# Generalized Playback Bar for Interactive Branched Video

Eric Lindskog Jesper Wrang Madeleine Bäckström Linn Hallonqvist Niklas Carlsson
Linköping University, Sweden

## ABSTRACT

During viewing of interactive *branched video*, users are asked to make viewing choices that impact the storyline of the video playback. This type of video puts the users in control of their viewing experiences and provides content creators with great flexibility how to personalize the viewing experience of individual viewers. However, in contrast to with traditional video, where the use of a playback bar is default for most – if not all – players, there currently does not exist any generic playback bar for branched video that helps visualize the upcoming branch choices. Instead, most branched video implementations are typically custom-made on a per-video basis (e.g., see custom-made Netflix and BBC movies) and do not use a playback bar. As an important step towards addressing this void, we present the first branched video player with a generalized playback bar that visualizes the tree-like video structure and the buffer levels of the different branches. The player is implemented in dash.js and is made public with this publication, is the first of its kind, and allows both the playback bar and the presentation of branch choices to be customized with regards to visual appearance, functionality, and the content itself. Furthermore, the design is generic (making it applicable to any video) and allows content creators to easily create large numbers of branched movies using a simple metafile format. Finally, and most importantly, we perform a three-phase user study in which we evaluate the playback bar, compare with alternative designs, and other branch-related features. The user study highlights the value of a branched video playback bar, and provides interesting insights into how it and other design customization features may best be integrated into a player.

## CCS CONCEPTS

• **Information systems → Multimedia streaming**; • **Human-centered computing → Interactive systems and tools**.

## KEYWORDS

Branched video; non-linear video; interactive film; progress bar; user interface; dash.js implementation; user study

**ACM Reference Format:**
Eric Lindskog Jesper Wrang Madeleine Bäckström Linn Hallonqvist Niklas Carlsson. 2019. Generalized Playback Bar for Interactive Branched Video. In *Proceedings of the 27th ACM International Conference on Multimedia (MM '19), October 21–25, 2019, Nice, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3343031.3350951

## 1 INTRODUCTION

Interactive storytelling using *branched video* streaming [18] (sometimes also called *non-linear*, *hypervideo*, and *multi-path* video streaming [5, 17, 25–28, 30, 38]) allows users to make viewing choices that impact the storyline (or plot sequences) while watching a video. This puts viewers in control of their own viewing experience.

At an abstract level, the potential plot sequences that a user can select from while viewing these videos can typically be captured by a tree structure. Here, each vertex corresponds to a *branch point*, at which the user is asked to make a path choice, and the edges represent video *segments* that are played out between these branch points (or the start and end of the video playback sequence).

More concretely, during a typical playback session, the user is normally presented path choice at-or-before each branch point, and the playback path is adapted based on the user's branch selections. This results in a customized plot sequence based on the user's path choices. Further personalization of the plot sequences can easily be achieved by presenting different users with different path choices.

The above examples illustrate some of the flexibility and power that branched videos streaming provides. However, despite attempts by production houses such as Netflix and BBC, for example, with the exception of the recent success of Netflix's Black Mirror episode "Bandersnatch" (released Dec. 2018) [37], branched video has yet to become mainstream. We believe that one reason for this is that most of these videos use per-video custom-made user interfaces that gives each video a unique feel. Instead, we argue that it is important that users are presented with a generalized interface that easily can be reused for many videos and that the interface provides a clear visual way to extract information about upcoming branch choices, playback progress and buffer levels. In the context of traditional "linear" video, these are important aspects that we all expect modern video players to provide. Yet, as of today, there currently does not exist any generic playback bar for branched video that helps visualize these aspects.

In this paper, we present (i) the design and implementation of a novel branched video player that includes a generalized playback bar, and (ii) the results and insights from a three-step user study in which we evaluate the use of such a playback bar, compare alternative designs, and evaluate the integration of the playback bar and other branch-related features.

The player is implemented in the dash.js (open source) player and is made public (at www.ida.liu.se/~nikca89/papers/mm19.html) with this publication. It is the first of its kind and includes a generalized tree-based progress bar that provides information about upcoming branch choices, the playback progress, and the current buffer levels. Our solution provides an overview of the path choices to be made, simplifies navigation within the set of candidate storylines, and (as regular players) aid users in understanding when a stall is due to the current playback buffer being empty (e.g., due to bad network connectivity) so as to avoid users unjustifiably blaming the player for such instances. The design is made generic enough

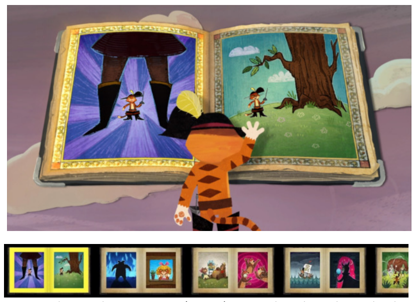
**Figure 1: Branch selection (top) and playback bar (bottom), when watching *Puss in boots: Trapped in an Epic Tale* [29], by Netflix. Here, pages in a storybook represent choices, and the animated main character helps make the choice.**

that it can be used for any branched video structure (assuming the underlying linear video(s) can be played by dash.js) and enables content creators to easily create large numbers of branched movies using a simple metafile format. Furthermore, the player is designed to allow easy customization of both the playback bar and the presentation of branch choices. This includes customization with regards to both visual appearance, functionality, and the content itself.

Our user study was performed using 32 participants and included three parts; each designed to answer different subsets of research questions. The first part used a combination of methods to measure potential user-perceived efforts and selection delays associated with using a playback bar. In the second part we evaluate the perceived value added by such a playback bar and other user-perceived tradeoffs. Finally, in the third part, a series of test feature experiments and discussion questions are used to investigate the best ways to integrate the playback bar and various branch-related example features into the player interface. Overall, the user study highlights the value of a branched video playback bar (e.g., for building a better understanding of the player state, upcoming branches, and various other branch related aspects) and shows that these benefits comes at a negligible increase in the users perceived effort.

