Beyond Show of Hands: Engaging Viewers via Expressive and Scalable Visual Communication in Live Streaming

John Joon Young Chung
University of Michigan
jjyc@umich.edu

Hijung Valentina Shin
Adobe Research
vshin@adobe.com

Haijun Xia
UC San Diego
haijunxia@ucsd.edu

Li-Yi Wei
Rubaiat Habib Kazi
Adobe Research
{lwei,rhabib}@adobe.com

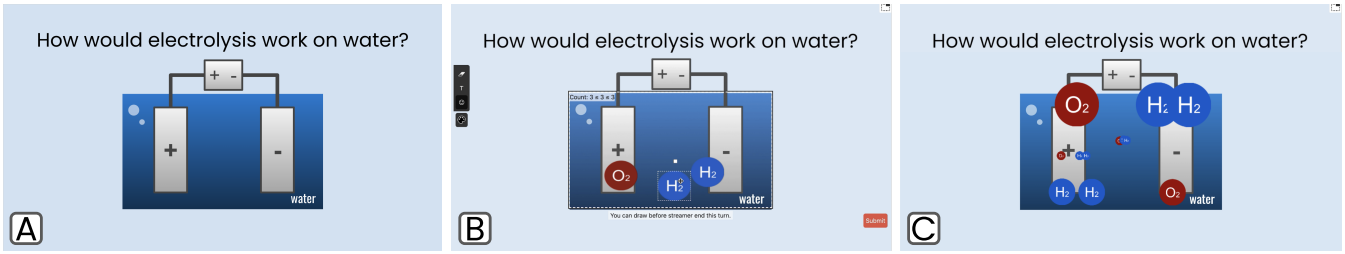


Figure 1: *Visual communication with our system, VisPoll.* VisPoll allows viewers to make visual input during live streaming or live online lectures. (A) Streamer prompts viewers for input about the position of hydrogen and oxygen molecules. (B) Viewers provide input by arranging the molecule symbols given by the streamer. (C) Streamer aggregates the viewers' inputs and presents a summary visualization to the viewers. Each row indicates different clusters of viewers' input, with the size of the molecules indicating the number of viewers corresponding to the cluster. Here we show only the viewer interface; the corresponding streamer interface is shown in Figure 8.

ABSTRACT

Live streaming is gaining popularity across diverse application domains in recent years. A core part of the experience is streamer-viewer interaction, which has been mainly text-based. Recent systems explored extending viewer interaction to include visual elements with richer expression and increased engagement. However, understanding expressive visual inputs becomes challenging with many viewers, primarily due to the relative lack of structure in visual input. On the other hand, adding rigid structures can limit viewer interactions to narrow use cases or decrease the expressiveness of viewer inputs. To facilitate the sensemaking of many visual inputs while retaining the expressiveness or versatility of viewer interactions, we introduce a visual input management framework (VIMF) and a system, VisPoll, that help streamers specify, aggregate, and visualize many visual inputs. A pilot evaluation indicated that VisPoll can expand the types of viewer interactions. Our framework provides insights for designing scalable and expressive visual communication for live streaming.

CHI '21, May 8–13, 2021, Yokohama, Japan

ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3445419

CCS CONCEPTS

• **Human-centered computing** → **Collaborative and social computing systems and tools**;

KEYWORDS

live streaming, user engagement, sketching, visualization

ACM Reference Format:
John Joon Young Chung, Hijung Valentina Shin, Haijun Xia, Li-Yi Wei, and Rubaiat Habib Kazi. 2021. Beyond Show of Hands: Engaging Viewers via Expressive and Scalable Visual Communication in Live Streaming. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3411764.3445419

1 INTRODUCTION

Over the past few years, live streaming, online videos broadcasted in real-time, is becoming increasingly popular in a wide range of domains, including gaming [16], art creation [11], finance [36], cooking [36], programming [9], and language learning [5]. The usage of real-time broadcasting of videos has increased more after the outbreak of COVID-19. For example, video conferencing tools have been used for virtual events, conferences, and remote lectures for K-12 and university-level education [31].

In live streaming, there are usually multiple viewers who actively communicate with the streamer and other viewers, enriching the content. Viewers usually interact via chat messages, conveying opinions and questions to the streamer or other viewers [9, 36]. However, as chat is separated from the video and takes the form of text, it has limited expressiveness when communicating around visual and spatial elements in the video. Visual input overlaid on the

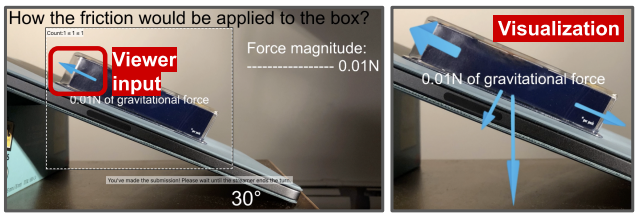


Figure 2: *Use case: Physics lecture.* The viewers are asked to annotate friction with an arrow. They can convey magnitude information with the length, and direction with the rotation of the arrow, along with the position information. The streamer can aggregate and visualize visual inputs, with the width of the arrow indicating the number of viewer inputs in each cluster.

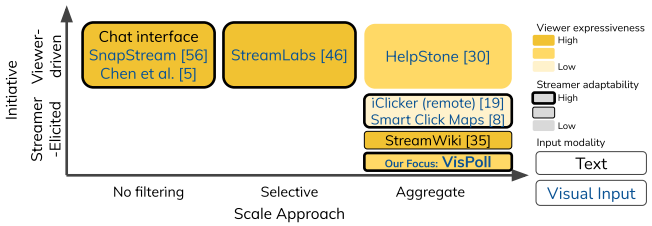


Figure 3: *Design space of viewer input systems in live streaming.* Our focus is on enabling sensemaking of viewer visual inputs through aggregation, by allowing streamers to prompt and manage viewer interactions.

video, like visual annotations, can further enrich communication in live streaming [21]. For instance, it allows viewers to easily refer to a part of the video [12, 56], express their ideas in rich ways [21], and help them consume streaming content actively with diagramming [7, 28, 45].

With these benefits in mind, some recent live streaming and video conferencing systems allow viewers to make visual inputs [19, 30, 46]. A core challenge in such systems is to manage and understand viewers' collective inputs. While scalability is a prominent problem for text-based interactions [16], it is even more challenging for visual inputs, as they lack appropriate structure for aggregation and sensemaking. To mitigate this problem, existing tools give up certain benefits of visual inputs by either limiting the number of viewers who can provide inputs [46] or by restricting their expressiveness [19, 30].

We conducted a formative study to understand challenges in enabling scalable visual interaction, and found a tension between viewer expressiveness, coherency, and low interaction threshold. Streamers mentioned that expressive viewer inputs tend to vary a lot between viewers, leading to low coherency and difficulty in understanding when there are many of them. Furthermore, expressive viewer inputs such as free form drawing or diagrams require a higher threshold of skill on the viewers. Finally, streamers also wanted the viewer inputs to be general and adaptable to their streaming content and settings.

Based on findings from the formative study, we introduce a visual input management framework (VIMF), which lets streamers specify, aggregate, visualize, and perform sensemaking on multiple viewer inputs with low threshold, high adaptability, and desired expressiveness (Figure 1). We enable this by structuring viewer visual inputs with the combination of visual attributes like shape, color, size, rotation, and position, so that visual inputs can be flexibly controlled and managed. Visual attribute structure allows streamers to specify and constrain the visual input. For instance, in Figure 1B the streamer is asking where oxygen and hydrogen will be generated with electrolysis on water, and the streamer can allow viewers to answer this question by specifying shapes to only hydrogen and oxygen symbols, while allowing positioning them on the diagram of water. It allows required expressiveness to viewers, which is positioning objects, while preventing viewers from giving irrelevant input, like random drawing. As streamers can configure these visual inputs from their side, it would also reduce workload from the viewers. Most importantly, with the structure, the aggregation and visualization of visual inputs become possible, enabling the sensemaking of many viewer inputs (Figure 1C). Furthermore, our framework allows streamers to bring up a variety of viewer interactions with varying levels of expressiveness. For instance, for physics lectures on dynamics (Figure 2), viewers can be allowed to annotate force vectors with arrows, which convey rotation, length, and position information.

We instantiated VIMF into an initial system, VisPoll. We first introduce application use cases in three domains: educational lectures, creative live streaming, and game streaming. Then, we conducted a preliminary evaluation with 3 streamers and 43 viewers, to learn how streamers and viewers would use our framework. We found out that streamers in diverse application domains could leverage our tool to further engage viewers with visual input interactions. Furthermore, streamers used visualization features with diverse strategies to understand the overall answers from viewers. Viewers also found our VisPoll to be helpful, for active engagements and better understandings of fellow viewer inputs. We further discuss how our framework can be expanded to include more types of viewers-streamer interactions, with more expressive visual attributes and visualizations. Our framework introduces a way of enabling scalable visual communication from viewers in live streaming without losing adaptability or expressiveness in viewer interactions.

This paper contributes the followings: 1) Opportunities and challenges in using visual inputs from viewers. 2) VIMF, which allows sensemaking of many viewer inputs without losing expressiveness, having a low threshold and high adaptability. 3) VisPoll, an initial live streaming system instantiating VIMF. 4) Preliminary evaluation with 3 streamers and 43 viewers and diverse use cases demonstrating the potential of VIMF in multiple application domains.

