Placating Plato with Plates of Pasta: An Interactive Tool for Teaching the Dining Philosophers Problem

Justin DeBenedetto*, Stephen Hutt*, Louis Faust*, Anqing Liu*, and Nathaniel Kremer-Herman*
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN
{jdebened, shutt, lfaust, aliu1, nkremerh}@nd.edu
* Denotes equal contribution to this work.

***Abstract*—There has been a recent surge in the need for computer science educational resources. Often, this need is addressed with computer-based learning tools. However, many of these tools target coding skills and software design rather than teaching foundational topics. This work proposes an interactive tool for teaching process management using the common abstraction of the Dining Philosophers Problem. We demonstrate the effectiveness and entertainment value of this tool with feedback from undergraduate students enrolled in computer science courses. We believe this type of educational tool more adequately addresses the upcoming need for additional teaching resources in computer science, especially when used in conjunction with a traditional teaching environment.**

I. Introduction

Imagine you are a student in an Operating Systems class, a required course for your Bachelor's Degree in Computer Science. For a homework assignment you have to write a process manager in C. While testing your code you accidentally enter an infinite loop that constantly spawns new processes, ultimately overloading the server you are working on. This results in the server being taken offline. Neither you, nor your classmates, are able to work on the assignment until the next morning. Now imagine you are the teacher of this course. You are about to be inundated with emails from students who cannot complete the assignment through no fault of their own since the server is down. This example should illustrate one of the challenges involved with teaching system-level computer science topics: the lack of a safe, learning-focused testing sandbox.

The challenges of teaching topics such as process management will only increase as computer science becomes more common in K-12 curricula. In 2015, the mayor of New York City announced that within ten years every public school in the city would offer computer science [1]. This announcement follows similar announcements from a string of other school districts across the country, raising the profile of computer science education in the United States [2] as well as around the globe [3].

As we know, the field of computer science covers complex concepts especially for novices. However, many of the available educational tools and modules utilize software and programming languages such as Scratch and Alice [4, 5] to reinforce coding practices rather than concepts and theory. This work aims to provide a new tool for educating students, focusing on operating systems concepts, specifically resource and process management, with the scope to expand the tool to include more topics. The need for covering operating systems content in computer science curricula has been discussed by ACM at the undergraduate level [6] and at the high school level by the Computer Science Teachers Association [7]. The importance of more than cursory student experience with operating systems topics has become relevant at both these stages of education, and the Dining Philosophers Problem has been explicitly mentioned as a useful thought experiment for introducing process management.

We present *Placating Plato with Plates of Pasta*, a tool for teaching the Dining Philosophers Problem, as a first step toward addressing the current lack of learning tools which teach computer science curricula beyond coding skills. This web-based learning tool consists of various activities designed to provide a balance between entertaining, engaging tasks and more traditional tasks which are helpful for the student's knowledge retention akin to a homework assignment. We also provide a safe sandbox for developing and testing solutions to the problem. The tool was evaluated by a group of undergraduate students, a large portion of whom were taking their required Operating Systems course. Our results demonstrate this first step was successful in providing both an effective learning experience and entertainment value.

A. Related Work

Computer-based training and assessment (CBT and CBA respectively) have been critical topics in both academic and corporate contexts since the new millennium. Based on a survey of experiments [8], researchers drafted

978-1-5090-5920-1/17/$31.00 ©2017 IEEE

a list of principles required for effective visualization. Engagement of the learner was key among these, noting even the best designed visualization would fail if not offering the user some level of engagement. Interactive visualizations offer dynamic learning environments through features such as adapting to the knowledge level of the user and feedback based on the learner's performance. A variety of research has been done with regard to these interactive learning systems, for example Guru [9] in which a system that uses a combination of lecture and interactive tasks to form a biology lesson is presented. Investigation of this tool has shown that such tools work well to augment classroom learning.

Other work has been done specifically targeting the Dining Philosophers Problem [10]. This work looked specifically at directive versus heuristic learning as well as computer-assisted versus lecture-based learning. The authors found that a directive model was more effective in teaching complex operating systems concepts, meaning a more structured and guided learning experience was best. It was also noted, as we might expect, that computer-assisted learning is *not* guaranteed to be better than lecture-based learning depending on the teaching style. This is an important aspect to keep in mind when designing a computer assisted learning tool since we have to carefully consider which learning styles we cater toward.

