CODING

Game Making Workshop on Scratch

Learning Outcomes

In this project, students create a simple game using Scratch. They key learning outcomes are:

- Video games are made from pictures and step-by-step instructions (code)
- We use **loops** to repeat an action over and over again
- We use **if blocks** to make a game character react to something happening
- Coding is easy and fun!
- Coding is a form of creative expression

Required materials

- One computer per student, connected to the internet.
    - Check that the computers can run Scratch, by opening a web browser, going to http://scratch.mit.edu and clicking 'Create' in the top left corner. The editor should load correctly – if it does not, you may need to install Adobe Flash or a newer web browser.
- A mouse for each computer. (Trackpads are not ideal.)
- Misc computer stuff: extension leads, tables, chairs
- Wifi - internet account and password.
- Scratch Cards, one set per 10 students
    - To make: Download from https://scratch.mit.edu/help/cards/ and print single-sided on A4 paper. You will need **scissors** and **glue or a laminator** to make the cards.
- Coding Activity Step 1-3 Sheets, two sets per 10 students
    - To make: download the three .png files from https://goo.gl/kIRIRi
    - Print each on A3 cardboard, in landscape
- For emergencies: USB drives with a copy of the Adobe Air installer and Scratch Offline Editor installer, both downloaded from **https://scratch.mit.edu/scratch2download/**
    - More of these USB drives will allow you to more quickly recover from a network outage. One per 5 students is ideal.

Preparation

Turn on all computers, connect to the internet, open a web browser and go to http://scratch.mit.edu.
Log each computer into a different OMG Tech! Scratch account.
username: **omgtech1** through to **omgtech60**
password: **scienceisfun**

One person, create a Scratch Studio for the day

- Use a computer to log into the main OMG Tech! scratch account
    - username: **omgtech**
    - password: **omglearningisfun1**
- Click on 'omgtech' (top right) then 'My Stuff'
- Click 'New Studio' (near top right)
- Click on 'Untitled Studio' and rename it for the event following this format:
    - OMG Tech! at St Annes, Manurewa - 30 May 2015
- Click 'Allow anyone to add projects'

Workshop overview

Format

1. introduction talk (1 minute)
2. demo while students watch (8 minutes)
3. students do the activity
4. students save their work
5. closing circle (5 minutes)
6. afterwards: collect student games

Instructions

We start with a brief introduction, then we demo the whole activity while the students watch. The students then go to their computers and work on the activity with mentors helping where they can.

Scratch has a lot of quirks. The way you demo the activity and emphasis the various quirks has a huge impact on how often students get stuck. If you find a lot of students having the same problem, that's a clue that you need to emphasise that step more next time you give the demo.

Introduction and demo

Here is a video of Matthew giving the demo: https://youtu.be/mfd0Utm7ePI (It is slightly out of date, some steps of the activity has changed slightly.)

Here's a transcript: *(Actions are in italics, everything else is said out loud.)*

First, let's introduce ourselves. *(Mentor introduces themselves very briefly, with their name and one interesting fact about them.)*

Today we're going to make games using Scratch. **Has anyone used Scratch before?** (If so, you might find this easy, but it should still be a lot of fun.)

Scratch is a very flexible tool that can be used to make different kinds of games, animations and other things. Today, because we only have a short time, we're going to focus on just kind of game you can make in Scratch.

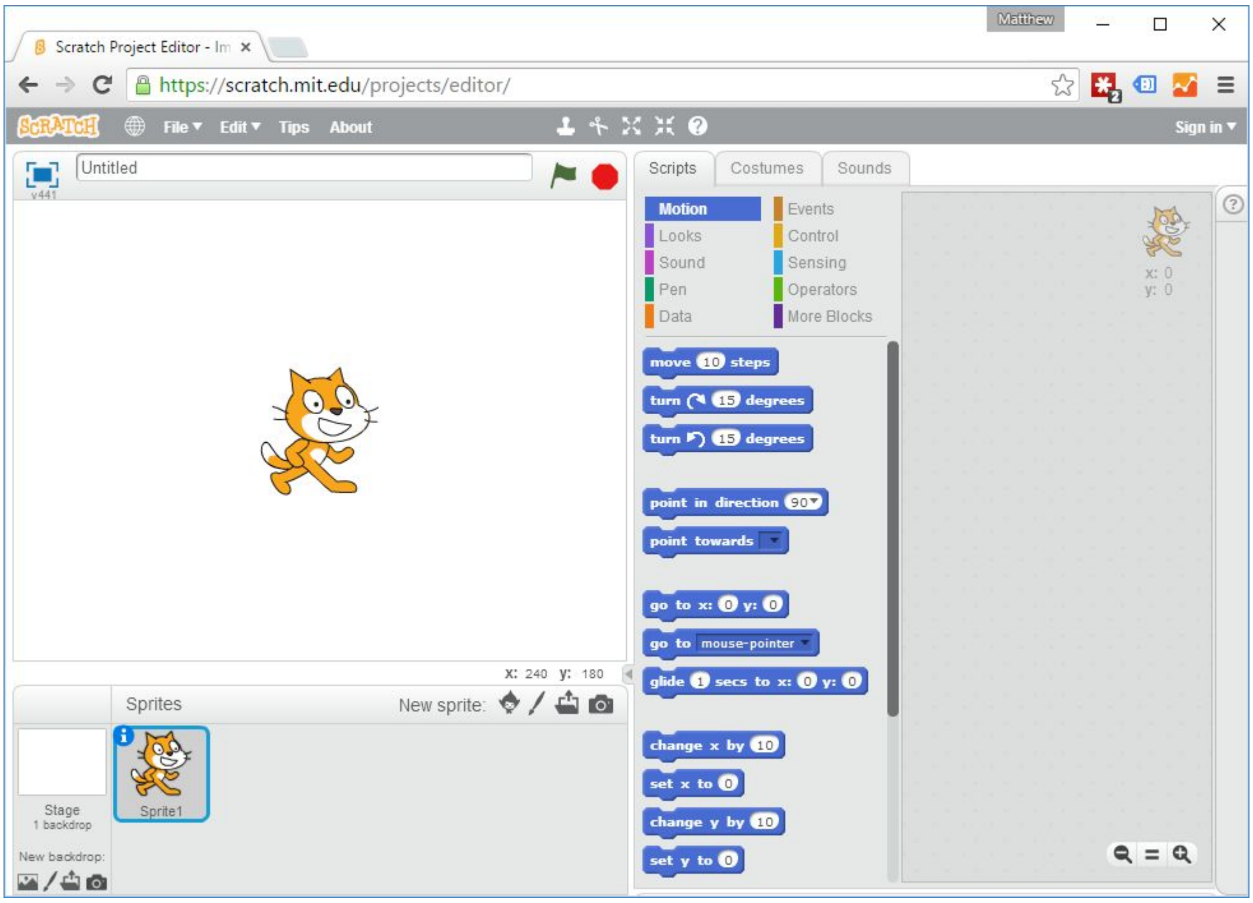
There are **three steps** to make this game. I'm going to show you the three steps, then we will all go off and make our own versions. First, if you can all sit somewhere where you can see this screen *(the mentor sits at a computer, ready to demo. If there isn't enough space, you might divide the class in two and have two mentors demo at the same time)*

