Towards Supporting Programming Education at Scale via Live Streaming

ANONYMOUS AUTHORS

Live streaming, which allows streamers to broadcast their work to live viewers, is an emerging practice for teaching and learning computer programming. Participation in live streaming is growing rapidly, despite several apparent challenges, such as a general lack of training in pedagogy among streamers and scarce signals about a stream's characteristics (e.g., difficulty, style, and usefulness) to help viewers decide what to watch. To understand why people choose to participate in live streaming for teaching or learning programming, and how they cope with both apparent and non-obvious challenges, we interviewed 14 streamers and 6 viewers about their experience with live streaming programming. Among other results, we found that the casual and impromptu nature of live streaming makes it easier to prepare than pre-recorded videos, and viewers have the opportunity to shape the content and learning experience via real-time communication with both the streamer and each other. Nonetheless, we identified several challenges that limit the potential of live streaming as a learning medium. For example, streamers voiced privacy and harassment concerns, and existing streaming platforms do not adequately support viewer-streamer interactions, adaptive learning, and discovery and selection of streaming content. Based on these findings, we suggest specialized tools to facilitate knowledge sharing among people teaching and learning computer programming online, and we offer design modifications that promote a healthy, safe, and engaging learning environment.

CCS Concepts: • **Information systems → Crowdsourcing**; • **Human-centered computing → Human computer interaction (HCI)**; • **Computing methodologies** → *Computer vision.*

Additional Key Words and Phrases: Crowdsourcing; Human Computation; Aggregation; 3D Reconstruction

ACM Reference Format:
Anonymous Authors. 2020. Towards Supporting Programming Education at Scale via Live Streaming. In *Proceedings of CSCW '20: ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW '20).* ACM, New York, NY, USA, 20 pages. https://doi.org/TBD

1 INTRODUCTION

In the past decade, video-centric learning materials from major platforms such as YouTube, Coursera, and Khan Academy have enabled people of all ages and countries to share and gain knowledge [1–3]. These videos are often well-structured, pre-recorded, and carefully edited by experienced instructors or even professional teams, and the content often covers specific topics, concepts, or techniques. However, the effort required to start sharing knowledge using these approaches is often high. Moreover, the asynchronous interaction and non-immediate feedback can impede learning and reduce student engagement [8, 54].

In the last few years, live streaming has quickly become a complementary mode of sharing and obtaining technical knowledge via video. To understand this emerging practice, we studied live streaming of computer programming where the stream primarily focuses on demonstrating programming knowledge and practical skills for educational purposes. We chose to investigate

CSCW '20, 2020,

ACM ISBN TBD…$15.00
https://doi.org/TBD

this phenomenon for two reasons: 1) there is now widespread global interest in learning to program [17, 53], and 2) prior work has found that live coding, in which instructors write code from scratch on a computer connected to a projector during the class [38], is effective for programming education [44]. Live streaming platforms scale and democratize this pedagogical technique for anyone who has a decent Internet connection and a passion for sharing their knowledge with the world.

Although live streaming is a promising format in the online learning ecosystem [14], there are many apparent downsides. Compared to massive open online courses (MOOCs) or other professional educational settings, streamers often lack pedagogical training, leading to large variations in content quality, presentation skills, and the ability to engage students. On the viewer's side, there are few signals to determine whether an upcoming stream will be a good fit for the viewer's learning needs and level of expertise, because unlike pre-recorded videos, live streams have no ratings, view counts, or previews. Together, these challenges could negatively affect the viewer's learning experience, making live streaming a less effective and less viable learning medium compared to other video-based alternatives.

To sustain and improve the developing practice of live streaming programming, this work aims to understand the reasons for the disconnect between the growing interest in stream participation and the aforementioned challenges. In particular, we explore the benefits that the major stream platforms (e.g., YouTube Live, Twitch) provide for learning-oriented live streams, and the challenges people overcome, work around, or fail to tackle on those platforms. We use pre-recorded video, a more established medium for learning programming, as a reference point to gain a better idea of the position of live streaming in the online programming education ecosystem. Specifically, we wanted to answer the following research questions. Compared to pre-recorded videos,

RQ1: why do streamers choose to share programming knowledge via live streaming?
RQ2: why do viewers choose to learn programming via live streaming?
RQ3: what are the barriers that streamers face when sharing programming knowledge via live streaming?
RQ4: what are the barriers that viewers face when learning programming via live streaming?

Through in-depth interviews with 20 participants (14 streamers, 6 viewers), we found that 1) the nature of live (e.g., more casual) content makes streaming sessions easier to prepare than pre-recorded video; 2) streaming gives viewers the opportunity to shape the content according to the influence of real-time interactions; 3) live interactions make these streams infeasible to script, intensifying streamers' privacy and harassment concerns; and 4) inadequate tool support makes learning hard to personalize for individual viewers (e.g., pacing). Based on these findings, we discuss four implications for platform design, including a) evolving generic video streaming platforms to support personalized learning, b) reducing the effort of documenting and retrieving points of interest in live streams, c) improving support for content moderation, and d) mitigating streamers' privacy and security risks. Also, we critique and elaborate on prior design prototypes and concepts applicable to this domain. These ideas work toward a form of improvisational, real-time, and personalized knowledge sharing to fill gaps in formal programming education. To sum up, we make the following contributions:

- We position live streaming in the online, and especially video-based, programming education ecosystem by highlighting unique advantages we should leverage and limitations we must mitigate and manage to make this learning medium more effective and viable.

- We propose specialized tools to facilitate programming knowledge sharing via live streaming, as well as design modifications to promote a healthier and more secure learning environment.

2 RELATED WORK

Our work mainly relates to two fields: common practices of programming learning and live streaming. In this section, we will review the key work in these two areas.

2.1 Informal Learning

Consuming knowledge on streaming platforms like Twitch.tv is a form of informal learning, which is a type of learning undertaken on the learner's own, without externally imposed criteria [47]. Unlike formal learning, which is often institutionally sponsored and structured, informal learning provides learners better control and personalization, representing a promising opportunity for learners to develop different kinds of skills for lifelong learning [9]. In recent years, a growing number of people have begun to engage in a wide range of technology-based informal learning opportunities. However, the Educating the Engineers of 2020 report highlights the lack of research on informal learning within engineering education [41].

In the context of live streaming, prior work has reported on the unique learning relationship that this medium creates for programming communities, a relationship that is not found in other online platforms. More specifically, the act of learning via live streaming often takes place in a collaborative format rather than an individual one, which enables a sense of social presence. However, researchers have raised the concern that the place of live streaming within the online learning ecosystem is still unclear, making it difficult for people to choose when to use this medium and what benefits they might receive from it [14]. Our work aims to more clearly position live streaming in the pool of different learning mediums, adding new understanding to the body of literature on informal learning within programming education [41].

2.1.1 Video Lecturing. The use of video lecture capture in higher education is becoming increasingly commonplace in universities worldwide and massive open online courses (MOOCs) [23, 53]. Video lectures provide extra resources that may complement students' studies and add flexibility, temporally and spatially, for students to catch up and revisit the learning materials [27]. Prior work has explored the cognitive value of educational videos on YouTube and found a set of cognitive features that correlates with students' learning gains [49]. However, video lectures have a negative impact on students' performance compared to regular classroom lectures. Prior work has shown that video lectures can be perceived as uninteresting and may lack beneficial student-teacher and student-student interactions, leading to low class attendance and less engagement [8, 26, 54].