2 RELATED WORK

2.1 Viewer Inputs in Live Streaming

In live streaming, viewer interactions have facilitated engagement [32, 36], the sense of community [32, 36], and exchange of information [6, 9, 11, 36]. While they have mainly been in chat [16], to maximize the benefits of viewer interactions, researchers and practitioners have been designing tools that allow viewers to communicate

with richer modalities, including images [5, 36, 56], videos [5, 46], audios [5, 17, 46], and controls in games [13, 29, 44]. Similarly, video conferencing tools also allow viewers to make inputs in various modalities, including video and audio [21]. Among these expanded modalities, visual inputs have a large potential in facilitating communication between viewers and the streamer with high expressiveness. For example, with visual inputs, viewers can more easily refer to a confusing part in the stream [12, 56].

However, existing tools for audience interactions suffer the problem of scalability. Many live streaming or video conferencing settings tend to have many viewers and as many existing systems allow viewers to make input at any time, unmanaged viewer inputs could easily overload the information channel. This has been a clear problem with chat-based interactions in live streaming [16], and is a generalizable problem for other modalities.

This problem made researchers and practitioners explore solution approaches (Figure 3). One set of solutions focused on systems where viewers take initiatives in making inputs. One specific approach is only showing selected inputs from viewers. For instance, many commercial live streaming systems only share viewer content when they donate to the streamer [46]. However, with this approach, the streamer cannot get the overall inputs from viewers. Another approach is giving structures to viewer inputs and aggregating them into a more succinct form [30]. However, it does not allow streamers to use it in other contexts than the specific setting.

Another set of systems allowed viewers to make inputs only when prompted by the streamer. It allows controllability in input flow and collection of homogeneous inputs. Some systems, like iClicker for remote learning [19] and Smart Click Maps [8], allow viewers to place a click on the video when prompted and show the aggregated view of clicks to the streamer. However, these systems did not allow expressive visual inputs. A more advanced version of the tool is StreamWiki [35], which collects textual inputs with prompting and aggregates them to create a summary of live streaming content. However, it only leveraged textual inputs, not visual inputs. Moreover, this system lacks flexibility on how viewer inputs can be visualized, as the streamer does not have control over it.

Building upon previous work, we propose an approach that helps presenters to understand many viewers' visual inputs. We hit the balance in strengths and limitations of previous tools, allowing more expressive visual inputs compared to previous tools that only allow simple visual inputs [8, 19], while allowing streamers to adapt our tool to diverse settings, unlike previous tools that only functioned in a specfic streaming setting [30]. We allow streamers to prompt viewers to collect visual inputs and do aggregation and visualization on collected inputs.

2.2 Polling Tools

There have been many tools for authoring and deploying polls to collect people's opinions and inputs [10, 39, 47]. Among them, some were designed for collecting audience responses in presentations, and questions were usually in multiple-choice [19, 24]. Forum [20] is one of the earliest systems that allows multiple-choice question polls while giving a remote presentation. Many commercial systems for multiple-choice question polls in virtual presentations were introduced afterwards [40, 50]. Among them, iClicker for remote learning [19] is the most similar to our work. It allows students to answer polling questions by placing visual markers on the presented slide. DataSelfie [26] is also similar to our work as it allows users to create questionnaires with visual elements. However, in DataSelfie, questions are created for answering qualitative and nuanced personal aspects. Our work extends previous work by allowing presenters to collect viewer inputs through rich visual elements and presents them in the visualization.

2.3 Visualization Autoring Tools

While many visualization tools have been developed, researchers have been pushing them to be more accessible to those without programming knowledge. Visualization toolkits and libraries such as D3.js [3], Vega [43], and ggplot2 [51] are on the one extreme which requires high expertise for users. Tableau lowers the required expertise for creating visualization by allowing users to create a visualization with drag-and-drop interactions, while automating visual encoding [49]. Lyra [42] and Data Illustrator [34] expanded flexibility and customizability for directly manipulated visualizations. DataInk [55] and Data-Driven Guides [27] allowed users to create a visualization with further expressiveness by allowing more flexible binding of data with diverse visual attributes, such as length, area, position, or color. DataQuilt even allowed users to leverage real images in visualization, by manipulating them with data values [57]. Our work is similar to these tools in that we also support users to do a visualization with direct manipulation and binding of visual attributes to data. However, our work's context is different in that we aim for allowing streamers to do visualization on visual inputs collected from viewers during live streaming. In our setting, input data is the viewer's created visual input, and we aggregate and visualize those inputs with input visual attributes.

2.4 Visualization of Mass Users

Visualization has been used as one of the approaches to ease the load of understanding a massive amount of data streaming in real-time, for social media [2, 4, 37], crowdsourcing [41], and education [14]. However, they did not visualize visual inputs from users. On the other hand, work like MudSlide [12] visualizes visual inputs from users, students' confusion marks put on educational slides. However, the collection and visualization did not happen in real-time. Our work allows streamers to aggregate and visualize visual inputs from many viewers in real-time live streaming settings.

3 FORMATIVE STUDY

To understand the current practice and needs around viewer visual input in live streaming, we conducted semi-structured interviews with three streamers and one expert researcher in live streaming. In the formative study, we primarily asked 1) how streamers interact with viewers, 2) how streamers leverage visual inputs from viewers, and 3) challenges in existing visual interaction approaches with viewers. We recruited a diverse set of participants to understand the broad needs of viewer visual interactions in live streaming. We recruited four practitioners: a secondary school teacher in social science in South Korea, who taught remote classes with around 20 students for a semester (F1), a full-time CS professor in the United States who taught classes with 200~300 students for half a semester

(F2), a creative live streamer in motion graphics, with 3 years of experience (F3), and an HCI researcher who has done live streaming research in diverse domains for four years with some experiences in doing gaming live streaming (F4). Interviews were recorded and transcribed. For the analysis, one author did iterative coding with inductive analysis, and other authors reviewed codes.

3.1 Findings

3.1.1 Expressive viewer inputs would facilitate viewer engagement and clear communication. Interviewees expected expressive visual inputs could increase engagement of viewers with interesting interactions. Participants in education domains (F1, F2) were eager to interact with the viewers to elicit their level of understanding, so that they can provide better lectures to viewers. For example, to engage viewers, F1 was asking viewers to create visual diagrams on the learned content with Google Jam Board. Participants also thought visual inputs can let viewers more accurately express their ideas. For example, in creative live streaming, F3 mentioned that when viewers were asking questions about the tool, as viewers might not know of the exact name for a button in the tool, they struggle to ask questions only with textual inputs. For such a case, F3 mentioned viewers "drawing on the screen" would be helpful. While viewer interactions can have benefits, streamers also mentioned possible challenges in using them. We explain them below.

3.1.2 Viewer interactions are hard to be coherent with expressive inputs. Interviewees expected that, when many viewers are allowed to make a flexible visual input, it would be easy to be *incoherent*. For instance, F3 mentioned: "If you have one person drawing a picture, it's good. But if it is ten people drawing a picture, it can be a mess." Furthermore, F4 mentioned that a lot of streamers were concerned about confronting unexpected trolling and abusive content when viewers are given flexible visual input tools. Due to these problems, with expressive viewer interactions, interviewees tend to invite a certain viewer and only interact with the viewer at a time. However, this is still limited as they cannot make sense of collective voices through viewer interactions.

3.1.3 Expressive viewer inputs can put a high bar for viewers to participate. Interviewees noted that expressive viewer interactions can be complex and put *higher threshold* for viewers. This high threshold can possibly hurt their self-efficacy and engagement. In an educational setting, F2 noted that keeping the viewer interaction at the right difficulty is important. Similarly, F4 noted that allowing viewers to do free drawing can be dangerous as they can lose the track of the streaming by focusing too much on the drawing.

3.1.4 Interesting viewer interactions are hard to adapt to other live streaming content or other contexts. Streamers mentioned difficulty in adapting viewer interactions. F1 mentioned that while more interesting and higher quality viewer interactions can be accomplished by preparation, it also forces the streamer to follow the prepared streaming design. Thus, such streaming would lack *adaptability* in reacting to viewers, which is one of the main benefits of doing live streaming. Furthermore, F1 preferred viewer interaction tools that are widely applicable to various settings, as F1 can reuse them for multiple rounds of live streaming with different content.

3.1.5 Other challenges: Interviewees also mentioned other challenges such as the time delay between a streamer and viewers, or the streamer's split of attention between viewer interaction interfaces and other interfaces [56].

3.2 Design Goals

Based on the formative study and previous work, we present design goals for viewer interactions through visual inputs. Previous systems showed trade-offs in designing viewer interactions, satisfying one strength among expressiveness, coherency, or adaptability, while having weakness to others (Section 2.1). Similarly, findings from formative study revolve around trade-offs in expressiveness (Section 3.1.1, 3.1.2, 3.1.3), coherency (Section 3.1.2), adaptability (Section 3.1.4), and having a low interaction threshold (Section 3.1.3). Followings are four design goals around these trade-offs:

G1-Expressiveness: Allow viewers to express themselves flexibly through visual inputs.

G2-Coherency: Collect relevant viewer inputs and present many viewer inputs in an easily understandable way.

