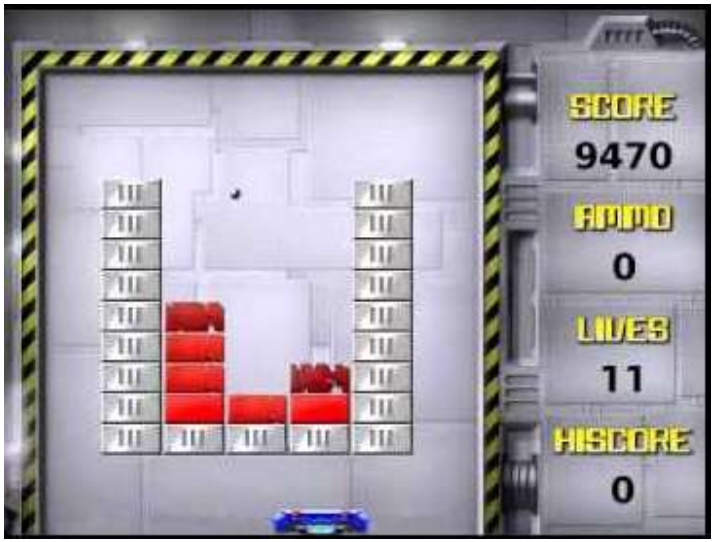
Scratch Brick Game

Project structure:

- ❑ Game design
- ❑ Sprites and backdrops
- ❑ Motion, control and actions
- ❑ Game features
- ❑ Testing and feedback

In this project we will build a version of the Brick game...



The scratch project is called CCC Brick starter project. It gives you lists for 10 levels and the basic frame and brick sprites. You can remix this so you can build from it:
https://scratch.mit.edu/projects/1308409898/

Scratch Brick

Over five sessions we will:

- ❑ Think about the design for our game
- ❑ Design sprites for the characters
- ❑ Code our sprites so they behave as we want
- ❑ Add game features like scores and sound effects
- ❑ Test each others' games and give feedback

We will use many feature of Scratch, including: backdrops, sprites and costumes, coordinates and animation, clones, broadcasts, variables, sensing and sound effects!

Scratch Brick

Game design

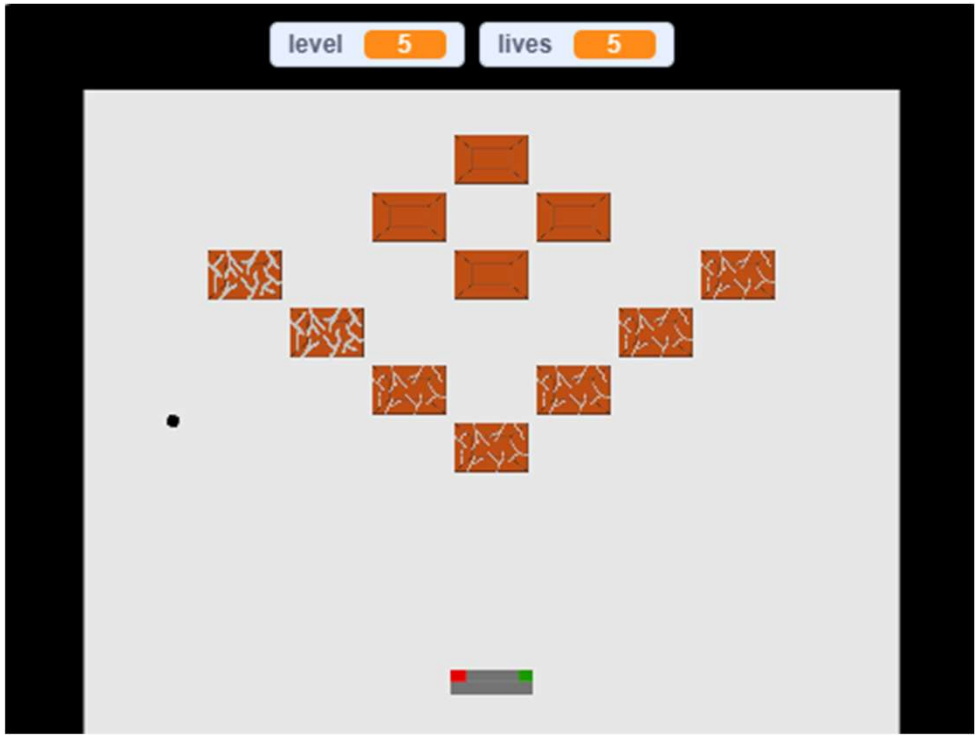
Scratch Brick

Game design:

Before we start to design and code, we should think about how we want our game to play.

What sprites do we need? What can they do? How will they interact with each other? How do you win or lose the game?

Always remember to save your work as you go!



Scratch Brick

Sprites and Backdrop

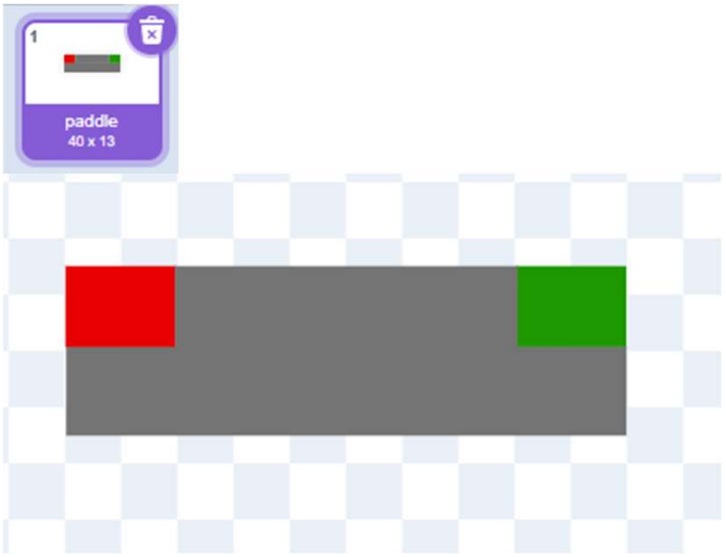
Scratch Brick

Sprites: Paddle

The paddle is used to play the game by bouncing the ball towards the bricks.

My paddle is very simple. The coloured areas at the corners are to help us control how the ball will behave as we'll see later.

Of course you can make the sprites look the way you want. Don't worry too much about the size as we can always adjust the size when we add the sprite to the game.

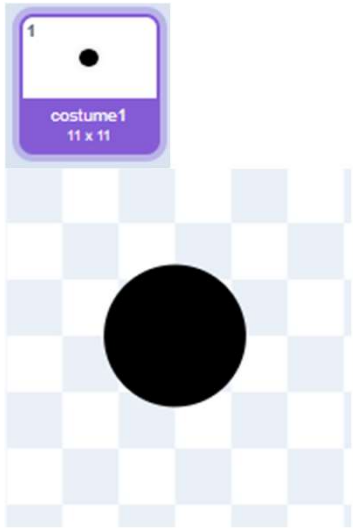


Scratch Brick

Sprites: Ball

The ball will have some quite complicated code so that it knows how to bounce off the paddle, the frame and the bricks as we build the game, but the sprite is very simple!

You can use a different sprite if you like.

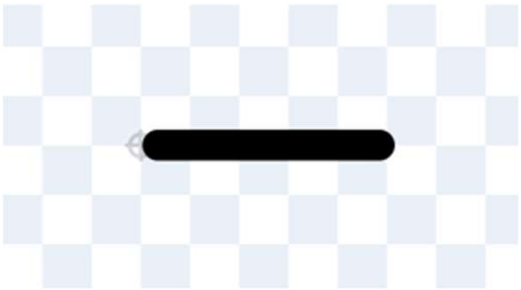


Scratch Brick

Sprites: Pointer

The pointer is used by the paddle when it fires the ball at the start of play to send the ball in the direction the player wants.

Look where the pointer sprite is placed with the centre guide at one end.



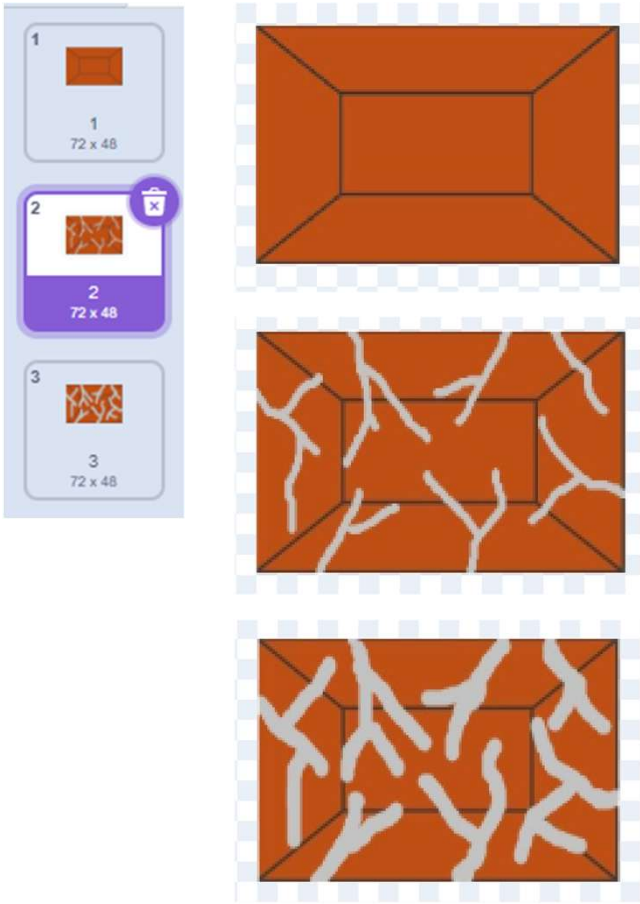
Scratch Brick

Sprites: Bricks

We'll add the bricks later, but we can create the sprite for them now if you like.

The bricks have several costumes so that they change each time they are hit by the ball until they eventually disappear.

You can design your bricks and add different types. I've used three in my game, with a costume for each. The first is the regular brock, then I add cracks after the brick is hit by the ball...



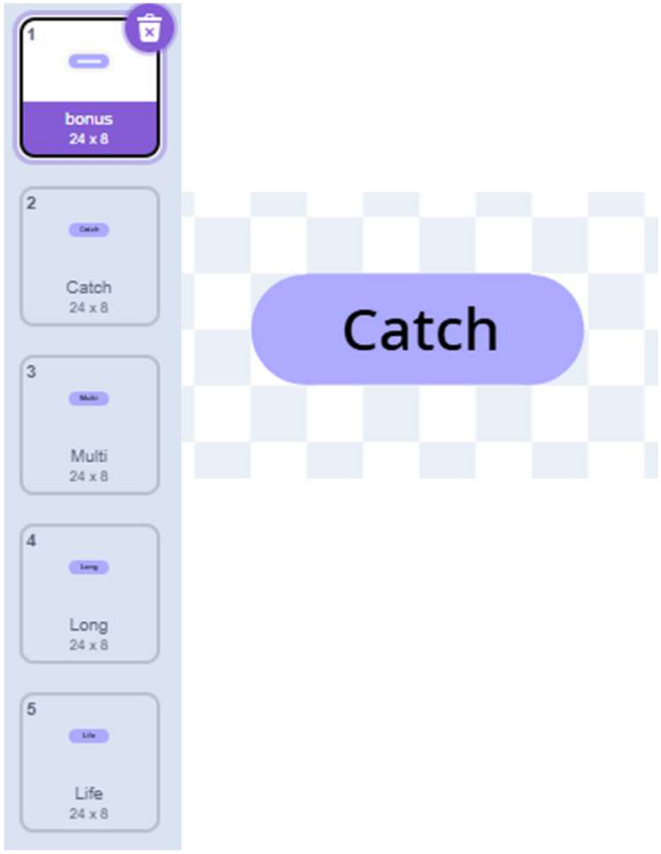
Scratch Brick

Sprite: Bonuses

In the game, when a brick is struck by the ball, it may release a bonus canister which drops down the screen. If the paddle catches the bonus canister it gets the bonus.

We'll add the code for this later on once the basic game is working, so you can wait to make these sprites later if you prefer.

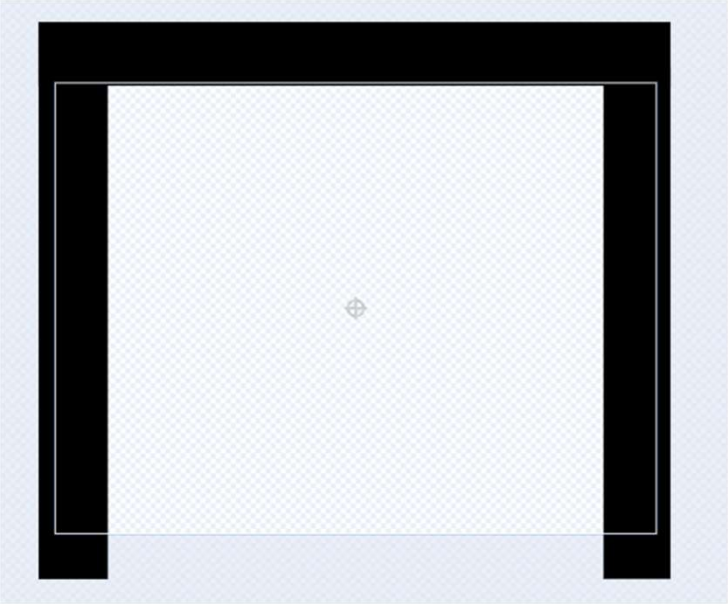
The bonus sprite has a costume for each type of bonus we think of, and we can add as many as we want.