Video delivery of programming solutions may be useful in enabling a lecturer to illustrate the complex decision-making processes and the incremental nature of the actual code development process. McGowan et al. analyzed students' activities during programming lecture videos and found that code-based, practical demonstrations are more valued by students than solo explanations [36]. In this work, we explore the potential pedagogical value of coding demonstration via live streaming to inform and improve educational benefits. We focus specifically on the informal learning context rather than institutionally structured alternatives such as the types of video lectures found in MOOCs.

2.1.2 Live Programming. The practice that our work focuses on closely relates to live programming, an existing common pedagogical practice for teaching computer programming where instructors write code with little preparation work and project the process to learners [38]. Learners benefit from watching instructors' iterative process of thinking, designing, coding, and testing,

which is rarely covered in the details of a slide-deck style of teaching [6, 16]. Prior work has explored the effectiveness of live programming for teaching introductory programming and found it to be on par with using static code examples [44]. Raj et al. [43] found that students prefer live-coding, as they benefit from seeing the process of programming and debugging. Our work will explore the benefit of the live programming practice in an informal and online learning settings.

2.2 Live Streaming

Recently, an increasing number of work have focused on studying live streaming in the HCI community, as this medium enables anyone to share their life experiences, opinions on different topics, and more. Prior work has explored the use of streaming platforms and found that streams involve a diverse range of activities, with branding as the most common motivation [52]. Dougherty explored the use of Qik, showing that 11% of videos had civic importance, such as journalistic and activist value [13]. Researchers have also explored the social norms of streaming given its "live" quality. Seering et al. found that viewers can also play an important role, shaping not only the streamer's content and approach, but also affecting other viewers' behavior through interactions such as moderation interventions [46]. In this section, we will first discuss prior work on entertainment-focused live streaming, which is the most popular topic on our targeted streaming platforms (i.e., Twitch.tv, YouTube Live). Then we will review work on knowledge-sharing-focused and programming-focused live streaming and discuss their differences.

2.2.1 Entertainment-focused Live Streaming. Live streaming has been primarily focused on entertainment content, such as eSports [20, 40], live events [19, 51], outdoor activities [33], or news updates [25, 48]. Streaming platforms like Twitch, which was founded in 2012, originally focused on game-related content. Prior work in entertainment-focused live streaming found two reasons viewers engage in live streaming: they are attracted to the unique content of the stream, and they like participating in that stream's community [21]. They also revealed a series of issues in the live streaming community, namely vicious trolling and harassment [46, 55].

2.2.2 Knowledge-sharing-focused Live Streaming. Other work has also studied knowledge sharing via live streaming. Fraser et al. explored how people share knowledge via creative live streams where the content focuses on creating a novel artifact, such as digital visual art [15]. Chen et al. studied multimedia tools to enrich interactions in language learning streaming [45]. Lu et al. investigated the sharing of intangible cultural heritage artifacts (ICH) in China via live streaming [32]. They found that streamers' motivations were less financially driven and more altruistic: streamers are driven to develop online communities with unique identities and to share knowledge through meaningful conversations. Other than skill-related knowledge-sharing, Lu et al. found 68% of their survey respondents use live streaming to share their life experiences, such as dealing with pressure from work [35]. They found that sharing life experiences in the stream encourages more conversational interactions between streamers and viewers, giving viewers opportunities to learn more about streamers and making them more willing to continue following the streamers.

Unlike entertainment-focused live streaming, streamers in knowledge-sharing streams found it challenging to keep their viewers engaged while focusing on more performative activities. In the same manner, viewers found these knowledge-sharing streams to be less efficient and engaging in terms of streamer-viewer interaction. Also, viewers wished the streaming archives had better documentation for further consumption [34]. These concerns imply that existing streaming platforms need to make changes to adapt to the interaction needs of knowledge-sharing-focused streams.

2.2.3 Programming-focused Live Streaming. Growing interest in the online streaming community has led to increased research related to programming live streams as a phenomenon in CS education. Haaranen identified educational moments in questions posed in programming stream chats [18]. Faas et al. conducted a participant-observer study that observed nine streamers and interviewed five streamers and four viewers regarding their live streaming programming experiences [14]. The authors reported that viewers played a key role while watching the stream due to their interactions with the streamer and each other. They also showed how viewers mentor both each other and the streamer, and how this mentorship extends beyond the live stream itself. Alaboudi et al. conducted a similar preliminary study that compared live streaming programming to pair programming by watching 20-hour stream archive videos and surveying five streamers [4]. They identified two challenges: streamers struggled to maintain engagement with watchers while focusing on development activities, and viewers who entered the stream late were not able to easily explore the codebase to understand the task streamers were working on, limiting their ability to collaborate effectively with streamers.

All these studies have suggested that programming live streams have great potential as a compelling and novel form of online learning, yet these streams face a few unique challenges compared to those of streamers that focus on other content. First, unlike game playing or visual art creation, programming often involves frequent application switching, from programming integrated development environment (IDE) and program output to browsers and other support tools. This application switching makes programming live streams more difficult, as streamers often share one application at a time, so if viewers do not follow along, they might miss the context and thus fall behind. Additionally, debugging and on-the-fly learning (for streamers) are common activities while developing software. However, these activities can be time consuming and tedious, making the stream less engaging. Finally, software development often requires a series of configuration settings, meaning streamers must provide personal information such as API keys and passwords, which could cause privacy issues.

3 METHODS

To better understand these challenges and fill in the research gaps, we conducted semi-structured interviews with 20 participants who engage in live streaming programming. We recruit our participants via purposive sampling, combined with snowball sampling. The first eight streamers were recruited based on their high ranking in Twitch.tv's programming and software development directories and in YouTube's search results with queries such as "live streaming programming" and "live programming." The rest of the streamers and all of the viewers were recruited through snowball sampling. We reached out to interviewee candidates by email and Twitter with a short description of our research purpose and a background survey about their live streaming experience and demographic information. Based on their responses, we selected our set of final interviewees to achieve diversity in gender, age, and occupation.