When utilizing computer-assisted learning, there are also different areas of focus. Some work has focused more on animations of concepts such as algorithms [11], while others focus more on gamification. Regalado [12] provides results from the impact of gamification aspects in a Virtual Learning Environment (VLE), showing that gamification achieves greater engagement of users in relation to higher frequency of use of the VLE: the greater number of views per video lesson, the greater number of completed challenges. When designing gamified elements of a learning tool, the amount of interaction and engagement we can expect from the user will determine how effective their learning experience will be.

Mateos [13] developed a computerized board game for education focused on combining learning goals with entertainment design principles including motivation, learning, and gaming. When evaluated across more than fifty students, results indicated a high level of satisfaction and showed positive results across the three principles, showing education-focused games can motivate and aid the learning process. When implementing gamified course concepts, these learning goals can provide a more clear design blueprint to follow.

Any time that research is done on helping students to learn, Bloom's Taxonomy is a popular model to follow. Thompson et al [14] analyzed Bloom's Taxonomy and how it is applied to computer science. They seek to provide an organized and consistent interpretation for the computer science learning structure. Specifically, they apply aspects of Bloom's Taxonomy to programming exams. We find that the structure of Bloom's Taxonomy provides a well-established foundation to better follow the guidelines like those previously provided by Mateos [13].

B. Current Study and Novelty

In this work we utilize the findings of previous research to design an educational tool which integrates several computer-assisted learning techniques into the framework of Bloom's Taxonomy. Specifically, we address a need within undergraduate computer science education, given the current lack of similar educational tools.

The tool, Placating Plato with Plates of Pasta, is designed to augment a traditional learning environment not to replace it. For example, we expect the tool to be used as an assignment or extra learning activity to reinforce the goals of an Operating Systems course. As previously discussed, many of the computer science learning tools available focus on coding skills or very practical applications. Tools such as these are often expected by the designers to be the only avenue for delivering the learning goals to their users (i.e. coding for a specific purpose). We expect our students to be currently enrolled in a computer science course, providing a second outlet for student questions and engagement.

We demonstrate the validity of our methods via initial experimental results with 29 undergraduate students. We analyze the results considering both quantitative and qualitative measures and show positive results from each, highlighting an effective educational experience that is also entertaining and accessible for students.

II. Design Goals

When designing the tool we considered three factors: (1) Environment, the tool should be able to explore concepts in an accessible and safe (e.g. no segmentation faults) manner. (2) Education, the tool should effectively implement features from the learning literature. (3) Entertainment, the tool should aim to be more entertaining and engaging than a traditional textbook-and-homework lesson structure.

A. Environment

An effective educational tool must be accessible to all users to facilitate user engagement. We note from the literature that greater engagement leads to higher achievement [12]. When ease of use is lacking, users may become frustrated or disconnected from the material.

Accessibility extends beyond ease of use when considering a tool of this nature, especially considering the content. If one user may accidentally cause a system to crash and prevent other users from using the tool, their experiences are hampered. We provide an environment that allows users to create and explore in a safe manner without the fear of major repercussions. In addition to this we also provide guided feedback on errors wherever possible in a manner that does not affect the entire user

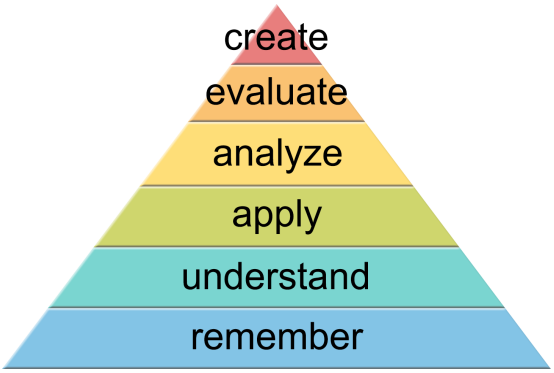

Fig. 1: A revised Bloom's Taxonomy hierarchy [15].

population. Errors are often the best way to learn and explore concepts, as a result a crucial design goal was to facilitate this without risk to system integrity.

B. Education

Although providing a safe environment for students to test is useful in itself, we also wanted to make sure the web tool was designed following the guidelines of the education literature. Anderssen et Al. [10] find that providing a computer-based learning environment is not inherently more effective than traditional methods. We designed the tool to have increasing levels of engagement and student interaction as they progressed, following suggestions from prior work [8]. Though not the sole focus of our work, we chose to 'gamify' one activity in the tool to further reinforce the students' interactivity following Regaldo et al. [12] as an example. Overall, our design drew inspiration from the lessons learned in previous literature.

