Supporting the learning of the Pythagorean theorem using Turtle Environments

Andrew Butcher

Abstract

This paper seeks to explore how Turtle-based programming environments (including, but not limited to LOGO) can be used to enable students to develop their understanding of the Pythagorean theorem and some of the associated concepts. It considers how traditional methods for teaching these concepts fall short of providing students with a sufficiently scaffolded and authentic learning opportunity, and looks at the ways in which these shortcomings can be addressed by using turtle-based programming environments as artefacts to support an instrumental approach to learning. Finally, it considers how students' learning could be impeded by the use of technology, and the ways in which software can be more carefully designed to meet students' and teachers' needs.

Traditional Teaching Methods

Students studying GCSE mathematics are required to be able to calculate the length of the hypotenuse of a right-angled triangle using the Pythagorean theorem, as well as calculating the length of one missing shorter side. The Pearson specification requires students to "apply angle facts, triangle congruence, similarity and properties of quadrilaterals to conjecture and derive results about angles and sides, including Pythagoras' theorem" (Pearson, 2014, p. 9) and they should "know the formulae for: Pythagoras' theorem and the trigonometric ratios" (Pearson, 2014, p. 10). The OCR specification makes similar stipulations, but differentiates skills between students expecting to take the foundation and higher papers; students studying to take the foundation paper should "know, derive and apply Pythagoras' theorem to find lengths in right-angled triangles in 2D figures" (OCR, 2020, p. 39) whereas students studying for the higher paper are also required to "apply Pythagoras' theorem in more complex figures, including 3D figures" (OCR, 2020, p39). With many schools preparing students to sit their exams from the start of Year 9, this is a topic which students will encounter several times during their secondary schooling.

Approaches to teaching the Pythagorean theorem are fairly standardised, and Chambers (1999) outlines four subtly different techniques. The first approach, he explains, is a statement which requires no diagram; "The square on the hypotenuse is equal to the sum of the squares on the other two sides" (Chambers, 1999, p. 22) although he argues that this approach "uses language that is not easily accessible to many 14-year-olds" (Chambers, 1999, p. 22). The other three approaches make use of the following diagrams:

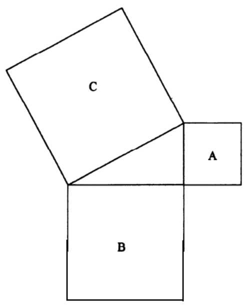


Figure 1

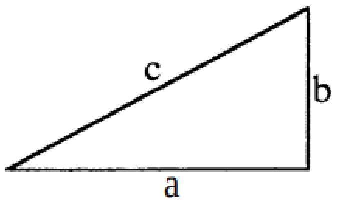


Figure 2

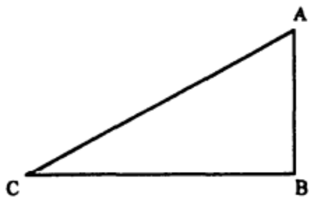


Figure 3

Figure 1 is accompanied by the statement C = A + B (Chambers, 1999, p. 22). Figure 2, which Chambers (1999, p. 22) points out is the most common form of labelling for right angled triangles, invites three different ways of stating the theorem; "(i) $c = \sqrt{(a^2 + b^2)}$ (ii) $c^2 = a^2 + b^2$ (iii) $a^2 + b^2 = c^2$" (Chambers, 1999, p. 22). The Pearson exam board advocates for this approach, and the formula students following the specification are required to know is "$a^2 + b^2 = c^2$" (Pearson, 2014, p. 10). The final approach, which uses Figure 3, labels the corners rather than the sides. Chambers (1999, p. 22) points out that even though this approach is falling out of fashion, "it is useful when dealing with more complex diagrams, but not necessary when dealing with a simple triangle".

Of these approaches, the method based on Figure 2 is the most widely applied, almost certainly because this is the approach most widely supported by exam specifications. Questions in textbooks and on worksheets reflect the popularity of this approach:

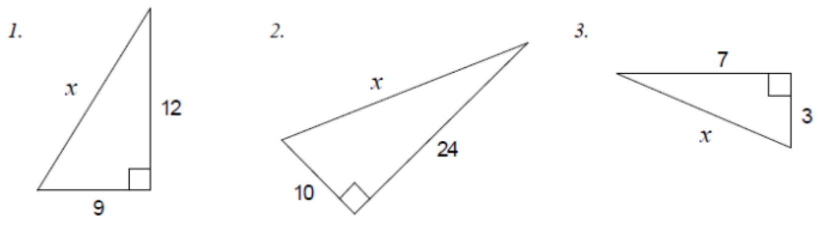


Figure 4: Taken from http://templatelab.com/pythagorean-theorem-worksheet/

Arwani (2011, p. 36) affirms, “as mathematics teachers we usually teach Pythagoras theorem by giving two side lengths so that the length of the third side can be calculated using the formula”. Arwani (2011, p. 36) suggests that “it would be beneficial for students as they would be able to see applications in daily life, such as finding the diagonal of a baseball ground, TV screen and so on”. Although the true value of such applications is questionable, this seems to be a popular notion. One worksheet presents students with the following question:

“A ladder is leaning against the side of a 10m house. If the base of the ladder is 3m away from the house, how tall is the ladder? Draw a diagram and show all working”

While this question is no doubt rooted in a bid to authenticate students’ learning, it does not reflect a genuine real-life application of the theorem; most adults have ever used this approach to calculate the length of a ladder. Studies have sought to add authenticity to the teaching of the Pythagorean theorem, including Spyrou et al (2009, p. 534) who “created a comprehensive learning environment that settles the Pythagorean Theorem within the students’ experiences, thus making it meaningful to them”. While the specifics of this study aren’t particularly relevant to the scope of this paper, it does serve to illustrate the drive to bring purpose to a topic historically devoid of authenticity.

Another widely cited challenge facing students studying the Pythagorean theorem is the abstract nature of the algebraic elements of the topic. Adhitama at al (2018, p. 1) explain that “the difficulty of students in learning the Pythagorean theorem was in determining the hypotenuse, specifically in the algebra operation” and that “in mathematics there are many abstract elements of algebra, especially for Junior High School students it is hard to understand abstract things. According to Piaget's learning theory, students in junior level (ages 12-15) had not yet fully been able to understand abstract material”.

Goldin et al (2017, p3) offer a more detailed interpretation of the algebraic prerequisite understanding, and draw a distinction between “procedural performance” and “knowledge and understanding of concepts”. Procedural performance “refers to the knowledge of operators and symbolic representations of mathematics and the use of algorithms and rules to reach certain goals” (Goldin et al, 2017, p3) and “If students do not have sufficient procedural fluency, they may experience difficulties in solving geometry problems and approving exams, which regularly test procedural performance" (Goldin et al, 2017, p3). This perspective presents mathematics teachers with an interesting problem, and raises the question of whether the Pythagorean theorem is a geometry topic or an algebra topic; arguably the approach taken by most teachers means that it rather more represents the latter.

So far, the evidence seems to suggest that there are two significant barriers to students' understanding of the Pythagorean theorem:

1. The topic draws on two separate areas of mathematics; geometry and algebra, and students who find the latter of these two topics difficult are likely to struggle to solve problems involving the former.

2. Students find it difficult to find meaning in their learning as a result of the apparent lack of authenticity.

Applications of Turtle-based Programming Environments

Pardamean et al (2017, p. 53) advocate for the use of LOGO to support the teaching of geometry topics, concluding that "understanding of geometric shapes is enhanced, leading to more sophisticated mathematical problem-solving abilities". The authors explain that "Logo Programming was originally developed at the Massachusetts Institute of Technology in 1967 by Seymour Papert et al., with the intention to allow people, even young children, to use computers as a learning tool". Since then, several academics have written in support of the platform as a tool to support the learning of geometry (Ernest, 1988; Rahim, 1997; Sheehan, 2000; Jimenez-Molotla et al, 2007; Jimenez-Molotla et al, 2009; Jimenez-Molotla et al, 2010; Pardamean, 2015). Further specific potentialities shared include helping children to learn "the concepts and skills of mathematics" (Ernest, 1988, p. 16), "making it easy to turn sequences of commands into procedures" (Sheehan, 2000) and, reflecting on the impact on students, "there is an increase in their problem-solving abilities" (Jimenez-Molotla et al, 2007, p. 5).