Each interview lasted 40 to 75 minutes and was conducted via video conferencing software. Each participant received a thank you gift appropriate for programmers at the U.S. market rate ($100/hr) for their time after the interview. We designed the interview based on our research questions mentioned in the introduction. We began by asking interviewees about their general motivation for participating in (hosting/watching) live streaming compared to pre-recorded video. We asked how interviewees started live streaming, why they continue participating, how they grew their community, and their reasons for doing it more or less often. We then asked interviewees to open a stream archive they recently participated in. We sought to understand what interactions interviewees had with others before, during, and after live streaming sessions and what their intentions and concerns were. We asked interviewees about their best practices for protecting their

Participant	Occupation	Age	Platform	Programming Experience (yr)
S1	P	18 to 23	Y	15
S2	P	41 to 50	Y,T	4
S3	P	18 to 23	Y	3
S4	P	31 to 40	Y	2.5
S5	P	31 to 40	Y,T	20
S6	P	41 to 50	Y	4
S7	H	24 to 30	Y,T	15
S8	S	18 to 23	Y	6
S9	P	24 to 30	Y	10+
S10	P	Prefer not to say	T	26
S11	P	24 to 30	T	10
S12	P	31 to 40	Y,T	10
S13	H	24 to 30	T	15
S14	P	31 to 40	T	On and off 20 years
V1	H	18 to 23	Y	2
V2	S	18 to 23	Y	8
V3	S	24 to 30	T	5
V4	P	18 to 23	T	3
V5	S	24 to 30	Y	4
V6	P	41 to 50	Y	21

Table 1. Participants' background information. For the Participant column, S1-S14 represent the 14 streamer interviewees, and V1-V6 represent the 6 viewer interviewees. For the Occupation column, P = professional developer (paid to develop software), H = hobbyist programmer, and S = student programmer. For the Platform column, Y = YouTube and T = Twitch.

or others' sensitive information, for information about their concerns regarding live streaming, and what they wished to improve or to see improved regarding their privacy concerns.

We used a GDPR-compliant paid transcription service to transcribe all the interviews. The lead author first went through the transcripts along with the audio, corrected machine translation mistakes, and then exported the transcripts to a Google Document. The lead author then conducted iterative coding to divide the interview responses into major themes using an inductive analysis approach [11]. The coder coded the first few interviews to form an initial codebook, and then discussed with other authors to merge similar codes, and identify important emerging themes. With the new codebook, the coder coded all the 20 interviews while discussing and iterating key themes with other authors from time to time. We lightly edited interviewees' quotes for readability.

3.1 Interview Participant Backgrounds

Table 1 lists part of the participants' information. We did not report detailed information (i.e., gender and streaming content) for each participant, as the small size of certain communities might reveal personally identifiable information. We strove for diversity across multiple dimensions, such as gender, age, and streamed programming languages. The twenty participants have two main roles: streamers (14), whose primary job is narrating their thoughts along with their programming development, and viewers (6), who watch the stream and interact with others via a single thread chatroom. For each role, we also required certain backgrounds to focalize our target group: 1) for

both streamers and viewers, the participation frequency should be at least once a month to indicate that they have learned certain context-appropriate behaviors, such as viewers responding to each other in a timely fashion or streamers engaging viewers with stories, and they should also have previously produced or watched video recordings of programming content, 2) for streamers, the stream's purpose should be sharing programming knowledge and practical skills, 3) for viewers, they must often watch live streaming sessions to the end or near the end, eliminating those with little engagement. We will use S1-S14 and V1-V6 to represent the 14 streamer (S) and 6 viewer (V) interviewees, respectively.

Individuals participate in a diverse set of live streaming content, including a variety of programming languages such as JavaScript (8), Python (4), C# (3), Dart (2), Rust (2), and C (1). Out of the 14 streamers, 7 of them primarily stream while working through programming tasks related to open source projects. The number of live viewers per session ranges from 5 to 500 people on average. The majority of these participants self-identified as male during our interview (13). Regarding the other seven, 6 self-identified as female, with 4 out of these 6 identifying as transgender women. One interviewee preferred not to identify their gender; we will use "non-male participants" (six are streamers) to present these seven participants in the rest of our paper for convenience. Note that the two closest related prior works that focused on programming streams [4, 14] had all male participants.

4 STUDY DESIGN LIMITATION

While we strove to recruit participants from diverse demographic and professional backgrounds, our email list and snowball sampling had some limitations: our sample had only one female viewer, no one older than 51, and only three individuals from outside of the U.S. (Europe). Follow-up studies that recruit from broader demographics, such as participants from Asia, would improve the external validity of our findings.

In terms of participants' motivation, our focus was on those driven by the developer community itself, rather than on receiving payment (e.g., leading MOOCs) or other incentive mechanisms. Although some of the interviewees were paid for their streams, they reported that their primary motivation was education. Most of our streamers stream their work instead of conducting a lecture-style sharing of knowledge. However, we have seen a few other platforms (e.g., Douyu.com) in which the majority of the live streaming programming channels follow a prepared classroom style, a model to which our findings might not apply. Also, in terms of streaming platforms, our participants used only YouTube Live and Twitch.tv, so our findings cannot speak to those who participated in streams on other platforms, which might have different norms or technical differences. Future work can compare the norms across platforms by recruiting those who have had related experience.

5 RESULTS

In this section, we present the findings around our four research questions.

5.1 Why do streamers choose to share programming knowledge via live streaming?

We found four main reasons for which streamers would share their programming knowledge via live streaming as opposed to a recorded video: 1) live streaming is less time-consuming than recording a video, 2) streaming is more casual, while recording can feel more formal or performative, 3) streaming allows streamers to connect with and update the community, 4) streamers can gain rewards and assistance from viewers. We describe each of these reasons in detail below.

5.1.1 Live streaming takes less time compared to record a video. When asking why participants chose live streaming over video recording, half of the interviewed streamers (7/14) expressed concerns that video recording is a process that can *"tak[e] forever"* (S6), is *"tedious"* (S3), is a *"nightmare"* (S11), and is typically *"really hard"* (S10). The dread of arduous behind-the-scenes work along with the slower pace of content generation associated with pre-recorded videos are barriers that interviewees were able to overcome by using live streaming.

> After I've tried live stream [*sic*], I've actually shifted from doing pre-recorded videos because it takes a lot of time to, you know, think everything through, prepare this crap, and record and edit. Then you know, make everything look nice and then publish. While on the live stream, we can just hit the play button and then uploaded [*sic*] to YouTube and you're basically done and people are just as happy about it. -S5

5.1.2 Streaming is more casual, while recording is more like acting. Compared to recording a video, streamers also mentioned that their mindset and behaviors are different while live streaming. For video recording, a third of the streamers would shift to a *"flight attendant"* (S9) persona, where the voice-over primarily focuses on the content itself, which is fully prepared and scripted, as if they were *"acting"* for the *"virtual audience"* (S11). Alternatively, when live streaming these streamers found their dialogue to be *"a bit more casual"* (S11, S9) and *"spontaneous"* (S10), and they *"don't care about the risk [of making mistakes] at all"* (S9).

> In prepared content, it's going to be much more scripted. It's going to be much more acted, and I'm going to focus more on my facial expressions and make sure that you know what I intended to be and whatever work I'm creating. I think the prepared content is going to be like an actual performance that you can reshoot and whatnot. But in streaming, you can't really do that unless you're really good at, like, improv acting or something. -S11

5.1.3 Streaming is a new way to reach out and update the community. Half of the streamers (7/14) often stream about programming work related to open source projects or online programming communities. This work includes reviewing pull requests, adding new features, or resolving repository issues. Unlike the other half of the streamers, who have focused on building their own communities, these streamers are core contributors in existing open source projects or programming communities. Live streaming allows them to more easily connect with the developer community as a whole, sharing information and updates without excessive effort.

> [Live streaming] seems like an opportunity to reach developer communities in a new way that I hadn't tried to access before. -S10

We also found that streamers use live streaming as a communication platform to inform things the developer communities. One interviewee found streaming to be an effective method of reaching out to viewers in an existing developer community with quick updates on new features.

