Engineering Educators Bringing the World Together
2025 ASEE Annual Conference & Exposition
Palais des congrès de Montréal, Montréal, QC • June 22–25, 2025 ASEE
Paper ID #47992

Experiences Using Live Streaming as an Informal Learning Tool in the Formal Classroom

Ella Kokinda, Clemson University

Ella Kokinda is a PhD candidate at Clemson University's Zucker Family Graduate Center in Charleston, South Carolina. Her research surrounds live streaming, software and game development, and developer communities.

Dr. D. Matthew Boyer, Clemson University

Dr. Boyer is a Research Associate Professor in the Department of Engineering and Science Education and an Educational Proposal Writer in the College of Engineering, Computing and Applied Sciences.

Paige Rodeghero, Clemson University

Experiences Using Live Streaming as an Informal Learning Tool in the Formal Classroom

Abstract

Despite growing demands for software development skill in the professional job market, companies are finding that students lack necessary programming and soft skills they deem necessary directly out of college. Given this, we propose a novel educational approach using live streaming as a means of giving students the opportunity to gain practical experience and knowledge about subjects that interest them and their subsequent professional interests.

In this work, we describe our experiences creating an undergraduate computer science course where students live stream software and game development projects of their choosing weekly over the course of the semester. The course was conducted over two semesters: an initial pilot, followed by a refined iteration incorporating lessons learned and student feedback.

In both iterations of this course, students live stream for a set amount of hours each week while maintaining a diary of their accomplishments and how they felt their individual streams went. We evaluate the students on their perceived self-efficacy and the evolving perceptions of their goals and desired achievements during this course through three reflection assignments.

Our observations reveal that students initially took the course to set aside time to work on personal projects and develop their programming skills, with motivations changing throughout the semester and often scaling back from original larger scope projects and goals. Following the pilot semester, we analyzed student recommendations and reflected on the course structure and outcomes. This reflection informed targeted interventions and improvements implemented in the subsequent semester. We present an analysis of both course iterations, and highlight the impacts of these interventions on student experiences and learning outcomes. Additionally, we analyze and discuss students' perceived self-efficacy in programming and live streaming skills and find that student self-efficacy decreases slightly. Additionally, we found that students may not continue to stream after the course, the experience may provide valuable professional development opportunities and help students bridge theoretical knowledge with practical applications of software development.

We conclude by discussing lessons learned from this two-phase implementation and proposing future research directions in educational live streaming for computer science students. Our iterative approach also demonstrates the potential for continuous improvement in innovative educational methodologies.

Introduction

In the post-pandemic era, we see many necessary changes in how we approach computer science and, more specifically, software development education to ensure student retention and success. Current research shows that undergraduate students lack soft skills like communication and other necessary professional skills upon graduation [1, 2, 3]. In addition, students are experiencing more anxiety and computer science culture-related challenges like personal obligations, lack of sense of belonging, in-class confusion, and lack of confidence, even more so when from an underrepresented group [4]. Given the challenges we see in student populations, the current research on increasing student success, and the rapidly changing nature of computer science and software development technologies, we believe it is time to take an informal approach to formal education pedagogy. Computer science students benefit from informal learning environments that allow them to apply theoretical concepts in practical contexts while building upon their previous learning experiences [5]. Often, these opportunities are outside of the formal classroom in the form of summer camps, hackathons, co-ops and internships, and other service learning activities [6, 7, 8]. Another potential candidate for informal learning opportunities is live streaming [9]. Live streaming is an increasingly popular medium for a behind-the-screen look in software development, where streamers and viewers share their knowledge and experiences as they happen in real time. Popular platforms like Twitch and YouTube enable developers to stream live coding sessions where people worldwide can engage in real-time collaboration and receive feedback, knowledge sharing, and skill development.

While there may be no one-solution-fits-all approach to fix every problem, this work aims to use live streaming as an informal learning opportunity within the formal classroom to aid in bridging the gap for student self-efficacy in programming skills, confidence, and communication skills while also providing a low-pressure environment to learn. We propose continued work in software development live streaming research by providing an opportunity for students in computer science curriculum to live stream software or game development. Our motivation behind this project is to continue to justify the importance of and promote informal learning opportunities for computer science students and software engineers. The potential benefits of students streaming software and game development reiterate the importance of continued education for developers and accessibility for individuals unable to participate in formal education. Live streams are traditionally free, meaning anyone can access and participate in them given access to stable internet connections; this can help improve STEM education and develop skills from novices to experts. Additionally, we aim to encourage community participation as a means of professional development for software and game developers at all levels to foster a more well-rounded individual student and eventual professional developer.

Background and Related Work

Expanding Computer Science Education As we expand computer science education further, it is important to consider the principle of Culturally Responsible Computing (CRC), which aims to promote and establish computing practices that enable participation from diverse and underrepresented learners [10, 11]. Structural barriers to computing education include access, lack of engaging content, and shortage of role models and peer networking [11, 12, 13, 14, 15].

Outside of structural barriers, social and societal barriers like misconceptions and perceptions of the field of computing, and stereotypes of the practitioners and working environments within computing [11, 16, 17, 18].

Prior work has shown that formal engineering and science curricula alone cannot begin to close the gaps and barriers seen in computing education [19]. In addition, learning outside of the formal class has been shown to benefit those who are underrepresented in STEM [20]. Knowing this, there have been many efforts to expand and commend computer science education outside of a formal curriculum through hackathons, summer camps, service learning, co-ops and internships, and live streaming [6, 9, 7, 8]. These informal learning opportunities enable educators and practitioners to broaden students' skill sets, interests, and experiences [21].

Informal learning opportunities play a crucial part in knowledge sharing and acquisition in ways that are not motivated by the specific curriculum but rather in indeterminate and opportunistic ways not defined by a set amount of time [22, 23]. Research suggests that informal learning can be effective when individuals can contextualize and situate their learning experiences into something meaningful and tailored to their expected experiences [5]. Within the workplace, professionals use informal learning for continuing education, seeking help, gathering information, finding support or feedback, collaborating, and gaining further experience in both their career and private lives [24, 25]. However, despite research showing the benefits of informal learning opportunities, many individuals and organizations push for formal education over informal or mixed educational pathways [26].