Jimenez-Molotla et al (2007; 2009; 2010) have engaged in extensive research into the use of turtle-based programming environments like LOGO to support the learning of the Pythagorean

Theorem, alongside other associated topics like Trigonometry. They began by creating “an ICT-based project for the learning of a topic which is traditionally difficult to teach and learn, and more so at the junior secondary level (children aged 12 to 15 yrs-old): trigonometry” (Jimenez-Molotla et al, 2007, p. 2), and while their initial research focussed more on trigonometric functions than on the Pythagorean theorem, it acted as a springboard to their subsequent studies which focussed much more on the latter whereby they “developed a long-term trigonometry project called “Painless Trigonometry” for introducing young students to the Pythagorean theorem” (Jimenez-Molotla et al, 2009, p. 93). Reflecting upon the impact of the “Painless Trigonometry” project, the authors attest that “in written tests after the project, the students showed an understanding of the “advanced” trigonometry concepts, as well as of other algebraic ideas” (Jimenez-Molotla et al, 2009, p. 93). The study focuses on using the LOGO programming language, alongside other tools like Excel and Cabri-Géomètre to develop a learning sequence which allowed students to investigate how to construct a 3D model of a Pyramid. They explain that students “discovered that in LOGO they could use Pythagoras Theorem to draw the inner diagonals” (Jimenez-Molotla et al, 2009, p. 96) and present the following code as an example of a student outcome:

```
1  to righttriangle
2  forward 100 back 100
3  right 90
4  forward 100
5  left 135
6  forward sqrt((100 * 100) + (100 * 100))
7  end
```

There are some curious design choices in this block of code, which should be noted. Firstly, on line 2 - 4, the student constructs the two shorter sides of the triangle first by backtracking from the end of the first line, back to the right angle before turning 90 degrees to construct the second line. This means that the student only needs to calculate one angle, on line 5. Secondly, the resulting triangle is isosceles. This means that they know that the angle on line 5 must be 135 degrees, rather than needing to calculate the angle using trigonometric functions. This was important, as the authors explain that they needed “to find a curricular topic for junior secondary grades 1 and 2”

(Jimenez-Molotla et al, 2009, p. 94). The researchers observed that “students who are shy and withdrawn in other classes and environments began to express brilliant and clever ideas” (Jimenez-Molotla et al, 2009, p. 99) and point out that “students [were] able to perceive mathematics as something meaningful, something that they can apply for solving a project; instead of something boring, meaningless and forced upon them” (Jimenez-Molotla et al, 2009, p. 100). This final point seems especially poignant, considering that the lack of meaning and lack of authentic application were cited as common barriers to learning of the Pythagorean theorem.

A year later, the researchers presented a final iteration of the project which allowed students to use Pythagorean ideas (alongside Trigonometry) to construct a model of the Eiffel Tower. The ambitious project drew upon a range of mathematical skills; “they need to develop an understanding of basic geometry (angles, triangles, polygons, circles, etc), they need to be able to solve arithmetic and algebraic problems, and, of course, they need to work in three dimensions”. Considering the Pythagorean concepts which needed to be considered, students were asked to “trace the diagonals across two faces of a cube to split the cube in half”, “Build the base of a pyramid and its height”, and “Finish the pyramid using right triangles with two equal sides and 45° angles” (Jimenez-Molotla et al, 2010, p. 5). The authors finish with a profound conclusion; “Logo is a philosophy; and just like mathematics is not just about numbers and is a way of thinking, Logo is also a way of thinking” (Jimenez-Molotla et al, 2010, p. 11).

Rahim (1997, p. 137) presents an alternative application for turtle-based programming, and presents “a Turtle Geometry activity [...] for the secondary school level. The contents of the procedures of the activity include trigonometry and advanced plane geometry that are part of high school mathematics” (Rahim, 1997, p. 137). The author explains that learners' success was “organized through a special set of Logo procedures” (Rahim, 1997, p. 131) which allowed students to construct more complex triangles without needing to calculate all of the angles. It meant that the activity could more easily be differentiated to allow students who were able to engage with more complex trigonometric functions to do so, while allowing other students more practise with Pythagorean problems. The appendix of Rahim (1997) includes several teacher-defined functions which allow students to achieve the learning outcomes. In a sense, this provides an alternative approach to using turtle-based programming to teach Pythagorean concepts, without being constrained to the isosceles example presented by Jimenez-Molotla et al (2009).

Looking beyond the central purpose of learning about the Pythagorean theorem, Ernest (1988, p. 18) echo's the notion that "algebra is traditionally one of the most difficult areas of mathematics because of its abstraction and formality", and argues that when students are using the LOGO programming language "the variable is embedded in a meaningful context, and is named in a way that relates to its meaning". The benefits of using programming environments to support students' understanding of algebra are well documented; one study examining students' use of a non-turtle based programming environment concluded that they had observed "students improve significantly on algebra word problems" (Schanzer et al, 2018) after following a course of study in programming. Arguing in favour of turtle-based programming environments, Ernest (1988, p. 18) contends that "Turtle Geometry may provide the quickest and least formal entry into programming situations in which the use of variables gives the user great power". Tramonti and Paneva-Marinova (2019, p. 84) agree, and argue that "Information and Communication Technology (ICT) can become artefacts used as meaningful learning tools, if they provide students with opportunities to learn with technologies and not from technologies".

