Bricolage Programming and Problem Solving Ability in Young Children : an Exploratory Study

ROSE, Simon

Bricolage Programming and Problem Solving Ability in Young Children: An Exploratory Study

Simon P. Rose
Sheffield Hallam University, Sheffield, UK
srose@steelminions.com

Abstract: Visual programming environments, such as Scratch, are increasingly being used by schools to teach problem solving and computational thinking skills. However, academic research is divided on the effect that visual programming has on problem solving in a computational context. This paper focuses on the role of bricolage programming in this debate; a bottom-up programming approach that arises when using block-style programming interfaces. Bricolage programming was a term originally used to describe the constructionist benefits of novice programming environments, yet more recent research has suggested it may promote poor programming practice that can negatively affect student ability and motivation. This paper describes an exploratory research study into bricolage programming aimed at exploring this concept in more depth.

The study used a post-test only experimental design to explore the effects of bricolage programming on problem solving when playing an educational programming game. Two versions of the game were created, one that used a Scratch-like visual programming interface to encourage bricolage, and one that used a more structured visual programming interface. A pre-test based on non-verbal reasoning was used to perform a matched assignment of forty, 6 and 7 year olds to the two conditions. Each child then played their version of the game for thirty minutes. It was hypothesised that children in the Scratch-like condition would insert, move and delete more programming instructions, in line with a bricolage approach. This in turn could result in differentiation in performance between conditions.

The results of this study showed that more indications of bricolage did occur in the Scratch-like condition. However, a range of measures of overall performance revealed no difference between the two groups. Post-hoc analysis of the data suggested that indications of bricolage may vary according to the relative progress made by participants in the game. Findings and opportunities for future work are discussed.

Keywords: Bricolage Programming, Problem Solving, Computational Thinking, Lightbot, Constructionism, Scratch

1. Introduction

The widespread use of visual programming is well documented (e.g. Maloney et al. 2010, Simpkins 2014) and Scratch is arguably one of the most commonly used visual programming environments. It aims to teach children aged 10 to 19 programming basics by constructing programs using instruction blocks (Resnick et al. 2009). This can be used to create media, ranging from short animations to fairly complex games. Since Scratch was released in 2006, several other visual programming environments have been developed that utilise a similar instruction block-based approach. Visual programming environments are derived from earlier work on constructionism and the LOGO programming language (Papert 1980). Papert's learning theory of constructionism, in which construction is used as a goal, is a key tenet behind the design of many visual programming environments.

The introduction of computing into educational curricula has brought with it a requirement for children to learn basic programming concepts, and to develop computational thinking and problem solving skills from as early as five years old. Visual programming environments are seen as a means of achieving these objectives, which has led to an increase in their use in an educational context. This has been justified by research into the link between visual programming and computational thinking (Brennan and Resnick 2012). However, other research, such as that by Meerbaum-Salant, Armoni and Ben-Ari (2011) raises questions over the widespread use of visual programming to teach problem solving skills.

The majority of research in this area focuses on children aged between 8 and 18. There is a lack of research into the effects of visual programming on younger children, below eight years old. This is of particular importance due to the growing exposure of visual programming environments to children of this age group as part of the school curriculum.

This paper will first explain bricolage theory, then review existing research into visual programming and its effect on problem solving ability in children. This will be followed by an explanation of the investigation undertaken, and justification of the game-based approach which was used. The experimental software is

described and hypotheses are outlined along with the methodology used during the study. Finally, the results of the study are discussed and conclusions made.

2. Bricolage programming and problem solving

Bricolage theory originates from Strauss (1962), who used it to contrast the analytic methodology of Western science with the "science of concrete" in primitive societies. The term is derived from the French verb "bricoler" ("to tinker"), and can be used to describe the process of creating work that begins with an endless range of possibilities. It is used by Turkle and Papert to describe two approaches to problem solving; the first being an analytical style where problems are approached from the top down or "planned", the second being a bottom-up or "bricolage" approach, where problems are attempted "by arranging and rearranging, by negotiating and renegotiating with a set of well-known materials" (1990, p136).