Scratch Brick

Sprite: Frame

We need a simple frame that sets the playing area. The frame should cover the top and sides, but the bottom should be open. I used three rectangles to build my frame.



Scratch Brick

Backdrop

We also need to make backdrops for the Stage.

I have one plain backdrop that we'll use while the game is being played, then versions for completing a level, completing the game and losing the game.

You can use the duplicate tool in the costume editor to create the different versions.

Scratch Brick

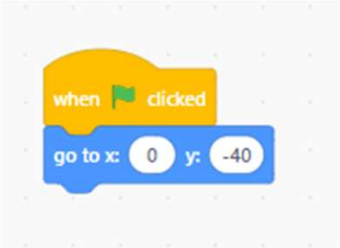
Coding sprites

Scratch Brick

Frame

Now we are ready to start coding. We'll start by placing the frame on the screen.

When the game begins, the frame should be placed on the screen. You can adjust the coordinates until it's in the right place.



Scratch Space Invaders

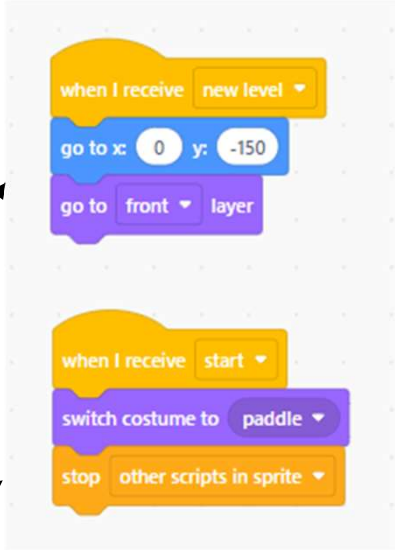
Paddle

Now we are going to place the paddle on the screen and code the controls to make it move.

We need some game controls to make the game work. We can create two broadcast messages called ‘new level’ and ‘start’. We will code how these broadcast messages are sent later.

When the paddle receives ‘new level’ it goes to its starting position near the bottom of the screen.

When the paddle receives ‘start’, we make sure it is in the correct costume (we’ll add a different costume later when we add bonus features) and that there aren’t any other scripts running.



Scratch Brick

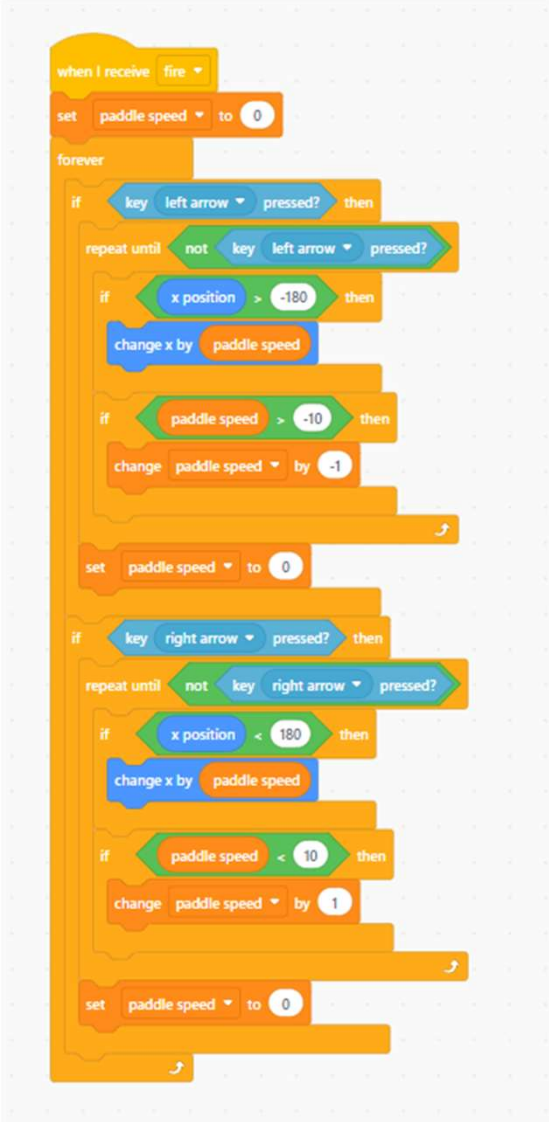
Paddle control

The paddle can move left and right only, and stops when it reaches the frame.

We can make the paddle movement more interesting by making it move faster the longer the player presses the key, but we have to make sure it can't leave the playing area. This means we need a variable which we can call 'paddle speed'.

The paddle can move after another broadcast message is received called 'fire'. This is the broadcast message that will be sent when the ball is fired.

At the start of the game the paddle isn't moving so paddle speed is set to 0.



Scratch Brick

Paddle control continued

We use a forever loop and the left and right arrow keys to control the paddle. Each time we pass through the loop we will change the x position of the paddle by the amount of the paddle speed variable.

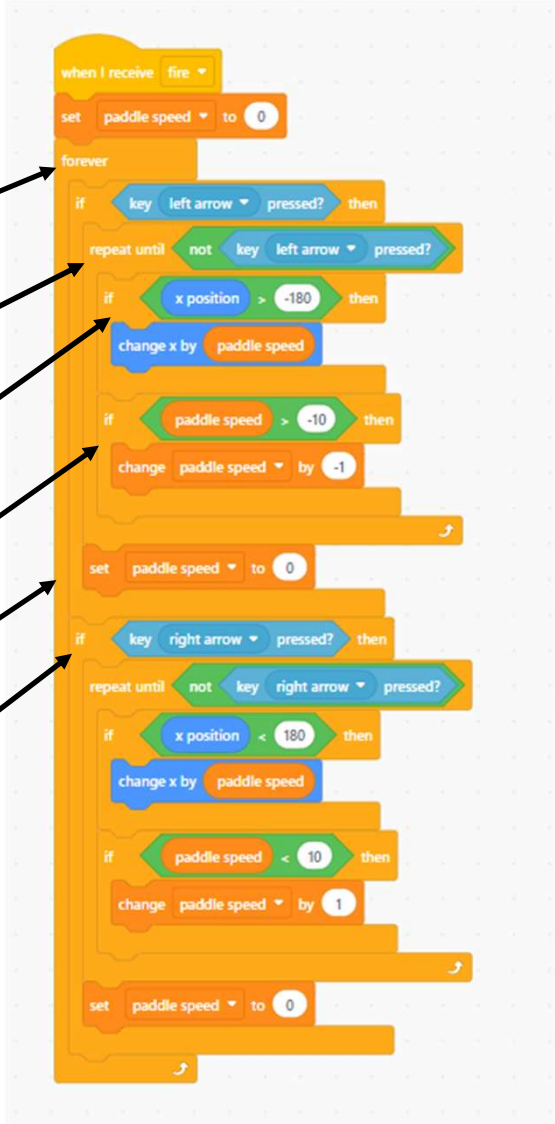
If the player keeps the arrow key pressed, we increase the paddle speed variable up to a maximum of 10 until the key is released. We use a repeat until block to increase the paddle speed variable in steps until the key is not pressed.

We need to check that the paddle can't go past the frame. We do this by making sure the paddle's x position can't be less than -180 when moving left.

Be careful...if we are moving left, paddle speed is a negative number!

If the player isn't pressing an arrow key, we set paddle speed to 0 so that the paddle stops

We need the same code for the right movement, but this time we make sure that the paddle's x position is always less than 180 and the paddle speed variable is a positive number.



Scratch Brick

Pointer

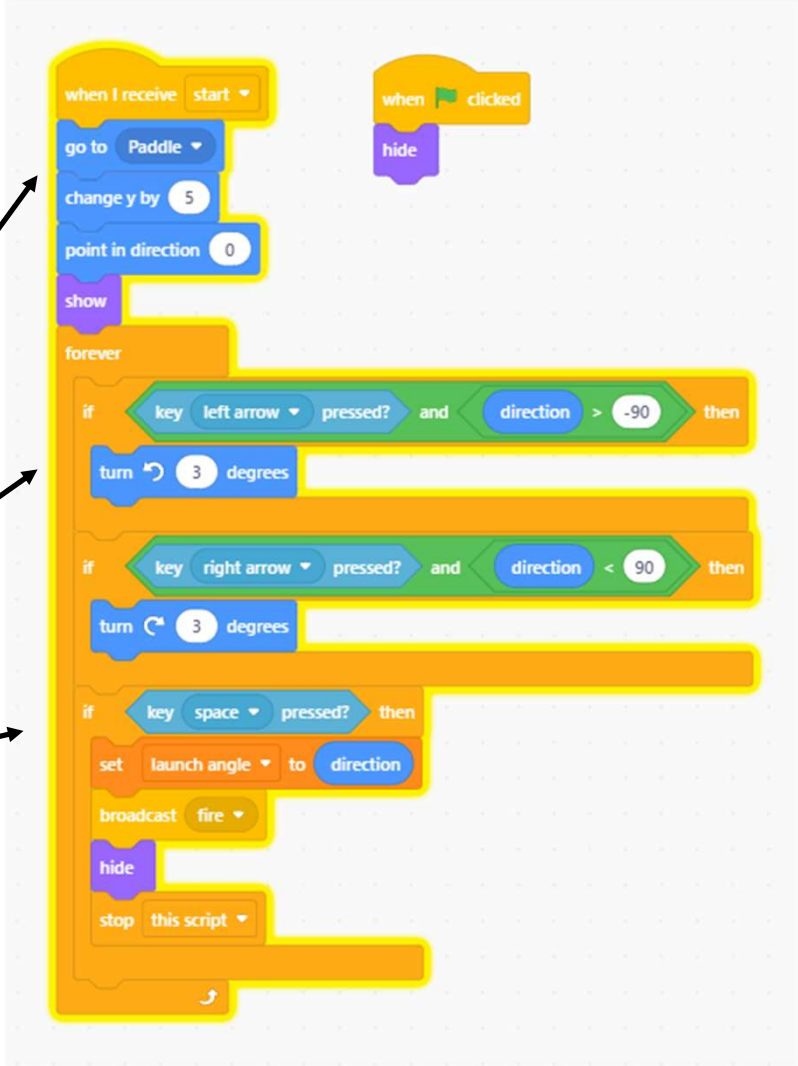
The pointer is used to fire the ball in the right direction at the start of the game. We hide the pointer sprite to begin with.

We create a variable called launch direction which we will use to tell the ball which direction it needs to move in.

When the start broadcast message is received, we place the pointer on the paddle and point straight up (direction = 0).

The player can then point in the direction they want using the arrow keys.

When the player is ready, they press the space key. We set the variable launch angle to the direction of the pointer when the space key is pressed. This action sends the broadcast message fire which tells the ball it can start to move. Once the ball has been fired we hide the pointer and stop the pointer script.



Scratch Brick

Ball

Now we need to code the ball. This is quite complicated but we will build it up in steps.

The main challenge is to make the ball bounce correctly when it hits the frame or a brick. To begin with, we'll just have the frame, then we can add the bricks later.

Scratch Brick

Ball

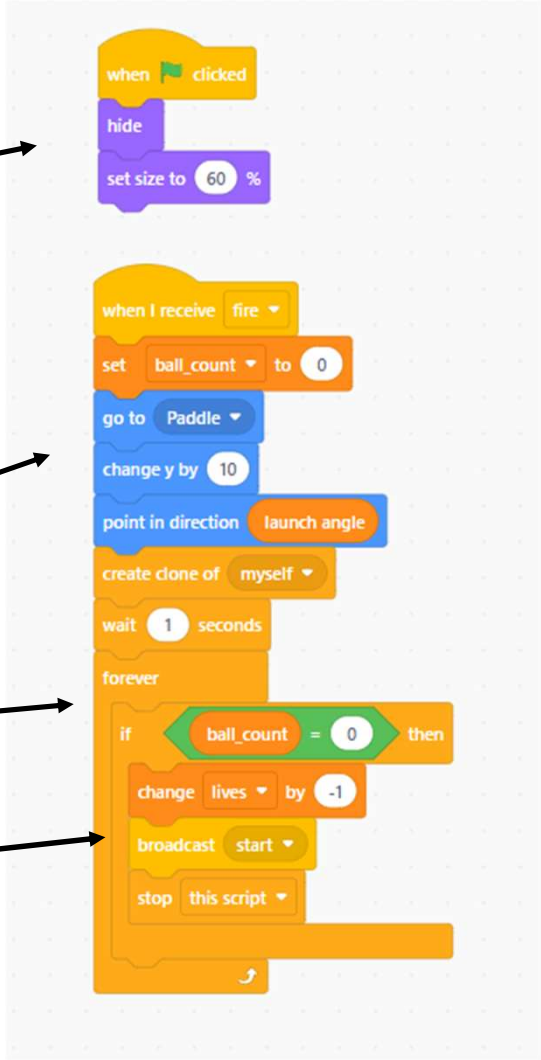
To start with, the ball is hidden.

We create a variable called ball_count which will keep track of how many balls are on the screen. This will be useful when we start to code bonuses later. The balls in the game will be clones of the ball sprite. This is set to 0 to begin with, and will increase each time a new clone ball is generated as we'll see on the next page.

When the broadcast message fire is received, we place the ball on the paddle and point it in the direction set by the pointer using the launch angle variable. We then create a clone of the ball sprite which is the ball that will actually move in the game.

In this section of code, we add a forever loop to check that there is always at least one ball in play using the ball_count variable. If there are no balls on the screen (ball_count = 0), the player loses a life. This is a good moment to create the lies variable we'll need to track lives so we can use it here.

If a life is lost, we broadcast the start message to continue playing. We'll test how many lives are left separately.