Theoretical Perspective

Gueudet et al (2014, p. 140) explain that the instrumental approach "draws on activity theory, as introduced by Vygotsky (1978). It studies the activity of one or more subjects, involved in a goal directed activity [...] and mediated by artefacts". While not explicitly documented by Jimenez-Molotla et al (2009) and Rahim (1997), the programming environment is an artefact; "a product of human activity, purposely designed for a human activity oriented towards a given goal" (Gueudet et al, 2014, p. 140). According to Gueudet et al (2014, p. 140), "students work with an artefact [...] and develop an instrument from this artefact. The instrument is a mixed entity, encompassing (a part of) the artefact, and a scheme built along the use of the artefact".

Gueudet et al (2014, p. 140) explain that during instrumental genesis, subjects (in these cases, the students who are engaging with the learning) use the artefact (i.e. the programming environments) to produce an instrument. They point out that there are two "intricate aspects" of instrumental genesis; instrumentation and instrumentalisation. Instrumentation involves a subject developing knowledge whereby the "features of the artefact influence the subject's activity and the knowledge developed" whereas instrumentalisation occurs when a subject's prior knowledge about a topic shapes their interaction with an artefact. Gueudet et al (2014, p. 145) cite a study carried out by Buteau and Muller (2014) whose students wrote programs to investigate the number of raindrops that will hit a person at various rain, wind and running velocities. Students “produce a mathematical model of the situation then they design a program corresponding to this model, and use this program to explore the situation”. Gueudet et al (2014, p. 145) argue that this “back-and-forth movement fosters instrumental genesis, leading the students to learn both mathematics and programming”. While this study didn’t involve the use of Turtle, it did involve the use of a programming environment and so seems comparable to the studies presented by Jimenez-Molotla et al (2009) and Rahim (1997).

Applying this idea of instrumental genesis toJimenez-Molotla et al (2009) and Rahim (1997), when students “discovered that in LOGO they could use Pythagoras Theorem to draw the inner diagonals” (Jimenez-Molotla et al, 2009, p. 96) this was an example of instrumentalisation because the students’ prior knowledge about the Pythagorean theorem shaped their interaction with the artefact. However, when learners' success was “organized through a special set of Logo procedures” (Rahim, 1997, p. 131) instrumentation occurred because the features of the artefact (i.e. the procedures written by Rahim (1997) to facilitate students’ learning of the Pythagorean theorem) influenced the mathematical knowledge that the students developed.

While Gueudet et al (2014, p. 145) write so hopefully about the back-and-forth interplay between mathematics and programming, they warn “about students’ cognitive load when required to learn programming for their exploration of mathematics” and explain how Buteau and Muller (2014) “carefully limited the amount of programming language for each mathematical task in hand”. Therein, perhaps, lies the greatest potential limitation of turtle-based programming environments. Ungvarsky (2019) defined cognitive load as “the amount of information the active part of human memory can process at one time” and points out the significant impact of the split-attention effect, which “impacts learning when people have to look at multiple sources to gather all the information needed to solve a problem”.

Indeed, Ernest (1988, p. 18) concedes that “one major obstacle to children's easy use of LOGO variables should be mentioned: the horrible syntax involving quotes and colons”, citing the following example of LOGO code:

```
TO SPIRAL :S REPEAT 100 (FD :S RT 90 MAKE "S :S + 5) END
```

Hence, it is conceivable that if students are required to focus their efforts on such complex syntax, they may fail to focus on the desired mathematical outcomes. This concern is echoed by Qian and Lehman (2017, p. 17) who explain that “In introductory programming courses, students exhibit various misconceptions and other difficulties in syntactic knowledge, conceptual knowledge, and strategic knowledge”. Kelleher and Pausch (2003, pp. 103 - 107) suggest that this may be more of an issue with general-purpose languages which require users (in this case students) to “type their program into a text editor, compile the program, fix any syntax errors, build the program, and then run it” and the authors advocate in favour of “small languages that students can master more quickly than general-purpose languages”. Although LOGO was the original turtle-based programming environment, modules and libraries now exist which enable users to complete similar activities using languages including SmallBasic (http://smallbasic.com/doc/?id=20) and Python (https://docs.python .org/3/library/turtle.html). Both of these languages are popular in secondary school Computer Science classrooms, and arguably offer students a more user-friendly programming interface. Alternatively, Perks and Prestage (2002, p. 38) explains that “challenging students to draw a rhombus from scratch may be inappropriate if they do not know the properties of the shape. Students could be asked to use a given procedure e.g. ‘shape’ and to experiment with the two variables to find the angles to make the shape closed”. Such procedures were also seen in the study conducted by Rahim (1997), which allowed students to focus effectively on the most relevant and necessary mathematical content.

