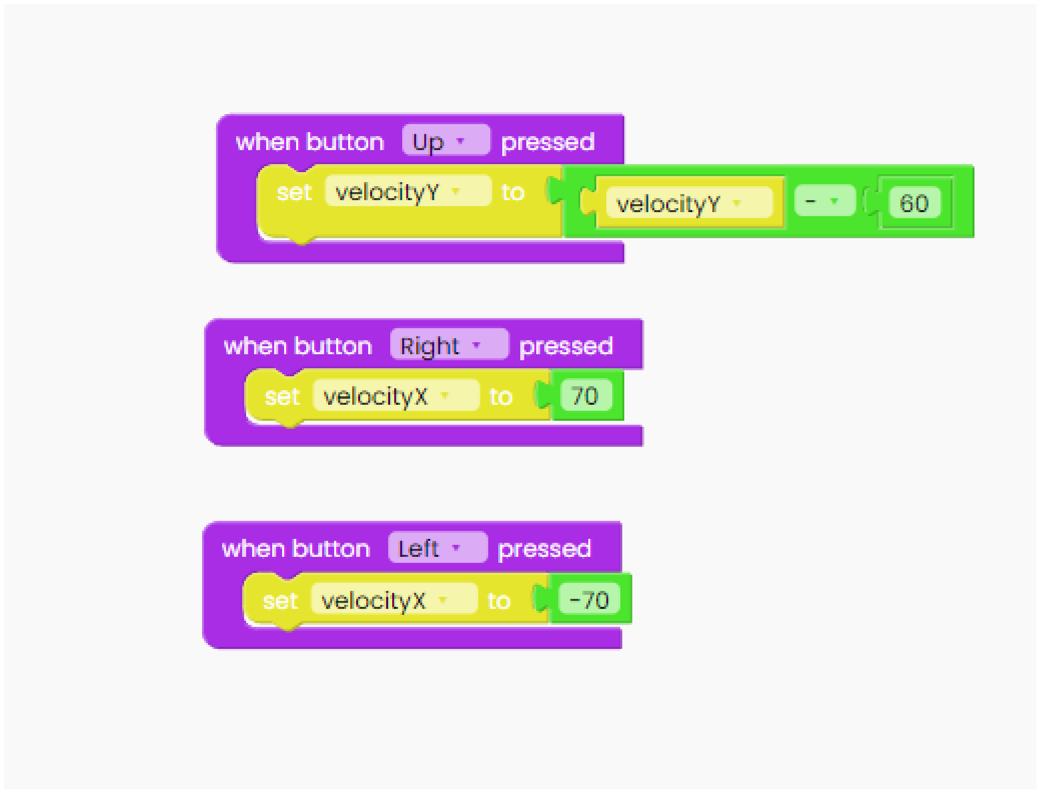
We'll use ByteBoi's pushbuttons to increase the velocity of the ball.

Open the I/O section, and click on the "When button up gets pressed" block.

If the right button gets pressed, velocityX will set to 70 in the positive direction,

and if the left button gets pressed, velocityX will be set to 70 in the negative direction on the x-axis.

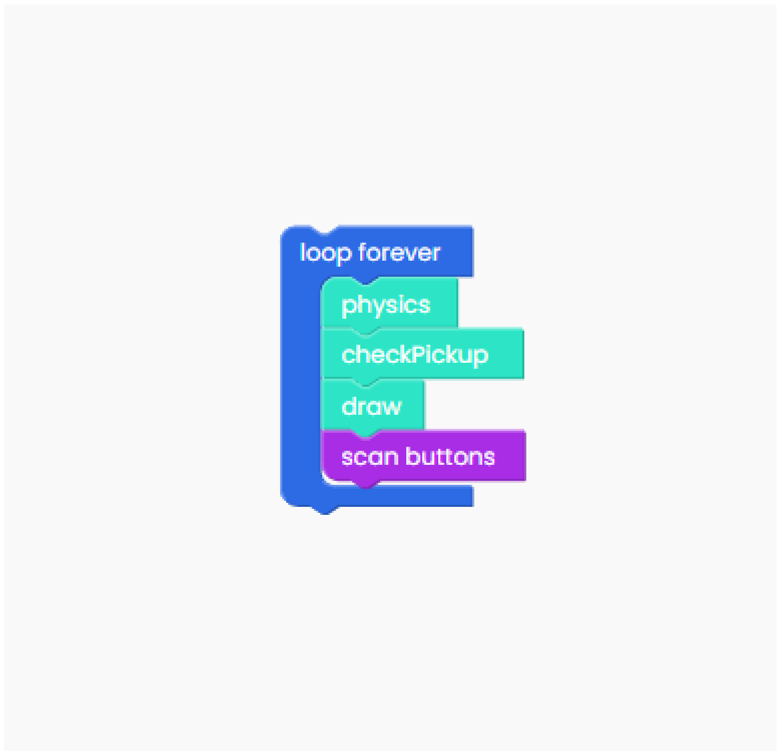
If you press the button up, velocityY will change by 60.



```
when button Up pressed
  set velocityY to velocityY - 60

when button Right pressed
  set velocityX to 70

when button Left pressed
  set velocityX to -70
```