**Outline:** Section 2 provides background. Sections 3 and 4 present system design and playback bar features, respectively. User study and results are presented in Section 5. Section 6 concludes the paper.

## 2 BACKGROUND AND RELATED WORK

**Commercial examples:** Use of branched video for interactive plot sequences in which users can select their own path through a movie has primarily been limited to custom-made solutions or innovative use of special features intended for other purposes. Examples of the first type include custom-made movies by big production houses (e.g., Netflix, BBC) and videos produced using Eko [8]. While these videos typically do not have a branched playback bar, some videos do allow users to keep track of their choices. For example, Netflix interactive video *Puss in Boots: Trapped in an Epic Tale*, frequently prompts viewers to choose between two alternatives pages in a book and overviews the users' past choices (Figure 1). While intuitive, this interface is movie specific and may not generalize well to other movies or scenarios. As an example of the second type we mention the use of annotations in YouTube to link viewers to alternative videos that can be viewed next to create a type of multi-path experience in which a new movie is loaded for each segment. With the YouTube approach, a new video needs to be loaded each time you make a choice, and there is no built in prefetching of upcoming branches. Most importantly, none of these services provide a playback bar that captures the video structure and how much video content have been buffered for each path.

**Related work:** HyperCafe [30] was one of the first systems to demonstrate how users can interactively switch between multiple videos. This was achieved using textual information and hyperlink objects. Since then, various types of annotations and links have been used to create non-linear video [25, 27], generalize the multi-video concepts to 360° videos [28], build novel navigation maps using geo-referenced 360° videos, and integrate videos and game engines [35], for example. These and other similar systems are often designed to illustrate some annotation and editing principles for linking videos; however, usability is seldom taken into account [22]. Other earlier work focused on multicast-based optimizations for popular content [5, 38], or the design of authoring and media representation tools for the development of interactive media [24, 26, 31].

Other research on branched (or non-linear) video streaming have focused on other technical challenges. For example, Krishnamoorthi et al. [17, 18] explored how the rate adaptive features of HAS can be used to optimize the prefetching of upcoming branches so to maximize the expected quality of experience. While it would be possible to extend their (open source) player [18], their player was implemented using Adobe's OSMF framework, making it increasingly less useful by modern browsers, as OSMF sees decreasing support and use. We therefore selected to implement our own branched video player within the dash.js framework. We also note that their player used a very simple user interfaces in which numbered branch choices are listed in a single textbox and the user simply selects a branch by pressing the corresponding number on the keyboard. Others have considered prefetching and cache management problems for hypervideos (similar to the YouTube example above) [21, 23]. Yet others have designed tag-based systems that allow tagged parts of different videos to be automatically stitched together into a single playback sequence [16], or provided prefetching frameworks for related contexts where users may quickly switch between multiple videos [4, 19, 33, 36]. Finally, we note that it has been shown that annotating the timeline of the playback bar (of linear players) with buffer state and adding bookmarks for such prefetched segments can improve users' playback experiences.

## 3 SYSTEM DESIGN

For the best possible playback experience, we believe that it is important that the solution is both *rate adaptive* and *interactive*. In this work, we therefore extend a *rate adaptive* open source player (dash.js) to allow playback of *interactive* branched video streaming.

**Adaptive streaming:** Almost all video streaming over the Internet is done using various HTTP-based Adaptive Streaming (HAS) systems [3, 14, 15, 32]. These techniques typically break the video into shorter *chunks* (e.g., 2-10 seconds long), each encoded into multiple video qualities, and then let the player adapt the requested video quality of each chunk so as to maximize the expected Quality of Experience (QoE) given current network conditions and buffer levels. While QoE is a subjective measure, it is well understood that playback stalls and the encoding quality used for each played chunk

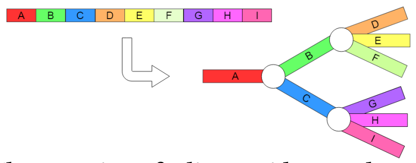

**Figure 2: The mapping of a linear video to a branched video**

significantly affect QoE [1, 2, 7, 13, 14, 20]. For this reason, most of these techniques try to maintain sufficient buffer conditions so to avoid stalls, while simultaneously maximizing the video quality and keeping the number of quality switches to a minimum.

**Dash.js:** For our implementation, we extend the *dash.js* player [6]. *dash.js* is an open source project driven by the DASH Industry Forum based on the Dynamic Adaptive Streaming over HTTP (DASH) standard [32]. The player is implemented in JavaScript and is built inside a webpage that can be viewed in the major browsers. It works by binding an HTML video element to the DASH implementation and requires a Media Presentation Description (MPD) file to be loaded. This metafile contains information such as the chunk lengths and encodings used, the different resolutions and bitrates available, and other information needed to determine what content can be requested, displayed, and where to request it from.

**Extending dash.js for branched video:** dash.js does not support branched videos by default and there currently is no universal branched video format. For these reasons, we extended the source code of dash.js and implemented our own methods and formatting to allow playback of branched video.

First, using the format suggested by Krishnamoorthi et al. [17, 18], we assume that all video *segments* (each corresponding to the playback sequence between two *branch points* in the branched video; i.e., a branch) have been concatenated into a single linear video (see Figure 2), and then introduce an extra metafile that specifies (i) a unique identifier for each segment, (ii) the start and end of each segment (as defined by playpoints in the underlying linear video), (iii) the branch options that follows each segment (if one or more option exist), and (iv) a descriptive name of the segment (that can be used when the segment is a branch option). The metafile uses JSON format and can easily be customized for each user or user category. However, in all cases it must contain a list of segment objects, where each such object has five fields: id, startTime, length, branches, and displayName. Note that this format easily allows segments that overlap in the original (linear) video to be defined as unique segments, and the unique identifiers provide us with fast (constant time) lookup and iterative execution.

