Learning Computational Thinking Through the Use of Flash Action Scripts

Preparing Trainee Elementary School Teachers for Teaching Computer Programming

Erni Marlina Saari, Peter Blanchfield and Gail Hopkins
School of Computer Science, University of Nottingham, Jubile Campus, Nottingham, U.K.

Keywords: Computational Thinking, Flash Action Scripts, Trainee Teachers.

Abstract: The teaching of Computer programming is now mandated in UK state run primary and secondary schools but few elementary teachers have any exposure to programming and are generally from backgrounds that have not given them fluency in using such technology. This paper outlines an attempt to develop a training scheme for trainee teachers that will introduce them to computational thinking through the use of the Flash Action Script Development environment. It is believed that learning how to use this tool will provide them with greater motivation to learn how to program as the scripts will be used to develop teaching instruments that they might use in their classroom. The current paper reports progress on the development of a tool that will allow the teaching of Flash Action Scripts and the testing of this approach on groups of trainee and experienced teachers. It reports on an experiment and the relative level of enthusiasm, motivation and attitude of the experienced teachers and the trainee teachers both before and after exposure to the pilot tool. As expected the trainee teachers had a more positive attitude towards the potential of the learning tool but both groups had improved attitude and enthusiasm after the experiment.

1 INTRODUCTION

The Computing and Schools (CAS) initiative in the UK was launched in 2008. It was in response to concerns that previous moves towards teaching Information & Communication Technology (ICT) in schools has led to misleading pupils as to what constitutes Computing and demotivation of pupils who already have Computing skills (Crick and Sentence, 2011). This led in the autumn term of 2013 to a program of curriculum changes to increase the degree of computer programming taught at secondary schools. This was also influenced by a response to leading industrialists asking that schools should be teaching the basics of programming to a wider audience. At the start of the 2014 autumn term this process was extended to cover elementary (primary) schools. This second process is likely to be even more difficult than the first as the nature of the elementary school teacher training programmes is not to produce teachers who are specialists; rather it necessarily attracts and develops generalists. The generalist nature of teaching in elementary schools has already meant that earlier programmes to promote maths and English teaching in elementary schools has required retraining programmes for teachers in those schools and further development of subject specialist teachers in those schools and school districts (Williams, 2008).

The CAS initiative has been built around certain facilitating universities with specialist teacher training or retraining programmes. For example Nottingham Trent University offers an MSc degree titled Computing in Education that includes modules that help teachers to learn how to teach programming skills. Regular meetings of the CAS allow experienced teachers in different areas to share their findings on how to run programming classes in schools. Different approaches have been taken but the focus must now be more on training the elementary teachers to be able to handle the material as they are seldom from a technical background and generally have less natural inclination to program. Many elementary teachers are in fact quite nervous of the whole area of technology. (See for example teacher comments in Section 4.1.)

2 THE ROLE OF COMPUTATIONAL THINKING IN LEARNING TO TEACH PROGRAMMING

Much work is currently being undertaken to help learners overcome the inability to program. Of particular value is work being undertaken with young learners themselves. Scratch (Franklin et al, 2013) and Python (Begosso et al 2012, Bogdanchikov, Zhaparov & Suliyev, 2013) have been popular for these studies. However, based on our experience with many elementary teachers, using a visual system like Scratch is off-putting due to its child-centred look and feel, and teaching a programming language like Python can seem daunting due to its considerably increased complexity over Scratch. Either way, programming languages seem intrinsically hard to learn and most of the examples of using them are of little intrinsic value to teachers. The teachers can see that children might be drawn to programming games and animations but feel a level of fear when approaching the idea of learning programming themselves. We illustrate this further in Section 4.1. below.

