Visualization as Effective Instructional and Learning Tools in the Computer Science Curriculum

Dr. Mahmoud K Quweider, University of Texas, Rio Grande Valley

M K Quweider is a Professor of Computer & Information Sciences at the U. of Texas at UTRGV. He received his Ph.D. in Engineering Science (Multimedia and Imaging Specialty) and B.S. In Electrical Engineering, M.S. in Applied Mathematics, M.S. in Engineering Science, and M.S. in Biomedical Engineering all from the University of Toledo, Ohio. He also holds a Bachelor of English and a Masters of Business Administration from the University of Texas at Brownsville. After graduation, he was employed at several corporations including Pixera, a digital multimedia processing company in Cupertino, CA, 3COM, a networking and communication company in Schaumberg, IL, and Mercantec, an E-Commerce company in Naperville, IL. He has more than 40 publications in the field, and has served as a reviewer/moderator for several scientific and educational journals and conferences. He joined UTB in the Spring of 2000. His areas of interest include Imaging, Visualization and Animation, Networking and Cyber Security, Web Design, Computer Graphics, and Linguistics.

Dr. Fitratullah Khan, The University of Texas Rio Grande Valley

Professor Fitratullah Khan has been teaching computer science courses since 1992. His areas of expertise are computer architecture, networking, database systems, computing platforms and languages. As the director of Infrastructure, Telecommunications, and Networking (ITNet), and later as a Chief Technology Officer, at UT Brownsville, he implemented state of the art networking using campus wide fiber ring with redundant links. He established diskless computer labs to provide uniform computing platform across campus, and modernized classrooms to make them congenial to online learning. He was the PI on NSF funded BCEIL (Beowulf-based Curriculum Enrichment Integrated Laboratory) and Co-PI on NSF funded MCALL (Multimedia based Computer Assisted Learning Lab).

Visualization as Effective Instructional Tool in the Computer Science Curriculum

Abstract

Visualization (the use of images, diagrams, presentations, animations, gaming, and video) represents a potentially effective aid in teaching and learning, especially in the STEM (Science, Technology, Engineering, and Mathematics) fields where abstract complex ideas and concepts are abound.

Educators, especially in academia, are always searching for effective pedagogical methodologies to use in the classroom to enhance students' understanding and retention of key concepts of the subject area they are teaching. With the rapid advancements in software, hardware, networking, computing and storage technologies, including laptops, tablets, smartphones, cloud and distributed computing, the use of multimedia as an effective tool and aid in the classroom has become a standard procedure, rendering obsolete the traditional pure "chalk and talk." The next natural step to take is to enhance the presented course content through effective multimedia techniques. Of those techniques, visualization and animation have the most potential to revolutionize the way students learn and understand complex concepts that usually arise in the STEM fields, as they inherently appeal to our highly developed and specialized visual system that effortlessly identifies patterns, trends, and outliers.

Our paper presents a set of student-developed visualizations in strategically selected CS courses that enabled them to learn STEM related concepts in general (such as limits, differentiation, integration, and projectiles to name a few), and CS concepts/algorithms in particular. In creating these visualizations, the professors and the graduate assistants focused on incorporating into them a set of overarching themes that are effective and can be expanded to other fields. The themes were inspired by findings from a leading NSF Cutting Edge grant on teaching with visualization in a closely related field, although not one in STEM. The visualizations created were clear and simple; they are built on top of proven educational activities that were used in the past; the students' feedback was a central component as the visualizations were built step by step; the visualizations defined the pre-conditions before which a student can watch or run them, so that context is well-defined and not lost; and finally, the visualizations were organized to reflect the mental organization that the student is creating.

The paper gives details about the visualization algorithms, the criteria for their selection and inclusion in the curriculum, the students' immediate feedback, and survey results, taken by the students, that contrast the traditional ways of teaching CS and STEM concepts vs. the additional use of the developed visualizations. Our survey results shed light on whether visualizations make good tools for teaching, and if they have an effect on the rate (how quickly) of learning. Conclusions and recommendations are also presented.

Keywords

CS Education, Visualization, Animation, Gaming, STEM (Science, Technology, Engineering, and Mathematics) fields, Analysis of Algorithms, Software Engineering, Computer Vision.

Introduction & Motivation:

STEM fields are known for their richness in theoretical, abstract, and mathematical concepts. Such concepts when understood by the students can lead to creative and innovative applications of great value to society at large. These concepts apply to many forms, sizes, and speeds. From the tiny atoms and molecules that combine according to different chemical bonds, to the biological microorganisms that reproduce according to specific genetic codes, to the Newtonian laws of physics that dictate motion, speed, and acceleration, to computer science algorithms that can have different complexities and speeds.

When building instructional material for STEM courses, a professor, even when not aware of the process (see Figure 1 to compare educational model and software development model), is usually following the "the systematic process of translating general principles of learning and instruction into plans of instructional materials and learning," by applying a model such as the ADDIE model, which consists of five interrelated phases—Analysis, Design, Development, Implementation, and Evaluation. Incidentally, the educational model is similar to the model that is used and taught in STEM courses such as Software Engineering, Project Management, and Quality Assurance courses.

