ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #31931

A Team Based, Student Guided Approach to Learning Computer Algorithms Utilizing Video Game Programming

Prof. Robert Allen Langenderfer, The University of Toledo

Robert Langenderfer is a Professor in the Computer Science and Engineering Technology program at the University of Toledo. He received his M.S. at the University of Toledo and is currently pursing his Doctorate. Robert has developed software for a variety of organizations, including Microsoft, Dana, Pilkington, the University of Michigan, and the University of Pittsburg.

Robert has researched and implemented neural networks, expert systems, image processing, and industrial control. His most recent interests are in deep learning neural networks applied to autonomous vehicles and AI assisted STEM education.

Prof. Hong Wang, The University of Toledo

Dr. Hong Wang is a Professor Professor in Computer Science and Engineering Technology program in the Department of Engineering Technology program in the University of Toledo. He received his M.A and Ph.D in Computer Science from Kent State University (Kent, OH, USA). His research has covered parallel system design, smart building design, unmanned vehicle navigation under GPS denied environment and secure robot communications. His recent research interest is in data science field and STEM education.

A Team Based, Student Guided Approach to Learning Computer Algorithms Utilizing Video Game Programming

Abstract

"Play" and "fun" are words not usually associated with learning computer algorithms, but by directing students to design and develop a game during a semester course, students are kept engaged and given a sense of accomplishment. This paper discusses a team based, student guided approach to learning computer algorithms utilizing game programming. At the beginning of the course, students are asked to deliver a game that meets a list of basic requirements. The game requirements are designed to coincide with the student outcomes for the Computer Science and Engineering Technology course curriculum. While students dictate the specifics of the game design, implementation, testing and even the timing of those activities, the instructor maintains an active role insuring a reasonable pace and consistent participation by all students. Interlaced with student guided sessions, the lectures cover the basic concepts and terminology to accelerate understanding of the language vital to communicating the underlying concepts inherent in game programming.

Students take turns engaging in the various roles necessary to develop a game; lead, architect, programmer and tester. By participating in these roles and working together, students also develop social and team skills that are often missing from the normally solitary programming tasks.

Students are assessed with traditional testing focusing on base knowledge covered in the lectures, but are also evaluated on the quality of the completed games, as well as by other students in their rotating roles.

The inspiration behind this pedagogical approach is the difficulty maintaining student engagement when approaching the concepts and principles of computer algorithms and related theories. Since many students neglect their studies for the feelings of control and escapism provided by computer games, this approach incentivizes students by cross-linking their studies with preexisting free time behavior. Students often find a standard sorting algorithm uninspiring but are very interested when these methods are approached from the vantage point of conquering a series of digital enemies. Additionally, by examining the techniques utilized by their favorite video games, students infuse the enjoyment excitement of their play-time activities into their coursework.

Industry Demands Algorithm Knowledge

Students with a background in computer science and software design are in high demand and significant job growth is projected. Jobs for high paid applications and systems programmers currently employ the largest number of people in the technology field and have a projected growth of 21% by 2028 [1]. To meet this demand, university faculty need to find innovative ways to keep students engaged and increase graduation rates.

Technology companies commonly ask software developer candidates questions related to computer algorithm development, such as sorting, hashing, and brute force. There are multiple websites listing example questions from Amazon, Google, Facebook and Microsoft. These questions require potential employees to run the gauntlet of algorithmic thinking.

A Challenging Subject

In a computer science curriculum, Algorithms is one of the most important courses. As demonstrated above, employers understand the importance of this subject as well. Unfortunately, most students find algorithms and related computer courses daunting. Overall, computer science has retention rates that are significantly lower than other degrees. An effective course covering algorithms requires problem decomposition skills, critical thinking, significant applied programming expertise, debugging skills, a knowledge of data structures, and even creativeness. This complexity and high skill level make algorithmic theory and implementation very difficult to learn and master.