Much of the work of the CAS has been to emphasise the use of "Computational Thinking" (CT) as a valuable thinking skill for all aspects of life (Wing, 2006, 2008, Grover and Pea, 2013). CT involves using the aspects of developing computer solutions to problems, algorithmic thinking, decomposition, abstraction, generalisation and evaluation (Dorling 2014). Wing (2008) has stressed that CT can be adopted in everyday life but it is doubtful that many people would consciously apply many aspects of this thinking when approaching most real life problems. In everyday life it is probably more appropriate to apply aspects of CT such as evaluating, looking for generalisations and use of decomposition. For example, a satnav will work out computationally the best route to take according to the algorithm it has been programmed to use but the user needs to evaluate the solutions and understand what is being prioritised in order to know whether to take the advice being given. Generalisation is used when we try to see how previous solutions might easily be adapted to new problems. Decomposition is used when we look at how to break down problems into parts that can be tackled more easily than the whole. For some groups the development of algorithms may be more valuable – for example those learning to program. In that regard we must now include elementary school teachers in this group as they will need to have some appreciation of how algorithms are designed and implemented in order to lead the pupils towards this understanding. To some extent this is going to be limited by how much abstraction an elementary school pupil is able to understand. Languages that are being used to teach programming in elementary school usually try to make the algorithms as concrete as possible, by using graphics and games design.

The motivation for the research presented in the current paper was our prior experience in trying to teach trainee teachers how to use ICT tools at a teacher training institute in Malaysia. Problems encountered in practice were largely due to a lack of computational thinking in the process applied. The students had a tendency to "copy and paste" examples which they felt were similar to the task they had to do rather than analyse the task, break it down into sub tasks, understand the set of steps needed to achieve the sub task and so on. This lack of CT may have been largely due to their unfamiliarity with building such processes. However, it leads us to pose the question: "how can we encourage primary school teachers to learn to think computationally and thereby be able to learn the languages they need to deal with to teach young children to program also?" Coming from a non-technical (engineering/physical science) background can make elementary school teachers fearful of using technology in general and view learning to program as a difficult task. One aspect of this, from previous observations and discussions, is their lack of intrinsic motivation to learn programming. A first step in answering the question of how to encourage them to think computationally will therefore be how can we provide intrinsic motivation to think computationally? The position taken in this paper is that using an approach where they can see that they could gain facility in using a tool that will enhance their ability to produce really attractive teaching aids will enhance their motivation to learn to think computationally if computational thinking is intrinsic to using that tool. Lin (2012) highlights that educators can become courseware designers and no longer just play the role of submissive users but instead become creative developers. For this reason this research has chosen to look at the ways in which teachers can learn to produce Flash Action Scripts to develop attractive teaching aids. It is believed that this will enhance their motivation to learn to think computationally and we move on to discuss this further in the next section.

3 TRAINING OF TEACHERS IN THE USE OF EDUCATIONAL TECHNOLOGY USING FLASH ACTION SCRIPT

One of the challenges identified for a teacher education program is engagement of pre-service teachers and teacher educators in conversations about their attitudes regarding the role technology should play in teaching and learning. Hammond (2013) mentions the significance of ICT even if engagement is not to be taken for granted and the effect of the learning is complicated. Others have sought to emphasise the need to integrate technology into education. For this to be done successfully it needs to be integrated into the training of the teachers. The arguments around this have been summarised in Tondeur et al (2012).

It is our aim that we can develop an electronic learning tool specifically aimed at helping trainee teachers in which ability to use Flash Action Scripts and the Flash development environment are the learning objectives. Utilisation of such an environment builds upon the views expressed by others of the value of using instructional video in e-learning (Zhang et al, 2006, Wieling & Hofman 2010).

The interface that is provided for the Flash Action Script developer is quite complex but has a lot in common with other IT tools that users will be familiar with – for example Photoshop and storytelling software like Microsoft Photo Story. Skills learned will also be generalizable to other animation development tools such as Anime Studio, Blender and even the Unity 3D game engine. The scripting language itself is similar in form to many other programming languages but the purpose of the scripts – such as actions performed when buttons are clicked – has a much simpler to understand connection to the purpose that is being looked at.

During interviews many of the teachers and trainees met during these studies indicated that they were familiar with using storyboarding software, for example, to produce artefacts to aid their teaching. The actual level of scripting needed to make simple animations within a Flash program is low. However as the complexity of the artefacts increases so also does the complexity of the scripting needed. This therefore potentially leads to a staged introduction to the scripting and at the same time help to develop the aspect of algorithm design within computational thinking. It can also introduce the concept of generalisation as the same basic format will be needed in many different problems. As the user progresses aspects of abstraction will also be possibly developed, for example through the use of functions to undertake often needed processes.