Conclusions

In conclusion, this paper has shown that traditional approaches to teaching students about the Pythagorean theorem usually fall short in one of two ways:

- Teachers may fail to recognise that without a solid grasp of algebraic content, students will be unable to effectively substitute values to find a solution, or rearrange a formula to change its focus. Given the emphasis placed by exam boards on knowing the formula for solving Pythagorean problems, students’ weaknesses in this area cannot be overlooked.

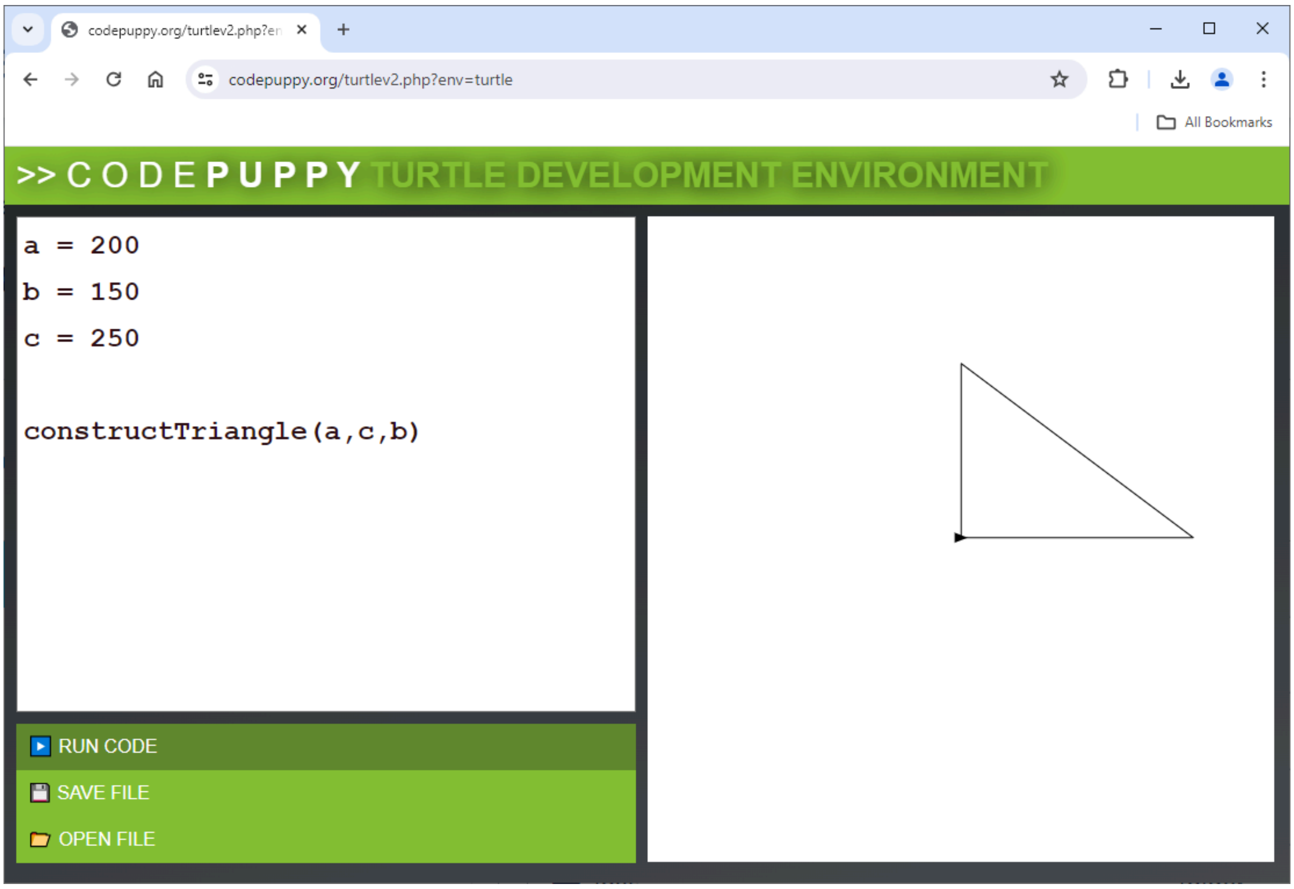
- Applied Pythagorean problems are rarely authentic, and rely heavily on contrived and unlikely scenarios. Without being able to see the purpose in their learning, students struggle to engage and so their learning is less effective.