Numerous techniques have been employed by instructors to aid students and instructors when working with algorithms in the classroom. There has been significant research interest covered in the literature concerning the development of games for instructing computer science concepts and theory. Course design involving games attracts student's interests and promotes active learning [2]. Visualization tools have been employed to help scaffold student understanding by allowing real-time observation of data structure content [3]. Games have been designed to produce an engaging student learning experience [4, 5]. Learner collaboration in games was shown to be an effective method of enhancing student motivation and interest [6]. It was determined that the visual nature of these games are very helpful for providing a more intuitive view of certain classes of algorithms. The approaches above require the designing of interfaces or animations that simulates payable games. This paper designed an approach that doesn't involve a complex interface designs but still motivates student's algorithm learning through simple game design.

A major disadvantage the aforementioned approaches is that the instructor must locate, purchase, or develop sophisticated software for each algorithm or class of algorithms. The approach here harnesses the toil and creativity of the students to develop visually oriented demonstrations of algorithmic results.

The Method

Using short lectures, the instructor presents an algorithms and the associated theoretical framework. Then students are asked to work as a group to implement a gaming feature that requires the use of the specific algorithm.

For example, graph traversal algorithms help determine the optimal path from point A to point B in a connected graph, which is a representation of real or simulated environment. As an example, a connected graph may represent a network of roads. A robot, or game character may use the algorithms to find the best navigational path in an unknown, partially known or known environment. One common path planning technique is the A* algorithm. The A* path planning algorithm is presented in lecture, then the students are asked to employ this algorithm as part of

the artificial intelligence in a game where tanks need to navigate around obstacles to approach an enemy. Similar algorithms, such as Dijkstra's algorithm, are also introduced and the students are asked to evaluate the impact the new algorithms have on the performance of the game, based on timing or frame rate. These algorithms vary significantly in performance. For example, Dijkstra's algorithm finds the shortest path but explores all possible paths and executes slowly. In contrast the A* algorithm selectively examines paths and therefore completes in a much shorter time than Dijkstra's . However, Dijkstra's method performs very well in an unknown environment and is widely used on autonomous vehicle navigation. In the game, students are asked to simulate the impact of large numbers of random paths, enemies, obstacles, etc. This process helps elucidate the algorithm performance and demonstrate theoretical upper, average, and lower bounds which are indications of the algorithm's execution efficiency with empirical results.

The semester begins with students designing their version of a 2D tank battle game. Each class period builds on the previous, starting with a simple object oriented framework that can be progressively enhanced over time. At first, they are asked to consider in detail all of the components, actions, data attributes, interface and graphics that make up the specified game. Next, students are asked to work as a team to work on the object oriented classes for each of the components in the game. For example a tank class needs to have attributes such a name, an X Y position, color, health and methods or actions such as draw, move, collide, and health adjust. Programs are written using Python 3 turtle graphics for simplicity and compatibility. At this stage, students are asked to incorporate simple Euclidian and/or Manhattan distance algorithms, along with an aiming algorithm into their code. For the next phase, the enemy needs to be attacked closest first, so sorting algorithms such as Insert, Bubble and Quicksort are introduced. Next, the data representation of the game space or 'board' is discussed and directed graphs are implemented for this purpose. Likewise, sorted linked lists are introduced to replace the sorting requirements, A* and Dijkstra's, mentioned above, are used for navigation, divide and conquer and greedy algorithms are implemented for sophisticated strategies. At various points, the game can be turned into more of a simulation for students to determine the best techniques to apply. All along this process, students are given first-hand experience with programming and see the impact their choices and the choice of algorithm has on game performance.

For part of the semester, students take turns engaging in the various roles necessary to develop a game; lead, architect, programmer and tester. The lead acts as 'the boss' and directs the tasks, assigns roles, and reports progress. The architect specifies features, creates levels, and creates or locates artwork. The programmer implements data structures, designs the program structure and implements code. The tester, of course, checks the resulting code for bugs and evaluates performance. By participating in these roles and working together, students also develop social and team skills that are often missing from the normally solitary programming tasks.

This pedagogical approach was inspired by the difficulty maintaining student engagement when approaching the concepts and principles of computer algorithms and related theories. Commonly, students neglect their studies for the feelings of control and escapism provided by computer games, this approach incentivizes students by cross-linking their studies with preexisting free time behavior. Games are as much a part of life for college students as attending classes and studying. Pew research has determined that 72% of US college aged males (18-29)

often or sometimes play video games [7]. Utilizing voice and video communications, video games are also becoming an increasing proportion of student social activity [8].

