SREDNJA ŠKOLA VALPOVO

VALPOVO

FUTURE LANGUAGE IS ROBOTIC CODING

2019-1TR01-KA201-077586

SCRATCH

Scratch is a high-level block-based visual programming language. It uses a simple structure of graphical bricks or blocks of computer code that snap and interlock together to build and control sound, music and images. It was developed in the Massachusetts Institute of Technology (MIT) Media Lab in the United States.

Scratch provides a rich learning environment for people of all ages. It allows you to create interactive stories, animations, games, music, and art

Scratch takes its name from a technique used by disk jockeys called "scratching", where vinyl records are clipped together and manipulated on a turntable to produce different sound effects and music. For it is a key element of the Scratch website that members can download other people's computer games and change the details if they wish so. The result is that many of these online projects are remixes of other projects on the website.

Types of Scratch games

1. **Clicker game**: In a clicker game, players click on-screen Sprites to earn points on each successful action. For instance, maybe it's clicking tacos to score points, where the smaller and tougher tacos to click earn more points.
2. **Chase game:** Control a sprite and move it around to score points. Perhaps it's a shark chasing fish and every time it gets in contact with one, points are earned.
3. **Pong game:** In this simple yet fun experience you can create a pong game in Scratch using the mouse to guide the paddle and knock the ball back to the opponent's side.

Availability

Offline

Offline versions of Scratch are available starting with Scratch 1.4. Version 2,0 is recommended if you have a Linux system, or version 3.0 if you have a Mac or Windows 10 machine.

Online

The latest version of Scratch, Scratch 3.0 will work on a PC, Mac or iPad… basically any computing device that can run a recent web browser (e.g., Chrome, Edge, Firefox, Safari) as it is written in HTML5 and JavaScript. The previous online version (Scratch 2.0) worked on computing devices that supported Flash Player. Therefore, it didn't work on iPads.

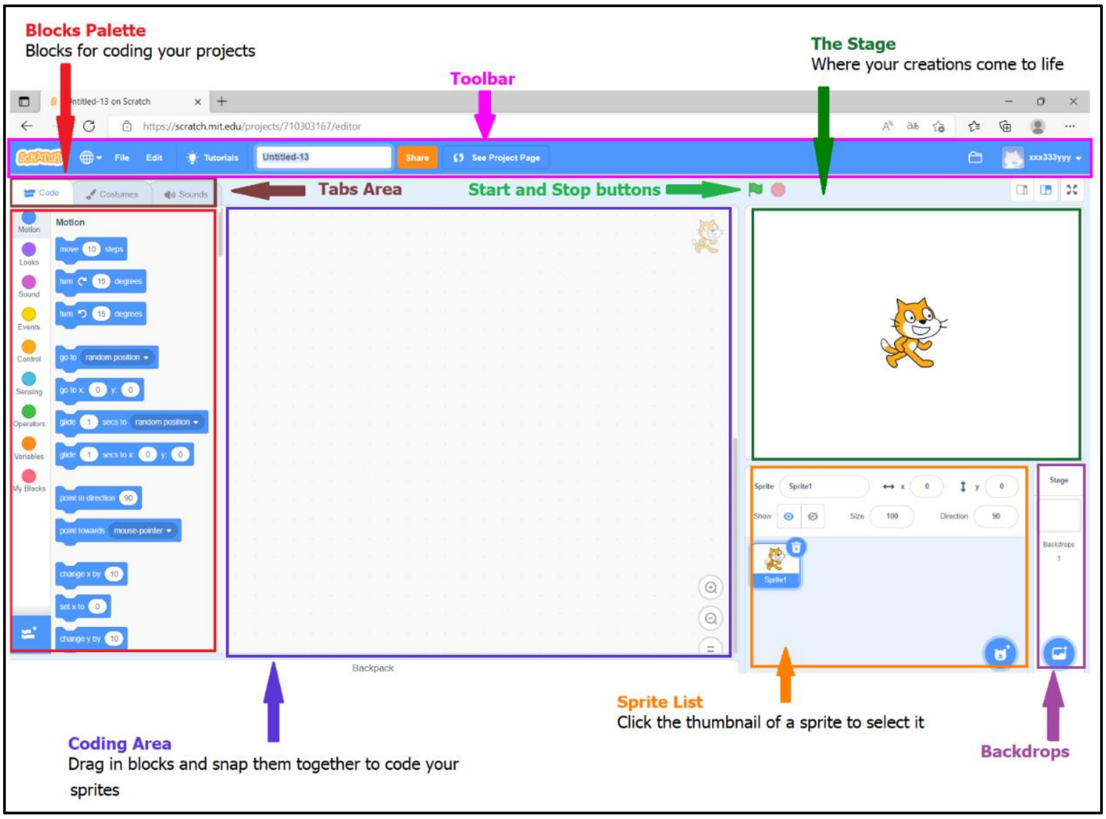
Scratch 3.0

Creating a Scratch account

1. Open a web browser, such as Google Chrome
2. Enter the website address http://scratch.mit.edu/
3. Click Join Scratch in the top-right
4. Make up a username
5. Pick a password and enter it twice. The second time is to make sure you've typed it correctly. Use a mixture of upper and lower case, numbers and symbols to make it more secure.
6. Click the Next button
7. Enter your date of birth, gender, country and email address. This personal information is used to help you recover your password if you forget it.
8. Click the Next button
9. Click OK Let's Go! and you will be logged in. Simply click Create on the navigation bar at the top of the screen, and you're ready to start programming!

Using the Scratch screen

To start using Scratch 3.0, visit http://scratch.mit.edu/ in your web browser and click Create at the top of the screen. This should take you to Scratch's project editor interface:



Blocks Palette
Blocks for coding your projects

Toolbar

The Stage
Where your creations come to life

Tabs Area

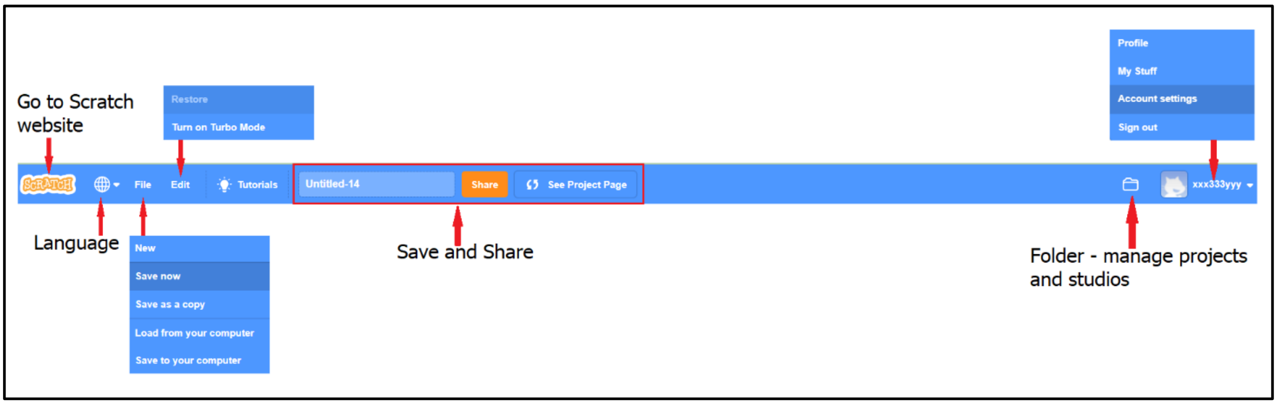
Start and Stop buttons

Coding Area
Drag in blocks and snap them together to code your sprites

Sprite List
Click the thumbnail of a sprite to select it

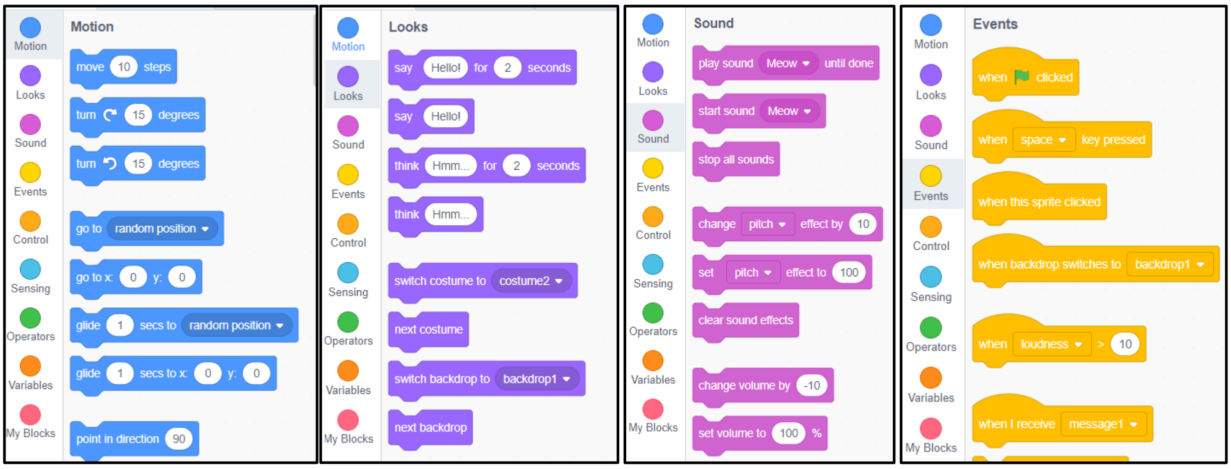
Backdrops

Toolbar

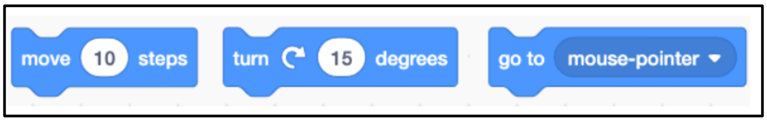


Blocks Palette (The programming blocks)

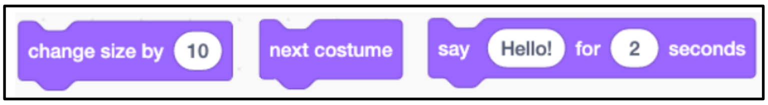
In the Block Palette you find 9 different block categories: ***Motion***, ***Looks***, ***Sound***, ***Events***, ***Control***, ***Sensing***, ***Operators***, **Variables** and ***My Blocks***. Each category has a different color. The coloring helps you to recognize the blocks more easily.



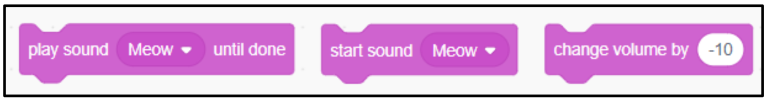
The Motion blocks are blue. This category contains all the blocks to move a sprite. Motion blocks, which control a Sprite's movements—slide, spin, move, and more. For example, this could be "Move 10 steps" to move the Sprite forward 10 steps.



