The Explanatory Visualization Framework: An active learning framework for teaching creative computing using explanatory visualizations

Roberts, J. C.; Ritsos, P. D.; Jackson, J. R.; Headleand, C.

IEEE Transactions on visualization and computer graphics

DOI:
10.1109/TVCG.2017.2745878

Published: 01/01/2018

Peer reviewed version

J. C. Roberts, P. D. Ritsos, J. Jackson, and C. Headleand, “The explanatory visualization framework: An active learning framework for teaching creative computing using explanatory visualizations,” *IEEE Transactions on Visualization and Computer Graphics*, vol. x, no. x, pp. xx-xx, Jan. 2018. DOI http://ieeexplore.ieee.org/document/8017594/

Abstract:
Visualizations are nowadays appearing in popular media and are used everyday in the workplace. This democratisation of visualization challenges educators to develop effective learning strategies, in order to train the next generation of creative visualization specialists. There is high demand for skilled individuals who can analyse a problem, consider alternative designs, develop new visualizations, and be creative and innovative. Our three-stage framework, leads the learner through a series of tasks, each designed to develop different skills necessary for coming up with creative, innovative, effective, and purposeful visualizations. For that, we get the learners to create an explanatory visualization of an algorithm of their choice. By making an algorithm choice, and by following an active-learning and project-based strategy, the learners take ownership of a particular visualization challenge. They become enthusiastic to develop good results and learn different creative skills on their learning journey.

Definitive Version at:
URL: http://ieeexplore.ieee.org/document/8017594/

@article{Roberts-et-al-TVCG-2018,
 author = {Roberts, Jonathan C. and Ritsos, Panagiotis D. and Jackson, James and Headleand, Chris},
 journal = {IEEE Transactions on Visualization and Computer Graphics},
 title = {The explanatory visualization framework: An active learning framework for teaching creative computing using explanatory visualizations},
 year = {2018},
 volume = {x},
 number = {x},
 pages = {xx-xx},
 month = jan,
 url = {http://ieeexplore.ieee.org/document/8017594/},
 DOI ={10.1109/TVCG.2017.2745878}}
}

The Explanatory Visualization Framework: An active learning framework for teaching creative computing using explanatory visualizations

Jonathan C. Roberts, *Member IEEE*, Panagiotis D. Ritsos, *Member IEEE*, James R. Jackson, Christopher Headleand

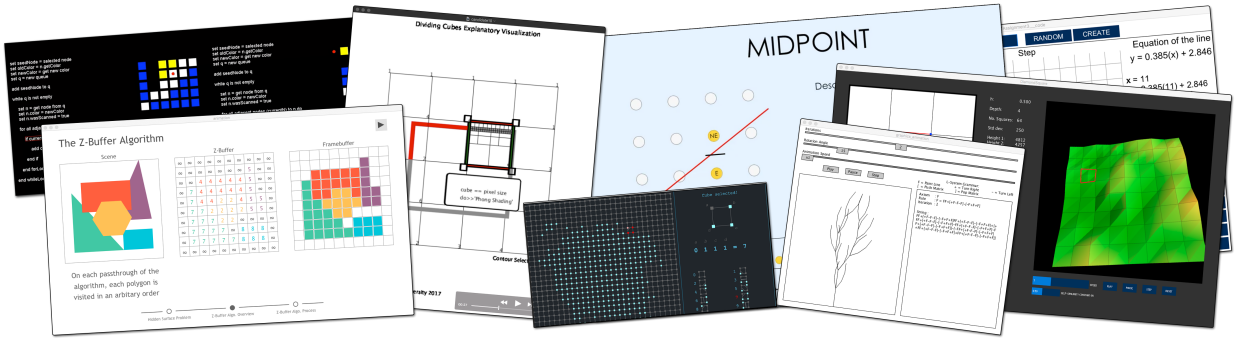


Fig. 1. Students are tasked with creating explanatory visualizations of computer graphics and visualization algorithms. The output from several final-year undergraduate students work is shown.

Abstract— Visualizations are nowadays appearing in popular media and are used everyday in the workplace. This democratisation of visualization challenges educators to develop effective learning strategies, in order to train the next generation of creative visualization specialists. There is high demand for skilled individuals who can analyse a problem, consider alternative designs, develop new visualizations, and be creative and innovative. Our three-stage framework, leads the learner through a series of tasks, each designed to develop different skills necessary for coming up with creative, innovative, effective, and purposeful visualizations. For that, we get the learners to create an explanatory visualization of an algorithm of their choice. By making an algorithm choice, and by following an active-learning and project-based strategy, the learners take ownership of a particular visualization challenge. They become enthusiastic to develop good results and learn different creative skills on their learning journey.

Index Terms—Explanatory visualization, Information Visualization, Teaching visualization, Learning Support

1 Introduction

Visualization now pervades both work and leisure. In parallel, we are seeing a democratisation visualization, with a broad range of people creating and publishing data visualizations. Consequently, we need people with the necessary creative and innovative skills to develop clear and informative visualizations. We certainly want to have visualization professionals and enthusiasts who are creative, passionate about design, knowledgeable about data and excited about creating data-visualization solutions. We require people who are independent creative thinkers who can tackle new challenges and design appropriate solutions. Robinson [53] writes "*We need to create [learning] environments ... where every person is inspired to grow creatively*". There is, therefore, a need to develop teaching methods that empower learners to think for themselves and apply the knowledge that they have already gained [55].

Especially for the study of data visualization, but also for all areas of computer science, we believe that learners need to develop their creative skills. In fact, our long-term vision is to develop strategies and techniques that help teach the next generation of creative thinkers and data-visualization engineers. This article fits within our larger vision to develop fundamental competences. Specifically, the aim of the framework we present, is to develop and hone skills that bolster creative design, promote the consideration of alternative solutions towards implementing more effective visual designs and foster critical evaluation, for learners in higher education.

In this article we present the explanatory visualization framework (EVF), a three-stage, six-part, *active learning* framework where learners design and code their own explanatory visualizations. The article is useful to anyone who is teaching data-visualization techniques or seeks to increase creativity and design acumen of learners.

- *Jonathan C. Roberts is with Bangor University. E-mail: j.c.roberts@bangor.ac.uk.*
- *Panagiotis D. Ritsos is with Bangor University. E-mail: p.ritsos@bangor.ac.uk.*
- *James R. Jackson is with Bangor University. E-mail: j.r.jackson@bangor.ac.uk.*
- *Christopher Headleand is with University of Lincoln. E-mail: cheadleand@lincoln.ac.uk.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

2 How to use this paper

This article expands and extends our poster presentation at IEEE VIS 2016 [51]. In our framework, the students are active participants in all aspects of their learning. Active learning encourages students to do tasks, and think what they are doing [6]. This is in contrast to traditional instruction where students passively listen and teachers do most of the talking. Through this process learners take responsibility for their own learning, guided by the teacher. This experiential learning approach [32] enables them to engage with the process and develop their own skills in reporting, explaining, coding, thinking creatively and reflecting. The EVF contains three consecutive stages, each developing different skills of the learner. For each stage we detail lectures, practical activities and assessment. Students can revisit their work and ideas within each stage. In fact, students are encouraged to regularly reflect on their work, whereas they receive ongoing feedback from the instructor

such that they can develop their skills (formative feedback) and improve their work before submitting for grading. They also receive appropriate feedback (summative feedback) after submission.

As our framework conforms to a whole module (unit of work), we include enough information for a teacher to apply the ideas to their situation. In fact we encourage teachers to adapt the framework for their situation. Consequently, we detail the learning outcomes (Sect. 5.1 and Fig. 3). The framework provides an overarching structure, driven by the requirement to develop an explanatory visualization. Thus, teachers could exchange activities with their favoured method; e.g., while we suggest a sketched planning method (and we use the Five Design Sheets method [49]) it could be possible to use other methods that help students consider alternative concepts and create a final realisation design. Teachers could choose to focus on some of the stages, especially if short of time. E.g., teachers could set stages 1 and 2 (which achieves all learning outcomes) or a single stage could be set as an assignment. But students learn more deeply by completing all stages. Another advantage of the model is that only a few resources are required: the students would need a computer (for stages 1 and 3) and pens, paper and maybe a scanner to achieve stage 2 of the EVF.

The ideas within this framework have matured over the past four years, with problems being ironed out in the first two years. We have used the whole framework in its totality for the last two years in our Computer Graphics & Visualization third-year module. We demonstrate the application of the EVF with examples of student work. The paper has, broadly, two parts. First we detail background and pedagogy on creative design (Sect. 3) and other related work (Sect. 4). Second, we explain the framework in detail (Sect. 5 onwards).

3 Background & Pedagogy

There are many and varied strategies to guide students' learning, beyond the traditional "chalk and talk" lecture. Strategies such as flipped learning and blended learning have been successfully applied to a range of disciplines. Some subject specific approaches are also beginning to emerge, such as, for example, practising computing techniques without computers [63]. Whichever the method used, the aim is still the same: to improve students' learnt knowledge of the area and to develop their skills. As students differ in their propensity to specific learning styles, we need to be flexible as educators. To foster the next generation of skilled data-visualization engineers we will need to develop a variety of suitable educational strategies and resources.