At the end, don't forget to add the "Loop forever" block and inside it place the "physics", "checkPickup" and "draw" variables.

Also, add the "scan buttons" block.


```
loop forever
    physics
    checkPickup
    draw
    scan buttons
```

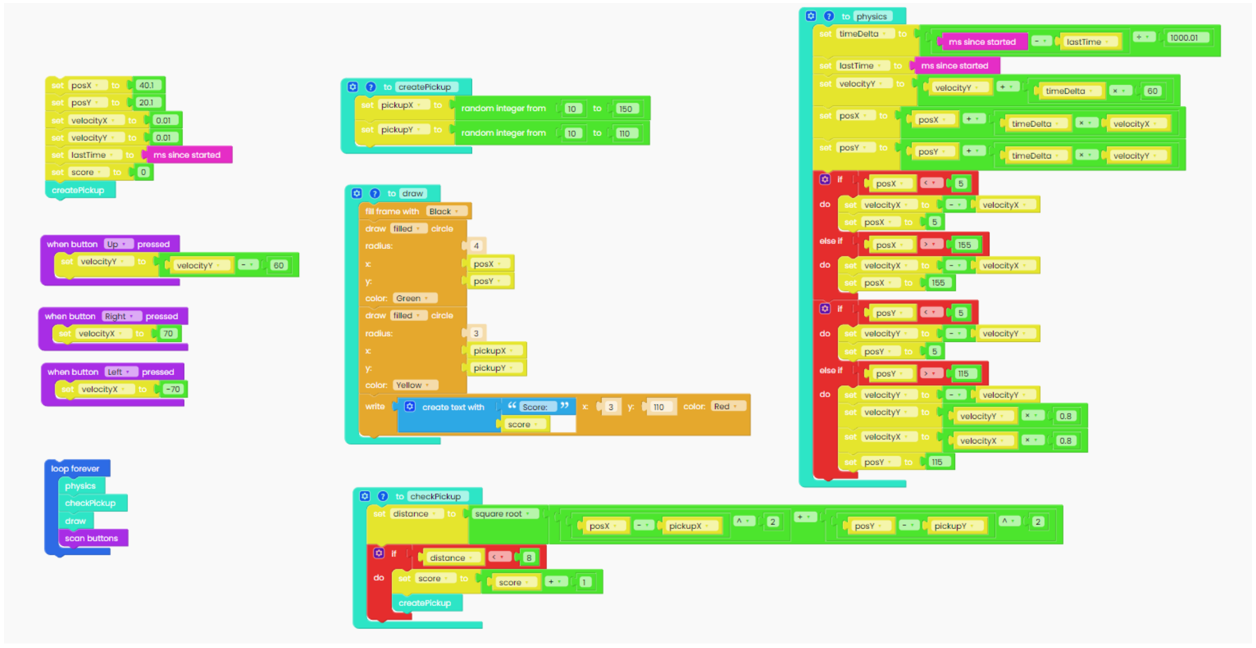
Complete game

This is it!

We tried to explain every part of the game, and now here it is all in one place.

We hope this tutorial has managed to help you make your first steps in video games creation.

Don’t forget to save and name your sketch. Then, click the “Run” button and grab the ByteBoi!