It is widely recognised that problem solving is an important skill for children to develop. Mayer and Wittrock (2006) state that problem solving consists of understanding, representing, planning and executing. Problem solving ability can be developed in many different ways, however research suggests that visual programming can have a significant impact on how children learn to solve problems, particularly in a computational context (e.g. Akcaoglu 2014). Furthermore, Meerbaum-Salant, Armoni and Ben-Ari (2013) and Sáez-López, Román-González and Vázquez-Cano (2016) indicate that Scratch can be used to teach computer science principles that underpin computational thinking. It is suggested that for visual programming environments to be successful in education, they require a constructionism-based low-floor high-ceiling strategy to solving problems (Flannery, et al. 2013). This is due to the variability of cognitive skills amongst younger children. As such, visual languages should cater for the lowest-ability child whilst providing suitable room to develop understanding using more complex mechanics.

However, Kalelioglu and Gülbahar (2014) suggest that visual programming environments do not have a significant impact on problem solving ability in children. Furthermore, Meerbaum-Salant, Armoni and Ben-Ari (2011) identify that bricolage and a bottom-up approach to programming arise when children create programs using visual programming languages. They suggest that this encourages extremely fine-grained programming (EFGP), where children would decompose programs into extremely small blocks of instructions that lack logical coherency. EFGP can make Scratch programs difficult to run and debug (as all blocks are executed concurrently) and has a negative impact on student motivation. This principle of program re-design with "incompletely understood meaning" (Beynon and Roe 2004, p217) can be seen as in conflict with traditional top-down programming approaches. The identification of bricolage programming and EFPG raises questions over the suitability of using Scratch-like visual programming environments to teach children programming principles.

3. A game-based approach

The field of digital game-based learning now spans five decades (e.g. Cullingford, Mawdesley and Davies 1979). It is stated that educational computer games should have clear but challenging goals (Malone 1980). By implementing a goal-based structure (which is often difficult to achieve in constructionism-based visual programming environments) younger children should be more engaged in the cognitive and emotional involvement of gameplay (Jabbar and Felicia 2015). It has also been suggested that the contextualisation of learning can increase motivation (Cordova and Lepper 1996). One game which incorporates both these elements is Lightbot (Lightbot Inc. 2008). Lightbot is an educational game with the objective of programming a small robot to light up all the lights in levels comprised of blocks (figure 1). This is achieved by arranging a fixed set of commands in a finite program space. As the levels progress, more complex programming principles such as procedures and conditionals are introduced. Gouws, Bradshaw and Wentworth (2013) state that Lightbot is useful for developing computational thinking, particularly in conceptual areas.

