iCompute

Computing Pedagogy

Insights

www.icompute-uk.com

Computing Pedagogy

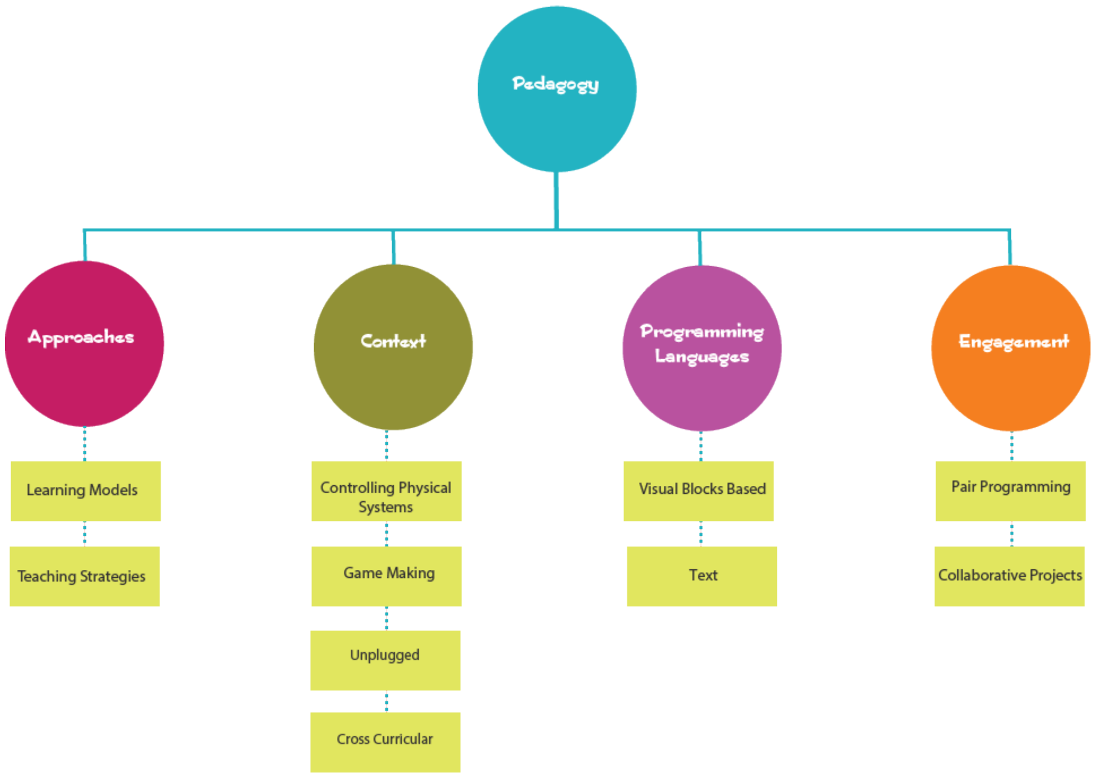
Liane O'Kane

Since the introduction to National Curriculum for Computing at Key Stage 1 and Key Stage 2 in England 2014, it has been a child's statutory entitlement to a computing education from the age of 5. There have been many challenges along the way since 2014 for primary teachers, not least, due to the subject being introduced throughout schools where the vast majority of teachers had never been trained to teach it.

Despite a number initiatives to improve teacher subject knowledge, notably driven by Computing At Schools (CAS) and the Network of Excellence (a grass-roots organisation I represent as a Computer Science Master Teacher) the Computing Education Project Report (The Royal Society, 2017) - exploring the issues facing computing in schools - concludes that computing education across the UK is 'patchy and fragile'. There is much to address in a system where many teachers do not feel confident teaching the subject and are in need of significant support.

According to the Royal Society (2017) research is needed to support teachers in understanding computing pedagogy, and those that are proven to be the most effective. I've been teaching computer science for years, well in advance of the National Curriculum. I'm a computer scientist and a primary teacher. Here, I hope to help other teachers get to grips with pedagogies they can use in their classrooms; which may offer some support towards teaching computing effectively and help pupils move towards computing mastery.

I've adapted the Royal Society's (2017) categorisation of computing pedagogies (for the purposes of literature review) to this for teaching purposes:



Approaches

Learning models is a tricky area for primary teachers as many feel that their pupils know more than they do. I've spoken extensively about this in the past on my blog and elsewhere and passionately believe that opting out of teaching computing by only providing one way of learning is not acceptable.

Copy-code and tutorial based approaches are understandably popular in schools given the limited input required of the teacher but we are doing our pupils a disservice by only using this or Use-Modify-Create models.

I believe in using a blended approach where pupils are engaged in making creative products that are open-ended, include lots of discussion, are scaffolded and use problem solving approaches to design solutions.

Designing solutions develops the fundamental computational thinking skills that lie at the very heart of the National Curriculum. Encouraging pupils to make 'problems' easier to solve by splitting them down into more manageable parts (decomposition) and taking the detail out of problems to develop solutions that solve multiple problems (abstraction) are key life skills – not just useful in programming.

For instructional techniques, I use teacher modelling a lot. As I do, I 'think aloud' talking through choices, asking for input/feedback. I frequently make mistakes and, when I do, emphasise the importance of that to my learning. Learning in computing is as much about developing resilience and persistence as it is about understanding how computers and computer systems work. It's important to let your pupils know that things going wrong is not only completely normal, but useful.