> Two years ago when I started at [Anonymous Community], we had four videos on the YouTube channel, and four of them were from when [Anonymous Community] was, like, an energy startup. So [the videos were] clearly not, like, aligned with our current vision, and having more technical content that I could produce without spending a ton of time on it was a nice way to go back [and update the users]. -S9

5.1.4 Streamers can gain rewards and assistance from viewers. When asked what made them continue streaming, the interviewed streamers mentioned that some of their motivation came from their audience. This motivation included live interaction around common interests (7), direct assistance from viewers (8), personal rewards (6), or indirect help (4). S2 said, *"[for] that pre-recorded*

material, you're never gonna get feedback on [it], you're never gonna interact with viewers on [it]. It's reference material." Additionally, three streamers benefited from their audience by using them as a sounding board or *"rubber duck"* (S3) for debugging or getting unstuck. Furthermore, hearing viewers' comments about the helpfulness of the streams felt personally rewarding for the streamers.

> It's just nice to hear when people show up and say 'Hey, just wanna let you know I got my first dev job, and thanks so much for keeping on.' -S5

5.2 Why do viewers choose to learn programming via live streaming?

Viewers' motivations found from prior work are limited, as some of them are inferred from behavioral observation by researchers [14]. Our findings revealed why, when, and how viewers choose whether or not to attend a live streaming session, watch a pre-recorded video, or watch a live stream archive. We describe each of these reasons in detail below.

5.2.1 Streaming creates an over-the-shoulder learning experience.

> It's not really about me trying to solve a problem. It's just me trying to learn more or see someone else coding. -V4

More than half of the viewers (4/6) reported that they have no specific *"problems to solve"* (V4,V3) or concrete learning objectives (V1,V5) when deciding to watch live streams. In contrast, when watching a pre-recorded video tutorial or attending online classes, these viewers mentioned that they often look for solutions to specific problems or have defined learning objectives in mind. Interviewees cited different reasons for this, including not approaching a live stream in the same way they approach a class, not having much information about the live stream provided ahead of time, and expecting to ask *in-situ* questions as the stream progresses. In fact, this phenomenon is similar to the concept of *informal learning*, or "over-the-shoulder learning." We will later discuss how live streams can benefit from this practice when used as a form of online learning.

Most viewers also mentioned that they are satisfied with the streams that they have watched fully, although only a few of them directly applied the techniques learned from the stream to their daily work. They said most learning happens when streamers share tips and tricks and explain the underlying structures of problems or applications.

> For more specific types of problems that I would have, like...I can also look [them] up on Stack Overflow. Like 'how do I fix this error?' So those are not the things that I would watch live streaming for. I'm going to the live stream more for, like, the structure of a project of the [chat]bot. -V4

5.2.2 Viewers like to see and hear streamers' thought processes. Content-wise, viewers often want streamers to share the thought processes behind the structure of their code, how they debug issues they encounter on stream, and how they learn new methods or techniques as they work. This finding is consistent with prior work on the motivation of viewers watching live programming [14, 43]. As viewers often bring this expectation to live streaming sessions, they may find it annoying when streamers are not actually live streaming but instead *"copying and pasting"* (V6) pre-written code.

> I like to see how the person thinks and makes mistakes, and a lot of people edit it too harshly. And then it ends up being this one fluid front-to-back program with no mistakes. They are not researching anything or looking anything up, and that's not realistic to me. So I think it's a little bit more entertaining, a bit more easy to sit through a lot of the live stream videos or even their archive videos. -V4

5.2.3 Real-time interaction makes the learning more authentic. More than half of the viewers (4/6) seek closer relations with others in the community as opposed to one-time, asynchronous interaction. *"I was looking for communities that I can join in,"* V4 said. *"They're almost like a mentor."* When asked how live streaming experiences are different from video recordings, V2 said *"It feels better, feels [like] you can interact with the actual streamer live and ask questions."* The real-time proceedings make viewers feel that *"they [streamers] can answer questions from you right there"* (V1). Similar to prior work in live streaming of other content [15], viewers also express their preference for watching certain live streams, as they favor the way that the streamer explains things.

> I think what pushes me to watch his [stream] specifically is that he has a reputation in my head of what he goes over. I know that what he says is going to be helpful. -V2

> If I find a person that teaches well, I usually end up following them. So then I kind of become their fan, and then I watch their live stream, learn more, and see coding in person. -V3

5.3 What are the barriers that streamers face when sharing programming knowledge via live streaming?

In this section, we focus on reporting streamers' barriers. Compared to prior work, we include topics that have not previously been considered, including privacy, preparation work for a streaming session, and issues that members of marginalized groups face.

5.3.1 Concerns about privacy leaking to an unknown audience. All streamers have concerns regarding the potential risk of exposing their personal information to an unknown audience. This information ranges from digital information such as email addresses, passwords, and API keys to physical information such as *"where I live"* (S5) or *"my family"* (S2). S2 raised some concerns:

> [It is] very easy to dox yourself. I have had guests on who inadvertently showed their phone number on stream. I've had issues where the API keys for various services have inadvertently been shown on stream.

Three of the streamers have live streamed products from the organizations that they work for, and all of them expressed the challenge of keeping their organizational information private, including *"unannounced projects"* (S11) or user-related information (S9). S9 said,

> My number one thing is that I will I never, ever, ever want to violate user privacy. That's why I never use a dev environment. I never use my admin account. I actually have a second non-admin account that I use [for streaming].

5.3.2 Preventing privacy leaks is a demanding and tedious task. Nearly all streamers (12/14) have found privacy leak prevention to be a challenge. These interviewees noted that they must carefully monitor both their current shared content as well as what they are about to share and say to avoid leaks. S3 said, *"recently I do work on [sic] projects in private repositories that aren't public. I'll need to keep in mind not to just open GitHub in the front page on the live stream. So I need to just go to a repository for example."* Even with such effort, these interviewees still occasionally encounter unintended information leaks:

> You code on screen, you dump an object [that] has a bunch of secrets in it. It happens. I average like one secret getting leaked out like every six hours of screen time. -S10

> You don't really realize how many times like a random API key or some other secret is in log files until you try and code on the stream. -S10

All streamers prepare to live stream by performing a series of protective actions to safeguard their sensitive information. Before live streaming, streamers log in to all necessary accounts (14

streamers), use anonymous mode in their browser (10), share only part of the screen (8), and use unrecognizable names or accounts only while streaming (6). This preparation can be very tedious, and even if these streamers are thoroughly prepared, we found that they were still worried about potentially leaking their privacy during stream.

> [The] easiest way is just to keep it away from anything related to the street, so I don't have any windows behind me. It's hard for people to tell where I am. I'm just doing it in an all closed room right now -S10

These worries over the potential risk of sensitive information leaks has changed streamers' behaviors and their strategies for preparing their content.

> If I'm signing in with Google on an application, and that is, like, recorded on the emulator, that would pop up all of my different emails and I would need to select one email. -S1

When we asked those streamers who had privacy concerns or incidents how they prevent or handle these incidents, the answers were mixed. Half of the streamers (6/12) reported that they do not do anything, as they think *"nobody made a big deal out of it"* (S9), and they trust their technology and platform to protect them (S10). The rest were very sensitive and cautious about their privacy and often applied advanced precautionary practices, such as "mute my mic while typing my password" (S7), or "notified the SWAT team in my city" (S10).