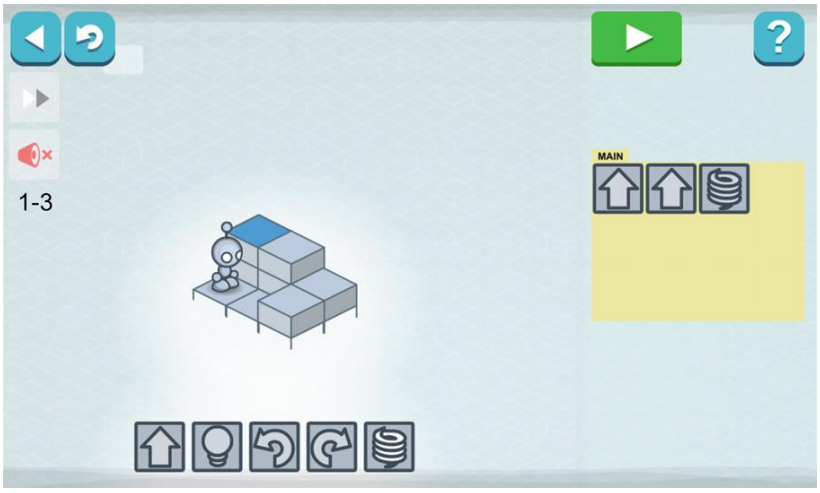

Figure 1: A simple level from Lightbot

There is an apparent contrast between the bricolage-based approach of Scratch and the more structured programming style used in Lightbot. The main difference being that in Scratch, a limitless number of instructions can be placed in the program space, which will not be executed unless explicitly linked to an "execute on start" instruction or similar. Alternatively, in Lightbot, all instructions in the main program will be executed in sequence when the play button is pressed. Meerbaum-Salant, Armoni and Ben-Ari (2011) observed that children using Scratch would add all the instructions to the program space that they thought may help solve the problem, and only then attempt to fit them together. This bottom-up approach to problem solving is not facilitated by Lightbot's programming interface. It was this central difference that provided the basis for this study. Researching the effect that Scratch-like instructions had on Lightbot gameplay, focusing on testing bricolage and problem solving ability.

3.1 The experimental software

A Lightbot-style game was created with two versions; one using a similar instruction and program mechanic to Lightbot, and one using Scratch-like instructions that can be added to the program space but will not execute unless chained to the "start" block (figure 2). Scratch-like instructions were implemented using Google Blockly, an open-source library used to create visual programming editors. Due to the age of the children being studied, the Lightbot game mechanics were simplified considerably. Up to 9 or 10 years of age, children often struggle with spatial transformations such as left or right rotation (Huttenlocher and Presson 1979). The view was altered from 2.5D to top down 2D to help the player anticipate where the robot would finish after each instruction was executed. The game contains 15 levels, introducing more complex movements and multiple lights in a difficulty progression designed to challenge the more-able participants.

A number of formative evaluations of the game were undertaken before the study took place, with the aim of iteratively refining the functionality of the application during development. The participants came from the year above the study participants at the same school (7 and 8 years old). This was done to avoid exposing possible participants of the formal investigation to the game in its primitive stages. The teacher of the class was asked to select low to mid-range ability students, therefore aiming to more closely match the ability level of the study participants (aged 6 and 7).

Aspects of the game such as instruction appearance, position of the user interface and participant understanding of the game were all trialled at this stage. Some participants misunderstood the purpose of the instructions signifying left and right rotation, mistaking them for left and right lateral movement. Text-based instructions were also trialled, however participants had difficulty understanding these. Therefore, the final rotation instructions were represented by symbolic arrows which were rounded (as they are in Lightbot) to better represent rotation. There was also confusion as to which way the robot was facing, so an arrow was added above the robot to clarify its current direction. Answers to repeated questions from participants with regards to gameplay were added to the tutorial video. Overall, the evaluation was effective in establishing

limitations of the age group and receiving feedback on the simplified game mechanics, and had a considerable impact on the final version the game.