Informal Learning Informal education and opportunities in STEM help bridge the gap between formal education and real-world experiences and foster continuing education throughout a career and beyond [27, 28]. Specifically within computer science education (CSEd), active learning techniques like "rubber ducking[1]" problems aloud and group problem solving can be effective methods in CSEd [29, 30].

Informal learning can also prove to be a useful tool in knowledge transfer. Knowledge transfer in software development is paramount for developers to stay ahead and keep their skills valuable and marketable. Knowledge transfer takes many forms, from interpersonal communication, static text documentation like wikis, how-to guides, question and answer forums, prerecorded video lessons and lectures to in-person classes [31, 32, 33]. Prior research shows alternatives to online text documents and documentation like prerecorded software videos are a helpful way to share knowledge and build a reputation in the developer community [34].

Live Streaming Software and Game Development Platforms like Twitch and YouTube are primarily known for gaming and entertainment content, but they also host a growing and diverse array of creative and educational streams. These streams include artists, makers, musicians, and STEM-related broadcasts – including a category specifically for software and game development. Recent research indicates that developers who stream their work find it beneficial for self-education, accountability, and perceived skill enhancement [35]. The community aspects of these streams are also significant, with viewers contributing both socially and technically to content creators they engage with, as well as providing an alternative platform for developer

[1] A method of debugging and problem-solving technique where the developer thinks aloud to an inanimate or abstracted object and explains line by line what they are doing with code; http://lists.ethernal.org/oldarchives/cantlug-0211/msg00174.html

advocacy and STEM careers [35, 36].

Early educational live streaming research indicates that software and game development live streams have the possibility to integrate with more mainstream online learning modalities [37]. Some have taken to these platforms specifically to teach computer science courses and found that this might be a beneficial way for learners to explore new content and engage with education professionals in their courses [38]. Live learning has shown to be beneficial for "over the shoulder" learners and those wanting to personalize their learning goals and interests [9, 39]. Compared to prerecorded video content, live streaming offers a lower barrier for entry and a real-time learning experience that allows viewers to follow along with experts and knowledgeable individuals as they work [9, 40]. Despite the growing popularity and research around educational live streaming, there is a notable gap in research regarding its use as a learning tool within formal education settings. This study starts to address this gap by exploring the potential use of live streaming as an educational tool in an undergraduate course.

Methodology

In this section, we describe our research questions, course structure and design, and methods we use for evaluating self-efficacy and tracking student progress in the course.

Research Questions One goal of this work is to identify usage patterns of students who are given the opportunity to learn at their own pace with a project of their choosing. Additionally, this study aims to demonstrate that informal learning opportunities, such as live streaming, can be integrated into a formal education program. Therefore, we ask the following research questions (RQs):

RQ_1: How does live streaming impact students' perceived self-efficacy in software development?

RQ_2: What usage patterns do students gravitate toward when given the opportunity to live stream software development?

RQ_3: What are the perceived benefits of live streaming as an informal learning opportunity for computer science students?

RQ_4: What are the perceived challenges of live streaming as an informal learning opportunity for computer science students?

Through this work, we aim to understand and evaluate whether or not live streaming impacts an undergraduate student's perceived self-efficacy in software or game development, RQ_1. To quantitatively measure self-efficacy, we have adapted questions from Ramalingam and Wiedenbeck's Computer Programming Self-Efficacy Scale and Hiranrat *et al.*'s survey measurements for software development career [41, 42]. As we allow the students to choose their own projects and set their own goals, we expect there to be some division among the participants on how quickly they believe themselves to be improved based on the gravity of the goals they set for themselves. Given this, we also expect to see if students changed their goals over time and why or why not these goals changed. While we speculate that perceived self-efficacy will increase, we understand that outside factors to the class may impact this, as we have already seen several students drop the course or need modifications to the course to complete their work. Next, we aim to understand better what students will choose to stream, RQ_2. By understanding what

students will use their allocated live streaming time for we can potentially see deficiencies in formal curriculum where the university does not offer a course a student deems necessary to their education. Through the students' usage patterns, we can also understand what students are most interested in, regardless if it is tied to professional or educational development.

Focusing on informal learning, we expect to understand the benefits and challenges that arise from this learning style, RQ_3 and RQ_4. When we understand the benefits and challenges of live streaming as an informal learning platform, we can provide recommendations and expectations for those interested in streaming and how the platform is useful for educational purposes. For students, live streaming benefits could include access to real-time interactions with those who may be experts or knowledgeable in what the student is streaming. These interactions may also lead to community building which could prove useful for students preparing for the workforce. We anticipate some challenges related to course load and time management for students. Understating these better will help inform our recommendations in the course syllabus for overall course planning and optimizing streaming requirements and inform those who may be interested in streaming in general.

Ultimately, understanding students' use of live streaming for informal learning and the associated benefits and challenges allows us to advocate for informal learning in formal software development education. Recognizing this enables us to justify the significance of informal learning as a viable avenue for both novices' and experts' education in the software and technology fields.

Course Design To address our research questions, we conducted an in-situ study with undergraduate computer science and computer engineering students. This course was offered by our University's School of Computing Department in Spring 2024 and Fall 2024 as a 3000-level variable credit hour elective course for the duration of a semester, approximately 15 weeks. These 3000-level elective courses typically are special topics courses for interest areas or undergraduate research opportunities and often a place to pilot a course prior to a full 3-credit hour course offering. In addition, courses in this designation count toward a student's graduation requirements for elective courses. As this course was listed as an upper-undergraduate level course, we expected students to have some experience in coding and have taken an introductory 2000-level software development course, preferably taken the 3000-level software development course to ensure that students have an understanding of basic development practices and have had exposure to larger projects. For either semester, we did not gate-keep enrollment in the course from anyone who has not taken either of these "prerequisite" courses and because this was an instructor preference and not a steadfast requirement, the course was open to students from other computing and engineering disciplines.

