Investigating the Relationship Between Spatial Skills and Computer Science

Jack Parkinson and Quintin Cutts

The relationship between spatial skills training and computer science learning is unclear. Reported experiments provide tantalising, though not convincing, evidence that training a programming student's spatial skills may accelerate the development of their programming skills. Given the well-documented challenge of learning to program, such acceleration would be welcomed. Despite the experimental results, no attempt has been made to develop a model of how a linkage between spatial skills and computer science ability might operate, hampering the development of a sound research programme to investigate the issue further. This paper surveys the literature on spatial skills and investigates the various underlying cognitive skills involved. It poses a theoretical model for the relationship between computer science ability and spatial skills, exploring ways in which the cognitive processes involved in each overlap, and hence may influence one another. An experiment shows that spatial skills typically increase as the level of academic achievement in computer science increases. Overall, this work provides a substantial foundation for, and encouragement to develop, a major research programme investigating precisely how spatial skills training influences computer science learning, and hence whether computer science education could be significantly improved.

Introduction

Skills in STEM subjects appear to be related to spatial skills (SS): STEM practitioners are reported to have high SS, relative to others [34]; training in SS can improve abilities in STEM subjects, particularly engineering [24]. There is tantalising evidence of such a relationship in computer science (CS), which, due to the cheap and easily accessible nature of SS training, could lead to higher achievement and lower dropout rates, as with engineering.

Unfortunately, current studies in this area are limited and inconclusive - correlation has been identified [12], but only one study [33] shows that SS training appears to help in computing. Based on this inviting start, further study is warranted.

SS are not easy to define strictly [30], and as such studies contain unclear and contradictory descriptions which are likely to hamper research efforts. Perhaps as a result of this, no studies postulate why the STEM/SS relationship exists to any great extent, and so current work may not be optimally focused. Furthermore, most studies in the field tend to concentrate on a single cohort, typically entry level CS students, without examining effects across experience levels.

Based on these gaps, we present three main additions to the research in the field. First, we summarise what is known about SS, defining core elements of SS and how they can be measured. Second, we propose a model for the relationship between SS and CS, drawing on key cognitive processes which appear to be shared by both fields.

Third, we describe an experiment to examine the relationship between SS and CS attainment across a range of CS practitioners, from entry level students to professors.

Related Work in SS and STEM

Spatial skills have been connected with STEM for almost seventy years, since Super and Bachrach examined the skills of mathematicians, engineers and scientists, and found SS to be a factor in all these fields [29]. In a broad study covering the work of dozens of researchers, Super and Bachrach attempted to classify the skills and traits of professionals in science and engineering, reviewing studies on such factors as mathematical ability, verbal ability and several other “special'” abilities, including SS. They found that not only are SS prominent in these fields, but that in cases where the relationship was tested, STEM practitioners outperformed non-STEM people in SS tests, even those recognised as being ``gifted'' in other fields.

Wai *et al.* undertook an investigation of SS pertaining to Project TALENT data [34, 35]. Project TALENT consisted of a series of tests given to over 400,000 high school students in the US in 1960 and subsequent follow up questionnaires up to the 1970s. Of the students who went on to achieve a PhD in a STEM field, most scored highly in the Project TALENT spatial skills tests taken eleven years previously (with 45\% being in the top 4\% of SS scores). Again, the relationship is not causational; SS are shown only to be correlated to progression in STEM subjects.

The STEM area with most research relating to SS is engineering. Sorby has investigated this relationship for over 20 years, showing that engineering students who receive SS training do better in their engineering courses and have lower dropout rates [24]. In addition to developing a SS training course [26], Sorby has shown positive effects of training SS initially on self-selecting groups of low SS scorers in engineering, and then a similar effect in compulsory courses provided by Michigan Tech~\cite{sorby2007developing}. The effect of these studies are significant and well replicated: one can reliably train SS to see an improvement in engineering success.

SS also have relationships with success in other STEM fields. In physics, Kozhevnikov *et al.* discovered that psychology undergraduates with better spatial visualisation skills performed better in, and could explain more clearly, kinematic physics problems [13]. Pallrand and Seeber conducted a separate examination in physics, identifying that not only did students taking a physics course show higher gains in SS compared with students taking liberal arts courses on pre/post tests, an experimental group undertaking additional SS training outperformed the placebo and control groups [16]. This study is like those undertaken by Sorby, showing the effectiveness of a training course which can be taken alongside standard teaching [24]. Crucially, it also shows that SS can be developed while studying a STEM subject, even with no explicit SS training, a point we will return to later.

Carter *et al.* showed that those with higher SS outperformed those with lower SS in a general chemistry course [3]. The same has also been found in organic chemistry, when manipulating and understanding 2D representations of molecular molecules [17].

Tartre identified that spatial orientation ability is applied in certain mathematical problems, and suggested that the ability was specifically related to particular mathematical skills, such as determining the area of irregular shapes and groupings of associated objects [30]. However, Tartre's chosen test for spatial orientation is more typically used as a test of closure speed [8], and one of the selected mathematical problems is very similar to an existing test of spatial relations (shown in figure~\ref{figure:3dc}). Another study indicating a connection between spatial visualisation and mathematics was conducted by Fennema and Sherman, who showed that spatial skills are a factor contributing to the gender gap found in mathematics [9].

In addition to these studies, Veurink and Sorby [33] have shown that the training course developed by Sorby and Baartmans [25] (and subsequently developed into a workbook [26]) can be used to potentially improve the results of engineering students undertaking non-engineering modules. Several cohorts of engineering students taking additional modules (in areas such as calculus, physics and chemistry) had their SS measured at the start of the course. Those who failed a SS test were offered a chance to increase their SS on the course, and ultimately those students who took up the offer did better in their respective elective modules than their peers who also failed the test but opted not to take the course [33].

In Veurink and Sorby's paper, another module in which students excelled after SS training was a computing module, specifically introductory programming. Students who initially failed the SS test and opted to take training showed significantly higher GPAs in their computing course than those who failed or marginally passed the test, but did not take additional training. This result is based on 6 cohorts, totaling 74 participants, of self selecting students between 1996 and 2002. This implies a causal relationship from SS to programming, though self efficacy may be a factor in these findings: students self-selected to take the additional training, and it is possible that the students who have a more proactive attitude were both likely to take the course when offered and excel in their elective modules anyway. Additionally, there was no prior measure of computing ability, which could be a confound in the study.

Though not making reference to the study by Veurink and Sorby, Cooper *et al.* attempted to show a similar result [5]. We are surprised that Cooper's study, of which Sorby is a co-author, does not reference this earlier, apparently highly-related, work. Cooper took a selection of summer school students intending to begin a university course in computing, and over a period of two weeks, trained their SS in an experimental group and compared their gains in a standardised computing test. The authors acknowledge some issues with the study, e.g. the questions used to test computing ability may not have been the most effective for the group of students they had. The increase in gains by the experimental group failed to reach significance, except when the six questions from the test with the highest item discrimination only were used in the analysis. Ultimately, the authors clearly state that they are not claiming causation, but a correlation which requires further research.