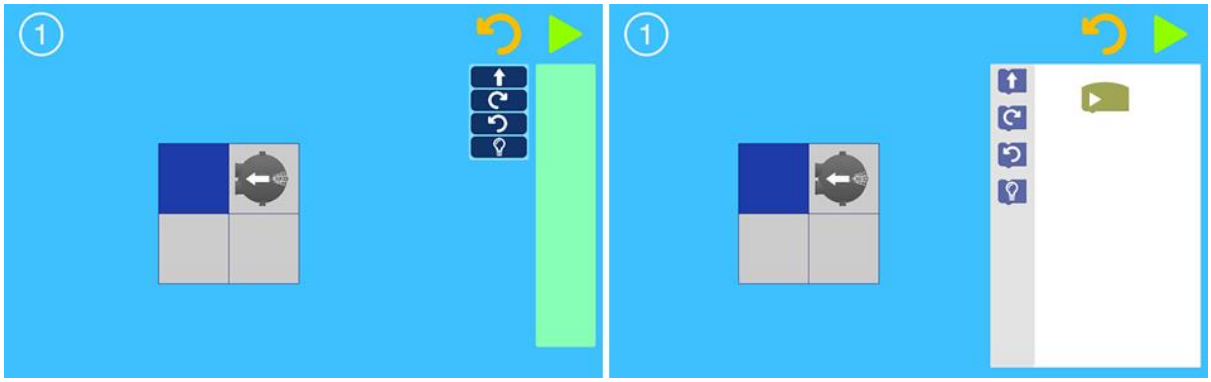

Figure 2: The Lightbot (left) and Scratch-like version (right) of the game

3.2 Hypotheses

Based on existing literature, it was hypothesised that:

a) In line with the observations made by Meerbaum-Salant, Armoni and Ben-Ari (2011), the version of the software which supported Scratch-like instructions would lead to more bricolage programming.
b) In line with the constructionist literature (e.g. Harel and Papert 1991), the version of the software which supported Scratch-like instructions would improve outcomes on problem solving tasks.
c) The free design approach offered by Scratch-like instructions would be more beneficial to higher-ability players. This comes from the theory that lower-ability children are more suited to a structured learning environment, whereas higher-ability children thrive on constructing their own learning (Flannery, et al. 2013).

Primary measures were outlined to test these hypotheses:

- The pretest scores for each participant.
- A measure of bricolage programming (instruction insertions, moves and deletions per level).
- The average amount of attempts taken per level by a participant, indicating problem solving ability.
- The maximum level reached by each participant, indicating problem solving ability.

The pre-test scores would give each participant a measure of non-verbal reasoning ability; the ability to solve problems using visual reasoning (which is required in the game). Bricolage programming was measured by the amount of instruction insertions, moves and deletions from the program. Increased modification of the program would show that the participant was performing more "tinkering" and "negotiating and renegotiating", in line with bricolage theory. Using the average amount of attempts taken per level to indicate problem solving ability was based on the theory that by taking fewer attempts to complete a level, participants would be better using problem solving techniques (understanding, representing, planning and executing) and relying less on trial and error. The maximum level reached by a participant shows how well they understood the game, with a higher level indicating a better ability to solve problems.

4. Participants

All participants were from a large primary school in a low-income area in northern England. The vast majority of the pupils are of White British heritage. The school has a well above average proportion of disadvantaged pupils and pupils that require special educational need. Forty-one boys and forty-one girls took part in the pre-test, from which the study participants were randomly selected. The forty study participants included 20 boys and 20 girls between the age of 6 years, 3 months and 7 years, 3 months (*M* = 6 years, 9 months).

5. Method

Eighty-two participants were tested using a non-verbal reasoning ability pre-test, with questions adapted from standardised school worksheets (an example question is shown in figure 3). This gave each participant a quantifiable indication of computational thinking ability, which it was hoped would transfer to the game. The

pre-test took place in the IT suite of the school in groups of fifteen participants. The pre-test contained forty questions, and the participants had five minutes to answer as many as they could. It was reinforced verbally that there was no rush to answer the questions, and that answers should be carefully thought through. It was hoped that a time-limit would help produce a greater range of pre-test scores, because not all children would answer all forty questions. Using the results of the pre-test, two groups of twenty participants were created using matched, randomised assignment (Habgood 2015) to ensure that the mean pre-test scores of both groups were the same. This group generation process also produced substitute participants, which were used when several participants were absent.