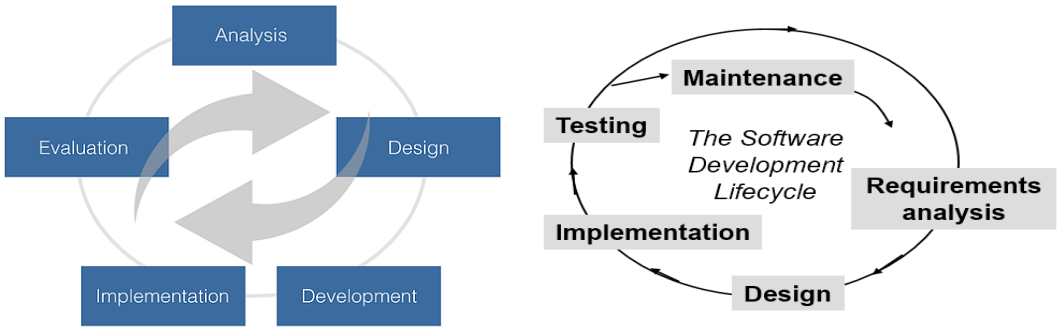


Figure 1. The Instructional Design Model & Software Development Model

Usually, it is within the development phase where professors would "develop the content and learning interactions," that could include textual, presentations, and assessment material.
To create material that engages the students through active learning, visualization is often cited as an effective tool to illustrate and break down complex concepts and to reinforce understanding of abstract models, mathematical formulas and algorithms. While there are many concepts and definitions related to visualization that include "visual representations, visual media, media literacy, visual communication skills, visual literacy, illustrations and media illustrations," (Vavra et al., 2011) our focus in this paper is on the use of simulations, mappings and transformations of model-generated data to graphical features that incorporate time, position, and other attributes in an animated way. We agree with Byrne et al. that "static visualizations provide people with the essence of how something looks, is laid out, or is constituted, animation appears better able to explain a dynamic, evolving process. Animations [on the other hand] may aid learners in constructing a mental model (a mentally-runnable simulation) of various processes' such as the movements of components of a mechanical system." (1999). By leveraging the "human visual systems' highly tuned ability to see patterns, spot trends, and identify outliers," (Heer, 2010) these animated visualization aid in understanding complex mathematical and algorithmic concepts such

as projectiles, metrics, algorithm complexity, and threading, just to mention a few. No doubt that the literature [1-10] is abound with visualization techniques use in the class room, however, our focus in this paper is to present them as a way to seamlessly reinforce and complement current educational methodologies.

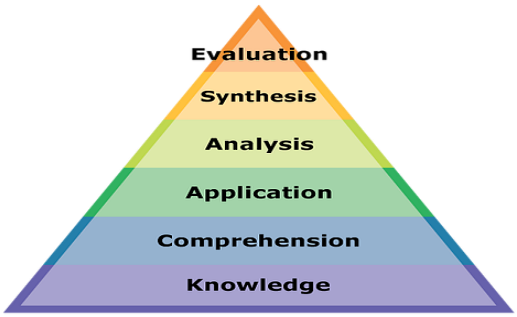


Figure 2. Bloom's Taxonomy Levels of Intellectual Behavior (https://en.wikipedia.org/wiki/Bloom's_taxonomy)

As it relates to Bloom's taxonomy (Wikipedia, 2016), the goal of the VRI model is to enhance the three lower levels (Knowledge, Comprehension, Application) of intellectual behavior important in learning. Table 1 shows the Verb examples that represent intellectual activity on each of those levels (Wikipedia, 2016).

Table 1. Intellectual Activity Verb Examples (https://en.wikipedia.org/wiki/Bloom's_taxonomy)

Level	Verb Examples
Knowledge	arrange, define, duplicate, label, list, memorize, name, order, recognize, relate, recall, repeat, reproduce state.
Comprehension	classify, describe, discuss, explain, express, identify, indicate, locate, recognize, report, restate, review, select, translate.
Application	apply, choose, demonstrate, dramatize, employ, illustrate, interpret, operate, practice, schedule, sketch, solve, use, write.

Our developed instruction method is called Visualization-Reinforced Instruction (VRI). It works by complementing current norms of instruction design (analysis, design, development, implementation, and evaluation). In particular, VRI aims to convert a student from a passive observer into an interactive participant by presenting them visual simulations, models, or algorithms to enforce the concept being studied. The VRI-modules are simple, fun, short, interactive, relevant, repeatable, and informative. They are designed to appeal to millennial learners who are computer savvy, adept at media devices such as tablets, smartphones, laptops, and computers, but have short attention span. By catering to the millennials evolved visual and auditory skills and their new preferences and social norms—which are undoubtedly different than the professors', it is fitting to revisit the instructional design tools that we are accustomed to and to incorporate new ones that leverage technology in the classroom and for the classroom.

Our conducted experiments and integrated modules confirm previous findings that visualizations such as VRI have the ability to make measurements and predictions based on the mathematical models at hand. However, these measures can be verified through the visualizations/simulations and even calibrated to match real-world dimension and values [1-4].