G3-Low Interaction Threshold: Keep interaction easy so that viewers without much skills in creating visual inputs can engage.

G4-Adaptability: Allow streamers to adapt viewer interactions to the different content and changing context of live streaming.

4 VISUAL INPUT MANAGEMENT FRAMEWORK DESIGN

Based on identified design goals, we propose our visual input management framework (VIMF) (Figure 4), which allows streamers to *specify* what visual inputs viewers can create, *aggregate* collected viewer inputs, and *visualize* aggregated visual inputs in a coherent and understandable way. VIMF is an interaction framework, which describes generalizable design elements to allow streamers to understand many visual inputs without losing low threshold, high adaptability, and desired expressiveness in visual inputs.

For example, in Figure 1, a streamer is interested in eliciting viewer answers about "how electrolysis would work on water", and more specifically, "from where oxygen and hydrogen would be generated, at the anode or the cathode". As the streamer wants to only elicit positional information of oxygen and hydrogen, not their size or shape, the streamer can *specify* that viewers should use oxygen and hydrogen symbols given by the streamer and only change their positions (Figure 4A). After collecting visual inputs from viewers, the streamer would want to understand how viewers answered. This can be challenging if there are a lot of viewers. To address this challenge, our framework allows streamers to aggregate viewer inputs with visual aspects they consider (Figure 4B), which are positions of objects in Figure 1. After the aggregation, the streamer can make sense of the result by *visualizing* them on top of the video (Figure 4C), as in Figure 1C.

VIMF accomplishes this by structuring viewer visual inputs and visualizations as a combination of *visual attributes*. In the following, we explain what visual attributes are and how they enable our framework to address our design goals.

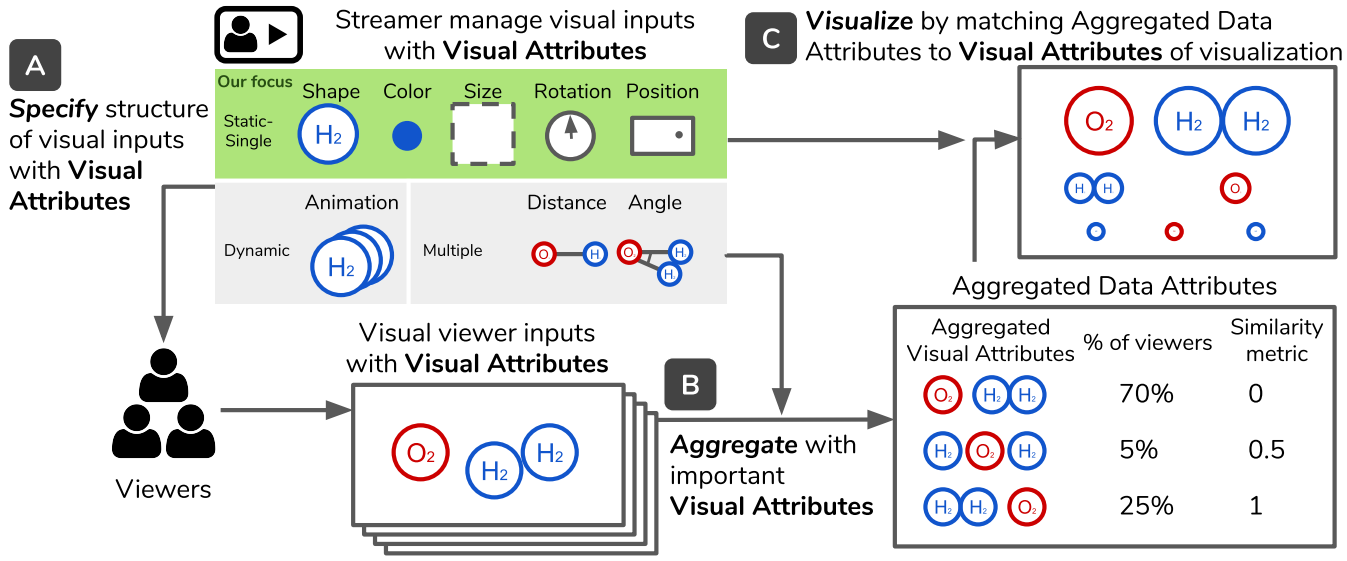


Figure 4: *Visual input management framework.* Our framework helps with sensemaking many visual inputs from viewers.

4.1 Visual Attributes

Visual inputs have different visual variables that distinguish them from each other, like shape, color, size, position, or rotation [38], and they can be leveraged to encode and convey information. For example, in Figure 1B, viewers can convey information about where each molecule will be generated by varying the *position* of the symbols. In this work, we consider such visual variables as *visual attributes*. Our framework structures visual inputs as the combination of these visual attributes, so that streamers can manage visual objects flexibly, but in an organized way. Note that visual attributes have a wide range, from those on a static, single object, to those on dynamic or multiple objects [54], like animation or distance between objects (Figure 4). While the full range of visual attributes can possibly be leveraged for VIMF, in this work we focus on a specific set of visual attributes. We explain our scope and rationale behind the scope in the following section (Section 5). In the following, we describe how visual attributes are leveraged to enable each step of our framework, and how they address design goals.

4.1.1 ***Specify*** *Visual Input Collection.* When streamers are eliciting visual inputs from viewers, they would want to collect inputs relevant to the streaming, while assuring the right level of expressiveness and difficulty. Structuring viewer inputs through visual attributes enable streamers to flexibly *specify* what visual inputs are allowed to viewers, addressing the aforementioned challenges. For instance, in the example in Figure 1, the streamer is interested in eliciting the positional information of the objects, while the size or the rotation is not crucial. Since visual attributes for a visual object can be controlled independently, with VIMF, in this case, the streamer would *loosen the restriction* of the position attribute, while *tightening the restrictions* on other visual attributes (e.g., size, rotation). This would enable viewers to manipulate the position only. It allows streamers to only collect relevant visual inputs (G2, G4) with enough expressiveness (G1) to viewers. Furthermore, based on the viewers' skill and expertise, the streamer can give more or fewer restrictions (G3). For instance, if viewers do not know that oxygen and hydrogen can be generated from either cathode or anode, streamers would be able to specify viewers to create visual objects only on anode and cathode. After the streamer is done with configuring the viewer input, the streamer can start collecting visual inputs from viewers.

4.1.2 ***Aggregate*** *Visual Inputs.* After collecting visual inputs from viewers, VIMF allows streamers to aggregate collected visual inputs, so that streamers and viewers can better understand collected inputs (G2). Visual attributes enable this by giving computational structures to visual inputs, which can be effectively aggregated with computational means. As not all visual attributes would be important in aggregating viewer inputs, our framework allows streamers to configure which visual attributes need to be considered for the aggregation. For instance, in Figure 1, as only the position of each molecule is important information, streamers can choose the position information to be considered for the aggregation. The aggregation would output aggregated clusters, with *aggregated data attributes*, which are the summarized variables of each cluster. Aggregated data attributes include the average of visual attributes from viewer inputs in the cluster, the number of visual inputs, and the similarity metric, which can be visual attributes reduced to a single value with techniques like PCA dimension reduction (bottom right of Figure 4).

4.1.3 ***Visualize*** *Visual Inputs.* While aggregation can effectively summarize received viewer inputs, the aggregation result by itself might not be understandable. To facilitate the understanding of summarized viewer inputs, the streamer can visualize the aggregated result on the video (G2). This is achieved with visualizations composed of visual attributes matched with data [34, 55]. In our framework, the visual attributes of visualization are matched with

aggregated data attributes. For instance, for visualization in Figure 1C, the sizes of objects in the visualization are used to indicate how many viewers answered in a certain way. The visualizations can bring further benefits if closely combined with the context of the underlying video content [52]. For example, in Figure 1C, as the video has visuals of anode and cathode, if visualization is done in relation to these, it would more effectively explain where viewers placed objects. Thus, while managing visualization mainly by matching with aggregated data attributes, our framework allows some flexibility in configuring the visualization layout (G4). We further explain which flexibility is allowed in our initial system out of the framework in the later section (Section 5).

5 VISPOLL: UI AND WORKFLOW

We instantiated the VIMF by designing it into the initial prototype, VisPoll. VIMF has the potential to be used for a wide range of settings with diverse visual attributes. However, in this work, we introduce an initial design of the framework, which considers the design goals and technical viability. We first describe the initial prototype's scope of design. Then, we introduce how we designed VisPoll, a prototype tool that enables visual polling by configuring viewer inputs, aggregation, and visualization.

5.1 Scope of Design

We first introduce the scope of design for VisPoll, why we chose specific visual attributes and visualizations.

5.1.1 Scope in Visual Attributes. We scope our design to mainly consider visual attributes on a static-single visual entity or object (Figure 4). Animations and groups of objects facilitate a richer set of visual attributes and parameterization. However, crafting animation and considering spatial relationships between multiple objects would require more complex interactions than specifying a single object. As viewer interactions need to have a low threshold (G3), we did not consider them in our initial prototype.

