VizCode: A Practical Real-time Tool for In-Class Computer Programming Tutoring

Yinuo Yang
University of Michigan
Ann Arbor, Michigan, USA
inon@umich.edu

Steve Oney
University of Michigan
Ann Arbor, Michigan, USA
soney@umich.edu

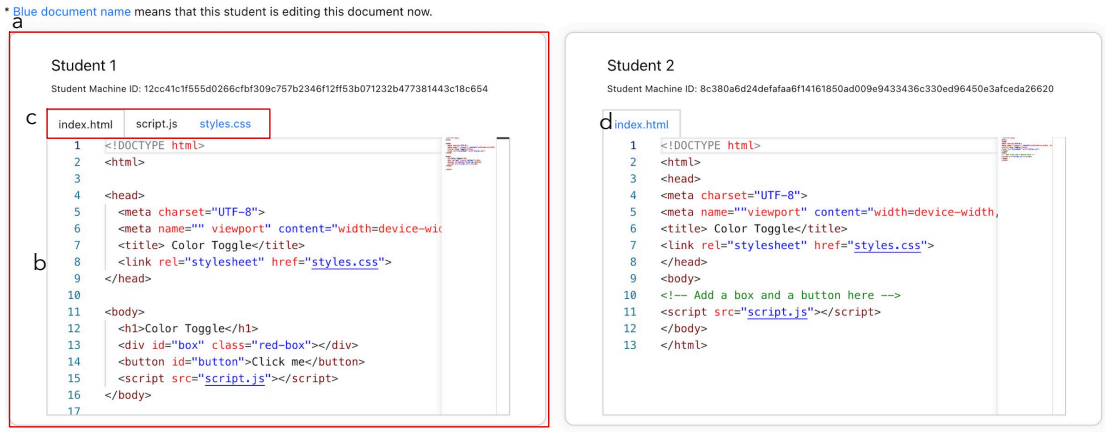

Figure 1: The VizCode Web Page interface consists of student boxes (a), code boxes (b), and document tags (c). This screenshot shows two code boxes with several code boxes in them, each displaying one student's code in real-time from the student's VSCode. Documents under editing are displayed in blue (d), indicating that the student is currently editing this document.

ABSTRACT

Prior research has shown the benefits and promise of allowing instructors in large programming classes to monitor students' coding activity in real-time. However, translating these findings into practical, user-friendly tools remains a challenge. This demonstration showcases VizCode, a tool that allows instructors to monitor students' code in real-time as they edit in the popular Visual Studio Code (VSCode) IDE. VizCode is designed to be practical (integrating with widely-used tools and requiring minimal server overhead), scalable (minimizing network latency by only communicating code changes), and easy to use (requiring minimal setup from students). By focusing on practicality and seamless integration with VSCode, VizCode bridges the gap between research and practice, making it easier for instructors to monitor students' code in real-time.

CCS CONCEPTS

• **Applied computing** → **Education**; • **Human-centered computing** → *Interactive systems and tools.*

KEYWORDS

programming education at scale; computer science education

ACM Reference Format:
Yinuo Yang and Steve Oney. 2024. VizCode: A Practical Real-time Tool for In-Class Computer Programming Tutoring. In *Proceedings of the Eleventh ACM Conference on Learning @ Scale (L@S '24), July 18–20, 2024, Altanta, GA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3657604.3664716

L@S '24, July 18–20, 2024, Altanta, GA, USA.

ACM ISBN 979-8-4007-0633-2/24/07
https://doi.org/10.1145/3657604.3664716

1 INTRODUCTION

Real-time monitoring of students' coding progress is useful to offer tailored feedback and make informed decisions regarding exercise progression in a programming class at scale [3]. Prior research has explored ways for monitoring students' real-time code [1–4]. However, these tools require working within ecosystems and online environments that are specific to these tools, which may reduce the scope of programming courses where they can be used.

To help bridge, the gap between research and practice, we introduce VizCode—a practical, real-time tool designed to observe students' code as they edit it within the Visual Studio Code (VSCode) environment on their own computers. VizCode effectively bridges the gap between the need for real-time monitoring and the widespread use of VSCode among programming students.

2 TOOL DESIGN

VizCode consists of three components: (1) a *VSCode extension* that students can install locally, (2) a *data server* that the VSCode extension communicates with (this can either be an external server or

run locally on an instructor's computer, and (3) a *dashboard* that allows instructors to monitor students' code in real-time. We plan to release VizCode as a free and open-source tool available to any instructor.

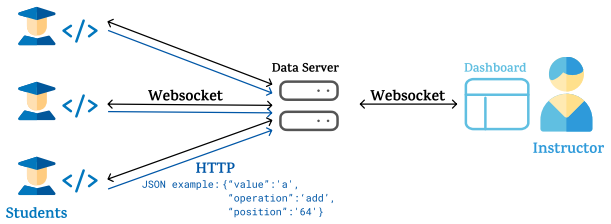

Figure 2: VizCode workflow. Each student's editing data would send to the VizCode Web Page in real-time.