Course activities focused on live streaming software or game development. Students were free to pick whatever project(s) they desired in software or game development, in any language, with the only requirement from the course being that projects they worked on could not be from another course offered at the university or part of proprietary work for a company. Students could choose to work in languages they know or pick something completely new to them, and they were not required to stick with the same project week after week but may choose to.

During the Spring 2024 semester and depending on the variable credit hours for the course, the

student would stream three times their course credit hours for this course each week, meaning taking the course for one credit hour meant three hours of streaming, two credits for six hours of streaming, and three credits for 9 hours of streaming per week. Students could break up streaming however they liked as long as their weekly totals matched their credit hour multiplier. After each stream, students would fill out a post-stream diary entry through Google Forms to track what they did and how they felt about it. Students were also required to upload recordings of their streams so that the instructor could verify stream content and duration reported in diary entries.

In addition to streaming and diary entries, students completed several non-streaming assignments throughout the semester about their goals with a pre-, mid-, and end-of-semester goal evaluation, watching another software and game development streamer assignment, and a self-efficacy assessment. Watching a stream assignment consisted of approximately one hour of viewership of another software and game development streamer not in the course and writing a summary of the stream and any notable features of the stream that stood out to them.

The class met once a week for approximately one hour. The instructor used this time for class management and resolving any issues with stream setup or questions about running streams at the beginning of the semester. Once students began streaming, the course was run in a stand-up style meeting where students summarize what they worked on and what they plan on working on to the instructor and other students, with time at the end for comments and questions. Grades for the class were based on meeting objectives and attendance and not the content of the streams or diary entries, i.e. students streamed the minimum number of hours a week, attended weekly meetings, and submitted diary entries on time.

Based on student feedback, detailed in the Discussion, we updated our second iteration of the course to a multiple of two hours of streaming per credit hour - meaning one credit hour would stream for two hours a week, two credits for four, and three credits for six. Additionally, we moved non-streaming assignments around to place the watch a stream assignment as the first assignment and added a pre-survey to the course to complement the end-of-semester post-survey on self-efficacy. All other assignments and course requirements remained the same. The weekly course flow and assignments for the Spring 2024 semester and updated course can be found in Table 1.

Course Methods We elected for a participant-driven diary study due to the duration of the course and students' ability to stream whenever they choose, as the class does not have a prescriptive time slot where students are required to stream [43]. The diary collects participant data on what they did during a stream and asks reflection questions on how they felt the stream went [44]. Students are asked to complete the diary after each streaming session, which is given through a reusable Google Forms link. The diary prompts the students for the date of the stream, what time they streamed, duration, the number of viewers, what they worked on during the stream, what did or did not go well during the stream, interactions with viewers, and then Likert evaluations of how they thought the stream went.

During the course, students completed a pre-, mid-, and post-class evaluation that will serve as a baseline for understanding their goals and objectives for the semester. Additionally, during class, students are expected to summarize and confer with their classmates about how their streams are proceeding and to discuss strategies that are working or not working for them in a stand-up style

Week	Spring 2024 Semester Topic/Assignment	New Semester Topic/Assignment
1	Syllabus and class expectations, research consent, initial goal assignment	Syllabus and class expectations, research consent, *self-efficacy survey*, *introduction to Twitch Content Dashboard* *watch a stream assignment*
2	Stream preparation, hardware and software testing, students preparing work to stream	*Watch a stream discussion, initial goal assignment*, hardware and software testing
3-6	Streaming	Streaming
7	Streaming, mid-semester goal reflection assignment	Streaming
8	Spring Break, no assignments or stream	Fall Break, *mid-semester goal reflection*
9	Streaming	Streaming, *Mid-semester reflection discussion*
10-13	Streaming	Streaming
14	Watching a stream assignment	Streaming
15	Final Reflection on goals, self-efficacy assessment	Final Reflection on goals, self-efficacy assessment *overall course feedback and open discussion*

Table 1: Course schedule and assignments by semester with *new and updated content schedule*.

process during once-a-week class time.

At the end of the Spring 2024 course, we administered a programming and streaming self-efficacy survey. In the subsequent semester, we administered the same survey at the beginning and end of the course. To measure self-efficacy, we have adapted questions from Ramalingam and Wiedenbeck's Computer Programming Self-Efficacy Scale and Hiranrat *et al.*'s survey measurements for software development career [41, 42]. Within our survey, we break down self-efficacy into five sub-scales - independence and persistence, complex programming tasks, self-regulation, communication, and self-views on live streaming software and game development. We administered our survey once as a retrospective of the whole course, asking students to consider the last 4 months of class and the work they completed as a part of this course.

The University's Institutional Review Board (IRB) reviewed and has granted permission to conduct this human subjects research with students. Incentives are not offered for participation, and participation in the research is not required to receive a grade for the course.

Data Analysis This work utilizes both qualitative and quantitative data analysis methods. Qualitative data includes diary entries and stand-up meeting transcripts. Each of these were iterated through to identify main themes related to our RQs – benefits, challenges, usage patterns and perceived self-efficacy. For quantitative data from the pre- and post-surveys for self-efficacy, we will analyze the data using paired sample t-tests and the Benjamini-Hochburg adjustment for p-values to account for any false discovery rates due to our smaller sample size [45].

Participants In Spring 2024, using the unofficial computer science Discord community for our university and word of mouth, we registered and recruited a total of 8 students as part of an undergraduate elective class with the incentive of completing the course granting course credits. Participation in the research component was not necessary to take the course, and no other incentives were offered for students. Through the first few weeks of the Spring 2024 semester, two students dropped from the course completely, leaving us with six total participants. In Fall of 2024, using the same recruitment methods, we registered and recruited nine total students, but

two dropped out, and one did not consent to research, leaving us with six total participants for the Fall. One student returned for the Fall semester and is repeating the course for additional credit hours. In total, we recruited 11 students across two semesters to participate in the research.