Scratch Brick

Ball continued

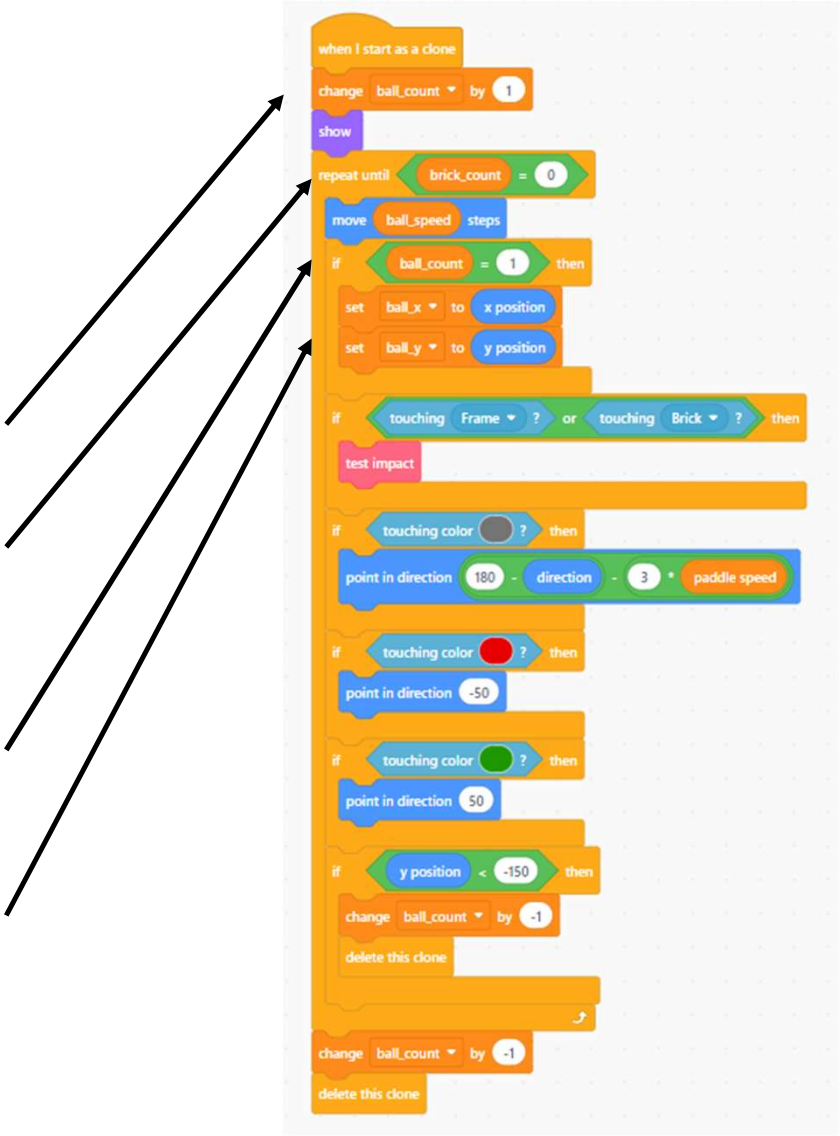
Now we need to code how the ball behaves when it moves and when it hits something.

When a new ball clone starts, we change the ball_count variable by 1 to keep track of how many balls are in play.

We need a variable called brick_count which we'll use to track how many bricks we've put on the screen. At the moment we don't have any bricks and we haven't set a value for the brick_count variable so we just need to use the variable name for now. We use a repeat until block which will continue until there are no brick clones left on the screen (brick_count = 0).

We create a new variable called ball_speed so we can easily adjust the difficulty of the game. We'll set the value for ball_speed later, for now we just need the variable name.

The ball clone moves ball_speed steps each time we pass through the repeat until loop until it hits something. We create two more variables ball_x and ball_y to keep track of the ball_position. This will be needed later when we might have more than one ball in play.



Scratch Brick

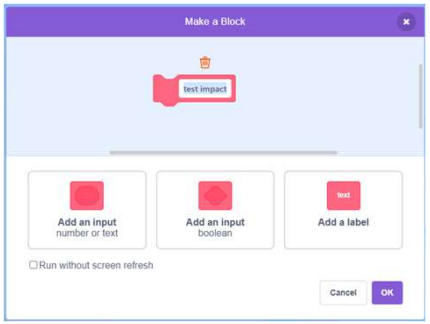
Ball continued

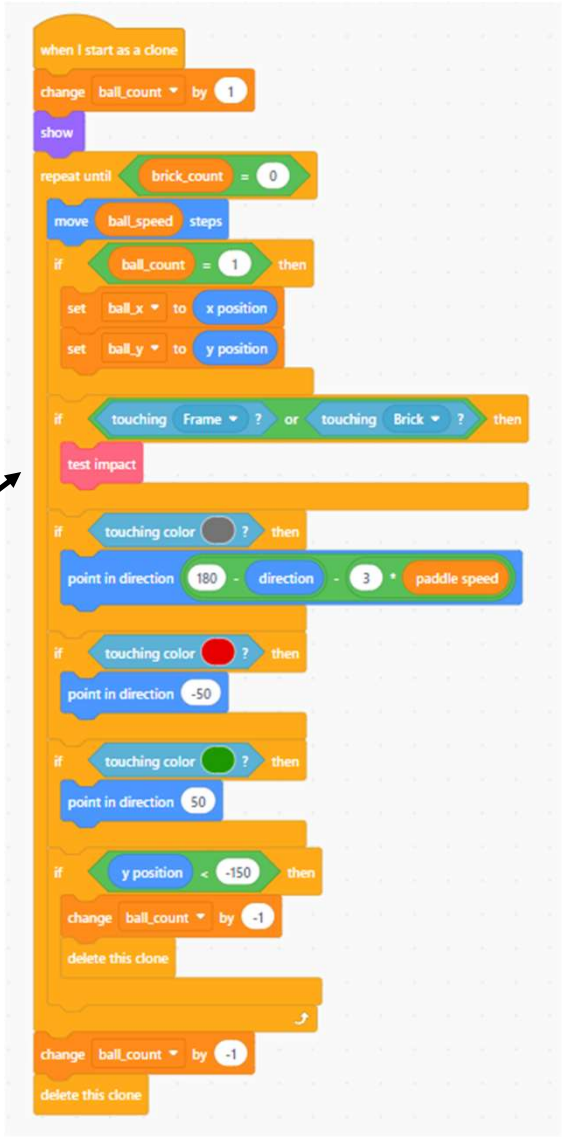
The ball needs to change direction when it hits the frame or a brick.

The problem is to decide how the ball should bounce. Are we hitting a side or a top surface? We'll look at this next. To begin with, we can create a new code block called test impact.

Whenever the ball touches the frame sprite or a brick sprite, we'll call the test impact block to decide what happens next.

We also need to tell the ball what to do when it hits the paddle. We'll look at this on the next page.





Catford Coding Club

Scratch Brick

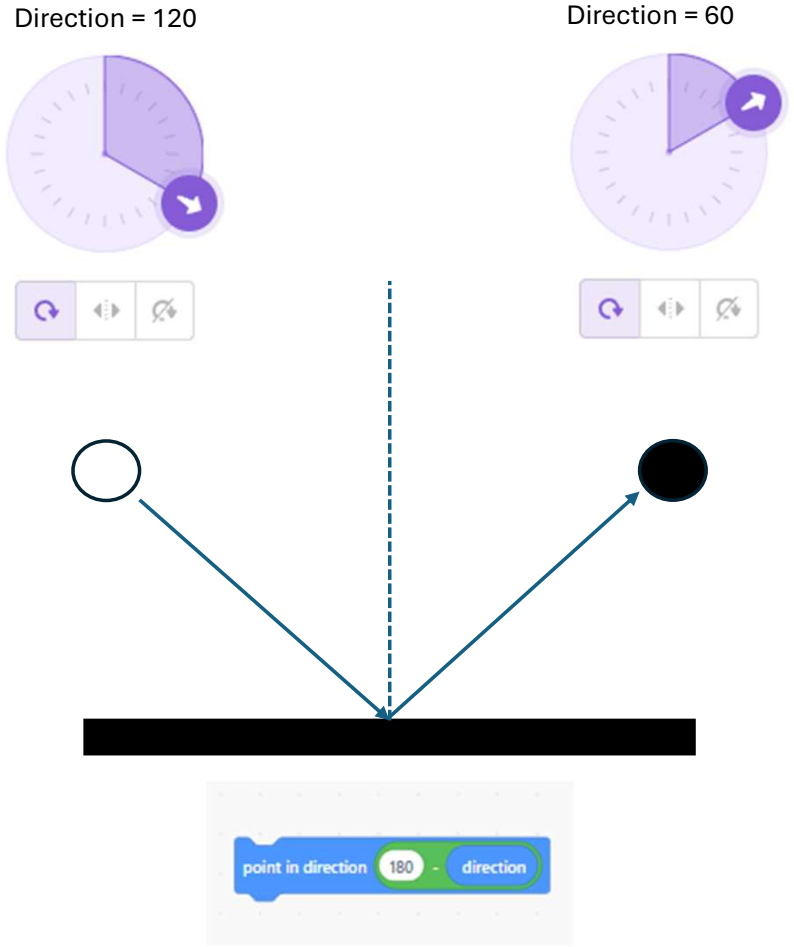
Ball continued

Let's think a little bit about how a ball bounces. You'll learn more about this in school in maths, but for now, we can tell the angle at which the ball hits an object from the direction of the ball sprite, and from that we can tell in which new direction the ball should bounce.

In Scratch, remember that we use degrees to measure direction. A direction of 0 is straight up, 180 is straight down, 90 is right and -90 is left.

The new direction of the ball when it hits the paddle is 180 degrees minus the old direction.

So for example, if the direction of the ball is 120 degrees and it hits the paddle, the new direction to bounce away is 60 degrees (180-120)



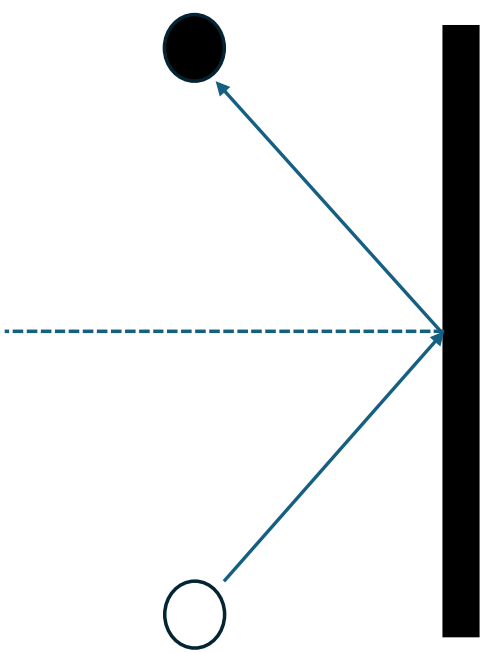
Scratch Brick

Ball continued

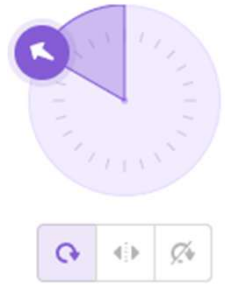
That's fine for when the ball hits a horizontal surface. What about a vertical surface?

This time, the new direction of the ball when it hits the paddle is 0 degrees minus the old direction.

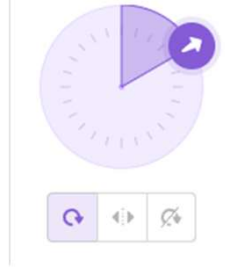
So for example, if the direction of the ball is 60 degrees and it hits the side of the frame, the new direction to bounce away is -60 degrees (0-60)

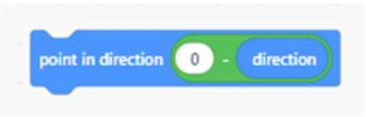


Direction = -60



Direction = 60





Scratch Brick

Ball continued

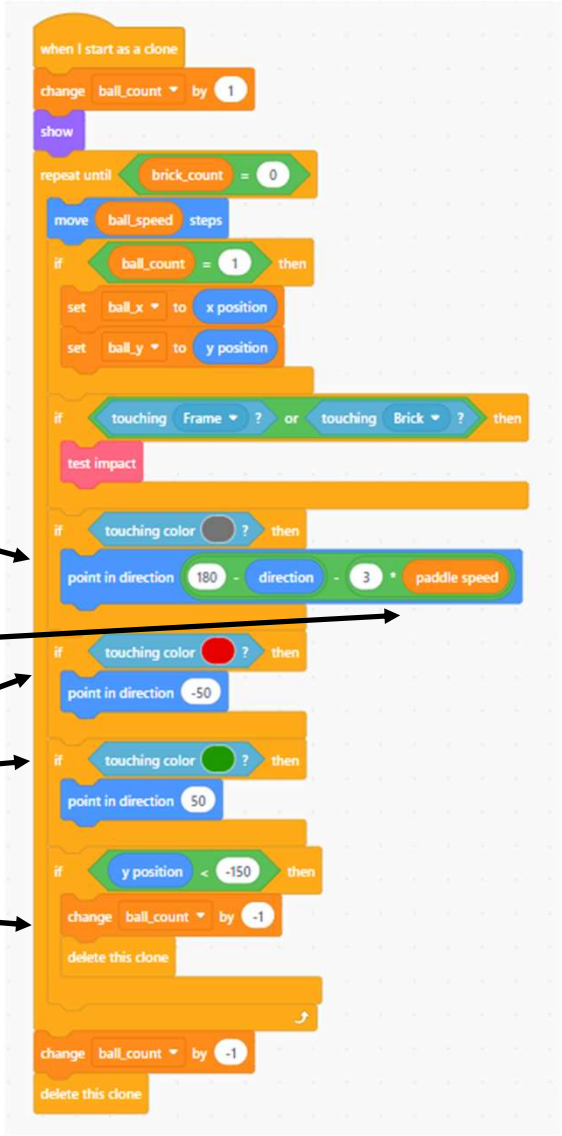
My paddle has three different colours so I can control how the ball bounces when it hits the paddle. That means I can use colour sensing in scratch to tell the ball how to react.