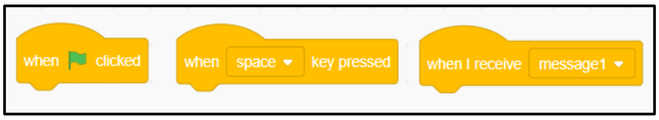
The Looks blocks, which change the appearance of a Sprite, either changing color or size, or even making Sprites say something. For example, "Change size by 10" or "Say Hello! for 2 seconds."



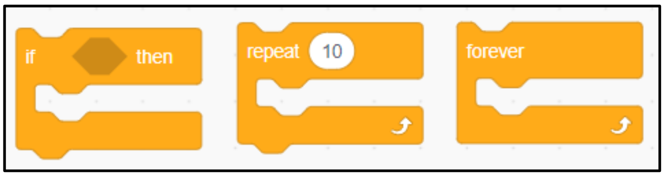
The Sound blocks, which add sound effects to a Sprite—play sounds, edit sound effects, or change the volume. For example, „Play sound meow until done”.



The blocks in the Events category are yellow. You find for example the block when clicked. This type of block is called a hat block and is used at the top of a programming stack. It allows to attach other blocks only below, not above.



Control blocks, which allow for more power over the code, used to create looping animations or pauses between events.



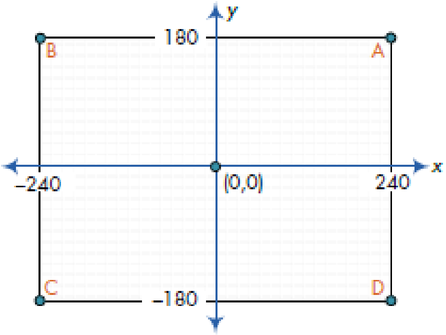
Coding Area

To make a sprite do interesting things, you need to program it by dragging blocks from the Blocks Palette to the Coding Area and snapping them together. Scratch blocks only snap together in certain ways, eliminating the typing errors that tend to occur when people use text-based programming languages.

The Stage

This is where you can see your animations and games in action. When Scratch starts, there’s a large orange cat in the middle of the Stage.

The Stage is 480 steps wide and 360 steps tall. The center of the Stage has an x-coordinate of 0 and a y-coordinate of 0.

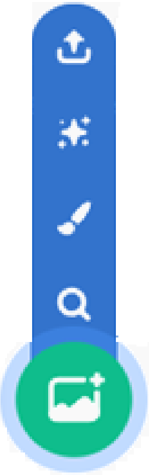


Backdrops

Use this tab to organize the Stage’s background images, which you can change with your scripts. For example, if you’re creating a game, you might show one backdrop with instructions to begin and then switch to another when the user starts the game.

New backdrop:

1. Click the Choose a Backdrop icon
2. Options:



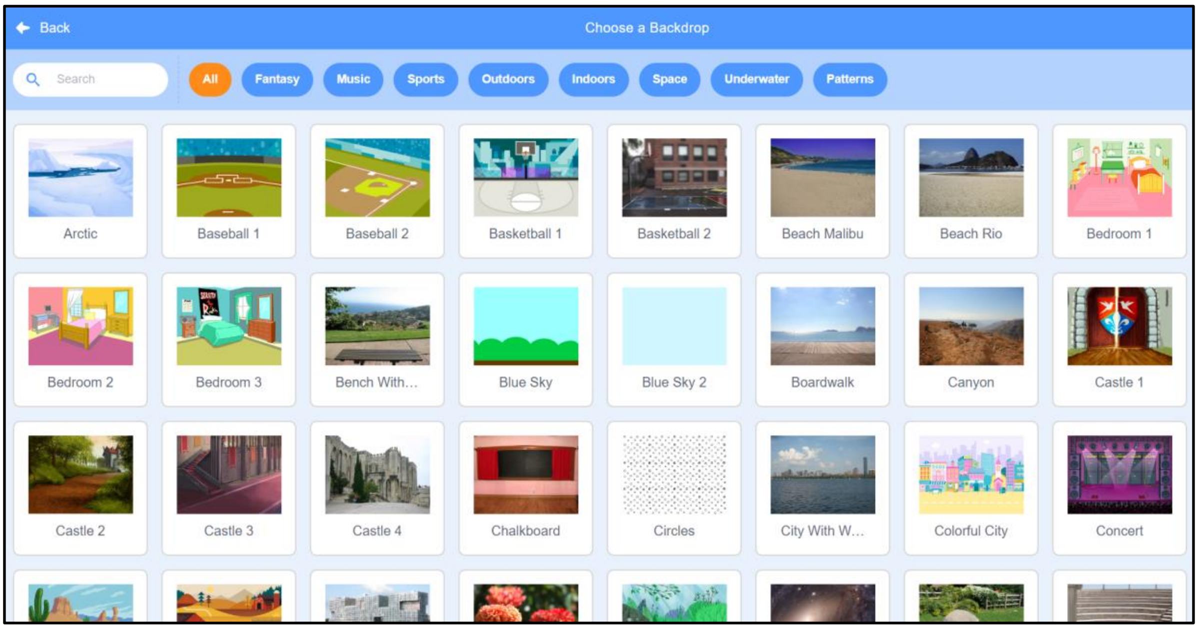
a. Upload an image from your computer.

b. Click for a surprise Backdrop.

c. Draw your own Backdrop.

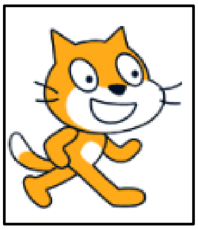
d. Choose a Backdrop from the library.

3. When the library opens, click the themes and categories to view different backdrops available, and use the scrollbar to see more designs. In the Backdrop from the library, you can choose from 8 different categories: ***Fantasy, Music, Sports, Outdoors, Indoors, Space, Underwater, Patterns***.



4. Click the hill image, and then click the **OK** button
5. Your backdrop is added to the Stage.

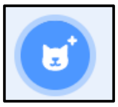
Sprite



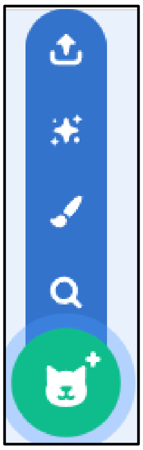
In Scratch, any character or object is called a sprite. Every new project in Scratch starts with the Cat sprite. Your project might include lots of sprites. In the Sprite List, you can see all the sprites that are in your project and click them to switch between them.

Each sprite in your project has its own scripts, costumes, and sounds. When you select a sprite, you can access its scripts, costumes, and sounds by clicking one of the three tabs located above the Tabs Area.

New Sprite



1. Click the New Sprite icon
2. Options:



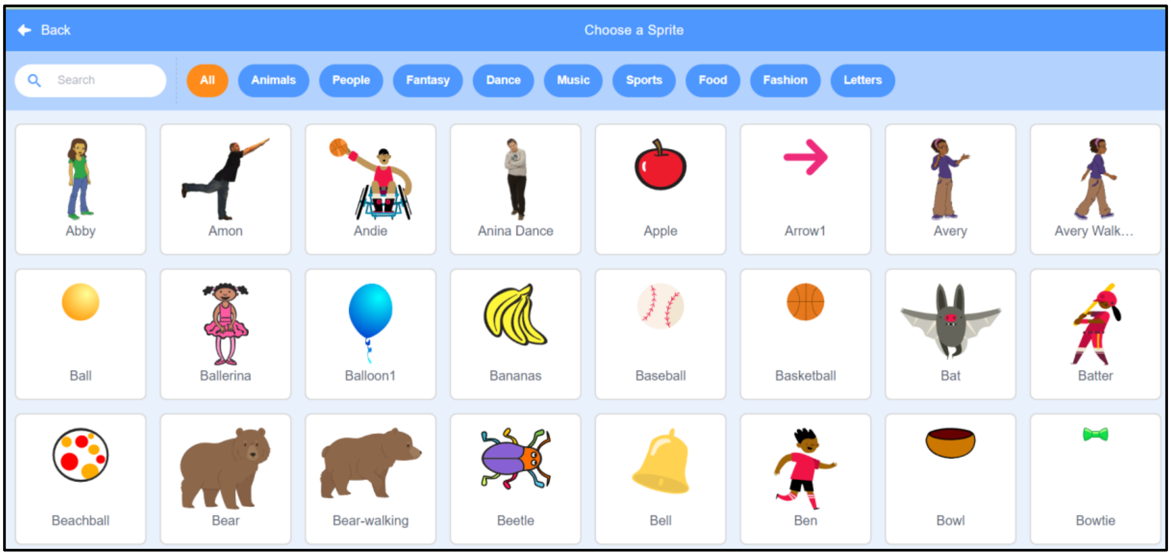
a. Upload an image from your computer.

b. Click for a surprise sprite.

c. Draw your own sprite.

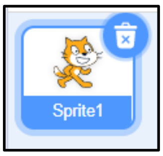
d. Choose a Sprite from the library.

3. When the library opens, click the themes and categories to view different sprite available, and use the scrollbar to see more designs. In the Sprite from the library, you can choose from 9 different categories: ***Animals, People, Fantasy, Dance, Music, Sports, Food, Fashion, Letters***.



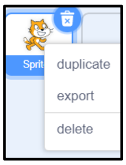
4. Click the hill image, and then click the **OK** button
5. Your sprite is added to the Stage.

Delete a sprite



1. Select the sprite by clicking on its thumbnail in the Sprite List
2. Click on the recycle bin to delete the sprite.

Duplicate a sprite

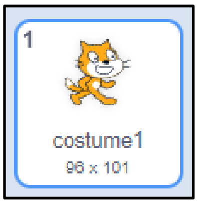


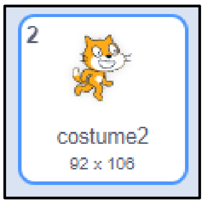
1. Right click on sprite thumbnail in the Sprite List
2. Duplicate

Costumes – Tabs Area

You can change what a sprite looks like by changing its costume, which is just an image. The Costumes tab contains everything you need to organize your sprite's costumes; you could think of it like a clothes closet.

For example, the Cat has two costumes: costume1 and costume2.





Sounds – Tabs Area

Sprites can also play sounds, which liven up your programs. You can, for example, give a sprite different sound to use when it's happy or sad. If your game contains a sprite that looks like a missile, you could make the missile generate different sounds when it hits or misses a target.

The buttons in the Sounds tab will help you organize the different sounds your sprites can play.