As the result, we scope on the following visual attributes, which we adopted from the principles for visual encoding [38]: shape, color, size, rotation, and position. For most of these attributes, they can be flexibly specified, aggregated, and visualized with existing techniques. However, for the shape, if inputs are highly expressive, like free sketches, there can be technical challenges in doing specification and aggregation, as free-form shapes have a very high degree of freedom (i.e., degree of freedom equal to the number of whole pixels). For expressive inputs, machine learning-based recognition might be able to be used for aggregation, but it would be limited as each live streaming channel is highly contextualized, requiring all different recognition systems for them. Furthermore, there is no clear approach to give restrictions on free-from drawing inputs. Hence, in the current design, instead of free-form sketches, we use *symbols*, or image icons, as discrete categories of shapes (e.g., image stickers). It also resonates with one of our design goals - the viewer interaction should not be complex (G3).

In addition to graphical primitives, VisPoll also provides visually anchored text boxes to facilitate expressiveness (G1). Texts can be aggregated via existing approaches such as tf-idf [23].

5.1.2 Scope in Visualization. VisPoll enables the direct configuration and manipulation of viewer input visualization. Within VisPoll, aggregated visual inputs from the viewers are visualized as glyphs (Figure 7A) for their effectiveness of conveying visual information [1]. The streamers can distribute these glyphs in *axis*-based visualizations (Figure 6) by assigning the desired aggregated data attributes to each axis. For example, in Figure 6, axes are positioning the glyphs according to the number of inputs in each cluster and x position, respectively. Furthermore, in Figure 6, the size of glyphs is used to visualize the number of inputs in each cluster.

VisPoll currently focuses on axis-based visualizations, as they are flexible enough to demonstrate diverse visualization views (Figure 7B), but also familiar enough to the general users. Prior systems in supporting the flexible construction of glyph visualizations can be applied to VisPoll to further improve its expressiveness [34, 55].

5.2 Setup and Implementation

We implemented VisPoll as a web application, using HTML, CSS, and javascript, with React as a front-end framework, and Feathers as a back-end framework. Feathers enabled our application to function in real-time. In our interface for streamers and viewers, we embed Twitch video through its API. Hence, both viewers and streamers can come to the url of our interface and use functionalities of VisPoll while having access to Twitch video.

For the broadcasting of the video, streamers can use applications like OBS[1]. These applications allow the flexible composition of the video with many visual sources, like a slide show or a capture of a video camera. With these broadcasting applications, the streamer should embed a visualization overlay as a browser overlay, to synchronize the video with the visualization.

5.3 Interaction and workflow

In this section, we explain the workflow and interactions in VisPoll with a running example. VisPoll is designed to allow streamers to configure visual interactions for viewers, even when they do not have programming expertise. Visual polls can be created in real-time during the streaming, or can be pre-configured as templates (Section 5.3.4). The user interface of the streamer is shown in Figure 8.

5.3.1 Specifying Viewer Visual Input. To create a visual poll, the streamer needs to specify three components - *region*, *shapes*, and *other visual attributes* that the streamer can allow viewers to control.

Region: The streamer first specifies a region by creating a rectangular area in the screen (Figure 9). All viewer interactions about visual inputs happen within this region. For example, in Figure 9A, the streamer specified a region on the graphical representation of water, the cathode, and the anode. This region is overlaid right on the video, and streamers can seamlessly combine it with the underlying video. Moreover, this region naturally gives restrictions on the positions and sizes of viewer-created visual objects. The streamer can create, revise, and delete the region.

Shape: The streamer can decide whether to enable the following two types of shapes: symbols and text boxes. For symbols, the streamer can add a new symbol either by drawing or uploading

[1] https://obsproject.com/

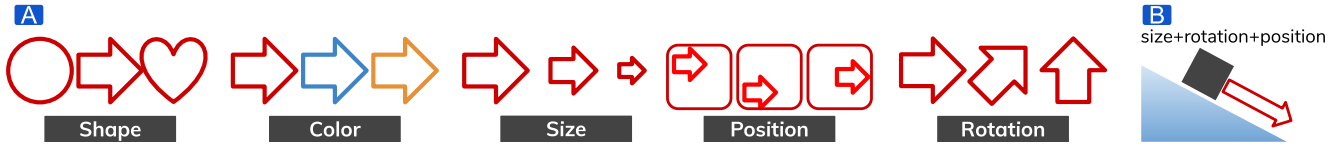


Figure 5: *Visual inputs in VisPoll.* A) Visual attributes. B) Multiple visual attributes can be combined together to convey information. Annotating net force vector applied on a box, using size, rotation, and position information.

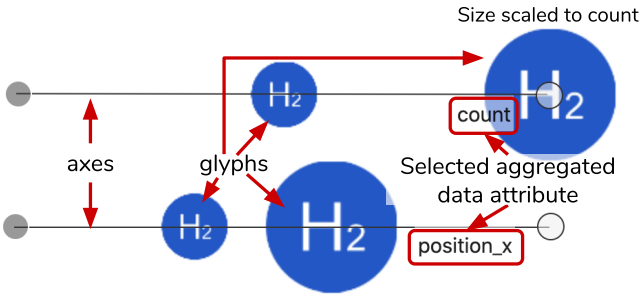


Figure 6: *Axis-based visualization.* Each axis visualizes clusters into glyphs, positioning them with an aggregated data attribute chosen by the streamer (count and x position). Visual attributes of glyphs (size) can be matched to aggregated data attributes (count) to convey information.

an image file. The streamer also can add preset symbols offered by VisPoll. With added symbols, the streamer can decide which symbol to allow in each area. For the case of Figure 9A, the streamer would upload images of oxygen and hydrogen to the tool. With the streamer's configuration, the viewers manipulate the shape of visual inputs with symbols or provide textual input on the screen.

Other Visual Attributes: After specifying regions and shapes, the streamer specifies other visual attributes, including rotation, color, size, and the number of objects in the area. The streamer can specify these visual attributes by giving restrictions. Essentially, these restrictions indicate to which values these visual attributes can be set by viewers. Streamers can specify valid visual attribute values in discrete values or continuous ranges. For example, in the electrolysis prompt (Figure 1), the streamer would want to tighten the restrictions for the size and the rotation of the object, as they are irrelevant to the task. For this case, the streamer can give one single discrete value for the size and rotation. For rotation, color, and the number of objects, the streamer can specify a discrete set or continuous ranges of values on wheels that are appended to each region (Figure 9B). On the other hand, restrictions on size are specified with direct manipulation, by directly creating these values on the area (Figure 9C).

In some cases, the streamer can initialize the visual objects. For instance, in Figure 9, the streamer initializes one oxygen and two hydrogen symbols by adding them in the region. When initializing objects in an area, the streamer is forced to follow the configured restrictions as in Figure 10.

5.3.2 Collecting Visual Inputs from Viewers. After configuring the visual inputs, the streamer can start collecting visual inputs from viewers. For this, the streamer would want the clear separation between when they are preparing input configurations and when viewers are allowed to make inputs. Thus, VisPoll has clearly separated turns for viewers to create visual inputs. During viewers' turn, VisPoll shows viewers the interface for making visual inputs with restrictions from the streamer (Figure 10).

Before starting the viewers' turn, the streamer also can configure how the turn ends, based on their needs. They can stop the turn manually, with a certain number of inputs from viewers, or with a timer. To let viewers know when the turn would possibly end, VisPoll shows the current status of the turn to viewers (e.g., the number of submitted viewer inputs). While viewers are allowed to make inputs, streamers can configure visualization, and when the turn ends, the visualization is presented to the viewers.

5.3.3 Configuring Aggregation. Once viewer inputs are collected, they can be aggregated by the streamer for further visualization and analysis. VisPoll enables the streamers to select 1) the visual attributes by which the viewer inputs are aggregated and 2) the number of clusters.

For the electrolysis example (Figure 1), to examine whether the viewers responded correctly to the streamer's question, the streamer can aggregate all the molecules by their positions, instead of other irrelevant visual attributes such as size or rotation. Streamers can also aggregate all data points to a few clusters to examine high-level patterns or to a number of clusters for low-level details, or examine individual viewer inputs as they were submitted.

Once the streamer specified the visual attribute and the cluster number for aggregation, for each data point, VisPoll concatenates the values of the selected visual attribute into a vector. For symbols, VisPoll appends symbol information into the vector as a one-hot representation. For texts, the system runs tf-idf [23] and appends the resulting scores from all words to the vector. Then, with these vectors from all viewer inputs, VisPoll runs hierarchical clustering [22] to create aggregated clusters. The aggregation produces aggregated data attributes for each cluster (Section 4.1.2).

5.3.4 Configuring Visualization. After viewer inputs are aggregated as clusters, streamers can configure and present glyph-based visualizations of the clusters to effectively communicate any pattern or insight of the data to the viewers.