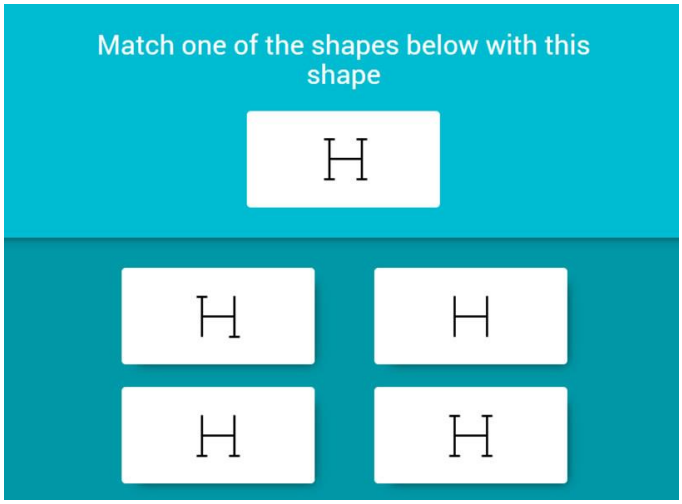

Figure 3: Pre-test application to test non-verbal reasoning ability

A quasi-experimental post-test only design was used to collect quantitative data. Each child played one version of the game for thirty minutes. Data was collected by logging player interaction; including time spent on each level, level attempts (by recording button presses), program state and instruction insertions, moves and deletions from the program (a measure of bricolage). The study took place in a small reading room joined to the children's classroom. Two laptops were set up, facing away from each other, so that one child from each group could play the game without being aware that they were using a different version to their classmate. The different conditions were tested together so that if any difficulties arose, results from both groups would be affected. Program settings, staff and time were kept as consistent as possible throughout the study. Non-verbal communication can have a significant effect on interaction behaviour (Poyatos 1977). Therefore, caution was taken when conducting the study that non-verbal signals were minimised between researchers and participants. This was primarily achieved using a tutorial video to give participants a uniform introduction to the game, instead of a verbal explanation that may have been subject to changes in vocal tone and body language.

6. Results

Forty children took part in the post-test, twenty in each group. The average pre-test scores of both groups was 24.85 (SD = 5.96). This statistic was achieved using a matched, randomised assignment process which sorts "participants based on a matching variable (often pre-test scores) and then randomly assigns similarly scoring participants between treatment groups" (Habgood 2015, p222). This can be done by hand, but for this study a computer program was written that continuously generated groups until the mean pre-test scores of both groups were matched to two decimal places. This process was done to ensure that any differences observed between groups can be reliably attributed to the independent variable.

This matched design is based upon the assumption that the pre-test variable is a reliable measure of a participant's potential to succeed in the game. This was supported in the final data by a correlation between the pretest scores and maximum level reached by each participant, $r(40) = .73$, $p < .01$.

6.1 Hypothesis testing

Evidence of bricolage programming would be shown by a significant difference between both groups in the amount of bricolage performed. A one-way analysis of variance (ANOVA) was performed for the average amount of bricolage per level, with two levels of the between-subjects factor, 'group' (Lightbot and Scratch-like). This showed that there was a significant main effect for the 'average bricolage per level', $F(1, 38) = 8.46$, $p < .01$. The mean values of each group can be seen in table 1.

	Lightbot (n = 20)	*Scratch-like (n = 20)*
Average bricolage per level	35.45 (SD = 21.34)	56.58 (SD = 24.5)

Table 1: The mean average bricolage per level for each condition

It was also hypothesised that Scratch-like instructions would increase problem solving ability. The average number of level attempts per completed level would need to be significantly lower for participants using Scratch-like instructions. This was done by comparing the average number of level attempts between groups. There was no significant difference found.

The third hypothesis was that Scratch-like instructions would be more beneficial to higher-ability players, and would be supported by a difference in the maximum levels reached of higher-ability participants using the Scratch-like environment. Higher-ability participants were defined by achieving a higher pretest score than the overall mean. When divided in this way, the difference between the average maximum levels was not significant.

7. Discussion

The significant difference in bricolage programming between-groups supports the theory that bricolage occurs when using Scratch-like instructions. This is not a surprising finding given the additional freedom afforded by Scratch's visual programming environment, which encourages more a, moves and deletions of instructions. However, in confirming this basic assumption, it also highlights the significance of the debate over the effects of bricolage programming on problem solving ability (e.g. Akcaoglu 2014, Kalelioglu and Gülbahar 2014). It is therefore important to better understand how bricolage is affecting problem solving ability and represents a key opportunity for future research.