How are Scratch sprites coded?

Motion blocks are used to control a sprite's movement, looks blocks are used to change a sprite's appearance, and sound blocks are used to give sprites the ability to make sounds.

How to make a game on Scratch

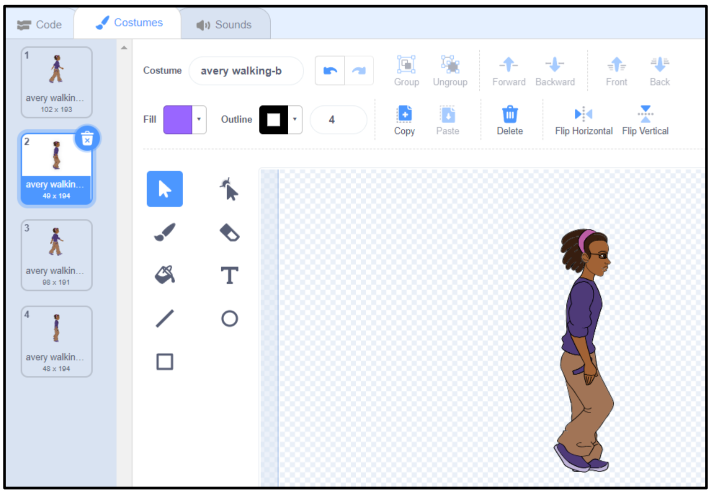
Making a Sprite Move

1. Select a Sprite

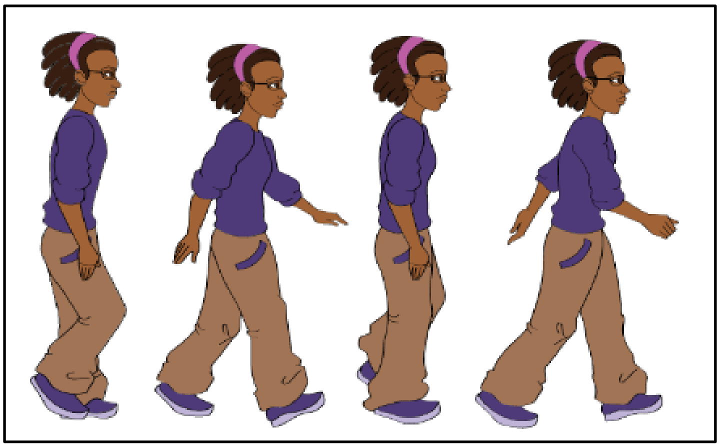
For our project we will use Avery Walking as our first sprite. It is one of the default sprites in the menu. Best to get the sprite that has walking animation. Now that we have chosen a character for our game, we no longer need the Scratch Cat sprite. Wipe it off.

You can later expand your project by adding a cool background and more characters.

When we click on the Costumes in Tabs area, we can see that Avery has 4 different costumes.



Costumes are just different pictures a sprite can have, and Avery has four that show its legs in different positions.

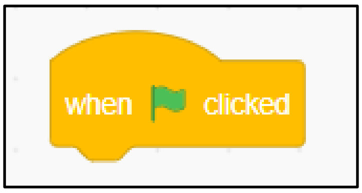


2. Program your sprite

Now that we have a sprite, it’s time to make it controllable. To make your sprite move, we need to use Scratch blocks in order to create a simple script.

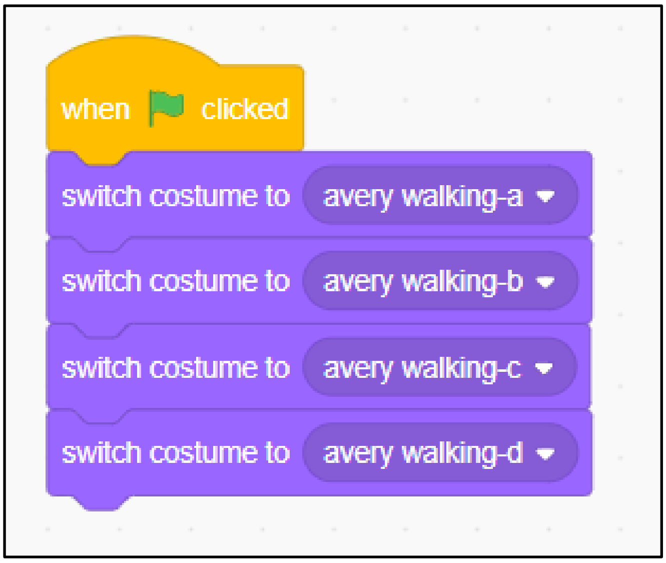
1. Continuous movement left and right across the scene

a) Drag a “when green flag clicked” block from the Events category in the Block Palette into the Code Area.



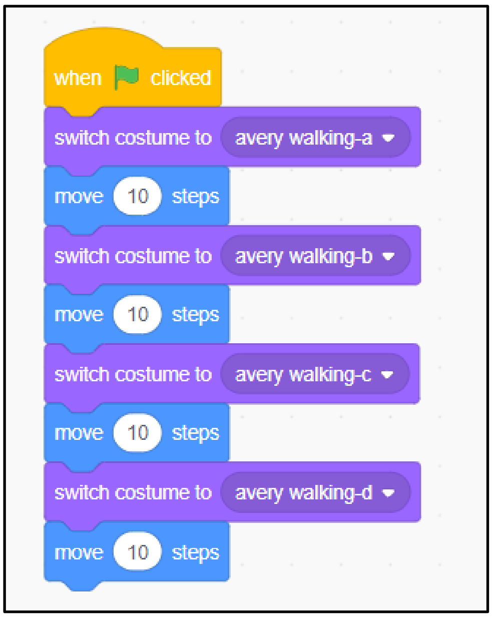
```
when ⚑ clicked
```

b) Avery has four different costumes. You need to drag the "switch costume to..." block four times and choose four different costumes.



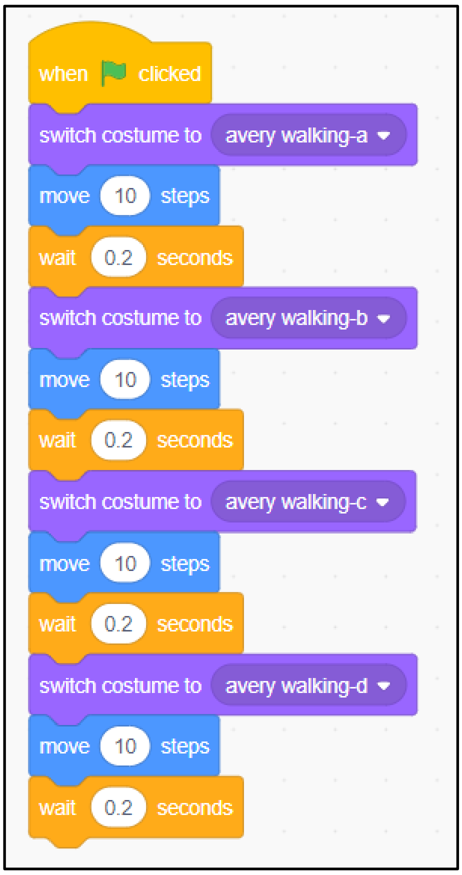
```
when ⚑ clicked
switch costume to (avery walking-a ▾)
switch costume to (avery walking-b ▾)
switch costume to (avery walking-c ▾)
switch costume to (avery walking-d ▾)
```

c) After each costume change block, we add a "Move 10 steps" block. The sprite on the Stage moves in the direction it’s facing to the right. The number of steps is how far across the screen you want the sprite to move. This block only changes the sprite’s position.



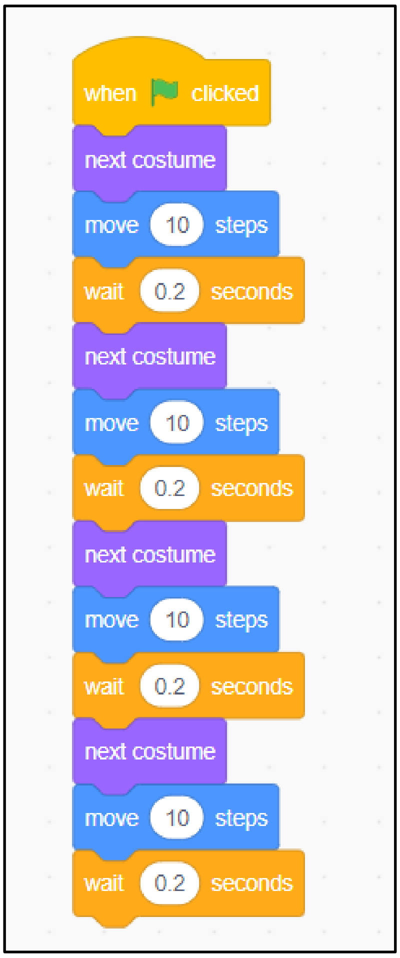
```
when 🏁 clicked
switch costume to avery walking-a ▾
move 10 steps
switch costume to avery walking-b ▾
move 10 steps
switch costume to avery walking-c ▾
move 10 steps
switch costume to avery walking-d ▾
move 10 steps
```

d) If you run the program, you won't see Avery's walking movements. That's why we put the "Wait 1 seconds" block. One second is too much. Avery walks too slowly. Change sizes. I chose 0.2 seconds.



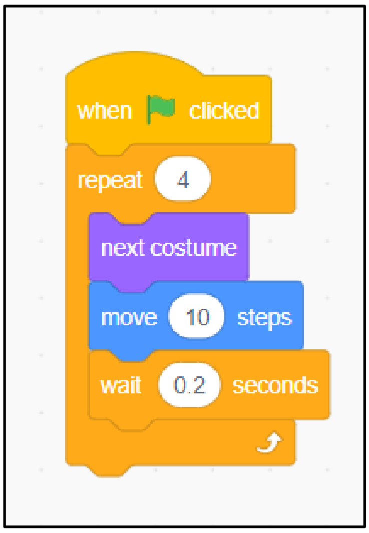
```
when 🏁 clicked
switch costume to avery walking-a ▾
move 10 steps
wait 0.2 seconds
switch costume to avery walking-b ▾
move 10 steps
wait 0.2 seconds
switch costume to avery walking-c ▾
move 10 steps
wait 0.2 seconds
switch costume to avery walking-d ▾
move 10 steps
wait 0.2 seconds
```

e) There is a costume change command that gives the next costume. We can replace the commands "Switch costume to..." with the command "Next costume".