The streamer can specify visual attributes for the axis, to distribute and visualize the glyphs representing the clusters. Two axes can be combined to create more complex visualizations (Figure 7), either connecting glyphs from the same cluster with lines (axes not in perpendicular), or creating 2-dimensional visualization such as scatter plots and bar plots (axes in perpendicular). For example, in Figure 11, the streamer configured the horizontal axis to indicate x

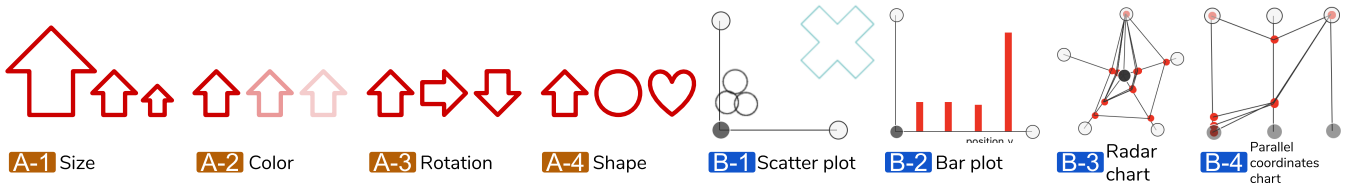
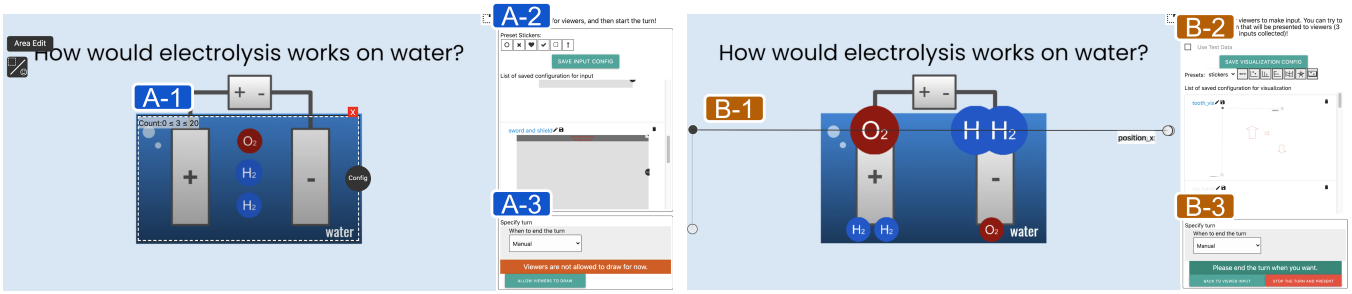


Figure 7: *Output visualizations in VisPoll.* Visualization with glyphs (A) and axes (B), reflecting aggregated data attributes either on visual attributes of the glyphs or position on axes.



a) Interface for specifying viewer visual input

b) Interface for configuring aggregation and visualization

Figure 8: *The streamer's interface.* a) Interface for specifying viewer visual input. A-1) A region for specifying visual attributes and visual inputs. A-2) Widget for symbol presets and saved visual input configurations. A-3) Widget for allowing viewers to start making visual inputs. b) Interface for configuring aggregation and visualization. B-1) Aggregated and visualized viewer inputs. Visualization is done with axis-based layouts and glyphs. B-2) Widget for visualization presets and saved visualization configurations. B-3) Widget for starting and ending presentation of visualization.

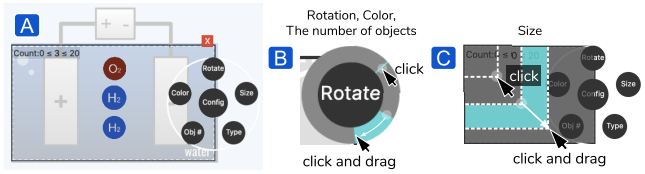


Figure 9: *Restricting visual attributes for viewer inputs.* A) A region where viewers can create and manipulate objects. Streamers can give restrictions on visual attributes with the Config widget on the right of each region. B and C) Specifying restrictions for input visual attributes.

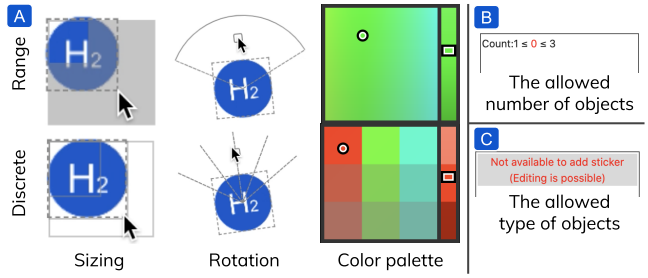


Figure 10: *Restricted object manipulation interactions.* The restriction can be put on size, rotation, color, the number of object, and the shape of object.

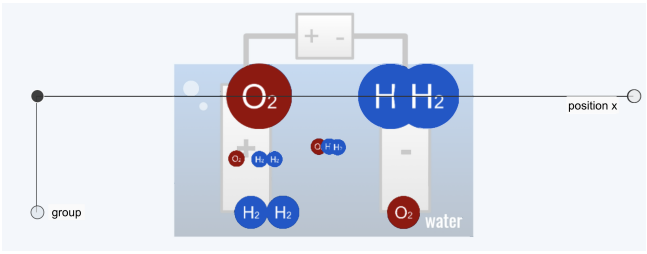


Figure 11: *Axes used for 2-dimensional visualization.*

position of objects and the vertical axis to show different categories of groups, according to their similarity. These axes are combined to make 2-dimensional visualization like a scatter plot.

Streamers can also encode aggregated data attributes with visual attributes of glyphs, like matching the number of inputs in each group to the size of glyphs (Figure 11). Built upon the flexible composition of visual encoding for expressive visualizations [34, 55], VisPoll enables streamers to flexibly map visual attributes of glyphs to any aggregated data attributes.

5.3.5 Configuration Library. As live streaming is a dynamic setting, the streamer would need to switch between different configurations during the presentation. However, doing all configurations on the fly can be overloading and take much time for streamers. To overcome this challenge, VisPoll has a configuration library, which stores both presets and presenter's own configurations. With this

library, first, streamers can store their own input and visualization configurations as templates before live streaming. During the streaming, they can select those configurations that most fit with the streaming content. Second, the library offers presets, which can be used as a starting point for improvising configurations. This configuration library would enable streamers to flexibly keep up with the dynamic nature of live streaming (G4).

For input configuration, VisPoll offers symbols as presets, which can be used for diverse purposes. For choosing preset symbols, we referred to existing graphics creation applications such as Google Slides[2] or Power Point[3], and included commonly appearing shapes as presets, such as hearts, circles, and arrows. For visualization, VisPoll offers preset view layouts, as they can be generally applied to different visualizations. Taking layouts from methods for making views [38], we included seven views that can be realized with axes: one-dimensional plot, scatter plot, vertical bar plot, horizontal bar plot, parallel coordinate chart, radar chart, and projection of viewer inputs as they were placed on the video.

6 APPLICATION SCENARIOS

In this section, we demonstrate the use cases of VisPoll in multiple application scenarios. In these demonstrations, viewers can leverage multiple visual attributes to express their answers, while streamers can effectively visualize them on the video.

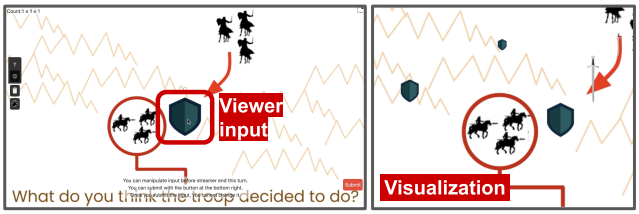


Figure 12: *Use case: History Lecture.* Predicting the action that the troop has taken. The viewer can indicate actions with symbols (shield or sword), and position. The visualization shows the aggregated summary of viewer inputs, with size showing the number of viewers.

6.1 Educational Lectures

In the education scenario, our tool can be used to conduct an in-lecture evaluation on viewers' understandings, getting the comprehensive feedback on how students understood the lecture. Science lecture is one specific use case, where information can be conveyed by multiple visual attributes. In physics, for example, the students can predict and answer the frictional force applied to the box (Figure 2), by drawing a vector on the real video [48]. In history lessons, the streamer can prompt viewers to predict or remember what decision a troop has made in history (Figure 12). In the above use cases, by actively creating visual diagrams, viewers would consume streaming content more critically, as previous research has shown that active diagramming helps with an accurate comprehension of information [7, 28, 45], and problem-solving [7].

[2]https://www.google.com/slides/about/
[3]https://www.microsoft.com/en-us/microsoft-365/powerpoint

6.2 Co-creation in Creative Live Streaming

VisPoll can also be used for collaborative creation in creative live streaming, by engaging and soliciting visual feedback from viewers. For example, in Figure 13A, the streamer asks with which color they should paint each part of the sketch. In another example, the streamer asks the viewer to leave feedback right on the intermediate art piece (Figure 13B). By leveraging visual inputs in the video, viewers can refer to the visuals directly.

6.3 Viewer Participation in Game Streaming

VisPoll can also be used for live game streaming, for interactions like streamers asking viewer's opinions about which action the streamer needs to take. For instance, Helpstone's stream overlay [30] can be replicated with VisPoll, which allows viewers to add arrows on the streamed video, indicating which action the streamer should take. Arrows added by viewers can be aggregated and visualized thicker if more viewers draw them similarly. With our tool, streamers can also choose to allow more freedom in drawing those lines or give more restrictions as Helpstone, allowing viewers to draw lines only with certain positions and angles.

7 PRELIMINARY EVALUATION

To gain insights about the viability and limitations of our tool, we deployed VisPoll in three real live streaming sessions and investigated how users interacted with it. Specifically, we focused on the following questions:

- Q1: Can streamers use VisPoll to come up with new types of visual input interactions that fit their streaming context?
- Q2: Does VisPoll enable expressive and easy visual interactions for viewers?
- Q3: Do aggregation and visualization help streamers and viewers understand the overall viewer inputs?

