Simr: A 3D virtual reality instructional tool for novice programmers

Nyala Jackson
njackson@vassar.edu
Vassar College
Poughkeepsie, New York

Monica Anderson
anderson@cs.ua.edu
The University of Alabama
Tuscaloosa, Alabama

Keanna Wright
kmwright1@aggies.ncat.edu
North Carolina Agricultural
Technical State University
Greensboro, North Carolina

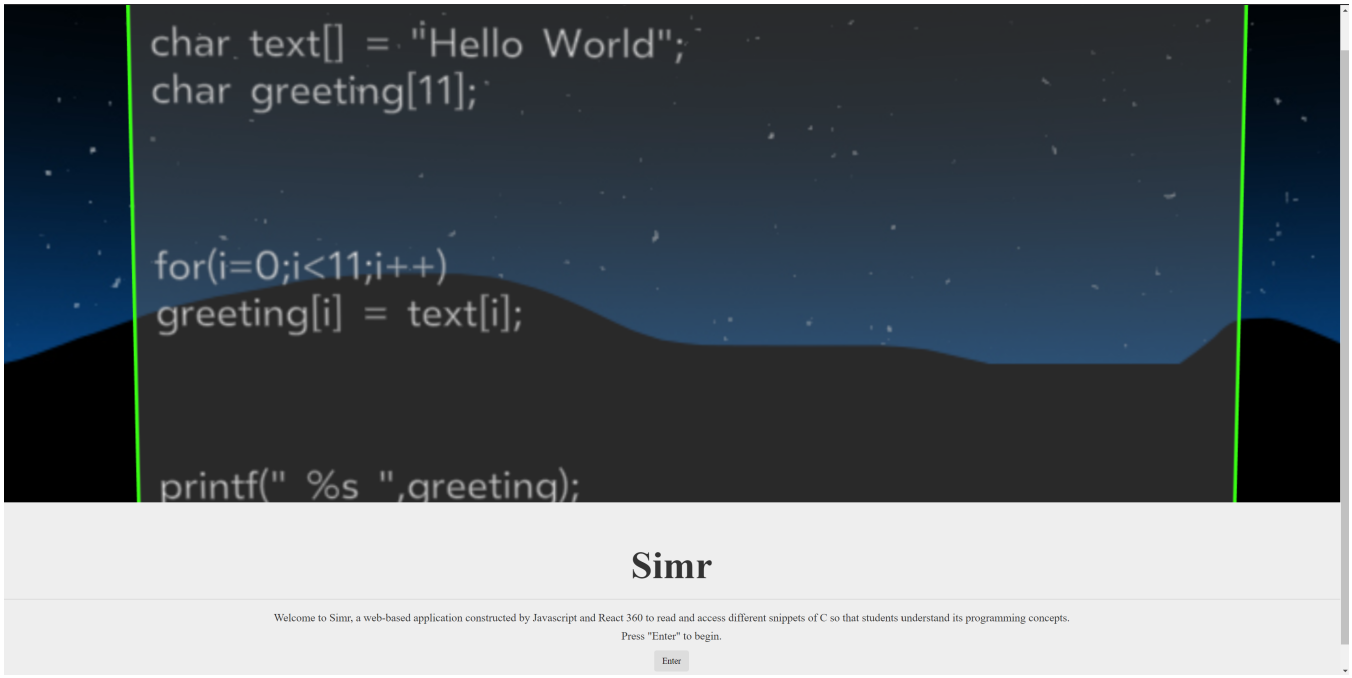

Figure 1. Simr Loading Screen

Abstract

Contemporary computer science lectures fail to provide students with a deep, functional understanding of runtime dynamics because of their heavy reliance on metaphors to explain abstract concepts. In this paper, we propose an immersive virtual reality application that avoids teaching incorrect notional machines by modeling the runtime dynamics of a sample program via memory tracing. The main objective of this study is to create a platform in which students do not have to encounter cross-cultural metaphors that may inhibit their understanding of C and its memory management. Our proposed interactive model highlights memory management in C programs by displaying the stack, heap, and the variable locations of the given program. The results of our proposed experiment would show that our learning model is not only immersive but teaches students a correct notional machine of C.