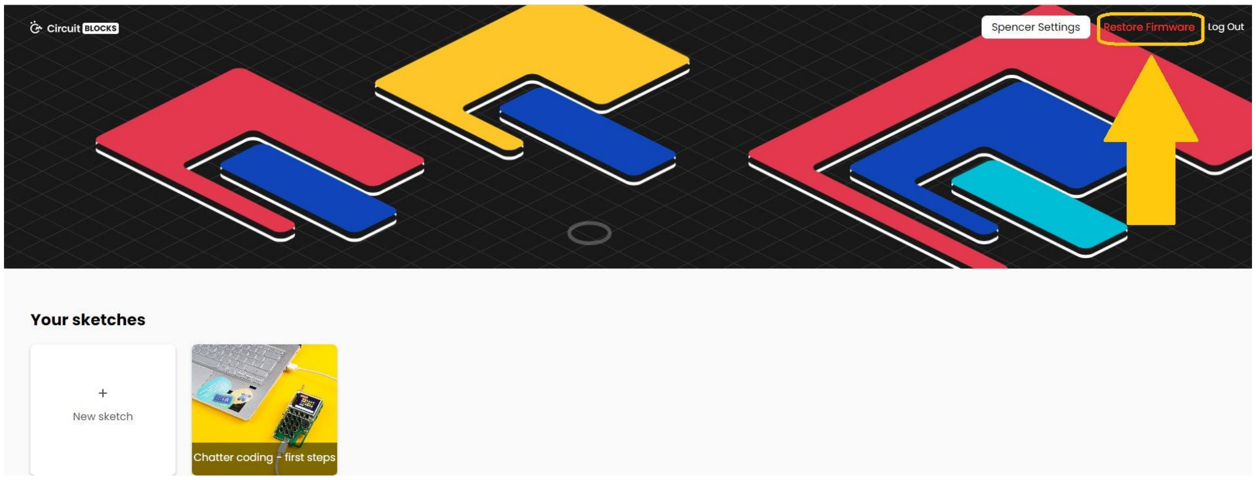
```
set posX to 40.1
set posY to 20.1
set velocityX to 0.01
set velocityY to 0.01
set lastTime to ms since started
set score to 0
createPickup

when button Up pressed
    set velocityY to velocityY - 60

when button Right pressed
    set velocityX to 70

when button Left pressed
    set velocityX to -70

loop forever
    physics
    checkPickup
    draw
    scan buttons

to createPickup
    set pickupX to random integer from 10 to 150
    set pickupY to random integer from 10 to 110

to draw
    fill frame with Black
    draw filled circle
        radius: 4
        x: posX
        y: posY
        color: Green
    draw filled circle
        radius: 3
        x: pickupX
        y: pickupY
        color: Yellow
    write create text with "Score:" score x: 3 y: 110 color: Red

to checkPickup
    set distance to square root ((posX - pickupX) ^ 2 + (posY - pickupY) ^ 2)
    if distance < 8
    do  set score to score + 1
        createPickup

to physics
    set timeDelta to (ms since started - lastTime) ÷ 1000.01
    set lastTime to ms since started
    set velocityY to velocityY + (timeDelta × 60)
    set posX to posX + (timeDelta × velocityX)
    set posY to posY + (timeDelta × velocityY)
    if posX < 5
    do  set velocityX to - velocityX
        set posX to 5
    else if posX > 155
    do  set velocityX to - velocityX
        set posX to 155
    if posY < 5
    do  set velocityY to - velocityY
        set posY to 5
    else if posY > 115
    do  set velocityY to - velocityY
        set velocityY to velocityY × 0.8
        set velocityX to velocityX × 0.8
        set posY to 115
```

Great job!