Note that we did not conduct a comparative study to previous tools as our focus is on investigating possibility of enabling new types of interactions with VisPoll.

7.1 Participants

We recruited streamers with prior experience in live streaming or remote lectures. S1 was a creative live streamer who had 5 years of experience in live streaming and streamed about twice a week. S2 was a Korean-language-arts teacher in a South Korean middle school. S3 was a computer science professor in the United States. Both S2 and S3 had remote teaching experience since the outbreak of COVID-19. For each streamer, we recruited viewers who were generally interested in each live streaming topic through a university mailing list and social media advertisements.

We recruited 15 viewers for S1, 14 for S2, and 14 for S3. S2's session was conducted in Korean, while others were in English. Only for S1, there were 2 non-recruited regular viewers, who voluntarily used VisPoll. The majority of viewers were recruited, not regular viewers, due to the practical concern: streamers were reluctant to use experimental tools with their existing viewers or students, as such a setting is closely related to their professional occupations and earnings. Still, our recruitment approach is valid as our focus was to see what visual interactions streamers and viewers would do

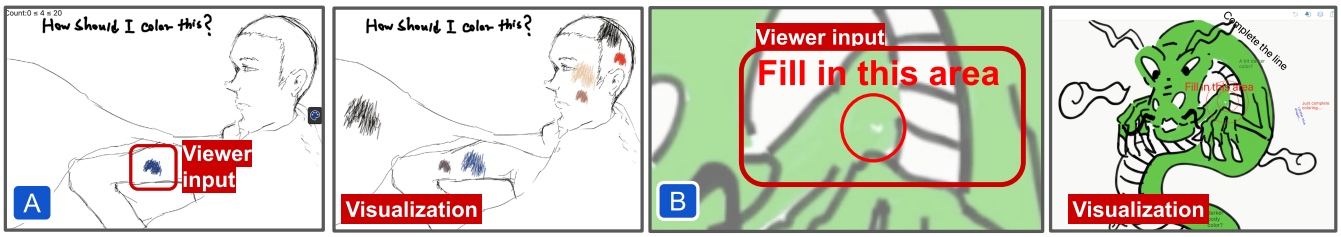


Figure 13: *Use case: Creative live streaming.* A) Collecting opinions on coloring. On the sketch, viewers can express their opinions by markers that convey color and position information. The streamer can visualize aggregated inputs, making sense of the number of inputs with size. B) Asking feedback on a drawing. Viewers can leave feedback on the adequate position of the drawing with text and symbol. The streamer can aggregate and visualize them to understand the overall feedback.

with our tool, instead of seeing how our tool would impact existing social dynamics in the streaming. Moreover, this approach has been widely used in the evaluation of live streaming systems [5, 17].

7.2 Procedure

The evaluation procedure is composed of three rounds of meetings with each streamer: 1) an initial meeting, 2) a pre-session rehearsal, and 3) a live session. In the initial meeting, we briefed about the purpose and use cases of the tool and conducted a tutorial on functionalities of VisPoll. Then, as homework, we asked streamers to bring up ideas on how they would use the tool for their sessions and bring configurations before the next meeting. The initial meeting took around one and a half hours.

In the pre-session, the streamer introduced configurations they created to the research team. We tried rehearsal on the tool and told them the procedure of the study for the live session. The pre-session took about an hour. Before the live streaming session, the streamer could ask questions about the difficulties they experienced with the tool to researchers.

With VisPoll and created configurations, the streamer conducted a live session. We asked participants to use Twitch for streaming the video. It can be limited that it does not allow certain interactions in video conferencing tools, such as seeing viewers' faces or hearing their voices. However, as our scope of interest is in studying effects of viewers creating visual inputs on the video content, this limitation is not a critical factor in our evaluation. The session took about 30~40 minutes. During the live session, we recorded the video of the session in the viewer interface.

After the live session, we conducted a post-session interview with the streamer. We also asked viewers to do a survey on their perceived engagement, helpfulness, and fluency in communication, in Likert-scale questions and open-ended questions. We present specific questions in the supplementary materials. We compensated each streamer $125 and each viewer $10.

7.3 Results

We present results based on observations of sessions, streamer interview data, and answers of viewer open-ended questions. Observations, streamer interview data, and viewer open-ended answers are analyzed by one author by iterative coding with inductive analysis, and other authors inspected it. We present quantitative results of viewer surveys in the supplementary material, as it generally showed viewers' favorability to our tool, but without further insights.

7.3.1 Q1: VisPoll enables visual input interactions that fit with the streamer's live streaming context. We first present how each streamer used VisPoll in their streaming context. Then, we explain their reaction to the authoring experience.

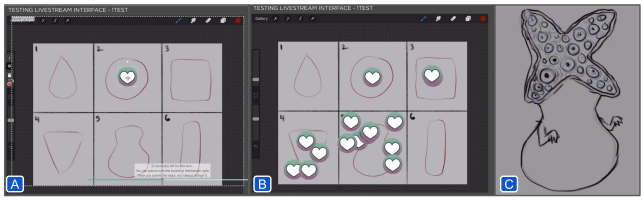


Figure 14: *S1's creative live streaming.* S1 used VisPoll to collect and visualize people's opinions on what S1 needs to draw. A) Collecting viewer inputs. B) Visualizing viewer inputs as raw viewer inputs. C) Final drawing

S1 - Getting opinions on what to draw. S1 did a session drawing a monster, considering the opinion of viewers. To visually get viewer opinions, S1 had a grid layout displaying different shapes. Viewers could decide one of them by placing their stickers on one of the grids (Figure 14A). The streamer checked which got the most votes by visualizing viewer inputs without the aggregation, as in Figure 14B. We explain S1's rationale for this in the later section. The streamer drew each part of the monster as viewers decided, and with rounds of this interaction, S1 ended up with a picture as in Figure 14C.

S2 - Annotating grammatical components on a sentence. S2 asked viewers to annotate grammatical concepts on a sentence shown on the slide. Figure 15 is one example, where S2 asks viewers to annotate clauses with boxes, while coloring them according to their roles in the sentence. Viewers could color the content clause with red, the adjective clause with blue, and the adverb clause with green. S2 aggregated viewer answers based on position, size, and the color of boxes. In the visualization, S2 specified the height of each glyph proportionate to the number of viewer inputs in the cluster. The streamer used aggregation flexibly, controlling the number of

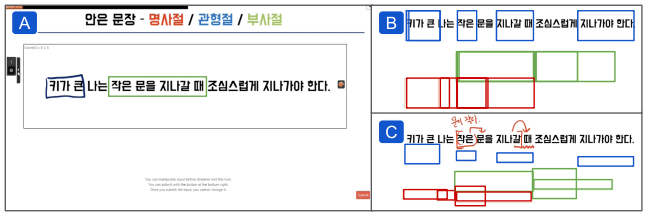


Figure 15: *S2's Korean lecture live streaming.* S2 collected and visualized people's annotation on clauses. Viewers annotated the roles of clauses in different colors. A) Collecting viewer inputs. B) Visualizing viewer inputs as raw inputs. C) Visualizing viewer inputs with the aggregation. The height of each box indicates the number of people in each cluster.

clusters based on their needs (Figure 15B, C). Based on visualized viewer inputs, S2 gave feedback to viewers.

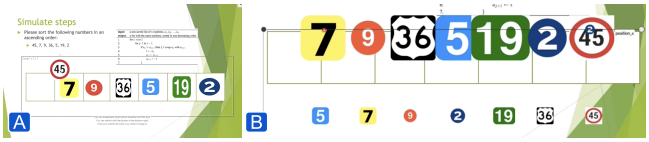


Figure 16: *The usage of VisPoll in S3's lecture on sorting algorithm.* A) S3 prompted viewers to simulate one iteration of a sorting algorithm with items given as visual objects. B) S3 visualized each group of answers on each row, with size indicating the number of viewers in the group.

S3 - Step-by-step execution of sorting. S3 prompted viewers to do one iteration of bubble sorting, by moving visual objects with different numbers (Figure 16A). After collecting inputs, the streamer visualized them by presenting each group of viewer answers in each row, with scaling size according to the count of viewer inputs in the group (Figure 16B).

VisPoll puts overhead on streamers, but worths the efforts. Streamers mentioned that they could bring up interactions that add further values to their live streaming. However, they also mentioned that the learning curve of the tool was a bit steep. For example, S1 mentioned that live streamers tend not to use more tools upon what they are already using, and making preparation simple would be crucial. Streamers suggested ways to lower the threshold for streamers, like the tool giving adaptive instructions during the usage or having more readily usable and contextualized presets.

7.3.2 Q2: VisPoll enables expressive and easy visual interactions that engage viewers and facilitate communication. Streamers thought that VisPoll enables engaging, expressive, but easy visual interactions for viewers. For example, S3 thought that VisPoll enables viewer interactions to be more expressive than multiple-choice questions, but with lower overhead compared to verbally asking viewers questions. Streamers also thought that VisPoll facilitated the communication with viewers with an additional communication mean. For example, S1 stated that it can alleviate the difficulty of referencing to a part of the video: *"To actually visually point out and say, 'Hey, this is, there's something a little off about this.' I think that's a whole different experience."*.