3.1 Active and Project Based Learning

With *active learning* [41] students participate in the process of learning, beyond passively listening. It is an especially effective approach for teaching creative and data-visualization skills. The learners actively practice, and even fail tasks, in order to gradually refine their adeptness in the related activities. Indeed students need a safe environment where failure can be embraced. They need to learn the boundaries and possibilities of what they can do, what is possible, and learn how to improve and adapt their behaviours. Creative thinking is a skill that can be taught, but it takes time and effort on the behalf of the learner. For instance, if sketching is being used in design, students may be afraid to commit on paper their ideas [49]. Yet, through practice and formative assessment they can become confident in design ideation and know how to improve and recover from mistakes. However, if learners focus only on fostering skills, rather than their own inclination and alertness, then they will not develop the mindset required to tackle unfamiliar challenges [22, 46, 47].

Moreover, one of the big challenges with developing creative thinkers is that creative questions are *ill-defined* [52], they are "*vague, fluid and not susceptible to formalization*" [61]. This burdens learners and educators alike; not only because students struggle to answer creative questions, or that they do not know how to break the problem down into smaller parts, but they are also more difficult to grade. When the teacher asks the student to "build a calculator application in Java using the Gridbaglayout", the question is convergent and there is little space for design creativity. But, when asking a more open question there is no one correct answer, and significant room for interpretation.

While educators should not shy away from setting divergent questions, they face a dilemma. On one hand, they should not expect learners to develop their own strategies by merely asking them to "go build a solution". On the other hand they should not say "build this tool with this specification" because that would not enable them to develop their skills sufficiently. What is required is a framework that leads learners through key stages, yet provides them with freedom to explore and design their own imaginative solutions. Our framework provides a balance between being prescriptive, yet providing freedom for the learner to be creative. We are using a particular style of active learning; we are employing project-based learning. With project-based learning (PBL) students work on challenges, resembling authentic real-world problems. They use a variety of skills to make decisions and create solutions, through high engagement and direct involvement. PBL has its origins in the early 1900s [3] and the theories of John Dewey on experiential learning, the constructivist views of Piaget and Kolb's experience-based learning systems [33]. It is an approach that has been gathering momentum across curricula and levels of education.

In our framework, we give the student a precise goal (build an explanatory visualization). However, the desired outcome is deliberately open to interpretation, in terms of how it can be accomplished, to engender creativity and innovation by the student. We outline a journey of how to achieve that goal using our framework, allowing and encouraging students to make choices along the way. When students make choices, they feel empowered and take a more positive attitude to their learning. PBL enables learners to create something new, through studying about the topic, and developing their metacognition and critical thinking skills. However, to create an effective project-based learning environment, students need to be given feedback on how they are progressing, they need to be able to revise their ideas, and they need to think critically over what they have done. We therefore ensure there are plenty of opportunities, for the students, to get feedback from their peers and tutor.

3.2 Explanatory visualization

The overarching activity of the EVF needed to be a task that is ill-defined such to allow creativity, yet constrained enough to keep the students focused on a specific goal. Our original approach was to get students to build a game. However, this was not well suited for our purpose, as students focused on game-play logic, rather than being creative and visual. Instead, we ask the students to create explanatory visualizations of well-known computational algorithms.

People create visualizations to *explore* unknown relationships within the data, *present* results of experiments with the goal to understand the statistical data, to illustrate and to *explain* [10] processes. Our attention is to the latter. There are subtle yet clear differences between *presentation* and *explanatory* visualization. First, the goal of explanatory visualization is to educate. Visual depictions are used to help instruct and upskill. Second, its focus is to elaborate on concepts or processes rather than data sets. Third, is to elucidate what is going on, why it happens, and how it relates to other principles. Within the research community there has been a huge effort in the area of exploratory visualization. Researchers have created investigative tools and interactive systems to help users explore and understand the underlying data. Likewise, visualizations to present results in a clear and meaningful way are used throughout science. However, little research has been published on explanatory visualization.

In particular, the task of creating explanatory visualizations requires the student to take six steps, that help develop the correct mindset and creative skills. With these steps, the student needs to: (1) comprehensively understand the subject material such to deliver a suitable explanation; we consider this to be a "see one, do one, teach one" approach, or in our case "explain one" instead of teach one, (2) understand the end-user and empathise with them to judge whether the end-user would apprehend the information through their explanation, (3) understand storytelling techniques and story development, (4) investigate alternative stories, such to decide upon the best way to present the information, (5) simplify and emphasise the main points (make minor points less obvious or leave them out). Finally, (6) students will need

EVF	Data vis.	Instructional models		Decision processing models		
Roberts et al.	*Sedlmair et al.*	*Jonassen*	*ADDIE*	*Simon*	*Wallas*	*D.Council*
Research	Learn, winnow cast	Articulate	Analyze	Intelligence	Preparation	Discover
Report	Discover	Relate			Incubation	Define
Design	Design	Generate alternatives	Design	Design	Illumination	
Plan		Accept/reject Validate		Choice		
Develop	Implement Deploy	Implement	Development Implement	Implementation		Develop
Reflect	Reflect Write	Adapt	Evaluation	Review	Verification	Deliver

Fig. 2. Comparison of the EVF with the nine-stage visualization model [56], Instructional models (Jonassen [30], ADDIE [8]), and Decision processing models (Simon [61], Wallas [68], Design Council [14]).

to edit the story and decide what information is presented [27].

For our purpose, we get the students to create explanatory visualizations of computer graphics and visualization *algorithms*. We use algorithms because they provide a complex and multifaceted task, are suitable for our computer science students, and the chosen algorithms implicitly extend students' knowledge. We can imagine other tasks to be equally suitable, such as visualising data structures, physics phenomena, or biological processes. While the focus on algorithm explanation can be swapped for another suitable activity, the use of the explanatory visualization style cannot: it is key to the success and uniqueness of this framework. As students need to explain an algorithm in a visual manner, they must understand said algorithm first. Visualization styles, such as exploratory visualization would not get the student to make the same cognitive decisions, as they may provide the means to *describe* or investigate, but not necessarily to *explain in detail*.

We, like many other pedagogical researchers [7, 54, 64, 67] voice a call to raise visual literacy in the classroom. Learners require structure, and direction, to help them develop skills. A part of this learning journey is well resourced. There are many strategies that can be used in class to help teach concepts, ranging from sketches on the board and diagrams shown sequentially in a presentation, to videos and demonstration of visualization tools for constructing physical models [26]. However, there are comparatively few frameworks available to guide and advance skills in creativity and visualization in the computing curriculum.

4 Related Work

There are several decision processing models that we draw upon, including those by Simon [61], Wallas [68] and the four-part model of the Design Council [14]. We present a comparison of these models in Fig. 2. In fact, there is remarkable similarly between these and instructional models such as ADDIE [8] (analysis, design, development, implementation and evaluation) and Jonassen's eight stages [30]. The instructional styles of Merrill [40] and Kolb's experiential learning cycle [33] also have relevance here. Our framework draws upon these models, especially Kolb's [32] experiential learning cycle. We integrate *thinking*, with *modifying* (through improvements from formative feedback). *Integrating*, *experiencing* and *practising* the ideas through sketching, implementing, and *evaluating* the work. Our approach takes the student through the entirety of Bloom's [35] taxonomy, from knowledge to synthesis and evaluation. Moreover, the data-visualization community have likewise created several models that help developers consider and build visualization tools. Models such as Munzner's nested model [42], McKenna et al. [39] (understand, ideate, make, deploy) and the nine-stage design-study model by Sedlmair et al. [56] could be used in teaching situation. However our three-stage framework is pragmatically designed to conform with a module's (unit of teaching) duration, be easily administered by a teacher and be understandable by a student. Our framework also provides frequent opportunities for the teacher to give formative and summative feedback, and grading.

Explanatory visualizations have proven useful to teach various concepts, such as mathematics [1, 15], algorithms and data-structures [19, 44, 62]. In fact, a simple Internet search of "visualization of sorting algorithms" returns a large quantity of animation videos and interactive tools that can be used by a learner. Algorithm visualization is relevant, because we are asking the students to create their own explanation of an algorithm. Most of the seminal work was done in the late 1990s and early 2000s. The surveys by Urquiza and Velázquez-Iturbide [66] and Shaffer et al. [59] provide a useful account of the early work, including work by Brown on Balsa, and tools such as Tango and Polka. Fouh et al. [19] and Sorva et al. [62] investigate how visualizations help with programming education. Unfortunately most of these explanatory visualizations are created by the educator in advance of the lesson, and thus are merely viewed passively by the learner. Certainly, there are some examples whereby the student can interact and explore the tool by changing different parameters. Yet, even when this happens the workload, especially the creative imagination and its design, resides with the educator rather than the student. In contrast, our approach is to get the students to be be active in their learning by creating their own visual explanations in code, and thus understanding the algorithm at hand in greater depth.