Of the students who participated in the research, eleven were males and one non-binary, with five students identifying as being in their third or junior year of undergraduate work and seven in their final year of coursework as a senior. All but two students were computer science majors, with one from computer engineering and one from computer information systems major taking the course. Ten students streamed on Twitch, and two on YouTube Live. All but one student did not indicate any prior experience with live streaming through gaming or other means, but all were aware of streaming platforms and the concept of live streaming.

Positionality and Limitations Several limitations and threats to validity may exist in this study. First, due to the nature of diary studies, feedback burden may be an issue [43]. However, we only expect students to stream once or twice a week and that completing one or two diary entries a week will not impact their overall load in the course or concurrent coursework load. Next, an internal validity threat might be a history threat, where the students might be doing something additional outside of this course that is helping them to perceive improved self-efficacy in their programming skills. To mitigate this, we have asked students to consider only the work conducted in this specific course throughout the semester. An additional history threat might be last-minute completion of assignments and procrastination. However, all participating students were junior and senior level, and thus have knowledge of and some expectation for time management related to streaming. Another threat to validity is attrition, and while every teacher hopes that students stay in the course, there is always the chance that a student may drop the course or cease work related to the course. Another limitation is that students may have a reaction due to their awareness of participating in research and may alter behaviors to better participate in the research or to achieve a grade within the course. By conducting the study over the course of the semester, we expect this effect will fade over time, in addition to ensuring that we are not overloading the students with surveys, as they will be completed on an as-needed basis after streaming. To mitigate this better, students not taking this course could be administered the self-efficacy pre- and post-surveys to establish external validity of our potential findings. Finally, another limitation that could arise from this research is that our sample size could be homogeneous in terms of gender and ethnicity, with our first iteration of the course being predominately male.

With respect to self-reflexivity, the authors would like to acknowledge that they bring backgrounds and experiences that shape this work. The first author is a millennial who views live streaming as an educational tool with significant potential for computer science education and has experience live streaming gaming and educational creative content. However, we acknowledge that a positive view of live streaming technology may bias us toward seeing benefits over potential drawbacks. Additionally, the first author's dual role as course instructor and researcher might create a power dynamic with students. To help mitigate the effects of this power dynamic, we strived to create an open environment where students could share openly and honestly without pressure to report only positive experiences and encouraged students to reach out if they faced issues with streaming.

Self-efficacy Subscale	Items	*n*	Min	Max	Mean	SD
Independence and Persistence	8	6	1	5	26.17	0.98
Complex Programming Tasks	5	6	2	5	16.50	3.94
Self-regulation	4	6	1	5	13.33	4.37
Communication	5	6	1	5	17.33	5.82
Self-views on Live Streaming Software and Game Development	8	6	1	5	27.33	7.00

Table 2: Students' scores on self-efficacy sub-scales in Spring 2024

Results

First Course Iteration - Spring 2024

Perceived Self-Efficacy An important aspect of this course is understanding student self-efficacy when they are given the autonomy to work on projects outside of another course's curriculum. We asked students to consider these questions with how they feel and relate to our course, particularly over the entire semester. Table 2 summarizes the overall descriptive data measured by our five sub-scales of self-efficacy. The students' highest mean sub-scale was Communication (3.40), followed by Self-views on Live Streaming Software and Game Development (3.36), Independence and Persistence (3.32), Self-regulation (3.29), and finally, Complex Programming Tasks (3.26). Based on a 5-point Likert scale, where a 3 represented a neutral response *neither agree nor disagree*, all sub-scales scored above the neutral, indicating that the students had middle to high self-efficacy regarding the content streamed and related programming skills they used during this course.

Students reported the highest self-efficacy in the Independence and Persistence sub-scale related to debugging. Within complex programming tasks, students felt the most efficacy in organizing and modularizing a program, writing a program someone else could understand and use, and rewriting confusing code. Then in self-regulation, students found the most efficacy in their ability to plan for a project in a short amount of time. Next, in communication, students felt they could best be able to listen to others and consider their thoughts about a programming concept or idea. They found the most inefficacy in being able to present technical information for feedback. Finally, within self-views, students reported that, overall, live streaming software and game development were an important part of their learning experience and that through live streaming, they were able to learn and find ways to engage in continuous learning.

Something interesting about the self-efficacy assessment was that the students felt the most inefficacy was presenting technical information for ideas and feedback. Still, they felt they could explain what they were working on well. There could be several reasons for this, but it may result from a lack of exposure to peer feedback opportunities or exposure to code review activities seen in a professional environment. Another discrepancy we observed was student reporting of positive experiences with live streaming and the benefits they see for this modality to benefit them for continuous learning, but many would choose not to continue to live stream after this course.

Student Goals Three times throughout the course, at the beginning, middle, and end of the semester, we asked students their goals for taking the course and stream content. We recorded

initial goals at the beginning of the semester before students started to stream. Students' initial goals centered around skill practice, diversifying programming abilities, setting aside time to work on personal projects, and using this course as motivation and practice learning to stream.

💬 *"I wanted an opportunity to work on the personal projects that I've been meaning to get around to, but haven't had the time or motivation to get started on. I saw this course as a way to iron out time in my schedule to work on the projects I've been wanting to make for years."* - S6

💬 *"I expect that working on some of these projects while streaming (which is something I enjoy, and have experience doing) will be both fun and helpful for my future career."* - S2

Students had a variety of projects and ideas they wished to work on from APIs, Python, game development, web development, and mobile application projects.