To make our tool as useful as possible with regard to student learning, we chose to make our design decisions based upon Bloom's Taxonomy [14]. This model has been well regarded for quite some time and has informed pedagogy at all levels of education. As shown in Figure 1, a hierarchy of learning objectives can be made on the basis of the objectives' complexity or specificity of the knowledge assessed.

This structure presented an iterative learning experience which we considered for the design of the system. By using this taxonomy from the outset, we were able to structure our design process and build a tool that met these goals specifically rather than trying to retrofit previously constructed software. From the taxonomy we were able to extract goals for our tool. After using the tool, students should be able to (1) understand and remember the basic concepts, such as deadlock and starvation, (2) apply concepts to situations within the philosophers paradigm (3) apply concepts to situations outside the philosophers paradigm, and (4) create and analyze solutions to process management problems.

C. Entertainment

Naps et al [8] discuss that a key design principle required for effective visualizations is engagement. We sought to implement novel methods for learning the material to make an enjoyable experience for the user.

Having considered our first two design factors, it seemed most beneficial for this tool to be used in students' study environments rather than directly in the classroom. With this in mind, entertainment required extra attention in our design. We wanted the tool to be more entertaining and engaging than a conventional textbook in order to make these topics more interesting and enjoyable for students. To keep students engaged, we chose to implement interactive activities, including a game with a scoring element, where students are challenged to beat their own score.

III. Implementation

The tool consisted of seven sections in a fixed order, each representing levels of Bloom's Taxonomy and building on concepts covered in previous sections. These sections, as well as their goals, are outlined in Figure 8.

Each of these sections was implemented as a separate webpage using a combination of HTML, JavaScript, and PHP. These pages were connected in an order fitting for Bloom's Taxonomy. Most pages use a combination of images, animation, and text. More interaction is added the further through the tool the user progresses.

To guide students through the concepts presented, we created a visual motif that was used consistently for all examples. This consisted of a dining table, a variable number of philosophers with different symbols representing their current task, as well as warning symbols for potential starvation. An example of this is shown in Figure 2 to present a coherent session to the student.

When introduced to the topic, students are shown brief animations of two philosophers at a table with two chopsticks; one animation shows deadlock and the other shows starvation. Accompanying text explains the concepts the animations display. We then transition to an explanation of how this applies to operating systems concepts. The student then works through a matching activity where they are presented with three real world scenarios, and they must decide whether the scenario is a case of deadlock, starvation, or neither. The student is given text feedback based on their choice, and they can then change their choice if desired. An interactive game (see Section III-A) is the next activity and the one which we considered the turning point of the students' experience from a sedentary experience to an interactive one.

Following the game, students are shown two solutions to the Dining Philosophers Problem: the resource hierarchy and arbitrator solutions. Each is shown in a step-by-step animation. Students then experiment with their own solutions using drag-and-drop pseudocode blocks. The

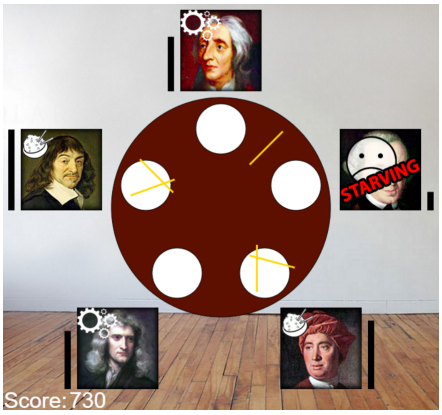

Fig. 2: Screenshot of the interactive game. The student must manually click philosophers to have them eat or give up their chopsticks. Once a philosopher becomes too hungry, they starve and the game will end providing a score to the student.

final activity provides two sets of side-by-side pseudocode implementations of potential solutions. The student must analyze and decide which solution would actually work. Like the matching activity, this solution identification game provides text feedback for the student's choice. Unlike the pseudocode for the code blocks activity, we designed the pseudocode to look more like *C* than the English language. This was to prepare students to implement their own code for their Operating Systems course.

Students progress through the tool in the order shown in Figure 8. They are able to revisit previous sections, but they cannot skip over sections they have not visited. This was implemented to ensure the Bloom's Taxonomy implementation order was followed. This also ensures the students always have the pre-requisite knowledge to complete a section. The total experience is designed to last between 15 and 30 minutes.