A similar correlation was displayed earlier by Jones and Burnett [12]. They took a cohort of Masters students who had not previously studied computing, tested their SS and examined their end of year results. They did not see any correlation in the Introduction to Human Factors or the IT Management courses taken by these students, but did see a correlation between the Introduction to Programming course and the Object Oriented Systems course, both of

which required a significant amount of programming. This suggests that it is possible that the connection with SS lies not strictly with computing generally, but specifically with programming.

Based on this existing research, we identify two points. First, evidence of a causal relationship between SS and CS is limited, though there is something of interest in the area. Second, no researchers have attempted to explain *why* this relationship exists. In an effort to remedy this, we shall lay groundwork for the existence of such a model. It is our view that a stronger understanding of both SS and how they relate to CS will help researchers to pinpoint the effect of SS training and what gains it may provide in a computing context. Our next step therefore is to chart the SS territory more clearly than we have found elsewhere in the literature.

Understanding Spatial Skills

Spatial skills is a broad term lacking a concise definition, and as such making clear, distinct arguments about them can prove difficult. Tartre effectively summarises problems faced in discussing and communicating spatial ability and their impact:

“Attempting to understand and discuss something like spatial orientation skill, which is by definition intuitive and nonverbal, is like trying to grab smoke: The very act of reaching out to take hold of it disperses it. It could be argued that any attempt to verbalize the processes involved in spatial thinking ceases to be spatial thinking.” [30]

To reduce ambiguity and overcome issues pertaining to the rift between written descriptions of SS and their practical applications, various tests of specific SS factors are given when they are introduced into discussion. Most of these tests have been extracted from Ekstrom *et al.*'s manual for factor-referenced cognitive tests [8].

Over years of discussion and exploration of this difficult field, Carroll collates a wealth of research into a cohesive model consisting of the following factors [2]:

- Spatial Visualisation
- Spatial Relations
- Closure Speed
- Closure Flexibility
- Perceptual Speed
- Visual Imagery (though Carroll identifies this factor as a theoretical factor, without coming to a clear conclusion on its definition)

Spatial visualisation is the factor that has been most examined in relation to STEM, including CS. McGee identified spatial visualisation prior to Carroll as one's proficiency in being able “to mentally rotate, twist, or invert pictorially presented visual stimuli” [14]. Tartre presents a substructure of two distinct factors contributing to spatial visualisation: mental rotation and mental transformation [31]. Mental transformation involves the manipulation and modification of objects, required for such practical applications as visualising cross sections or intra-part

movements. This can be seen in practice in the Mental Cutting Test (MCT) [4] in figure 1. Mental rotation is the ability to perform rotations on mental constructs. Practically this typically translates to the ability to see a physical representation of a structure (a block on a table or an image on a piece of paper) and mentally imagine what this object or shape would look like rotated in a different orientation. Ho and Eastman discovered that 2D and 3D rotations are closely related, supporting Carroll, but also that one capable of performing 2D rotation may not be capable of performing 3D rotation [11]. An example of a test of 2D rotations is displayed in figure 2 [34], and an example of a 3D rotation test can be found in figure 3 [36].

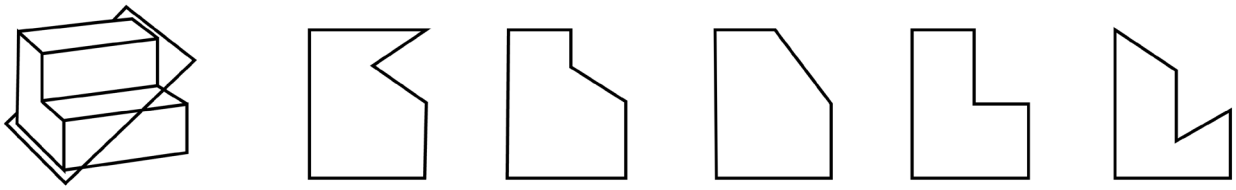

Figure 1: Test of mental transformation, the Mental Cutting Test, consisting of 25 items in 20 minutes - identify the cross section after the following transformation has occurred (answer: second from right) [4]

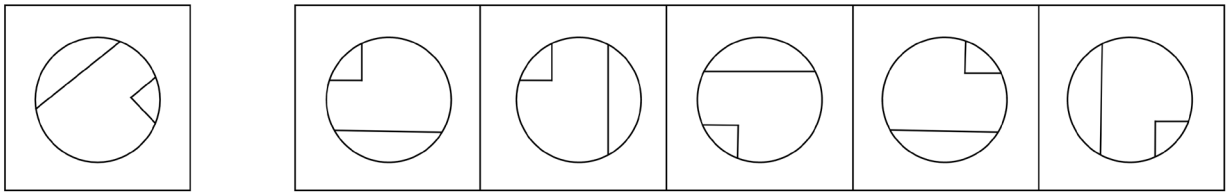

Figure 2: Test of 2D mental rotation, consisting of 24 items and presented as part of a larger test - which of the following corresponds to the original shape (answer: first from left) [34]

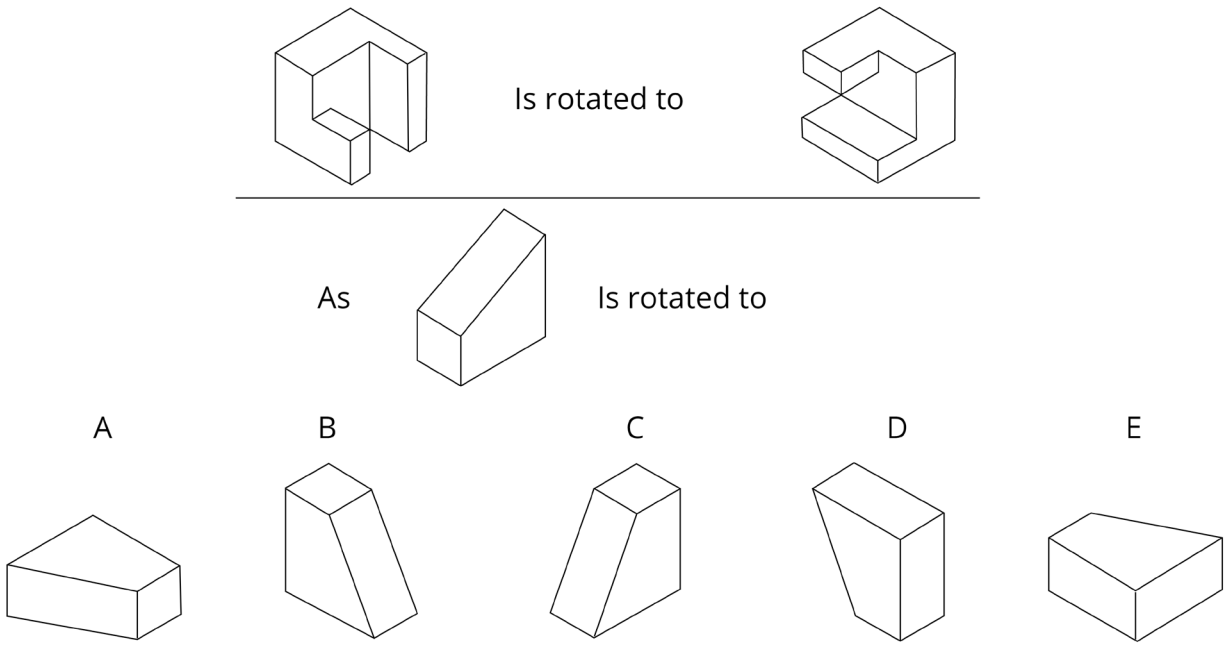

Figure 3: Test of 3D mental rotation, the revised Purdue Spatial Visualisation Test of Rotations (PSVT:R), consisting of 30 items in 20 minutes (answer: B) [36]

Mental rotation is related to another core factor of SS: spatial relations, which is the ability to understand the arrangement and orientation of objects or patterns within their environment. While this initially appears very similar to mental rotation, spatial relation applications do not strictly require rotation to take place, merely a decent understanding of object orientation. In practice, a test used to measure spatial relations is the Cube Comparison Test [8], displayed in figure 4 - as can be seen, to find the correct answer, objects do not need to be rotated (which in fact, would be difficult to do with the lack of information of the object); the examinee just needs to be able to relate each face of the cube to its neighbours.