During playback, the player keeps track of the current playpoint and uses the information in the metafile to determine the next branch point, present branch options for that branch point, and prefetch data from the start of each upcoming candidate segment, so to allow seamless playback of the next segment to play. Ideally, to avoid playback stalls at a branch point, at least the first chunk of each candidate segment of an upcoming branch point should be prefetched ahead of that branch point being reached [18]. However, since dash.js does not have a built in way to request out-of-order chunks during playback, we had to implement an extension that intercepted all requests sent by dash.js, and that made additional requests when approaching a branch point. For prefetching, our solution starts by downloading the first segment of each segment (in round-robin fashion) and then continue prefetching along the default path, but note that further optimizations are possible that weight the different candidate paths and adjust playback quality of each candidate segment so to optimize the expected quality [18].

Finally, when reaching a branch point, the playpoint jumps to the startpoint of the segment corresponding to the branch selected by the user. In the case no branch have been selected, there are two options: (i) pick the default path, or (ii) stall playback until a choice is made. While we have implemented both options and users appears to prefer (i), as explained later, for the purpose of the experiments presented here we used option (ii).

**A novel customizable playback bar:** We have implemented a customizable playback bar that allows us to easily test and compare different candidate features (described in Section 4).

The playback bar was written in JavaScript using a canvas element for drawing the graphics. The canvas has support for different transforms and allows us to render both simple and complex 2D shapes and bitmaps inside of the browser. For our example implementations, the bar can either be placed directly under the video or overlayed (as a see-through object) at the bottom of the video. In the second case, the bar can be shown at all times or conditioned on the mouse placement, the current playpoint, or some other criteria. We then created two classes of visual objects: segments and branch points. To allow flexibility in the design of the playback bar, we allow the placement, size, and shape of these objects to be both manipulated and transformed during playback. In general, we took some inspiration of existing popular linear players (e.g., *YouTube* and *Netflix*), and used a dark grey color to indicate content that have not yet have been buffered, light grey to indicate content that have been buffered but not yet played, and red to indicate that the content have been played. However, rather than a single playback bar for the entire video, we use one (sub)bar per segment.

## 4 PLAYBACK BAR FEATURES

This section outlines some of the design alternatives we have considered and their implementation.

### 4.1 How much of the tree to show?

One question that arises when designing a progress bar for branched video is how much of the tree structure that should be displayed. With traditional linear video, progress is almost always displayed relative to the full playback duration; start to finish. This provides a quick overview of the current status and allows users to quickly skip to different parts of the video (e.g., by clicking on the playback bar or using fast forward/rewind). However, with branched video, the potential graph structure can be large. We must therefore consider the tradeoff between providing an overview of all playback paths and focusing on the part currently most relevant to the user. Here, we compare three alternative features addressing this tradeoff.

- *Full structure:* Motivated by the success of traditional playback bars, in default mode we show the full structure.
- *Simple zoom-and-follow:* This feature only displays the segment currently being played, the most recently played segment, and all upcoming branches up-to a depth $D$.
- *Prune non-selected paths:* This feature removes all branches that are no longer reachable from the current playpoint.

Ideally, the progress bar should not take up too much of the potential viewing area and should not interfere with the viewing of the video. For structures with many branch points, the simple zoom-and-follow features therefore provides an advantage, as it limits the part of the tree structure that needs to be shown.

Focusing more on the current and nest-coming segments can also help avoid that users look too far ahead or reveals too much information about differences in how far away different story endings may come. Clearly, a zoomed in version can allow the content creator to hide such hints. On the other hand, its narrower time focus may cause users to loose part of the benefits that comes from providing an overview of the graph structure.

## 4.2 Positioning, visibility, and timing

Other important aspects are the positioning of the playback bar, how it is integrated, and when it is best displayed. In early implementations, we placed the playback bar beneath the video. However, feedback in initial user studies (using SUS and thinking aloud with 11 participants on an early prototype, to help ensure that we studied the right questions) suggested that it should be integrated with the rest of the video player. Here, we therefore focus on implementations in which a semi-transparent playback bar is placed inside the lower part of the video element. The playback bar can also be hidden between branch points, and are only shown whenever there is a branch point within the next $T$ seconds. We expect that this parameter $T$ is best set such that the playback bar appears when the user is prompted to make a path decision. However, for the user study presented here, we considered the two extreme points when the playback bar always is shown (corresponds to $T \rightarrow \infty$) and when it never is shown (corresponds to $T=0$).

Another important timing-related aspect is whether to pause (stall) playback of the video when reaching a branch point for which the user has not yet selected a path. While it is relatively easy to define a default path (e.g., the first path), for the purpose of the user study, we chose to pause playback at these instances, providing users with more time to familiarize themselves with the playback bar (and allowing us to better measure users' decision times).

## 4.3 Visual appearance of branches

**Generalized shape of branches:** We (i) generalized the presentation of segments to be based on an arbitrary function that connects two (branch) points, and (ii) used functions that result in a quick visual separation of the alternative branches. Examples of functions that met our criteria include the arctangent function, the square root function, the error function, the hyperbolic tangent function, and a shifted logarithmic function that passes through the origin. Assuming such function $f(x)$, for our implementation, we first determine the desired start point $(x_i^0, y_i^0)$ and end point $(x_i^1, y_i^1)$ for each segment $i$, and then scale the y-values to go through these two points: $y_i^{scaled}(x) = y_i^0 + (y_i^1 - y_i^0)\frac{f(x-x_i^0)}{f(x_i^1-x_i^0)}$.

**Focus-based visual distortion:** We also considered transformations of the 2D space in such a way that additional visual focus is put on the region of most current interest. For this purpose, we implemented and tested a basic *fish-eye* effect (magnifying the area around the current playpoint) and a basic *mouse-eye* effect (magnifying the area around where the user has the mouse cursor); both designed to provide visual distortion for wide panoramic images. While fish-eye effects can be implemented in a few different ways [11], the most common way (called hemispherical) creates a spherical view of the image where the center of the image appears closer and larger than the surrounding objects, and corresponds to using an ultra-wide angle lens on a camera. The choice to try the use of these effects was motivated by past work that has showed that they can make it easier to interact with larger interfaces and things that require wide field-of-view (in our case a potentially larger structure) [12].