This is the scratch website. *(show **http://scratch.mit.edu**)*. You click on "Create" up the top left to get started.
This brings you to the editor, which has lots of parts:

- On the left is the **stage**, which shows the actual game.
- Down below is a list of **sprites**, which are all the actors or objects in the game.
- This little white square to the left of the sprites is the **Backdrop**.

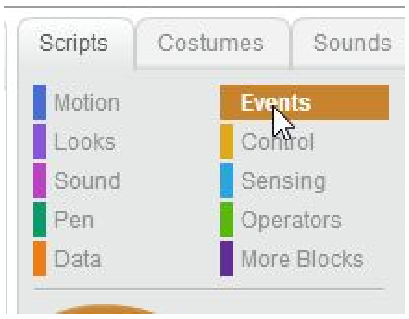
Across the top center are three tabs: Scripts, Costumes and Sounds. The most important is **Scripts**.

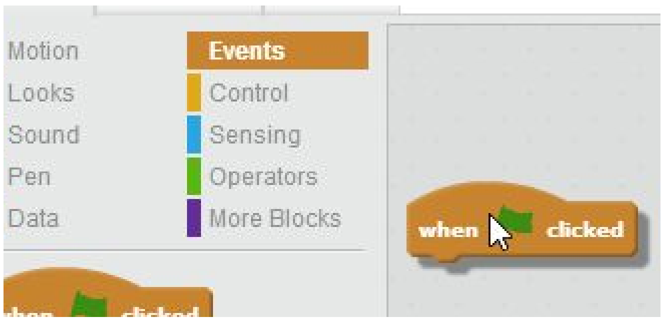
In the middle is the **tool box**, which has all the blocks you use to tell the sprites what to do. There are different sections, and they're colour-coded. *(Click on a few different sections in the toolbox).* And on the right hand side is an empty space where you put the blocks to make the script - the instructions for the character.



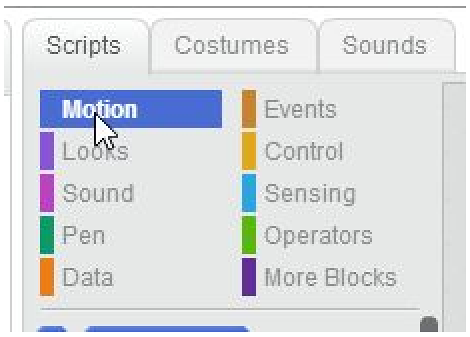
Now I'm going to go through the 3 steps that make our game.
Step 1: a character we can control
Step 2: walls we have to avoid
Step 3: a goal we try to get to

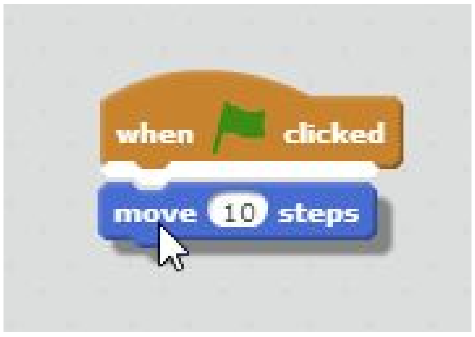
For **step one**, I'm going to the Events section and taking the very first block - "when the flag is clicked". This tells the cat when it should start reading its script.