The link between problem solving ability and Scratch-like instructions was based on the theory that more experimentation would be put in to creating programs using Scratch-like instructions, ultimately leading to improved problem solving ability. The lack of evidence to support this may be linked to the validity of using average attempts taken to complete a level to indicate problem solving ability. Participants who played the game using a trial and error technique may have completed levels in fewer attempts simply by chance. Furthermore, there may not have been enough difference in the opportunities for experimentation between the two conditions. The main difference between the versions was the ability to add a limitless amount of instructions to the Scratch-like program, which would not be executed unless chained to the start block. However, it was observed that participants often did not use this feature, instead choosing to remove unused blocks from the program entirely (similar to the Lightbot approach). The amount of unused instructions in the program could be used as a primary measure in a future study.

The final hypothesis addressed the educational theory that high-ability learners benefit more from a free design approach. The mechanics of testing this hypothesis required the dataset to be narrowed down considerably (halving the number in each condition). This reduction in sample size reduces the statistical power of the results, and therefore the ability to properly explore this question.

7.1 Post hoc analysis

This study was primarily undertaken as a preliminary, explorative study into this field and so the data was further analysed to identify any trends that could potentially be explored for further research. This focused on the statistics for each individual level (as opposed to each individual participant):

- The amount of participants who completed a level.
- Average bricolage (insertions, moves and deletions) per attempt on a level.

- Average attempts taken per level.
- Average time taken to complete a level.

Levels were also ranked by difficulty:

- Levels requiring only forward instructions, no rotation (levels 1 and 2).
- Levels requiring a single rotation (3, 4 and 5).
- Levels requiring multiple rotations of the same direction (6, 7, 8 and 9).
- Levels requiring multiple rotations of multiple directions (10 and 11).
- Levels requiring multiple lights (12, 13, 14 and 15).

Interestingly, when viewed level by level, there was an increase in the average amount of bricolage used per attempt between groups as the difficulty progressed. Figure 4 shows the opposite effect that each environment had on the amount of insertions, moves and deletions of instructions as the difficulty level increased. This may suggest that participants' understanding of the Scratch-like environment increased as they progressed through the game.

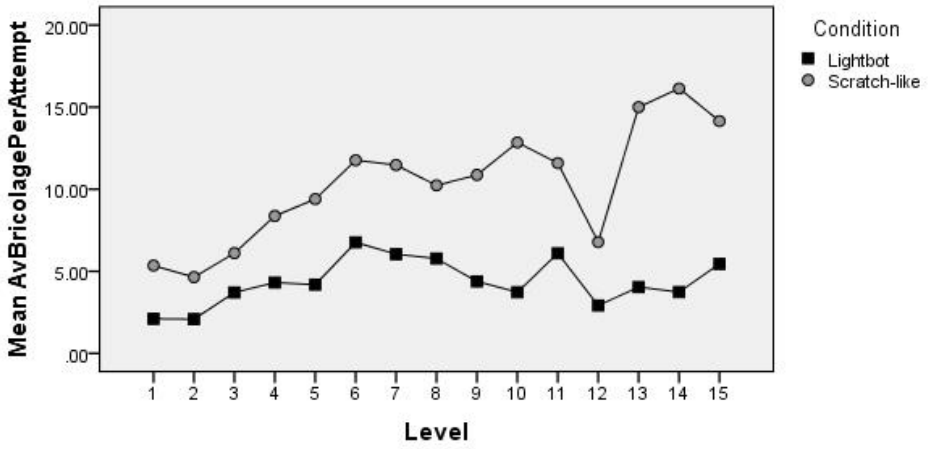

Figure 4: Line graph showing the average amount of bricolage used per attempt as the difficulty increased