Visualization-Reinforced Instruction (VRI)

The VRI-modules are a form of active learning, in which traditional instructional material is complemented and enhanced with carefully designed animations and simulations of key concepts, models, and algorithms. The short interactive visualization allows students to observe patterns and relationships that would be hard to see otherwise. For example, the parabolic behavior of projectile is easily observed in a simulation of a projectile or set of projectiles that students can relate to in games such as Angry Bird or shooting games such as Golf and Basketball. Similarly, the order of execution of certain algorithms in comparison to others (that are easier to understand, but are slow or inefficient) can be easier to see when visualized as an image mosaic that is rearranged after a shuffle. Multithreading is another concept that we found to benefit from VRI due to its abstract nature.

The VRI-modules role is rather complementary and not disruptive to the current instruction design tools. The modules are an aid to enhance spatial mapping where learners form mental images of concepts that are vivid, colorful, and full of detail because of the reinforcement afforded to them by the visualization.

Characteristics of VRI-Modules

When designing the VRI modules we elicited the students input as to what changes would they like to the course materials to make them interesting. We captured these changes and incorporated them into the designed modules. Following are the characteristics that the students suggested, along with our own vision for the modules.

Simplicity:

The VRI-modules were implemented with simplicity in mind. The GUI used is intuitive and can run on any platform because it is implemented in Java. Students' feedback was sought as the modules were implemented.

Fun:

The visualization metaphors were selected to relate to millennial students and to the local culture which is predominantly Hispanic at our school. We used Angry bird for projectiles and threading; we also used soccer because the sport is very popular locally in the Rio Grande Valley.

Short:

The VRI-modules are very short and take a few minutes to setup and run. These can be run in class or lab without losing focus or the attention of the students.

Interactive:

The VRI-modules require some interactions from the students by inputting, for example, the initial velocity and angle, or the phrase to be encrypted, or the number of instances for a multi-threading application.

Relevant:

The implemented VRI-modules are relevant to the courses selected. We admit they are not inclusive of all the topics of interest. However, the framework we created is easily expandable to more modules.

Repeatable:
The implemented VRI-modules can be run multiple times so the students can check their predictions or confirm certain results.
Informative:
The implemented VRI-modules have useful information that the user can check and reinforce certain concepts from. The GUI will add more fields and visual objects as we try them again in the courses.
We have additionally involved students (from class or from other upper level courses) in the creation of these modules. Professors provided templates to the students from which they can start the visualization programming. The code was made available to all students through the course management system used by the university.

Targeted Areas for VRI
We have created VRI-modules for several computer science courses at different levels, and with different specialization areas. VRI-modules were created that can be used to reinforce key concepts in the following areas:

- Fundamentals of Programming I, II (CS-1, CS-2)
- Analysis of Algorithms
- Software Engineering
- Gaming
- Computer and Network Security

The modules are overlapping and can be used in other courses. They constitute a nucleus of a repository that will continue to grow as we share our findings and as more professors get involved.

Table 2 gives the description of the courses for which the modules have been designed. The courses are selected to cover all four-years of the CS degree. Table 3 on the other hand shows the concepts that were targeted within each course. This is not an exhaustive list, and future modules can be created to target other concepts that the student might have problems with understanding.

Table 2. Targeted VRI-modules

Prefix and Number	Course Description
CSCI-1370	CS-I: An introduction to computer science and computer engineering. The fundamentals of a high-level programming language will be introduced. Methods of problem solving, techniques of algorithmic development and concepts of procedural and object-oriented programming will be emphasized. Societal and social issues related to computer science/engineering will be introduced. Prerequisites: Grade of 'C' or better in MATH 1314 or placement in a higher level Math course.
CSCI-2380	CS-II: A second programming course includes problem solving by structured design; provides an introduction to elementary data structures, including linked lists, stacks, queues, trees and graphs, and advanced programming techniques, including recursion, sorting and searching. Prerequisites: CSCI 1370/CSCI 1380 (or CSCI 1387) or consent of instructor.
CSCI-3333	Algorithm and Data Structures: This course is a continuation of data structures topics covered in CSCI 2380. Content includes theoretical topics in algorithmic efficiency and complexity, along with abstract data types, including graphs, networks, trees, and priority queues. Search topics, including hashing, trees, external search trees (B-trees), and sorting algorithms including external sorting are introduced and compared. Computational complexity topics

	include the class P and NP, NP-completeness and reducibility, NPcompleteness proofs, and NP-complete problems. Prerequisites: CSCI 2380
CSCI-3340	Software Engineering: A formal approach to the state-of-the-art techniques in software design and development. Emphasis will be on Project Planning, Requirements, Specification, and System Design and includes object design, testing, and implementation. Provides the student with the opportunity to work on large projects in a group situation. Prerequisites: CSCI 2380.
CSCI-4365	Computer and Network Security: This course examines the internetworking architecture and routing, design and implementation issues related to secure and reliable networks, cryptography, firewalls, digital signatures, worms, viruses, logic bombs and spyware. Prerequisites: CSCI 4345 or CMPE 4345.