The ultimate aim of our learning environment is to make this a self-help tool that will analyse a user's performance in lesson exercises and that any lesson will be automatically adapted to a given user's needs. In order to investigate this a pilot tool was developed in which a set of lessons was produced aimed at teaching the use of the Flash tool to qualified school teachers and to a set of trainee teachers on a PGCE course. The lessons took the learners through the stages of building an artefact that they were expected to see as useful for their intended teaching. The progressive development of the artefact focused on providing a set of increasingly complex animations aimed at helping children in early years' education to obtain basic literacy and numeracy skills. The teachers and trainee teachers were taken individually through a set of their own lessons to build the artefact. During this process faulty or non-computational thinking approaches adopted by the teachers/trainees were identified. Any incorrect approaches were then corrected by the trainer and the success or otherwise of the strategy taken in attempting to correct them was examined.

4 EXPERIMENTAL METHOD

In order to investigate the difficulties observed in the Malaysian trainee teachers and the role of our proposed tool in teaching CT, two groups of Malaysian test subjects were recruited for a pilot study. The first was a group of experienced teachers. The second was a similar set to those in the original Malaysian trainee classes, except that they were studying on a PGCE course in the UK. Both groups had no background experience in the use of software development.

The study was subject to the ethics approval process. All participants were informed of the nature and purpose of the study before partaking and were aware of the recording methods that were to be used. They all signed consent forms for their data to be gathered and used in the study.

Prior to using our pilot tool each subject was interviewed initially about their attitude to using the Flash environment and other aspects of using technology, particularly the learning of programming tools including the use of Scratch. The purpose of this was to identify attitudes already held by the subjects and enable us to identify in post-session interviews whether their attitudes had changed as it was expected that encounter with the actual process would improve their attitude. They then individually took part in a staged set of training lessons using the tool. The lessons were presented one-on-one but otherwise in a way similar to the original group (lecture, PowerPoint presentation and subsequent practical exercise). Each subject's comments and progress were recorded during their performance. The data captured included length of time to complete actions, evidence of enthusiasm for the task and number and types of errors made. After the lesson the subjects were once again interviewed about their attitudes towards use of Flash tools. The recordings were transcribed for analysis. A set of thematic codes was developed to use in determining the level of motivation and engagement with the activity. The codes reflected the following positive attitudes:

- I - Interest
- E - Engagement
- M - Motivation
- C - Confidence
- En - Enjoyment
- A - Affordance

And the following negatives:

- B - Boredom
- L - Lack of Motivation
- Co - Confusion
- F - Fear
- D - Difficulty

The transcriptions were then coded to identify statements and actions that indicated successful and failed understanding of the task and enthusiasm for the process (such as their attitude towards the Flash tool). Examples of positive and negative statements for each of the codes are given in Table 3 in Section 5.4 below. The time taken to develop a level of engagement was also recorded as a separate measure of participant engagement/motivation.

These lessons used in the pilot study are described in more detail in the next section.

3.1 Format of the Lessons

In the pilot study each participant worked through the sequence of lessons which aimed to start the development of the teaching/learning asset. As the lessons progressed the participants attempted to produce the asset. Their performance was monitored in order to determine causes of any failure to produce the expected outcome. These were then classified to determine if they were due to a lack of application of computational thinking. An example of non-computational thinking was the previously described use of a copying and pasting approach rather than understanding the development algorithm and following the algorithm for a new step in the process. As the stages of the lesson developed so did the complexity of the scripting needed to produce the correct performance.

In the lesson being followed an artefact was developed to create a teaching/learning aid for Elementary Level learning in Literacy and Numeracy. The lesson content is a subset of foundation level lessons in reading and number concepts. The artefact being produced was designed to meet expectations of early years learners for an attractive interface with colourful illustration and animations designed to gain attention. In the various stages the test subjects were given lessons in producing elements for the artefact that were successively more difficult to produce. The first step was to produce a simple menu – see Figure 1. This task was also broken down into stages, such as producing the title text, and then adding simple graphic elements, such as the button shapes. The next stage was producing the action for the buttons. The process of breaking tasks down into a set of simpler stages is a fundamental part of CT so in addition to the learning of the development tool users also got the opportunity to see the value of the CT approach in solving tasks.