While most of the sections involve basic logic, two present a more complex task. The interactive game and coding by blocks were more complex implementations and are discussed further below.

A. Interactive Game

The interactive game was implemented in JavaScript. The student is given control over the Dining Philosophers Problem with the ability to switch the states of the philosophers between thinking and eating; following the same restrictions as the original problem. Each philosopher has an associated "health" meter which slowly drains when a philosopher remains in the thinking state. Changing a philosopher's state to "eating" will refill the meter in the same manner. If the meter diminishes completely, the game will end notifying the player: "A philosopher has starved." This addition was made to highlight the need for solutions to the problem and demonstrate how easy it is for a process to starve in an unstructured approach.

The rates at which the health meters increase and decrease become faster as the game continues. This forces the player to develop practices early on for managing the health meters of the philosophers, since they will need to fall back on a sequential, deterministic approach for late-game switching between philosophers before they starve.

A scoring feature was also added to the game. The score is based on the fullness of each health bar for each second that passes. This allows players to compare different management techniques for the philosophers based on their score and note any improvement.

B. Code Blocks

Students implement their own solutions to the Dining Philosophers Problem using a creative environment with drag-and-drop elements and a provided framework. Students were given an initial skeleton of code which served two purposes, (1) to guide students to areas of the activity that are most beneficial to their learning and (2) to optimize student time and limit time wasted on syntax and coding issues. The skeleton code also reduces the prerequisite knowledge required to complete the activity.

An example of the code blocks is shown in Figure 3. Code with a grey background is a fixed part of the skeleton. Blocks with amber or green backgrounds may be swapped with other blocks of the same color. Amber blocks relate to the philosopher process and green is for main function behavior. When a student is confident with their answer, they can then test their code. Students receive written feedback on their solution as well as an animation, showing the behavior of their solution. Any outstanding syntax errors (i.e. incompatible code combinations being selected) are also explicitly reported.

This activity has an underlying logic system implemented in JavaScript. This system tracks syntax errors present in the code. Given the small number of code blocks, an exhaustive check was used to look for issues. Similar to a compiler running the code, all errors are reported back to the student, but because of the limited scope we are able to provide more verbose, specific feedback on each error.

In addition to checking for syntactical errors, the activity also checks for task related errors, specifically, whether deadlock or starvation can occur. If a correct configuration of code blocks is selected, neither deadlock nor starvation is possible. An animation shows their successful solution to the Dining Philosophers Problem.

IV. Evaluation

Our next step was to evaluate if our implementation was useful for students. We did this by providing access to our tool to a group of undergraduates. We captured their on-screen interactions within each activity of the tool, basic metrics such as time spent per page, and their performance in both a pre- and post-test.

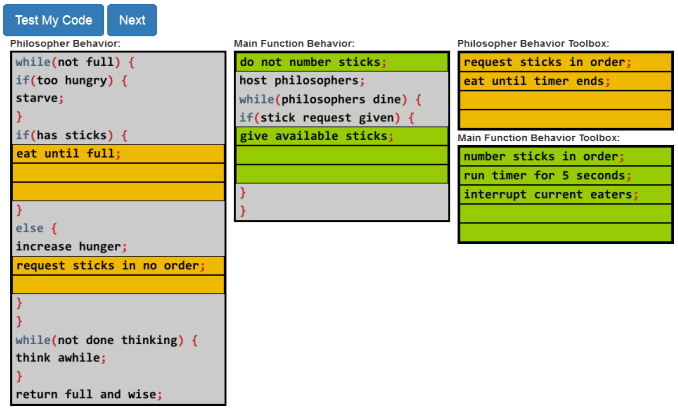

Fig. 3: Screenshot of the drag-and-drop code blocks activity. The student must implement their own solution to the Dining Philosophers Problem using code blocks from two toolboxes: the philosopher process behavior and the main function. The student can test their code and see an animation detailing the resulting behavior: deadlock, starvation, or success.

A. Experiment

Participants were 29 undergraduate students at a mid-sized private Midwestern university. Of the 29, 24 were currently enrolled in an undergraduate Operating Systems (OS) class, however, the Dining Philosophers Problem was not explicitly covered. Two random participants were chosen to receive one $25 Amazon gift card each as a thank you for their participation.