But, are these animation systems effective? One of the first evaluations on the matter was done by Hundhausen et al. [25], followed by another study by Grissom et al. [20], with mixed results. While some users did benefit from the explanatory animations, others seemed hindered by them. Reviewing the past two decades of algorithmic visualization, Fouh et al. [19] provide a useful reflection, saying that the issue is not the algorithmic visualization per se rather it is how students use it, and that it "*is effective when the technology is used to actively engage students in the process*" [19]. This too is one of our motivations; if we can improve student engagement with the material then we will be able to develop the skills of more students. Grissom et al. [20] write "*The true value of using visualizations may lie not in their content but rather in their serving as a motivational factor to make students work harder*". This is exactly what we are doing. The students make a choice of algorithms to follow, they design their own solutions and then build them. Because they are choosing their own project to follow and setting their own goals, they are more motivated [31], have more freedom, they feel in control of their learning. The structure gives clear grading points, and helps them understand their own learning pathway.

Within the past (approximately) ten years, computing education has taken another step change. There are now several environments that provide a sandbox for students to learn programming. For example, Greenfoot [34] employs animation as an integral part of teaching computer programming skills (and is now used throughout Wales, UK to teach 11-19 years). JavaScript and Python animation libraries have been used to provide animation interfaces for learning coding principles. Furthermore, there is a rise of online interactive development environments used throughout education. However, our goal is not to teach programming skills, but to develop higher level design and critical-thinking skills within computer scientists. This is a similar goal to Sengupta et al. [58] who try to teach broader skills such as abstraction, encapsulation and complexity, and Hsieh and Cifuentes [23] who compare learning between paper-based and computer-generated visualizations.

In other related work, VisitCards [21] provides a way to explore the design space with users, while the Five Design-Sheet method (FdS) [49, 50] enables users to explore alternative design solutions. We use the FdS in this framework for the design stage. Apart from educational algorithmic visualizations only a few other examples of explanatory visualizations exist. E.g., Weiskopf et al. [70] developed an explanatory visualization of general relativity, Natali et al. [45] focused on the communication of geological concepts, while Tufte [65] and Lipsa et al. [37] look to more general visualizations in other sciences.

5 Framework Overview

We define six components: *research*, *report*, *design alternatives*, *plan*, *develop* and *reflect*, that are split into three stages. When a module or unit of work is first created, the teacher needs to devise its learning outcomes, aims for each part of learning and develop the learning

content. Such to address this point, and to enable teachers to use material from this article, we write these in language that could be given to the student to explain what they will be expected to do.

Our aim is to develop skills in learners that correspond to the higher levels of Bloom's taxonomy [48], especially to enhance creative and critical reflectionabilities[1]. At the end of our class we expect students to demonstrate knowledge of a chosen algorithm, be able to analyse the problem and critically evaluate the solution. We also expect students to be able to think through different designs, plan how they will be implemented, and be able to prepare technical reports and communicate their ideas effectively.

We get every student to choose a different algorithm to study. Each student then develops their own explanatory visualization that will be different to his/her peer. This a very important point within the framework, because it enables every student to have a different problem to tackle and therefore allows students to discuss their creative ideas with each other without any issues of plagiarism. We need the students to be comfortable to share and discuss ideas. This is unusual in an academic situation, because most of the times every student is developing similar code and working on the same problem, whereas the students are typically not encouraged to share ideas. However, we want them to share their ideas, concerns, discuss their implementation strategies and peer review each others work.

The framework addresses eight learning outcomes, outlined in Fig. 3. We also indicate where each outcome is measured, within the six-component framework. The six components are grouped in three conceptual stages. Each stage comprises of **traditional lectures** (denoted TLs), **lab-based activities assessment activities**, and **self-study** activities. In Fig. 3 we present a schematic of the whole framework, and articulate each of the three stages in Sect. 6,7 and 8.

5.1 Learning Outcomes

There are various learning outcomes that result from EVF. The learners need to develop and demonstrate deep understanding of their chosen algorithm, in order to be able to explain it sufficiently (LO1). They are directed to look into background work an pick out the information they believe is significant and will contribute to their explanation (LO2 and LO3). They need to devise effective ways to describe the algorithm and its execution over time (LO2, LO4 and LO6). They use sketching to create alternative designs of their algorithm depiction and discuss them with their peers and tutor (LO4, LO5 and LO7). This works as a form of formative assessment where they begin to assess whether their solution has the desired outcome (LO3). They implement their preferred solution (LO6) and write a report about it (LO5), reflecting on the quality of their work, its efficiency and potentially alternative strategies (LO3). All of the above need to be done with a creative and innovative attitude, which to a large extend needs to be established by both the learner and the tutor.

In particular, we believe that reflection is an important skill, necessary throughout the whole design and implementation process. Reflection traditionally comes at the end of a computing project, and certainly for individual computing-projects most academics would expect the students to reflect on their work and achievements in the last chapter, or section. This type of reflection is definitely important and helps the student to consider successes and failings. Yet, reflection must be taking place throughout the whole process. We (as educators) want the learners to consider the decisions as they make them, and when they make them. Learners must contemplate the decisions they are making, to evaluate and reflect upon the quality of their work as they do it, and make judgements as they move from the outline research to the final build. What we are asking is that learners perform *metacognition* [17], such to develop critical thinking skills. Metacognition is "thinking about thinking". It encourages the student to think why they are making a decision, and what would happen if they make that decision. Facione [16] writes about this idea, and his statement gives us the ultimate set of traits that we would like to see in every student. He writes, "*The ideal critical thinker is habitually inquisitive, well-informed, ... willing to reconsider, clear about issues, orderly in complex matters, diligent in seeking relevant information, reasonable in the selection of criteria, ... and persistent in seeking results, which are as precise as the subject and the circumstances of inquiry permit*" [16].

[1]Bloom's taxonomy defines six levels of action verbs; defining skills that learners should grasp from *knowledge*, *comprehension*, *application*, *analysis*, *synthesis* to *evaluation*.

5.2 Aims of the components

It is important that the students know what they will be doing from the outset of the teaching process. **Background** information needs to be provided, such to explain the framework and importantly, what is expected from the students. Managing student expectations is also important, and the students need to realise that they will be developing their own explanatory visualizations, and that they are encouraged to share ideas and concepts. Beyond this background stage, and after the students have chosen an algorithm to depict, the six components of the EVF are as follows:

Research. The aim is to get the student to develop a deep understanding of the algorithm. Reference material should be investigated, including books, papers, videos and other resources to: i) learn about the topic, and (ii) learn more about explanatory visualizations and see if other people have done any for the chosen algorithm.

Report. The aim is to write a clear, concise document that communicates effectively the information from different perspectives and summarises salient information. The concise, two-page (approximately 1000 word) report, is structured by the following sections: summary, history, algorithms, maths, diagram, application, similar-to and references.

Design alternatives. The aim is to analyse the algorithm, be creative and design sketches of different potential solutions. We use the Five Design-Sheet [50] method that starts with many initial ideas, refines them down to three, and eventually one (realisation) design, which will be implemented.

Plan. The aim is to develop an appropriate solution, ascertain how to tell the story, and work out the main (key) frames of the story. The code development is planned and divided into manageable chunks (e.g., apply the use of UML or design patterns to achieve the goal).

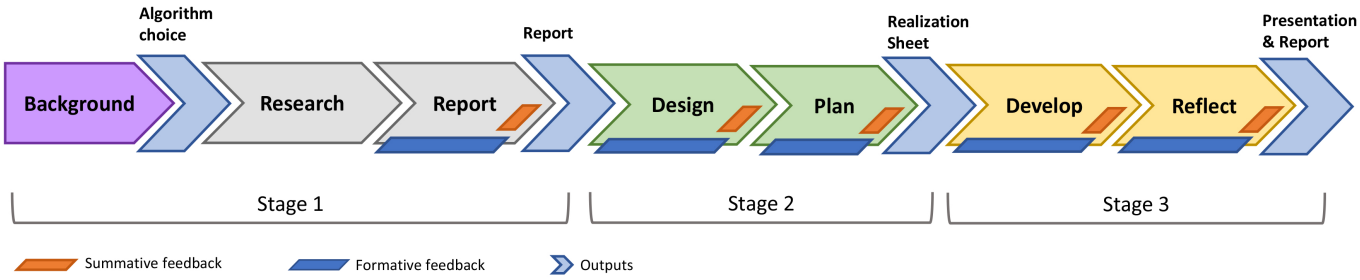
Develop. At this stage, the goal is to code an effective explanatory visualization solution, i.e., not only to consider the syntax of the code, or the semantics of the processes but also consider the design of the depiction, how it is designed to be effective, its aesthetics and whether it appropriately conveys the story.

Reflect. The aim of the final stage is to perform a critical analysis of the work, and to judge whether it is fit for purpose, i.e, to criticise, evaluate and assess the work achieved. This is done by creating a 2000 word report. (Note, when considering Bloom's verbs [35], most lists use the term *evaluate* as the main category, with minor verbs such as judge, critique, reflect, rate, etc. We choose to use the term *reflect*, so not to confuse with evaluation and usability of human-computer systems.)