Table 3. VRI-Modules

VRI-Module-01	**VRI-Module-02**	**VRI-Module-03**	**VRI-Module-04**
Fundamentals of Programming I, II (CS-1, CS-2), Gaming	Analysis of Algorithms	Software Engineering	Computer and Network Security
Concepts Targeted: - Algorithm development - Threading	**Concepts Targeted:** - algorithmic efficiency and complexity, - sorting algorithms	**Concepts Targeted:** - Metrics, Quantifications	**Concepts Targeted:** - Cryptography algorithms

VRI in Action

In this sections, we go over the detailed of each modules and show which concept was selected for the VRI module.

Fundamentals of Programming I, II (CS-1, CS-2), Gaming:

VRI-Module-01 focused on projectiles. The projectile equations were derived and pertinent equations identified to create a model as seen in Figure 3. The equations were then used to create a user-interface where students can enter different velocities and angles for the projectile. The interface allows the students to check their own calculations against those given by the simulation.

Distance Equations:	$x(t) = v_0 \cos(\theta) t$
with (x_0=0, y_0=0)	$y(t) = v_0 \sin(\theta) t - (g t^2)/2$
Velocity Equations:	$v_x(t) = v_0 \cos(\theta)$
	$v_y(t) = v_0 \sin(\theta) - gt$
	$v(t) = \sqrt{v_0^2 - 2gtv_0 \sin(\theta) + g^2 t^2}$
Maximum Height:	$H = (v_0^2 \sin^2(\theta))/(2g)$
Occurs when $v_y(t) = 0$ (i.e. $y'(t) = 0$)	
Time to Impact:	$T = (2 v_0 \sin(\theta))/g$
Range:	$R = x(t)_{\|t=T} = v_0 \cos(\theta) T$

Figure 3. Projectile equations

These equations were then used to create different projectiles through a simple interface as shown below in Figure 4. Students had a choice between a soccer ball and an angry bird as their projectile object. A more advanced model with threading was created for the Software Engineering course and Gaming.

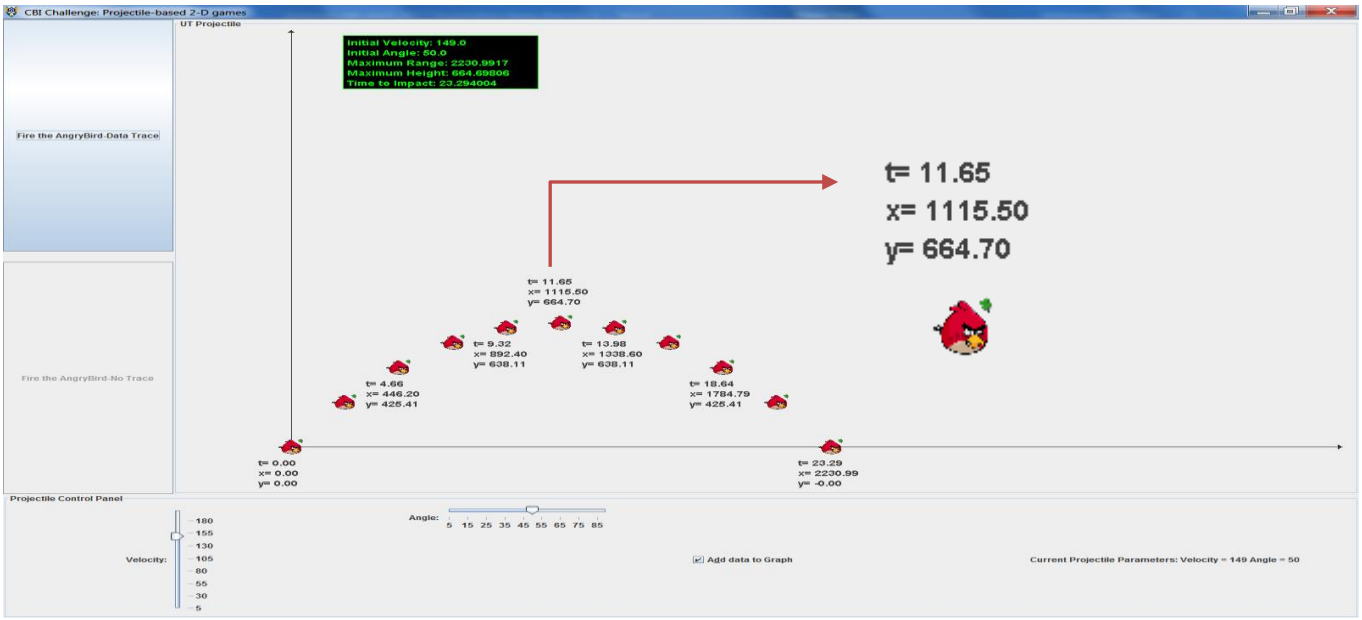
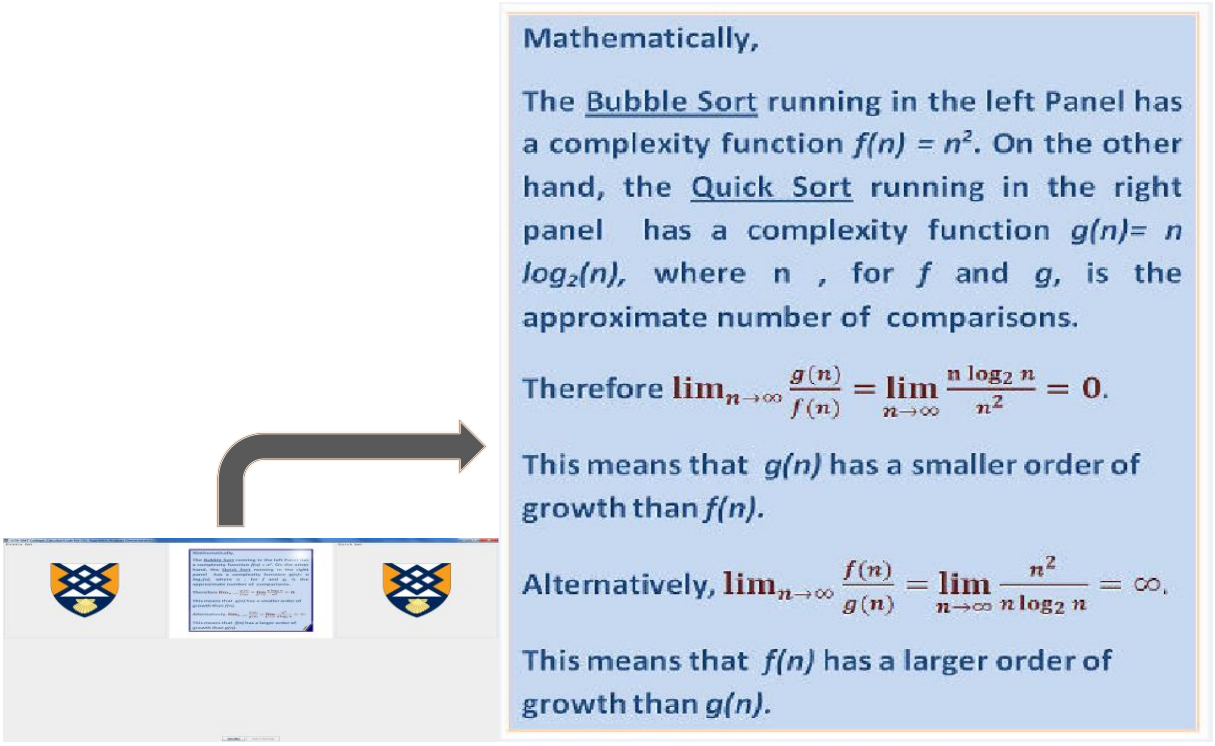