When the ball hits the main area of the paddle (grey) we can set the new direction to 180 minus the old direction.

To make it slightly more interesting, we can use operators (plus, minus, times, divide) and the paddle_speed variable to make an adjustment to the new direction based on the speed at which the paddle is moving, a bit like playing table tennis. You can try different values here to make it behave how you want.

If the ball hits the corner of the paddle it should bounce differently. By sensing for the two corners which are green and red, I can send the ball in the direction that looks realistic.

Finally, if the ball's position is less than -150 then the paddle has missed it, the ball has dropped out of the playing area and the player loses a life.



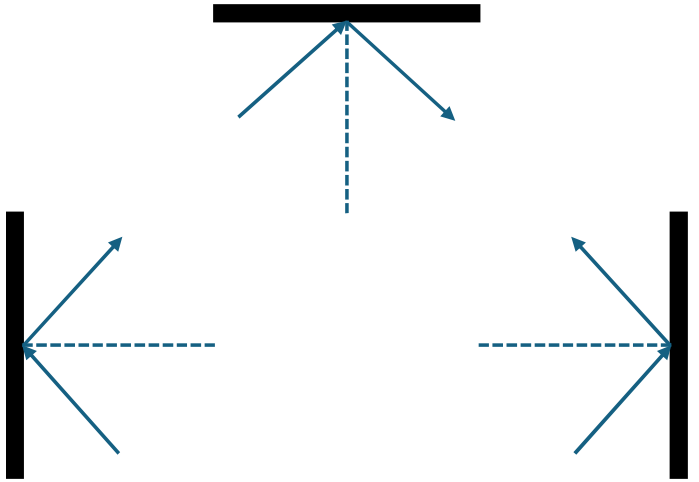
Scratch Brick

Ball continued

Now we need to code the test impact block we created.

When the ball hits an object, we need to decide how it should bounce. For the paddle, that was easy because we know where the paddle is. However, we don't know where the ball will hit a brick or the frame. We need a way to decide if we have hit a horizontal or a vertical surface.

Look at the examples opposite…when the ball hits the frame we need to know if it's hit the side side or the top.



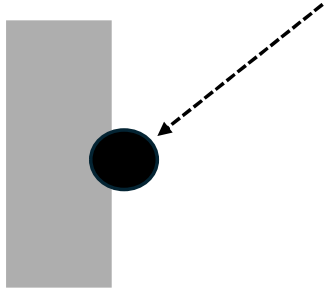
Scratch Brick

Ball continued

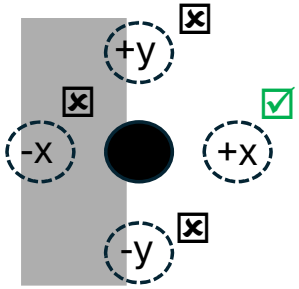
When the ball hits an object, we can test what happens if we move another step in each direction.

Look at the examples opposite…when the ball hits the frame we need to know if it's the side or the top.

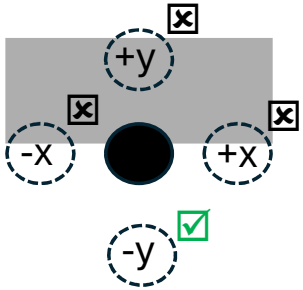
This is what we are going to build in our test impact code block.



Ball touches object. We know ball position (x,y) and direction…



Try changing x and y in turn to see if the ball would still be touching the object…looks like we hit a side wall



This time looks like we've hit the top wall

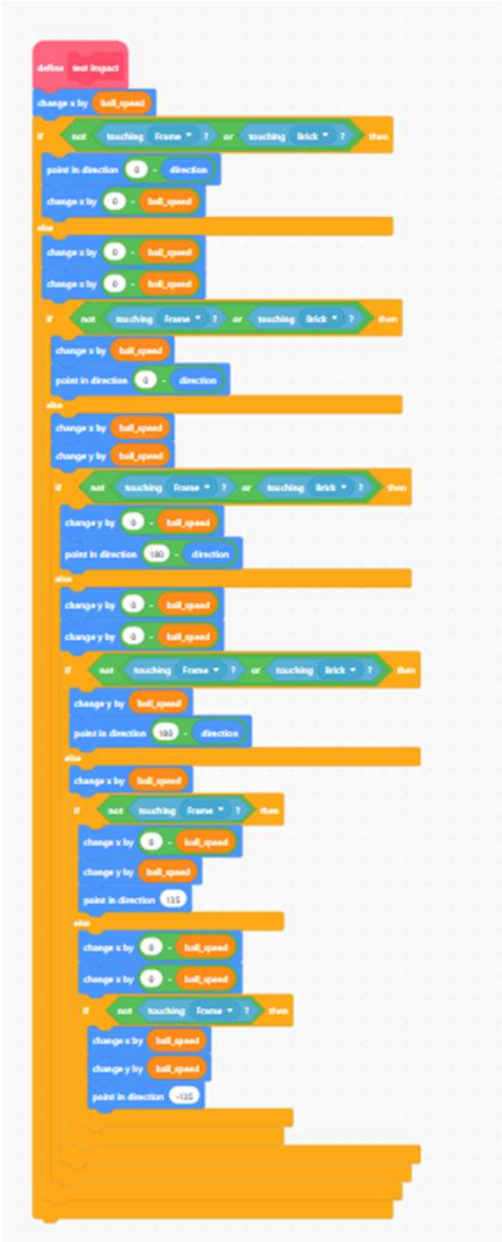
Scratch Brick

Ball continued...the test impact block

Our test impact block is going to test the hit by checking what happens when we change the ball position as we saw on the last page.

We are going to use the ball_speed variable to adjust x and y positions in turn.

The test impact code block uses if-else blocks for each of the adjusted position tests. We'll step through it on the next pages.



Catford Coding Club

Scratch Brick

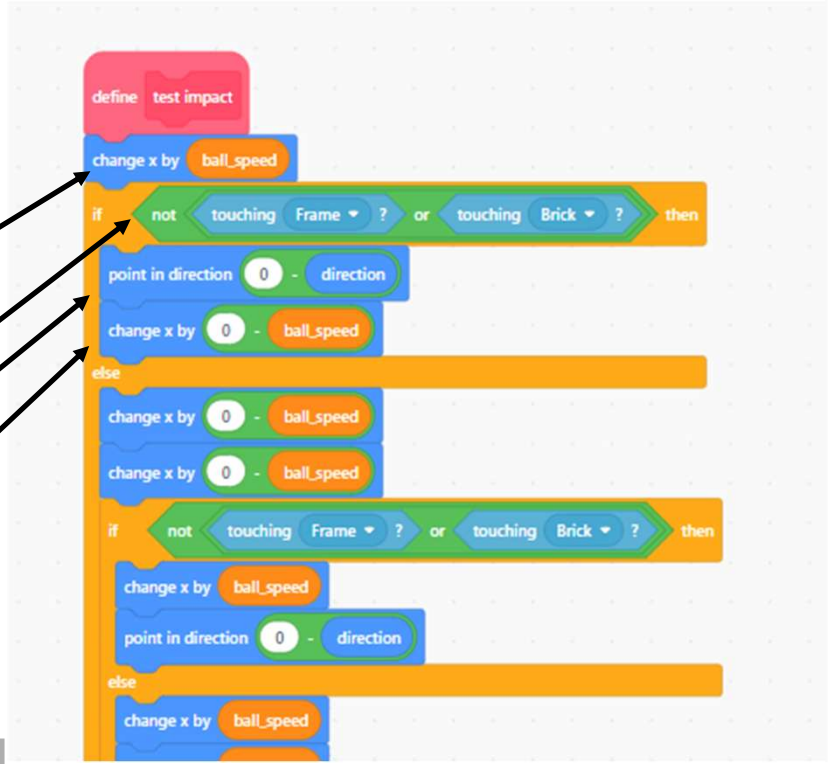
Ball continued…the test impact block

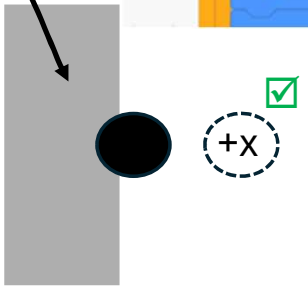
The first test is to increase x (move the ball to the right). We move the ball by the value of the ball_speed variable.

If the new position is not touching the frame or the brick, then we have hit a vertical surface to the ball's left like this:

Now we know to point the ball in the new direction which is (0 – direction).

We then move the ball back to its original position when it hit the frame or brick by changing x by –(ball_speed).





Scratch Brick

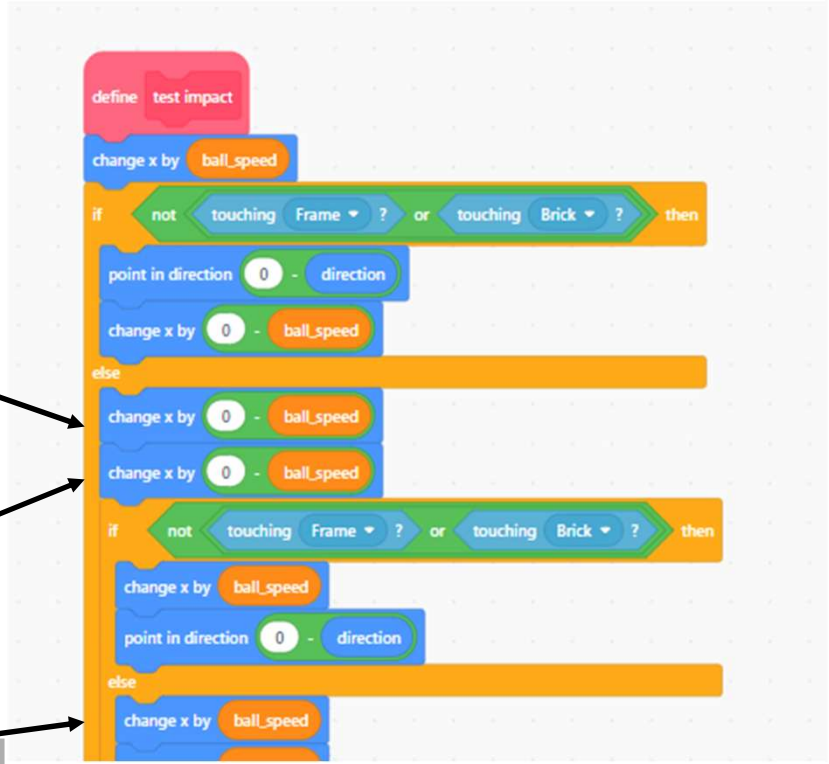
Ball continued...the test impact block

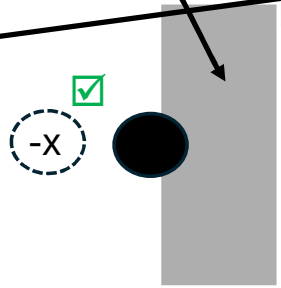
If the first test fails, next we try to reduce x (move the ball to the left). We move the ball by the value of the ball_speed variable twice, first to return it to its original position then to move to the new test position.

If the new position is not touching the frame or the brick, then we have hit a vertical surface to the ball's right like this:

Now we know to point the ball in the new direction which is (0 – direction).

We then move the ball back to its original position when it hit the frame or brick by changing x by ball_speed.





Scratch Brick

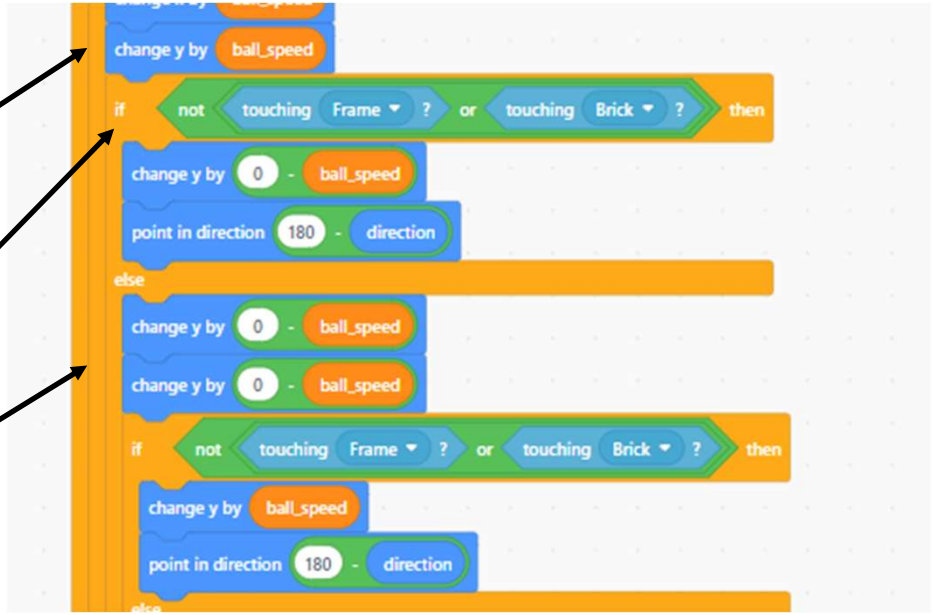
Ball continued...the test impact block

Now we repeat the process by adjusting the y position. First we increase y by ball_speed and test if the new position touches the frame or brick.

If it does, we have hit a horizontal surface from above and we need to change the direction to (180 – direction).