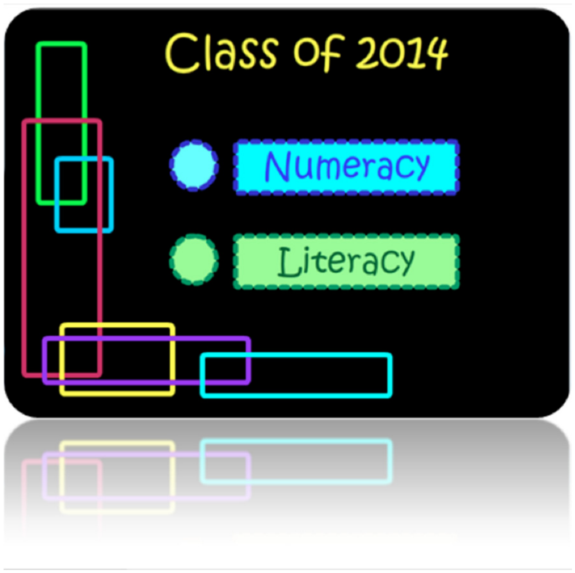

Figure 1: Original page.

This first stage of the lesson leads the learner through the process of producing the first screen to the point of producing an action to display the next screen. So, for example, a second screen is designed to appear when the Literacy button is pressed. This will display a view as shown in Figure 2 below.

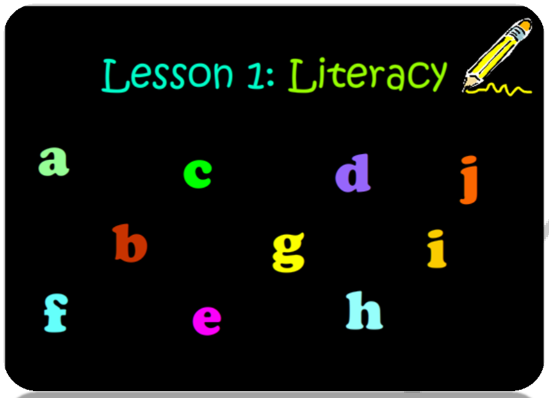

Figure 2: Literacy screen.

Each of the letters in the screen shown in Figure 2 will be turned into a button that leads to an action. The learner has to add the required components and thus practice the process from the earlier learning activity. (The letters and heading are all originally added as text.) The symbol requires the adding of a picture and the action script required is like that of the original button but the appropriate action had to be added to the letters which themselves had first to be converted into "buttons". While the next stage can be more or less derived from the previous the level of "scripting" required increases as the action now becomes more complex leading to the need to create an animated scene after an intermediate button press (see Figure 3). However, the repeated process has a lot of commonality so it should be possible for the participants to begin to understand the value of generalisation (one of the CT skills) and apply this by adding function calls to the Action Scripts to allow common code to be reused. This would not be done in early exercises as the needed instruction for using such functions is not given early in the set of tasks. Figure 3 also shows an additional feature that is taught at this point. The green path shown in the figure is added as the movement route (Tweening in Flash Action Script terminology) for the apple to follow during the animation that will be prescribed in the script.

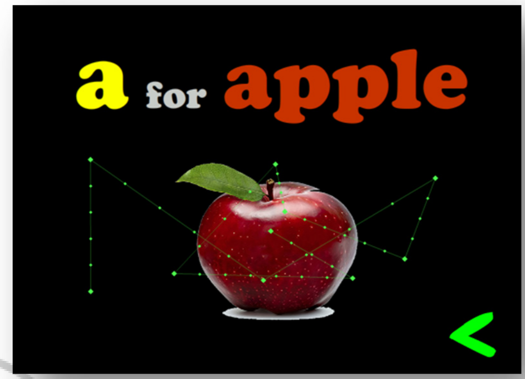

Figure 3: Scene with added animation path.

The next section details the results that were obtained from the pilot study and we then move on to draw our conclusions and suggestions for the next stage of this work in the final section.