Keywords: notional machine, virtual reality, memory tracing

1 Introduction

Virtual reality (VR) is slowly captivating the world of technology. Through VR applications like Beat Saber, a rhythm video game, and VR Chat, a social platform, the power that lies in virtual reality is shown; both envelop the user in a 360° active, engaging environment. It's because of these attributes that intrigued professors and students to begin looking at it as an instructional tool. Students can immerse themselves in learning environments and gain experiences through active learning.

In this paper, we utilize the opportunities within virtual reality for educational purposes as we create an application focused on ensuring people of color (POC) students are not at a disadvantage from misunderstanding cross-cultural metaphors during lectures, and can effectively learn programming concepts through visualization.

Simr is an educational 3D VR instructional tool that shows students a visual representation of the changes in memory during certain parts of the program. This version of Simr is

a web-browser based program that enables students to use their computers and phones, in addition to VR headsets. The other version of Simr is a Unity application. For this paper, we will be focusing on the web-based version, which is more accessible to institutions with limited resources.

Simr attempts to solve the disparity of POC in computer science by ensuring their lack of interest in computer science is not because of subject confusion or cross-cultural dependent issues.

This research impacts current and future computer science students who struggle to learn solely from lectures, and professors who would like more supplementary programs for their students. Professors are most interested in this research because it would supplement their courses and aid in the education of future computer scientists. Without this research, computer science programs utilizing memory tracing and developing proper notional machines of C may be created, but the issue of unclear cultural references will remain, and POC computer scientists may remain confused.

Current research does not focus on using VR in C introductory computer science courses, and current researchers focus more on getting people interested in computer science, not on maintaining their interest and strengthening core C concepts. Their solutions don't provide a program that visually shows the changes in memory during program execution and don't offer such programs in easily accessible formats. This new solution provides a break from the abstractness of learning memory tracing and developing a correct C notional machine. Simr also benefits students because they will gain a correct underst-anding of memory utilization and will easily grasp programming concepts as they understand how their code affects and changes memory.

2 Related Works

For context, the concepts of program tracing, notional machines, and immersive virtual reality in education are discussed .

2.1 Program Tracing

Program tracing is a debugging process that records and logs information about a program. Simr uses programming tracing in the form of memory visualization. Program memory visualization offers students the ability to see and understand how and what computers store in the stack and heap as programs run.

Article [5] discusses the concept of *program memory traces*, a trace format that includes abstract represent-ations with the program's stack and heap. By providing a visual model and reinforcing it with in-class activities in which students' trace code, Hertz and Jump hoped the students would develop more viable mental models of how the program executes. By organizing the classes around traces, they found that visualizations provide improvements to student learning throughout a course.

Though Hertz and Jump did not see a statistically significant increase in the lecture grades (though were strong), they saw a notable improvement in their students' lab results, which supported their hypothesis.

We hoped to achieve similar positive results in our study, for the use of program tracing has been proved to essential in providing an accurate visualization of C memory dynamics.

2.2 Notional Machines

A notional machine is an abstract model of a computer system and how it interacts with a programming language. Articles [1], [8], [4], and [6] discuss the effectiveness in students developing a correct abstract mental model of a computer. If a student can visualize a correct model, this means that they understand the material, specifically the rules of their programming language and how it interacts with a computer.

It is not easy to develop a correct notional machine, Article [6] demonstrates that through their qualitative results as the students (overall) failed to produce a fully correct notional machine even when the professor taught them about program memory tracing. This paper proved that lectures enriched with program memory tracing is insufficient to ensure the students attain proper constructions of notional machine mental models. They state that their study also suggests that lectures are the main cause for students' lack of understanding in notional machine concepts and are insufficient in guaranteeing that students acquire a correct understanding of the notional machine of a programming language. Simr wishes to be the application that helps supplement lectures and labs to improve students' overall understanding of the C notional machine by using program memory tracing to foster experiential learning. Simr aids the development of a correct notional machine of C via its trace through a given program; specifically, the visual memory trace supplements students' understanding of runtime dynamics.