Having given informed consent, participants answered a few demographic questions (Class Year, Major, Current Computer Science Courses) and completed an eight question pre-test. The questions were modeled after the assessment from [10]. Participants then completed a full session with the tool, completing all sections outlined in Figure 8. Following this, participants completed a post-test. This consisted of the same eight questions as the pre-test. Students then completed a short survey about their experiences with the tool.

All student answers were recorded as well as basic interaction data (e.g. time spent on each task, task completion, exercise scores).

B. Evaluation Methods

We considered success to be on two scales, (1) whether students learned from using the tool, and (2) did students enjoy using the tool.

To measure learning we chose to primarily use the pre- and post-tests. By taking a pre-test we are able to control for pre-existing knowledge and compare it to the post-test to get a measure of improvement. Questions in the pre-test were the same as those in the post-test to provide a direct comparison. It should be noted that this method does not allow us to evaluate long term retention of information, which is discussed further in Section V-C.

Assessing how much students enjoyed the tool was done through the survey at the end of the session. Students were asked on a five point scale to rate whether they found the tool as a whole helpful and a further question on the same scale whether they found the tool enjoyable. These questions were then repeated for each individual section of the tool

By assessing whether students enjoyed interacting with the tool and found it useful, we gained some understanding of student experience and whether students would be likely to use a tool of this style again.

C. Results

Students completed pre- and post-tests consisting of the same eight questions. Table I presents the mean, minimum, and maximum scores based on the number of correct responses. Students were separated into two groups based on whether they were currently enrolled in the Operating Systems (OS) course.

TABLE I: Score statistics

	Mean	SD	Min	Max
All Students				
Pre-Test	5.93	1.73	2	8
Post-Test	7.14	0.74	5	8
OS Students				
Pre-Test	6.33	1.50	2	8
Post-Test	7.25	0.68	6	8
Non OS Students				
Pre-Test	4.00	1.58	2	6
Post-Test	6.60	0.89	5	7

Out of 29 total participants, 20 improved on the post test, 5 remained the same, and 4 received lower scores. The average increase in post-test scores was 1.21, which was 20.4% of the average pre-test score. The minimum score increased from 2 to 5.

Compared with students taking Operating Systems (OS), non-OS students had lower pre-test scores on average, but more significant increases in post-test scores. OS students had higher pre-test and post-test scores, with an average post-test score of 7.25. Although the improvement was smaller than that of non-OS students, the high pre-test scores suggest OS students were already familiar with the concepts tested, allowing less room for improvement. These increases in scores in both groups signal that students benefited from the tool whether or not they had prior knowledge of the topic.

A paired t-test on the scores of the 29 students gave a 95% confidence interval of 0.61 to 1.80 for the mean improvement from pre-test to post-test for the participants. The results were statistically significant with a p-value of 0.0003. When broken up into the 24 OS students and the 5 non-OS students, both groups also showed statistically significant improvement.

Following the post-test, participants rated the tool on a five point Likert scale from “definitely yes” to “definitely not” whether it was helpful and enjoyable. In Figures 4