As mentioned previously, an overall difference in average bricolage per level was observed between the two groups. The graph shown in figure 4 raised the question as to whether the difference is more pronounced for participants who experienced success with the game. The eighth level was identified as the point in the game which split the sample size evenly, with half the children progressing beyond that level. A one-way ANOVA was performed for the 'average bricolage per attempt' using just the participants in the study who failed to complete the eighth level. This showed a significant main effect of 'average bricolage per attempt' between conditions, $F(1, 18) = 9.84$, $p < .01$. A second one-way ANOVA showed that there was also a significant main effect of 'average bricolage per attempt' for participants who got further than the eighth level, $F(1, 18) = 35.137$, $p < .001$. While between group differences were significant in both cases, the mean difference for participants who got further than the eighth level was over double that of participants who didn't get as far as the end of the eighth level, as can be seen in tables 2 and 3.

	Lightbot (n = 9)	*Scratch-like (n = 11)*
Average bricolage per attempt	4.47 (1.6)	7.06 (2.01)

Table 2: The mean average bricolage per attempt for participants who failed to complete the eighth level

	Lightbot (n = 11)	*Scratch-like (n = 9)*
Average bricolage per attempt	4.85 (2.17)	11.51 (2.86)

Table 3: The mean average bricolage per attempt for participants who completed the eighth level

It could be suggested that the low achieving participants displayed a smaller difference in bricolage because they did not fully understand the fundamental concepts of the game. This implies that visual programming environments need to be inclusive for less-able children, supporting the low-floor high-ceiling approach suggested by Flannery, et al. (2013). Moreover, that attention should be paid to bricolage in problem solving tasks that challenge younger children. This is of particular importance due to the growing exposure of visual programming environments to children of this age group as part of educational curricula.

8. Conclusion

This study has provided evidence to confirm the existence of a bricolage approach to young children's programming in Scratch-like environments. This in turn confirms the importance of further research to investigate the effects of visual programming on problem solving ability, in order to confirm their suitability to teach computational thinking to younger children.

The post-test only design used for this exploratory study has clear limitations in terms of attributing any between-group differences to the intervention rather than random differences between groups. Matched assignment does go some way to addressing this, but the matching variable would ideally have been the same as the dependant variable of the study, rather than a separate measure. Future studies will attempt to use a pre to post-test design based on a common measure of computational thinking in order to address this, but developing an instrument which reliably measures computational thinking is a non-trivial task (Jenson and Droumeva 2016). It is also worth noting that the background of the majority of participants in this study was low-income White British heritage, which reduces the generalisability of the results. Further studies could take place in schools with a more diverse range of pupil backgrounds.

This research has also revealed an interesting relationship between the difficulty of tasks and indications of a bricolage approach. This may suggest that future studies should focus on the effect of bricolage in problem solving tasks that present a real challenge for the child. It could also be that there is a differential effect of bricolage on familiar and unfamiliar problem solving tasks. Future research will be undertaken with the aim of exploring such questions.

References

Akcaoglu, Mete (2014) "Learning Problem- Solving through Making Games at the Game Design and Learning Summer Program", *Educational Technology Research and Development,* Vol 62, No. 5, pp 583-600.

Beynon, Meurig and Roe, Chris (2004) "Computer support for constructionism in context", In: *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies,* IEEE, pp 216-220.

Brennan, Karen and Resnick, Mitchel (2012) "New frameworks for studying and assessing the development of computational thinking", In: *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*.

Cordova, Diana I. and Lepper, Mark R. (1996) "Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice.", *Journal of Educational Psychology,* Vol 88, No. 4, pp 715.

Cullingford, G., Mawdesley, MJ and Davies, P. (1979) "Some experiences with computer based games in civil engineering teaching", *Computers & Education,* Vol 3, No. 3, pp 159-164.

Flannery, Louise P., Silverman, Brian, Kazakoff, Elizabeth R., Bers, Marina U., Bontá, Paula and Resnick, Mitchel (2013) "Designing ScratchJr: Support for early childhood learning through computer programming", In: *Proceedings of the 12th International Conference on Interaction Design and Children,* ACM, pp 1-10.