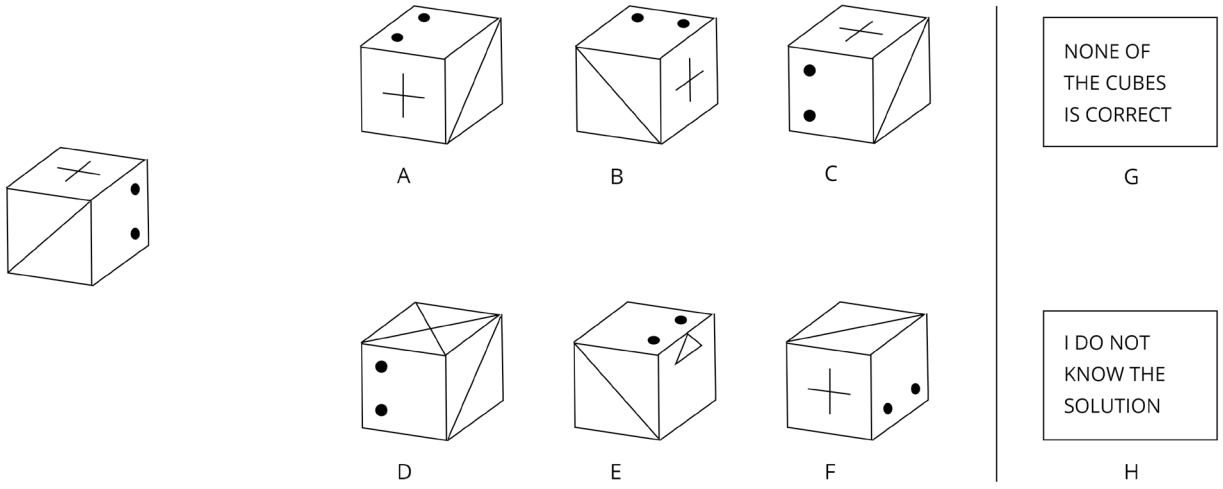

Figure 4: Test of spatial relations, the Cube Comparison Test, consisting of 25 items in 20 minutes - identify which of the following options corresponds to the original cube (answer: D) [4]

Three further factors can be defined as follows:

- **Closure Speed:** speed in identifying an *unknown* pattern from an *obscured* environment
- **Closure Flexibility:** speed in identifying a *known* pattern from an *obscured* environment
- **Perceptual Speed:** speed in identifying a *known* pattern from an *unobscured* environment

The easiest way to perceive the application of these skills is using the tests associated with them. Closure speed is measured by the Gestalt Completion Test [28] (figure 5), which requires the test subject to pick out a representation of an object or image from a highly distorted image (in this example, a flag and hammer head). Closure flexibility can be tested by the Hidden Figures Test [8] (figure 6), in which the test subject is provided with a selection of figures (which are *known*) and a complex pattern, and are required to identify which of the given figures is obscured within the pattern. Perceptual speed is tested by the Identical Pictures Test [8] (figure 7), in which the test subject is presented with a figure and a lineup consisting mostly of figures *similar* to the given figure, with one figure being identical, and must identify the figure from the lineup matching the one provided.

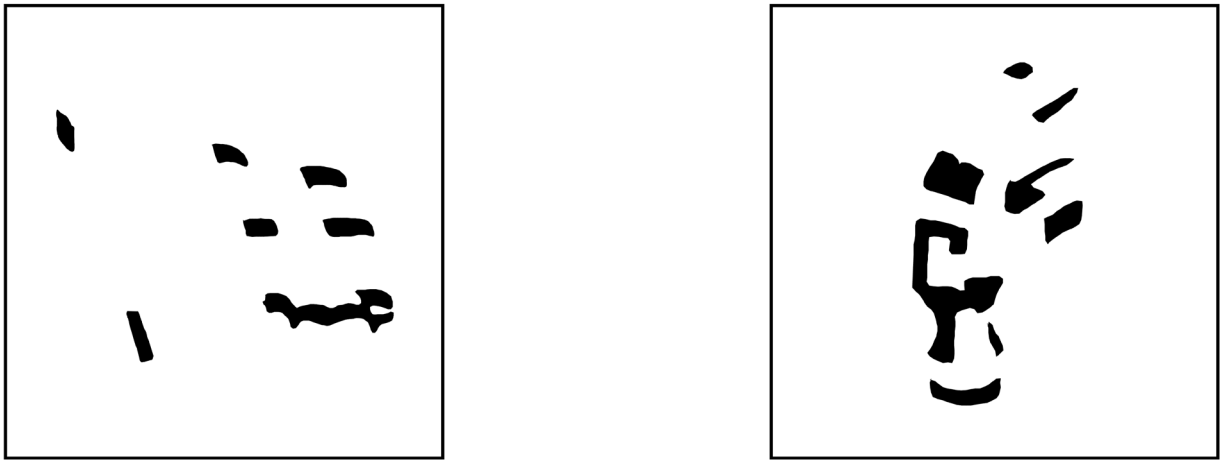

Figure 5: Test of closure speed, the Gestalt Completion Test, consisting of 20 items in 4 minutes (answers: flag, hammer) [28]