If that test fails we move to –y and test again. If we have hit a horizontal surface from below then we need to change the direction to (180 – direction).

Always we place the ball back at its original position after each test.



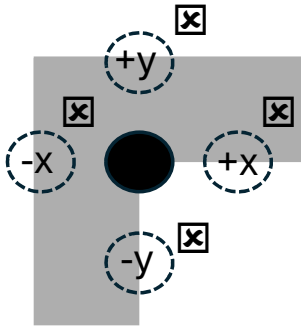
-y ☑

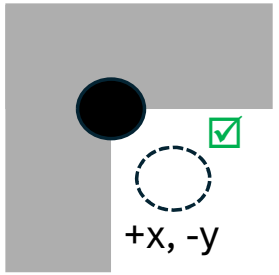
Scratch Brick

Ball continued...the test impact block

There is one final case...what if the ball hits right in the corner of the frame? Now none of the four tests work!

If all the other tests have failed, this might be what has happened. We can adjust both x and y to see of we can find the way out.





Scratch Brick

Ball continued...the test impact block

The complete test impact block looks like this:

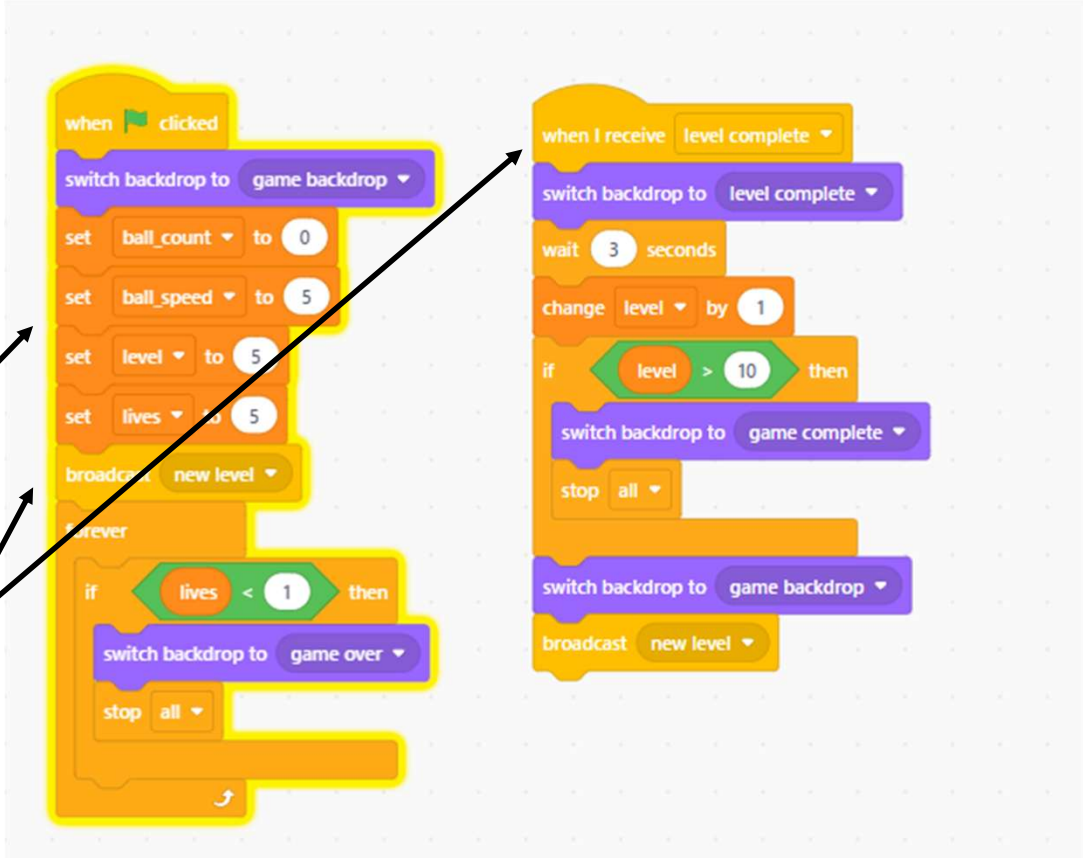
Scratch Brick

Game features

Some of the game feature variables are set in the code for the sprites that we have created. Some are set up in code for the backdrops.

We set the initial values for game variables in the backdrop.

I also use the backdrop code to send the broadcast messages to create a new game or a new level and then to switch backdrops when the player wins or loses.



Scratch Brick

Setting the bricks

At this stage we can play the game by bouncing the ball with the paddle. Now we need to add bricks.

Each level has a different pattern of bricks. I've used lists to create the layout for each level. You can add more.

We'll look at how the lists work, then build the code to place the bricks in the playing area.

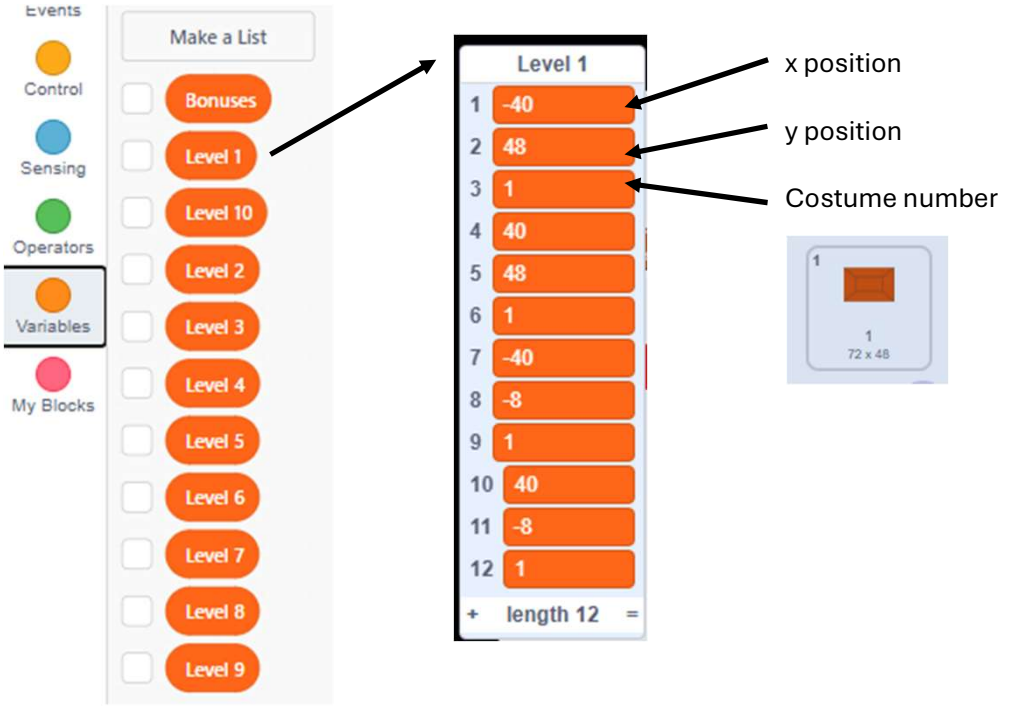
Scratch Brick

Level lists

I created some lists in the starter project so you have some game levels to play. There are 10 levels.

Each list describes where the bricks are placed and which costume number. Each brick is defined by three numbers: x position, y position and costume number.

Let's look at the Level 1 list. It has only four bricks, so the list has 12 numbers in total.



Scratch Brick

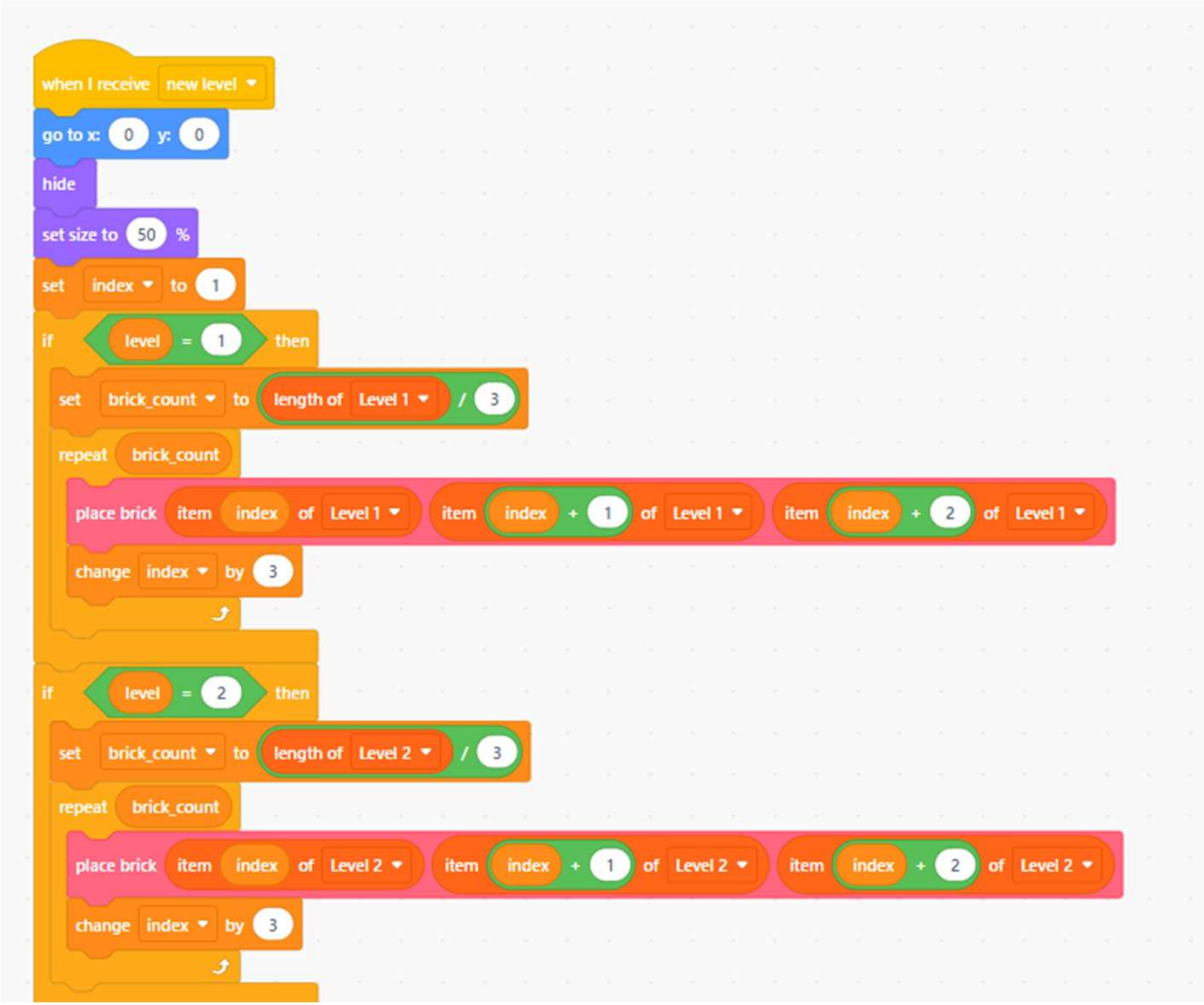
Placing bricks

We'll use the Level lists to place bricks on the playing area when a new level starts.

The bricks that will appear on the screen will be clones of the brick sprite. The main brick sprite will be hidden.

For each level we will read the list or that level and use a code block called place brick to put the bricks in the right place.

We need a variable called index which we will use to move down the lists. To begin with index variable is set to 1 as the first item in a scratch list is item 1.



Scratch Brick

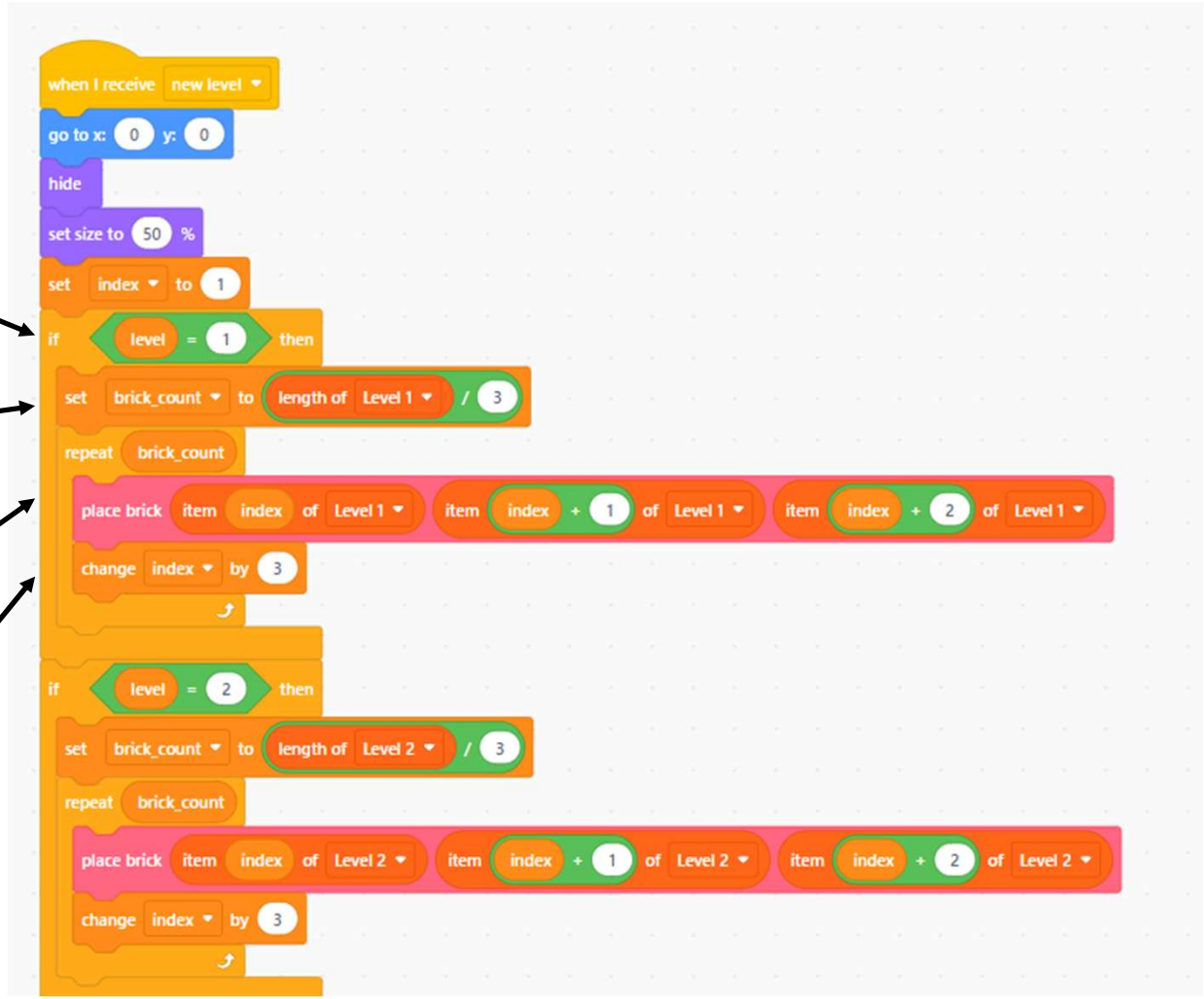
Placing bricks continued

First we check which level we need using the level variable. There will be an if block like this for each level and they all work in exactly the same way.