An examination of an extended study carried out by Jiménez-Molotla et al (2007, 2009, 2010) and a smaller-scale study presented by Rahim (1997) demonstrate how turtle-based programming languages have been used to add authenticity to students' learning of the Pythagorean theorem, alongside other papers published by Ernest (1988) and Schanzer et al (2018) which demonstrate how turtle-based programming languages and programming languages in general can be used to support students' learning of algebraic content. Arguably the experience of programming supports the development of students' understanding (not the Turtle).

This paper has considered how the instrumental approach can be used to develop students' learning in mathematics and programming bilaterally, and has reflected upon how the studies completed by Jiménez-Molotla et al (2007, 2009, 2010) and Rahim (1997) have demonstrated instrumental genesis (albeit, perhaps unintentionally). Thought has been given to the ways in which the programming environment both shapes students' understanding of mathematical concepts and allows students to re-frame the mathematical understanding they brought with them. Reflecting on cognitive load and the challenges faced by students as they try to learn a new syntax, this paper leaves open the suggestion that purpose-built learning programming languages and carefully designed libraries of procedures may allow students to develop their mathematical understanding without diluting the programming capability which will allow them to further develop their appreciation of algebraic content and problem solving.

Development of a Learning Activity

Appendix A shows a scheme of work designed to use a turtle-based programming environment to support students' learning. The proposed environment, shown below and available at http://www.codepuppy.org, was developed by the author of this paper to help students overcome some of the difficulties students typically face when working in general-purpose programming environments:


codepuppy.org/turtlev2.php?env=turtle
All Bookmarks
>> CODEPUPPY TURTLE DEVELOPMENT ENVIRONMENT
RUN CODE
SAVE FILE
OPEN FILE

```
a = 200
b = 150
c = 250


constructTriangle(a,c,b)
```

In particular, the side-by-side configuration of the code window and program output is designed to address many of the issues highlighted by Kelleher and Pausch (2003, pp. 103 - 107) while reducing cognitive load.

The accompanying sequence of lessons is designed to allow students to use this programming environment as an artefact whereby the "features of the artefact influence the subject's activity and the knowledge developed" (Gueudet et al, 2014, p. 140). In particular, lessons 1, 5 and 6 encourage students to employ trial-and-improvement approaches to arrive at their own conclusions and develop their mathematical understanding through an authentic investigation. At the same time, the scheme of work is designed to promote students' programming ability through use of variables and functions; rather than being a standalone sequence of lessons this scheme of work is presented as part of a pedagogical approach which sees programming embedded within the mathematics curriculum and makes the assumption that students would have done some more basic work with Turtle already. The scheme of work draws on two specific approaches seen in the review of literature, in terms of allowing students to focus on the Pythagorean theorem without becoming unduly concerned with trigonometric functions. Lessons 1-4 introduce students to the concepts using an isosceles triangle; an approach employed by Jimenez-Molotla et al (2007, 2009, 2010), while lessons 5 and 6 use a built in function (written specifically for the purpose of this scheme of work, and included in Appendix B) to allow students to construct non-isosceles triangles; an approach employed by Rahim (1997). Arguably, it is important for students to construct non-isosceles triangles in order to generalise the concepts. Finally, the scheme of work provides opportunities for students to use variables to develop their algebraic understanding, with variable placeholders used in lesson 3, 4 and 6.

The scheme of work culminates in an open-ended investigation into Pythagorean Triples. This is designed to extend students' mathematical and computing understanding, and has been selected based on the capacity to solve the problem computationally. Here, students can choose to swap their trial and improvement approach for an iterative computational approach, whereby a, b, and c are each incremented automatically and tested to determine whether a right angled triangle is returned.

References

Adhitama, I., Sujadi, I., Pramudya, I. (2018) *Discover the pythagorean theorem using interactive multimedia learning,* IOP Conf. Series: Journal of Physics: Conf. Series 1008 (2018) 012066 doi :10.1088/1742-6596/1008/1/012066.

Arwani, S. S. (2011) 'The Unforgettable Experience of a Workshop on Pythagoras Theorem', *Mathematics Teaching*, (225), pp. 35–36.