5 RESULTS

The experimental subjects were all volunteers and were not incentivised to take part in the study. They fell into two groups. All the subjects were Malaysians. The subjects included a group of practicing teachers and a group of trainee teachers engaged in pursuing a PGCE qualification. There were significant differences between the qualified and trainee teachers both before and after the sessions and these differences will be explored in future publications.

5.1 Initial Interviews

As mentioned above a set of pre interviews were undertaken in order to allow for comparison between the pre and post session attitudes to programming and the use of Flash Action Scripts in particular. Certain differences also emerged between the qualified and trainee teachers in this session. Of the established teachers the responses to questions regarding the likely usefulness of learning Action Scripting were negative and all were nervous of the need to use a technical tool. A typical response was "*I am not good in using technology especially programming. I am afraid of using technology actually. I don't like to use computers …*"

The PGCE students were less reticent to take part. They all felt it would be good to learn to use

Action Scripts as it would enable them to use the tool to make their own teaching/learning instruments. However, they generally did not feel it would be easy to learn as it appeared to involve a similar process to learning more formal programming systems like C++. A typical quote from this group was "*I want to learn this programming language and I will come out with my own teaching module soon!*" There were some negative attitudes, for example one subject asked "*Why would I need to learn Flash or have to design a learning tool as all of my lesson material will be provided by the school?*"

5.2 Performance in the Practical Sessions

During the practical sessions the volunteers were required to produce codes that performed the set of functions shown in Figures 1 to 3. The performance was monitored and the actions were recorded using screen capture so they could be analysed afterwards. The major problem was the tendency to copy and paste previous examples rather than to analyse the problem, understand the components that were needed and develop the appropriate set of steps to produce the artefact. When they found their actions had not produced the correct behaviour they were asked to analyse why this had happened. They were taken back to previous steps and the lesson was repeated until they were able to produce the correct output. Different processes in producing a good learning artefact were identified against the Computational Thinking approach. A good example is the idea of abstraction as embodied in the concept of a layer. In Flash Action Scripting the layer is a basic abstract component that gets non-abstract components – actual text, pictures, and scripts – attached to it. As a result a user thinking computationally would be expected first to engage with the abstraction – by invoking a new layer for a new action – then to populate it. Thinking non-computationally exhibited itself in various different ways. One typical example was when users wanted to remove an existing layer. Instead of selecting the layer and deleting it the users who were not thinking about the abstraction would delete the contents of the layer rather than the layer. This would lead to an error when the code was compiled and an error message that was not understood by the user. Such error statements alert the teacher with experience to the lack of correct thinking. The user is then directed to the part of the lesson on layer creation.

The PGCE students all grew in confidence and enthusiasm for the process of learning Flash Action Scripting more quickly than the experienced teachers, however, all participants were able to finish the first task correctly. Enthusiasm for and motivation to use the tool were indicated by comments made by the experimental subjects. The trainee teachers generally were more enthusiastic to start with and thus were more motivated to understand and correct their mistakes.

The average time to complete the initial task was 35 minutes. The longest time taken for the process was one hour and this was from a subject who had shown the lowest motivation for the task as he intended to return to administration within a school and not teaching.

5.3 Post-session Interviews

After the first practical session the subjects were all re-interviewed. They were asked questions about their attitude to learning Flash Action Scripting having now been exposed to it. Questions centred on their views regarding the potential usefulness of learning to use Flash Action Scripting, technology in general, their motivation and ability to learn Flash Action Scripting and their understanding of the steps used in computational thinking.

Most of the experienced teachers felt an improved attitude towards the usefulness of the process. The teacher who intended to return as an administrator had not improved his attitude at all. One of the others still did not see that it could be useful to them in their teaching or to teaching in general. The other five were enthusiastic about the possibilities. Of the PGCE students all were convinced not only of the value of the tool but also of their ability to learn to use it.

When asked about the use of technology in general to support their teaching the experienced teachers had generally gained an improved attitude.

Most of the experienced teachers did not feel intrinsically motivated to learn programming but had found the process of the lesson enjoyable and felt they had been successful in learning how to use the tool. Eight of the ten trainee teachers expressed the idea that they found the learning intrinsically motivating as they felt they would benefit in their own careers. These have all moved on to the further study of the subsequent lessons and have been willing to try to learn further by themselves!