## 4.4 Integration and branch choice labeling

Figure 3 shows a basic (default) player implementation with large, clickable, transparent buttons that are overlayed on-top of the video itself when close to a branch point. For the best possible integration, we have found (as part of the study presented here) that many users appreciate additional augmenting features that clearly demonstrate the connection between branch selection buttons and the playback bar. We next summarize the main features that at least some users found helpful clarifying this connection and/or that further helped integrate the playback bar into the branch decision process.

- *Matching branch labeling:* This feature adds a "branch choice" letter (e.g., "A", "B", "C") to each of the segments following the upcoming branch point as shown in the playback bar, as well as to the corresponding path selection buttons.
- *Highlight path when hovering over button:* This feature visually connects a button to a branch in the playback bar by highlighting the branch in a bright blue color whenever the viewer hovers over the button associated with that branch. This is implemented by changing the background color of the highlighted branch, and allows the viewer to easily visualize the different paths that the alternative branches enable.
- *Clickable playback bar:* This feature allows the user to select a playback path by simply clicking on the segment within the playback bar corresponding to its desired path choice. With this feature, we always highlight the path associated with the branch over which the user hovers, but allow it to be run either with or without the default playback buttons.
- *Explicitly place buttons in tree:* With this feature, the choice buttons in the player are placed directly over the appropriate branches in the tree. (In this case we use smaller buttons so as to line them up with the branch segments.)

All these features are optional and do not need to be used simultaneously. The best selection of features will differ from designer to designer and from user to user.

## 4.5 Example implementations

Using the above functionalities, we have implemented and tested many different candidate combinations. Figure 3 shows the default player used in our user evaluation. Here, both the branch choices and a transparent playback bar are placed on top of the video, branches are using an arc-tangent function, and there is no other scaling or transformation made to the playback bar. Furthermore, none of the branch choice labeling features from Section 4.4 were used in the default player. As our baseline comparison (Sections 5.2-5.4) we use an identical player without a playback bar. We also

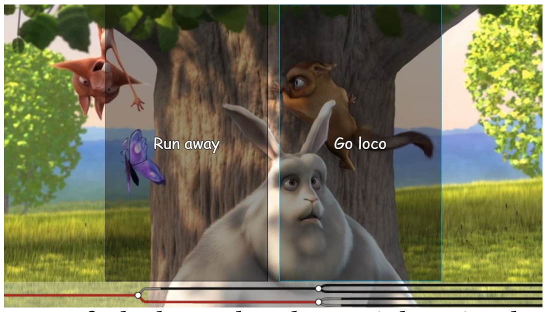

**Figure 3: Default player when the user is hovering the right choice at the second branch point.**

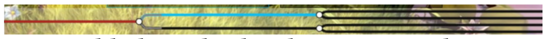

**Figure 4: Highlight path when hovering over button; in this case, the left button.**

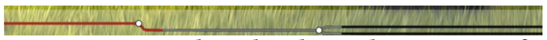

**Figure 5: Prune non-selected paths; in this case, just after the first story choice has been made.**

include example figures for two of the most popular features in our per-feature analysis: *highlight path when hovering over button* (Figure 4) and *prune non-selected paths* (Figure 5). These features are described in Sections 4.4 and 4.1, and evaluated in Section 5.5.

## 5 USER STUDY

### 5.1 High-level overview

The user study consisted of three parts, each part focusing on a separate aspect. In the first part, we use a combination of standard questions (i.e., NASA-TLX and SEQ; see Section 5.2) and measurements of the users' actual response times when presented a branch choice to evaluate how the playback bar itself may add most/least to the complexity and perceived effort of the users. For this part, we consider users' interactions with the default interfaces (Section 4.5) *with* and *without* playback bar. To ensure that we captured how the interfaces were perceived by first-time users (e.g., with regards to ease of use, intuitiveness, and complexity), we did not include any training phase, but instead introduced the interfaces using minimal instructions. The second part uses head-to-head comparison questions of the two player implementations to identify potential tradeoffs associated with the playback bar (Sections 5.3 and 5.4). Before asking these questions, we made sure that the users had understood the main difference between the interfaces evaluated in the first part of the study. Finally, in the third part, we evaluate and report the users' opinions about most of the key features one-by-one (Section 5.5). At this point of the study, the users had gained a better appreciation for the problems that a playback bar may be able to address and were able to provide valuable feedback on different feature alternatives.

**System setup:** For our experiments, we hand-crafted and used different branched video storyline examples. All storylines used *Big Buck Bunny* [9] as the underlying linear video but differed by the segments and branch points defined in their respective metafiles. Here, we used an encoding for a screen size of 1920×1080, with 30 frames per second, and encoded with H.264 (MPEG-4 AVC).

The program *x264* [34] was used to encode the video into four qualities: 1080p, 900p, 720p and 480p. We used *MP4Box* [10] to encapsulate the raw H.264 files in separate MP4 containers and split the files into segments. We gave each quality a unique id and used 4 second segments for each quality. The program automatically generates an MPD file that can be used directly in dash.js.

**Equipment and screen setup:** The user study was performed on a laptop (Asus Zenbook UX430) with a 14-inch, non-touch, 1920×1080 screen, Intel processor (i7-8550U), running Windows 10. For each individual playback session, we setup the player to use the full screen and placed the cursor within the main viewing area. This ensured that users of the default implementation with the playback bar always were exposed to the playback bar.