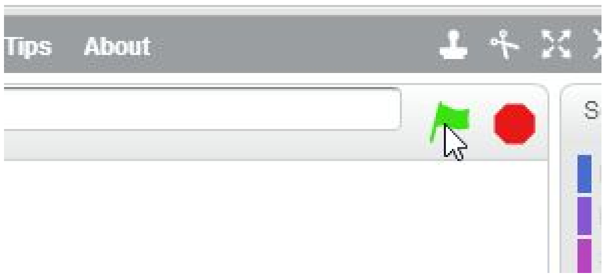


I'm then going to the Motion section, and taking the block **move 10 steps**, and attaching it.

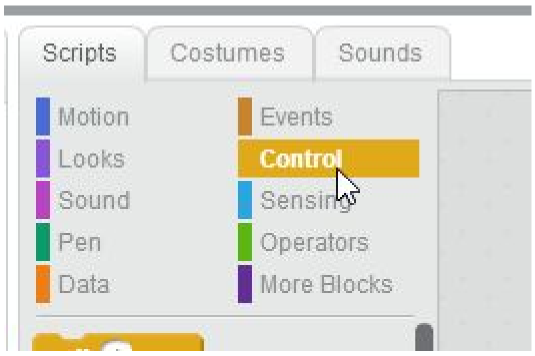




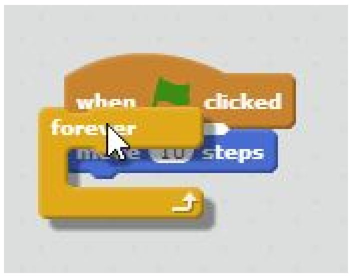
Now the script says "when the flag is clicked, move ten steps." So what will happen when I click the flag?
(Click on the green flag above the stage)
The cat moves a little bit.

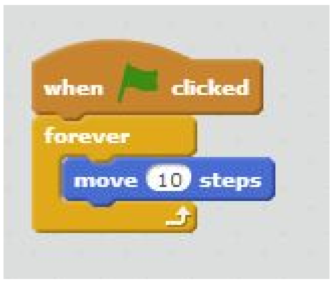


(Click the flag many times rapidly)
If I keep clicking, it keeps moving. This is good, but I don't want to sit here clicking forever. So I'm going to the **Control** section to grab the **'Forever'** block.



Forever is like a mouth and it can go over other blocks. Put the mouth of forever over the blue **move 10 steps** block so it fits around it.

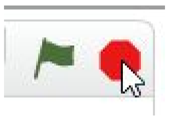


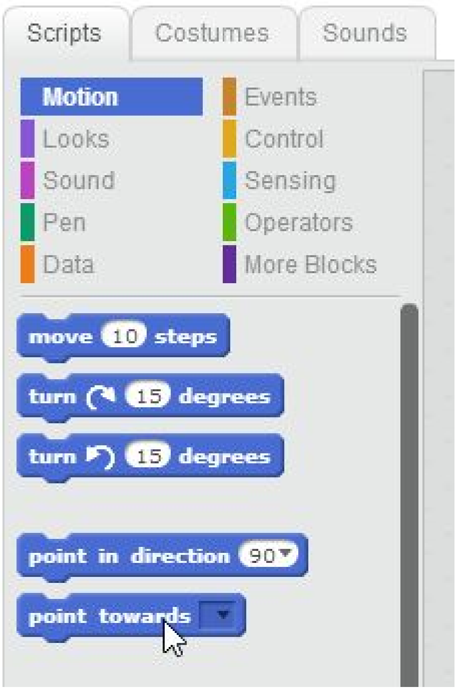


The way Forever works is: when the cat is reading its script, and it gets to the end of the forever block, it goes back up to the top and starts reading again. (See the little white arrow? That tells the cat to go back to the top and read everything again.) So now it's going to go "when the flag is clicked, move 10 steps, then move 10 steps, then move 10 steps" - over and over forever.

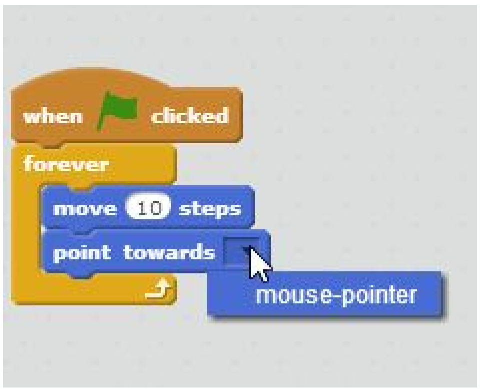
Let's try it. (click the flag again. The cat will run off screen. Drag it back to the middle and watch it keep running.)

It keeps running forever - or until I hit this red stop sign above the stage.



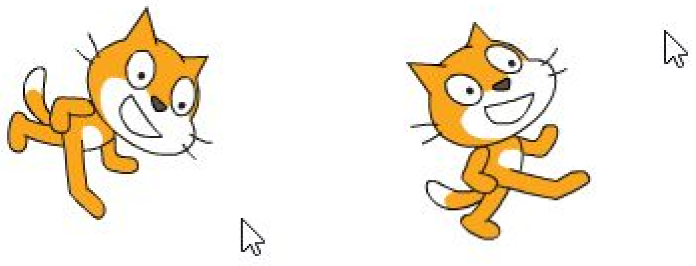


Cool, we have a character who moves, but it's not a game yet because you can't control it.
Let's go back to the Motion section (dark blue) and grab **Point Towards**. We'll put it inside the mouth of Forever so the cat will read this instruction over and over again.
If you click the tiny little black arrow at the end of the block, we can say **what** it should point towards - choose 'mouse-pointer''.