In the middle of the semester, after about 5 weeks of streaming, we checked in with students again and asked them to reflect on their initial goals and content objectives from the beginning of the semester. About four of our six students had a change in their personal goals and outcomes from streaming and the course. These changes include narrowing the scope down from many small projects to one project, choosing to work in only one language due to projects in other courses, and stepping back from a full-scale project to work on general skill improvement and practicing LeetCode[2] problems. We also saw one student, S2, who needed additional accommodations to complete the course, citing that they were not motivated to complete the necessary streaming per week (˜3 hours a week for this student). At this point in the semester, the student was unable to drop, so the instructor and student met to discuss fulfilling course requirements. We agreed on the student streaming when they can, but any week where they are unable to stream they must submit a watching a stream assignment where they provide a summary of watching another software and game development streamer and anything notable they took away from the stream. At this time, we also offered other students this same opportunity if they needed to substitute one week of streaming. No other students chose the stream summary assignment over a weekly stream.

💬 *"I originally wanted to do a full-scale project, but I realized I just did not have enough time as I have 2 project-based classes."* - S3

💬 *"I changed this goal from many projects that were simple to larger more complex ones to help myself digest how all of the parts interact better."* - S1

Finally, student reflections were mixed. Many students stuck with their mid-semester goal pivot, while others changed directions still, like S3, who also chose to practice LeetCode problems for skill and professional development, and S2, who did not submit a final reflection. Overall, most students appeared content with their progress over the semester, but through written assignments wanted to accomplish more.

💬 *"Python was easy but I really wanted to progress C as it is more relevant to my field of Computer Engineering and aids my current and future classes."* - S1

Additionally, we asked students about their perceptions of live streaming as a whole, what impacts they think it had on them, and if they would continue to stream after this course. Several

[2] LeetCode is an online platform for programming and coding problems that are often used in technical interviews. https://leetcode.com/

of our students had prior streaming experience or wanted to use this course as an introduction to streaming. Two students reported that they would not like to continue streaming, while another would like to but believes their schedule would not permit them to commit the time to it. One student, S10, felt that streaming was "invasive" to their development processes and that prevented them from doing work quickly. Three students reported that they would consider or will live stream after this course and noted it was a helpful way to set aside time to work, and they enjoyed the progress they made while live.

💬 *"It was too stressful the feeling that someone was watching and judging my creative process and it is hard to entirely focus."* - S4

Course Observations and Lessons Learned Throughout the course, students generally completed tasks on time without much intervention and were positive and receptive to the course. However, we struggled with attrition rates. Two students in the research study, not included in this report, dropped the course without contacting the instructor, with both taking a W in the course, meaning they dropped after the semester drop date, and it would show on their transcript. The instructors reached out to these students as they were not attending the weekly meeting to understand if anything in the course was unmanageable or unreasonable, but did not receive a response before or after the decision to drop the course entirely.

We observed additional challenges with student retention and gathered midterm feedback on desired course improvements. Our first observed challenge was overall motivation. Students taking more than one credit hour tended to complete all their required streaming hours in a single session, despite being told they could divide their time, and resulted in lack of motivation to sit and work for several hours at a time. Students also struggled with the technical aspects of setting up streams, while others faced connectivity issues due to inadequate internet access. When this came up, the instructors offered personal research lab space on campus, with hardwired internet and connections to students who needed it, and directed them to on-campus rooms that could be booked in advance with the same access if they did not want to use the lab space. Next, unfamiliarly with streaming platforms did not afford a good first-time experience when coupled with fulfilling course assignments on time if left to the last moment.

We received mid-term feedback from students that an introductory overview to a streaming platform would be beneficial, along with walking them through an example diary entry. Finally, students also recommended changing the due date of assignments to be more streamlined by the end of the week.

Second Iteration - Fall 2024

Perceived Self-efficacy

Independence and persistence questions focused on perceived ability in completing programming tasks with or without aid from other sources and averaged pre-semester to 33.50 out of 40 points and decreased post-semester to 30.0 out of 40 points. Paired sample t-test group results indicate a significant mean decrease in students perceived independence and persistence, $t(5) = 4.13$, $p < 0.01$, BH corrected $p < 0.04$. The largest change in perception by specific question surrounded a decrease in belief that students could complete a project if they could ask someone for help if they

	Pre-Class			Post-Class						
Section	**Mean**	**SD**	**SE**	**Mean**	**SD**	**SE**	**Cohen's *d***	**Hedges' *g***	**p-value**	**BH p**
Independence and Persistance	33.50	4.76	1.95	30.0	5.29	2.16	1.69	1.42	0.01	0.03
Complex Programming Tasks	19.17	4.46	1.82	20.83	3.06	1.25	-0.85	-0.72	0.05	0.12
Self-Regulation	14.00	3.16	1.29	14.83	2.79	1.14	-0.24	-0.20	0.29	0.37
Communication	19.67	3.39	1.38	20.00	3.46	1.41	-0.11	-0.09	0.40	0.40
Self-Views on Live Streaming	31.17	6.18	2.52	33.00	4.34	1.77	-0.40	-0.33	0.19	0.31

Table 3: Descriptive statistics for six (6) students in the second iteration of the course with Standard Deviation (SD), Standard Error (SE), Hedges' correction to Cohen's *d*, Benjamini-Hochberg (BH) adjusted p-values for Fall 2024

are stuck and completing a project if someone showed them how to solve the problem first.

Complex programming task questions focused on students' ability to organize large programs, write code that others could comprehend, and general comprehension of large programming files or projects. Perceived ability in complex programming tasks pre-semester averaged 19.17 out of 25 total points, and post-semester averaged 20.83 out of 25. Paired sample t-test group results indicate a potential significant mean increase in students perceived ability to navigate complex programming tasks, $t(5) = -2.08$, $p < 0.05$, BH corrected $p = 0.12$. The largest change in perceived abilities around complex programming tasks was an increase in scores surrounding organizing and designing programs modularly and the ability to rewrite confusing programs to be clearer.

Self-regulation questions focused on motivation, time management, and concentration on tasks with scores pre-semester averaging 14 out of 20 points and post-semester to 14.83 out of 20 total points. Paired sample t-test group results indicate no significant mean increase, $t(5) = -0.56$, $p = 0.29$, BH corrected $p = 0.37$. The largest change in specific questions in the perceived ability to self-regulate while programming was an increase in finding motivation to program.