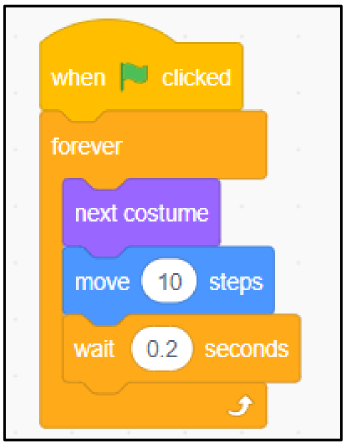
```
when 🏴 clicked
next costume
move 10 steps
wait 0.2 seconds
next costume
move 10 steps
wait 0.2 seconds
next costume
move 10 steps
wait 0.2 seconds
next costume
move 10 steps
wait 0.2 seconds
```

f) The program repeats the same three commands four times. Order a costume change, move and wait. The program will do the same if we put a repeating loop. We will set it to repeat 4 times. We will enter each command only once inside the loop, but in the same order.



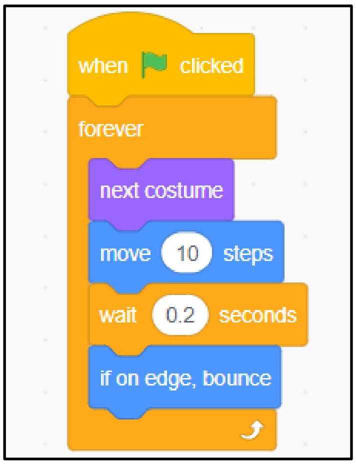
```
when 🏴 clicked
repeat 4
    next costume
    move 10 steps
    wait 0.2 seconds
```

g) We want Avery to continuously walk left and right across the scene. That's why we'll put a forever loop instead of a repeat loop.



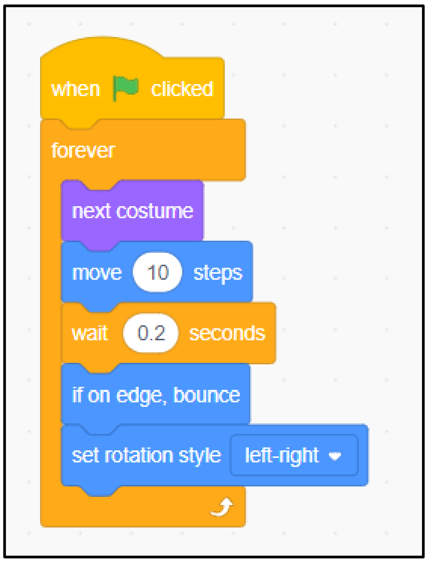
```
when ⚑ clicked
forever
    next costume
    move (10) steps
    wait (0.2) seconds
```

h) Avery remains on the edge of the scene. We want her to turn around and go to the other side. We will achieve this with the "If on edge, bounce" block.



```
when ⚑ clicked
forever
    next costume
    move (10) steps
    wait (0.2) seconds
    if on edge, bounce
```

i) The new problem is turning vertically and not horizontally. To ensure that the cat is standing upright and moving face front, go to the Motion category and add “Set rotation style left-right”.

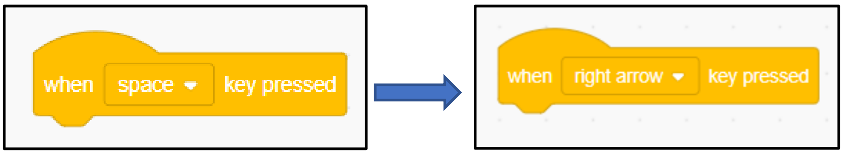


```
when ⚑ clicked
forever
    next costume
    move (10) steps
    wait (0.2) seconds
    if on edge, bounce
    set rotation style [left-right ▾]
```

2. Move Sprites with the Arrow Keys

The task is that when we press one of the arrow keys, the sprite moves in that direction. That's why we'll use the "When ... key pressed" commands.

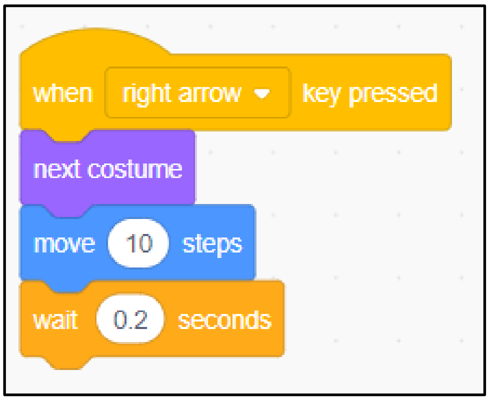
a) Drag a “When space key pressed” block from the Events category in the Block Palette into the Code Area. Instead of the space option, select the right arrow.



```
when space ▾ key pressed
```

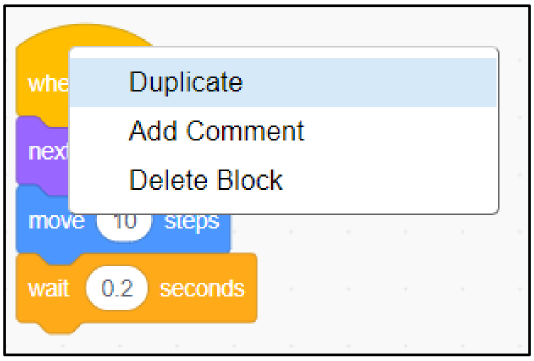
```
when right arrow ▾ key pressed
```

b) We will add the commands we used in the previous example: the command to change costumes, move and wait.



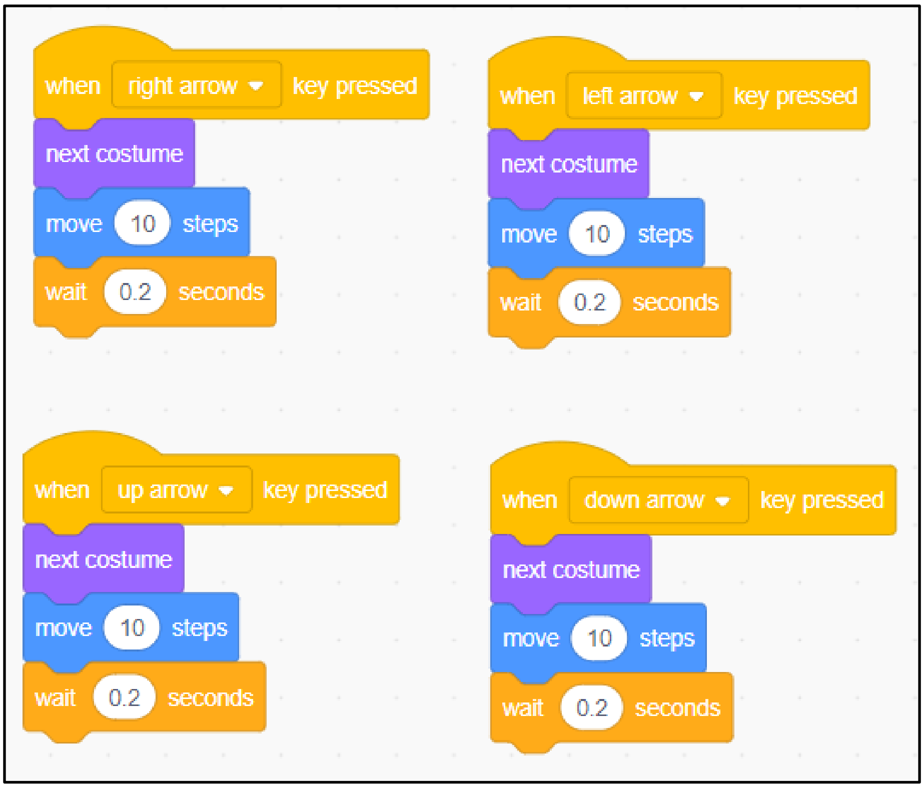
```
when right arrow ▾ key pressed
next costume
move 10 steps
wait 0.2 seconds
```

c) Now that we can move to the right, let’s create more scripts to make our hedgehog move in all four cardinal directions! To make this process faster, you can right click on your existing code and click Duplicate. This will create a copy of these blocks, which means you don’t need to drag and drop as much.



```
whe
next
move 10 steps
wait 0.2 seconds
```

d) Duplicate your code three times by right-clicking and selecting "Duplicate". Then change the options to Left Arrow, Down Arrow, and Up Arrow.



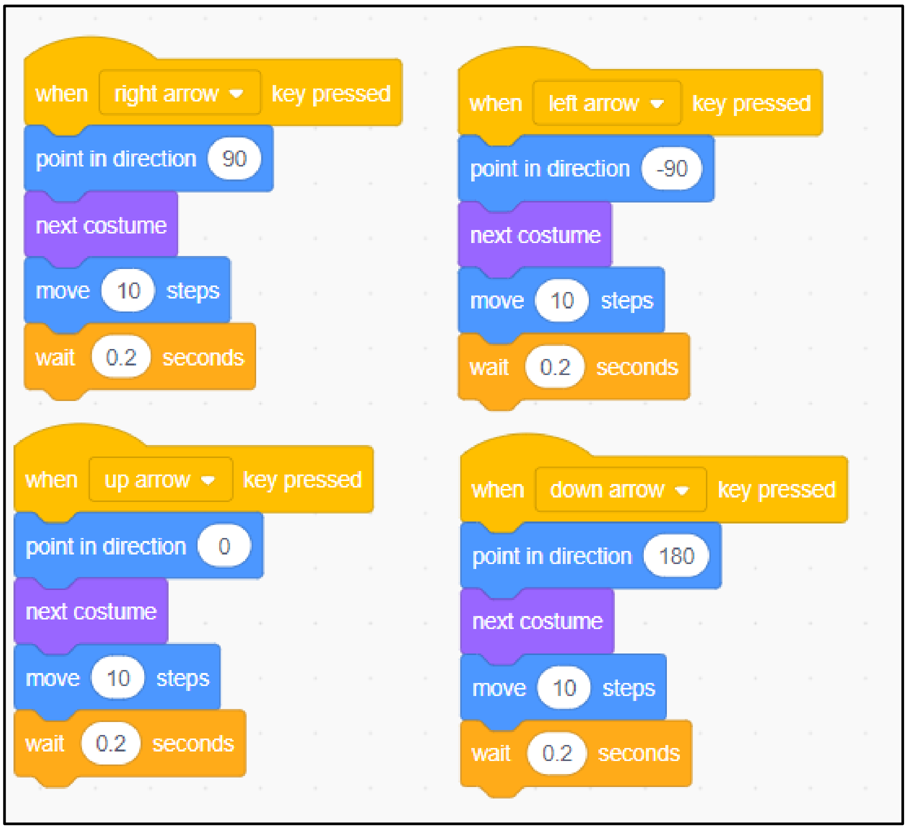
```
when right arrow ▾ key pressed
next costume
move 10 steps
wait 0.2 seconds
```

```
when left arrow ▾ key pressed
next costume
move 10 steps
wait 0.2 seconds
```

```
when up arrow ▾ key pressed
next costume
move 10 steps
wait 0.2 seconds
```

```
when down arrow ▾ key pressed
next costume
move 10 steps
wait 0.2 seconds
```

e) Now we have scripts to move in all four directions. Let's take a look at how our sprite moves in each direction. But we have a problem, the sprite always moves in the same direction. We need to add a command to move in a specific direction. The command is "Point in direction ...". For left it is -90, for right 90, for down 180 and for up 0.



```
when right arrow ▾ key pressed
point in direction 90
next costume
move 10 steps
wait 0.2 seconds
```

```
when left arrow ▾ key pressed
point in direction -90
next costume
move 10 steps
wait 0.2 seconds
```

```
when up arrow ▾ key pressed
point in direction 0
next costume
move 10 steps
wait 0.2 seconds
```

```
when down arrow ▾ key pressed
point in direction 180
next costume
move 10 steps
wait 0.2 seconds
```