5.3 EVF in our teaching

We have used our strategy as the learning mechanism for final-year undergraduate major students who are taking a Computer Graphics and Visualization module. The participating students already have some software engineering skills, and they are fluent in Java, know JavaScript, Python and C++. The module runs throughout the academic year and the whole assessment makes 6 ECTS (European Credit Transfer and Accumulation System). In Europe one academic year corresponds to 60 ECTS credits and is equivalent to 1500–1800 hours of total workload.

The module content comprises of a series of foundational traditional lectures (see Sect. 6) and lab sessions. For our use, the students receive an one hour lecture, with a two-hour laboratory session, per week. The laboratory sessions are used for several purposes, including setting exercises, giving one-to-one formative feedback, self study, peer discussions, and formative and summative assessment. The students self-study is supported with one-to-one meetings to provide personal guidance and feedback. Students need to work on the project in their own time, as well as in the lab sessions, and they are also encouraged to ask feedback and make discussions with their peers.



	Domain	Definition	Research	Report	Design	Plan	Develop	Reflect
			Where measured					
LO1	Knowledge	Demonstrate deep knowledge and understanding of the algorithm.	✓	✓				✓
LO2	Skills, cognitive	Analyse a problem and define appropriate computing solutions.			✓			
LO3	cognitive	Reflect critically, analyse if the solution meets the criteria.			✓			✓
LO4	practical	Plan, sketch and consider alternatives of design.			✓			
LO5	practical	Prepare technical reports and deliver presentations.		✓				✓
LO6	practical	Develop and deploy (explanatory visualization) software.				✓	✓	
LO7	interpersonal	Communicate effectively.		✓				✓
LO8	Attitude	Be creative and innovative in developing computing solutions.			✓		✓	
		ECTS credits		1	½	½	3	1

Fig. 3. We divide the framework into three stages. Stage 1 covers the research and reporting. Students choose and learn about the algorithm and present it. Stage 2 includes design and planning, whereby students learn about ideation and sketching, and then practice these skills before designing and making plans of what they are to build. Finally, in stage 3 students develop the explanatory visualization and make a critical reflection of their work, and demonstrate what they have done. *Knowledge*, *skill* and *attitude* related learning outcomes are addressed in our framework. They are measured at different stages (1) Research, (2) Report, (3) Design, (4) Plan, (5) Develop and (6) Reflect. To provide an idea of workload, and to allow study effort to be compared, we include values for the European Credit Transfer and Accumulation System (ECTS). One academic year corresponds to 60 ECTS credits.

6 Stage 1: Establishing Fundamental Knowledge

The principal goal of Stage 1 is to increase and assess knowledge (see Fig. 3). Learners should develop deep understanding of the algorithm (or artefact they are explaining) and how it is used. The learners will need to perform research, take notes and then write up the discovered knowledge in a well structured, well written report. The following subsections explain the components of stage 1 in more detail.

6.1 Lectures

Introductory lectures are used to explain the whole process and also to present an overview of all the algorithms they could choose. By explaining the process, the student can better understand the journey that they are about to embark and they have knowledge of what they are to expect. By providing a summary of every algorithm they can make an informed decision of which algorithm to choose.

TL1: Introduction. The first lecture provides a description of the complete framework, its learning objectives and the teaching activities, including details of deliverables and marking scheme. They need to consider an end-user. We get the students to develop their explanatory visualizations for an imaginary second-year computer-science undergraduate student. Students can empathise with this situation, e.g., one student said *"I wish I had these explanatory visualizations when I was a second year student"*. While there is benefit to using real users [12], it is not always practical or feasible, particularly in a learning environment. We want students to learn how to make decisions on their own designs, rather than being potentially directed by client goals, or develop user/client engagement skills.

TL2: Research techniques and strategies. Students are taught how to search, identify and summarise related previous work. This lecture complements other modules which include more in-depth teaching of research strategies.

TL3: Presentation of available algorithms. We briefly explain the available algorithms, using a "chalk and talk" session and sketches on the wipe-board. The idea is to provide a summary of each techniques, rather than detail information for each algorithm. In our case, the students would have been taught about half of the algorithms in the previous year, so some explanations act as revision. We explain all new algorithms in greater detail.

TL4: Explanatory Visualization techniques. The focus of this session is to explain visualization techniques in the context of Explanatory Visualizations. We include details of graphical marks, symbols and Gestalt laws of similarity, proximity and synchrony. We refer students to books, including those by Ware [69], Munzner [43] and Roberts et al. [50]. Particular attention is given to animation techniques which are especially suited for Explanatory Visualizations. The material covers the basic types of animation (flipbook, keyframe and procedural) and the twelve principles of animation [36] (staging, solid drawing, slow-in/out, anticipation, timing, arc, pose, appeal, follow-through, secondary action, squash and stretch, exaggeration). Chevalier et al. [13] provide a useful summary of uses of animation.

6.2 Practical Activities

In the first laboratory session each student chooses a different algorithm or technique. To achieve this task practically, we used a shared/editable document which all students can access. They write their name against the algorithm they wish to study and can instantly see if an algorithm has been chosen. We have a list of over 80 algorithms, including: Anti-aliasing, Colour spaces (RGB, CLS), Colour-mapping, Transfer function, Coordinated views. Direct and Indirect manipulation, Direct volume rendering (gel-like), Dividing cubes, etc.

Next, the students research their chosen algorithm. They take notes and receive *formative feedback* on what they have discovered and suggestions where to look and what to learn. The aim of these one-to-one sessions is to allow students identify any strengths and weaknesses, and to encourage them in their research. Furthermore, it gives the tutor an idea of the progress of all students. In the formative feedback session the tutor needs to perform two tasks: First, to acknowledge successes of the students' work. It is important that aspects of quality, good work, correctness are described. Second, the work should be guided forward; from asking questions to identifying aspects to correct. Imperatively, formative feedback is positive, constructive, encouraging, confidential, and evidenced by the work presented.

6.3 Assessment

We have observed that writing is not easy for computing students. Learners within a technical field often find it difficult to locate relevant material, struggle over the actual process of writing (because it is not so familiar with them) and may be less-motivated to do well. But, while the output of this stage is merely a report, it is an important part of the

process. The student needs to understand the algorithm in great detail in order to be able to discuss it in depth. Therefore, the very act of writing a summary report helps to gel the ideas in their mind [18]. In this part, we are engaging the learner in *directed thinking*: the very act of writing something down on the page requires deep thought, creative thinking and organisation of the ideas. Through the act of writing, the learner is engaging in creative thinking and a good/well-written report requires that a student has deeply reflected on their work.

For this report, we define a specific structure. This acts as a prompt to lead the students to find specific information and look at many sources, helps the students structure the report, and helps tutors grade it. We encourage learners to consider each of these sections as stories that cover the same topic but are told from different perspectives; a historical perspective, mathematical perspective, as follows:

Title of algorithm and author name
Summary of the algorithm or technique,.
History or who invented it and for what purpose, etc.
Algorithm description/pseudo code.
Maths including equations, algorithm speed, efficiency etc.
Diagram or schematic of the process.
Application describing how and where it is used, extensions, etc.
Similar to other work or interchangeable.
References

7 Stage 2: Design and Plan

The aim of the second stage is to investigate alternative design ideas and to increase creative skills. After some intoductory lectures there are practical activities to upskill creativity, including exercises to practice sketching skills, explanation, demonstration and practice of the Five Design-Sheets method [49], and storyboarding techniques.

7.1 Lectures

For Stage 2 there is some foundational material that need to be presented to the students as lectures or seminars. Apart from encouraging the students to put pen to paper and design their solutions, we give lectures on creative thinking **(TL5)** and the FdS methodology **(TL6)**. There are several resources that cover creative thinking and where ideas come from. We adapt the list in Roberts et al. [49] and ideas from Johnson [29]. In summary, we cover ideas of (i) convergent and divergent thinking, (ii) reflection of the ideas, (iii) collaboration and asking for opinions (and this is why we insist that everyone follows their own algorithm, such that students can share ideas). (iv) inspiration from nature, biomimicry and bioinspiration; (v) inspiration from the environment and workplace (objects on your desk, pictures on the wall, etc.); (vi) inspiration from related work and (vii) serendipity of ideas coming from mistakes.

7.2 Practical Activities

One of the main challenges in developing creative skills is low self-esteem and low-self belief. This may exist due to several reasons: perhaps the student had a bad experience in an art class at lower School, where they were told they could not draw, which makes them reluctant to sketch and make drawings. Or that they are their worst critic and claim their own drawings as being inadequate, and so will not draw anything because they feel that they are not good enough. Other students, who traditionally excel at programming, do less well in the creative activities. Indeed, some computing students say "we code, we do not design".