Visual interactions also promoted the viewer's engagement. For instance, S1V10 mentioned: *"It really makes the user feel like they're a part of the stream and that their voices are being heard... which you can often feel like it's not heard when there are a lot of users in chat."* In education settings, viewers tried to more actively consume content with VisPoll. S2V12 mentioned: *"I definitely put more thought into the live stream as I knew these activities were coming and I wanted to do the activities correctly."*

7.3.3 Q3: Aggregation and visualization helped streamers understand the overall viewer inputs, while engaging viewers more. The aggregation and visualization made streamers better understand how viewers answered and led them to have further interactions with viewers. For instance, S3 could identify *"different types of combinations of answers"* and asked viewers motivations behind answers. Streamers also explored viewer inputs by changing the aggregation setting. They tend to control the number of clusters to see different views of collected viewer inputs. S2's approach was one example (Figure 15 B, C): *"First, I saw the whole inputs to see outliers, ... Then, I decreased the number of clusters to see how the overall viewers think."* Furthermore, streamers used visualization strategically to maximize viewer engagement. For example, S1 maximized engagement of viewers by showing all viewer inputs collected: *"Everyone's like, 'Oh, I could see my sticker'."*

The aggregation and visualization also promoted viewer engagement, as streamers showed reactions to viewer answers. For example, S2V7 felt more engagement as the streamer could give feedback immediately with the help of VisPoll: *"I could better concentrate as I could submit my answer and get immediate feedback."* Furthermore, viewers felt more engagement with visualization, by seeing how others answered. For example, S2V4 mentioned: *"By looking at other people's answer and comparing them to mine, I could get the sense of engagement that I could have felt in a physical classroom."*

While the benefits of aggregation and visualization were significant, there were some uncertainties in using clustering as the aggregation algorithm. The streamer did not fully understand the exact mechanism of clustering, and sometimes the clustering could have done 'wrong' in streamers' notion, like grouping viewer inputs that should not be grouped together from their perspective.

8 LIMITATIONS AND FUTURE WORK

8.1 Expanding Input Visual Attributes

VIMF can be further extended towards allowing more visual attributes. First, when aggregating, VisPoll only considers visual attributes on a single visual object. However, the relationship between multiple objects can be important for some cases, and VIMF can be extended for these cases. For example, if a lecturer asks viewers to diagram a water molecule with two hydrogen and one oxygen atoms, the angle between two hydrogen atoms would be important information.

Second, VisPoll only allows symbols and text boxes as shape, but can be extended to receive more flexible visual inputs. One type is free sketches recognized by deep learning algorithms [15]. With rapidly advancing machine learning, this extension would be more

plausible. Parametric shapes [18] can also be used in our framework for enhanced expressiveness.

Third, VisPoll only allows static visual inputs. In the future, we would like to incorporate dynamic inputs. While there have been some tools supporting easy authoring of dynamic animations [25], it is worth exploring how to structure, aggregate, and visualize animations. Furthermore, the value of dynamic inputs would be further multiplied if it can consider the changing dynamics of the underlying video. It would possibly be instantiable with the help of tracking algorithms [53].

8.2 Expanding Visualizations

While VisPoll's visualization design centers around axis-based layouts with glyphs, it can be expanded to other types of visualizations. VIMF can employ other flexible and expressive visualization approaches used in works like Data Illustrator [34] or DataInk [55]. However, as more expressive visualizations may put more overload in authoring a visualization, it would be also important to prepare readily usable presets for visualizations.

8.3 Exploring Collected Viewer Inputs

The current version of VisPoll only supports aggregation on the whole viewer inputs, hence, is limited in exploring a more specific viewer group's answers. The future version of VisPoll can be extended to allow streamers to select with which viewer data they would do aggregation and visualization. It would allow streamers to understand viewer inputs more thoroughly. For instance, if a streamer is curious about how viewers in a specific cluster vary, they can choose viewer inputs in this cluster and do more specific clustering on them.

8.4 Multiple Rounds of Visual Input Collection

VisPoll can be expanded to visualize viewer inputs from multiple rounds of collections. It can be helpful for cases where visualizations can make synergy by overlapping them. For example, if a teacher prompts students to solve physics questions step-by-step, the teacher can visualize student answers for each step, leave the correct answer, and overlap correct answers from every step. Through this, the teacher can visualize a step-by-step solution to the problem. Likewise, this gradual accumulation of visualizations from different rounds of prompts also can be used for a collective visual summarization of lectures [33].

8.5 Deployment in Real-world

We could see the benefits of VIMF and VisPoll from the preliminary evaluation. We showed that they can be used in diverse streaming domains and demonstrated general strategies that can be used for viewer visual communication (e.g., eliciting multiple types of visual information from viewer inputs). In principle, the benefit would likely hold also for the large-scale situation with more viewers, but we did not run an observation with such a situation. Hence, we are planning to deploy our tool into real live-streaming settings with more viewers for an in-depth and long-term study. This would help us to gain further insights about how VisPoll's impact would change with a large-scale viewer group.

8.6 Required Efforts from Streamers

One limitation of our tool is that it requires significant efforts on streamers. To address the limitation, future work can improve the learning curve of the tool. For instance, the tool can show instructions automatically, considering the user's current context. As participants noted, the tool can also help users by having more presets they can readily use. Moreover, having a platform where streamers and viewers can share and co-create presets, similar to Github[4] or Scratch community[5], can be another future work direction. These directions would help users of diverse expertise easily use and customize visual interactions.

9 CONCLUSION

We present VIMF, which allows streamers to flexibly specify, aggregate, and visualize visual inputs from viewers. With VIMF, the streamer sense-make of visual inputs from viewers while preserving expressiveness and adaptability in viewer interactions. The core idea of VIMF is to structure visual inputs with visual attributes, which allows flexible management of inputs. Through the implementation and initial deployment of VisPoll, we found that VIMF enables new types of interactions between a streamer and viewers, which is challenging with existing techniques. While video is becoming a prevalent and ubiquitous communication medium, we hope this paper opens up new opportunities for expressive and scalable visual communications, and inspires HCI researchers to further explore multi-modal communication channels.

10 ACKNOWLEDGEMENT

We want to thank our participants for their time and feedback. We also want to thank Adobe Research for their support and Jane Im for valuable feedback on our system.

REFERENCES

[1] Rita Borgo, Johannes Kehrer, David H. S. Chung, Eamonn Maguire, Robert S. Laramee, Helwig Hauser, Matthew Ward, and Min Chen. 2013. Glyph-based Visualization: Foundations, Design Guidelines, Techniques and Applications. In *Eurographics 2013 - State of the Art Reports*, M. Sbert and L. Szirmay-Kalos (Eds.). The Eurographics Association, Switzerland, 39–63. https://doi.org/10.2312/conf/EG2013/stars/039-063

[2] Harald Bosch, Dennis Thom, Florian Heimerl, Edwin Püttmann, Steffen Koch, Robert Krüger, Michael Wörner, and Thomas Ertl. 2013. ScatterBlogs2: Real-Time Monitoring of Microblog Messages through User-Guided Filtering. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (Dec. 2013), 2022–2031. https://doi.org/10.1109/TVCG.2013.186

[3] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (Dec. 2011), 2301–2309. https://doi.org/10.1109/TVCG.2011.185

[4] N. Cao, Y. Lin, X. Sun, D. Lazer, S. Liu, and H. Qu. 2012. Whisper: Tracing the Spatiotemporal Process of Information Diffusion in Real Time. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2649–2658.

[5] Di (Laura) Chen, Dustin Freeman, and Ravin Balakrishnan. 2019. Integrating Multimedia Tools to Enrich Interactions in Live Streaming for Language Learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300668

[6] Yan Chen, Walter S. Lasecki, and Tao Dong. 2021. Towards Supporting Programming Education at Scale via Live Streaming. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW3, Article 259 (Jan. 2021), 19 pages. https://doi.org/10.1145/3434168

[7] Richard Cox. 1999. Representation construction, externalised cognition and individual differences. *Learning and Instruction* 9, 4 (1999), 343 – 363. https://doi.org/10.1016/S0959-4752(98)00051-6

[4] https://github.com/

[5] https://scratch.mit.edu/

[8] ExMachina. 2017. Smart Click Maps. https://dashboard.twitch.tv/extensions/c8okel68mmobvnso7ty0cygj8easam-0.1.5 Accessed: Sep, 2020.

[9] Travis Faas, Lynn Dombrowski, Alyson Young, and Andrew D. Miller. 2018. Watch Me Code: Programming Mentorship Communities on Twitch.Tv. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW, Article 50 (Nov. 2018), 18 pages. https://doi.org/10.1145/3274319

[10] Google Forms. 2020. Google Forms. https://www.google.com/forms/about/ Accessed: September, 2020.

