Pīkau Name: What is Programming?

Video Name: The Big Picture of Programming (EMP07-3)

Presenter: Professor Tim Bell

Kia ora tātou. What I want to do in this screencast is have a look at programming. We are going to look at the big picture using a programming language called Scratch. Scratch is popular in schools, particularly for junior classes, but don't be fooled, it's a fully powerful programming language, it's just that often it's only used for simple purposes. But I want to use it to illustrate what the range of programming is. What we are going to do is put together a program. Now, if you are looking at what I'm writing and going 'Well, how would I know how to do that?' that's entirely reasonable because I'm going to be using things we are looking at in later the pīkau; we are going to use things that are coming up. But I wanted to give you a sneak preview to show you the big picture.

The task that I've decided to give myself is to write a program that displays numbers in te reo. I have a final version of the program here, so let's just run it. This is how I've chosen to make it work - there's many ways it could work, but this is this particular version. The cat says 'What number shall I display?' and I type in a number like '3', press 'return', and it says, 'That number is toru'. Fantastic. Then it comes round and says 'Would you like another one?' I can type in, let's put in the number '12', and the cat says well, 'That number is tekau mā… rua'. Tekau mā rua is the full name and so this one can display a whole lot of different values. There's 14: tekau mā whā. Let's just check it with another one, 5, and that number is just 'rima'. So, we've got a program here that displays numbers in Māori. Fantastic.

I'm going to write this program starting with nothing, and we will put the thing together. I just want to remind you that you're not expected to be able to do this but I want to show you what the process would look like to give you the big picture.

I needed to plan how it would work, and since you've seen the final version, you've got a rough idea of that, but I'm just going to put a comment in here (I've grabbed this from the internet, in fact), a list of what the main numbers are and how they should be displayed. That's just a guide for me, that's not part of the program, it's a comment, so the computer's just going to ignore that but it will be useful as I develop this program.

With Scratch programs we normally tell it the first thing is to wait for the green flag to be clicked and then it's going to do something. In the program that I've designed the first thing that happens is the cat asks you what number you want displayed. The command to get the cat to ask something is this one here, and we'll say 'What number shall I display?' That will make the cat, well, let's just try it, because there's no harm in trying it out so far. So there we go: the cat says 'What number shall I display?' oh, I've found a bug already, I've accidentally

typed in a greater than sign. I can fix that in here, that should be a lot better now, I'll run my program again. The cat asks for a number, I can type in a number, and, of course, it does absolutely nothing with it. Now, what we want to do is stash that number away somewhere so that we can refer to it later. For that we have what's called a variable. I'm going to call that variable the 'number to convert'. The idea of a variable is quite a deep one and I won't get into it too much but suffice to say it's like a box where we store values, and we can store things in it and get them out later on. The way Scratch works is what you've type into that 'ask' thing comes up in a thing called 'answer', so my 'number to convert' is going to remember that. It actually shows up in the top left-hand corner here what the value is. Let's run the program again, see what's going on. The cat says 'What number shall I display?'. Let's type in '4' and you notice that, that variable has stashed away the number 4, I can use that later on if I need it.

What we want to do next is, of course, display that number in te reo. I can do that. There's many ways of doing it actually, and I'm not going to do it the most efficient way, but this is going to be a way that, hopefully, is fairly straightforward to understand. In Scratch, and in most programming languages, we have a thing called an 'if' command that checks for some kind of condition. The 'if' command, I'm going to check if the 'number to convert' is equal to a particular value, and, in fact, if it is equal to 1, then I'm going to get the cat to say what the word is for one: tahi. There's a lot going on here and I'll just briefly explain it, but we will be going over this in a lot more detail, but basically this 'if' statement says 'well, we've got this thing called 'number to convert', and check if it is equal to 1 and if it is equal to 1 then do this thing inside the yellow box here, which is to say 'tahi'. If I run that the cat says 'What number shall I display?'. I'll try that with the number '1'. The cat says 'tahi', fantastic. If I type in a different number of course, I haven't placed anything in there, let's type in the number '2', and the cat doesn't do anything. So I need another 'if' statement for the number '2', and what it's going to do is it's going to say, well, 'if the number…', well it's not going to 'say', it's going to have to decide, if the number that I'm converting is equal to 2, then I'll get it to say 'rua' for two seconds. So let's test that out. If I run it and I type in the number '2' then the cat says 'rua'. I'll just test it again, if I type in the number '1' then the cat says 'tahi'. Fantastic, I've got those two going.