Buteau, C., & Muller, E. (2014). Teaching roles in a technology intensive core undergraduate mathematics course. In A. Clark-Wilson, O. Robutti, & N. Sinclair (Eds.), *The mathematics teacher in the digital era* (pp. 163–185). New York, NY: Springer.

Chambers, P. (1999) 'Teaching Pythagoras' Theorem', *Mathematics in School*, 28(4), p. 22.

Ernest, P. (1988) 'What's the Use of LOGO?', *Mathematics in School*, Vol. 17, No. 1 (Jan., 1988), pp. 16-20.

Gueudet, G., Buteau, C., Mesa, V., & Misfeldt, M. (2014) 'Instrumental and documentational approaches: From technology use to documentation systems in university mathematics education', *Research in Mathematics Education*, 16(2), 139-155.

Goldin, A. P., Pedroncini, O. and Sigman, M. (2017) 'Producing or reproducing reasoning? Socratic dialog is very effective, but only for a few', *PLoS ONE*, 12(3), pp. 1–12.

Jiménez-Molotla, J., Gutiérrez-Gómez, A., Sacristán, A.I. (2007) Painless Trigonometry: A Tool complementary School Mathematics Project. In: Kalaš, I. (ed.) Proceedings 11th EuroLogo Conference: 40 Years of Influence on Education. Comenius University, Bratislava (2007), http://www.di.unito.it/~barbara /MicRobot/AttiEuroLogo2007/proceedings/P-Jimenez-Molotla.pdf.

Jiménez-Molotla, J.; Gutiérrez-Gómez, A., Sacristán, A.I. (2009). Pyramids in Logo: A School Project in 'Search' of the Fourth Dimension. In Education and Technology for a Better World, IFIP

Advances in Information and Communication Technology series. Edited by A. Tatnall & A. Jones. Springer, Boston. Pp. 92-101.

Jiménez-Molotla, J. and Sacristán, A. I. (2010) Eight years of journey with Logo leading to the Eiffel tower mathematical project, Constructionism 2010, Paris.

Kelleher, C. and Pausch, R. (2003) 'Lowering the Barriers to Programming: a survey of programming environments and languages for novice programmers', *ACM Computing Surveys*, Vol. 37, No. 2, June 2005, pp. 83–137.

OCR (2020) *GCSE (9-1) Specification: Mathematics.* Available at https://www.ocr.org.uk/Images /168982-specification-gcse-mathematics.pdf.

Pardamean, B., Suparyanto, T. and Evelyn (2015) 'Improving problem-solving skills through Logo programming language', *New Educational Review*, 41(3), pp. 52–64.

Pearson (2014) *GCSE (9-1) Mathematics*. Available at https://qualifications.pearson.com/content/ dam/pdf/GCSE/mathematics/2015/specification-and-sample-assesment/gcse-maths-2015-specificati on.pdf.

Perks, P. and Prestage, S. (2002) 'Does the Software Change the Maths? Part 2', *Micromath*, 18(2), pp. 37–41.

Qian, Y. and Lehman, J. (2017) 'Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review', *ACM Trans. Comput. Educ.* 18, 1, Article 1 (October 2017).

Rahim, M. H. (1997) 'Turtle Geometry at a Secondary School Level', *Computers in the Schools*, 14:1-2, 129-142.

Schanzer, E., Fisler, K., Krishnamurthi, S. (2018) 'Assessing Bootstrap:Algebra Students on Scaffolded and Unscaffolded Word Problems', SIGCSE '18, February 21–24, 2018, Baltimore, MD, USA.

Sheehan, R. (2000) ‘Lower floor, lower ceiling: easily programming turtle-graphics’, Proceeding 2000 IEEE International Symposium on Visual Languages, Visual Languages, 2000. Proceedings. 2000 IEEE International Symposium on, pp. 87–88.

Spyrou, P., Moutsios-Rentzos, A., Triantafyllou, D. (2009) ‘Teaching for the objectification of the Pythagorean Theorem’, Conference: 10th International Conference of the MEC21 Project.

Template Lab (2020) *48 Pythagorean Theorem Worksheet with Answers*, Available at http://templatelab.com/pythagorean-theorem-worksheet/.

Tramonti, M. and Paneva-Marinova, D. I. (2019) ‘Maths, Art and Technology: a Combination for an Effective Study’, *TEM Journal*, 8(1), p. 82.

Trouche, L. (2018) ‘Instrumentalization in mathematics education’, *Encyclopedia of Mathematics Education*. NY: Springer.