Communication questions focused on the perceived ability to explain what a student is currently working on, listening to others, and asking programming questions. Perceived communication abilities pre-semester averaged 19.67 out of 25 points, and post-semester averaged 20.00 points out of 25 total. Paired sample t-test group results indicate no significant mean increase, $t(5) = -0.27$, $p = 0.40$, BH corrected $p = 0.40$. The largest change in perceived communication abilities was an increase in ability to explain what a student was currently working on.

Finally, self-views on live streaming questions focused on students' perceived enjoyment of live streaming and the impacts that live streaming could have on their abilities and potential career goals. Perceived views of live streaming pre-semester averaged 31.17 points out of 40 and 33.0 out of 40 total points post-semester. Paired sample t-test group results indicate no significant mean increase, $t(5) = -0.97$, $p = 0.19$, BH corrected $p = 0.31$. The largest change in score by question surrounded an increase in overall enjoyment of live streaming and live streaming being a positive experience.

Student Goals and Usage Patterns Like the previous semester, we collected student goals and reflections three times throughout the semester - once before they started streaming, again after 5 weeks of streaming, and a goals reflection at the end of the semester after all streams were

completed. Students' initial goals overwhelmingly focus on personal and portfolio work and motivations to complete this work, next one student wanted to work on Leetcode problems for interviews, and one student wanted to use the class as a way to resolve doubt about software development as a career choice. By the middle of the semester, most student goals have not changed, with one student, S7, notability shifting focus to Leetcode for a few weeks while they prepare for coding interviews. A few weeks after mid-semester evaluations, S7 reported in class that he received a job offer and credited "[the] coding challenges helped him with his interviews and converting his internship to a job." Most students reported they are still working on personal projects from their backlog, but changing which project they are working on – Discord bots, learning a APIs, and game mods. Finally, student reflections were mostly positive, with most students believing they achieved their goals or enough of their original or updated goals to be satisfied with their progress throughout the course. Two students were not satisfied with their progress toward meeting the goals they set for themselves.

This semester, students' challenges related mostly to difficulties and roadblocks in the work they streamed and not motivations to stream.

💬 *"During several of my streams I struggled with choosing between some options for the foundation of my projects."* - S6

Some students expressed some hesitation and challenges about how they perceived themselves as streamers and felt some pressure to perform while they were live. These sentiments were expressed especially in class stand-ups if a student said someone visited their stream that they did not know.

💬 *"I think my most challenging thing was overcoming the idea of many people watching me and expecting a lot from me."* - S8

Student benefits of taking the class include the ability to obtain skills and projects they could put on a resume and overall benefit their professional development.

Course Observations Overall, students completed tasks on time and without intervention. One student believed that they could use the "replace a stream" assignment every week, and after submitting this assignment two weeks in a row, the instructor emphasized that that replacement was one used for when students may have had a midterm or needed a break from streaming. No other challenges were observed relating to keeping up with assignments or motivation to complete the necessary hours of streaming based on their credit hours.

Discussion

In this section, we discuss our lessons learned and proposed interventions based on the experiences and observations from the Spring 2024 semester and then discuss the implemented interventions and overall experiences based on our subsequent semester.

Lessons Learned From First Course Iteration and Procedural Course Changes Several adjustments based on our observations and student feedback will be necessary to rerun the course. First, we believe we will need to change the formula for how many hours streamed for credit hours taken. We followed the general rule of 3 hours of work per credit hour taken, but both

students and instructor felt that this was too much work relative to other courses for an elective course. In subsequent semesters, we plan to adjust to two hours of streaming per credit taken for two, four, and six hours of streaming for one, two, and three credits, respectively. We believe this change will alleviate stress for students, and help improve motivation to complete streams. Additionally, we will continue to encourage students who take the course for more than one credit hour that they can split up their stream times. We had emphasized distributing streaming hours throughout the spring semester throughout the week; however, most students chose to complete their work in one sitting, no matter how many credits they took. We will also adjust what defines a "week" for streaming assignments. Previously, we followed a Sunday to Saturday approach for a "week"; however, student feedback suggested making a "week" Monday to Sunday to give students the entire weekend to complete stream assignments for that week.

Next, we will adjust assignments and the order of the assignments for the course. Initially, we wanted students to take their own approach to streaming and how they chose to set up and conduct their streams. While one student was familiar with streaming, and others were familiar with the structure and expectations of streaming as viewers, we felt that many still had hesitations and apprehension about jumping into live streaming themselves. We propose adding two more watching a stream assignments to the beginning of the course in Weeks 1 and 2 to help students familiarize themselves with software and game development streaming. Watching other streamers might help students pick up on good streaming habits or other ideas they can implement in their own streams throughout the semester and alleviate hesitation in starting to stream. In addition, we would remove the final watch-a-stream assignment from the end of the semester in Week 14 and make it a floating assignment for students to use instead of a week when they might not have the time or mental energy to stream.

Next, we will add to the start of the course an introduction to a basic streaming setup on Twitch and YouTube using the free and open source video recording software Open Broadcaster Software[3] (OBS). In this introduction, we will cover the basic features of OBS and content creator options on the video platforms. The variety and options of new software may overload the students, and providing them with what they need to get started, we believe, will help with some of the issues we faced at the beginning of the semester.

Finally, we need to administer the self-efficacy assessment pre- and post-course for better insights into programming self-efficacy changes. Administering both a pre- and post-test will enable us to see statistically measurable quantitative data that coincides with our qualitative data from student reflections and teacher observations.

Lessons Learned From Second Course Iteration Compared to the Spring 2024 semester, Fall 2024 students did not apparently have as many struggles getting started with streaming. Several provided feedback that introducing one of the platforms and going through the user interface was helpful prior to starting to stream, as well as the assignment to observe a streamer before their own streaming started. Within the mid-term course feedback, one student mentioned they would like to see additional information about streaming in a "Tips and Tricks" format to reference for students throughout the semester. Providing this type of reference document could be useful in several ways, especially with motivation, time management, and potential engagement strategies that have worked based on the author's own streaming and observed strategies from others in the

[3] https://obsproject.com/

software and game development category.