Now, another thing that happened it the program that I was demonstrating before is it kept on asking for more numbers. That requires a loop, it requires it to do things over and over. Over here I have a thing called the 'forever' loop, and I'm going to put that around all of my commands. That's going to mean that when it gets to the end of that it goes back to the top again and keeps on doing all of those commands over and over. Bear in mind it's still a sequence, it goes from beginning to end, it's just that the forever loop says 'when you get to the end go back to the beginning again'. So we've got a sequence within a loop, and now we are starting to get a lot of the elements of programming.

Let's just try out this program first. I'm going to run it. It says 'What number shall I display?' We'll type in '2', it says 'rua'. Then it comes back and says 'What number shall I display?' I'll type in '1', 'tahi', and 'What number shall I display?' I'll type in '3' and the cat doesn't do anything with the number 3, it justs asks for another number to display because I haven't put in the 'if' statement to do that.

I'll just quickly put in some more 'if' statements so that we can do some more numbers. One trick you can do is copy and paste these because I've got most of the pattern there so if I duplicate that, and I duplicate it again (I've now got room for two more numbers) and, of course I need to change that so if we get a three or a four then we need to have 'toru' or 'whā' (make sure we get the macron right). Okay, so hopefully the program will work now, let's test it again. We always test our programs because that way if I make a mistake I know it was just in the last few things that I did. It says 'What number should I display?'. Well, let's try 4, there it is, 'whā'. Let's try 3, there it is, 'toru', and so on. I'm going to just stop there. Hopefully you can see how you would do it for the full set of numbers at least up to 10. Beyond that it gets a bit more complicated. But what I have in this program is all the elements that are important when you are learning to program. It has the six elements. These elements appear in the Progress Outcome so it gives us all the raw material to talk about the Progress Outcomes.

The first one is the idea that we should be able to ***input*** a value. That's going to come up in Progress Outcome 3. The 'ask' command is what allows us to input a value. It takes input from the outside world. It's not the only way of getting input for Scratch or for programs in general, it could be from the mouse, it could be sensing it from a video camera or something like that, but we have some kind of input.

The next thing is the ability to ***store*** a value. We've got this 'set' command here that says 'store the answer away in the value called "number to convert"' because we want to refer to that later, in fact, we keep on referring to it down here. If we were doing numbers beyond 10 we'd probably have to manipulate it a wee bit as well, so we've stored that away. The ability to store data is really important in computers. One of the main ways we do that when we are writing a program is in what's called a variable, the thing that actually stores values for us. We could also store data on a disk or in a database or something like that.

The next element that's quite important is the idea of ***selection***. Selection is simply the 'if' statement. Selection means that we are making a decision, shall I do this, or shall I do that, and so on. It's the 'if' statement that gives us the ability to say 'well, if they've typed in the number 1 say this, if they've typed in the number 2, say that.' You may have noticed that under the operators for selection here I could check if a value was equal to something, but I could also check if it was less than or if it's greater than, and there's a few other things that I can check about it as well, which is actually quite powerful. You might check that someone's age is over a certain limit or that their date of birth is more than a certain value. We could be checking whether the number of cars in a carpark is more than the number of spaces available, in which case it would say 'car park closed'. There's so many things that the 'if' statement is used for just to check values.

So that's called 'selection' and that enables us to check values, to make decisions. That's going to come up in Progress Outcome 4 so we'll look at it in more detail there.