A standard sorting algorithm is rather uninspiring, but most students are very interested when these methods are approached from the vantage point of conquering a series of digital enemies. Additionally, by examining the techniques utilized by their favorite video games, students infuse the enjoyment excitement of their play-time activities into their coursework.

Evaluation

There is an immediately noticeable difference between classrooms employing team collaborative approaches verses traditional lecture style instruction. There is a buzz of excited discussions amongst team members as they actively collaborate and coordinate. Students motivate each other with developments and discoveries as they progress through aspects of a game based project. There is a clear joy as they show off their accomplishments to teammates, classmates and the instructor. Traditional instructional techniques create a quiet, passive classroom that must compete with distractions like texting and the wide variety of subjects on the Internet.

A side benefit of game development is that students will have an interesting and significant project to demonstrate to future employers as well as family and friends. Few people would delight at the sight of a row of numbers output from an insertion sort, but most would express interest in a game character pursuing the closest digital enemy.

Students attending this course are composed of a wide variety of traditional and nontraditional students, and hail from many countries. Ages range from 15 to 65, though the majority is 18 to 25, and only a single digit percentage are female.

Students are assessed with traditional testing focusing on base knowledge covered in the lectures, but are also evaluated on the quality of the completed games, as well as by other students in their rotating roles. The lead student on each team evaluates the participation of each of his ‘workers’. This process encourages that all students participate equally in the project tasks.

Conclusions

It has been shown that approaching algorithmic pedagogy from a collaborative game themed environment is an effective method to improve student interest and engagement. Student projects leave students with a source of pride and a truly interesting demonstration of their knowledge and effort.

However, better methods of surveying students need to be developed, as well as a distance learning version of the course. Future work areas of research could involve cross course collaboration for greater impact. For example, the department’s client/server programming course could incorporate the algorithms coursework to develop a massively multiplayer network based game.

References

[1] “Occupational Outlook Handbook: Software Developers Job Outlook”, *Bureau of Labor Statistics,* Sept. 2019. [Online]. Available: https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm#tab-6 [Accessed Jan., 2020]

[2] Ashok R. Basawapatna, Kyu Han Koh, and Alexander Repenning. “Using scalable game design to teach computer science from middle school to graduate school,” *In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education. (ITiCSE ’10) Association for Computing Machinery*, New York, NY, USA, 2010, pp. 224–228. DOI:https://doi.org/10.1145/1822090.1822154

[3] S. Simoňák, "Algorithm visualizations as a way of increasing the quality in computer science education," *2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Herlany, 2016, pp. 153-157. doi: 10.1109/SAMI.2016.7422999

[4] I. Hatzilygeroudis, F. Grivokostopoulou and I. Perikos, "Using game-based learning in teaching CS algorithms," *Proceedings of IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* 2012, Hong Kong, 2012, pp. H2C-9-H2C-12. doi: 10.1109/TALE.2012.6360338

[5] S. S. Shabanah, J. X. Chen, H. Wechsler, D. Carr and E. Wegman, "Designing Computer Games to Teach Algorithms," *2010 Seventh International Conference on Information Technology: New Generations,* Las Vegas, NV, 2010, pp. 1119-1126. doi: 10.1109/ITNG.2010.78.

[6] F. Grivokostopoulou, I. Perikos and I. Hatzilygeroudis “A Collaborative Game for Learning Algorithms,” *In Proceedings of the 9th International Conference on Computer Supported Education 2017 - Volume 1*: CSEDU, ISBN 978-989-758-239-4, pages 543-549. DOI: 10.5220/0006377405430549

[7] A. Perrin, “5 facts about Americans and video games,” *Pew Research Center*, Sept. 2018. [Online]. Available: https://pewrsr.ch/2vbbMxD [Accessed Jan., 2020].

[8] C. Uz, K. Cagiltay, and K. Cagiltay, “Social Interactions and Games”, *Comparative Education Review*, 27. 1-12, 2015