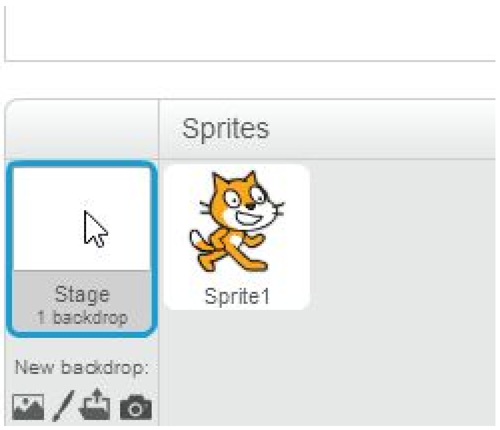
So now the script says: when I click the flag, point towards the mouse pointer, move ten steps, then point towards the mouse pointer again, move another ten steps, over and over. Let's try it out.

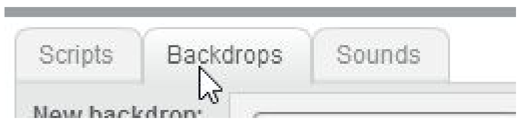
(click the flag and control the cat with your pointer so it moves around the screen.)



Cool, that was step 1 - making a character we can control. Now we have Step 2 and Step 3 to do.

Step 2: make some walls you're not allowed to touch.

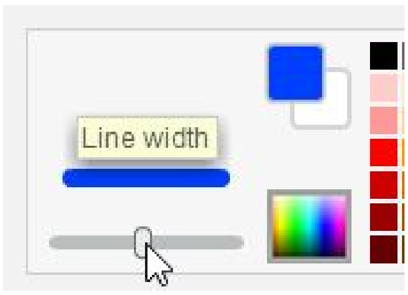


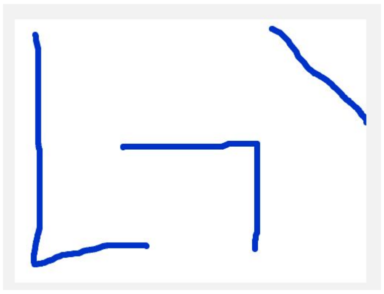


I'm going to click on the backdrop over here (the small white box on the far left). Now I'm going to change from script mode to drawing mode (the 'backdrops' tab near the top center). Now I can use these drawing tools to draw some walls that you can't touch.

Two things: **first**, the walls all have to be the same colour, because we'll tell the cat that it has to avoid that colour. Pick any colour you like, **except black -** other colours will work better.

Secondly, use this slider (at the bottom left of the drawing area) to make the brush nice and thick. That makes it work better.

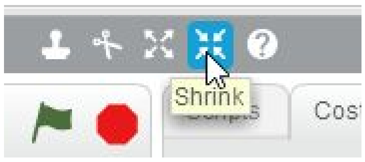


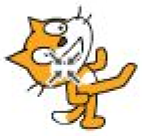


(draw some walls quickly and crudely, leave plenty of room for the cat to fit between)

Great, that's good enough for now - I can come back later to improve it, but first I want to write the scripts so I can check how it works.

I'm also going to shrink the cat a little bit. Click on this shrink icon at the top of the screen, then click a whole lot of times on the cat (on the stage) to shrink it.

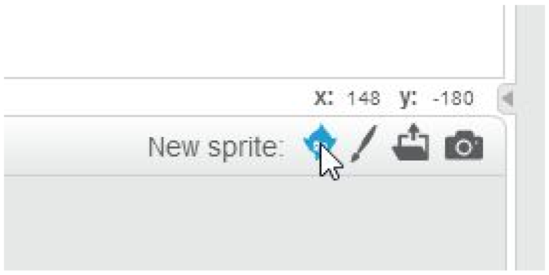




I'll switch back to Scripts *(tab up the top center)*. Now, all my old scripts have disappeared, but that's OK - that's because I'm talking to the background, not the cat. I need to click on the cat in the sprites list *(bottom left)* to go back to seeing the cat's scripts. I only see the script for the person I'm talking to; now I'm talking to the cat.

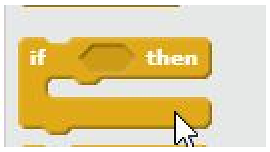
Now i have the cat's scripts, I'm going to make it so the cat goes back to the start if you touch the walls.

Actually, I need to tell it where to go back to, so we'll add a new sprite. Click on the little face next to the words 'new sprite' by the sprites list, and choose a sprite from the library. *(i.e. the Bell.)*



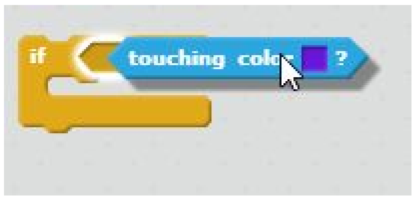
This will be your starting point. Let's move the sprite to the right place by dragging it.
Now we need to click on the cat in the sprites list again, to see the cat's script.