Having the right attitude is important. If the student comes into the class with a negative mindset, then they will not perform well. They need to relax and forget about these past criticisms. Robinson [54] explains that students do better and want to learn more when they are "in their element", when they are happy and doing something they enjoy. But likewise, teachers should have the right attitude [5]. The teacher can easily create a bad climate, by being negative about design and creativity skills, showing a dislike of the topic "you'll hate this activity, but you need to do it", distrusts the students, or assumes they will do poorly whatever happens. Teachers need to convey a positive attitude, be encouraging, and expect that everyone will do well. Deep learning is not only difficult and time consuming for the learner, but the tutor also needs to invest time and effort into the activities. In order to mitigate these problems and try to pre-empt some of the challenges, we emphasise and encourage the following:

1. **Sketching and not drawing**. This distinction is important. It is possible to make the argument that drawings are works of art, whereas sketches are artefacts that are merely fit for purpose, and that purpose is to communicate. Students are not being judged on their creative ability, rather the process they follow and the ideas that are created.
2. **Improve your skills through practice**. Sketching is a skill and therefore it can be learnt. Practising different exercises and merely sketching more will improve students' confidence and ability.
3. **Persist in creativity**. Shneiderman, reflecting on creative models wrote: "*..creative work starts with problem formulation and ends with evaluation plus refinement. They acknowledge the balance of 1% inspiration and 99% perspiration – a flash of insight followed by much hard work to produce a practical result" [60].*
4. **Enjoy the experience**. If the students enjoy the experience then they are more likely to do better, and develop better creative ideas.
5. **Be curious**. Curiosity can be a motivating factor and if the students are curious and want to learn, then they will make effort to self study.

Another challenge, that can limit the students' capacity in creativity, is knowing where to start. Given a blank sheet of paper, it can be daunting for a student to know what to do first. Indeed, the Five Design-Sheet structure [49] does help with this issue. However, for the first design sheet we suggest to sketch what they know. They know about the algorithm, its input parameters and what the output looks like.

7.2.1 Practising sketching and example exercises

We use several exercises to help develop confidence in sketching. We encourage students to use large sheets of A3 (tabloid or ledger) paper for their sketching. It is much easier to sketch and be creative on a bigger sheet of paper. We also encourage the use of black drawing-pens to create the outlines, along with coloured pencils or some felt-tip pens to add colour to the sketches. Again while we do encourage colour to be used it should be used sparingly. We use the following exercises:

Students draw straight lines. Horizontal, vertical and diagonal lines are drawn on the page. The task is to get them as close together as possible, but also sketched as quickly as possible. While this is a simple exercise it helps build confidence in sketching.

Students perform a continuous line-study. The aim of this exercise is to look at something (such as a plant) and sketch the result without looking at the page or the pen. This starts to build confidence in creativity and commitment to the page. If this is done in a group, the students can show their sketches, and it creates a positive atmosphere.

Tutor gives live demonstration of the Five Design-Sheets (FdS) [50] We have used several different scenarios, including designing a heritage-capture application and a smart-phone puzzle-game. For this walk-through of the methodology, we keep it separate to the specific problem to hand, such that the students do not take this live-demonstration as the result that they need to copy.

Students do their own FdS. Individually they make their own sketches. We do this activity in a large room, where many students do it at the same time, this allows students to share experiences and ideas.

7.2.2 Design alternatives, the FdS method

This aim of this part is to look at the problem through different perspectives, and to investigate alternative design solutions, which are refined to create a final realisation design. Often students struggle with creating different ideas. The reason we use the FdS is that it provides a structure. It leads the students on a journey, from less-formed creative ideas to a developed plan that they can build. By the end of the FdS task, they will have intensely thought about their design algorithm, explored alternative design ideas, and thought deeply about one (potential) solution. The method gets students to sketch their designs by hand on five sheets of paper. For more information on the method see Roberts et al. [49, 50].

We acknowledge that other creative design methods could be used, such as less formal sketching methods [11] but, in our experience, students who use less formal approaches struggle to explore the expanse of potential alternative possibilities. Token and constructive based tasks using physical objects [26] are useful for understanding concepts, but are less suitable for our purpose of creating graphical exploratory visualizations. Wireframe drawing software could be used, but again it is far more difficult to structure the design process, explore design alternatives and students can get distracted by getting to grips with the software rather than generating alternative design ideas.

7.2.3 Planning using storyboarding

The aim of this part is to get the students to plan and think how they move from their designs into something that they can build. First they need to confirm the story that they are telling. For that, we get students to sketch a storyboard. Second they need to start planning the implementation and ascertain how they are to code the solution.

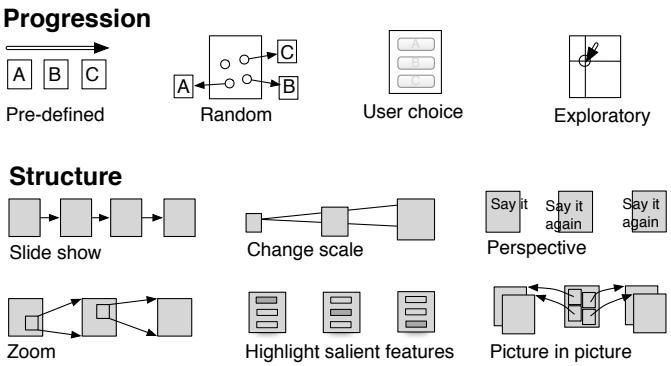


Fig. 4. Several different explanatory styles exist. First, the overarching progression can be pre-defined, random, chosen by the user or exploratory. Different structures can be used to control the frames.

While there is less work on explanatory visualizations, several researchers have investigated narrative styles in data-visualization [24,57] and we refer the students to this body of work. We deliver a lecture on explanatory visualization covering *progression* and overarching *structures*, (see Fig. 4). The progression of the story can be *predefined* and linear, random, or the user can *choose the path* of the learning. Hybrid approaches exist that mix two styles together. Indeed, several students choose to use a hybrid approach, where the user is led through a pre-defined story at the start (for about 1 minute). Then, afterwards, the mode changes, and the user can interact with the tool to explore specific values (e.g., see Fig. 6).

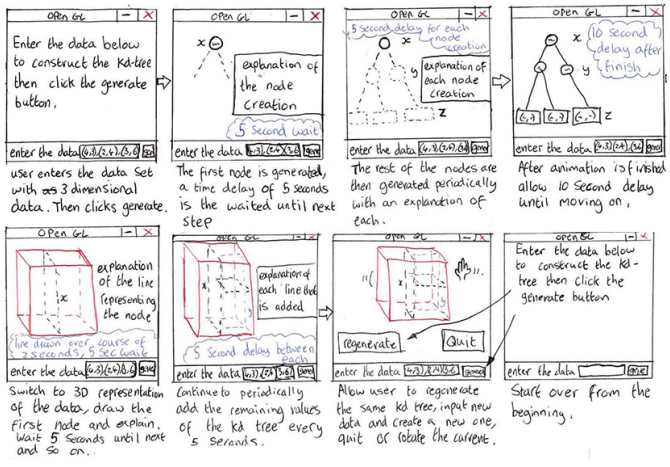


Fig. 5. Storyboard of a kd-tree explanatory visualization, showing how the student has thought through the key frames to tell the story.

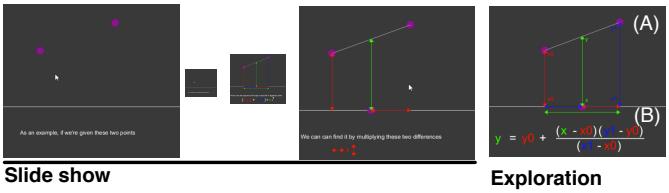


Fig. 6. Explanatory visualization of linear interpolation. The explanation has two parts: a pre-determined animation sequence (slide show), followed by an exploratory mode where users can select and translate any circle in part (A) to interactively change the values displayed in (B).

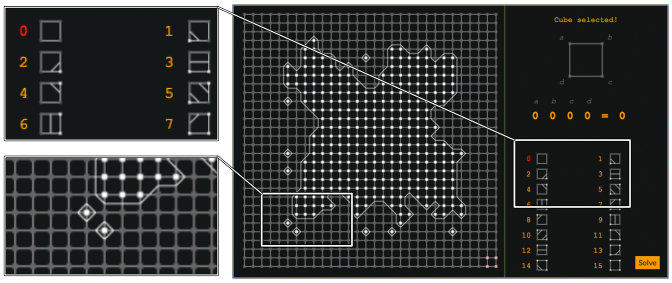


Fig. 7. Explanatory visualization of the Marching Squares algorithm by a student, developed in WebGL, showing the final contoured data. The animation automatically progresses through four stages (load or generate data, threshold, march and lookup index). At each stage the animation pauses to allow the user to interact and explore values.

Sketching a storyboard enables the student to think about the key events in their explanatory story. It not only makes a plan that they can follow, but it also demonstrates if they understand the story that they need to explain. For example, we can imagine the Marching Cubes algorithm [38] has a few key stages: (i) to setup all the variables required, and to create the lookup table, (ii) load the data, (iii) threshold the data, (iv) for each cell calculate the index and look it up in the lookup-table, (v) retrieve the normalised geometry and interpolate exact intersections, (vi) calculate normal, colours and (vii) render the triangles. This type of algorithm could be described in a slide-show style. Fig. 5 shows a student's storyboard of a KD-Tree visualization. The student has considered the opening screen, how to generate or load some data, the two-dimensional tree representation and a three-dimensional representation. Two other students used similar slide-show structures, first to explain linear interpolation (Fig. 6), and second the Marching Squares algorithm (Fig. 7). They first introduced the concepts using a pre-defined set of animations, before breaking into an exploratory mode, where the user can explore different values.