The CT skills needed in the task to perform it efficiently were those of Algorithmic thinking and Abstraction. Eight of the trainees exhibited

Algorithmic thinking by the end of the first lesson and seven had demonstrated use of Abstraction. However, only four of the experienced teachers had developed this CT approach. For example in the use of layers – linked to the concept of abstraction – seven of the trainee teachers and one of the experienced teachers had understood how to properly use layers.

5.4 Thematic Analysis

As was explained in Section 4 above the recordings of the interviews during the practical session were all transcribed and the transcriptions were then thematically coded looking for different indicators explained in section 4 relating to positive and negative attitudes. Table 1 below shows themes that emerged prior to the experiment taking place.

Table 1: Frequency of appearance of code phrases in use by the Trainee Teachers (TT) and the Experienced Teachers (ET) **before** taking part in the experiment.

Codes	**TT Frequency**	**ET Freq.**	**Total**
Interest (I)	23	17	50
Engagement (E)	24	15	39
Motivation (M)	27	15	42
Confidence (C)	8	0	8
Enjoyment (En)	12	11	23
Affordance (A)	2	3	5
Boredom (B)	3	11	14
Lack of Motivation (L)	6	15	21
Confusion (Co)	27	32	59
Fear (F)	15	12	27
Difficulty (D)	32	26	58

As can be seen from the results in Table 1 the trainee teachers demonstrated higher levels of positive attitudes to the potential value of the training they were about to undergo, when compared with the experienced teachers (indicated by a total of 96 coded positive statements by the trainee teacher group and only 61 by the experienced teachers). However both groups had negative attitudes in some of their statements and in particular felt the process of learning Flash Action Scripting would be difficult (roughly equal numbers of negative comments on difficulty) and that they found what they were being asked to do potentially confusing (again with similar numbers of negative comments from both groups). None of the experienced teachers expressed any confidence that they would develop the ability to use the tool but eight comments from the trainee teachers expressed such confidence.

Table 2: Frequency of appearance of code phrases in use by the Trainee Teachers (TT) and the Experienced Teachers (ET) **after** taking part in the experiment.

Codes	**TT Frequency**	**ET Freq.**	**Total**
I	39	19	58
E	29	24	53
M	54	27	81
C	22	20	42
En	22	19	41
A	12	8	20
B	0	2	2
L	0	2	2
Co	13	10	23
F	0	0	0
D	7	3	10

Table 2 demonstrates the frequency of code words in the post-experiment interviews and it can be seen that both the trainee and experienced teachers had significant improvements in their attitudes towards learning Flash Action Scripts and programming technology in general. Of particular note is the fact that no phrases emphasising "Fear" (F) were recorded in the post-experiment responses by either group though some "Confusion" (Co) still remained in both groups (a total of 23 such comments roughly equally divided between the groups) and there was still a residual view that the process was "Difficult" (D) though this was much reduced with a total of ten comments expressing difficulty after the experiment compared to the 58 difficulty comments made before the experiment. Only one of the participants (the experienced teacher now moving into administration) used "Boredom" (B) and "Lack of Motivation" (L) phrases – two responses fitting this description for both categories. Together with the increased number of positive comments on motivation (a total of 81 such comments after the experiment compared to 42 before the experiment) this indicates that those who were expecting to be teachers after their courses all believed the use of Flash Action Scripting to be interesting and motivating. There were few phrases mentioning "Confidence" (C) before the experiment- 8 from the trainee teachers and none from the experienced teachers; after the experiment only one of those involved (the administrator) failed to respond with any confidence statement. A total of 42 such comments were recorded from the rest of the subjects.

The time taken to begin to show signs of engagement and enjoyment of the use of Flash was generally lower for the trainee teachers than for the experienced teachers but only one (the administrator) failed to complete the task and failed to become engaged in the process after 20 minutes.

Examples of quotes reflecting the codes used are listed in the Table 3 below. Some of the phrases quoted show more than one of the coded attitudes. They were coded as representing both attitudes but are only shown as illustrating one. All of the comments listed were given by trainee teachers (respondent TT5 means trainee teacher 5) though similar comments were made by the experienced teachers.