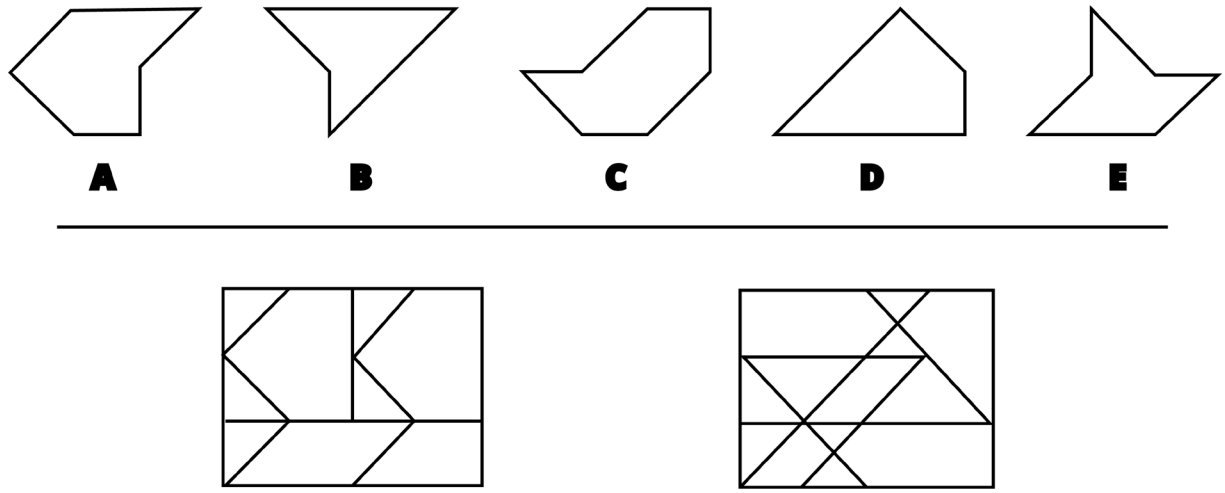

Figure 6: Test of closure flexibility, the Hidden Figures Test, consisting of 32 items in 24 minutes (answers: A, D) [8]

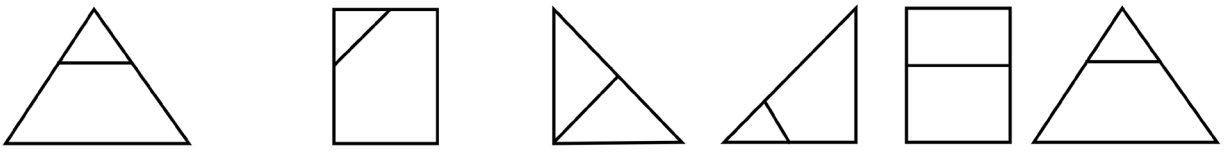

Figure 7: Test of perceptual speed, the Identical Pictures Test, consisting of 96 items in 3 minutes (answer: first from right) [8]

Carroll also identifies a final first order factor of SS as visual imagery. Visual imagery is a somewhat vague factor in the discussion of SS, and lacks the definition and clarity of other first order factors of SS. Burton and Fogarty attempted to measure this factor, and ultimately decided that the best model they constructed was one which included three second order factors contributing to visual imagery [1]. These are:

- **Quality:** "the ability to generate, maintain, and transform a clear visual image"
- **Self-report:** "ability to generate, control, and/or rotate a visual image"
- **Speed:** "latency measures derived from the experimental tasks" - that is, the tasks which were used to determine the existence of the above two factors

These factors fit into spatial skills beneath the term visual imagery, contributing to the theoretical factor which Carroll identified.

Modeling Spatial Skills and CS

With an understanding of SS and the factors contributing to them, we can now attempt to show their connection to CS. We note that existing studies relating SS and CS have focused on programming, and we recognise that the underlying skills in programming, such as the development and manipulation of models and the ability to represent these textually and graphically, are core skills across much of CS. Hence we too will focus on aspects of programming.

A fundamental ability in programming is program comprehension. Much research has gone into examining methods and cognitive frameworks involved in program comprehension [20]. One such model, presented by Détienne and Soloway [6], is the model of a mental schema. A schema is a kind of data structure stored in memory which represents some construct: it consists of a plan, which is some generic process or operation as the user understands it, the function the plan carries out, and cues, which are points of reference used to match up a plan with an associated function. In practice, an application of a schema may consist of identifying key variable declarations or structures in code (such as MAX or COUNT, or the beginning of a loop) and matching them with an associated schema (e.g. a find max schema).

The schema model is of significance because operations involved in building and using a schema can be mapped to SS operations. The identification of cues requires that patterns be extracted from obscured environments, not unlike the process required in the application of closure flexibility. These cues are pointers to a model or structure which must be constructed mentally in order to formulate a process. This is similar to several exercises in Sorby's workbook involving the composition of isometric 3D objects from a selection of 2D orthographic views, taking note of specific, useful data points and constructing a more complex structure combining this data.

Another code comprehension framework is the Block Model proposed by Schulte [21]. This involves a process of examining code at four levels, to identify (1) atoms (single words or simple statements in the code), an understanding of which is used to construct (2) blocks ("regions of interest that syntactically or semantically build a unit"), (3) relations (connections involving blocks and atoms such as a *find maximum* code section) and (4) the macro structure (the overall operation of the program). The method of building up from atoms to blocks and relations is similar to Détienne and Soloway's process of schema construction, and likely requires the same cognitive processes, again relating to the application of SS.

Another important aspect of program comprehension is the notional machine, first identified by du Boulay as a combination of knowledge - of the programming language, environment and data - and a mental model [7]. Sorva describes the function of a notional machine as "an idealized abstraction of computer hardware and other aspects of the runtime environment of programs." [27] Sorva closely connects the ability to form notional machines, and therefore appropriately and effectively comprehend programs, with the ability to construct abstract mental models. Experts develop more robust, adaptable mental models than novices, whose mental models tend to be "fragile". Sorva discusses the "runnable" nature of a mental model, based on Norman's work [15], involving the user being able to "envision with the mind's eye how a system works," and directly associates this with working memory and visualisation.

When reviewing spatial skills factors, there are only two which match up with this process of forming a mental model: spatial visualisation (as Sorva briefly suggests) and spatial relations. Closure speed, closure flexibility and perceptual speed are all related to identifying patterns from environments, and visual imagery relates to capturing and recalling images, leaving the two aforementioned factors. An element of spatial relations would be required to construct a mental model, as the user requires an understanding of how various components are linked together (of how they *relate*), but spatial visualisation provides more robust abilities for these tasks. A robust mental model must be subject to development and restructuring as required - the ability to perform these actions mentally is closest to mental transformation (the ability to manipulate or modify a structure mentally) which is part of the spatial visualisation factor. An element of spatial relations may also be included, but typically spatial relations consist of a simple inter-object understanding (see figure 4 for an example) compared with mental rotation, which requires a deeper understanding of the constructs involved (see figure 3). This indicates that when trying to understand more complex constructs in a mental model and what they would look like in a different orientation or situation, spatial relations are likely to work to an extent, but the more complex operations are more likely to require mental rotation (another subset of spatial visualisation).

A difficulty here arises with the definitions of spatial skills factors as given in the literature. From the CS side, considering mental models, we are constructing a mental representation of some operation or process. However, this does not directly relate to a specific factor of spatial skills, forming a neat, clear connection between the two. We identify the closest match as spatial visualisation, where typically the same ability to construct a mental structure is

required before then performing some operation on it, such as a rotation or transformation. As such, we theorise that spatial visualisation is very likely to contribute to program comprehension in this regard.

In addition to program comprehension, another core aspect of computing is the procedure of program generation. While generation must be closely related to comprehension, as any generation plan must also involve a process of debugging and review [20], there are elements of program generation not included in comprehension.