7.3 Assessment

The students develop their FdS sheets over the duration of a few weeks. This allows the teacher to give formative feedback on the sheets (the blue parallelogram in Fig. 3). Students may need to re-do their sketches to improve them and go through a few iterations before they eventually submit the work for grading. Because we use an electronic course management system, we get the students to scan of their sheets into a PDF and upload it to the submission portal. To grade the FdS, we allocate four marks for every panel, per sheet. This gives 20% for each sheet (100% for five sheets). The grader should focus on the process rather than how good the ideas are. However, the students need to have something buildable, at least, on sheet 5 (the realisation sheet). They receive a grade and written feedback for their FdS. We allow students to resubmit their sheets if they wish to do so. However, every time they resubmit they are deducted 10% of that grade. This allows students to improve their designs (in addition to any previous formative feedback) and recover from failure. We view this re-submission policy as a safety net. In our experience only 5% of students took this option, yet 100% of the students re-submitting did get better results. The higher graded students did not view the re-submission as time well-spent, as their benefit was small. Nonetheless, this approach helps to improve the grades of some students with lower grades.

8 Stage 3: Develop, Reflect and Present

The aim of the final stage is to develop an implementation of the final design from sheet 5 and the organisation of the story from the storyboard sketches. Following that, the students reflect on their work, make a presentation and demonstrate their explanatory visualization.

8.1 Lectures

As we are focusing on final-year undergraduate students, the students should know how to program. Therefore, our primary goal is not to develop implementation skills but to guide the students to build informative explanatory visualizations of high usefulness, expressiveness and informational strength. In addition, we want students to focus on their own design ideas, rather than use other peoples' creativity. We ask the students to implement their solution in Processing.org or OpenGL. We choose Processing and OpenGL because academics in the school have long experience with using them for visualization activities, whereas the students receive training in these libraries from a number of modules, including the one discussed in this paper. To supplement their knowledge of OpenGL we give some lectures on the Processing library. We acknowledge that there are many excellent visualization libraries and toolkits, such as D3.js, VTK, Tableau, Qlik or even some of the visualization aspects of Excel. However, we want the students to develop their own ideas, rather than being influenced by pre-defined visualization tools.

8.2 Practical Activities

Students develop their solutions over ten weeks. The timeframe is pragmatic, and chosen to fit in with other assessment activities and within a semester (in the UK). It is long-enough, for the students to develop something substantial, yet short enough to keep them focused. Some students may say "we design, we don't code". These students may need more tutoring, but they too should be able to develop an appropriate solution in Processing. Students may not be able to implement everything they have designed. In fact, we encourage iterative code development; starting with basic functionality and improving it over time. They should continually reflect on their development and how it fulfils the requirement. If it does not meet the goals then they need to appropriately adapt their implementation. It is fine if their solution migrates from their FdS design; we do not compare their final deliverable to their original design (each stage is individually graded), but it is important that they discuss how their implementation has changed from their design in their critical reflective report. They receive weekly formative feedback (like the other stages). Regular tutor contact helps to keep the students on track, and for most students these meetings are quick, where they give a demonstration and receive verbal feedback. Students are also encouraged to self-reflect and ask their peers for feedback and suggestions. To prompt them in this task, we get them to think through the following questions: *Does your explanatory visualization explain the main parts and have enough detail for someone to understand the algorithm? Is it clear and does it explain the process and state changes? Would a second year computer scientist understand the algorithm from your visualization? Is it well designed? Does it use good graphics and animation principles? Could it be published, or is it good-enough quality to show to second year students?*

8.3 Assessment

Students give presentations of their explanatory visualizations to the teacher and write a critical reflective report of their work and achievements. Their presentation and report are both graded. Reflection on what has been achieved is extremely important. A significant stage of Kolb's experiential learning cycle is to perform reflective observation [32]. When learners understand what they have achieved, and how they could have done it better, then they will perform at a higher level next time. We note that students often find it difficult to perform a critical reflection of their own work. They can fall into a trap of either being too critical and negative, or say that everything is perfect. The ideal approach is when students provide a balanced reflection of their explanatory visualization. We ask the students to structure their report into three sections: (1) work done, achievements, description of successes and brief discussion of challenges, (2) description of how their result explains the algorithm and related concepts, including screenshots of their work, and (3) a well-argued critical reflection on their work, discussing what they have done well and what they would do differently if they had the opportunity.

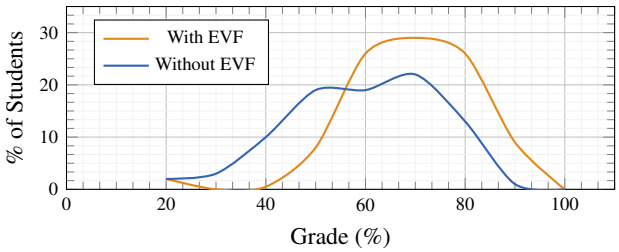


Fig. 8. Cohort statistics, indicating a clear shift in average grades and a reduction of work under the pass threshold (40%).

9 Evaluation & Discussion of the EVF

To evaluate the success of the EVF we performed a usability study, and looked at a series of indicators on how students achieved the learning outcomes, in relation to *knowledge*, *skill* and *attitude* (see Fig .3). We discuss our observations, through a series of investigative questions.

Our cohorts include students with a wide range of abilities, ranging from academically strong students, who are driven by natural curiosity, to students who are attending, simply to obtain a qualification. Often, underachieving students believe that academic assignments are not relevant to life, and are thus less likely to engage. Problem Based Learning naturally makes the task relevant [4]. This is demonstrated by analysing students grades. When we compare students' averages from two years of using the EVF, with the average of students grades of three years without (see Fig. 8) we see that there are less failing students, and the overall Normal distribution has shifted higher. There are certainly many factors involved; nonetheless it is a positive indicator of the use of the EVF. We believe three aspects were influential: (1) the use of PBL and the explanatory task, has helped engage more students. Students can empathise with the task, see relevance to their life, and that they could use the project as an example of work undertaken in the CV. Such realisations motivate students to perform well. (2) The experiential style has meant that students developed and improved; they began to realise that they can try ideas, and recover when these do not come to fruition. Finally, (3) The tutor/student meetings has enabled students to ask for advice, guidance, and has helped students stay on track.

Is the EVF perceived as being usable? We evaluated the EVF using a modified System Usability Scale (SUS) questionnaire [9]. We modified the SUS, using the method explained by Roberts et al. [49], by replacing the word "system" with "framework" and "technical person" with "tutor". We chose the modified SUS because it short, simple to administer, easy to understand, suited for small sample sizes and systematically used to evaluate different systems in terms of their usability [2]. 17 undergraduate students completed the questionnaire. The reliability of the modified scale was measured high, with a Cronbach's alpha of (α=0.786). We calculated an average SUS score of 70.53 indicating that the framework was perceived to have good usability, by means of the scale determined by Bangor et al. [2].

What do students think is a positive aspect of the EVF? Regarding the positive aspects of the EVF, most students commented favourably about the overall process and the planning of the task. For example, one student wrote *"It's a good way of planning out a project because ideas can be improved as you progress through the framework"*, adding *"encourages trying different ideas"* and *"creates a clear path for achieving the task"*. Many students talked positively about creativity, e.g., *"It opens up your creativity and ideas"* and *"Can see the development through multiple stages allowing you to think and reflect at each"*. In terms of how EVF facilitated their learning process the students said: *"It helped me plan out a project well and to keep developing and improving my idea from the very start all the way to*

the end". They also reported, *"I arrived at a stronger idea at the end of it than I likely would have otherwise"* and that it helped *"structure my design efforts that are usually unstructured and vague, at least initially"*. Finally, one student discussed how reflecting on their work helped them, stating *"it does achieve what it sets out to do, it does allow for self-reflection and improvement of the work being done"*.

What do students think is a negative aspect of the EVF? Looking at the negative comments, there were two kinds of responses. First, a few students focused on the process, stating *"it took a lot of time to figure out what exactly was expected of me to complete each activity"* and that *"some people may not like these set design methods"*. Such comments highlight the need to spend time at the start of the module, talking about learning objectives, expectations and ensuring that the students understand the whole process. Secondly, negative replies also focused on the need to create alternative designs. One student wrote: *"[it is] tempting to just concentrate on one idea from the beginning"* and *"I tend to prefer not to storyboard and just go with it"*. Responding specifically on the Five Design-Sheets method, one student wrote: *"I wasn't sure about some parts of the FdS – how the first sheet worked, but got there in the end"*. As co-authors of the Five Design-Sheet method, we understand fully the challenges and benefits of using sketching and creative design in academic teaching. We have had personal experience of computing students saying bluntly "we code, we don't design", a concern acknowledged by others [28]. As teachers we need to continue with a positive attitude and try to explain the benefits of being creative in design, not only for innovation and product design, but for every-day code development.