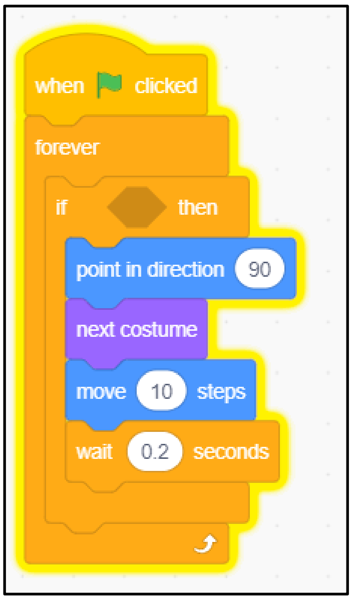
3. More Advanced Movement

Our character moves feel a little clunky. The sprite kind of stutters at first, and it takes a moment to start moving quickly in any direction. Additionally, we can't move diagonally! We are limited to just the four cardinal directions.

Luckily, there is a method to make your sprite move that solves all these problems.

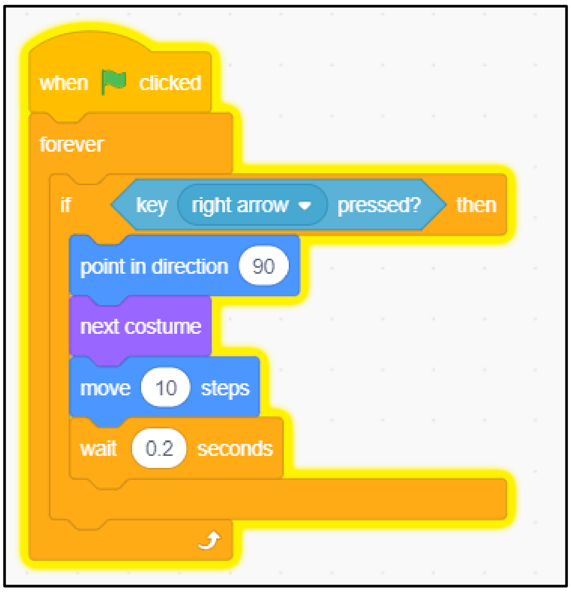
a) We shall be using the:

- "When green flag clicked"
- "Forever loop" and the "If ... then" blocks from the Control section
- Movement commands and the "point in direction" blocks from the Motion section.



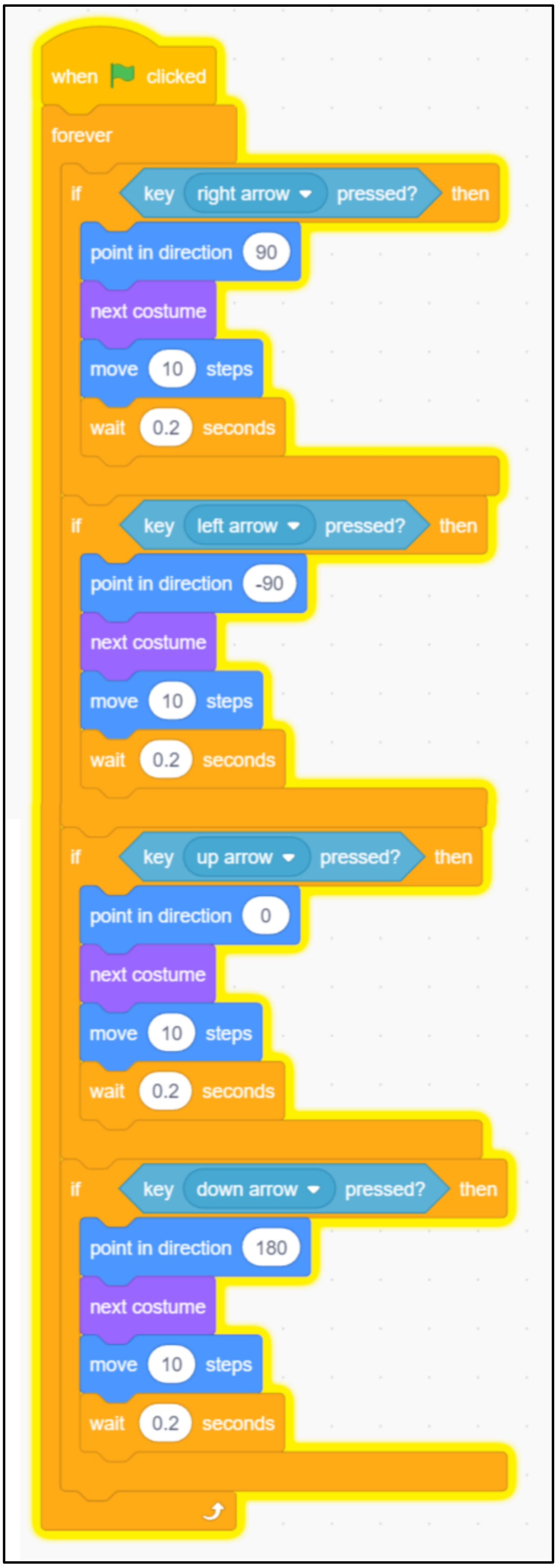
```
when ⚑ clicked
forever
    if ⬢ then
        point in direction (90)
        next costume
        move (10) steps
        wait (0.2) seconds
```

b) In the if loop, add "Key ... pressed" block from the Sensing section. Instead of the space option, select the right arrow.



```
when ⚑ clicked
forever
    if <key (right arrow ▾) pressed?> then
        point in direction (90)
        next costume
        move (10) steps
        wait (0.2) seconds
```

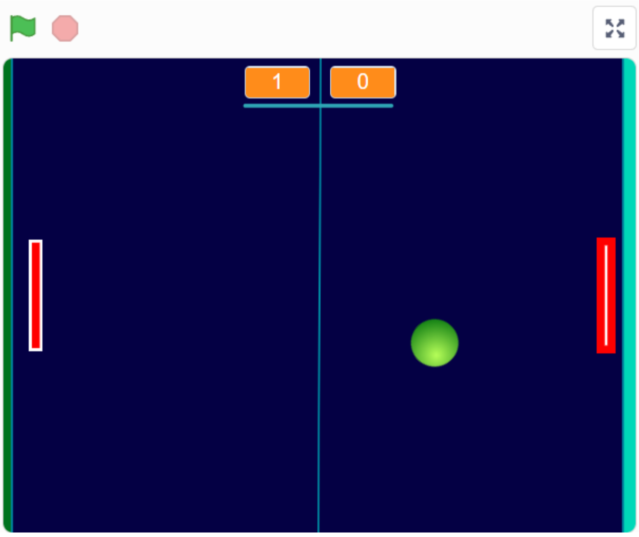
c) Duplicate the if loop three times and place them one below the other. Change the values of the right arrow and "point in direction 90" options.



```
when 🏴 clicked
forever
    if key right arrow ▾ pressed? then
        point in direction 90
        next costume
        move 10 steps
        wait 0.2 seconds
    if key left arrow ▾ pressed? then
        point in direction -90
        next costume
        move 10 steps
        wait 0.2 seconds
    if key up arrow ▾ pressed? then
        point in direction 0
        next costume
        move 10 steps
        wait 0.2 seconds
    if key down arrow ▾ pressed? then
        point in direction 180
        next costume
        move 10 steps
        wait 0.2 seconds
    ⤴
```

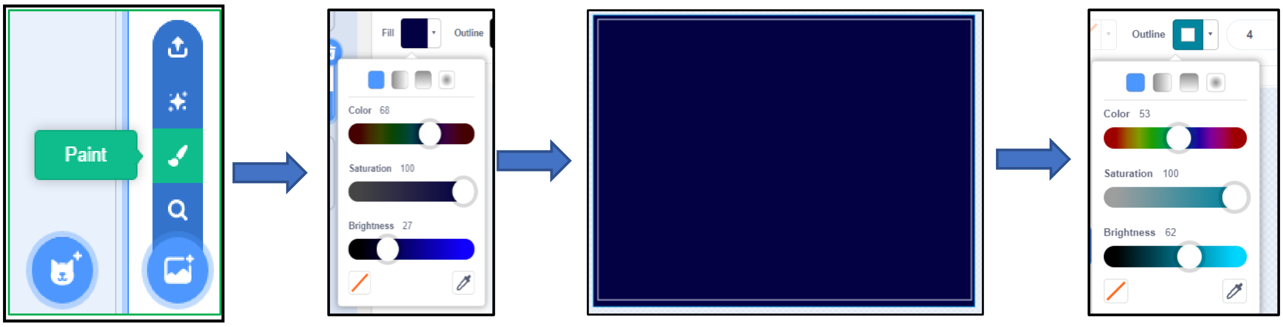
Make a Ping Pong game in Scratch

Pong is a game based on ping-pong. A ball bounces off by three walls and the the player must hit the ball with a paddle.