**Participants:** Using emails and in-person contacts, we recruited 32 participants (10 female, 22 male), ages 20-30 ($\mu$ = 22.7, $\sigma$ = 1.9), at a local university. The participants included a mix of majors, had varied, but limited, prior exposure to branched video (≈ half had no prior experience; rest had seen 1-5 videos, and often mentioned Bandersnatch as their first such video), and all gave written consent to participate in the study. The study took approximately 30-45 minutes and participants received a sandwich and a non-alcoholic drink (e.g., coffee or soda). Our participant sample provides a relatively well-defined group (university students in the typical age-range of 20-30 years old). We acknowledge that this may bias the results. We further note that some users explicitly compared with their Bandersnatch experience, influencing their expectations.

**Potential influence during the study:** To avoid influencing the participants' answers, we made sure to not ask any leading questions. In some cases, a participant struggled to understand or observe some of the features presented to him/her. In these cases, instead of directly pointing out and explaining the feature, we showed the participants the video again to give him/her another chance to notice the feature. The few times that the participant still did not notice the feature, we provided hints on where on the screen to focus, after which everyone noticed the feature. In the cases that a participant did not understand a question (provided in writing) and asked us about any question, we tried to clarify the question without influencing the participant's answer. For example, a few participants said that they had not noticed any information about buffering when viewing the video and therefore did not understand our question regarding how much the playback bar help understand this aspect. Here, we simply encouraged them to answer the questions according to their own experience and note that these users could select the option "don't know".

### 5.2 Perceived effort and selection delays

Ideally, the benefits of a playback bar should come at low cognitive overhead. To understand the impact on the users' perceived effort, we asked the participants to watch two different branched videos ("A" and "B") using the default player implementations *with* and *without* a playback bar. Both videos had three binary branch points; only the storylines differed. During the viewing of the videos we recorded the time to make each branch decision (from the time branch options were presented) and any stall times due to users failing to select a branch option until after the branch point is reached (here, causing the video to stall until a decision is made), and after each playback session we asked the user to fill out NASA-TLX and Single Ease Question (SEQ).

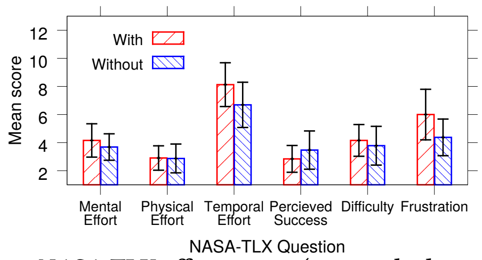

**Figure 6: NASA-TLX effort scores (1-20 scale; low-to-high).**

**Table 1: Summary of SEQ (scale 1-7; low-to-high).**

| Implementation | Mean ($\mu \pm \sigma$) | 95% interval |
|---|---|---|
| With PBB | 6.63±0.83 | (6.33,6.92) |
| Without PBB | 6.56±0.76 | (6.29,6.83) |

To account for differences in the order that the implementations were introduced and what video were used together with each implementation, we ran equally many experiments in which each player implementation was run first, and among these experiments, video "A" was used first half of the times (and "B" the other half). Furthermore, to reduce any time-based biases, each of the four possible player-video combinations were tested first according to a round-robin schedule. This ensures that we had a balanced dataset after every fourth participant. With 32 participants, our final dataset included eight complete rounds.

Figure 6 summarizes the average effort scores (1-20 scale; low-to-high) obtained with NASA-TLX, with two-sided 95% confidence intervals. While the average scores may suggest a slight increase for all metrics except the perceived success, the 95% confidence intervals are overlapping in all cases. None of these perceived differences are therefore significant. These results are encouraging as they suggest that any additional effort potentially introduced by the playback bar is minimal.

A similar analysis was performed on the SEQ. Table 1 summarize these results. Here, we note that the SEQ range is 1-7 (with 7 being the most positive answer possible). We again note small differences that are non-significant for the confidence levels considered here. It is also encouraging to see that the two implementations have average scores of 6.63 and 6.56, respectively, giving a slight edge to the implementation with the playback bar, and both implementations have the highest possible median score (7 out of 7).

To better understand whether the playback bar adds extra delay to the decision process, Figure 7 shows the CDFs of the branch selection times when using the player with and without the playback bar. Here, we also included a "stall region" line indicating the average measured time (4.75±0.09 seconds) at which playback was frozen ("stalled") when reaching the branch point before a decision was made. (Branch options presented approximately 5 seconds before the branch point.) In general, we observed somewhat larger decision times when using the playback bar. This may indicate that some users spent some extra time trying to understand how to use the playback bar. We also saw significantly higher delays for the first branch (average of 5.62 seconds) compared to for the second branch (3.90 seconds) and for the first played video (average of 5.22 seconds) compared for the second played video (4.31 seconds). These results suggest that it sometimes take some time to get used to the branched video concept.

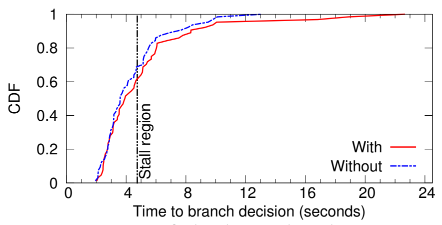

**Figure 7: CDFs of the branch selection times.**

**Table 2: Increase in understanding of different aspects provided by the playback bar. (scale 1-20; low-to-high).**

| Aspect of consideration | Score ($\mu \pm \sigma$) |
|---|---|
| The branched video concept | 13.6 ± 5.2 |
| The video structure and choices at hand | 14.0 ± 4.7 |
| Whether there will be upcoming branches | 18.3 ± 2.7 |
| Remaining playback time | 16.8 ± 4.3 |
| Amount data buffered | 13.3 ± 6.9 |

## 5.3 Perceived value added by the playback bar

After having completed the first part of the study, we made sure that the users were aware of the difference between the two example implementations that they had just used to view branched video (i.e., the playback bar). We then asked explicit questions regarding to what degree they felt that the playback bar helped increase their understanding for different aspects related to branched videos.