Rist observed a method of program generation which he named “focal expansion” [18]. The process of focal expansion involves reviewing a problem and identifying a core function or plan on which to base the implementation. The process which follows involves taking the core plan and building outward, adding and expanding as necessary to facilitate the generation of a program that fully satisfies the problem. Rist links this process to working memory, and associates the ability to track the program generation mentally, from the focal point out to the full solution, with working memory capacity~\cite{rist1995program}. While this does appear to be the case, it is also possible that visualisation factors into the programmer's ability to track the expansion: to quote Sorva again, ``to envision in the mind's eye." Also pertinent to program generation is problem comprehension, the process of identifying a problem from some specification - this process is similar to the schema process of identifying a plan in practice, except that rather than looking for cues in a program they must be extracted from a problem description.

Cues are a recurring concept in both program comprehension and problem comprehension which has briefly been touched on. This involves the process of identifying potential patterns from a broader environment consisting of more details than the user is currently interested in. There are also factors of SS which, in practical use, are used in performing a very similar task to these operations: closure speed, closure flexibility and perceptual speed. Recall that these processes involve the extraction of patterns (known or unknown) from environments (obscured or unobscured).

The simplest factor is perceptual speed, which is simply identifying a known pattern from an unobscured environment - though rare, this may have an application in cue identification. In code comprehension a case may arise where the user knows the construct they are looking for and the code is laid out in such a way that there is minimal interaction and obscurity between lines (an example of this may be looking for a known variable name in a list of declarations, such as at the top of a file). More likely, the user will be searching for an pattern to match against a record of terms which they feel may have significance based on prior knowledge (such as the start of a loop or declaration of some telling variable). Pictorially, this would be very similar to the test in figure 6, so it is possible that the cognitive process involved in closure flexibility is relevant to this style of program comprehension. And finally we have a case for closure speed, in which an unknown pattern must be derived from an obscured environment. This is akin to an application of the schema model, in which the user searches the code space to identify cues which are not previously known to match them to known schema - at least this would be the case for experts; novices are likely to take a different approach which will involve less searching and pattern matching and more construction.

In this section so far, we have analysed significant aspects of CS and connected them to SS, forming the elements of a model, a diagrammatic representation of which is presented in figure 8.

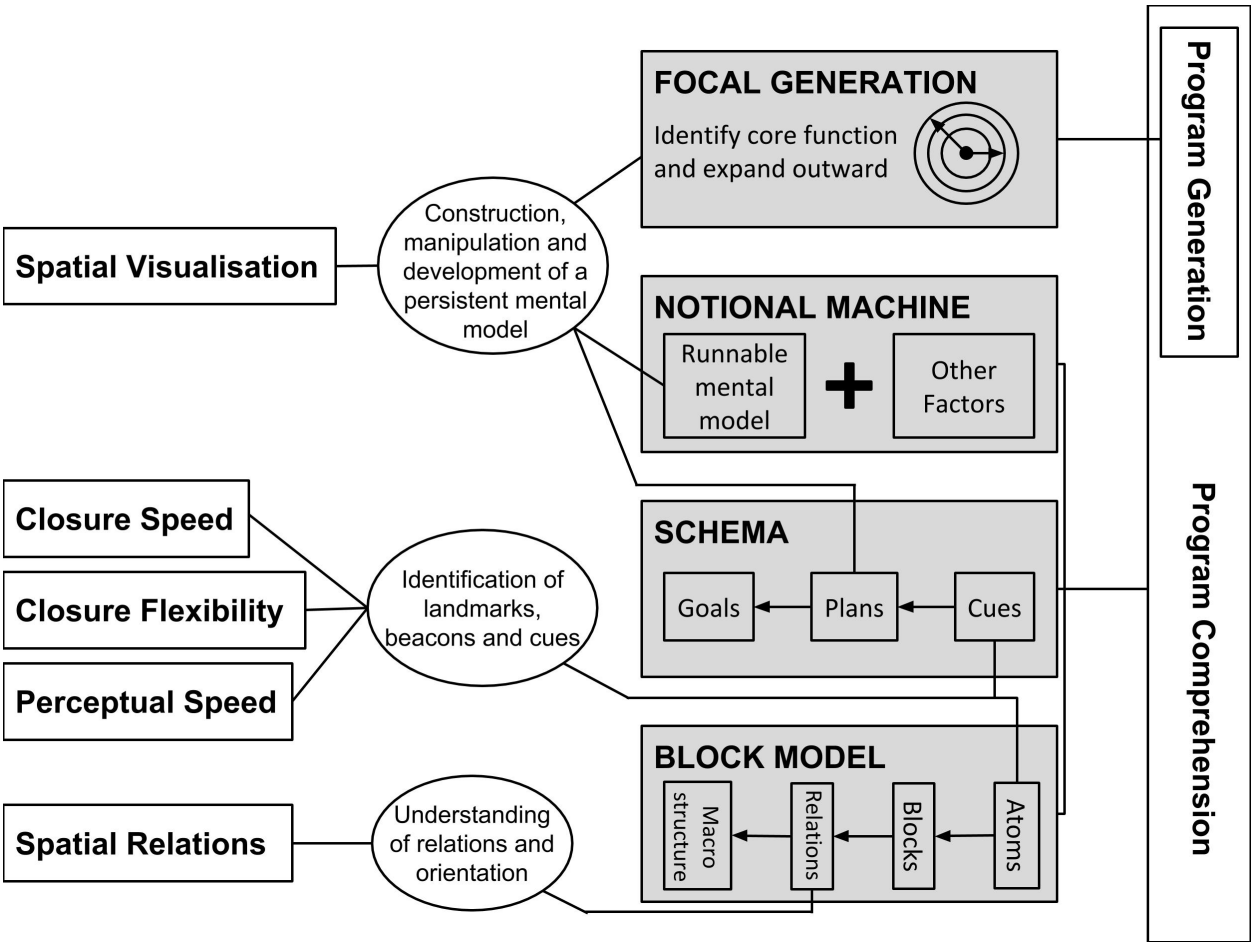

Figure 8: Diagram of the relationships observed between spatial skills and computing science

Bearing this model in mind, we shall now discuss the implications for the development of SS and what this could mean for CS. Sorby notes that the most effective method of training spatial skills is by hand sketching diagrams and drawings [22]. This can be seen in action in her workbook, which poses dozens of short form drawing questions to be completed over a relatively short period of time. We expect that the reason why spatial skills are connected with computing is because \textit{the same }cognitive functions are involved in computing and also in other more obvious applications of spatial ability, such as Sorby's exercises. This view suggests that while SS training could affect one's computing ability positively, as Pallrand observed in physics [16], so too could training in CS develops SS, also as observed by Pallrand for physics.

If this view is correct, why would SS training be of any benefit when the same could be achieved with a standard computing course? We propose that SS training, such as Sorby's workbook, is far more focused and directed than a typical programming course. Whereas in an entry level programming laboratory, students may be expected to write

a handful of short programs to achieve given goals over the space of a couple of hours, Sorby's exercises can consist of up to forty sketches to be completed in a similar time frame. Furthermore there are far fewer barriers to advancement: any given drawing could be attempted regardless of the student's experience (a complete novice who has never done any spatial skills training could pick up Sorby's book and attempt the questions), compared with a programming student who must first learn code snippets required for tasks before they can be reused in later tasks. The same skills are being developed, but at a slower rate for programming students who are learning both the CS content and the underlying skills we are interested in here. It is also possible that the students who fall behind in programming are the ones whose SS are not as developed as their peers, and the barriers to their progression are rooted in their inability to construct robust and adaptable mental models (key to programming comprehension and generation).