Ungvarsky, J. (2019) ‘Cognitive load’, *Salem Press Encyclopedia*.

Appendix A: Proposed Learning Sequence

Objective	**Learning Activity**	**Example Outcome**

1. Estimate the length of the hypotenuse of an isosceles right angled triangle	*Activity*
	Use Turtle to construct a right-angled triangle where the two shorter sides each have a length of 10
	
	Extension
	Use Turtle to construct a right-angled triangle where the longest side has a length of 20 and the two shorter sides are equal
		`forward(10);`
		`left(135);`
		`forward(14);`
		`left(135);`
		`forward(10);`
		`hide();`

2. Accurately calculate the length of the hypotenuse in an isosceles right angled triangle	*Activity*
	Use the following code to calculate the length of the longest side
	`Math.sqrt(sum(10 * 10, 10 * 10))`
	
	Extension
	Construct a range of isosceles right angled triangles by changing the lengths of the shortest sides and the calculation to work out the length of the hypotenuse.
		`forward(10);`
		`left(135);`
		`forward(Math.sqrt(sum(10 * 10, 10 * 10)));`
		`left(135);`
		`forward(10);`
		`hide();`

3. Understand how variables can be used to store values	*Activity*
	The following program takes three variables (a, b, and c) and uses them to construct a triangle:
	`a = 10;`
	`b = 10;`
	`c = 14;`
	`forward(a);`
	`left(135);`
	`forward(c);`
	`left(135);`
	`forward(b);`
	`left(90);`
	Enter the code into the editor, and modify it so that the value of c is calculated dynamically
	
	Extension
	Modify the program so that the value of c is declared, and the values of the other two lengths are calculated dynamically
		`a = 10;`
		`b = 10;`
		`c = Math.sqrt(sum(a * a, b * b));`
		
		`forward(a);`
		`left(135);`
		`forward(c);`
		`left(135);`
		`forward(b);`
		`left(90);`

4. Understand how functions can reduce the need for repeated computation	*Activity*
	Complete the following function:
	`function pyth(a,b) {`
	
	`}`
	
	`pyth(10,10);`
	
	Extension
	Test your function using a range of input values. Document your testing process logically.
		`function pyth(a,b) {`
		`    c = Math.sqrt(sum(a * a,`
		`        b * b));`
		`    forward(a);`
		`    left(135);`
		`    forward(c);`
		`    left(135);`
		`    forward(b);`
		`    left(90);`
		`}`
		
		`pyth(10,10);`

5. Use trial and error to construct a triangle with side lengths 8, 10 and 6	*Activity*
	Copy the following program into the code editor:
	`forward(8);`
	`left(0);`
	`forward(10);`
	`left(0);`
	`forward(6);`
	`hide();`
	
	Change the values of the left() functions to construct a complete right angled triangle
	
	Extension
	Find an alternative combination of integer values which can be used to construct a right angled triangle.
		`forward(8);`
		`left(143);`
		`forward(10);`
		`left(127);`
		`forward(6);`
		`hide();`

6. Investigate Pythagorean triples	*Activity*
	Copy the following program into the code editor:
	`a = 8;`
	`b = 6;`
	`c = 10;`
	
	`constructTriangle(a,c,b);`
	Change the values of a, b and c to find combinations of inputs which yield a right angled triangle with three integer lengths.
	
	Extension
	Investigate Pythagorean Triples.
	- Can you find any patterns?
	- What is a primitive triple?
	- Can you write a program to automatically find all Pythagorean triples where the hypotenuse is less than 100?
	- What about where the hypotenuse is less than 1000?
	- Can you write a program which returns only the primitive triples?
	- How many primitive triples are there with a hypotenuse length less than 1000?
		`a = 8;`
		`b = 6;`
		`c = 10;`
		
		`constructTriangle(a,c,b);`

Appendix B: The constructTriangle(a,b,c) function

The following function was created to allow students to construct triangles by passing in the lengths of three sides, without having to concern themselves with the complexities of Trigonometric functions

```
function constructTriangle(a,b,c) {
    var A = 180 - Math.degrees(Math.acos((b*b + c*c - a*a)/(2*b*c)));
    var B = 180 - Math.degrees(Math.acos((c*c + a*a - b*b)/(2*a*c)));
    var C = 180 - Math.degrees(Math.acos((a*a + b*b - c*c)/(2*a*b)));
    forward(a);
    left(C);
    forward(b);
    left(A);
    forward(c);
    left(B);
}
```