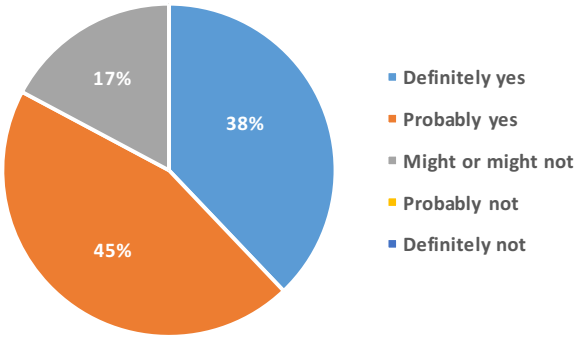

Fig. 4: Was the tool helpful

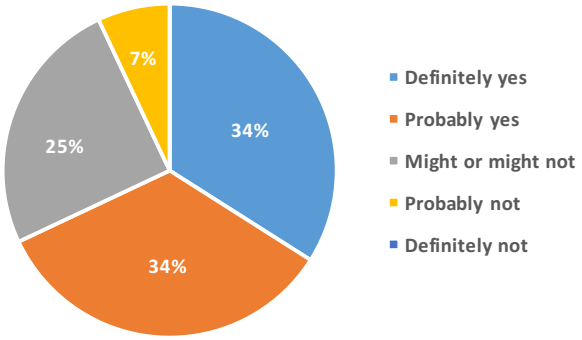

Fig. 5: Was the tool enjoyable

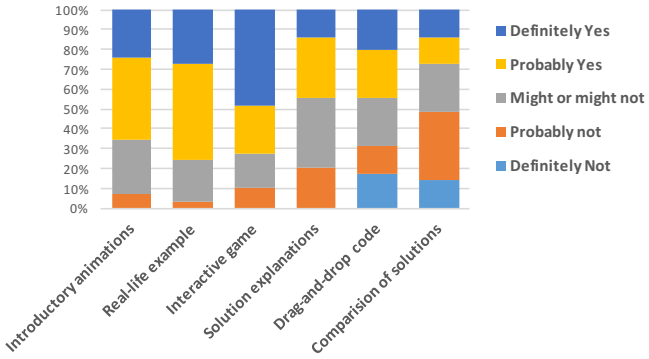

Fig. 6: Was each section of the tool entertaining

and 5, we see 83% of participants thought the tool was "definitely" or "probably" helpful, and 69% of the respondents considered it enjoyable. No student considered this tool as "definitely not" enjoyable or helpful, while only two students answered "probably not" to the question "Did you find this helpful?". Almost all participants found this tool helpful, and the majority of participants enjoyed

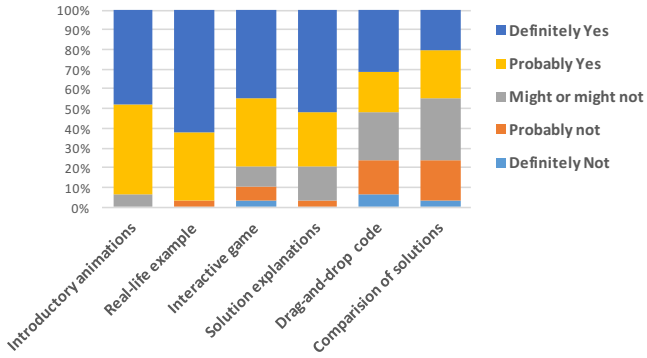

Fig. 7: Was each section of the tool helpful

using this tool. A more detailed section-based rating summary is shown in Figures 6 and 7.

V. General Discussion

Computer science is a rapidly developing curriculum in the college environment and also now in K-12 education. Over the next ten years there will be multiple schools offering computer science courses for the first time [16]. This presents a variety of challenges for educators, not least of which is a lack of resources for students to explore and experiment with complex computer science topics in a safe environment. We address a small part of this challenge by applying known learning theories when designing and creating a cyber-learning tool and explore how effective this is for learning and in the eyes of students.

A. Main Findings

We have designed and implemented a cyber-learning tool that encompasses all levels of Bloom's Taxonomy. Considering each level in turn we designed short activities, combining levels where appropriate. With this in mind, we were able to develop a tool that presents and builds upon concepts in a logical manner. With each additional layer of the tool comes additional interactivity and opportunities for creative thought from the student. We have focused here on a topic from the Operating Systems section of the computer science curriculum.

We designed our tool to be as accessible as possible by implementing it in a web environment. No installation, specialist equipment, or specialist knowledge is required to explore these concepts, unlike in traditional Operating Systems courses. Often students explore process management using test servers at their institution. Such experimentation is susceptible to failure, and mistakes carry the potential to cause considerable damage. This tool removes some of the risks of experimentation whilst still allowing students a level of creative thought.

We tested the tool with 29 undergraduate students who took a test before and after a session. A paired t-test on these scores showed a significant learning effect,

providing our methodology an initial validity. In addition to this, students reported that they found the tool helpful and enjoyable to use. This combined information highlights that we met our design goals and produced a tool that provided a safe *environment* for students to explore systems topics, that was both *educational* and *entertaining*.

B. Applications

The primary application of this work is to integrate this tool, and other similar tools, into a computer science curriculum. We would propose targeting thought experiments similar to the Dining Philosophers Problem which can help alleviate common problems encountered in immediate, experiential learning such as affecting hardware availability in the case of thread management as covered by the Dining Philosophers Problem.

This tool has applications not only in undergraduate institutions but also in high schools where the curriculum is still developing and specialist equipment such as testing servers may not be available. While we have not yet tested the tool with high school students, initial results suggest that this tool is beneficial to students encountering these topics for the first time and thus has potential applications in a high school classroom. Alterations would most likely be needed to adapt to the full K-12 curriculum.