5.3.3 Gender related harassment is common and hard to prevent. More than half of the non-male streamers (4/6) said their biggest concern is gender harassment, such as receiving comments about their physical appearance. In contrast, none of the eight male streamers mentioned gender harassment as a concern during their interviews. One non-male streamer noted the frequent occurrence of these unwanted comments:

> For me, it's all gender aspect. So [people just keep] making comments about irrelevant things like my physical appearance and things like that. -S10

These comments are often from the live chat, which is difficult to ignore or filter out. However, some comments are made outside of live stream: *"I tend to get like more of the creepy messages to my like email or to my Twitter"* (S12). These comments have made streamers feel *"horrible"* (S9), *"uncomfortable"* (S12), and worried about their safety (S10).

Although all non-male streamers felt that the developer communities are *"pretty respectful"* (S12) and *"open to LGBT communities and friendlier in general"* (S11), none of the streamers believes the current tools on their streaming platforms are sufficient to prevent harassment. *"I wish I had one better set up so I could like block people from within the to overlay"* (S12). When harassment occurs, streamers often warn and ban the offending commenters while working to build a healthier community.

> If there are people who troll you and grassroots and whatnot, if you can get rid of them as soon as possible, then you're going to start building a community that's really supportive of you, and that support gives you a lot of strength. -S11

Three out of the six non-male streamers mentioned that finding other streamers who share a similar identity has helped them overcome this barrier: *"one of the reasons why I felt like it was OK to start was I saw another person doing it"* (S12).

5.3.4 Non-male streamers spent more effort signalling their ability.

> I think it also helps that I have a PhD. -S9

When asked why there are few non-male programming streamers, all six non-male streamers mentioned the imbalanced gender ratio in developer communities and the hassle of dealing with

potential harassment as two main reasons. S11 said, *"there are challenges that men just don't face when doing this type of stuff online. Men generally don't have or generally aren't being harassed for their gender."* We also noticed that all six non-male streamers validated their qualifications with evidence (e.g., a certification), while none of the male streamers we interviewed mentioned qualifications.

> [I] wrote the standard introduction book to [framework], and so most of the people in the [framework] world—not all, but most of them—are pretty respectful. -S12

5.4 What are the barriers that viewers face when learning programming via live streaming?

5.4.1 Related resources are not shared in real-time. As we mentioned previously, all viewers wish to follow streams without worrying about missing content while researching new information and questions. One viewer wished for note *"sharing on the fly."* Another viewer mentioned that materials should be shared before the stream starts.

> It would be good for the streamers, maybe before they do the live stream, [to] make an announcement and they release, like, resources that they're going to use during the coding. ... Other people can definitely follow along when they have a package and stuff. - V1

However, the live nature of streaming means that streamers must improvise their knowledge sharing from time to time. This modality makes it challenging for streamers to share materials in advance, since live streams are not typically scripted or planned concretely.

5.4.2 Finding high-quality and helpful live streams can be difficult. Unlike YouTube videos, which include indications of video quality (e.g., number of views, likes, comments.), there are often limited indications of a live stream's content quality. Previous viewers do not make comments or rate content, meaning that new viewers have little information to guide them to high-quality live streams that match their interests. Additionally, since streamers usually do not share their streaming resources in advance, it is difficult for viewers to quickly search and find a stream they would like to watch. Although the viewers in our study are experienced with live streaming, when asked how they choose which live streams to watch, some of them mentioned choosing solely based on the title, which can be inefficient.

> I just, I kind of look through and go by what the title is describing. And then after, I start watching it for like maybe 5 - 10 minutes. If it's going too slow or [it's] just not topics that I'm interested in, then I'll probably switch and find something. -V5

Other viewers have found their own methods of choosing a stream to watch, using indicators such as prior experience with the streamer or examining the stream's archives. For example, V4 uses their prior experiences with the streamer to indicate authenticity and content quality: *"[the streamer] has a PhD in computer science, and so when I watch it, I know that I'm getting legitimate, helpful knowledge."*

5.4.3 Impromptu research may be needed if viewers watch streams without specific problems in mind. As we reported earlier, without specific questions or learning objectives in mind, viewers may initially behave as"shoppers" who are looking for the right match, but this can be an inefficient process:

> I [don't] realize this topic isn't that much of interest to me till the streamers [start] coding, which is 20 minutes in already. -V5

Another barrier related to this mindset is that viewers may unexpectedly encounter new concepts during live streams and engage in impromptu research to learn more about them. This style of learning is similar to "over-the-shoulder learning," in which a learner looks over someone else's shoulder as they work on a project on screen [5].

However, due to the "live" nature of streaming, our viewers (learners) often felt they had "fallen behind" (V6) or were "lost" (V4) but did not want to interrupt the streamer due to social pressure. When asked what tools they wished they had to address this challenge, viewers expressed the desire for easy access to reference material.

> I mean, it would be cool to see some sort of, like, live description box where you can have it. -V2

5.4.4 Inaccessible streaming content hinders personalized learning. Learners have different needs and preferences, from their learning pace to the technology they like to use. The fact that live streaming is free and has no minimum requirement for the viewer's level of expertise also means that viewers often have more diverse backgrounds than those who attend structured classes. Five interviewees expressed their wish to enable more options for personalization, such as an adjustable pace or different content layout options.

> The text is quite large [on my screen], and I jump around the files a lot, so it may be confusing to people who haven't done reading everything I've typed. So I'd like there to be a way to view the code live in their editors instead of everybody seeing my screen, necessarily. -S3

We also noticed that different learning paces influenced viewing behaviors. One viewer coded along with the streamer consistently (V6), three took notes (V1,V4,V6), two watched the stream while taking notes, coding, or intermittently researching concepts (V4,V5), and two watched the stream without engaging in other behaviors (V2, V3).

> I would listen to the live stream, and then as they were talking I'd be, like, googling things or googling more about certain topics. -V5

All viewers thought coding along was *"difficult"* and *"tak[es] away from it [live streaming]"* during stream. V4 said, *"If I'm coding alone then I'm not focused on the stream itself. I could just go on a YouTube video pre-recorded."* The one viewer who coded along (V6) said,

> [I've] fallen behind because I'm doing research or just because he's moved fast. It would be pretty awesome [to] sync with him again, like a desperate [*sic*] at that point because I can't catch up to him if there's something wrong or I've missed a line or a typo and he's just cranking on.

5.4.5 Chat lacks context for others to understand. When asked about the live Q&A experience with viewers, all streamers reported that they have no problem understanding and responding to viewers' questions in the chat. This makes sense, as viewers' questions are often related to concepts that streamers have recently discussed. However, we found that both streamers and viewers had trouble understanding others' conversations in the chat. The challenge of understanding others' chat messages arises from shared context loss and the chat volume, which has been mentioned in other related work [34]. In the context of programming, we found the challenge comes from the fact that the code is constantly changing, which means viewers' references (e.g., line number) are only correct at the time the information appeared in the chat.