There's also the need to be able to have ***output,*** and in this program that's the 'say' command. Again, there are other ways of having output, but 'say' enables the cat to say

something. The output might also be moving things around or making a sound or displaying an image, things like that. As long as we have output, some way of getting something out of the program, then the user can see the result of the program, or see the effect of it. Output - we actually already have used in Progress Outcome 2 when we thought about the little robots and how they move around and where they end up, that's the output.

The next thing that's an element of computer programs is ***iteration***. We've got that with this 'forever' command. Iteration means doing something over and over, and that's a really powerful thing about computers: they can do billions of operations every second and so you can just say 'do this a million times'. As long as you write a short program and get it to repeat a million times it will do that so you can deal with a million customers, or a million pixels in an image, or things like that very quickly by writing a loop. The technical word, iteration, is anything that enables us to do something over and over. The form of iteration I've used here is the 'forever' loop. But there are other ways of making it do things over and over.

You'll also notice that one of the things I was doing was constantly testing the program. That's an approach that's really important. You'll see that in the Progress Outcomes. It's not so much a property of programs, but testing - trying to find what's out wrong with them, trying to find out the edge cases and so on - makes sure that it's behaving so that when someone goes to use it, you know, they don't get surprises, that's really important. As we test programs, of course, inevitably we will probably find some bugs. It's really important that we get used to the idea that there probably are bugs there, we need to track them down, and we need to fix them.

For an example of testing and debugging let's go back to the bigger program that I'd written before, that I'm basing my new one on. I've got it here in full screen mode so we can't see the program, of course, that's the experience the user will usually have. Now, to run a Scratch program normally we click on the green flag and my program asks me for a number that it will display for me. So we will start with, say, the number 5, and we will check on that and it says 'That number is… rima'. That's good. Let's test another one. I'll try a number that's in the 10s because that's going to get a bit more complicated, so I'm going to put in the number 13. And the cat comes back and says 'That number is tekau mā… toru'. That's good, that's accurate. Now, it's always good to test more extreme examples so let's try the number 20, because that's actually going to be out of range, and just check that it does something sensible. That's good, the cat says 'Sorry, that's too big for me.' So that's good. Now let's try a number that's near the end of the range because it's always at the boundaries where you often get things going wrong. I'll try the number 19. The cat says 'That number is tekau mā…' Oh, it didn't actually display anything. It should have displayed more than just 'tekau mā…' which is the 10 part of it. So there's something wrong with the number 19. Just to check things out, let's see if it can do 9 correctly, because that's related. No, it didn't display anything for 9 either. So there is something going wrong around the number 9 in my program. Let's go and have a look at the program and see what's going on around the number 9. Here's the actual program (we'll make this available on line, you'll be able to follow as link to get at the program). If we look down here it's saying 'if the number displayed is 8 then say 'iwa'.' I've found the bug already because, of course, iwa is the number 9 so that should be saying that that's the one for number 9. Then if we look at number 8 we see

that that's not correct either. In fact, if we go back, I put 6 in twice here, so, in fact, we had 6 as ono and then, also, whitu which should be, of course 7 and I'll put in an 8 there. I think that's it! I think I made a typing mistake, and I think I've got the bug out of my program. Now I can go back and test that, actually, lets do that. So we'll go into fullscreen mode, we'll run the program again, and this time I'll try typing in the number 19, and it says 'That number is… tekau mā… iwa'. Great! I think I've fixed that bug, I need to do a bit more testing to be sure. We'll make this program available to you. You can test it. You can experiment with it, and maybe even add some more features to it if you want to, for example, go beyond the number 20 and things like that.

So that's what's involved in testing and debugging. We really want to make sure that we find out the problems before the person who is going to use our program finds them for themselves.

Now we are starting to get an idea of what programming is about. The key thing is that there are these six building blocks that keep coming up. Any program that you write in any programming language ultimately will come down to manipulating these six blocks, these six ideas, in different variations. We need to be able to have **input**, we need to have **output**, we need to be able to **store a value**, we need to have **sequence** (the idea of doing one thing after another), we need to be able to have **selection** (being able to make a decision, which is usually an 'if' statement), and we need **iteration** (the ability to do things over and over, in this case, it was the 'forever' statement).