Restore ByteBoi's firmware

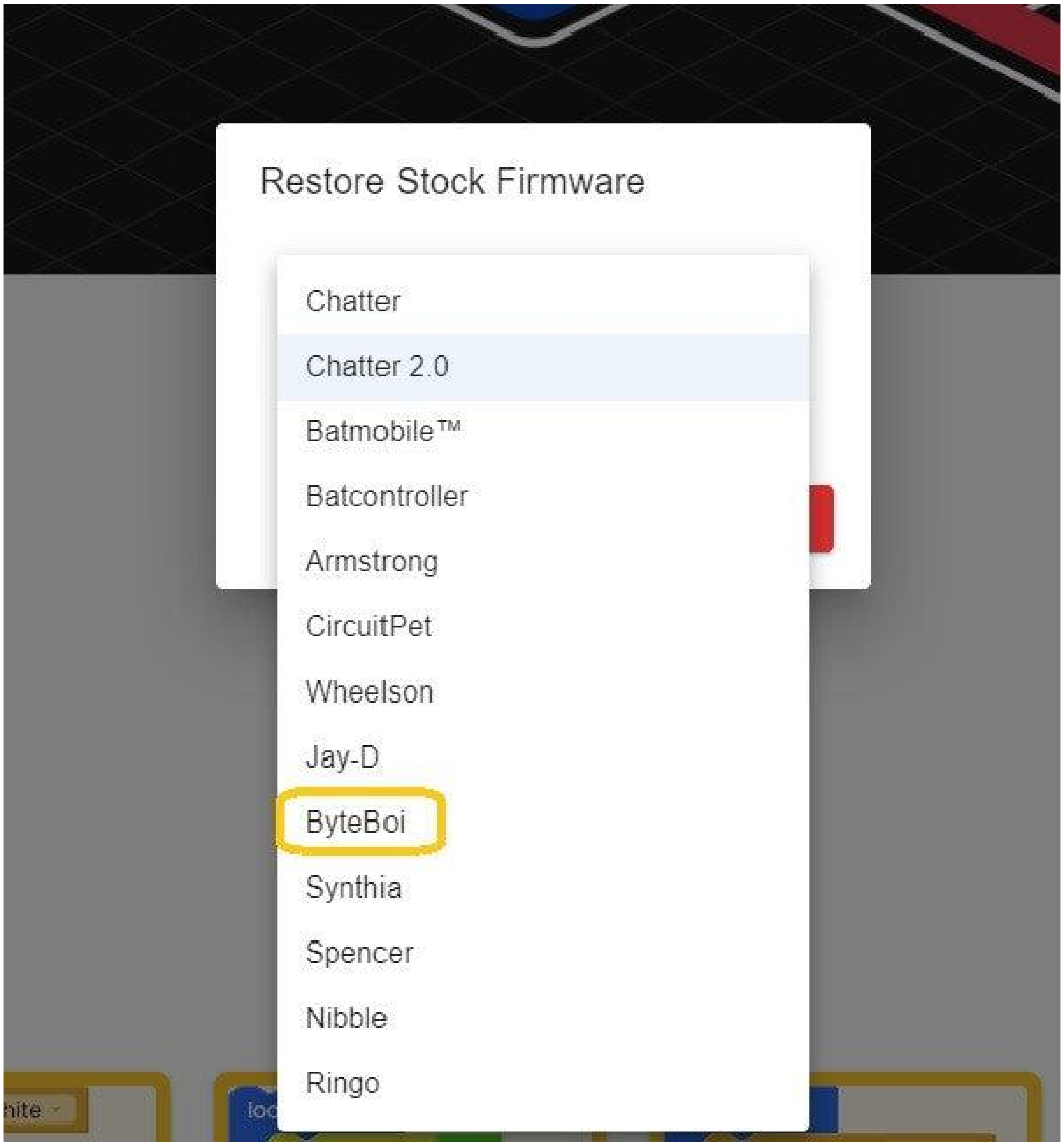
If you, for any reason, want to restore ByteBoi's firmware, follow these steps.

Just connect your ByteBoi to the USB port of your computer and press the "Restore firmware" button on the top right.



You will be prompted with a window to choose the device you are restoring the firmware for.

Choose ByteBoi, of course.



Wait for a few seconds, and your ByteBoi will be back and running like usual.

You need to do this whenever you're done coding your ByteBoi if you want him to revert to his initial out-of-the-box functionality.

What's next?

You've reached the end of our first ByteBoi coding tutorial, congratulations!

We hope you're as excited as we are about ByteBoi's future since there are so many cool things we want to do with it in the future firmware and CircuitBlocks updates.

In the meantime, continue exploring on your own and show us what you've done with your ByteBoi by sharing it on the CircuitMess community forum or via our Discord channel.

If you need any help with your device, as always, reach out to us via contact@circuitmess.com, and we'll help as soon as we can.

Thank you, and keep making!