Context

Controlling Physical Systems

Controlling physical systems forms part of the National Curriculum for Computing at Key Stage 2 objectives. There are many ways of meeting these objectives from BeeBots at Key Stage 1 through to LEGO WeDo, Sphero, drones etc. As I have written previously on this blog, programming physical systems means that fundamental principles of computer science are applied and made easier as models and devices can be designed, constructed, programmed and executed in front of pupil's eyes. This makes it much easier to learn what robots can and cannot do.

Thus far, there is no evidence that teaching controlling physical systems improves learning and attainment in computing (Waite, 2017); therefore there is no need to spend vast amounts of money on new equipment if your school has limited funds. You can still meet National Curriculum primary objectives using the alternative '*simulating* physical systems'.

Game Making

I use game making pedagogy extensively in my teaching and my computing schemes of work. It helps set the children's learning in a familiar context, motivates and engages all learners; including girls. I'm passionate about exciting children about the art, as well as the science, of computing. Setting creative, open-ended, games projects means pupils can apply and develop their computational thinking skills and are free to get creative and extend their knowledge and skills by engaging in discovery-based learning.

Unplugged

Every one of my lessons, from EYFS through to Year 6 involves an element of unplugged activities – teaching computing without computers. When I first began developing iCompute in 2013, unplugged activities were primarily planned due to a lack of resources. In the years since, experience has taught me that these activities away from devices and computers are crucial to young children's learning in computing. They help them understand abstract concepts and deepen learning.

Cross Curricular

Computing is one of the most fundamentally cross curricular subject areas in education. The best primary practice includes a blend of rigorous, discrete, subject teaching and equally effective cross curricular links. Both approaches are needed for effective learning to take place, to enable children to make links between subjects and to set learning in meaningful contexts.

I recommend to all schools that they teach computing discretely throughout the primary phase in order to develop the knowledge, understanding and skills required to fully meet the objectives of the National Curriculum. These can then be further developed in other subjects in order to enrich and deepen learning through engaging, interconnected, topics without focusing on skills acquisition and potentially adversely affecting learning in the other subject(s).

I dedicate at least 40 minutes per week for Year 1-6, and 15-20 for EYFS as well as embed it in other subjects enabling pupils to make authentic links.

Programming Languages

It is important to note here that the National Curriculum does not specify any particular languages to teach in the primary phase. What it does specify is that pupils should use a ***variety*** of software to create a ***range*** of programs, systems and content.

I've commented before on my frustration with schools only using one programming language for six years – Scratch. Not only is that incredibly boring for the pupils, it fails to give them the opportunity to use and apply their knowledge, understanding and skills in other contexts and environments. Computational thinking is not language specific. Abstraction, decomposition, generalisation, pattern making, logical reasoning, testing and debugging is universal. Please use as many tools and programming environments as you can with your pupils. There are so many out there and most are free.

In the primary phase, I recommend focusing on using visual programming languages rather than a text-based language. This means your pupils are not constrained in their creativity and development by the syntax and typographical errors that are inevitable with text based programming languages.

In Upper Key Stage 2, I transition some pupils to text-based programming languages once pupils are on their way to computing mastery. At this stage, they perform better in a range of text-based programming activities, recognise programming concepts earlier, have fewer difficulties and higher confidence (the Royal Society, 2017).

Engagement

Pair programming

I've long used pair programming for learning and have spoken previously of its use as a powerful tool for progression and engagement. It has had such an impact on pupil progression and computing attainment in my years teaching that I recommend its use regardless of access to technology.

Pair programming is an approach to programming where two people work at one computer to complete a single design, algorithm, coding or testing task (Williams & Kessler, 2000). One person takes the role of the pilot, controlling the computer/device, and the second person is the navigator. The navigator monitors progress, reviews the program and checks against the design. These roles are periodically swapped. Changes to the program are only made by mutual agreement.

In primary settings, classroom management can be an issue with this approach. Pupils need to be carefully paired to ensure that learning is beneficial for both and not simply used to scaffold less able pupils. Equal time spent as 'pilot' and 'navigator' needs to be managed; discussion and collaboration encouraged. I tend to swap regularly – every five or so minutes – otherwise risk the navigator loosing focus and going off task.

Collaborative Working

Working collaboratively in groups is a constructionist approach to learning: children learning with and from each other. It also most accurately represents working environments in the tech industry where teams work on projects together.

When designing progressive units of work, I like to design activities that require pairs or groups to work together to design and develop something. This can be anything from making a LEGO model to a multi-level Xbox game or creating a web page about a cross-curricular topic. This means the children use and develop the problem solving approaches and computational thinking that I believe underpin computing and equip children with skills that are useful in all areas of their daily lives.

As the Royal Society observes, there is much research still to be done so that teachers can be fully supported with evidence about which pedagogies work best for teaching and learning in computing for mastery. In the meantime, teachers sharing their experiences and best practice should hopefully see us moving, together, in the right direction towards achieving the National Curriculum's purpose of providing 'A high-quality computing education [that] equips pupils to use computational thinking and creativity to understand and change the world' (DfE, 2014).

References:

- The Royal Society. (2017, Nov). ***After the reboot: computing education in UK schools***
- Crick. (2017, April). Royal Society Computing Education Project Review of Literature: Literature on effective computing pedagogy.
- Waite, J. (2017, April). Pedagogy in teaching Computer Science in Schools: A Literature Review *Queen Mary University of London and King's College London*.
- Williams, L. A., & Kessler, R. R. (2000). All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM*, *43*(5), 108–114.