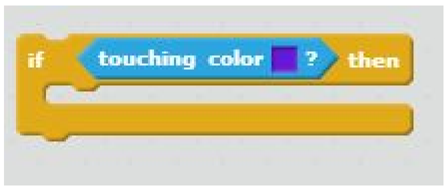
We want to tell the cat: **if you touch something blue, then go back to the start**
First go to the **Control** section and grab the **if then** block.
(drag it over, but leave it disconnected from the rest of the code for now)



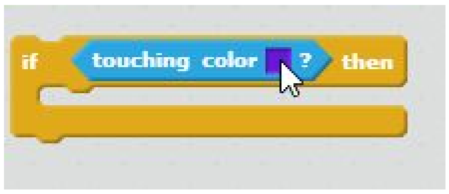
Next, go to the light blue sensing section and grab the 'Touching color' block. There are two different touching blocks, we want the color one.

The touching colour block has pointy ends, and it fits into the little pointy hole in the if-then block (between the word if and the word then). You slide it in from the right hand side until it pops into place.

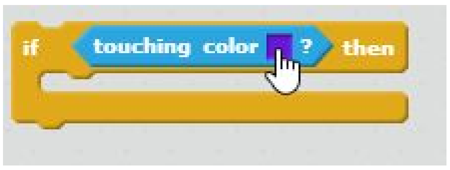




To set the colour, we click on this little coloured square. See how normally my mouse controls a little arrow? If I click once on the square, the arrow turns into a little hand.

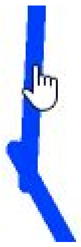


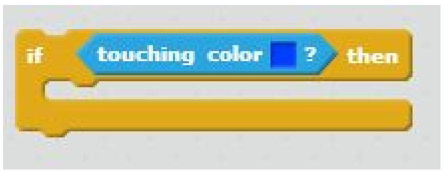
```
if <touching color [ ]?> then
```



```
if <touching color [ ]?> then
```

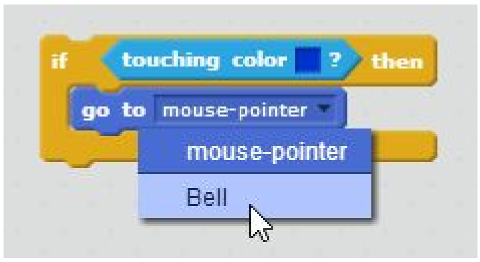
Then use the hand to go and click on the colour you want - like you're putting your finger in some paint. That sets the colour. *(It may be worth demoing this twice as it's hard to see what is happening.)*





```
if <touching color [ ]?> then
```

Now we need to see what happens if you're touching the colour blue. I want the cat to go back to the start - we've moving the cat, so look in the motion section (dark blue). In Scratch, moving instantly like a teleport is called 'go to', so let's grab the block that says **go to [mouse pointer]** and put that inside the if-then block's mouth. Then we click the tiny little black arrow and change it to say **go to [Bell]**.

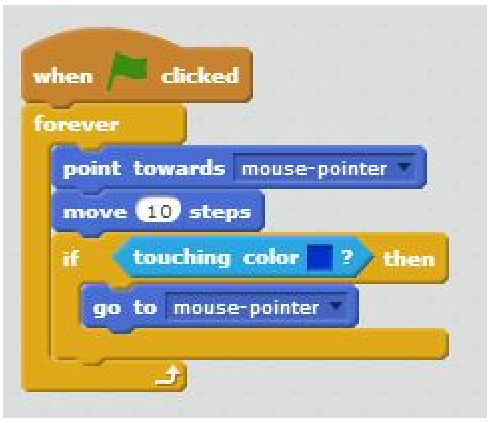


```
if <touching color [ ]?> then
    go to [mouse-pointer ▾]
```

Now, is this going to work? Almost, but we haven't told the cat **when** to ask this question.

When should it read this instruction? We want it to read it over and over again, so it keeps asking this question. It's a bit like when you're driving somewhere, and you want to know "Are we there yet?" - if you only ask once, the answer will be "no" and then you'll never find out when we get there. So you need to ask over and over again. Are we there yet? No. Ok, are we there yet now? No. Keep asking.

To make it keep asking, we put the whole if-then block inside the forever loop. And now the cat has to ask that question over and over, as it keeps repeating all those instructions. This means as soon as it touches something blue, it will read that question and go back to the start.



```
when [flag] clicked
forever
    point towards [mouse-pointer ▾]
    move (10) steps
    if <touching color [ ]?> then
        go to [mouse-pointer ▾]
```

(Demo step 2, moving around and touching the walls to restart.)

That was step 2 - walls we can't touch.

There's just one more thing to do, and then we'll all go off and make our own game following the three steps.

Step 3 - a goal to get to.

Let's make it so you can win the game by getting to the end.

First, we'll add a new sprite to be the goal. *(Maybe some bananas.)* As before, we click on the little face next to the words 'new sprite' and choose a character to be the end point. Then we drag the bananas to the right place.
Now, we need to click on our cat to see the cat's script.

We are going to tell the cat: **if you touch the bananas, you win**.

We'll need another **if-then** block from **Control**. This time we'll put it inside the forever loop because we know that's where it needs to be.

Next, we'll go to sensing and grab the **Touching [blank]** block. Last time we used Touching Color, but this time we'll use the **Touching [blank]** instead. that lets us put in the name of something. That's because the bananas sprite has some yellow, and some black, it's easier to just use its name instead of all the colours.

We click on the tiny black arrow again to say **touching [bananas]**.

Next, we say what happens when you get the bananas. Let's make the cat say "You win!" in a speech bubble. Now, this is a bit tricky - you might think it would be in the sounds section, but it's actually under the **Looks** section. That's because it doesn't really make a sound.