Table 2 shows the average scores of the understanding added by the playback bar due to different aspects. For consistency, we used the same 20-point scale (without visible numbers) and low-to-high score labeling as NASA-TLX. In addition to the listed aspects, we believe that a playback bar helps clients understand when a stall is due to low buffer conditions, but did not include any questions regard this aspect, as such stalls did not occur with our setup. This may also help explain why nine participants answered "don't know" regarding the improvement of their understanding of the "amount data buffered". These participants were not included in the calculation of the average scores. One participant also answered "don't know" regarding whether the playback bar helped increase the understanding of the "branched video concept".

It is encouraging to see that the average scores were consistently above 13 (out of 20) for all five considered aspects, with the highest scores being reported for whether there will be upcoming branches (score = 18.3) and how much playback time remains (score = 16.8). This shows that the playback bar helped the participants gain some understanding of the branch videos that otherwise may not be readily available to them, even from their relatively brief playback sessions. Of course, it should be noted that the two highest scoring aspects also come with a downside, as such an understanding may reveal to the user that the ending is near, for example, and may therefore impact their path choices. Hiding the playback bar after branch choices and using some variation of the basic zoom-and-follow feature (not yet introduced to the users at this point in the study) may therefore provide a better balance than the example implementation evaluated first.

## 5.4 Other user-perceived tradeoffs

At this point of the study we also asked the participants to evaluate the two example implementations regarding (i) the speed of picking

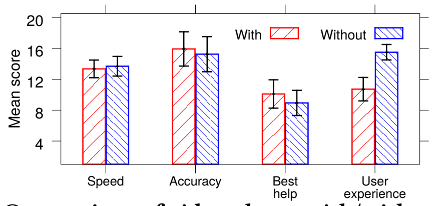

**Figure 8: Comparison of video player with/without playback bar (1-20 scale; low-to-high).**

a desired path, (ii) the accuracy of picking a desired path, (iii) the help provided to make a good decision, and (iv) the perceived user experience. Again, we remind the reader that the two example implementations were not designed to optimize any of the above criteria, but rather to make users aware of the playback bar and allow a more well-informed discussion about how it could best be designed and integrated (Sections 4.2 and 4.4). For example, we used much larger buttons than most users would desire (to make sure that first-time users quickly were made aware of their choices, as we did not tell them that they would need to make choices) and exposed users to the playback bar even when the they were not expected to make branch choices (to make sure that the users were made aware of the playback bar).

Figure 8 summarizes our average results; again using a 1-20 scale (low-to-high) and 95% confidence intervals. Here, we only observe significant differences for the perceived user experience (e.g., non-overlapping confidence intervals and only 7 out of 32 users preferring the implementation with the playback bar). While this user preference result at first may seem disheartening, it is important to note that we intentionally (over) exposed users to the playback bar by not hiding it between branch points, for example, expecting that users needed some time to build an increasing appreciation for the general concept. Furthermore, the question was asked relatively early in the study, after the participants only had seen the two example implementations, and the majority of the users repeatedly (in the later part of the study; Section 5.5) brought forward that they wanted the possibility to hide the playback bar during regular playback. Based on the per-feature evaluation and discussion (Section 5.5) it appears that simply making the playback bar disappears from the video whenever the viewer keeps the mouse pointer still (or alternatively whenever the user is not presented any branch options) should increase these scores significantly.

With the exception for the perceived user experience, the average differences are again small, and the results align well with our previous results. For example, as for the measured branch selection delays, the self-reported "speed" differences are small both in terms of averages (Figure 8) and in terms of users reporting same-or-higher scores for the implementation with playback bar (16 out of 32). Despite not seeing any significant differences in the average "accuracy" scores, 21 out of 28 (or 25 out of 32 if including 4 "don't know" answers) report same-or-higher score for player with the playback bar. Finally, on the "best help" question, 23 out of 32 users give same-or-higher score for the playback bar implementation.

Interestingly, the playback bar typically scored relatively higher when shown as the second implementation. Among the users that gave same-or-higher score for player with the playback bar, 10 out of 16, 15 out of 25, 12 out of 23, and 6 out of 8, respectively, used the playback bar implementation second. This may further suggest that users need some time to digest the branched video concept and that they build a higher appreciation for branched videos as they interact more with the video content and the player. For this and prior mentioned reasons (e.g., regarding the naive integration of the playback bar into the player) it is perhaps not surprising that the implementation without a playback bar had higher user experience scores (at this time of the study).

**Table 3: Summary of like/dislike evaluation results. Here, we use (**) to indicate when a result is significant.**

| Feature | Like | Dislike | Don't know |
|---|---|---|---|
| Simple zoom-and-follow | 13 | 13 | 6 |
| Prune non-selected paths | 26 ** | 4 | 2 |
| Fish-eye | 1 | 28 ** | 3 |
| Mouse-eye | 3 | 26 ** | 3 |
| Matching branch labeling | 8 | 21 ** | 3 |
| Highlight path when hovering over button | 18 ** | 8 | 6 |
| Clickable playback bar, without any buttons | 0 | 28 ** | 4 |
| Explicitly place buttons in tree | 16 | 14 | 2 |

## 5.5 Feature-by-feature analysis

Finally, we evaluated each candidate feature outlined in Section 4 one at a time. For each feature, we first presented the player with the particular feature turned on. We then confirmed with the user that he/she had noticed and understood the effect of the feature, asked whether the user liked/disliked the feature, and finally asked the user to motivate and explain his/her answer. Table 3 summarizes the results of all features with a clear like/dislike test. In the following we discuss these and other features, one at a time.

**Tree-size related questions and what to show:** Here, we compared the *simple zoom-and-follow* and *prune non-selected paths* against showing the *full structure* (default). The *simple zoom-and-follow* feature received mixed reactions; half liked it and half did not. The primary reasons that users liked this feature was that it is "more dynamic", "does not reveal the structure of the entire tree" (e.g., ability to look too far into the future), being more area efficient, and that it has a "nice pace". In contrast, the users that did not like this feature commented that they felt that it "draws more attention" and increased the stress in their decisions (e.g., as branches move relative the player, and that this subconsciously increases the cognitive load), and others did not like the feature as they prefer a better overview of past and future choices.