Gouws, Lindsey A., Bradshaw, Karen and Wentworth, Peter (2013) "Computational thinking in educational activities: an evaluation of the educational game light-bot", In: *Proceedings of the 18th ACM conference on Innovation and technology in computer science education,* ACM, pp 10-15.

Habgood, Jacob (2015) "Zombie Division: a methodological case study for the evaluation of game-based learning", In: *ECGBL2015-9th European Conference on Games Based Learning: ECGBL2015,* Academic Conferences and publishing limited, pp 219.

Harel, Idit Ed and Papert, Seymour Ed (1991). *Constructionism.* Ablex Publishing.

Huttenlocher, Janellen and Presson, Clark C. (1979) "The coding and transformation of spatial information", *Cognitive psychology,* Vol 11, No. 3, pp 375-394.

Jabbar, Azita Iliya Abdul and Felicia, Patrick (2015) "Gameplay Engagement and Learning in Game-Based Learning A Systematic Review", *Review of Educational Research,* Vol 85, No. 4, pp 740-779.

Jenson, Jennifer and Droumeva, Milena (2016) "Exploring Media Literacy and Computational Thinking: A Game Maker Curriculum Study", *Electronic Journal of e-Learning,* Vol 14, No. 2.

Kalelioglu, Filiz and Gülbahar, Yasemin (2014) "The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective", [online]. *Informatics in Education,* Vol 13, No. 1, pp 33.

Lightbot Inc. (2008). *Lightbot* [computer program].

Malone, Thomas W. (1980) "What makes things fun to learn? Heuristics for designing instructional computer games", In: *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems,* ACM, pp 162-169.

Maloney, John, Resnick, Mitchel, Rusk, Natalie, Silverman, Brian and Eastmond, Evelyn (2010) "The Scratch Programming Language and Environment", [online]. *ACM Transactions on Computing Education (TOCE),* Vol 10, No. 4, pp 1-15.

Mayer, Richard E. and Wittrock, Merlin C. (2006). Problem solving. In: Vol 2, 287-303.

Meerbaum-Salant, Orni, Armoni, Michal and Ben-Ari, Mordechai (2011) "Habits of programming in scratch", In: *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education,* ACM, pp 168-172.

Meerbaum-Salant, Orni, Armoni, Michal and Ben-Ari, Mordechai (2013) "Learning computer science concepts with Scratch", *Computer Science Education,* Vol 23, No. 3, pp 239-264.

Papert, Seymour (1980). *Mindstorms: Children, computers, and powerful ideas.* Basic Books, Inc.

Poyatos, Fernando (1977) "Forms and functions of nonverbal communication in the novel: A new perspective of the author-character-reader relationship", *Semiotica,* Vol 21, No. 3-4, pp 295-338.

Resnick, Mitchel, Maloney, John, Monroy-Hernández, Andrés, Rusk, Natalie, Eastmond, Evelyn, Brennan, Karen, Millner, Amon, Rosenbaum, Eric, Silver, Jay and Silverman, Brian (2009) "Scratch: programming for all", *Communications of the ACM,* Vol 52, No. 11, pp 60-67.

Sáez-López, José-Manuel, Román-González, Marcos and Vázquez-Cano, Esteban (2016) "Visual programming languages integrated across the curriculum in elementary school: A two year case study using “Scratch” in five schools", *Computers & Education,* Vol 97, pp 129-141.

Simpkins, N. (2014) "I Scratch and Sense But Can I Program?: An Investigation of Learning with a Block Based Programming Language", *International Journal of Information and Communication Technology Education (IJICTE),* Vol 10, No. 3, pp 87-116.

Strauss, Claude Lévi (1962). *Savage mind.* University of Chicago.

Turkle, Sherry and Papert, Seymour (1990) "Epistemological pluralism: Styles and voices within the computer culture", *Signs,* Vol 16, No. 1, pp 128-157.