With this in mind, we suggest that spatial skills themselves do not directly contribute to CS or other STEM domains, but rather than the cognitive functions involved in SS are also involved in STEM domains: such functions as the ability to form, manipulate and develop mental models, identify key points in an environment and understand relations between structures. Based on this theory, we present a simple model for this relationship in figure 9.

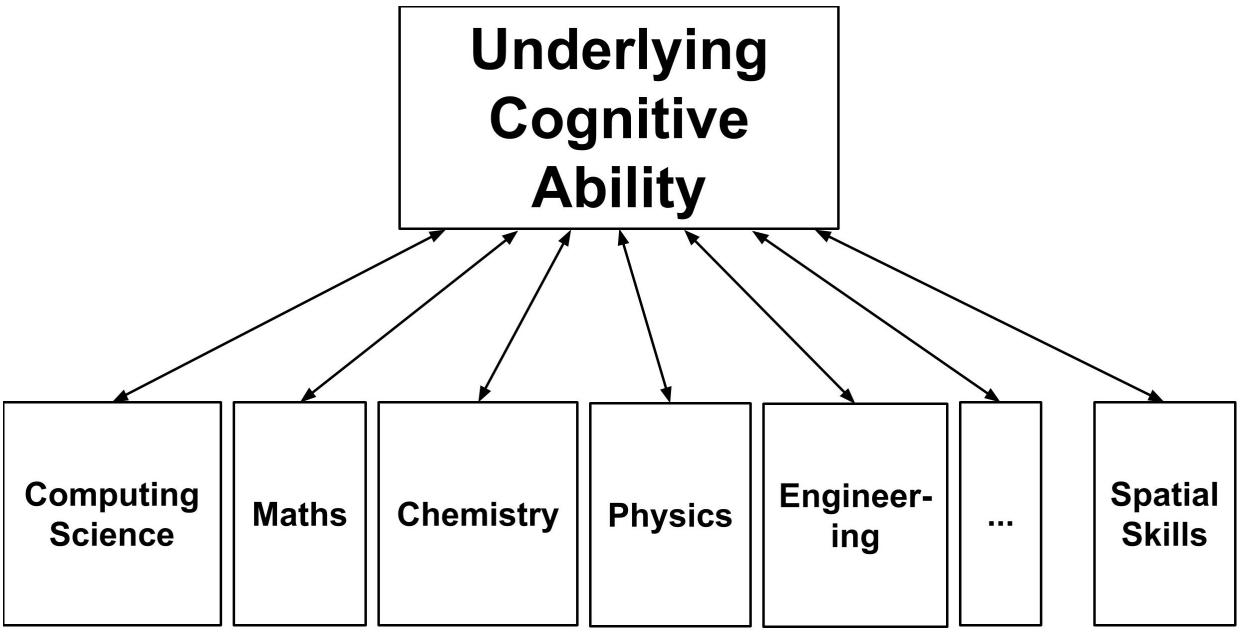

Figure 9: Relationship between cognitive functions behind spatial skills and STEM domains

Notice that the relationships between domains and the underlying cognitive ability are bi-directional. As stated, we believe that these cognitive skills can be developed by pursuing a STEM domain or by training SS, however due to the direct and precise nature of spatial skills training, this route is likely to produce results more effectively. Moreover, training this ability in one area is likely to have an effect on other areas which make use of the same skills: by training spatial skills, we may see one's ability to write or understand programs improve, or to study physics, chemistry, and so on.

Spatial Skills and CS Attainment Level

In previous experiments and studies of SS and CS, it has been typical to examine students either in their first year of study in computing (as an undergraduate degree, a Master's degree or an elective module) or about to start the former. Jones and Burnett's study suggests that those who are better at programming are likely to have better SS, but this is only across a single year.

To better understand the relationship between SS and CS beyond a single year, the SS of students and staff at different levels of attainment in CS at the authors' R1 institution were measured. The study involved two research questions:

- **RQ1:** Do spatial skills vary with academic attainment in CS?
- **RQ2:** Do spatial skills vary with specialisation areas in CS?

For RQ1, given Jones and Burnett's results, it was expected that the higher the attainment and so the “better” the CS skills, the higher the SS will be.

RQ2 draws on Jones and Burnett's finding that SS are not significantly connected to non-programming courses. Each test participant was asked to record their specialised or most favoured area of computing. With this data, it was expected that those involved in heavily programming oriented areas of CS - such as software engineering or systems development - would overall have higher SS than those who were focused on more human based courses, such as HCI or human-centered security.

Furthermore, studies have indicated that gender can affect the SS of participants [9, 32]. In order to account for this potential confound, the gender of each participant was also recorded.

Method

Five cohorts were selected from which to draw participants:

- First year students, taking a CS0-style course designed for those without programming experience, many of whom are not intending to major in CS
- Honours undergraduate students in their 3rd/4th year majoring in CS, who predominantly take the same courses at the same level
- MSci students, in the fifth year of an undergraduate Masters programme
- PhD students
- Academic staff from the CS department

30 participants were randomly selected from each group and then invited to take the SS test, with the exception of the first years who all took the SS test during a lecture and 30 were randomly selected.

Participants were also required to indicate their specialised or preferred area of research. To reduce the granularity of the data, the participants were arranged according to which of the department's CS primary research areas they fell under, of which there are four. For the purpose of this paper, they are named as follows:

- **HCI:** Human-computer interaction and human factors
- **Data:** machine learning, info retrieval and data science
- **Sys:** systems engineering and networks
- **Th-Alg:** algorithms, computational thinking, formal analysis and mathematical modeling

First year students were not required to indicate a preferred area of computing. Additionally, some participants opted not to provide this information. Table 1 details the breakdown of the participants.

Level	Male	Female	Sys	HCI	Data	Th-Alg	Total
Level 1	19	11					30
H Level	9	4	4	1	3	1	13
MSci	7	2	1	0	4	3	9
PhD	5	5	3	3	1	3	10
Staff	8	2	2	1	1	5	10
Total	43	29	10	5	9	12	72

Table 1: The characteristic breakdown of participants

The test used was the Revised PSVT:R [36]. For all intents and purposes the Revised PSVT:R consists of the same questions as the original PSVT:R by Guay [10], but has been updated to have some graphical errors fixed and the questions arranged in order of difficulty. The test consists of 30 items with a 20 minute time limit.

The test was provided in two separate formats: online and on paper. Online, the test was accessible through the institution's Moodle platform via a quiz with a time limit. On paper, one of the authors was present to ensure that the test was completed within the time limit and that the participants were not looking up answers or conferring. While we cannot absolutely confirm that those who completed the test online did not confer, the timer was not pausable and once the test was begun could not be reset, so in order to cheat participants realistically would have had to have begun the test with the intention of doing so. In addition to the fact that the answers to the PSVT:R are not readily available online, we do not expect that any of the participants would have attempted to invalidate the research deliberately. A more realistic concern is that people who completed the test without supervision may have used scratch paper or some similar aid in completing the questions.