Have students increased their knowledge? As tutors we saw students start by knowing very little, and left knowing substantial knowledge of the algorithm. We look look to several indicators as evidence that the students extended their knowledge. (Teachers can use these indicators to help them grade). (1) The wording in their report (stage 1) substantiates that they looked at many sources. They cited book, journals and conference papers, and made reference to online videos. For instance, if students only looked at, say, Wikipedia it would be one of their references, and they may not include other citations. While we acknowledge the benefit of Wikipedia to learn the basics about the topic, validity of information is often questionable. In our case only 6 of 71 students cited Wikipedia as a referenced source, indicating that students did look more widely. (2) Citations can also act as a proxy for judicious behaviour. We investigated the quality and type of references. Reference quantity could be another indicator, but since we ask students to create a short report quantity is not a suitable metric. As authors we graded the references as being excellent (full detail), good, and poor (e.g., with missing information or only with URLs). 80% of the citations were rated as being good or suitable. (3) From our experience students who do not care about the assessment often plagiarise (more than 25% of the material). We analysed copying within the reports using the *turnitin.com* plagiarism detector and found our students score between 0% and 20% copied material. This figure includes copied quotations, citations, equations and pseudo code – which are generally acceptable. The low plaigiarism score indicates that students wrote the text in their own words.

Have students developed their skills? Our framework is designed to challenge learners to take ownership of a problem and explore a divergent range of creative solutions. All students results implicitly demonstrate that students have implementation skills, e.g., see Fig. 1, Fig. 6 and Fig. 7. Counting designs on sheet 1 of the FdS is one way to evaluate creativity. At the start of the process it was evident that about half of the computing students were less convinced by sketching, and they were only creating a few different ideas. Nonetheless, by the time they submitted their FdS sketches they had created, on average, ten different ideas (with a range 4 to 23). For example, Fig. 9 shows twelve ideas created by a student investigating anti-aliasing.

Have student attitudes changed by following the EVF? We gain insight into students' attitude from their critical reports (stage 3), where they reflected over the whole process. Interestingly, the students achieving the top 80% of the grades used verbs from the higher levels of Blooms' taxonomy [35]; phrases like *"I learnt"*, *"I ensured that the information was accurate"*, *"I made decisions of how to choose the right papers to cite"*. E.g., one student wrote *"The biggest thing I learned was that the algorithm was such a broad topic* [...] *and did not consist of only one algorithm, but is bundle of them* [...] *accomplishing the same task"*. However, when we look at the lower 20% students we see students reflecting mainly on bugs in their code, rather than making a holistic reflection.

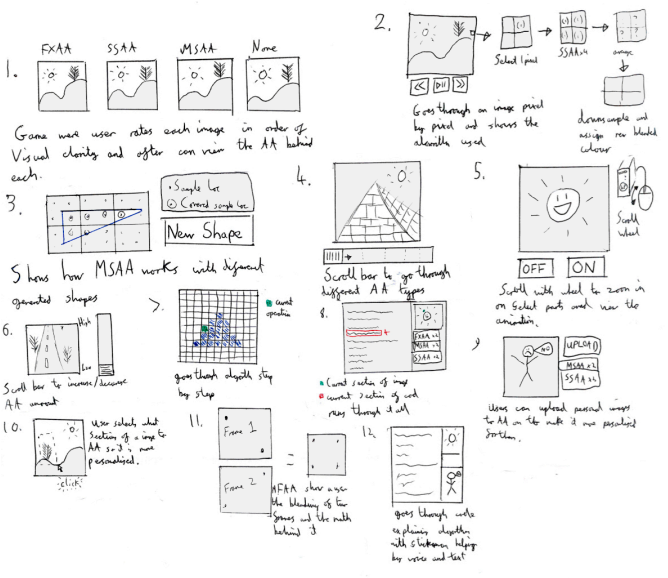


Fig. 9. Student sketch (on their FdS sheet 1) showing 12 different ideas.

While the EVF approach provides significant flexibility it also requires buy-in from both the students and the educator. The students need to change their mindset. They need to mentally commit to a new way of thinking and approaching problems. When faced with a creative problem, many students have genuine concerns that they do not know what to do, or what is expected of them. Computing students often lack creative training in their pre-university education, so forcing them to engage with these processes can be daunting. But also the teacher needs to change their mindset. If they are not positive or encouraging then the students will likewise be apathetic and disengage from the teaching. It is the teachers' responsibility to invest time in talking and discussing with students more, providing frequent feedback and guidance.

10 Conclusion

Teaching creative skills in Higher Education has a number of benefits; the most important being that the students become more adept in thinking through problems, considering alternative strategies and ultimately creating more effective solutions. While creativity is implicit throughout disciplines in Higher Education it is rarely discussed or promoted within the teaching. One of the challenges of integrating creative exercises in the curriculum is that (by its nature) creative thinking is an ill-defined question. There is no one answer, which not only makes it more difficult for students (who are typically not used to thinking divergently) to create their assessments, but it implicitly makes it more difficult for the teacher to judge and grade.

The EVF guides a learner to think through the explanatory visualization task, consider alternative solutions and reflect on their design, implementation choices, and actions. The framework is designed to provide a good balance between fostering creative thinking and providing the structured guidance that students need. It also enables teachers to swap-in their favoured exercise or assessment and apply the model to their situation. By creating their own explanatory visualizations, students learn and develop their creative, reflection and communication skills. While visualization techniques are starting to pervade our work and leisure, we feel that there are many opportunities to use explanatory visualizations in teaching and learning.

REFERENCES

[1] A. Arcavi. The role of visual representations in the learning of mathematics. *Educational studies in mathematics*, 52(3):215–241, 2003.

[2] A. Bangor, P. T. Kortum, and J. T. Miller. An empirical evaluation of the system usability scale. *International Journal of Human-Computer Interaction*, 24(6):574–594, 2008. doi: 10.1080/10447310802205776

[3] S. Bell. Project-based learning for the 21st century: Skills for the future. *The Clearing House*, 83(2):39–43, 2010. doi: 10.1080/00098650903505415

[4] J. Biggs. What the student does: Teaching for enhanced learning. *Higher Education Research & Development*, 18(1):57–75, 1999.

[5] J. B. Biggs. *Teaching for quality learning at university: What the student does*. McGraw-Hill Education (UK), 2011.

[6] C. C. Bonwell and J. A. Eison. *Active Learning: Creating Excitement in the Classroom. 1991 ASHE-ERIC Higher Education Reports.* ERIC, 1991.

[7] J. Boy, R. A. Rensink, E. Bertini, and J. D. Fekete. A principled way of assessing visualization literacy. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1963–1972, Dec 2014. doi: 10.1109/TVCG.2014.2346984

[8] R. K. Branson, G. T. Rayner, J. L. Cox, J. P. Furman, and F. King. Interservice procedures for instructional systems development. executive summary and model. Technical report, DTIC Document, 1975.

[9] J. Brooke. SUS – a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[10] F. Brooks. Keynote address: A vision for visualization. In *Proceedings of 4th IEEE Visualization Conference*, p. 2. IEEE, San Jose, California, October 1993.

[11] B. Buxton. *Sketching user experiences: getting the design right and the right design*. Morgan Kaufmann, 2010.

[12] C. Chen and Y. Yu. Empirical studies of information visualization. *International Journal of Human-Computer Studies*, 53(5):851–866, Nov 2000. doi: 10.1006/ijhc.2000.0422

[13] F. Chevalier, N. H. Riche, C. Plaisant, A. Chalbi, and C. Hurter. Animations 25 years later: New roles and opportunities. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '16, pp. 280–287. ACM, New York, NY, USA, 2016. doi: 10.1145/2909132.2909255

[14] D. Council. Eleven lessons: managing design in eleven global companies. desk research report. Design Council, 34 Bow Street, London, 2007. www.designcouncil.org.uk.

[15] M. de Guzman. The role of visualization in the teaching and learning of mathematical analysis. In *International Conference on the Teaching of Mathematics (at the Undergraduate Level) 2002*, July 2002.

[16] P. A. Facione. Critical thinking: A statement of expert consensus for purposes of educational assessment and instruction. research findings and recommendations. *ERIC*, ED315423, 1990.

[17] J. H. Flavell. Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American Psychologist*, 34(10):906, 1979. doi: 10.1037/0003-066X.34.10.906

[18] L. Flower and J. R. Hayes. A cognitive process theory of writing. *College Composition and Communication*, 32(4):365–387, 1981.

[19] E. Fouh, M. Akbar, and C. A. Shaffer. The role of visualization in computer science education. *Computers in the Schools*, 29(1-2):95–117, 2012. doi: 10.1080/07380569.2012.651422

[20] S. Grissom, M. F. McNally, and T. Naps. Algorithm visualization in CS education: Comparing levels of student engagement. In *Proceedings of the 2003 ACM symposium on Software visualization*, pp. 87–94. ACM, 2003. doi: 10.1145/774833.774846

[21] S. He and E. Adar. Vizitcards: A card-based toolkit for infovis design education. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):561–570, Jan 2017. doi: 10.1109/TVCG.2016.2599338

[22] L. Hetland. Connecting creativity to understanding. *Educational Leadership*, 70(5):65–70, 2013.

