Pīkau Name: Computational Thinking: the International Perspective

Video Name: How does it relate to computers?

Presenter: Professor Tim Bell

One of the questions that comes up around computational thinking is 'whether it's all about computer programming or whether you can do it without programming? Or whether you have to do programming to call something computational thinking?' I thought to explore this, it would be easiest to actually do a little programming. Now I appreciate that at this stage you might not have encountered programming as an activity, or maybe you have, but I'm going to use a system called Scratch, which is quite widely used in primary schools, and it is a programming system – it's a full blown system that allows all sorts of programming to happen.

We're just going to write a simple program. In fact, this program is going to involve sound, and what I want to do is write a program that plays the tune Twinkle Twinkle Little Star. What I am doing here is putting together the elements of Twinkle Twinkle Little Star as a sequence of notes. Now, the way this system works is when I click on the green flag, that's over here, then it will follow my instructions in sequence. I've just told it to play this note three times so we'll just play that (music). Not very interesting, but actually the first two of those notes I will use to start my tune, and here I'm putting in a G, which is the next note from Twinkle Twinkle Little Star; and I'll put in another one, and it's going to be another G, and let's just play that so far (music). Ok it's starting to get there, now we need a few more notes so let's just put those in quickly.

Ok, let's test my program and see how it sounds so far (music). OK, pretty good so far; let's carry on and I'll put in a few more notes. OK, let's try out my program now (music). Oh (music). Ok, I heard a bug in my program because it didn't pause long enough between, at the end of that phrase. So let's try and de-bug it. Now don't worry too much about these details, but it was around about here, that note there, instead of being half a beat it should have been one beat. Again, I'm not trying to teach you how to program at the moment, but just the kind of processes that are involved. So let's see how that sounds (music). Oooo, ok, another problem with my program. Now when people are de-bugging programs it will often be displaying the wrong thing, or giving the wrong answer, or not letting you type something in when you should, the bug here is one that you can hear. That last note wasn't quite right; and in fact, there it is (music). I will put the right one in, and in fact it needs to be a longer note as well. Let's carry on. Should we check it? Ok (music). Ok, now I'm going to carry on a wee bit, but you can see that the program is getting a little bit long and it would be better if I could break it up into sections. So let's create the next section of sound. Ok I've written this phrase in, let's just have a listen to just that phrase, I can do that by just double clicking on this phrase (music).

Now that's pretty good, but of course that phrase is about to be repeated again and I don't want to do the whole thing, you know type it out again, luckily there's a thing in here that allows me to repeat that whole phrase twice. Let's try it (music). Ok, now of course that's a musical idea, repeating a phrase, but actually in computer science and in computer programming, the idea of repeating or iterating over something is a very powerful idea. Normally we don't just repeat things twice, we repeat them lots of times. Ok, let's see how that sounds (music). Very good, the last phrase is going to be exactly the same as the first one, so there is a little trick I can do. I can take this whole thing and I can make up my own block which I will call, I don't know, "opening phrase", and I will define this thing called the opening phrase (music) and so on, you get it?

Now I can say that when I play this song I want the opening phrase, then I want those two repeated (put the opening phrase over there) and then I want the opening phrase again, and I think I have put together the whole song. A quick

listen (music). That's the opening phrase (music continues); that's that phrase twice (music continues); and now the opening phrase again (music). My program is working.

Now if you've done much with programming you will be aware that there's other ways that I could have done this. For example that thing that was repeated twice, I could have created a block that does that as well and just repeated that block twice. One of the things about writing programs is there's never one correct answer – there are lots of ways of doing it – but I've got one that works. The main point of this is that I wanted to illustrate some ideas from computational thinking.

I've been talking about computational thinking: we've had things like **decomposing** a problem into parts, and you notice that here I have decomposed the song into an opening phrase, and then this little bit, and then the opening phrase again. So that's just one of the things when people are putting together a program: they need to be able to break it down into parts.

Another of the concepts was the concept of **algorithm**. Here I've got an algorithm happening because I've given instructions – the instructions were 'to do the opening phrase' and then it was to 'repeat something twice' and so on. It's a very simple algorithm, but it *is* an algorithm; it's the process that we are putting together.

Another idea in computational thinking is **abstraction**, looking for general cases; and a good example of that is when I came up with that idea of instead of doing that opening that phrase over and over every time I needed it, I abstracted it out, I gave it a name, I put it aside and I just used that name whenever I want to use it. That's quite a powerful idea that, that we can abstract out things that we want to do a lot and you just get it right once and use it when you need to use it.

There are other ideas in computational thinking as well but I just wanted to show you how writing a computer program really exercises them, now you get back to that question of; do you have to write a program to be doing computational thinking? Well if computational thinking is about *computation*, then it is limited to what computers can do, what digital devices can do. So the only way you can tell a computer to do something (give it instructions) is by programming, and in that sense, computational thinking has to be about computation. The machine dictates what we can do. There are somethings I can do by writing a program, such as the one I've done [here], and there are some things that you cannot do. You can't give instructions like, make it sound better, or just add a bit more feel to it, or something like that. That doesn't mean anything to a computer; it might mean something to a human who you were asking to do this but computation only allows a very limited number of things to be done. Those limited number of things do all the stuff that we see in our digital world so they're very powerful, but nevertheless there's a limited number of things.

From that point of view, yes, ultimately when a student writes a computer program they're showing that they can absolutely articulate what the process is that they've come up with in a way that a computational device can do it, and of course the device itself will do exactly what they said. If they make a mistake it will follow that mistake, and in that sense it is when things get onto a computer that computational thinking really comes into focus. On the other hand, there's a lot of things you can do away from a computer, and we've already done some of those exercises, just working with physical cards, and with running around in the playground, and things like that. Giving instructions obviously helps students to get ready for doing this kind of thing, but there is a risk that if you're off the computer that they can give instructions that are more meaningful to a human, but perhaps couldn't actually be done in a computational setting.

I don't think there is a black and white answer of whether you have to or don't have to have a computer. You can think about this a lot away from a computer and it's also very valuable to do it on a device, most of the research these days is pointing to the idea that we should do both. Thinking away from the computer enables students to reason and use a lot of logic. Putting it onto the computer forces them to get their ideas straight, so computational thinking can really involve both.