2.3 Immersive Virtual Reality in Education

The concept of using virtual reality in education is not new. Articles [2],[7],[3], demonstrate the capabilities of utilizing virtual reality to create an informative, immersive environment to teach students.

Article [2] focuses on using VR to create immersive, simulated, personalized learning paths for students. Mainly focusing on eighth-grade biology and chemistry subjects, their program allowed the students to virtually stand inside a machine and see the functions of DNA and bio-molecules from a microscopic level. The students virtually traveled inside a DNA strand to view and learn how they are replicated in the human body. Bhattacharjee et al. had three cases: one study on a group of 100 students, another study on a group of 60 students diagnosed with Attention Deficit Hyperactivity Disorder (ADHD), and the other study on a group of 30 students with Audio Processing Disorder (APD), a hearing

disorder where the student is not able to process the words they hear. All cases resulted in a significant increase in learning effectiveness, retention, creativity, and practical implementation in the students, who were tested via a quiz. They noted a 116.25% increase when both processes (the traditional learning model and the proposed learning model) were combined. The development of this program showcases the incredible capabilities of VR in education. Simr desires to see such an increase in grades when traditional lectures and labs are combined with the program.

Article [3] demonstrates the abilities of educational virtual reality by developing a 3D game that is focused on garnering interest in computer science through puzzles that teach core programming concepts such as arrays, loops, and if statements. Though Bolivar and Perez were unable to test their program, the potential for experiential learning using virtual reality is proven. The creation of this program inspires the development and highlights the potential of Simr as a resource for experiential learning in the classroom.

Article [7] further analyzes the potential of virtual reality in supporting experiential education. Jantjirs et al. focus on examining virtual reality in dentistry and how VR applications like DentSim and The Virtual Dental Patient gives students the practical experience of dentistry in a realistic environment. They note how experiential learning requires a student to conceptualize and actively experience material, and they describe how VR and AR (Augmented Reality) show pedagogical promise of deepening students' learning experiences. This paper proves the effectiveness of using VR technology as way to educate students.

3 Methodology

3.1 Approach

Current approaches fail to consider the effect unapparent metaphors have on the education of people of color in computer science, and they do not offer easy, accessible ways to use their applications. Inspired by the struggles of being a person of color (POC) computer scientist and the linguistic barriers they have to cross to attain correct notional machines, we developed Simr. Simr, our interactive visual C model which is available on browser or as an application, goes through the stack and heap changes of an example piece of code as each line runs. By displaying the alterations to memory as the example code runs in a virtual reality environment, students immerse themselves not only in the world of virtual reality but also in a computer process.

Similar approaches to Simr have had success in other areas, such as medicine and educational theory, these programs successfully created and implemented ways to explore the uses of program memory tracing and virtual reality immersion in preparing students for their field of study. Because of the past successes in similar experiments, we are optimistic on the success of Simr. This version of Simr is a web-based application constructed by Javascript and React 360 that was designed to read and access different snippets of C so that students understand its memory management concepts. We chose C because it is one of the main programming languages taught in novice programming classes (alongside Java). With this new approach, POC students will have a resource that illustrates the abstractness of memory management.

3.2 Creation of Simr

Simr was developed using React as a user-interface building Javascript library, React 360 as the 360° web-browser capable library, Javascript as the language, and Atom as the integrated development environment (IDE). The program consists of a starting page and a 360 web page that includes three panels: the code panel, variable panel, and stack panel.

The start loading page consists of a picture of the code panel, a quick introduction, and an "Enter" button to begin the program (See Figure 1).

The code panel contains the example C program used in the prototype. Users can interact with it by selecting highlighted code snippets that change memory and seeing the effects each snippet has on the stack.

In this version of Simr, we used a JavaScript Object Notation (JSON) file to import the code and its states into the program (See Fig. 2).