C. Limitations

The tool itself is limited by the method of implementation. Web environments do not afford the freedom of more traditional programming environments. This is a limitation that is accepted due to the broad accessibility that web-based systems provide. The tasks also follow a linear path through the concepts of Bloom's Taxonomy. This was by design, however it may limit some learners. The static design is also a factor when considering the code blocks and the game. For example, in the game there are always five philosophers and in the coding the basic framework remains the same. This could be expanded upon in the future.

While the experiment presents a learning effect, there are limitations to be acknowledged. When considering a niche topic such as the Dining Philosophers Problem there is not a wealth of questions within the scope. While designing our test we had two goals: (1) to include questions relevant to the topics we were teaching and (2) to avoid testing the same concept twice. To expand our question pool further would have had an impact on one or both of those goals. A further limitation is that the methods presented make no attempt to evaluate long term retention of information. This is beyond the scope of the present work.

Experimentation with this tool was limited by a small participant population. While we show encouraging results, further investigation is required to insure that these results generalize to larger populations of students across multiple institutions. This tool is initially designed for undergraduate students, however there are potential applications in K-12 as the curriculum expands.

D. Future Work

This project invites several possibilities for future work. First, we hope to expand the tool to include other topics in a similar style, forming a chapter-based system where each chapter is a topic, in this case the Dining Philosophers Problem, and follows Bloom's Taxonomy to build concepts as we have done here.

In line with the learning literature, we would also look to further personalize the tool. This could take a number of different forms. For example, we could ask the student's name at the beginning of the session and then address all feedback accordingly or we could consider a more complex personalization such as converting the software to an Interactive Tutoring System and maintaining a student model throughout the session. In this personalization, all responses would be recorded by the system and would have direct influence over future sections so that the tool may best address the weakness of the current student. This presents a technological challenge, but has been shown to produce strong results in previous cyber-learning environments [9]. There is also the potential to use further sensors such as facial features and eye tracking to better understand the user experience and adapt accordingly [17, 18].

Initial testing was performed with undergraduate students, one avenue of future work would be to repeat the experiment, possibly with a further refined tool, with high schoolers. This would allow us to further investigate the learning effect as well as gain a better understanding as to whether tools such as this would be useful in the K-12 computer science curriculum.

VI. Concluding Remarks

In this work we have addressed an upcoming need in computer science education through the creation of an accessible, interactive, web-based learning tool. Based upon curriculum guidelines set out by ACM [6] and CSTA [7], we have implemented the operating systems topic of process management via the Dining Philosophers Problem as our first course module using this web interface. The educational effectiveness of this tool has been demonstrated by undergraduate students enrolled in computer science courses. We have also shown overall that the tool is entertaining, and some sections were very popular amongst undergraduate students. There is plenty of opportunity to pursue future work now that our concept has been tested and verified.

VII. Acknowledgements

We would like to thank Dr. Aaron Striegel for giving us the initial push toward pursuing Placating Plato with Plates of Pasta as a research project. We also thank Dr. Douglas Thain, Dr. Aaron Striegel, and Dr. Scott Emrich for their advice and support of our independent work.

	Function	Goal	Bloom's Taxonomy
Introduction	Introduces the problem using the common analogy as well as introducing two key concepts (deadlock and starvation) via animated examples that they will need going forward.	Introduces the problem. Shows a visual representation of the problem. Introduces visual motif that will be used going forward.	Remember
Real-World	Gives the users an explanation of what deadlock and starvation mean in concurrency.	Lets students understand the two situations that could cause concurrency to fail.	Remember Understand
Identify	Provides the user with three real-world situations, and lets them identify whether each situation is an example of deadlock, starvation, or neither.	Tests students' ability of identifying the deadlock and starvation situations based on the real-world examples.	Apply
Game	Requires the player to manage the order in which philosophers ate while preventing any from starving. Scoring feature allows the student to compare their performance to their previous attempts.	Offers students a chance to apply solutions to the Dining Philosophers Problem in a novel way.	Apply
Solutions	Demonstrates the resource hierarchy and arbitrator solutions to the Dining Philosophers Problem. Provides two step-by-step animations for students to click frame-by-frame through how each solution applies to the problem.	Provides students with two solutions to the problem in broad strokes as a baseline for judging their own solutions in the next section.	Understand Apply
Code-blocks	Presents starting pseudocode with toolboxes on the side with the pseudocode necessary to solve the Dining Philosophers Problem. The students are given feedback after each attempt via error messages and an animation showing their code implemented.	Gives students time to interact with the problem hands-on and craft their own solution given some starter code.	Evaluate Create
Compare	Provides students with two solutions to the problem in a similar pseudocode to what they have seen before. Students must then select which is the better solution. The page provides feedback based upon their choices.	Further develops students' application of what they have learnt in a computer science context. Develops students' ability to analyze and evaluate code and make decisions based on what they have learnt.	Analyze Evaluate

Fig. 8: Overview of sections included in the tool, their primary goal, and relation to Bloom's Taxonomy.