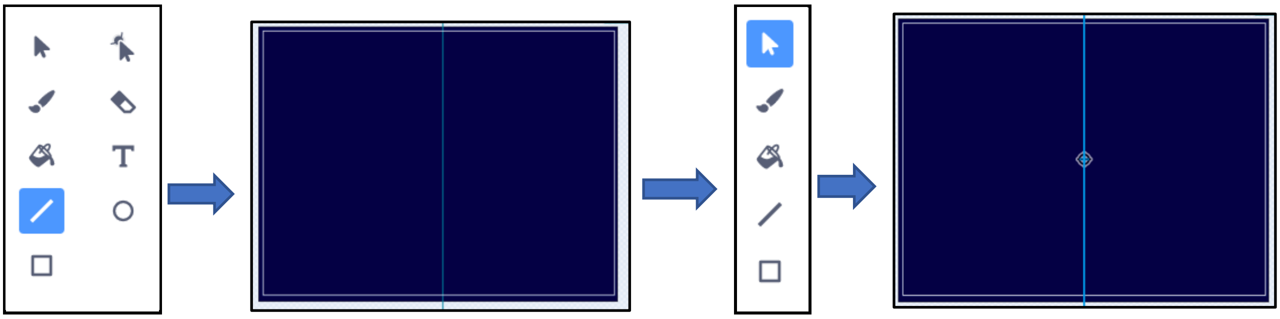
1. **Delete the Scratch Cat sprite**
2. **Choose a Backdrop**

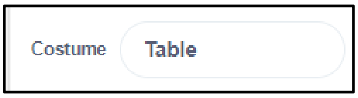
Example:



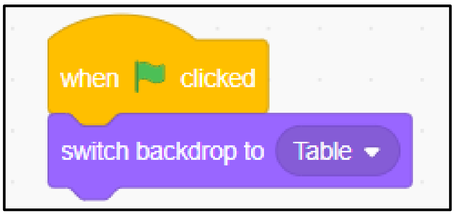
Fill
Outline
Color 68
Saturation 100
Brightness 27

Outline 4
Color 53
Saturation 100
Brightness 62



Change the name: 

Costume Table

Code the backdrop: 

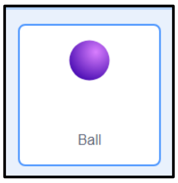
```
when 🏁 clicked
switch backdrop to Table ▾
```

3. Set up the sprites

This game uses 3 sprites:

- ball
- paddle
- text (game over)

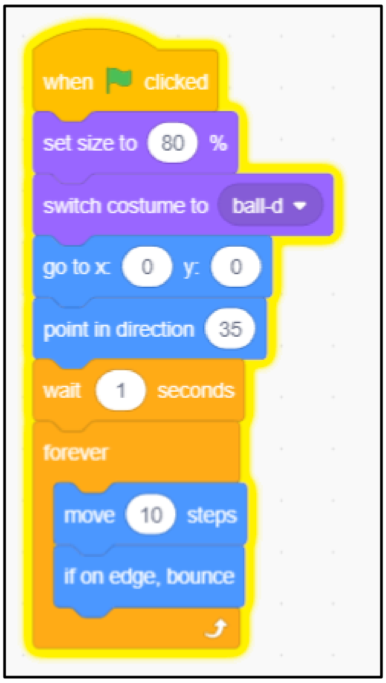
a) The Ball - initial settings and movement



Add a new sprite Ball sprite.

When the program starts:

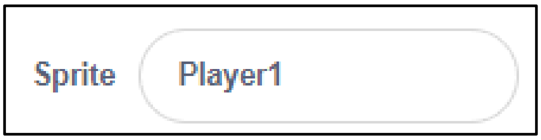
- set ball size to 80%
- switch to costume ball-d (green)
- go to the center position (0, 0)
- point to left and down (35 degrees)
- wait 1 second until the bricks are placed
- enter the forever loop
- move 10 steps and bounce from edges



```
when 🏁 clicked
set size to 80 %
switch costume to ball-d ▾
go to x: 0 y: 0
point in direction 35
wait 1 seconds
forever
  move 10 steps
  if on edge, bounce
```

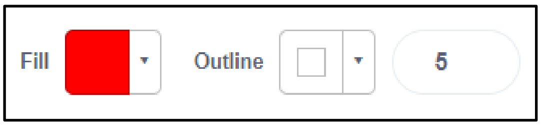
b) The Paddle

Add a new sprite Paddle sprite.
Change the name.



Sprite Player1

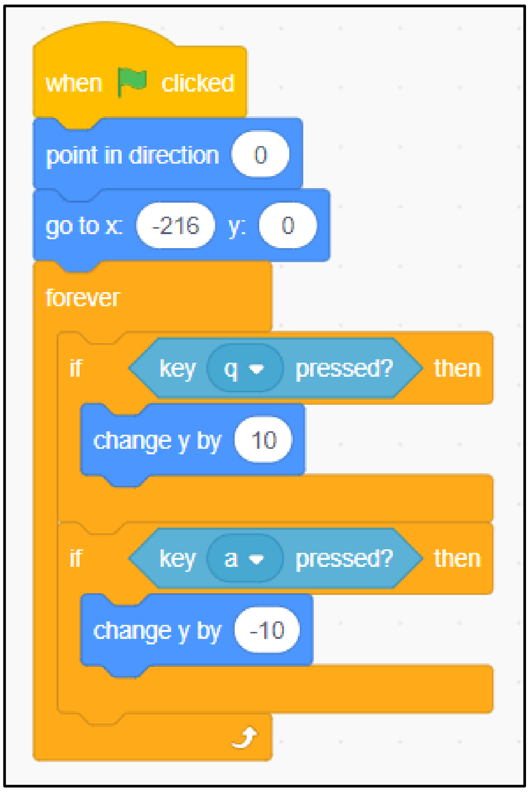
Change the color.



Fill 🟥 ▾ Outline □ ▾ 5

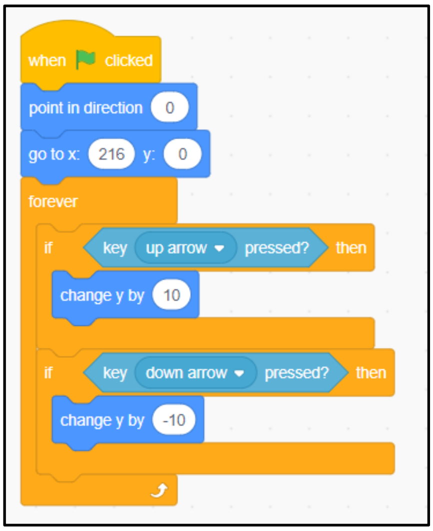
When the program starts:

- set the paddle's Direction to 0
- go to x, y position to -216, 0
- enter the forever loop
- if we press the **q** key on the keyboard, let the paddle go up by 10 (we change y by 10)
- if we press the **a** key on the keyboard, let the paddle go down by 10 (we change y by -10)



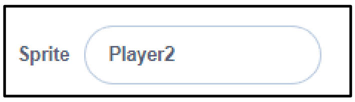
```
when 🏁 clicked
point in direction (0)
go to x: (-216) y: (0)
forever
    if <key (q ▾) pressed?> then
        change y by (10)
    if <key (a ▾) pressed?> then
        change y by (-10)
```

Duplicate the paddle. Change position to 216,0 and key in the up arrow and the down arrow in the if loop.



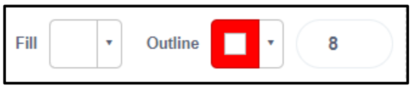
```
when 🏁 clicked
point in direction 0
go to x: 216 y: 0
forever
    if key up arrow ▾ pressed? then
        change y by 10
    if key down arrow ▾ pressed? then
        change y by -10
```

Change the name.

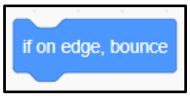


Sprite Player2

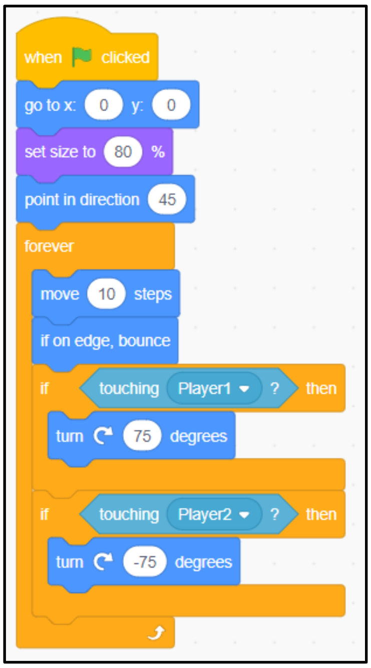
Change the color.



c) The Ball and the Paddle – when the Ball touches the Paddle

Behind the command  if on edge, bounce add:

- if it touches the paddle named Player1, make it rotate 75 degrees
- if it touches the paddle named Player2, make it rotate -75 degrees

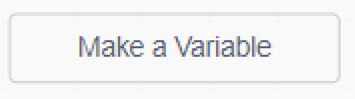


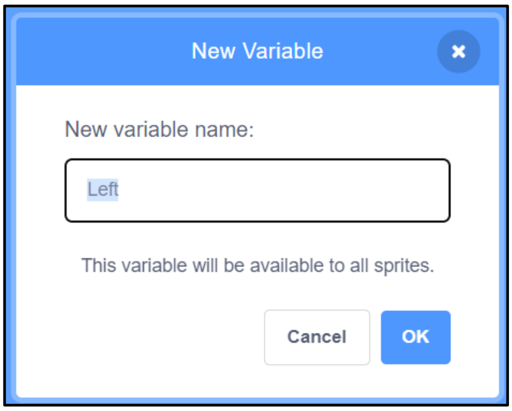
```
when 🏁 clicked
go to x: 0 y: 0
set size to 80 %
point in direction 45
forever
    move 10 steps
    if on edge, bounce
    if touching Player1 ▾ ? then
        turn ↻ 75 degrees
    if touching Player2 ▾ ? then
        turn ↻ -75 degrees
```

4. Scoring

a) Variables

We use variables. We will call them Left and Right.
Creating variables:



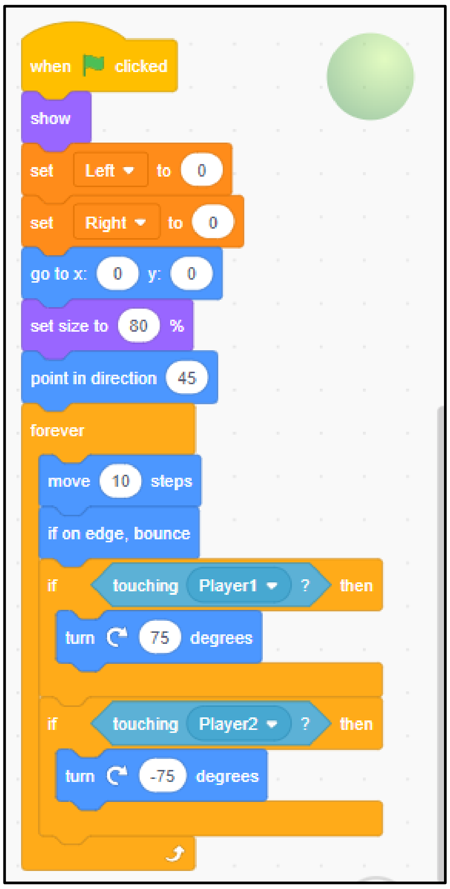


New Variable

New variable name: Left

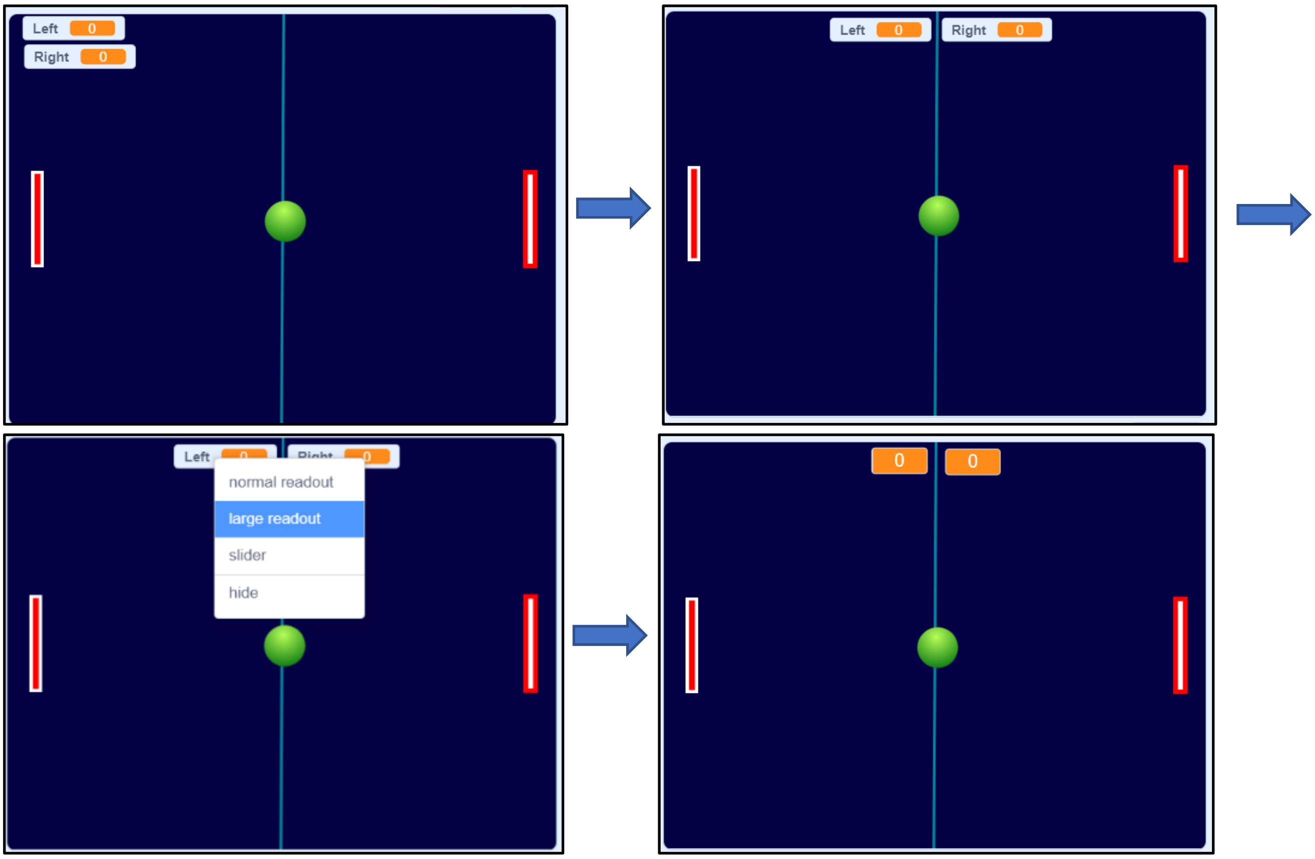
This variable will be available to all sprites.

The initial values of these variables are 0. We will use them when the ball touches the wall. That's why we use them when programming the ball.



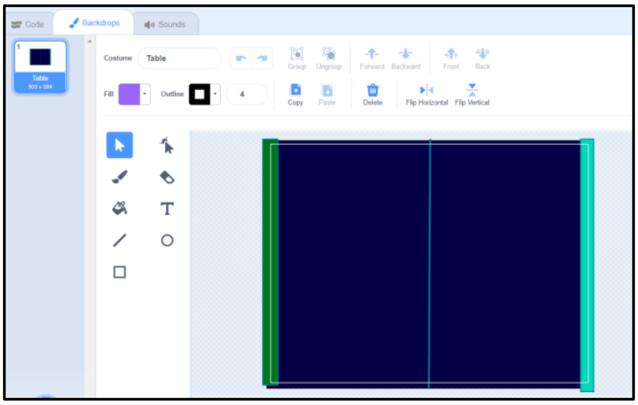
```
when 🏳 clicked
show
set Left ▾ to 0
set Right ▾ to 0
go to x: 0 y: 0
set size to 80 %
point in direction 45
forever
    move 10 steps
    if on edge, bounce
    if touching Player1 ▾ ? then
        turn ↻ 75 degrees
    if touching Player2 ▾ ? then
        turn ↻ -75 degrees
```

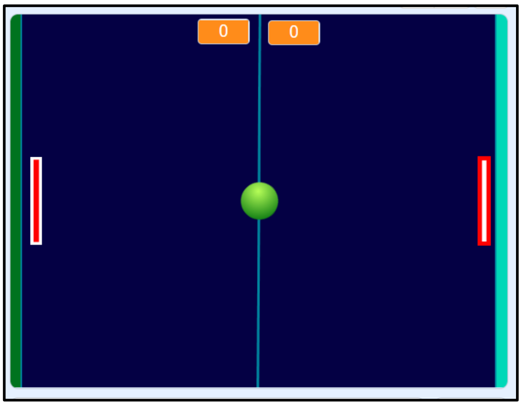
The Stage:



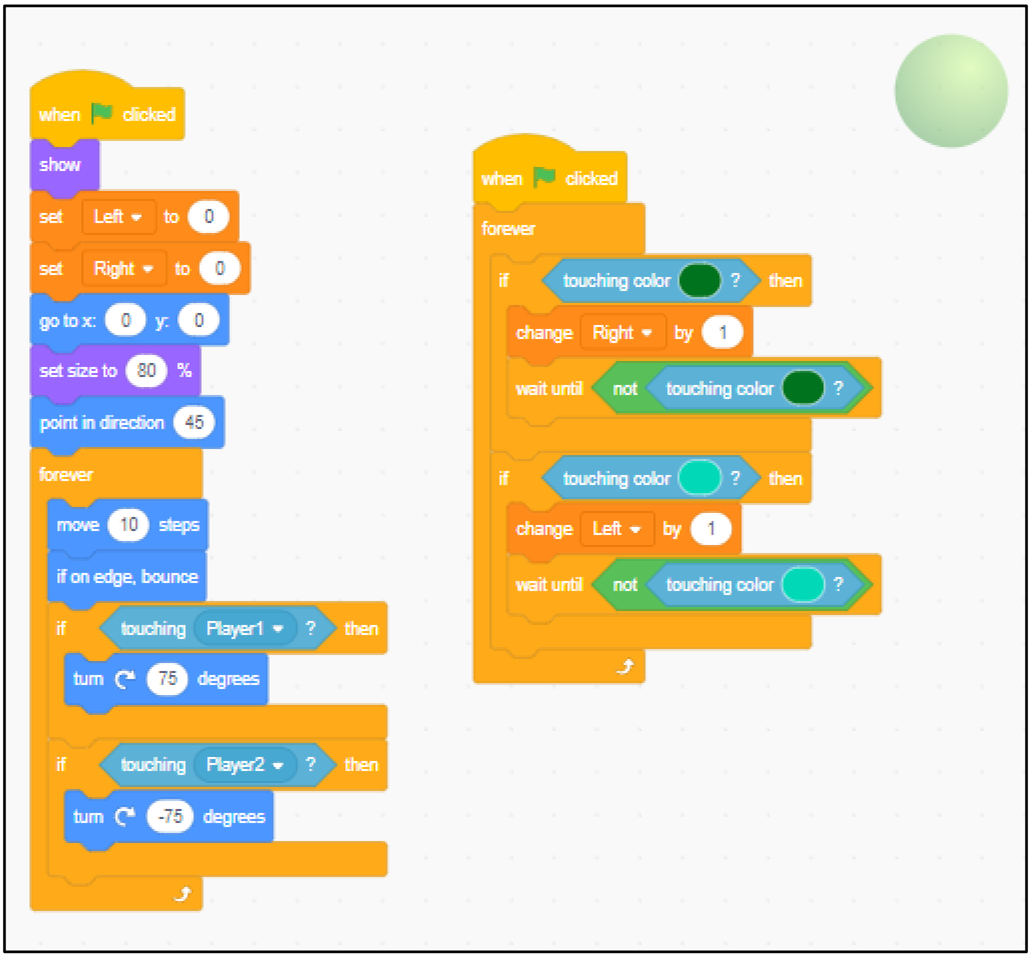
When a player fails to bounce the ball and the ball hits the wall, a point should be added to the other player.

a) We will create rectangles on each wall behind both Paddles. We will paint them in different colors.





b) When the ball touches a certain color, another player gets a point.



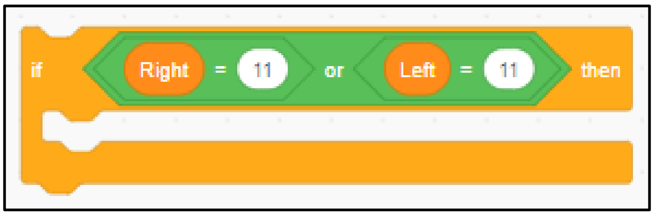
```
when ⚑ clicked
show
set Left ▾ to 0
set Right ▾ to 0
go to x: 0 y: 0
set size to 80 %
point in direction 45
forever
    move 10 steps
    if on edge, bounce
    if touching Player1 ▾ ? then
        turn ↻ 75 degrees
    if touching Player2 ▾ ? then
        turn ↻ -75 degrees
```

```
when ⚑ clicked
forever
    if touching color ● ? then
        change Right ▾ by 1
        wait until not touching color ● ?
    if touching color ● ? then
        change Left ▾ by 1
        wait until not touching color ● ?
```

b) Game Over

When any player's score equals eleven, the game ends. Change the background, hide the Ball and both Paddles and stop everything.

a) Checking points



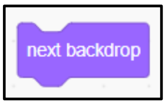
```
if Right = 11 or Left = 11 then
```

b) Change Backdrop

— Make your own Backdrop with the written text: Game Over

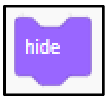
— upload it

— switch Backdrop with code:



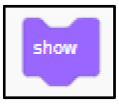
```
next backdrop
```

c) Hide all sprites.