Additionally, two students gave their feedback about wanting to incorporate more community into the class, as seen with other streamers on the Twitch and YouTube platforms. As the course is right now, we do not require students to participate in other streams, but facilitating sharing when live could lead to better overall satisfaction with streaming and having a viewer presence in chat for students and create a sense of community within the class at a minimum. To do this, we propose using a tool such as Discord with a bot that would notify when classmates are live and provide a space to ask questions about stream setups and have a space to communicate and collaborate more synchronously outside of the classroom.

Decreasing the credit hour to streaming multiplier appears to have aided students' ability to keep up with the coursework. Additionally, updating the definition of a streaming week reduced confusion about due dates for both the instructor and the students. The due dates were more apparent. Additionally, we modified and updated the schedule due to unforeseen circumstances in the semester, such as university closure due to weather and lack of access to electricity and the Internet.

While students may not indicate that they will continue to stream after the course, general satisfaction with what they achieved during streams and how streaming went, we believe that streaming time positively impacts students' attitudes to learning. Practicing and participating in informal learning with projects that were self-selected may contribute to the decrease in mean scores for independence and persistence. This exposure to more complex problem-solving and open-ended development work may have given students a more realistic understanding of where their skills and abilities reside, and this awareness may have led them to assess their abilities more critically compared to the beginning of the semester.

Over the two iterations of this course, we have seen students begin to bridge their skills between academics and their future professional lives. Giving students autonomy to work on self-directed projects while using streaming as an accountability tool helps foster professional skills like time management, technical communication, and project planning. Students have used this course to build their portfolios, practice interview preparation, and gain programming experience while working on personally meaningful projects that may not be covered in a traditional curriculum. We believe that the informal learning environment, combined with the structure of the course, provides an effective scaffold for independent student work. This format allows students to explore potential career directions, build confidence in their abilities, and develop self-directed learning skills crucial to professional success, making it an especially beneficial addition to final-year computer science education.

Future Work

Future work surrounding live streaming software development as part of formal education should address the limitations of the present study by expanding the sample size and implementing a comparative approach for self-efficacy. A larger sample size would enhance statistical power and potentially reveal stronger patterns in self-efficacy changes and learning outcomes. Incorporating a control group of students engaged in a traditional capstone course could also provide comparative data to evaluate the efficacy of live streaming as a pedagogical tool. A controlled

comparison with a capstone course would allow us to determine whether the observed improvements in complex programming tasks and self-regulation skills are directly attributable to the live streaming format or are common to other project-based learning environments. Finally, a more diverse student population would also strengthen the generalizability of our findings across different demographic groups in computer science education.

Conclusion

Through two iterations of an undergraduate live streaming course, we have demonstrated the potential for informal learning opportunities that would compliment traditional computer science education. Our findings suggest that while students may not continue to stream after the course, the experience provides a valuable professional development opportunity and helps students to bridge theoretical and practical applications of software development. Future work should explore ways to better facilitate a community aspect of the course that would serve as a peer support and collaboration tool. Finally, investigating the long-term impacts on students' professional development and collecting data from a more diverse student population would provide deeper insights into the effectiveness of live streaming as an educational tool.

References

[1] S. Haji Amin Shirazi, M. Salloum, A. Speer, and N. Watkinson, "An experience report: Integrating oral communication and public speaking training in a cs capstone course," in *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, pp. 450–455, 2024.

[2] K. Anewalt and J. Polack, "A curriculum model featuring oral communication instruction and practice," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pp. 33–37, 2017.

[3] E. Tuzun, H. Erdogmus, and I. G. Ozbilgin, "Are computer science and engineering graduates ready for the software industry? experiences from an industrial student training program," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training*, pp. 68–77, 2018.

[4] A. Salguero, W. G. Griswold, C. Alvarado, and L. Porter, "Understanding sources of student struggle in early computer science courses," in *Proceedings of the 17th ACM Conference on International Computing Education Research*, pp. 319–333, 2021.

[5] A. Coelho and L. M. Costa, "The integration of augmented reality and the concept of sticker album collection for informal learning in museums," in *Immersive Learning Research Network: Third International Conference, iLRN 2017, Coimbra, Portugal, June 26–29, 2017. Proceedings 3*, pp. 107–115, Springer, 2017.

[6] J. Dean, L. Barker, and A. Voida, "Bite-sized experiential education for computer and information science," 2024.

[7] A. Nandi and M. Mandernach, "Hackathons as an informal learning platform," in

Proceedings of the 47th ACM Technical Symposium on Computing Science Education, pp. 346–351, 2016.

[8] J. Miller, S. Raghavachary, and A. Goodney, "Benefits of exposing k-12 students to computer science through summer camp programs," in *2018 IEEE Frontiers in Education Conference (FIE)*, pp. 1–5, IEEE, 2018.

[9] Y. Chen, W. S. Lasecki, and T. Dong, "Towards supporting programming education at scale via live streaming," *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. CSCW3, pp. 1–19, 2021.

[10] R. Eglash, J. E. Gilbert, and E. Foster, "Toward culturally responsive computing education," *Communications of the ACM*, vol. 56, no. 7, pp. 33–36, 2013.

[11] A. Scott, A. Martin, F. McAlear, and T. C. Madkins, "Broadening participation in computer science: Existing out-of-school initiatives and a case study," *ACM Inroads*, vol. 7, no. 4, pp. 84–90, 2016.

[12] G. Ladson-Billings and W. F. Tate, "Toward a critical race theory of education," *Teachers college record*, vol. 97, no. 1, pp. 47–68, 1995.

[13] J. J. Ryoo, J. Margolis, C. H. Lee, C. D. Sandoval, and J. Goode, "Democratizing computer science knowledge: Transforming the face of computer science through public high school education," *Learning, Media and Technology*, vol. 38, no. 2, pp. 161–181, 2013.