Table 3: examples of coded comments given by the Trainee Teachers (TT4 to TT10).

Code	Quote
I	"As a first time learner like me, this software is easy to learn and the instruction is quite simple..." (respondent TT5)
E	"Very good! I think students will feel more enthusiastic to learn the subject if the teaching instrument used is interesting" (respondent TT4)
M	"I think it is attractive software to learn and teach also. Besides teachers learn how to program, I think we can teach our students to learn how to program as well." (respondent TT10)
C	"I am looking forward to studying more on this and producing my own lesson.." (respondent TT9)
En	"I knew that Flash can do interesting tasks like interesting courseware to gain students' attention." (respondent TT6)
A	"Can you give me a few minutes to try on my own on this software? I would like to produce a small interactive lesson about our lesson just now. I am just so excited to do animation and navigation buttons." (respondent TT4)
B	It is too hard for me to start learn the new software and need to struggle about the technical terms...." (respondent TT5)
L	"... there is just a waste of time to learn new software, I prefer to get the teaching tools that will be provided..." (respondent TT5)
C	make me feel confuse about technical terms and interface too..." (respondent TT5)
F	" I am afraid I could not produce the module at all because I am not good handling the technical tasks.." (respondent TT5)
D	" ..sometimes it very hard to understand especially the complex syntax..." (respondent TT7)

Our results demonstrate that use of the tool increased interest in Flash Action Scripting and enhanced enthusiasm for programming with reduced levels of confusion and frustration. The next section provides some conclusions from this study and proposes the next stage of our work.

6 CONCLUSIONS

The need for elementary school teachers to engage with learning to program and to think computationally has been identified. Pressure in the UK and elsewhere has led to the development of programming as a required part of the curriculum in both elementary and secondary schools. However, elementary school teachers are for the most part ill-equipped to learn to program themselves. It is vital that they also learn to understand the Computational Thinking (CT) process. However, they find that the programming languages they encounter are difficult to learn and lack the ability to motivate them to learn. They are at the same time wary of technology and this also makes them nervous of their potential to learn to program. The current research contends that learning the use of Flash Action Scripting would be viewed as more motivating as a first approach to learning to program for elementary teachers. This is in agreement with the findings of Lin (2012) which sees the need to motivate teachers to become creators of courseware rather than just consumers. Thus the current research aims at producing a software tool for helping teachers to learn to use Flash Action Scripting. It is acknowledged that Flash is not without its issues in terms of compatibility with some systems and security issues. Ultimately the ideas expressed in this paper could be applied to another programming environment such as JavaScript, although this isn't typically used in a visual programming environment. However, Flash has provided an appropriate environment for our study and upon which we could build our idea. In order to test this idea out and to find out how such a tool might be designed a pilot study has been undertaken. In this study lessons have been developed and presented to subjects in a more or less conventional manner through one-on-one tuition. The purpose of this was to observe the learners and ascertain where they were developing "non-computational" approaches to completing their tasks. The pilot study consisted of a total of seventeen participants consisting of established teachers and PGCE students. The attitudes of the PGCE students towards the idea of learning Flash Action Scripting were generally more positive than those of the established teachers. For example after coding of the interviews that took place before use of the pilot eight comments expressing motivation were found from the trainee teachers while none were given by the established teachers. Both groups were then given one-on-one training in the use of Flash Action Scripting. While the trainee teachers with their higher level of motivation were quicker to learn and had established good computational thinking processes by the end of the study compared with the established teachers, the latter group (of experienced teachers) were still generally found to have improved attitudes after the exercise. In fact only one of the latter group (one who was intending a future in administration rather than active teaching) failed to overcome boredom and a lack of interest in the scripting process. The results from our pilot study indicate that the use of Flash Action Scripting as a means to develop teaching lessons motivated and engaged both experienced and trainee teachers to learn to program and to develop in their Computational Thinking skills. For example both groups gave increased numbers of responses indicating motivation to learn the Flash Action Scripting after the experiment – the trainee teachers giving 54 positive responses after the study compared to 8 before and the established teachers giving 27 positive comments after the study compared to none before.