Take the very first looks block, **say [Hello!] for 2 secs**. Put it in the mouth of the if, and backspace the text and type in "You win!".

Now we can test all three steps - a character we can control, walls we're not allowed to touch, and a goal we're trying to get to.

(test all three steps).

Great, that's it - now go and make your own version of the game. We'd like you to follow the three steps first, then once you get to the end, we'll talk about ways you can add your own ideas to the game. Let's go!

Helping students as they work

If a student forgets what to do next, give them the activity step card for the step they are on. Leave the other step cards lying around for reference.

If a student gets stuck or has a weird bug or glitch, help them out. The most common problems and their solutions are described later in this document.

Some students will ignore the instructions and do their own thing. For some advanced students this is OK, but for others it means they are stuck and don't want to admit it. It might be best to help them do the three steps, and tell them they can work on their own stuff after that. The key is that everyone finishes with something working so they feel a sense of achievement.

Some students will finish early. Ask if they can think of anything to add to their game, and give them some of the Scratch cards for ideas.

If everyone finishes early, consider making this a class discussion instead (and then hand out all the Scratch cards).

Here are some achievable ideas to suggest to students. Instructions for these ideas are later in this document:

- control with the keyboard instead of the mouse
- When you get to the end, a new level starts
- enemies that move back and forwards, and you can't touch
- an enemy that chases you, that you can't touch
- coins or fruit that give you points when you pick them up

Helping students publish their work

10 minutes before the end, get the class' attention and ask them to add their games to the Scratch Studio for the day's event.

Tell them the process, which is as follows:

- Click on the 'Share' button at the top right
- Click on the title 'Untitled' and put in your name, so you can find your game again later
- Now to find the studio, go to https://scratch.mit.edu/users/omgtech/
- Scroll down to the Studios, and the Studio for today's event should be the first in the list. Click on it.
- Click Add projects
- Your project should appear at the bottom left of the screen, click on it to add it.
- You're done!
- To keep working on your game, you can click on your project again (in the main area of the screen) to see your project page, then click 'See inside' to get back to the editor.

Go around and make sure every student has saved their work and added it to the studio. For younger children, you may need to do this for them.

Closing circle

5 minutes before the end, get everyone up and away from their computers to stand in a circle. Ask everyone in turn to say:

- something they learned
- something that surprised them

The point of this is to give the activity some closure and let students reflect on what they did.

Finish up by saying: Scratch is free, and you can use it from any computer with an internet connection. Just go to Google and type in 'Scratch', it will come up. As well as making your own games, you can play other people's games, or even look inside them and see the code and learn how they work. We will email you a link to our studio so you can find your game and show it to people or keep working on it. See you later!

After the class leaves

For each computer:

- Make sure it is logged in to the correct Scratch account. (Some students will log out or log into their own Scratch accounts.)
- Click on 'Create' in the top left corner to start a fresh project

One mentor can do this while another is giving the demo.

At the end of the day

You're finished! Hopefully your students had great experiences and there are lots of creative games in the Scratch studio. Send a link to the Scratch Studio to Zoe or whoever's in charge, so they can send out an email like:

> I hope you enjoyed the game making activity at OMG Tech!
>
> If you have a Windows or Mac computer, you can play all our games at this link:
>
> //LINK GOES HERE
>
> You can also create new games, stories or other programs. Have fun!

‘The Science’

Our Scratch game has sprites. Each sprite can have scripts that tell it what to do when the game starts. A sprite can tell when it is touching another sprite.
Many other games work in a similar way, even if they were not made in Scratch. Can you think of some of the sprites in other games you play? (Flappy Bird? Crossy Road?) What are some of the **if blocks** you might see in their code?

Troubleshooting

If many students are having the same problem, try to explain that part of Scratch differently or in more detail when you demo for the next class.

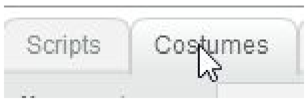
Students having trouble putting the Touching Color block into the If Then block

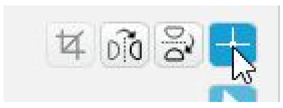
You need to line up the **left edge** of the sensing block with the hole in the if block.

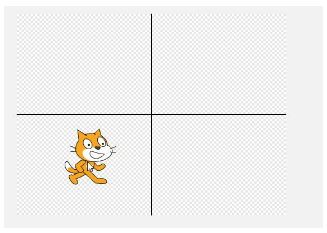
Matt explains this by saying “slide the block in from the right until you see the white border appear” while demonstrating, which seems to work.

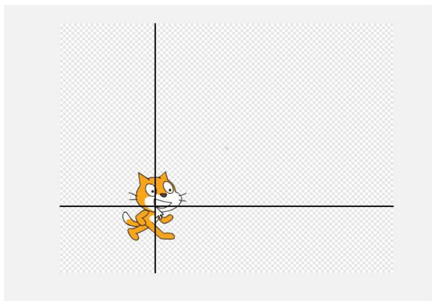
Sprites jumping around, moving in unexpected ways especially when turning

This can happen if the sprite is off-center. Imagine that the sprite is drawn on a see-through sheet of A4 paper. When the paper turns, it turns from the center of the page. If the sprite is on the center, it turns normally. But if the sprite was in the corner of the page, then when the paper turns the sprite actually moves a long way.
To fix, go to the sprite’s costume view and use the Set costume center button to reset the center.