[14] S. González-Pérez, R. Mateos de Cabo, and M. Sáinz, "Girls in stem: Is it a female role-model thing?," *Frontiers in psychology*, vol. 11, p. 564148, 2020.

[15] C. Hill, C. Corbett, and A. St Rose, *Why so few? Women in science, technology, engineering, and mathematics.* ERIC, 2010.

[16] J. Kim, *Buffers and Barriers to Female First-Generation Students' Career Development*. PhD thesis, The University of Iowa, 2021.

[17] R. E. Wilson and J. Kittleson, "Science as a classed and gendered endeavor: Persistence of two white female first-generation college students within an undergraduate science context," *Journal of Research in Science Teaching*, vol. 50, no. 7, pp. 802–825, 2013.

[18] A. L. Wright, V. J. Roscigno, and N. Quadlin, "First-generation students, college majors, and gendered pathways," *The Sociological Quarterly*, vol. 64, no. 1, pp. 67–90, 2023.

[19] N. A. of Sciences, D. of Behavioral, C. on National Statistics, P. to Evaluate the National Center for Science, E. S. A. to Measuring the Science, and E. Workforce, *Measuring the 21st century science and engineering workforce population: Evolving needs*. National Academies Press, 2018.

[20] L. Espinosa, "Pipelines and pathways: Women of color in undergraduate stem majors and the college experiences that contribute to persistence," *Harvard Educational Review*, vol. 81, no. 2, pp. 209–241, 2011.

[21] M. Polmear, S. Chance, R. G. Hadgraft, and C. Shaw, "Informal learning as opportunity for

competency development and broadened engagement in engineering," in *International Handbook of Engineering Education Research*, pp. 312–335, Routledge, 2023.

[22] P. Hager and J. Halliday, *Recovering informal learning: Wisdom, judgement and community*, vol. 7. Springer Science & Business Media, 2007.

[23] M. A. Kelly and P. Hager, "Informal learning: relevance and application to health care simulation," *Clinical Simulation in Nursing*, vol. 11, no. 8, pp. 376–382, 2015.

[24] T. J. Conlon, "A review of informal learning literature, theory and implications for practice in developing global professional competence," *Journal of European industrial training*, vol. 28, no. 2/3/4, pp. 283–295, 2004.

[25] E. Schürmann and S. Beausaert, "What are drivers for informal learning?," *European Journal of Training and Development*, vol. 40, no. 3, pp. 130–154, 2016.

[26] A. Manuti, S. Pastore, A. F. Scardigno, M. L. Giancaspro, and D. Morciano, "Formal and informal learning in the workplace: A research review," *International journal of training and development*, vol. 19, no. 1, pp. 1–17, 2015.

[27] K. Sacco, J. H. Falk, and J. Bell, "Informal science education: Lifelong, life-wide, life-deep," *PLoS Biology*, vol. 12, no. 11, p. e1001986, 2014.

[28] K. A. Benjamin and S. McLean, "Change the medium, change the message: creativity is key to battle misinformation," 2022.

[29] W. Zuill and K. Meadows, "Mob programming: A whole team approach," in *Agile 2014 Conference, Orlando, Florida*, vol. 3, 2016.

[30] D. Schweitzer and W. Brown, "Interactive visualization for the active learning classroom," in *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pp. 208–212, 2007.

[31] C. Treude and M.-A. Storey, "Effective communication of software development knowledge through community portals," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pp. 91–101, 2011.

[32] M. Meng, S. Steinhardt, and A. Schubert, "Application programming interface documentation: What do software developers want?," *Journal of Technical Writing and Communication*, vol. 48, no. 3, pp. 295–330, 2018.

[33] E. Aghajani, C. Nagy, M. Linares-Vásquez, L. Moreno, G. Bavota, M. Lanza, and D. C. Shepherd, "Software documentation: the practitioners' perspective," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pp. 590–601, 2020.

[34] L. MacLeod, M.-A. D. Storey, and A. Bergen, "Code, camera, action: How software developers document and share program knowledge using youtube," *2015 IEEE 23rd International Conference on Program Comprehension*, pp. 104–114, 2015.

[35] E. Kokinda and P. Rodeghero, "Streaming software development: Accountability, community, and learning," *Journal of Systems and Software*, vol. 199, p. 111630, 2023.

[36] S. Chattopadhyay, D. Ford, and T. Zimmermann, "Developers who vlog: Dismantling stereotypes through community and identity," *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CSCW2, pp. 1–33, 2021.

[37] T. Faas, L. Dombrowski, A. Young, and A. D. Miller, "Watch me code: Programming mentorship communities on twitch. tv," *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, pp. 1–18, 2018.

[38] J. Pirker, A. Steinmaurer, and A. Karakas, "Beyond gaming: The potential of twitch for online learning and teaching," in *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, pp. 74–80, 2021.

[39] N. Selwyn, "Web 2.0 applications as alternative environments for informal learning - a critical review," 01 2007.

[40] A. Gandsas, T. Dorey, and A. Park, "Immersive live streaming of surgery using 360-degree video to head-mounted virtual reality devices: A new paradigm in surgical education," *Surgical Innovation*, p. 15533506231165828, 2023.

[41] V. Ramalingam and S. Wiedenbeck, "Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy," *Journal of Educational Computing Research*, vol. 19, no. 4, pp. 367–381, 1998.

[42] C. Hiranrat, A. Harncharnchai, and C. Duangjan, "Theory of planned behavior and the influence of communication self-efficacy on intention to pursue a software development career," *Journal of Information Systems Education*, vol. 32, no. 1, p. 40, 2021.

[43] S. Carter and J. Mankoff, "When participants do the capturing: the role of media in diary studies," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 899–908, 2005.

[44] K. Baxter, C. Courage, and K. Caine, *Understanding your users: a practical guide to user research methods*. Morgan Kaufmann, 2015.

[45] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *Journal of the Royal statistical society: series B (Methodological)*, vol. 57, no. 1, pp. 289–300, 1995.