5.4.6 Stream archives can help dig deeper into the content but hard to navigate. All viewers mentioned that the information they seek when watching live stream archives is different from the information found in pre-edited videos. This information includes things like a streamer's *"thought*

Main Results	Design Implications
- Related resources aren't shared in advance - Finding quality, helpful live streams can be difficult - Impromptu research may be needed if viewers watch streams without specific problems in mind - Inaccessible streaming content prevents personalized learning	Designing personalized live learning environments
- Chat rooms lack methods of capturing context - Stream archive is useful but hard to navigate	New Tools for Content Retrieval and Understanding
- Concerns about privacy leaking to unknown audience - Protecting privacy leaking is a highly demanding and tedious task	Ease the effort of privacy protection
- Gender-related harassment is common and hard to prevent - Non-male streamers validated their skills with extra evidence	Designing healthier and cleaner learning environments

Table 2. Four design implications that are drawn from our results.

process," a *"Q&A section,"* or the live social aspects. V5 said, *"I think it's a little bit more entertaining, a bit more easy to sit through a lot of the live stream videos or even their archive videos."* This aligns with our earlier findings regarding viewers' motivations: viewers do not often decide to watch live streaming to solve a specific problem. Instead, they are typically patient, relaxed, and open to receiving new ideas when participating in live streaming.

Additionally, four out of six viewers have revisited stream archives that they attended the live version. This often occurs when viewers consider certain parts of the stream *"interesting"* (V4,V5) or when they encounter complex new content that they could not digest well while participating. Watching stream archives allows them to further analyze this knowledge and learn at their own pace.

> The topic is just hitting me the first time, and then if I notice later [that] I was like 'oh, I really have to get into this and use this,' that's, like, the first wave of knowledge and information about this thing. Then I'll go back and be like, let me break down this live stream or whatever I watched and, you know, plan out some topics that I'm going to go over myself and get more deeply into it and figure out what's up. -V5

When asked how they navigate the stream archives, all viewers mentioned that they find it difficult to locate specific information. This is because videos are often unplanned, and there are no timetables to facilitate skipping to specific content. Live streams are often very long, so useful information is more likely to be scattered throughout the video, and since the stream is not edited, not all of the content may be engaging or helpful.

6 DISCUSSION

In light of our findings, how can we redesign streaming platforms to better support this informal learning practice in developer communities? To address this question, we discuss four design implications in this section (Tab. 2).

6.1 Designing Personalized Live Learning Environments

Part of our participants' behaviors are driven by the streaming platforms, from their UI design to the social norms of their users. The UI of these platforms are inherited from the versions designed

with only gamers or entertainers in mind, not learners. The design principles used on these platforms focus on optimizing the UI for extended visual attention, such that viewers are less likely to look away from streamed content. However, prior work and our observations have shown that viewers can learn more effectively by looking away from the video content and digesting the incoming knowledge through other behaviors, like asking questions, taking notes, or doing relevant exercises [12]. The behaviors that benefit learners most are the opposite of what is encouraged by live stream platforms designed for entertainment content. This indicates a need for platform design updates that consider the context of learning, allowing platforms to meet the needs of creators and viewers of educational content.

Before online learning became popular, the education community was already grappling with the challenge of providing appropriate educational opportunities at scale in traditional in-person settings (e.g., in large lectures) [42]. As we found, the difficulty in adapting teaching and demonstration to viewers is due in part to differing expertise levels. We discuss the design implications based on our findings and prior work on learning at scale and personalized education.

Scale learning with better real-time information acquisition tool. Streaming platforms could better support serendipitous learning by integrating help-seeking tools to ease the process of knowledge acquisition in real-time. We propose that future streaming systems could automatically provide related resources as streamers narrate, such as specific document sections or concepts. This could be done by combining the techniques of auto-transcription and Google search APIs. Along with this feature, we suggest that platforms enable real-time code sharing, with streamers' permission. This would help those viewers who code along but worry that they "would have fallen behind" because they took the time to look up a new concept. Prior work has proposed effective tools with automation techniques [7, 22, 31], or crowdsourcing approaches [10, 28] to provide easy access to code-related support. We think combining these features would help viewers to learn at their own pace.

Personalized stream matching process. We can also consider mechanisms that help align a stream's difficulty level with the viewer's current knowledge and skills. This will help build towards a constructivist learning environment where viewers are more likely to retain knowledge attained by engaging in contextualised problem-solving [39]. Examples could include showing a visual "difficulty" badge for an upcoming stream, including a section of knowledge prerequisites in the stream's description and real-time displays of viewers' self-reported expertise level. This could allow streamers to adjust their teaching strategy and content, shaped indirectly by viewers, and potentially help indecisive viewers determine whether they should join the stream. Future work would be needed to conduct deeper analysis on streamers' live streaming styles to match them with viewers' viewing preferences.

6.2 New Tools for Content Retrieval and Understanding

A core challenge for many of our viewers was the difficulty of retrieving and understanding information that they missed or revisited. This information was mostly from the previous chat and the video content. In particular, the lack of context in the chat and poor tool support on video navigation can make viewing more difficult. We envision two directions for specialized tools to facilitate retrieving and understanding information: context captured chat interface, and lightweight stream archiving tools.

Context captured chat interface. The current chat tool on streaming platforms does not support context referencing (e.g., line number), which makes the chat hard to understand during programming streams. Additionally, the fact that streamers constantly change their code makes it more challenging to understand past references in the chat. Prior work has analyzed similar issues with referencing. For example, Chat.codes [37] enables students to link code snippets to their text

messages for context reference within IDE. Tools like Collaboratory [50] allow learning groups to create text and sketch annotations on existing videos. However, our problem requires context that might link to multiple points of the video. As a result, we suggest developing a chat interface organized by anchor points, such as a specific frame or a short clip, in the streaming content. In this way, the system would automatically anchor the stream content based on the chat time and natural language context.

Light-weight stream archiving tools. While most of the attention is focused on the live interactions during the stream, our reflections on the underlying causes for some of the challenges led to suspicions of low-quality stream archives. This includes 1) uncut irrelevant content (e.g., idle stream time), 2) no navigation support (e.g., a timetable), 3) a lack of relevant and interesting interactions during the stream. Ironically, all streamers had the desire to better archive the stream for various benefits, such as to build viewership or for future content reference. However, when we asked streamers about their process for documenting live streams, they expressed their willingness to add timetables but often felt "tired" after a few hours of live streaming, making it more difficult to follow through with the documenting process.

To address this challenge, prior work has attempted to leverage viewers to help with annotation and documentation to ease the on-boarding process [34]. However, they found that viewers often have low incentives to contribute to the work. We also asked viewers what kind of timetable they prefer; all of them ranked the key points discussed by streamers as the top choice. Automated tools that can help summarize or organize videos could allow viewers to quickly make sense of important and useful streamed content. We propose a light-weight annotation tool that allows streamers to take notes as they stream along, with automatically captured timestamps. Once live streaming is over, the timetable would be automatically generated.

6.3 Ease the Effort of Privacy Protection

The challenge of protecting privacy is much more difficult in live streaming than it is pre-recorded video. Unlike pre-recorded video, where people can always edit their recordings before posting, the nature of live streams makes it infeasible to edit their recording [29]. Writing scripts for the stream would not work, as interacting with the live audience would break the plan by introducing new content, which could result in streamers showing or saying sensitive information. Streaming programming work makes privacy control more challenging, as streamers constantly switch between applications (e.g., IDE, browser, terminal) for different coding-related tasks.