We use the brick_count variable to keep track of how many bricks there are in the level. The brick_count variable is set to the length of the list divided by three (remember there are three numbers for each brick).

Then, using a repeat block for the total number of bricks, for each brick we call the place brick code block which we describe on the next page.

Each time we place a brick, we move to the next one by increasing the value of the index variable by 3 to the first number for the next brick.

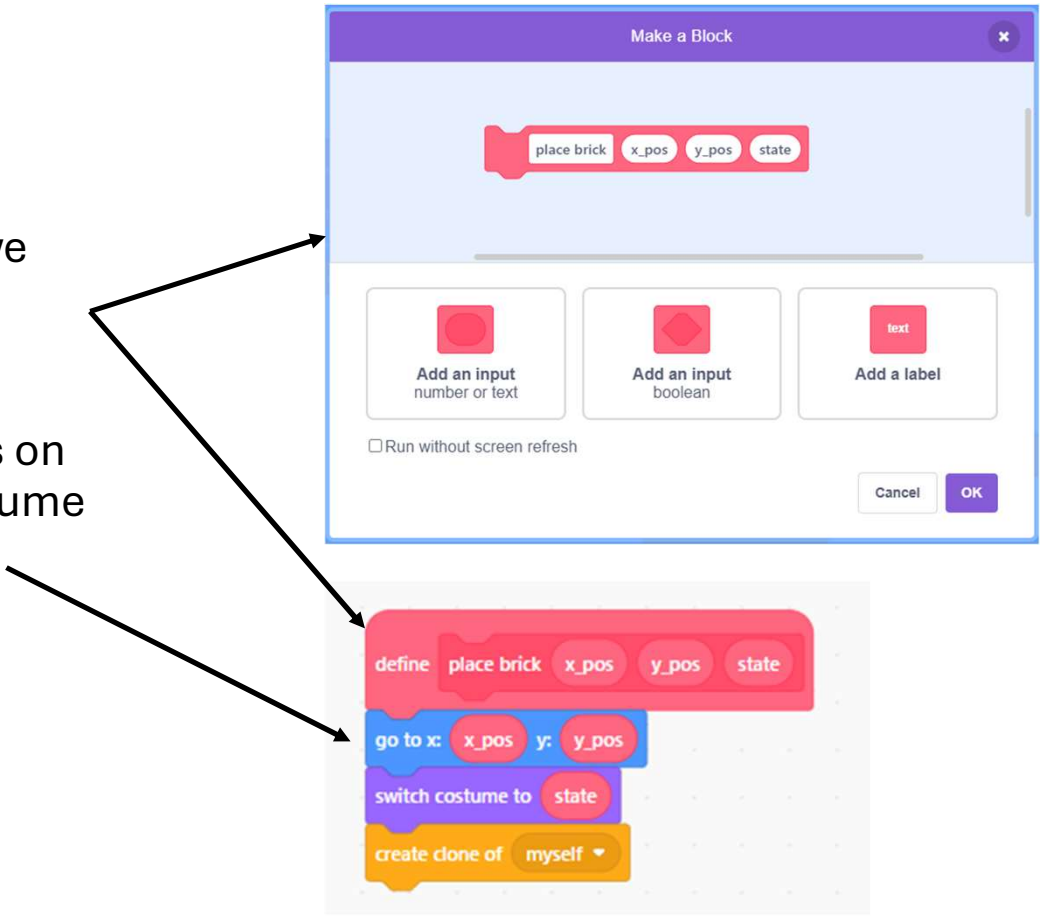


Scratch Brick

Placing bricks continued

When we call the place brick code block, we pass three variables to it: x_pos, y_pos and state.

Then, we go the correct x and y coordinates on the screen, switch to the correct brick costume and create a clone of the brick sprite.

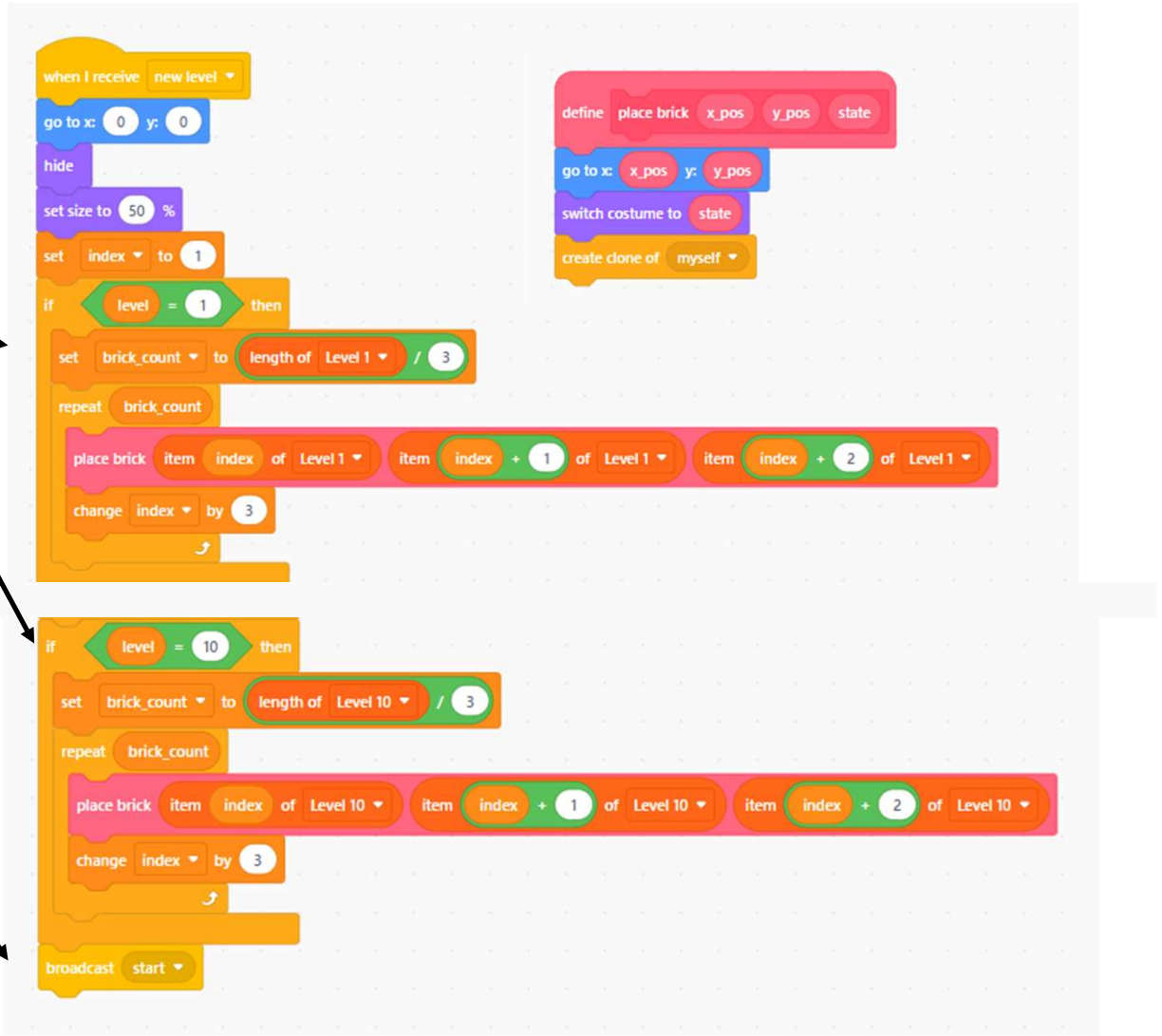


Scratch Brick

Placing bricks continued

So now we can code an if block for every level. The level variable and the if blocks control which set of bricks is placed on the screen.

Once the bricks have been placed we are ready to begin, so we broadcast the start message.



Scratch Brick

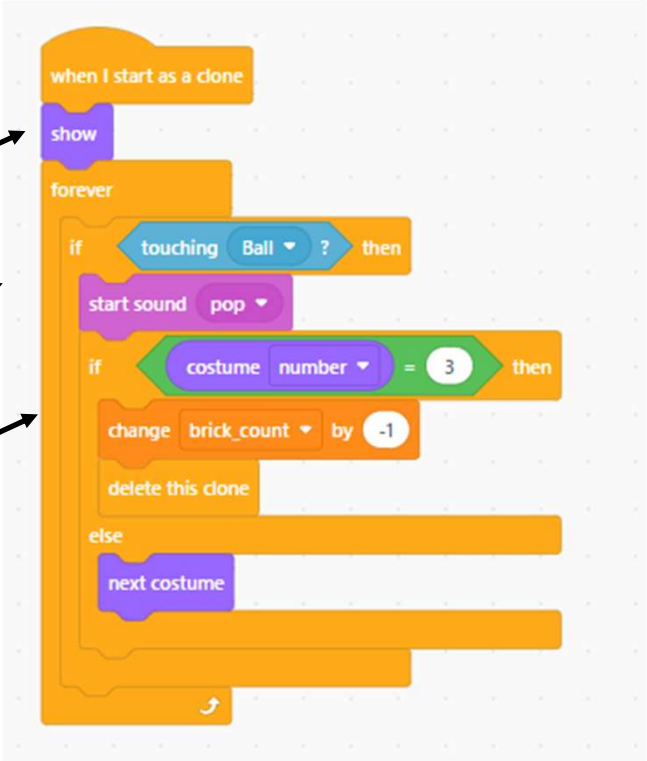
Brick actions

Now we need to code the bricks so they change when they are hit by the ball.

When a brick clone is placed, we show it and then test in a forever loop if the ball sprite touches it.

If the ball touches the brick clone, we play a sound and we change the brick clone's costume to the next costume so that it looks like the brick is cracking.

If the brick clone has costume 3, then it disappears when hit by the ball. We reduce the brick_count variable by 1 so we keep track of the number of bricks left, and we delete the brick clone.



Scratch Brick

Catford Coding Club

Game Features

We can now add some more features to the game to make it more interesting. I've added four bonuses:

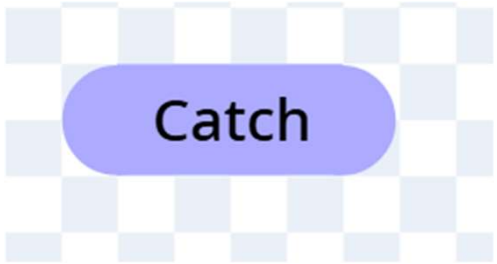
- Multi: one ball turns into many balls
- Catch: paddle catches the ball and lets the player aim the next shot
- Life: extra life
- Long: paddle is loner to make the game easier

Scratch Brick

Bonus sprite

First we need to add a bonus sprite. The bonus sprite will appear sometimes when a brick is hit by the ball. The bonus sprite falls to the bottom of the screen, and if the paddle catches it, the bonus feature is activated.

The bonus sprite has a costume for each bonus.