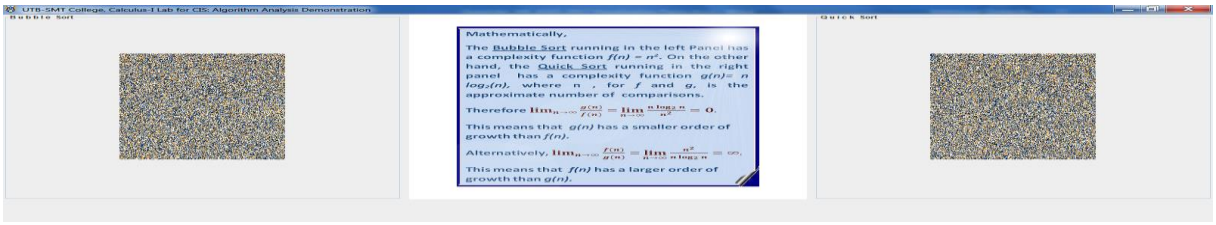
Figure 4. Projectile-based 2-D Interface with Angry Bird

Analysis of Algorithms:

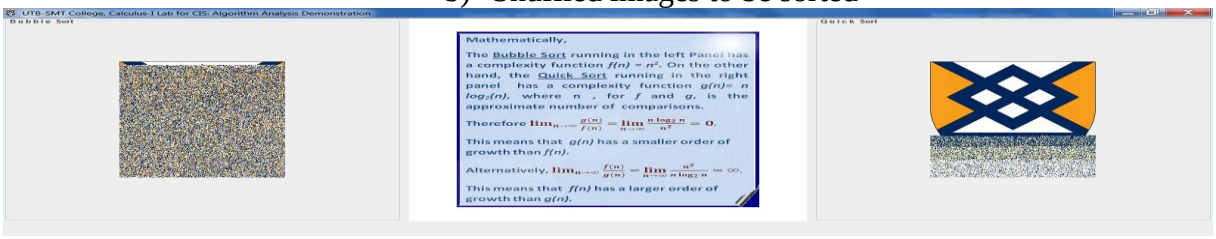
VRI-Module-02 focused on complexity functions as they relate to sorting algorithms. Two algorithms were chosen for comparison by shuffling the pixels of two images, and then rearranged to see which one finishes first. The sort time for each algorithm was measured and reported.