Once the tests were completed, the scores were collated along with level of attainment and demographic data, by which stage no names or other sensitive data was attached to any of the results.

Analysis of Results

Once the data had been collected, the mean and standard deviation for each group was calculated and are displayed in table 2. After breaking down participants into groups based on attainment level and gender, the SS means of these groups are displayed in table 3.

	Level 1	H Level	MSci	PhD	Staff
Mean	18.97	22.92	24.67	22.00	25.50
SD	6.21	4.59	4.00	6.13	3.60
n	30	13	9	10	10

Table 2: The mean, standard deviation and number of participants for each cohort

	Level 1	H Level	MSci	PhD	Staff	Total
Male	20.42	24.71	24.71	24.60	25.00	22.46
Female	16.46	26.00	24.50	19.40	27.50	20.25
HCI		23.00	-	19.00	27.00	21.40
Data		22.00	24.00	13.00	29.00	22.67
Sys		22.25	20.00	23.00	27.00	22.40
Th-Alg		27.00	25.67	27.00	24.20	25.50
Total	18.97	22.92	24.67	22.00	25.50	21.72

Table 3: The means for each factor being analysed

To confirm the validity and significance of the experiment, a two way analysis of variance (2-way ANOVA) was conducted. The results of this statistical method are displayed in table 4. Due to the unbalanced nature of the data, sum of squares Type II was used.

Source	DF	SS	MS	F	p
Academic Level	4	429.534	107.383	3.893	0.007
Gender	1	39.507	39.507	1.432	0.236
Academic Level * Gender	4	202.473	50.618	1.835	0.133
Error	62	1710.410	27.587		
Corrected Total	71	2420.444			

Table 4: 2-way ANOVA significance and interaction between factors (in this instance, SS denotes Sum of Squares, MS denotes the Mean Square and DF denotes Degrees of Freedom)

As can be seen here, the main effect of academic level is significant ($p<0.01$). Although the average SS score of male participants was slightly higher, neither this nor the interaction between gender and attainment level were found to be significant.

Once the ANOVA identified that the main effect was significant, the effect size between groups was calculated using Hedges' *g*, favoured in this case over Cohen's *D* due to the small sample sizes of some groups. The results of this analysis are displayed in table 5. While recommending caution Cohen suggests that effects of 0.2=small, 0.5=moderate and 0.8=large.

	Level 1	H Level	MSci	PhD	Staff
Level 1	-				
H Level	0.672	-			
MSci	0.963	0.384	-		
PhD	0.480	-0.168	-0.487	-	
Staff	1.124	0.592	0.210	0.667	-

Table 5: The effect size between groups, using Hedges' g

One third of participants completed the test on paper and two thirds completed the test online (with the exclusion of the first year students, who all completed it on paper). In order to check for bias, the average scores two groups were compared. There was a slight bias in favour of the participants who completed the test on paper.

Discussion

As expected, with the exception of the PhD students, the average SS ability of each cohort increased as academic attainment increased. By examining the effect size between groups, we can see the Honours cohort being better than the level 1 cohort, the MSci cohort were better than them and so on. Although the incremental effect sizes are quite small, they compound to display large differences between cohorts on either ends of the scale.

One theory for the PhD students not fitting this pattern is that their backgrounds are considerably more varied than any of the other cohorts tested. While the first year students will have graduated from differing high school programmes and curricula, all the students tested from the level 1 course specifically chose this course as they had limited programming experience, significantly balancing the background of the cohort. Honours and MSci students will have undertaken different modules, have different preferences and specialisations, but will all be some way along the same course at the same level of assessment. Staff members will also have had a varied background, however it can safely be assumed that they have achieved a relevant PhD and will have several years of experience.

Conversely, PhD students attending the institution in question come from a wide range of first degrees undertaken at universities around the world. Each of these courses will have different focuses, teaching styles and methods of assessment, which may have had an influence on SS development. The test requires some reading at the start,and so there may be language issues. Note also that the deviation in scores for the PhD students is high, and in fact the highest recorded score on the test (a perfect score of 30) was achieved by a PhD student - indicating that rather than the group generally having SS out of sync with their level of advancement, instead, the spread is much broader than in other cohorts.

With the exception of the anomalous PhD students, a clear pattern can be seen in that the average SS of those with higher levels of attainment are higher than those with a lower level of attainment. While this takes us a step closer to understanding the relationship between SS and CS, there are multiple conclusions which can be drawn from this result. One is that as one progresses in computing science, their SS are improved by the exercises and practices they are required to develop, as Pallrand noticed in physics. An alternative theory is that as cohort members progressed, only those with initially higher SS advanced, either because those with lower skills *could not* or *chose not* to. Both options are possible, and a longitudinal study of a cohort progressing through the system would be required to decidedly identify which hypothesis is true, if either.

Concerning the research area of each participant and the mean scores of these groups, the results partially support the study by Jones and Burnett, as the HCI area has the lowest average SS. However, if programming were the primary factor to which SS contributed, it would be expected that the Sys area would have the highest SS, since this is the area most focused on working with programs. The Th-Alg section have the highest average SS, indicating that some other factor in CS is likely to be related to SS.

It must be noted that there are significant differences between the participants involved in this study and those in Jones and Burnett's study. Jones and Burnett's cohort were a relatively known quantity, with participants having

different backgrounds but none having done computing, and all being required to take the mentioned courses. The research groups in this institution are far more diverse, with members of staff having differing levels of experience, track records and overlapping interests. Additionally, some participants were not members of a research section and were only able to express their interests. Further, regardless of what research section one is associated with, this does not strictly indicate how much work they do which directly ties into this field. The purpose of collecting this information was to investigate in broad, preliminary terms whether or not this study's matched Jones and Burnett's. We feel that primarily it does, though also indicates that there is more at play than just programming, which supports our model.

Owing to the fact that there was a slight bias in favour of those completing the test on paper vs those completing it online, we should highlight that future experiments of this nature should be conducted using one method only to eliminate this bias. It is an interesting result, however, since we expected that anyone who was not supervised as they attempted the test would be more likely to have access to scratch paper or some other tools and would therefore score higher. Our suggestion moving forward would be to have everyone complete the test on paper under supervision.

A final thought on the experiment described here was how these results would compare with other subjects, both in STEM and outwith. This was considered, but unfortunately was not feasible with the time and resource constraints of the project. It would be useful to see how closely related the results would be in other STEM fields and particularly to see if non-STEM fields follow the same pattern. However, regardless of what these results may indicate, it is still felt that the somewhat narrower view of this experiment yields valuable insights into the relationship between SS and CS.

Conclusion

In this paper we have reviewed literature concerning the relationship between SS and STEM, particularly in CS. This literature indicates that a correlation between SS and CS exists, with one study displaying what has been interpreted as a causal effect. Furthermore, we have identified that in one STEM field, SS improve over a period of learning - not as much as if they had received directed SS training, but more than a liberal arts student - which indicates that the relationship is likely to be a biased two-way relationship.