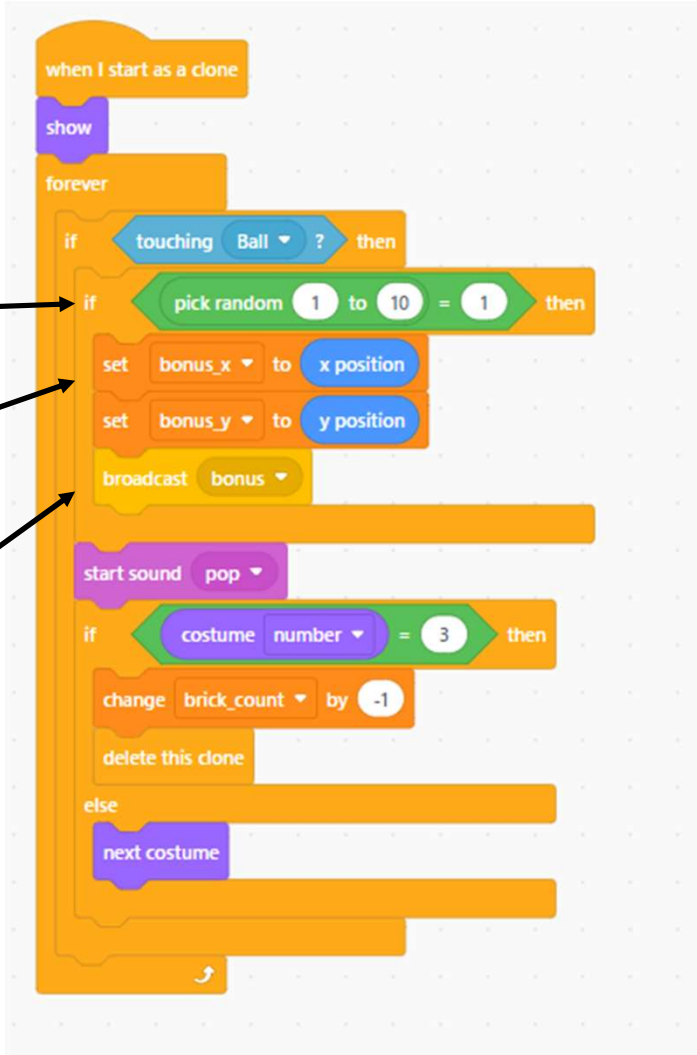
Scratch Brick

Bonus event

Now we need to decide how a bonus appears when a brick is hit. We will add a new if block to the brick sprite code for this.

When the ball hits a brick, we pick a random number (1 to 10 in this case). If the random number is 1, then we capture the position of the brick in two new variables, bonus_x and bonus_y. These will be used by the bonus sprite so it knows where to appear. This means that any hit on a brick has a 1 in 10 chance of creating a bonus. You can adjust this so that more or fewer bonuses appear in the game.

Now we broadcast a new message bonus so other sprites know that a bonus is in play.

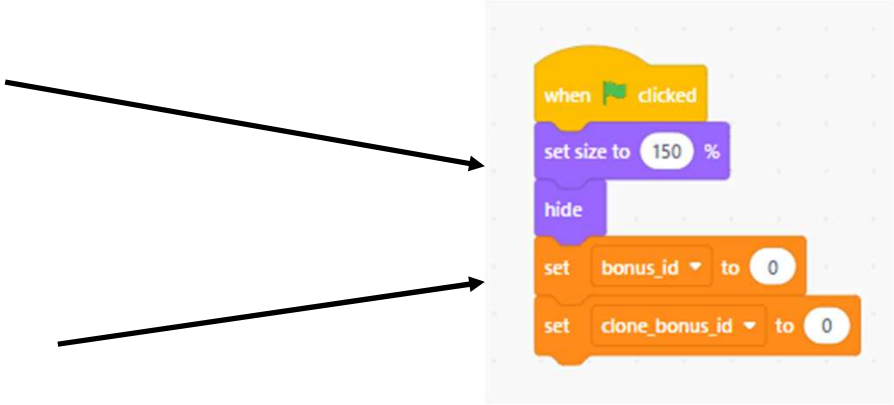


Scratch Brick

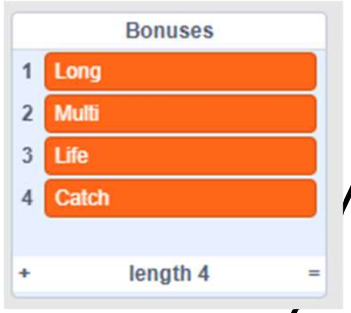
Bonus

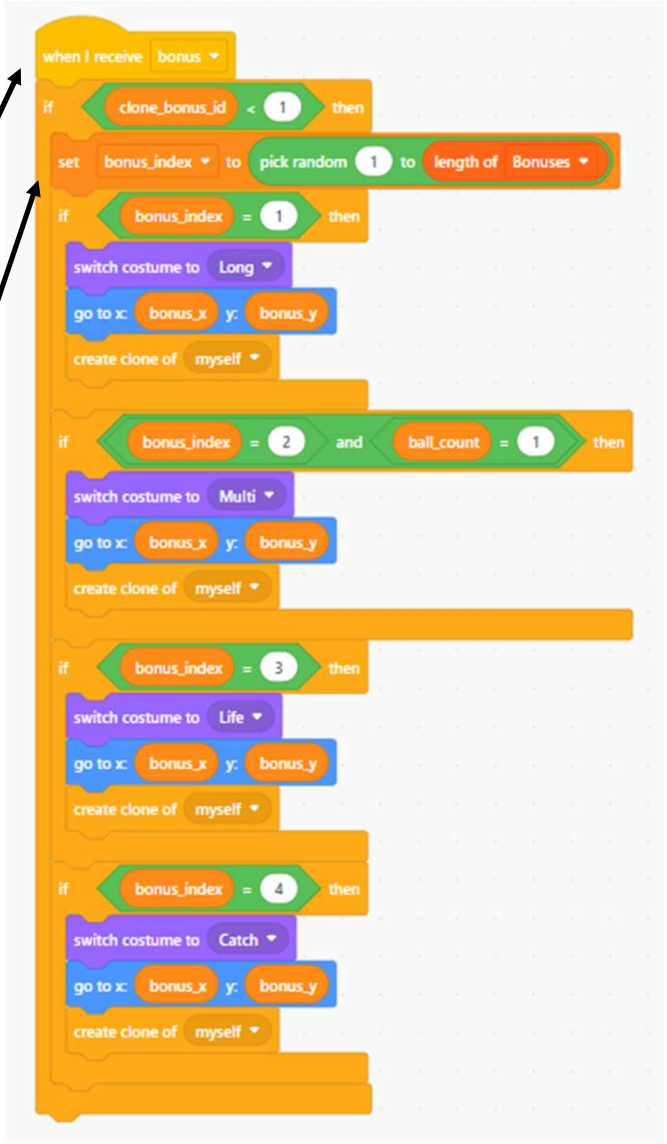
The bonus sprite is hidden. When a bonus event happens we will create a clone of the bonus sprite with the correct costume.

We need two more variables: bonus_id and clone_bonus_id. Both are set to 0 to begin with. These will be used so that we can decide which bonus will appear.



Scratch Brick





Bonus sprite code

When the bonus sprite receives the bonus broadcast message, it needs to decide what kind of bonus to show.

We have another list called Bonuses which contains the names of the different bonuses. We set the bonus_index to a random number between 1 and the length of the Bonuses list. In this case we chose one of the four available bonus.

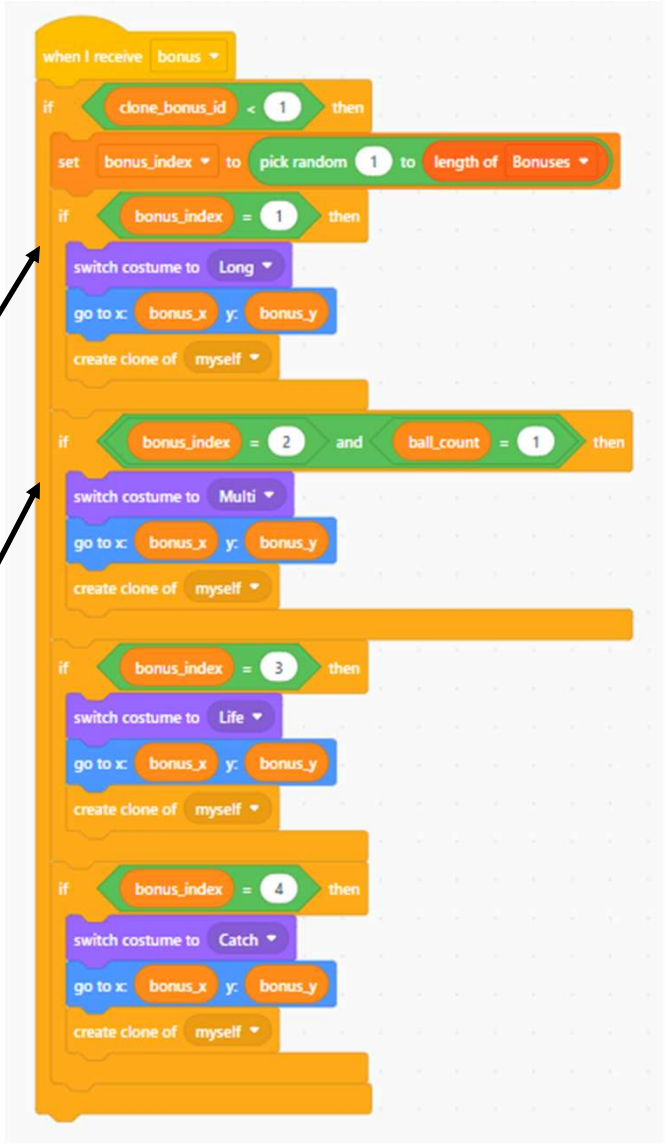
Scratch Brick

Bonus sprite code

Now, for each bonus type we can code the instructions.

For bonus index =1, we need the long bonus. We switch the bonus sprite costume to the long costume and we place a new clone of the bonus sprite where the brick was hit using the bonus_x and bonus_y variables.

The other bonus events work in the same way. The one different one is the multi bonus where we only allow a multi bonus if there is only one ball in play. We do this by checking the ball_count variable.



Catford Coding Club

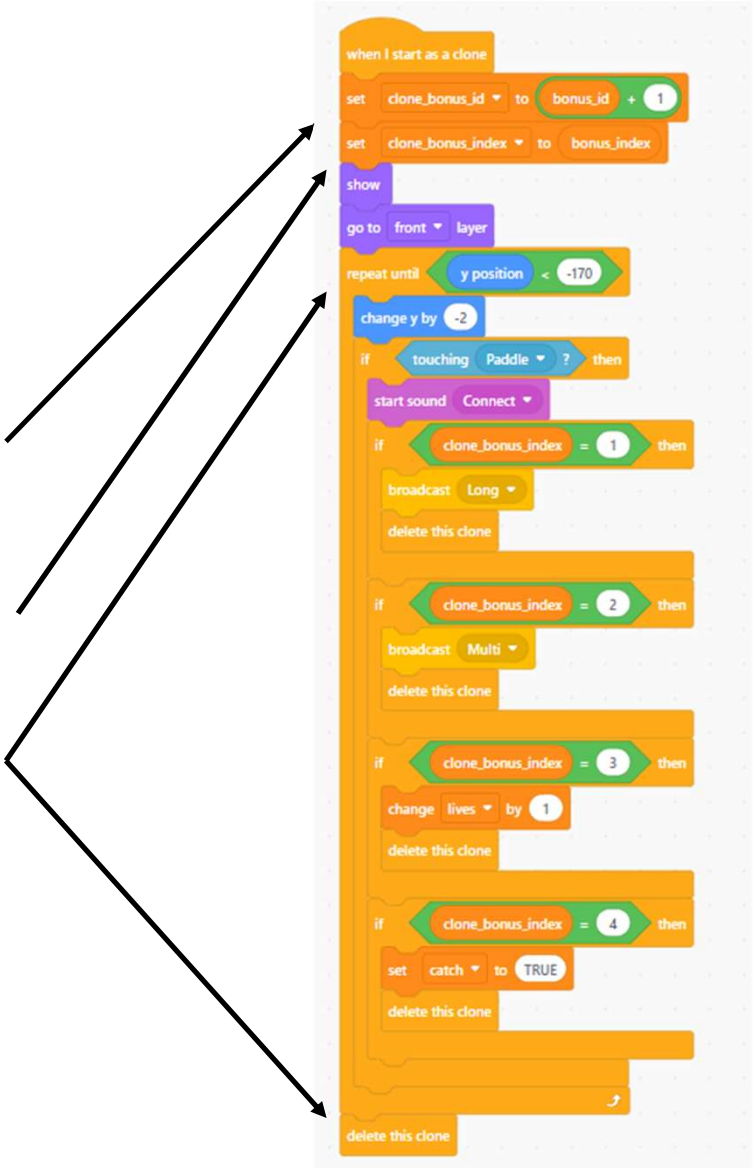
Scratch Brick

Bonus sprite code

Next, we need the code for each bonus. When a new close starts, we set the clone_bonus_id to the value of the clone_id variable + 1. This allows us to manage more than one bonus event at a time.

We also set the clone_bonus_index to the bonus_index variable so each clone knows which bonus it carries.

We show the new bonus sprite clone and bring it to the front layer, then we use a repat until block to move the clone bonus sprite down the screen. If the paddle doesn't catch it and the clone bonus id reaches the bottom of the screen at y = -170, then we delete the clone so that it disappears and the bonus is lost.