Students use black as their wall colour. The sprite reacts like it is touching a wall when it touches something else

Many sprites have black borders or other black parts, so it is not ideal to use black to indicate walls. Change to another colour.

At the start of the class, mention that students can choose any colour except black, because black doesn’t work well.

A student accidentally deletes lots of blocks

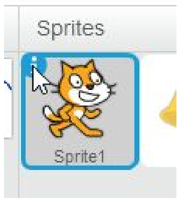
They can get them back by clicking ‘Edit’ and then ‘Undelete’, but only if they do this straight away, before taking other actions. (Matt doesn’t mention this in the demo, but you could if you think it’s a common issue.)

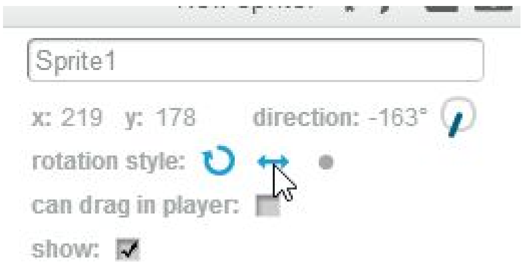
A sprite is leaving a trail everywhere it goes

The student has put down the pen. Add a ‘pen up’ block (green) and click on it once to stop this behaviour.

The Cat (or main character) turns upside down when walking to the left

This always happens but only a few students see it as a problem. If you want to change it click on the ‘i’ on the sprite in the sprites list, then the left-right icon to change the rotation mode.





Sprite1

x: 219 y: 178 direction: -163°

rotation style:

can drag in player:

show:

The Cat (or main character) is going back to its starting point, but is stuck there and can’t move away

This usually happens if the starting point is too close to a wall. As soon as the Cat returns to the start, it is already touching a wall, so it gets sent back to the start again, over and over forever. Move the start away from the walls, OR shrink the cat, OR change the background so there is more room.

This can also happen if the wall colour is the same as a colour used in the starting sprite. This most commonly happens if the student’s walls are black, because many sprites have black parts. The student will have to recolour their walls, and change the script, OR remove all black from the starting sprite.

Students don’t know how to switch to the backdrop

Show the students what to do \ go over this more in the demo next time

Students don’t know how to switch from drawing mode back to scripting mode

Show the students what to do \ go over this more in the demo next time

Students select a different sprite, all their code disappears, they think it is gone

Show the students what to do \ go over this more in the demo next time

Scratch is offline for maintenance

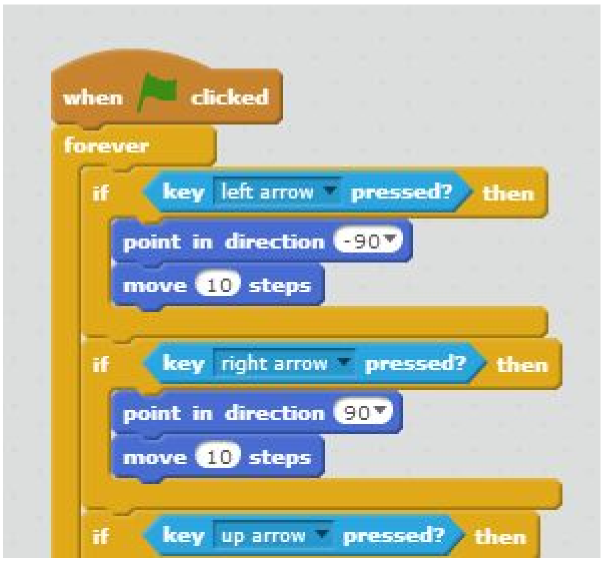
Don’t panic! You need to install Adobe Air and the Scratch Offline Editor on each student’s computer. Then open Scratch (the offline version) and continue as normal. You can download these files from https://scratch.mit.edu/scratch2download/. Be efficient! Put the files on a USB drive. Have one mentor plug it into a computer, copy the files off, remove the USB and move on to the next computer. Have a second mentor follow along doing the actual installs.

In this scenario, students will have to save their games to their computer (by clicking File -> Save) and you will need to use the USB drives to copy the games off each computer at the end of the session, then later painstakingly upload them to the Scratch and add them to the studio one by one.

Extra for Experts

Control a character with the keyboard instead of mouse

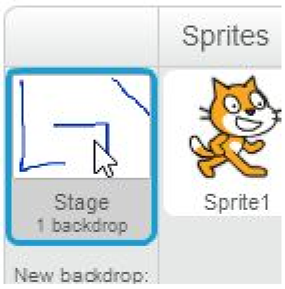
One of the Scratch cards shows you how to do this, but this way works better:

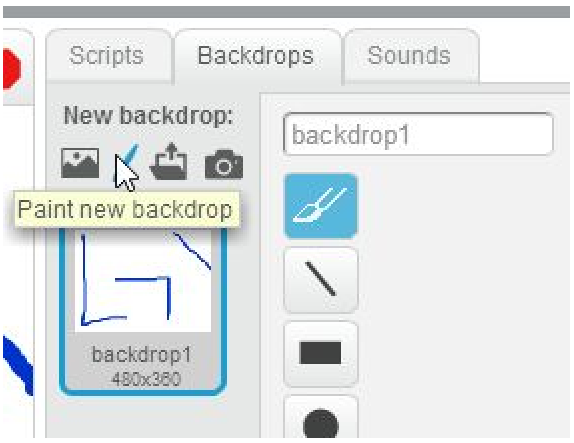


```
when [flag] clicked
forever
  if key left arrow pressed? then
    point in direction -90
    move 10 steps
  if key right arrow pressed? then
    point in direction 90
    move 10 steps
  if key up arrow pressed? then
```