Prior work has studied people's privacy concerns regarding information disclosure during others' live video game streaming [30]. Given that their main findings are concerns that a participant might have regarding the team that she/he was in, their design suggestions do not apply in our case.

One approach would be to simply provide a privacy checklist that guides streamers through the steps that they need to take before streaming. An automatic checker could also require that certain privacy settings are in place, such as accessing the stream in incognito mode. Additionally, existing automated systems, such as OCR, can help detect patterns that are potentially sensitive, such as cell phone numbers, emails, or home addresses, and then we can use the existing user interface design to hide them or notify streamers. The computational cost of conducting OCR can be high when done in real-time. Instead, a post-hoc video process would be better to help reduce the risk of exposing sensitive information to malicious viewers.

6.4 Designing Healthier Learning Environments

While enhancing streaming techniques can help address certain problems in programming education, the solution to many other issues in this emerging socio-cultural environment may exist fully

or partially outside the purview of tools and technologies. Gender-related harassment, in particular, has been an issue in live streaming in other content areas, such as gaming [55]. In developer communities, gender-related harassment in live streaming is not yet well studied, but the general gender imbalance makes the issue even more prominent. Prior work has discussed such issues in the developer community on sites such as GitHub [24].

These problems were reflected during our research as well. Participants occasionally raised various gender-related issues and concerns during our interviews. Even during the recruiting process, we found it difficult to recruit non-male streamers (self-identified); it often took multiple searches across different platforms (e.g., GitHub) to contact them. This correlates to our finding that streamers are concerned about their privacy. Aside from gender-related harassment, we also observed skill-related stereotypes from the audience. Using our findings, we suggest a few social strategies and design adjustments to benefit both platforms and streamers wishing to create a healthy and safe learning environment.

Improving usability of the moderation tools. To make live streaming a healthier and gender-inclusive medium, we suggest the streaming platforms ask their streamers' gender and take gender-related harassment into consideration when developing their moderation tools. Also, a few streamers mentioned that they did not set up the platform's built-in moderation tool because they perceived it to be too complicated to initiate. Streaming platforms could make the moderation setup process easier in the future.

Incentivizing viewers to help moderate the chat. Prior work has studied moderation in live streaming for games. Researchers found that viewers' motivation to volunteer to help moderate streams is not very high. During our research, we asked viewers how willing they would be to help streamers to moderate the chat. Some viewers expressed concerns, such as the excessive time and commitment needed, and noted that their motivation depends on the streamer in question. For viewers who have followed a certain streamer for an extended time or who have developed a close relationship with the streamer, they expressed a willingness to help moderate the streamer's chat. In the live streaming programming community, the audience size is usually not large, making it even more difficult to find a volunteer to help moderate for the entire session. We propose that the streamers or the streaming platforms could develop incentives, such as gamification or various gifts, for those who moderate the chat. They might also allow viewers to volunteer for only a portion of the entire streaming time.

7 CONCLUSION

In this research, we sought to understand the benefits and challenges of live streaming as a new way to share and acquire technical knowledge. In particular, we aimed to position live streaming in the ecosystem of online programming education by comparing it to traditional video-based learning experiences. Through in-depth interviews with 14 streamers and 6 viewers who regularly participate in programming live streams, we identified the unique educational value live streaming affords due to its improvisational, real-time, and authentic nature. We found that these features of live streaming not only lower the barrier to sharing knowledge but also allow the viewer to shape the content being taught while observing error recovery and workarounds they would otherwise miss in well-edited videos. Live streaming thus plays an important gap-filling role in the rather crowded space of online learning technologies.

Nonetheless, our study also revealed a number of challenges users face in education-focused live streaming. In addition to a lack of features specifically designed to support learning in video live streaming platforms, streamers also have to deal with privacy risks and sometimes online harassment. Drawing from our findings, we have suggested better tools and strategies for leveraging

the beneficial properties of live streaming and mitigating the drawbacks and risks of participation. We contribute a critical examination of this young practice to the HCI literature and several design directions to evolve live streaming platforms into safer, more effective, and more inclusive learning environments.

REFERENCES

[1] 2020. Coursera. https://www.coursera.org Accessed: Jan., 2020.

[2] 2020. Khan Academy. https://www.khanacademy.org Accessed: Jan., 2020.

[3] 2020. Youbue, Google. https://www.youtube.com Accessed: Jan., 2020.

[4] Abdulaziz Alaboudi and Thomas D LaToza. 2019. An Exploratory Study of Live-Streamed Programming. *VL/HCC* (2019).

[5] Liam J Bannon. 1986. Helping users help each other. In *User centered system design*. CRC Press, 399–410.

[6] Jens Bennedsen and Michael E Caspersen. 2005. Revealing the programming process. In *ACM SIGCSE Bulletin*, Vol. 37. ACM, 186–190.

[7] Joel Brandt, Philip J Guo, Joel Lewenstein, Mira Dontcheva, and Scott R Klemmer. 2009. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1589–1598.

[8] Rovy F Branon and Christopher Essex. 2001. Synchronous and asynchronous communication tools in distance education. *TechTrends* 45, 1 (2001), 36.

[9] John Bransford. 2007. Preparing people for rapidly changing environments. *Journal of Engineering Education* 96, 1 (2007), 1–3.

[10] Yan Chen, Sang Won Lee, Yin Xie, YiWei Yang, Walter S Lasecki, and Steve Oney. 2017. Codeon: On-demand software development assistance. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 6220–6231.

[11] Juliet Corbin, Anselm L Strauss, and Anselm Strauss. 2015. *Basics of qualitative research*. sage.

[12] National Research Council et al. 2009. *Learning science in informal environments: People, places, and pursuits*. National Academies Press.

[13] Audubon Dougherty. 2011. Live-streaming mobile video: production as civic engagement. In *Proceedings of the 13th international conference on human computer interaction with mobile devices and services*. ACM, 425–434.

[14] Travis Faas, Lynn Dombrowski, Alyson Young, and Andrew D Miller. 2018. Watch me code: Programming mentorship communities on twitch. tv. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 50.

[15] C Ailie Fraser, Joy O Kim, Alison Thornsberry, Scott Klemmer, and Mira Dontcheva. 2019. Sharing the Studio: How Creative Livestreaming can Inspire, Educate, and Engage. In *Proceedings of the 2019 on Creativity and Cognition*. ACM, 144–155.

[16] Rex E Gantenbein. 1989. Programming as process: a "novel" approach to teaching programming. In *ACM SIGCSE Bulletin*, Vol. 21. ACM, 22–26.

[17] Philip J Guo. 2018. Non-native english speakers learning computer programming: Barriers, desires, and design opportunities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 396.

[18] Lassi Haaranen. 2017. Programming as a performance: live-streaming and its implications for computer science education. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 353–358.

[19] Oliver L Haimson and John C Tang. 2017. What makes live events engaging on Facebook Live, Periscope, and Snapchat. In *Proceedings of the 2017 CHI conference on human factors in computing systems*. ACM, 48–60.