We have also collated and presented a substantial discussion of spatial skills themselves, condensing and summarising a broad field in a format which is easy to grasp for the relatively uninitiated. Based on this, we have presented a model for the relationship between SS and CS. This model is rooted in existing research into cognition in CS, particularly in program comprehension, program generation and problem comprehension. The model indicates that particular factors of SS are likely to have an effect in the reading and identification of key points in code or problems, as well as the mental models constructed in attempting to understand programs and theoretical problems.

Finally, we conducted an experiment to strengthen our understanding of the relationship between SS and CS achievement, showing that in general the average SS of a cohort increases with academic attainment, extending the research undertaken by Jones and Burnett. The experiment also supports our model connecting SS with CS, as the research area with the highest average SS was the section who engage mostly in abstract and theoretical thinking.

Our contribution furthers our understanding of SS and their relation to CS and lays the groundwork for a larger experiment to determine if the relationship is causal. If SS training does benefit computing ability substantially, then it is worth introducing on a large scale, due to its cost-effectiveness, high accessibility and easy implementation.

References

1. Lorelle J Burton and Gerard J Fogarty. 2003. The factor structure of visual imagery and spatial abilities. *Intelligence* 31, 3 (2003), 289–318.
2. John B Carroll. 1993. *Human cognitive abilities: A survey of factor-analytic studies*. Cambridge University Press.
3. Carolyn S Carter, Mary A Larussa, and George M Bodner. 1987. A study of two measures of spatial ability as predictors of success in different levels of general chemistry. *Journal of research in science teaching* 24, 7 (1987), 645–657.
4. CEEB. 1939. CEEB Special Aptitude Test in Spatial Relations, developed by the College Entrance Examination Board, USA. (1939).
5. Stephen Cooper, Karen Wang, Maya Israni, and Sheryl Sorby. 2015. Spatial skills training in introductory computing. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*. ACM, 13–20.
6. Françoise Détienne and Elliot Soloway. 1990. An empirically-derived control structure for the process of program understanding. *International Journal of Man-Machine Studies* 33, 3 (1990), 323–342.
7. Benedict Du Boulay. 1986. Some difculties of learning to program. *Journal of Educational Computing Research* 2, 1 (1986), 57–73.
8. Ruth B Ekstrom, Diran Dermen, and Harry Horace Harman. 1976. *Manual for kit of factor-referenced cognitive tests*. Vol. 102. Educational Testing Service Princeton, NJ.
9. Elizabeth Fennema and Julia Sherman. 1977. Sex-related differences in mathematics achievement, spatial visualization and affective factors. *American educational research journal* 14, 1 (1977), 51–71.
10. Roland Guay. 1976. *Purdue Spatial Vizualization Test*. Educational testing service.
11. Chun-Heng Ho, Charles Eastman, and Richard Catrambone. 2006. An investigation of 2D and 3D spatial and mathematical abilities. *Design Studies* 27, 4 (2006), 505–524.
12. Sue Jones and Gary Burnett. 2008. Spatial ability and learning to program. *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments* (2008).
13. Maria Kozhevnikov, Michael A Motes, and Mary Hegarty. 2007. Spatial visualization in physics problem solving. *Cognitive Science* 31, 4 (2007), 549–579.

14. Mark G McGee. 1979. Human spatial abilities: Psychometric studies and environmental, genetic, hormonal, and neurological influences. *Psychological bulletin* 86, 5 (1979), 889.
15. Donald A Norman. 2014. Some observations on mental models. In *Mental models*. Psychology Press, 15–22.
16. George J Pallrand and Fred Seeber. 1984. Spatial ability and achievement in introductory physics. *Journal of Research in Science Teaching* 21, 5 (1984), 507–516.
17. Jeffrey R Pribyl and George M Bodner. 1987. Spatial ability and its role in organic chemistry: A study of four organic courses. *Journal of research in science teaching* 24, 3 (1987), 229–240.
18. Robert S Rist. 1989. Schema creation in programming. *Cognitive Science* 13, 3 (1989), 389–414.
19. Robert S Rist. 1995. Program structure and design. *Cognitive Science* 19, 4 (1995), 507–561.
20. Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and teaching programming: A review and discussion. *Computer science education* 13, 2 (2003), 137–172.
21. Carsten Schulte. 2008. Block Model: an educational model of program comprehension as a tool for a scholarly approach to teaching. In *Proceedings of the Fourth international Workshop on Computing Education Research*. ACM, 149–160.
22. Sheryl A Sorby. 1999. Developing 3-D spatial visualization skills. *Engineering Design Graphics Journal* 63, 2 (1999).
23. Sheryl A Sorby. 2007. Developing 3D spatial skills for engineering students. *Australasian Journal of Engineering Education* 13, 1 (2007), 1–11.
24. Sheryl A Sorby. 2009. Educational research in developing 3-D spatial skills for engineering students. *International Journal of Science Education* 31, 3 (2009), 459–480.
25. Sheryl A Sorby and Beverly J Baartmans. 1996. A Course for the Development of 3-D Spatial Visualization Skills. *Engineering Design Graphics Journal* 60, 1 (1996), 13–20.
26. Sheryl A Sorby and Anne Francis Wysocki. 2003. *Introduction to 3D Spatial Visualization: an active approach*. Cengage Learning.
27. Juha Sorva. 2013. Notional machines and introductory programming education. *ACM Transactions on Computing Education (TOCE)* 13, 2 (2013), 8.
28. Roy Frink Street. 1931. A Gestalt completion test. *Teachers College Contributions to Education* (1931).
29. Donald E Super and Paul B Bachrach. 1957. Scientifc careers and vocational development theory: A review, a critique and some recommendations. (1957).
30. Lindsay Anne Tartre. 1990. Spatial orientation skill and mathematical problem solving. *Journal for Research in Mathematics Education* (1990), 216–229.
31. L A Tartre. 1990. Spatial skills, gender and mathematics. In *Mathematics and Gender*, E Fennema and G Leder (Eds.). Teacher's College Press, New York, Chapter 3, 27–59.
32. David H Uttal, Nathaniel G Meadow, Elizabeth Tipton, Linda L Hand, Alison R Alden, Christopher Warren, and Nora S Newcombe. 2013. The malleability of spatial skills: A meta-analysis of training studies. (2013).

33. Norma L Veurink and Sheryl A Sorby. 2011. Raising the bar? Longitudinal study to determine which students would most beneft from spatial training. In *American Society for Engineering Education*. American Society for Engineering Education.
34. Jonathan Wai, David Lubinski, and Camilla P Benbow. 2009. Spatial ability for STEM domains: Aligning over 50 years of cumulative psychological knowledge solidifes its importance. *Journal of Educational Psychology* 101, 4 (2009), 817.
35. LL Wise, DH McLaughlin, and L Steel. 1979. The Project TALENT data handbook, revised. *Palo Alto, CA: American Institutes for Research* (1979).
36. So Yoon Yoon. 2011. *Psychometric properties of the revised purdue spatial visualization tests: visualization of rotations (The Revised PSVT: R)*. Purdue University

Author Details

Jack Parkinson
School of Computer Science
University of Glasgow
Glasgow
Scotland
jack.parkinson@glasgow.ac.uk

Quintin Cutts
School of Computer Science
University of Glasgow
Glasgow
Scotland
quintin.cutts @glasgow.ac.uk