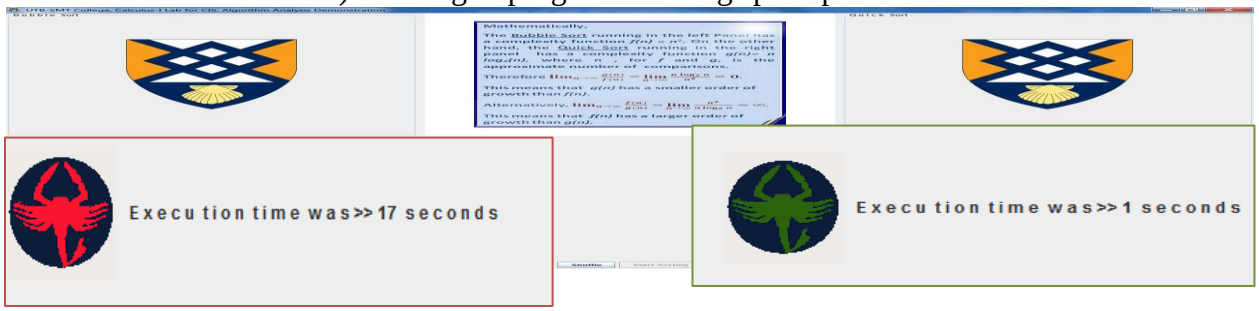
a) Initial Sorting Panel



b) Shuffled images to be sorted



c) Sorting in progress showing speed performance



d) Final sorting results with execution times

Figure 5. Sorting Performance Comparison

Software Engineering:

VRI-Module-03 focused on metrics and threading for software engineering. In the metric modules, students convert their GPA into a GPA widget (Figure 6), which later becomes part of a student's personal assistant that includes other metrics and functionalities. The projectiles/sorting modules were expanded to allow for multi-threading, a concept that our students struggle with tremendously. The threading modules (projectile and sorting) show how the same piece of code can be run as many times as the user requires and the system allows. Students interact with the system by deciding the number of threads before running the module (Figure 7/8).

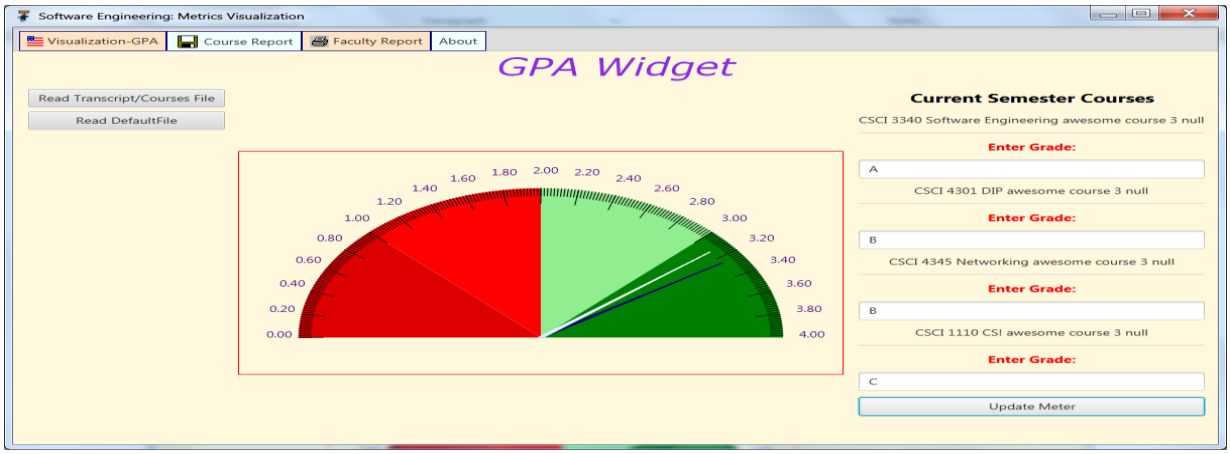
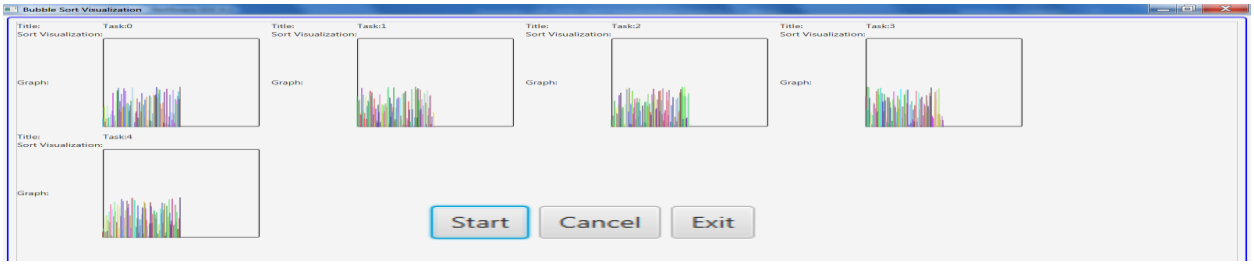
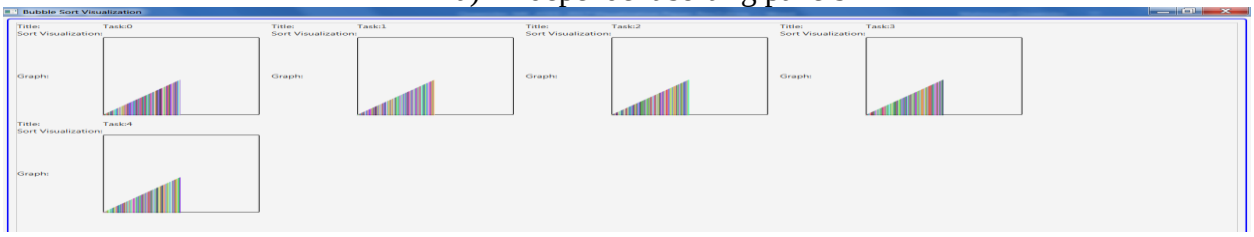