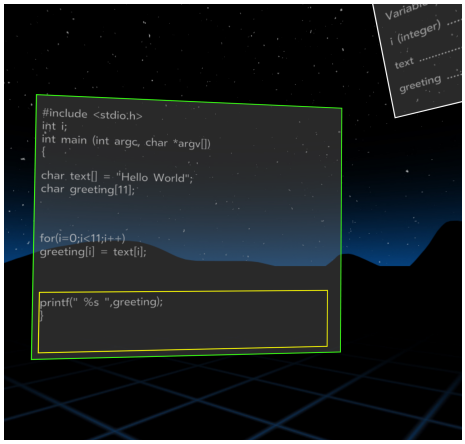

Figure 2. Simr React 360 Code Panel

The variable panel contains all the initialized and declared C variables in the program and the variables' location in the stack (See Fig.3).

The stack panel contains the stack in the form of a grid ranging from 0 to 100 squares. Depending on which code snippet is selected, the state of the grid will change to reflect the memory state after that code snippet has run (See Fig.4).

The sample C program with an unapparent memory bug is shown in Program 1.

The program works by splitting the attained program into selectable snippets — sections of executable code that

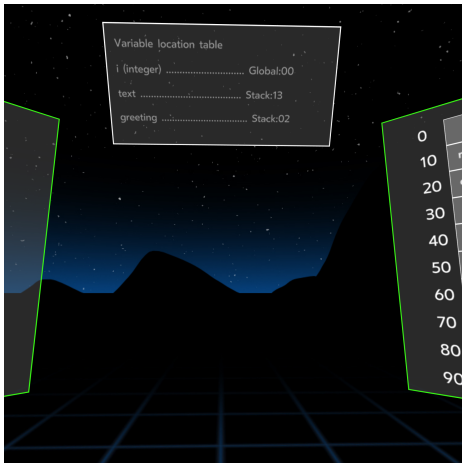

Figure 3. Simr React 360 Variable Panel

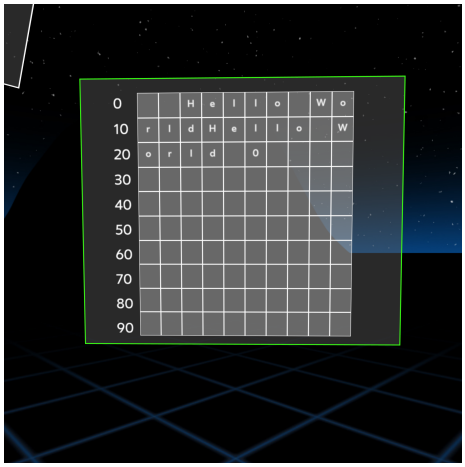

Figure 4. Simr React 360 Memory Panel

```
#include <stdio.h>
      int i;
      int main (int argc, char *argv[])
        {
        char text[] = "Hello World";
        char greeting[11];
        for(i=0;i<11;i++)
            greeting[i] = text[i];
        printf(" %s ",greeting);
      }
```

Program 1. The example program used in Simr

changes memory — on the code panel and selecting which code snippet/memory state the user would like to see be loaded into the stack (Figure 4). With that selection, the memory panel will change its output based on the user-selected code snippet, revealing the state of the stack after that snippet of code has been run. Currently, the user can choose between four different code snippets from the JSON file that contains such states and their outputs. This grants the user the ability to easily switch between states without them needing to be accessed in order.

The completed version of the JSON file will also include the variables used and their location in the stack. Like the code panel and the stack panel, the variable panel will attain its information from the JSON file to avoid being hardcoded.

4 Experimental Design

Unfortunately, due to the circumstances of COVID-19, we were unable to execute an experiment to test the success of Simr, so we will describe how our potential experiment would take place.

Ideally, we would have five randomly-selected colleges and select two sections of introductory computer science classes from each of them. We would have one introductory section be the control group - those learning without Simr — and the other be the experimental group — learning with Simr. Classes using Simr would interact with the program by using it in and out of class for code demonstrations and examinations. For instance, a student may use it to help them understand why their code for a lab is not working after using it in class over a sample piece of code. Our approach to the experiment would be similar to the approach described by the article [5] in which classes would focus on trace-based teaching to supplement their program.

We would have all classes take a test at the end of their courses to approximate the effectiveness of our supplemental program based on the students' understanding of core C memory management concepts, specifically memory management. We would compare the scores of the different computer science introduction courses for each university, comparing the test scores of the classes that used Simr and the classes that didn't use Simr.