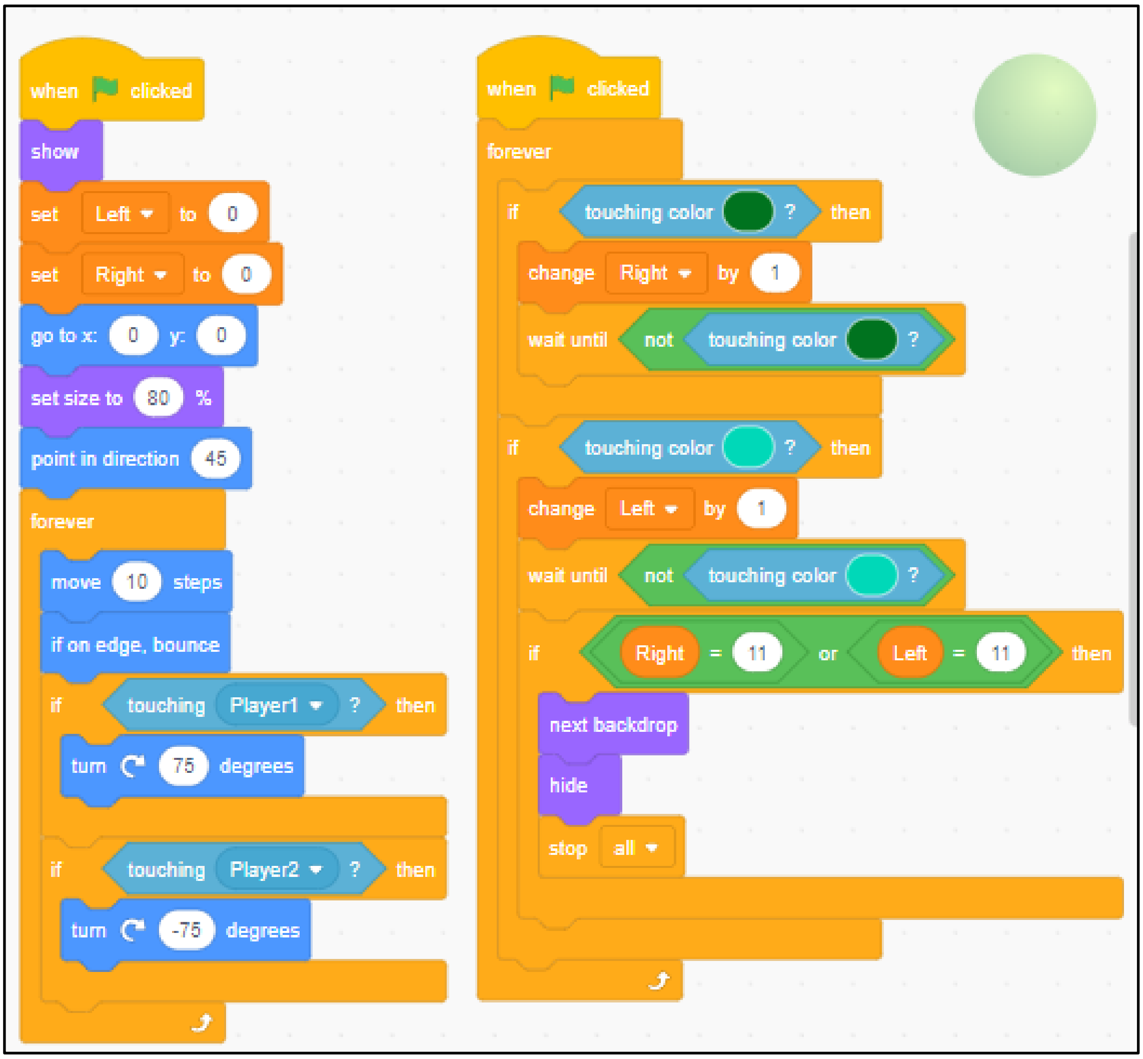
```
hide
```

d) If you are hiding a sprite, you must show it first. Under the block “when green flag clicked”.



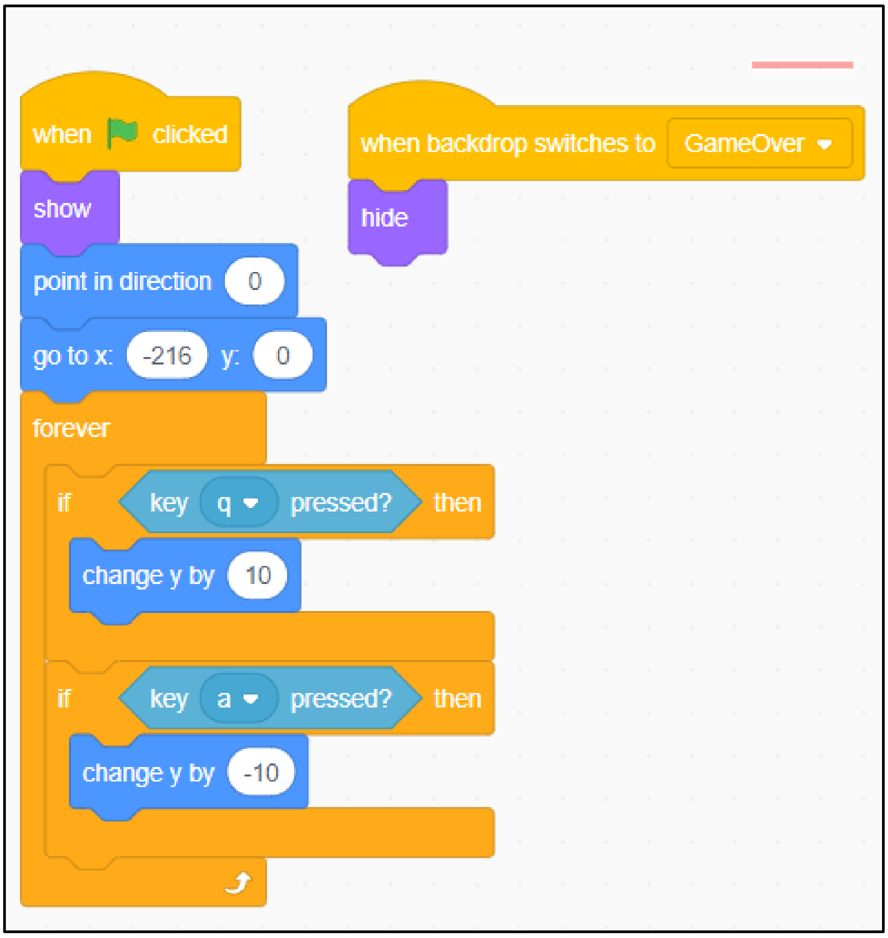
```
show
```

e) Stop All



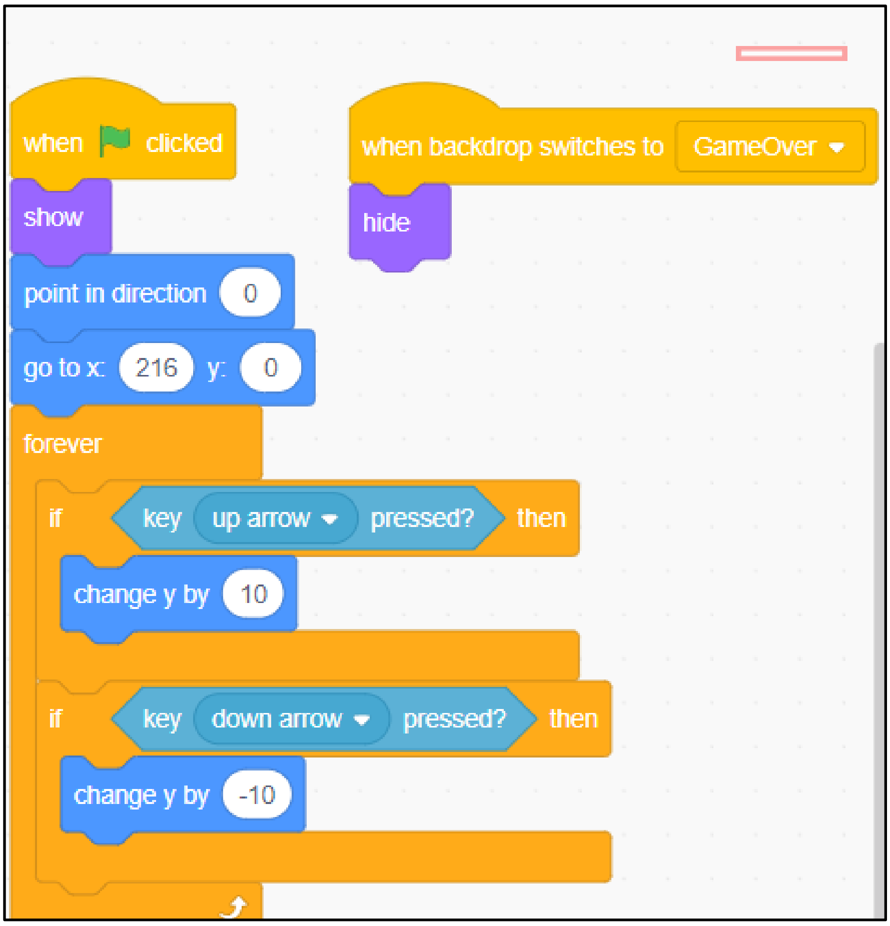
```
when 🏁 clicked
show
set Left ▾ to 0
set Right ▾ to 0
go to x: 0 y: 0
set size to 80 %
point in direction 45
forever
    move 10 steps
    if on edge, bounce
    if touching Player1 ▾ ? then
        turn ↻ 75 degrees
    if touching Player2 ▾ ? then
        turn ↻ -75 degrees
```

```
when 🏁 clicked
forever
    if touching color ● ? then
        change Right ▾ by 1
        wait until not touching color ● ?
    if touching color ● ? then
        change Left ▾ by 1
        wait until not touching color ● ?
        if Right = 11 or Left = 11 then
            next backdrop
            hide
            stop all ▾
```



```
when ⚑ clicked
show
point in direction (0)
go to x: (-216) y: (0)
forever
    if <key (q ▾) pressed?> then
        change y by (10)
    if <key (a ▾) pressed?> then
        change y by (-10)
↻

when backdrop switches to (GameOver ▾)
hide
```



```
when ⚑ clicked
show
point in direction (0)
go to x: (216) y: (0)
forever
    if <key (up arrow ▾) pressed?> then
        change y by (10)
    if <key (down arrow ▾) pressed?> then
        change y by (-10)
↻

when backdrop switches to (GameOver ▾)
hide
```