References

[1] K. Taylor and C. C. Miller, "De Blasio to Announce 10-Year Deadline to Offer Computer Science to All Students," *The New York Times*, Sep. 2015. [Online]. Available: http://www.nytimes.com/2015/09/16/nyregion/de-blasio-to-announce-10-year-deadline-to-offer-computer-science-to-all-students.html

[2] Associated Press, "Largest school districts to embrace computer science." [Online]. Available: http://www.sfgate.com/nation/article/Largest-school-districts-to-embrace-computer-5943380.php

[3] B. Gardiner, "Adding Coding to the Curriculum," *The New York Times*, Mar. 2014. [Online]. Available: http://www.nytimes.com/2014/03/24/world/europe/adding-coding-to-the-curriculum.html

[4] J. Pierce, "Introducing alice to a squeak wonderland," in *Creating, Connecting and Collaborating Through Computing, 2003. C5 2003. Proceedings. First Conference on*, Jan 2003, pp. 40–43.

[5] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, Nov. 2009.

[6] A. f. C. M. A. Joint Task Force on Computing Curricula and I. C. Society, *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York, NY, USA: ACM, 2013, 999133.

[7] D. Seehorn, "K-12 computer science standards–revised 2011: The csta standards task force," *ACM, October*, 2011.

[8] 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 Bull.*, vol. 35, no. 2, pp. 131–152, Jun. 2002.

[9] A. Olney, S. D'Mello, N. Person, W. Cade, P. Hays, C. Williams, B. Lehman, and A. Graesser, "Guru: A computer tutor that models expert human tutors," in *Intelligent Tutoring Systems*, ser. Lecture Notes in Computer Science, S. Cerri, W. Clancey, G. Papadourakis, and K. Panourgia, Eds. Springer Berlin Heidelberg, 2012, vol. 7315, pp. 256–261.

[10] E. C. Anderssen and C. P. H. Myburgh, "The acquisition of operating systems concepts by computer science students," *Comput. Educ.*, vol. 19, no. 3, pp. 309–320, Oct. 1992.

[11] C. Kann, R. W. Lindeman, and R. Heller, "Integrating algorithm animation into a learning environment," *Comput. Educ.*, vol. 28, no. 4, pp. 223–228, May 1997.

[12] M. R. Regalado, E. Aranha, and T. R. da Silva, "Gamifying an online approach for promoting game development learning and contest: An experience report," in *Frontiers in Education Conference (FIE), 2016 IEEE*. IEEE, 2016, pp. 1–8.

[13] M. J. Mateos, P. J. Muñoz-Merino, C. D. Kloos, D. Hernández-Leo, and D. Redondo-Martínez, "Design and evaluation of a computer based game for education," in *Frontiers in Education Conference (FIE), 2016 IEEE*. IEEE, 2016, pp. 1–8.

[14] E. Thompson, A. Luxton-Reilly, J. L. Whalley, M. Hu, and P. Robbins, "Bloom's taxonomy for cs assessment," in *Proceedings of the tenth conference on Australasian computing education-Volume 78*. Australian Computer Society, Inc., 2008, pp. 155–161.

[15] H. Coffey. Bloom's taxonomy. [Online]. Available: http://www.learnnc.org/lp/pages/4719

[16] L. Camera, "States look to expand computer science classes," Dec 2015. [Online]. Available: https://www.usnews.com/news/articles/2015/12/10/states-look-to-expand-computer-science-classes-in-k-12-schools

[17] S. Hutt, C. Mills, S. White, P. J. Donnelly, and S. K. D'Mello, "The eyes have it: gaze-based detection of mind wandering during learning with an intelligent tutoring system," in *Proceedings of the 9th International Conference on Educational Data Mining, International Educational Data Mining Society*, 2016, pp. 86–93.

[18] S. D'Mello, K. Kopp, R. E. Bixler, and N. Bosch, "Attending to attention: Detecting and combating mind wandering during computerized reading," in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '16. New York, NY, USA: ACM, 2016, pp. 1661–1669.