Figure 6.GPA Metric



a) independent sorting panels



b) Final sorting results with Multithreading

Figure 7. Final sorting results with Multithreading

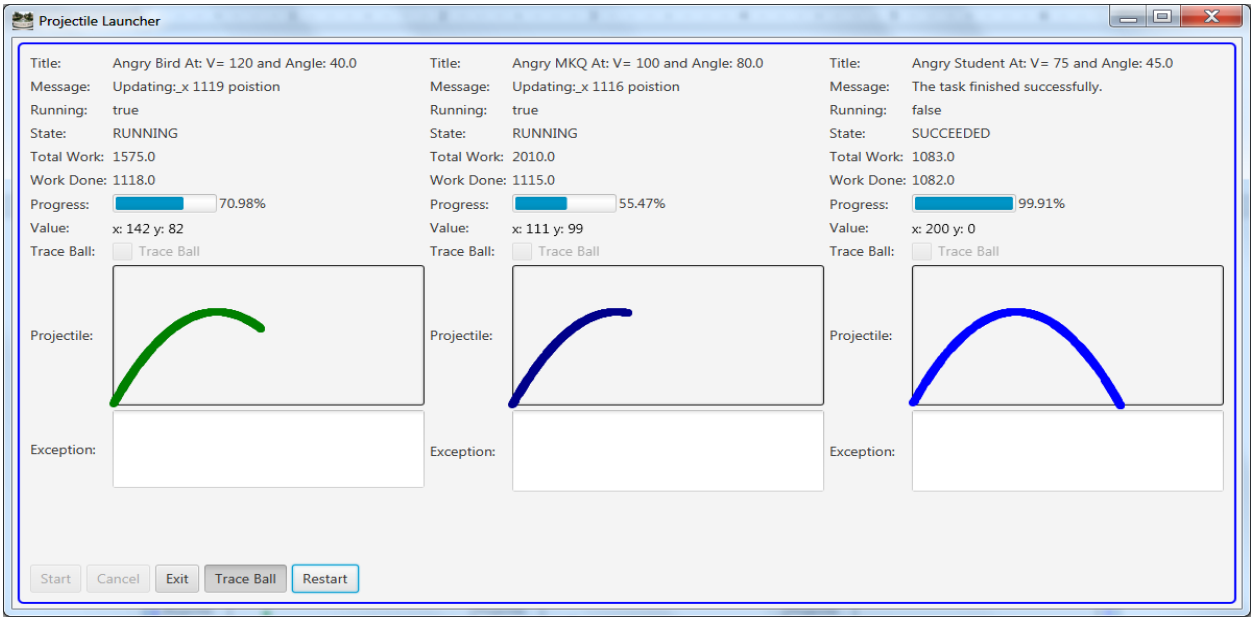


Figure 8. Projectiles with Multithreading

Computer and Network Security:

VRI-Module-04 focused on Cryptographic algorithms related to Internet Stack security at the application, transport, and networking layers. The VRI-module allows the user to encrypt and decrypt a plain text into cipher text and vice versa. The students are given the template and are asked to implement more advanced encryption techniques on images. The visualization goes through the whole cycle of encryption-decryption after transmission over a simulated network connection using TCP or UDP transport layer.