Scratch Brick

Bonus sprite code

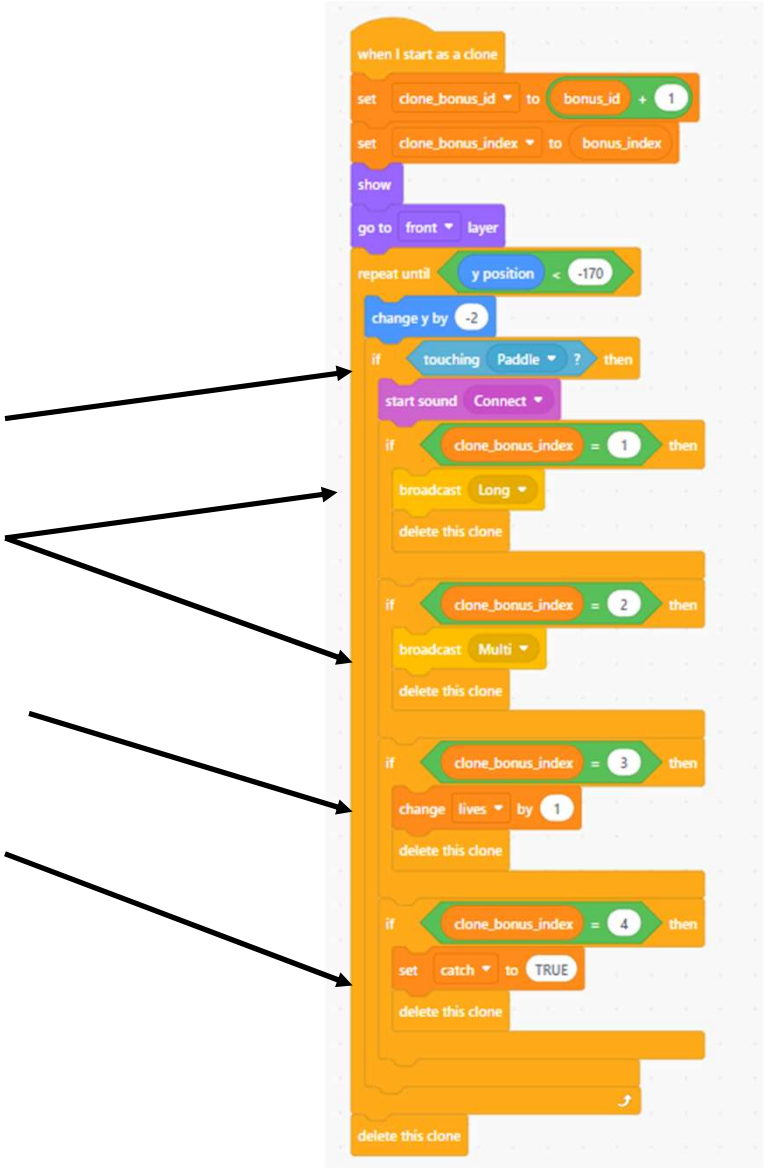
If the paddle catches the clone bonus sprite we play a sound and then check what type of bonus we have using the clone_bonus_index variable.

For the long (clone_bonus_id = 1) and the multi (clone_bonus_id = 1) bonuses, we broadcast new messages so the paddle and ball sprites can receive them.

For the lives bonus, we an just change the lives variable by 1.

For the catch bonus, we need a new variable called catch which we set to TRUE.

Finally, we delete the bonus sprite clone so that it disappears.



Catford Coding Club

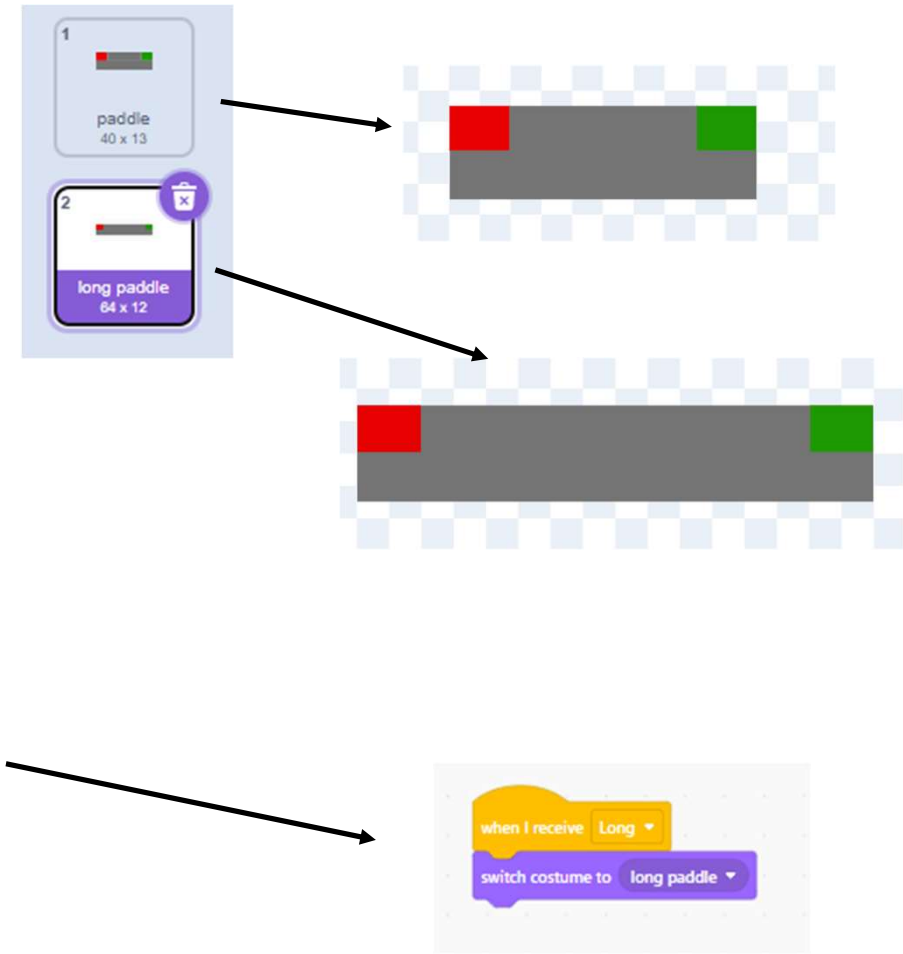
Scratch Brick

Long bonus

When the long bonus happens, the bonus sprite broadcasts the message long. This message is received by the paddle sprite.

We add a new costume to the paddle sprite which is just a longer version of the main paddle sprite.

We then need to add code to the paddle sprite so that it changes to a new long costume when it receives the long broadcast message



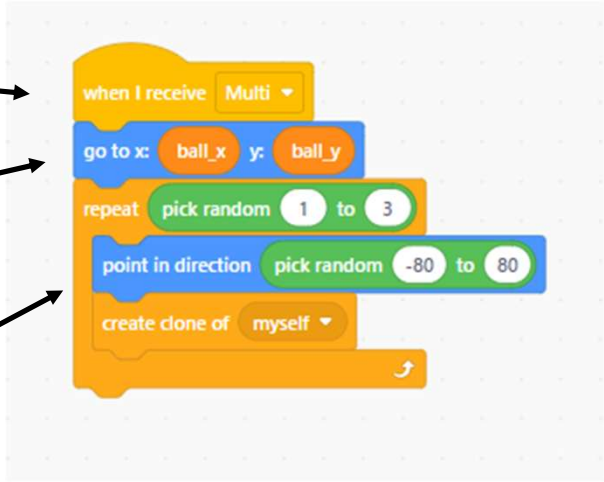
Scratch Brick

Multi bonus

When the multi bonus happens, the bonus sprite broadcasts the message multi. This message is received by the ball sprite.

When the ball sprite receives the multi broadcast message, it goes to the position of the ball sprite using the ball_x and ball_y variables.

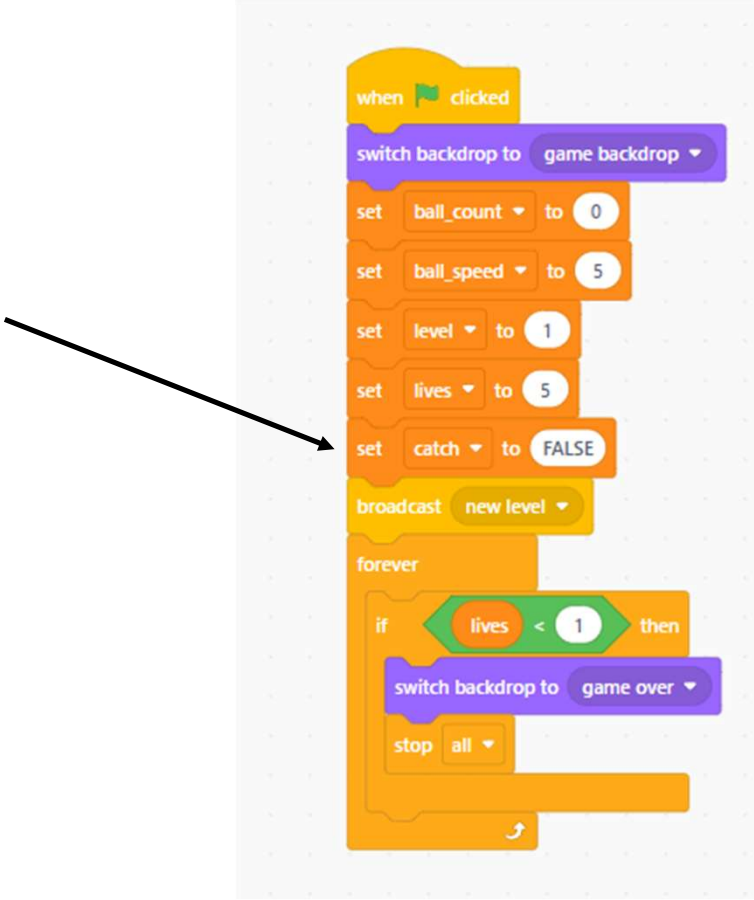
It then generates a random number of ball clones using a repeat loop. The new ball clones fly off in random directions between -80 and 80. You'll see lots of balls on the screen but it's quite difficult to keep them all in play!



Scratch Brick

Catch bonus

When the catch bonus happens, we set a variable called Catch to TRUE. This variable needs to be set to 0 in the backdrop code at the start of the game.



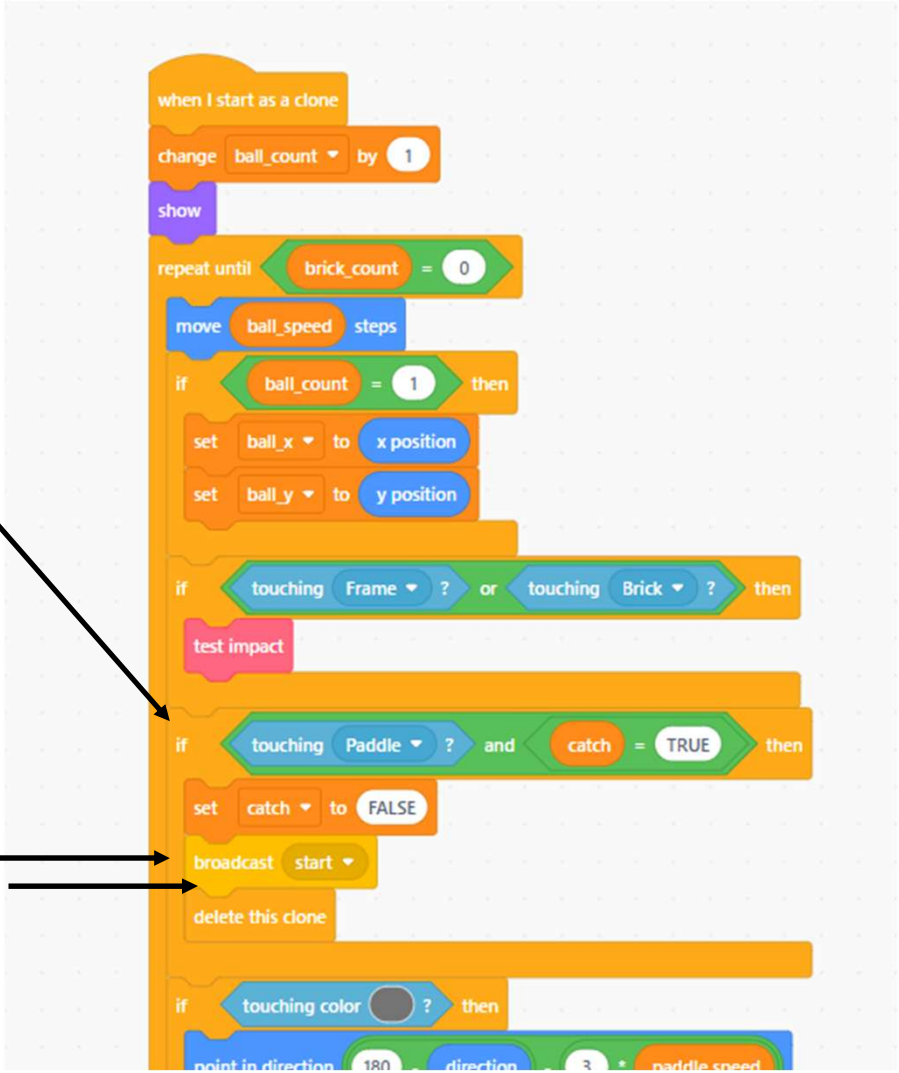
Scratch Brick

Catch bonus continued

Now we need to make a change to the ball code. We add an extra if block that checks if the ball hits the paddle and the catch variable is TRUE. If the catch variable is FALSE, the ball just bounces on the paddle as usual.

If the catch variable is TRUE, we broadcast the start message. The paddle then catches the ball and the player can use the pointer to aim the ball, just like at the start of the game.

We need to set the catch variable to FALSE now so that the paddle doesn't catch the ball every time, and finally we delete the old ball clone as a new one will be generated when the start broadcast message is received.



Scratch Brick

Bonuses

All the bonuses should now be working. Perhaps you can think of others?

A laser bonus where the paddle can shoot at bricks to destroy them? You could use code from the space invaders project for this.

A bonus that makes the ball go faster or slower using the ball_speed variable?

A bomb variable that lets the ball destroy not just one but several bricks in one hit using sensing?

Scratch Brick

Testing and feedback

Once you have finished, share your game with another Scratch coder and ask them to test it.

When you are testing someone else's game, look for ideas to improve the game:

- How do the sprites look?
- Do the defences generate properly?
- Is the game playable?
- What features and effects could you add?

When you have feedback, think about whether you agree with it and use it to make your game even better!

Scratch Brick

Well done!

Hopefully you now have a working game that you can share with friends.

There's always ways to make your game more interesting. You can add new features, add sounds and effects, a player guide...

There are also different and maybe better ways to code the game. These instructions give you a start but you can think of better ways to set up the game or code your sprites.

Have fun!