The *prune non-selected paths* feature received very positive scores (26 out of 30 with a distinct opinion liked it). Here, the users stated that they liked that it "removes unnecessary information", "reduces the remaining tree size", "focus on the part of interest", "reduces the chance regretting past choices", and that it "feels more realistic".

We also asked participants about the maximum number of branch choices per branch point and the maximum number of branch points per video session. The majority of users argued for at most 2 or 3 path choice per branch point and nobody suggested more than 4. (9 prefer 2; 10 prefer 2-to-3; 4 prefer 2-to-4; 3 prefer 3; and 1 said 3-4.) In contrast, we saw high variability in the maximum number of branch points per video. However, in general, most users responded that the number depends highly on the length and type of video. Many argued that they would not want to have to make choices too often, as they typically would prefer to sit back and relax, but would be okay with more choices if they felt that their choices mattered and improved the storyline.

**Visual appearance of branches:** We first asked users to select their preference among the branch shapes defined by four different mathematical functions. Here, most users (15 out of 32) selected our default function *arctan*, 11 out of 32 desired the use of 90-degree branches, 3 out of 32 could not decide between the 90-degree option and arctan, 3 out of 32 selected acos, and nobody wanted to use straight lines between the branch points (*abs*). This shows that our default choice (arctan) indeed is a good choice. Second, we note that almost no user liked the *fish-eye* feature (1 out of 32 users) or *mouse-eye* feature (3 out of 32 users), suggesting that such distortions may not help the users in this context. The more positive answers for *mouse-eye* may suggest that these type of features may be better used to explore the branch structure at the time of video creation or when the video is paused, for example.

**Integration and branch choice labeling:** Out of the four features considered here, *highlight path when hovering over button* received the most positive reviews (picked by 18 out of 26 with a distinct opinion). The users that liked this feature said that it was "simple", "made it clear what path you consider choosing", "made the user feel more in control", and "connects the purpose of the playback bar to the buttons". The users that did not like this feature felt that it "did not contribute anything useful", "was not that noticeable", "takes focus away from the video", "reveals information about the number of candidate paths", or felt that "a similar feature may be better used for the purpose of informing users what paths they have already watch" (e.g., when watching a video the second time). We note that the comments again are highly diverse, highlighting that there likely is no silver bullet that will be best for all users.

Perhaps not surprisingly, nobody liked the idea of completely replacing the branch selection buttons with a clickable playback bar. However, interestingly, more than half (16 out of 30 with a distinct opinion) liked the feature that explicitly placed the buttons in the tree shown on the playback bar. Some of the users that liked this feature felt that this approach was less distracting than the large see-through buttons, whereas users that did not like the feature felt it blocked the playback bar, or that it draws focus from the screen. Finally, only a few users (8 out of 29 with a distinct opinion) liked the feature that adds matching branch labels (e.g., "A" and "B") to both buttons and branches. Instead, some users argued that it may be best to give users one way to select a path and that using the playback bar for branch selection may take some getting used to.

**Positioning, visibility, and timing:** As noted above, many users felt that the buttons were too large and that the playback bar should be hidden between branch points. This is perhaps not surprising and matches well with how the size and time parameters were selected in the two example players used for the study (i.e., so to help ensure that users quickly would be made aware of the branch choices and playback bar). Naturally, in a production environment, a more subtle design likely would be desirable.

We showed example videos with branch options presented 3, 5, and 7 second before the branch points. Out of the 32 participants, 16 prefer 3 sec, 8 prefer 5 sec, 2 prefer 7 sec, 4 said that either of alternatives 3 or 5 sec would be good, and 2 said that 5 or 7 sec would be good. These results suggest that users tend to prefer shorter path selection times (main reason: less view interference).

**Table 4: Fraction of branch choices not made within users' selected threshold.**

| | | With PBB | | Without PBB | |
|---|---|---|---|---|---|
| **Threshold** | **Users** | **First** | **Second** | **First** | **Second** |
| 3 seconds | 16 | 0.69 | 0.81 | 0.81 | 0.75 |
| 3-5 seconds | 4 | 0.25 | – | – | 0.00 |
| 5 seconds | 5 | 0.33 | 0.20 | 0.30 | 0.17 |
| 5-7 seconds | 2 | 0.00 | 0.00 | 0.00 | 0.00 |
| 7 seconds | 2 | – | 0.25 | 0.00 | – |

Interestingly, many users mentioned that they would have liked if the video simply continued along a (pre-selected) default path in the case that they did not make a choice in time of the branch point. This is similar to the ideas assumed in prior work by Krishnamoorthi et al. [18] and what is used by Eko [8]. To investigate what fraction of the users would have made their choice within different decision periods, we refer back to Figure 7. Combining all measurements, only 29.7% of the selections were done within 3 sec; 66.4% within 5 sec; and 86.7% within the first 7 sec.

Finally, Table 4 shows the fraction of times that the observed branch selection times (of different categories) are larger than the threshold suggested by the participant. Here, we break down the results per threshold, per player implementation used, and whether the data point is from the first or second experiment. We note a significant number of violating cases, suggesting that users perhaps were too optimistic of their ability to make fast decisions.