In addition valuable information was gained through the experiment into how poor CT skills interfere with learning and how good CT skills can be nurtured. Examples of this were the use of the Flash Action Script concept of layers. All users had not encountered this idea initially and did not initially understand how this concept worked. This invariably led to errors in performance. A particular example was given in section 5.2 where the contents of a layer were deleted rather than the layer itself. This led to a subsequent error statement from the compiler. This error statement was then used to help direct the learners back to the lesson component on layers. However, in the current pilot only eight of the seventeen users were able to complete such a task on their own after a second experience of this tutorial component. It is expected that the tool developed will use these lessons to provide a training tool that will work in a standalone context and provide encouragement to use CT as well as teaching the use of Flash Action Scripting. Error statements from the compiler can be parsed to get inputs to direct the user to the correct part of the lesson to tackle the error they encountered.

The next stage of the work is to develop the automated set of lessons in which the performance of the learners will be monitored by the system and the lesson adapted to their needs as perceived by the system. These lessons will be built in a learning instrument that will take the trainees through a set of steps to build a teaching/learning artefact. Currently the users are expected to be trainee elementary teachers or other teacher trainees from non-computer science/engineering backgrounds and, we surmise, therefore likely to be motivated to learn to produce a teaching/learning asset for early years' education.

REFERENCES

Begosso, L. C., Begosso, L. R., Gonçalves, E. M. & Gonçalves, J. R. 2012. An approach for teaching algorithms and computer programming using Greenfoot and Python. In *2012 Frontiers in Education Conference Proceedings (pp. 1-6). IEEE.*

Bogdanchikov B., Zhaparov M. and Suliyev R., 2013. Python to learn programming. In *Journal of Physics: Conference Series 423.*

Crick, T., & Sentance, S. 2011. Computing at school: stimulating computing education in the UK. *International Conference on Computing Education*, 122–123. Retrieved from: http://dl.acm.org/citation.cfm?id=2094158.

Dorland, M. 2014. Developing creativity and computational thinking in your computing classroom. *CAS meeting Birmingham UK Sept 2014.*

Franklin, D., Conrad, P., Boe, B., Nilsen, K., Hill, C., Len, M., Dreschler, G., and Aldana, G. 2013, Assessment of computer science learning in a scratch-based outreach program. In *Proceedings of the 44th SIGCSE technical symposium on Computer science education, SIGCSE '13. ACM, 2013.*

Grover, S., and Pea, R. 2013. Computational Thinking in K-12: A Review of the State of the Field. In *Educational Researcher, 42(1).*

Hammond, M. 2013. Introducing ICT in schools in England: Rationale and consequences. *British Journal of Educational Technology*,

Lin, C. H. 2012. An Innovative Change in Technology Integration: Training Pre-Service Kindergarten Teachers to Be Courseware Designers. *Creative Education, 03(07), 1177–1183.*

Meerbaum-Salant, O., Armoni, M., and Ben-Ari, M. M. 2010 Learning computer science concepts with scratch. In *ICER '10: Workshop on Computing education research, pages 69–76.*

Tondeur, J., Van Braak, J., Sang, J., Voogt, J., Fisser, P., & Ottenbreit-Leftwich, A., 2012 Preparing pre service teachers to integrate technology in education: A synthesis of qualitative evidence. In *Computers & Education 59(1).*

Wieling, M. B., Hofman, W. H. A. 2010 The impact of on-line video lecture recordings and automated feedback on student performance. In *Computers and Education, 54(4).*

Williams, P., 2008. Independent Review Report of Mathematics Teaching in Early Years Settings and Primary Schools, *dera.ioe.ac.uk/8365/1/Williams%20Mathematics.pdf [accessed October 21, 2014]*

Wing, J. M., 2006. Computational Thinking CS @ CMU and Grand Vision for the Field.

Wing, J. M., 2008. Computational thinking and thinking about computing. In *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences, 366(1881), 3717–25. doi:10.1098/rsta. 2008.0118.*

Zhang, D., Zhou, L., Briggs, R. O., & Nunamaker, J.F., 2006, Instructional Video in e-learning: Assessing the impact of interactive video on learning effectiveness. In *Information and Management, 43 (1).*