More than one level

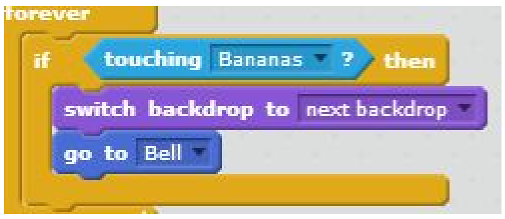
Draw another backdrop:





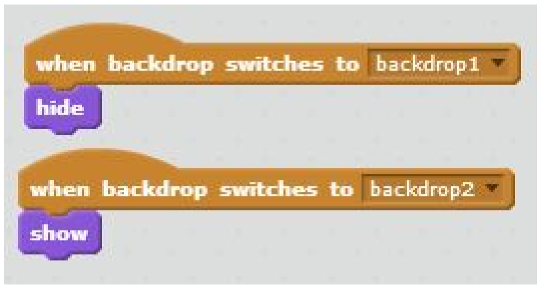
Add some code:

- When the green flag is clicked, Switch Backdrop to Backdrop 1
- When the main character touches the goal, Switch Backdrop to Next Backdrop AND move the main character back to the start



```
forever
  if <touching Bananas ?> then
    switch backdrop to next backdrop
    go to Bell
```

You can make a sprite appear on some levels but not others by using **Event** blocks:

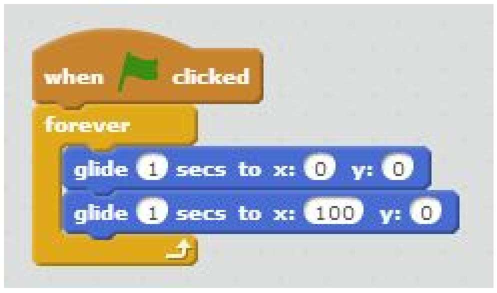


```
when backdrop switches to backdrop1
hide

when backdrop switches to backdrop2
show
```

Make a character move back and forwards

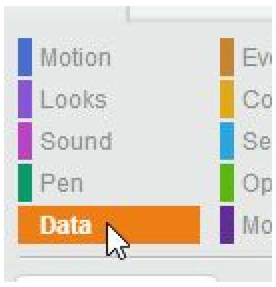
This is great for enemies that you must avoid touching. Change x and y to change the places it moves between. You can add more blocks for a more complex path.

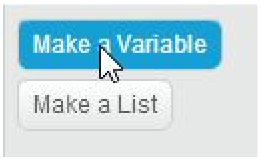


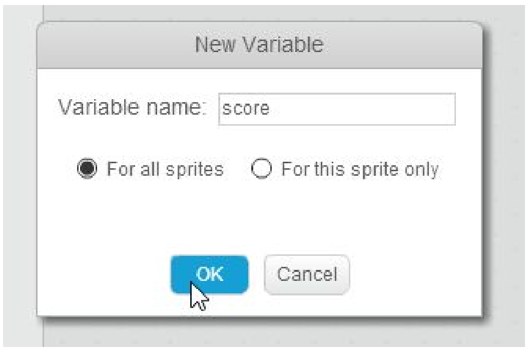
```
when clicked
forever
  glide 1 secs to x: 0 y: 0
  glide 1 secs to x: 100 y: 0
```

Collect apples to get points

Fist add a score variable:







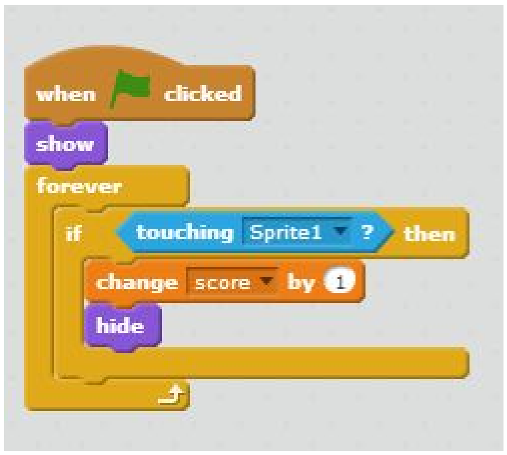
New Variable

Variable name: score

[✓ For all sprites] For this sprite only

OK Cancel

Then in the Apple's script:

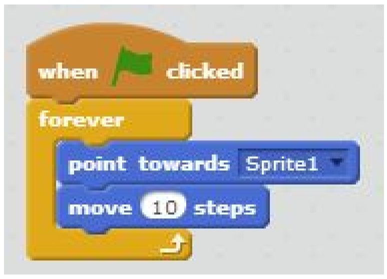


```
when [flag] clicked
show
forever
    if touching Sprite1 ? then
        change score by 1
        hide
```

Also make sure someone has **Set Score to 0** when the green flag is clicked. **Set** means "make it this exact value", **Change** means "add or minus this many".

An enemy that chases you

Give the chasing enemy this code:



```
when [flag] clicked
forever
    point towards Sprite1
    move 10 steps
```

Change the '10' to a smaller number to slow it down.
Maybe make it go to a starting point when the green flag is clicked.