2.1 VSCode Extension

As a first step, students can install the VizCode extension for VSCode from the built-in Extension Marketplace[1]. This extension has a minimal User Interface, only notifying students if and when their code modifications are being tracked.

Students are not required to do anything other than install the extension; there is no additional configuration or setup. Instead, most of the configuration is done by *instructors*—to specify where (which server) data should be logged to and which files to keep track of along with links to the private policies for every exporter. These configuration options can be specified in a `.vscode` configuration file that instructors can distribute alongside assignments to students. When students first start the extension, the extension will send them links to the private policies to handle the consent. As students make code modifications to the relevant directories, this extension will communicate code *changes* to the data server in real-time.

2.2 Data Server

The *data server* is a Node.js application that listens for code changes and distributes it to connected "dashboard" clients. The data server connects with students' VSCode extensions and dashboard through Web Sockets. It is lightweight enough that it can run on any standard modern laptop, meaning that instructors can host it locally. In situations where this is not practical (for example, if the instructor's computer is on a different network or hidden behind a firewall), the data server can instead run on a cloud service (we have tested with AWS).

2.3 Dashboard

The VizCode *dashboard* connects with the data server to offer a user-friendly platform for monitoring students' coding progress. As students edit their code, their code updates are immediately reflected in the dashboard. Figure 1 shows its main structure:

- Figure 1.a: Student boxes, showing a student's real-time documents within VSCode.
- Figure 1.b: Code boxes, showing student's code (using the Monaco web-based code editor).
- Figure 1.c: Document tags, the instructor can click the tag to see codes in different documents.
- Figure 1.d: Document name color would be blue when this student is editing this document.

[1]https://marketplace.visualstudio.com/items?itemName=educational-technology-collective.telemetry

3 USING VIZCODE

To use VizCode, instructors must (1) tell students to install the VizCode VSCode extension (this can happen once, at the beginning of the semester), (2) set up a data server, either locally or a cloud server, and (3) distribute assignments with a `.vscode` configuration file that points to the data server. Once configured, instructors can visit the dashboard to monitor students.

Student names are 'anonymized' by default but if instructors need to identify students, they can request that they add their name as a comment in their codebase. When an instructor identifies a student who is struggling or would benefit from additional help, they can contact these students through established communication channels (e.g., talking to the student in-person or through video-conferencing tools). Future versions of VizCode could be modified to to allow instructors to send messages to students directly in the dashboard [1, 2].

4 DEMO SETUP AND REQUIREMENTS

We plan to focus our demonstration on how instructors can set up VizCode to monitor students' coding activity. We will also briefly walk through how students can enable the VizCode extension on their machines. We will use simulated student data to emulate the experience of working with VizCode in a mid-size classroom for instructors. Our demo requires minimal hardware (as does VizCode). Visitors will optionally be allowed to experiment with our demo, either as a 'student' or 'instructor'.

5 LIMITATIONS

One limitation is the handling of streamed data containing sensitive information. While the extension includes functionality to ignore data within the `.env` file, there remains a risk of data leakage. Another limitation is the lack of robust security protocols. One way to address this would be to add optional authentication through systems such as Canvas' LMS.

6 CONCLUSION

This demo presents VizCode, showing a practical way to collect and use data from programming classes in real time to facilitate programming education at scale, thereby empowering instructors to provide timely assistance and personalized feedback within the lecture setting. We will continue implementing our system. Future work includes LLM (Large Language Model) assistant teaching and better visualization of students' code-writing progress.

ACKNOWLEDGMENTS

The authors thank Mengyan Wu and Chris Brooks for their work on the VSCode Telemetry extension, which VizCode uses, and Ashley Zhang for her feedback. This material is based on work supported by the National Science Foundation under DUE 1915515.

REFERENCES

[1] Philip J. Guo. 2015. Codeopticon: Real-Time, One-To-Many Human Tutoring for Computer Programming. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (Charlotte, NC, USA) *(UIST '15)*. Association for Computing Machinery, New York, NY, USA, 599–608. https://doi.org/10.1145/2807442.2807469

[2] April Yi Wang, Yan Chen, John Joon Young Chung, Christopher Brooks, and Steve Oney. 2021. Puzzleme: Leveraging peer assessment for in-class programming exercises. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–24.

[3] Ashley Ge Zhang, Yan Chen, and Steve Oney. 2023. VizProg: Identifying Misunderstandings By Visualizing Students' Coding Progress. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (, Hamburg, Germany,) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 596, 16 pages. https://doi.org/10.1145/3544548.3581516

[4] Ashley Ge Zhang, Xiaohang Tang, Steve Oney, and Yan Chen. 2024. CFlow: Supporting Semantic Flow Analysis of Students' Code in Programming Problems at Scale. *arXiv preprint arXiv:2404.10089* (2024).