[20] Juho Hamari and Max Sjöblom. 2017. What is eSports and why do people watch it? *Internet research* 27, 2 (2017), 211–232.

[21] William A Hamilton, Oliver Garretson, and Andruid Kerne. 2014. Streaming on twitch: fostering participatory communities of play within live mixed media. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 1315–1324.

[22] Andrew Head, Codanda Appachu, Marti A Hearst, and Björn Hartmann. 2015. Tutorons: Generating context-relevant, on-demand explanations and demonstrations of online code. In *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 3–12.

[23] Richard Holliman and Eileen Scanlon. 2013. *Mediating science learning through information and communications technology*. Routledge.

[24] Nasif Imtiaz, Justin Middleton, Joymallya Chakraborty, Neill Robson, Gina Bai, and Emerson Murphy-Hill. 2019. Investigating the effects of gender bias on GitHub. In *Proceedings of the 41st International Conference on Software Engineering*. IEEE Press, 700–711.

[25] Gerald Iorio. 2016. TrumpClimber. In *Periscope*. https://www.periscope.tv/w/1vOxwpkOjYLKB
[26] Juho Kim, Philip J Guo, Daniel T Seaton, Piotr Mitros, Krzysztof Z Gajos, and Robert C Miller. 2014. Understanding in-video dropouts and interaction peaks inonline lecture videos. In *Proceedings of the first ACM conference on Learning@ scale conference*. ACM, 31–40.
[27] Marc Kruger and Reinhold Nickolaus. 2005. Self-directed and cooperative learning with lecture recording. *PhD Dissertation, Available at http://stadium.open.ac.uk/prolearn/summer05/documents/marc_krueger_abstrakt_of_dissertation.pdf* (2005).
[28] Thomas D LaToza, W Ben Towne, Christian M Adriano, and André Van Der Hoek. 2014. Microtask programming: Building software with a crowd. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 43–54.
[29] Sang Won Lee, Aaron Willette, Danai Koutra, and Walter S Lasecki. 2019. The Effect of Social Interaction on Facilitating Audience Participation in a Live Music Performance. In *Proceedings of the 2019 on Creativity and Cognition*. ACM, 108–120.
[30] Yao Li, Yubo Kou, Je Seok Lee, and Alfred Kobsa. 2018. Tell Me Before You Stream Me: Managing Information Disclosure in Video Game Live Streaming. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 107.
[31] Xieyang Liu, Jane Hsieh, Nathan Hahn, Angelina Zhou, Emily Deng, Shaun Burley, Cynthia Taylor, Aniket Kittur, and Brad A. Myers. 2019. Unakite: Scaffolding DevelopersâĂŹ Decision-Making Using the Web. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. ACM.
[32] Zhicong Lu, Michelle Annett, Mingming Fan, and Daniel Wigdor. 2019. I feel it is my responsibility to stream: Streaming and Engaging with Intangible Cultural Heritage through Livestreaming. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 229.
[33] Zhicong Lu, Michelle Annett, and Daniel Wigdor. 2019. Vicariously Experiencing it all Without Going Outside: A Study of Outdoor Livestreaming in China. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 25.
[34] Zhicong Lu, Seongkook Heo, and Daniel J Wigdor. 2018. StreamWiki: Enabling Viewers of Knowledge Sharing Live Streams to Collaboratively Generate Archival Documentation for Effective In-Stream and Post Hoc Learning. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 112.
[35] Zhicong Lu, Haijun Xia, Seongkook Heo, and Daniel Wigdor. 2018. You watch, you give, and you engage: a study of live streaming practices in China. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. ACM, 466.
[36] Aidan McGowan, Philip Hanna, and Neil Anderson. 2016. Teaching programming: understanding lecture capture YouTube analytics. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 35–40.
[37] Steve Oney, Christopher Brooks, and Paul Resnick. 2018. Creating Guided Code Explanations with chat. codes. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 131.
[38] John Paxton. 2002. Live programming as a lecture technique. *Journal of Computing Sciences in Colleges* 18, 2 (2002), 51–56.
[39] Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Bennedsen, Marie Devlin, and James Paterson. 2007. A survey of literature on the teaching of introductory programming. In *ACM sigcse bulletin*, Vol. 39. ACM, 204–223.
[40] Anthony J Pellicone and June Ahn. 2017. The Game of Performing Play: Understanding streaming as cultural production. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 4863–4874.
[41] II Phase et al. 2005. *Educating the engineer of 2020: Adapting engineering education to the new century*. National Academies Press.
[42] Joseph Psotka, Leonard Daniel Massey, and Sharon A Mutter. 1988. *Intelligent tutoring systems: Lessons learned*. Psychology Press.
[43] Adalbert Gerald Soosai Raj, Jignesh M Patel, Richard Halverson, and Erica Rosenfeld Halverson. 2018. Role of Live-coding in Learning Introductory Programming. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*. ACM, 13.
[44] Marc J Rubin. 2013. The effectiveness of live-coding to teach introductory programming. In *Proceeding of the 44th ACM technical symposium on Computer science education*. ACM, 651–656.
[45] Nurain Adila Abdul Samat, Harwati Hashim, and Melor Md Yunus. 2019. Live Streaming: A New Platform for ESL Learning. *Creative Education* 10, 12 (2019), 2899–2906.
[46] Joseph Seering, Robert Kraut, and Laura Dabbish. 2017. Shaping pro and anti-social behavior on twitch through moderation and example-setting. In *Proceedings of the 2017 ACM conference on computer supported cooperative work and social computing*. ACM, 111–125.

[47] Neil Selwyn. [n.d.]. Web 2.0 applications as alternative environments for informal learning-a critical review.
[48] Catherine E. Shoichet. 2016. Facebook Live video offers new perspective on police shootings.. In *CNN*. http://www.cnn.com/2016/07/07/us/facebook-livevideo-minnesota-police-shooting/index.html
[49] Abdulhadi Shoufan. 2019. Estimating the cognitive value of YouTube's educational videos: A learning analytics approach. *Computers in Human Behavior* 92 (2019), 450–458.
[50] Vikash Singh, Sarah Abdellahi, Mary Lou Maher, and Celine Latulipe. 2016. The video collaboratory as a learning environment. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 352–357.
[51] John Tang, Gina Venolia, Kori Inkpen, Charles Parker, Robert Gruen, and Alicia Pelton. 2017. Crowdcasting: Remotely Participating in Live Events Through Multiple Live Streams. *Proceedings of the ACM on Human-Computer Interaction* 1, CSCW (2017), 98.
[52] John C Tang, Gina Venolia, and Kori M Inkpen. 2016. Meerkat and periscope: I stream, you stream, apps stream for live streams. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 4770–4780.
[53] Moshe Y Vardi. 2012. Will MOOCs destroy academia? *Commun. ACM* 55, 11 (2012), 5–5.
[54] Dawn Wilson and David Allen. 2011. Success rates of online versus traditional college students. *Research in Higher Education Journal* 14 (2011).
[55] Donghee Yvette Wohn. 2019. Volunteer Moderators in Twitch Micro Communities: How They Get Involved, the Roles They Play, and the Emotional Labor They Experience. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 160.