[11] 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 (C&C '19)*. Association for Computing Machinery, New York, NY, USA, 144–155. https://doi.org/10.1145/3325480.3325485

[12] Elena L. Glassman, Juho Kim, Andrés Monroy-Hernández, and Meredith Ringel Morris. 2015. Mudslide: A Spatially Anchored Census of Student Confusion for Online Lecture Videos. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. Association for Computing Machinery, New York, NY, USA, 1555–1564. https://doi.org/10.1145/2702123.2702304

[13] Seth Glickman, Nathan McKenzie, Joseph Seering, Rachel Moeller, and Jessica Hammer. 2018. Design Challenges for Livestreamed Audience Participation Games. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '18)*. Association for Computing Machinery, New York, NY, USA, 187–199. https://doi.org/10.1145/3242671.3242708

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

[15] David Ha and Douglas Eck. 2017. A Neural Representation of Sketch Drawings. *CoRR* abs/1704.03477 (2017), 1–15. arXiv:1704.03477 http://arxiv.org/abs/1704.03477

[16] 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 SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1315–1324. https://doi.org/10.1145/2556288.2557048

[17] William A. Hamilton, John Tang, Gina Venolia, Kori Inkpen, Jakob Zillner, and Derek Huang. 2016. Rivulet: Exploring Participation in Live Events through Multi-Stream Experiences. In *Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video (TVX '16)*. Association for Computing Machinery, New York, NY, USA, 31–42. https://doi.org/10.1145/2932206.2932211

[18] Brian Hempel, Justin Lubin, and Ravi Chugh. 2019. Sketch-n-Sketch: Output-Directed Programming for SVG. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. Association for Computing Machinery, New York, NY, USA, 281–292. https://doi.org/10.1145/3332165.3347925

[19] iClicker. 2020. iClicker. https://www.iclicker.com/ Accessed: September, 2020.

[20] Ellen A. Isaacs, Trevor Morris, and Thomas K. Rodriguez. 1994. A Forum for Supporting Interactive Presentations to Distributed Audiences. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW '94)*. Association for Computing Machinery, New York, NY, USA, 405–416. https://doi.org/10.1145/192844.193060

[21] Hiroshi Ishii and Minoru Kobayashi. 1992. ClearBoard: A Seamless Medium for Shared Drawing and Conversation with Eye Contact. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*. Association for Computing Machinery, New York, NY, USA, 525–532. https://doi.org/10.1145/142750.142977

[22] Stephen C. Johnson. 1967. Hierarchical clustering schemes. *Psychometrika* 32, 3 (1967), 241–254.

[23] Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (1st ed.). Prentice Hall PTR, USA.

[24] Kahoot! 2020. Kahoot!. https://kahoot.com/ Accessed: September, 2020.

[25] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, and George Fitzmaurice. 2014. Kitty: Sketching Dynamic and Interactive Illustrations. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. Association for Computing Machinery, New York, NY, USA, 395–405. https://doi.org/10.1145/2642918.2647375

[26] Nam Wook Kim, Hyejin Im, Nathalie Henry Riche, Alicia Wang, Krzysztof Gajos, and Hanspeter Pfister. 2019. DataSelfie: Empowering People to Design Personalized Visuals to Represent Their Data. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300309

[27] Nam Wook Kim, Eston Schweickart, Zhicheng Liu, Mira Dontcheva, Wilmot Li, Jovan Popovic, and Hanspeter Pfister. 2017. Data-Driven Guides: Supporting Expressive Design for Information Graphics. *IEEE Trans. Vis. Comput. Graph.* 23, 1 (2017), 491–500. https://doi.org/10.1109/TVCG.2016.2598620

[28] Yea-Seul Kim, Katharina Reinecke, and Jessica Hullman. 2017. Explaining the Gap: Visualizing One's Predictions Improves Recall and Comprehension of Data. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, New York, NY, USA, 1375–1386. https://doi.org/10.1145/3025453.3025592

[29] Pascal Lessel, Alexander Vielhauer, and Antonio Krüger. 2017. CrowdChess: A System to Investigate Shared Game Control in Live-Streams. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '17)*. Association for Computing Machinery, New York, NY, USA, 389–400. https://doi.org/10.1145/3116595.3116597

[30] Pascal Lessel, Alexander Vielhauer, and Antonio Krüger. 2017. Expanding Video Game Live-Streams with Enhanced Communication Channels: A Case Study. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, New York, NY, USA, 1571–1576. https://doi.org/10.1145/3025453.3025708

[31] Cathy Li and Farah Lalani. 2020. The COVID-19 pandemic has changed education forever. This is how. https://www.weforum.org/agenda/2020/04/coronavirus-education-global-covid19-online-digital-learning/ Accessed: September, 2020.

[32] Jie Li, Xinning Gui, Yubo Kou, and Yukun Li. 2019. Live Streaming as Co-Performance: Dynamics between Center and Periphery in Theatrical Engagement. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 64 (Nov. 2019), 22 pages. https://doi.org/10.1145/3359166

[33] Ching Liu, Juho Kim, and Hao-Chuan Wang. 2018. ConceptScape: Collaborative Concept Mapping for Video Learning. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173961

[34] Zhicheng Liu, John Thompson, Alan Wilson, Mira Dontcheva, James Delorey, Sam Grigg, Bernard Kerr, and John Stasko. 2018. Data Illustrator: Augmenting Vector Design Tools with Lazy Data Binding for Expressive Visualization Authoring. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173697

[35] 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. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW, Article 112 (Nov. 2018), 26 pages. https://doi.org/10.1145/3274381

[36] 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 (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3174040

[37] Adam Marcus, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Robert C. Miller. 2011. Twitinfo: Aggregating and Visualizing Microblogs for Event Exploration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. Association for Computing Machinery, New York, NY, USA, 227–236. https://doi.org/10.1145/1978942.1978975

[38] T. Munzner. 2015. *Visualization Analysis and Design*. CRC Press, USA. https://books.google.de/books?id=NfkYCwAAQBAJ

[39] Qualtrics. 2020. Qualtrics. https://www.qualtrics.com/ Accessed: September, 2020.

[40] Quizizz. 2020. Quizizz. https://quizizz.com/ Accessed: September, 2020.

[41] Jeffrey Rzeszotarski and Aniket Kittur. 2012. CrowdScape: Interactively Visualizing User Behavior and Output. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. Association for Computing Machinery, New York, NY, USA, 55–62. https://doi.org/10.1145/2380116.2380125

[42] Arvind Satyanarayan and Jeffrey Heer. 2014. Lyra: An Interactive Visualization Design Environment. In *Proceedings of the 16th Eurographics Conference on Visualization (EuroVis '14)*. Eurographics Association, Goslar, DEU, 351–360.

[43] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer. 2016. Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2016), 659–668.

[44] Joseph Seering, Saiph Savage, Michael Eagle, Joshua Churchin, Rachel Moeller, Jeffrey P. Bigham, and Jessica Hammer. 2017. Audience Participation Games: Blurring the Line Between Player and Spectator. In *Proceedings of the 2017 Conference on Designing Interactive Systems (DIS '17)*. Association for Computing Machinery, New York, NY, USA, 429–440. https://doi.org/10.1145/3064663.3064732

[45] Elsbeth Stern, Carmela Aprea, and Hermann G. Ebner. 2003. Improving cross-content transfer in text processing by means of active graphical representation. *Learning and Instruction* 13, 2 (2003), 191 – 203. https://doi.org/10.1016/S0959-4752(02)00020-8 External and Internal Representations in Multimedia Learning.

[46] Streamlabs. 2020. Media Share+Media Request Overlay: Viewers Tip to Share Videos & Songs. You Play Them On-Stream. https://streamlabs.com/obs-widgets/media-share Accessed: September, 2020.

[47] SurveyMonkey. 2020. SurveyMonkey. https://www.surveymonkey.com/ Accessed: September, 2020.

[48] Ryo Suzuki, Rubaiat Habib Kazi, Li-Yi Wei, Stephen DiVerdi, Wilmot Li, and Daniel Leithinger. 2020. RealitySketch: Embedding Responsive Graphics and

Visualizations in AR through Dynamic Sketching. arXiv:cs.HC/2008.08688

[49] Tableau Software. 2020. Tableau. https://www.tableau.com/products/desktop Accessed: September, 2020.

[50] Twitch. 2020. How to Use Polls. https://help.twitch.tv/s/article/how-to-use-polls?language=es&sf222407025=1 Accessed: September, 2020.

[51] Hadley Wickham. 2016. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, USA. https://ggplot2.tidyverse.org

[52] W. Willett, Y. Jansen, and P. Dragicevic. 2017. Embedded Data Representations. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 461–470.

[53] Y. Wu, J. Lim, and M. Yang. 2013. Online Object Tracking: A Benchmark. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, USA, 2411–2418. https://doi.org/10.1109/CVPR.2013.312

[54] Haijun Xia, Bruno Araujo, and Daniel Wigdor. 2017. Collection Objects: Enabling Fluid Formation and Manipulation of Aggregate Selections. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, New York, NY, USA, 5592–5604. https://doi.org/10.1145/3025453.3025554

[55] Haijun Xia, Nathalie Henry Riche, Fanny Chevalier, Bruno De Araujo, and Daniel Wigdor. 2018. DataInk: Direct and Creative Data-Oriented Drawing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173797

[56] Saelyne Yang, Changyoon Lee, Hijung Valentina Shin, and Juho Kim. 2020. Snapstream: Snapshot-Based Interaction in Live Streaming for Visual Art. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376390

[57] Jiayi Eris Zhang, Nicole Sultanum, Anastasia Bezerianos, and Fanny Chevalier. 2020. DataQuilt: Extracting Visual Elements from Images to Craft Pictorial Visualizations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376172