A post-experiment survey would also be developed to determine the interest in using programs like ours in the future. The survey would consist of Likert scales and open-ended questions that ask the students' experiences with virtual reality. Some sample questions would include:

- If you used Simr, how likely are you to use this program in the future?
- How much previous experience have you had with using VR?
- If you used Simr, how helpful did you find the program to be? Why?
- If you haven't used Simr, would you be interested in a supplementary virtual reality program?
- If you could change some things in your class, what would they be?

We would also collect demographic data on the students such as age, gender, race, and location to test the success of the program among minorities. Asking questions like:

- How often do you experience intelligent system failures such as erroneous facial recognition or automatic speech recognition? For example, when Siri/Alexa/Bixby/Google does not understand you or misunderstands what you said.
- How often does your learning material include metaphors to explain abstract concepts? For instance, how did your professor explain variables to you?

With this information, we would be able to identify how helpful programs like Simr are for novice programmers and professors, especially for minorities.

5 Analysis

The results from the test would reveal that the classes that used Simr performed better on the tests than the classes that did not. We would compile the test results into a bar graph with two classes per university with and without Simr (See Experimental Table 1).

Additionally, the results from the survey would show that introductory students enjoyed Simr and would use the program (and similar ones to it) in the future.

Table 1. Test Average

University	Intro Course	Test Average	Simr Use
Fake Uni	CMPU 101-01	86	No
Fake Uni	CMPU 101-02	90	Yes
Fake Uni_2	CMPU 106-01	84	No
Fake Uni_2	CMPU 106-02	94	Yes

We expect the results of the test to show that those who utilized Simr had a better understanding of C runtime dynamics and abstract memory management concepts.

6 Future Work

The next steps for this research project are to finalize the application, test its capabilities on students, create the test and survey, and add any improvements needed (attained from the survey).

7 Conclusion

We proposed a web-based application focused on program memory tracing. This application is designed to help POC students learn memory concepts in C through an immersive virtual reality environment without abstract metaphors and unclear references. With the combination of Simr and in-class lectures, we think that students will understand C runtime dynamics and maintain their interest in computer science. The results from our experiment that relates to our hypothesis show that the students who were in the classes that used Simr had higher lecture and lab grades in addition to higher post-test scores. These results imply that with Simr, students develop a cohesive mental model of computer memory management during C programs that help them understand how to code efficiently.

References

[1] Michael Berry and Michael Kölling. The design and implementation of a notional machine for teaching introductory programming. In *ACM International Conference Proceeding Series*, pages 25–28. Association for Computing Machinery, 2013.

[2] Deblina Bhattacharjee, Anand Paul, Jeong Hong Kim, and P. Karthigaikumar. An immersive learning model using evolutionary learning. *Computers and Electrical Engineering*, 65:236–249, jan 2018.

[3] Santiago Bolivar, Daniel Perez, Armando Carrasquillo, Adam S. Williams, Naphtali D. Rishe, and Francisco R. Ortega. 3D Interaction for Computer Science Educational VR Game. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11572 LNCS, pages 408–419. Springer Verlag, jul 2019.

[4] Paul E. Dickson, Neil C.C. Brown, and Brett A. Becker. Engage Against the Machine: Rise of the Notional Machines as Effective Pedagogical Devices. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, pages 159–165, 2020.

[5] Matthew Hertz and Maria Jump. *Trace-Based Teaching in Early Programming Courses.* 2013.

[6] Jeisson Hidalgo-Céspedes, Gabriela Marín, and Vladimir Lara-Villagrán. Understanding Notional Machines through Traditional Teaching with Conceptual Contraposition and Program Memory Tracing. *CLEI Electronic Journal*, 19(2):1–17, 2016.

[7] Mmaki Jantjies, Trevor Moodley, and Ronel Maart. Experiential learning through virtual and augmented reality in higher education. In *ACM International Conference Proceeding Series*, pages 42–45, 2018.

[8] Juha Sorva. Notional machines and introductory programming education. *ACM Transactions on Computing Education*, 13(2):31, 2013.