[23] Y.-C. J. Hsieh and L. Cifuentes. Student-generated visualization as a study strategy for science concept learning. *Journal of Educational Technology & Society*, 9(3):137–148, 2006.

[24] J. Hullman and N. Diakopoulos. Visualization rhetoric: Framing effects in narrative visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2231–2240, Dec 2011. doi: 10.1109/TVCG.2011.255

[25] C. D. Hundhausen, S. A. Douglas, and J. T. Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, 13(3):259–290, 2002. doi: 10.1006/jvlc.2002.0237

[26] S. Huron, S. Carpendale, A. Thudt, A. Tang, and M. Mauerer. Constructive visualization. In *Proceedings of the ACM conference on Designing Interactive Systems (DIS)*, pp. 433–442. ACM, 2014. doi: 10.1145/2598784.2598806

[27] N. Iliinsky and J. Steele. *Designing Data Visualizations*. O'Reilly, September 2011.

[28] N. Jackson, M. Oliver, M. Shaw, and J. Wisdom. *Developing Creativity in Higher Education: An Imaginative Curriculum*. Routledge, 2014.

[29] S. Johnson. *Where good ideas come from: The natural history of innovation*. Penguin UK, 2010.

[30] D. H. Jonassen. Instructional design models for well-structured and ill-structured problem-solving learning outcomes. *Educational Technology Research and Development*, 45(1):65–94, 1997.

[31] J. Klerkx, K. Verbert, and E. Duval. *Enhancing Learning with Visualization Techniques*, pp. 791–807. Springer New York, New York, NY, 2014. doi: 10.1007/978-1-4614-3185-5_64

[32] A. Y. Kolb and D. A. Kolb. Learning styles and learning spaces: Enhancing experiential learning in higher education. *Academy of Management Learning & Education*, 4(2):193–212, 2005.

[33] D. Kolb. *Experiential learning: experience as the source of learning and development*. Prentice Hall, Englewood Cliffs, NJ, 1984.

[34] M. Kölling. The greenfoot programming environment. *ACM Transactions on Computing Education (TOCE)*, 10(4):14, 2010.

[35] D. R. Krathwohl. A revision of bloom's taxonomy: An overview. *Theory Into Practice*, 41(4):212–218, 2002.

[36] J. Lasseter. Principles of traditional animation applied to 3d computer animation. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH 1987, pp. 35–44. ACM, New York, NY, USA, 1987. doi: 10.1145/37401.37407

[37] D. R. Lipsa, R. S. Laramee, S. J. Cox, J. C. Roberts, R. Walker, M. A. Borkin, and H. Pfister. Visualization for the physical sciences. *Computer Graphics Forum*, 31(8):2317–2347, Dec. 2012. doi: 10.1111/j.1467-8659.2012.03184.x

[38] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, (SIGGRAPH)*, pp. 163–169. ACM, New York, NY, USA, 1987. doi: 10.1145/37401.37422

[39] S. McKenna, D. Mazur, J. Agutter, and M. Meyer. Design activity framework for visualization design. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2191–2200, Dec 2014. doi: 10.1109/TVCG.2014.2346331

[40] M. D. Merrill. First principles of instruction. *Educational Technology Research and Development*, 50(3):43–59, 2002.

[41] C. Meyers and T. B. Jones. *Promoting Active Learning. Strategies for the College Classroom.* ERIC, 1993.

[42] T. Munzner. A nested process model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, 15:921–928, Nov 2009. doi: 10.1109/TVCG.2009.111

[43] T. Munzner. *Visualization Analysis and Design*. A.K. Peters Visualization Series. A K Peters, 2014.

[44] T. L. Naps, G. Rössling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and J. A. Velázquez-Iturbide. Exploring the role of visualization and engagement in computer science education. *SIGCSE Bulletin*, 35(2):131–152, June 2002. doi: 10.1145/782941.782998

[45] M. Natali, I. Viola, and D. Patel. Rapid visualization of geological concepts. In *25th Conference on Graphics, Patterns and Images (SIBGRAPI)*, pp. 150–157, Aug 2012. doi: 10.1109/SIBGRAPI.2012.29

[46] D. N. Perkins, E. Jay, and S. Tishman. Beyond abilities: A dispositional theory of thinking. *Merrill-Palmer Quarterly*, 39(1):1–21, 1993.

[47] D. N. Perkins and D. N. Perkins. *The mind's best work*. Harvard University Press, 2009.

[48] A. Renkl, R. K. Atkinson, U. H. Maier, and R. Staley. From example study to problem solving: Smooth transitions help learning. *The Journal of Experimental Education*, 70(4):293–315, 2002. doi: 10.1080/00220970209599510

[49] J. C. Roberts, C. Headleand, and P. D. Ritsos. Sketching designs using the five design-sheet methodology. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):419–428, Jan 2016. doi: 10.1109/TVCG.2015.2467271

[50] J. C. Roberts, C. J. Headleand, and P. D. Ritsos. *Five Design-Sheets – Creative design and sketching in Computing and Visualization*. Springer

International Publishing, 2017. doi: 10.1007/978-3-319-55627-7

[51] J. C. Roberts, J. R. Jackson, C. Headleand, and P. D. Ritsos. Creating Explanatory Visualizations of Algorithms for Active Learning. In *Posters presented at the IEEE Conference on Visualization (IEEE VIS 2016), Baltimore, MD, USA*, October 2016.

[52] J. C. Roberts, D. Keim, T. Hanratty, R. Rowlingson, R. Walker, M. Hall, Z. Jackobson, V. Lavigne, C. Rooney, and M. Varga. From Ill-defined Problems to Informed Decisions. In M. Pohl and J. Roberts, eds., *Fifth EuroVis Workshop on Visual Analytics (EuroVA),*, pp. 7–11. Eurographics Association, Swansea, UK, 910 June 2014. doi: 10.2312/eurova.20141138

[53] K. Robinson. *Out of Our Minds: Learning to be Creative*. Capstone, 2001.

[54] K. Robinson. *The Element – how finding your passion changes everything.* Viking Penguin, 2009.

[55] A. Ryan and D. Tilbury. Flexible pedagogies: new pedagogical ideas. *Higher Education Academy, London*, 2013.

[56] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2431–2440, 2012.

[57] E. Segel and J. Heer. Narrative visualization: Telling stories with data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1139–1148, Nov 2010. doi: 10.1109/TVCG.2010.179

[58] P. Sengupta, J. S. Kinnebrew, S. Basu, G. Biswas, and D. Clark. Integrating computational thinking with k-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, 18(2):351–380, 2013. doi: 10.1007/s10639-012-9240-x

[59] C. A. Shaffer, M. L. Cooper, A. J. D. Alon, M. Akbar, M. Stewart, S. Ponce, and S. H. Edwards. Algorithm visualization: The state of the field. *ACM Transactions on Computing Education (TOCE)*, 10(3):9:1–9:22, Aug. 2010. doi: 10.1145/1821996.1821997

[60] B. Shneiderman. Creating creativity: User interfaces for supporting innovation. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(1):114–138, Mar. 2000. doi: 10.1145/344949.345077

[61] H. A. Simon. The structure of ill structured problems. *Artificial Intelligence*, 4(3-4):181–201, 1973.

[62] J. Sorva, V. Karavirta, and L. Malmi. A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)*, 13(4):15, 2013.

[63] R. Taub, M. Armoni, and M. Ben-Ari. CS unplugged and middle-school students views, attitudes, and intentions regarding CS. *ACM Transactions on Computing Education (TOCE)*, 12(2):8, 2012. doi: 10.1145/2160547.2160551

[64] J. Trumbo. Visual literacy and science communication. *Science Communication*, 20(4):409–425, 1999. doi: 10.1177/1075547099020004004

[65] E. R. Tufte. *Visual Explanations*. Graphics Press, Cheshire, Connecticut, 1997.

[66] J. Urquiza-Fuentes and J. Á. Velázquez-Iturbide. A survey of successful evaluations of program visualization and algorithm animation systems. *ACM Transactions on Computing Education (TOCE)*, 9(2):9, 2009. doi: 10.1145/1538234.1538236

[67] A. Vande Moere and H. Purchase. On the role of design in information visualization. *Information Visualization*, 10(4):356–371, Oct. 2011. doi: 10.1177/1473871611415996

[68] G. Wallas. *The art of thought*. Cape, London, 1926.

[69] C. Ware. *Information Visualization: Perception for Design*. Elsevier (Morgan Kaufmann), Amsterdam, 2012.

[70] D. Weiskopf, M. Borchers, T. Ertl, M. Falk, O. Fechtig, R. Frank, F. Grave, A. King, U. Kraus, T. Muller, H. P. Nollert, I. R. Mendez, H. Ruder, T. Schafhitzel, S. Schar, C. Zahn, and M. Zatloukal. Explanatory and illustrative visualization of special and general relativity. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):522–534, July 2006. doi: 10.1109/TVCG.2006.69