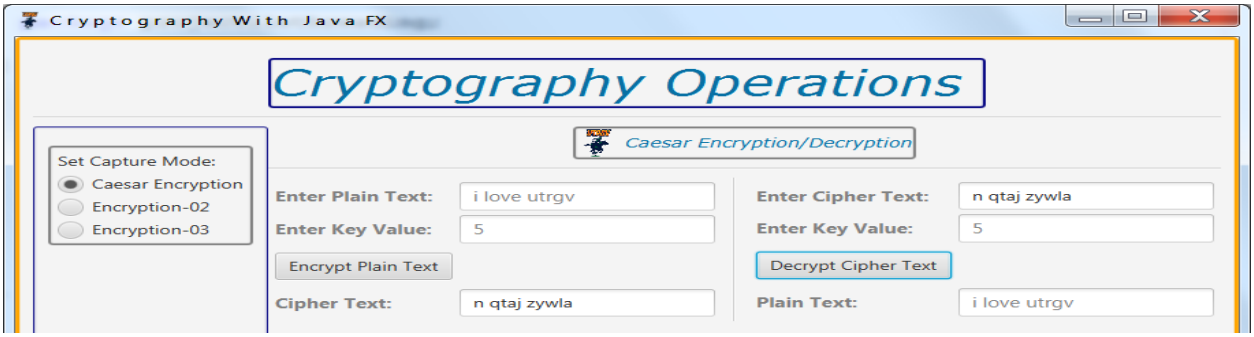


Figure 9. Cryptographic Algorithms

Visualization Reinforced Instruction (VRI) Analysis

We implemented several of the VRI-modules as we were teaching the courses or supervising senior level students. Students were either Computer Science or Computer Engineering majors, some with double majors with applied mathematics. So far, we have been able to analyze the results from two courses of Software Engineering and one course of Programming I (CS1).

When the students were asked if the visualization helped them understand the underlying concept, procedure, or algorithm, more than 76% answered in the positive. They indicated that, based on observing the VRI module, they were able to better understand the concept and even predict behavioral of other similar algorithms or problems. More importantly, the VRI-modules seem to improve the learnability of the concepts at hand by the students, that is to say how fast did the students comprehend the material. Compared to a previously taught course without the VRI modules, the professor had to spend two 1:15 hours of lectures (6.25% of class time) to explain the concepts.

We also noticed that students who did not have or master the proper prerequisites for the concepts did not benefit from the VRI-modules until they repeated the experiments several times, and were directed by other students or faculty to pay attention to certain variables or parameters. The motivational role for learners and their preferences are hard to ignore as well since many of our students learn be example.

Using the VRI modules, we were able to quickly identify many misconceptions and misunderstanding when the students were running the modules. In particular, the VRI-modules allowed the professors to update presentations, create numerical examples, and emphasize certain parameters or variables to clear these misconceptions. Examples include the role of velocity and angle in projectiles, the role of complexity functions as they relate to performance.

Since the source code was hosted on their course management portal (Blackboard), one of the pleasantly surprising findings is that many students took the initiative to modify the code and try different algorithms, from sorting to projectile, to multithreading.

Student Feedback

Below are some of the comments received from the students through the course evaluation form:

- The hands on part of the course was really helpful and I feel like I grew better at programming,
- I liked them for having us practice more with databases and visualization.
- Too intense, but you acquire great knowledge in theory and hands-on.
- I have learned more in his class than I ever did in my previous computer programming courses. I hope to take more courses with Dr. xxx.
- Great Course!

Conclusions and Future Work

Based on students' feedback, the VRI modules, through their animations, simulations, and algorithmic design were shown to facilitate learning about complex concepts and abstractions. The initial VRI modules have been a great success, but also highlight a number of issues to tackle in the future. In the future, we will examine how can VRI-modules be incorporated with other methodologies such as CBI [Quweider, 2016] (Challenge-based Instructions) and others.

Acknowledgment:

Part of this work was implemented by upper division students under my supervision either as part of their senior projects or as project assignments in Software Engineering. Their work and efforts are greatly appreciated.

References

[1]. Bonwell, C., and Eison, J. Active Learning: Creating Excitement in the Classroom. ASHE-ERIC Higher Education Report 1, 1991.

[2]. Briggs, T. 2005. Techniques for active learning in CS courses. J. Comput. Small Coll. 21, 2 (Dec. 2005), pp. 156- 165.

[3]. Bull, G., Bell, R., Garofalo, J., & Sigmon, T. (2002). The case for open source software. Learning and Leading with Technology, 30(2), 10-17. Available: http://www.iste.org/LL/pdfs/index.cfm?sku=30210b.

[4]. Clement, J. (2008). Creative model construction in scientists and students: The role of imagery, analogy, and mental simulation. Dordrecht, the Netherlands: Springer.

[5]. Hundhausen, C., Douglas, S., and Stasko, J. A meta-study of algorithm visualization effectiveness. Journal of Visual Languages and Computing 13, pp. 259-290, 2002

[6]. Kaput, J. (2001). Implications of the shift from isolated, expensive technology to connected, inexpensive, ubiquitous and diverse technologies. New Zealand Mathematics Magazine, 38(3), 1–26.

[7]. Mazza, Riccardo. Introduction to Information Visualization. 2009. Springer-Verlag London. ISBN: 978-1-84800-218-0

[8]. ODEP. (2016). Essential skills to getting a job. Retrieved from http://promotions.usa.gov/odep/essential_job_skills.pdf

[9]. Quweider, MK and Khan, Fitra "Implementing a Challenge-based Approach to Teaching Computer Graphics in Computer and Information Sciences," ASEE-2016 conference, New Orleans, LA.

[10]. Saraiya, P., Shaffer, C. A., McCrickard, D. S., and North, C. Effective features of algorithm visualizations. In Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education (Norfolk, Virginia, USA, March 03 - 07, 2004). SIGCSE '04, pp. 382-386.

[11]. Stasko, J. T. (1990). TANGO: A framework and system for algorithm animation. Computer, 23(9), 27±39.