## 6 CONCLUSIONS

In this paper, we first presented a novel open-source player (made available with this publication) implemented using dash.js, which includes a generalized playback bar that visualizes the tree-like branched video structure and the buffer levels of the different branches. The player allows both the playback bar and the presentation of branch choices to be easily customized with regards to visual appearance, functionality, and the content itself. Second, we presented the results of a three-step user study in which we evaluated the playback bar and compared with alternative designs and branch-related features. The user study highlights that the branched playback bar can add value at the cost of very limited perceived client effort. Our findings also suggest that further improvements are possible by hiding the playback bar between branch points and enhance the playback bar with high-scoring features such as *prune non-selected paths* and *highlight path when hovering over button*. Based on the study, we have implemented a version in which (i) the playback bar is hidden whenever the mouse is still, (ii) playback continues along the default path (when a path selection is not made before reaching a branch point), (iii) a count-down timer has been added to show how much time the user has left to make a decision (before a branch point), and (iv) both the choice buttons and branches in the playback bar have been made smaller and more discrete. Creating larger content catalogues and evaluating such further improved players on larger and more diverse client populations provide interesting avenues for future work.

**Acknowledgements:** The authors are thankful to the participants of the study and to our shepherd Axel Carlier and the anonymous reviewers for their feedback. Early designs also benefited from discussions with Derek Eager and Anirban Mahanti. This work was funded in part by the Swedish Research Council (VR).

## REFERENCES

[1] 2018. ITU-T P.1203: Objective video QoE standard. (2018). https://www.itu.int/rec/T-REC-P.1203

[2] 2018. VQEG: Objective video quality assessment. (2018). https://www.its.bldrdoc.gov/vqeg/projects/audiovisual-hd.aspx

[3] S. Akhshabi, A. C. Begen, and C. Dovrolis. 2011. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *Proc. ACM MMsys*.

[4] N. Carlsson, D. Eager, V. Krishnamoorthi, and T. Polishchuk. 2017. Optimized Adaptive Streaming of Multi-video Stream Bundles. *IEEE Trans. on Multimedia* 19, 7 (7 2017), 1637–1653.

[5] N. Carlsson, A. Mahanti, Z. Li, and D. L. Eager. 2008. Optimized Periodic Broadcast of Nonlinear Media. *IEEE Trans. on Multimedia* 10, 5 (2008), 871–884.

[6] dash.js. 2019. (2019). https://github.com/Dash-Industry-Forum/dash.js/wiki

[7] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang. 2011. Understanding the Impact of Video Quality on User Engagement. In *Proc. ACM SIGCOMM*.

[8] Eko. [n. d.]. Eko (interactive video website). ([n. d.]). helloeko.com

[9] S. Goedegebure, A. Goralczyk, E. Valenza, N. Vegdahl, W. Reynish, B. V. Lommel, C. Barton, J. Morgenstern, and T. Roosendaal. [n. d.]. Big Buck Bunny (video). ([n. d.]). https://peach.blender.org/download/

[10] GPAC. [n. d.]. MP4Box (multimedia packager). ([n. d.]). https://gpac.wp.imt.fr/mp4box/

[11] C. Gutwin and C. Fedak. 2004. A Comparison of Fisheye Lenses for Interactive Layout Tasks. In *Proc. of GI*. 53–60.

[12] C. Gutwin and C. Fedak. 2004. Interacting with Big Interfaces on Small Screens: A Comparison of Fisheye, Zoom, and Panning Techniques. In *Proc. of GI*. 145–152.

[13] T. Hossfeld, M. Seufert, C. Sieber, T. Zinner, and P. Tran-Gia. 2015. Identifying QoE optimal adaptation of HTTP adaptive streaming based on subjective studies. *Computer Networks* (2015).

[14] T. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. 2012. Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard. In *Proc. ACM IMC*.

[15] T. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. 2014. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. In *Proc. ACM SIGCOMM*.

[16] D. Johansen, P. Halvorsen, H. D. Johansen, H. Riiser, C. Gurrin, B. Olstad, C. Griwodz, Å. Kvalnes, J. Hurley, and T. Kupka. 2012. Search-based composition, streaming and playback of video archive content. *Multimedia Tools and Applications* 61 (2012), 419–445.

[17] V. Krishnamoorthi, P. Bergström, N. Carlsson, D. Eager, A. Mahanti, and N. Shahmehri. 2013. Empowering the Creative User: Personalized HTTP-based Adaptive Streaming of Multi-path Nonlinear Video. *ACM CCR* (2013), 53–58.

[18] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, and N. Shahmehri. 2014. Quality-adaptive Prefetching for Interactive Branched Video Using HTTP-based Adaptive Streaming. In *Proc. ACM Multimedia*. 317–326.

[19] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, and N. Shahmehri. 2015. Bandwidth-aware Prefetching for Proactive Multi-video Preloading and Improved HAS Performance. In *Proc. ACM Multimedia*. 551–560.

[20] S. Krishnan and R. Sitaraman. 2012. Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-experimental Designs. In *Proc. IMC*.

[21] B. Meixner. 2016. A pattern-based evaluation of download and cache management algorithms for annotated interactive non-linear videos. *Multimedia Systems* (2016), 1–35.

[22] B. Meixner. 2017. Hypervideos and interactive multimedia presentations. *Comput. Surveys* 50 (2017).

[23] B. Meixner and C. Einsiedler. 2016. Download and Cache Management for HTML5 Hypervideo Players. In *Proc. ACM HT*. 125–136.

[24] B. Meixner and H. Kosch. Sep. 2012. Interactive Non-linear Video: Definition and XML Structure. In *Proc. ACM DocEng*.

[25] B. Meixner, J. Köstler, and H. Kosch. 2011. A Mobile Player for Interactive Non-linear Video. In *Proc. ACM Multimedia*. ACM, 779–780.

[26] B. Meixner, K. Matusik, C. Grill, and H. Kosch. 2014. Towards an easy to use authoring tool for interactive non-linear video. *Multimedia Tools and Applications* (2014), 1251–1276.

[27] B. Meixner, B. Siegel, P. Schultes, F. Lehner, and H. Kosch. 2013. An HTML5 Player for Interactive Non-linear Video with Time-based Collaborative Annotations. In *Proc. ACM International Conference on Advances in Mobile Computing & Multimedia*. 490–499.

[28] L. A. R. Neng and T. Chambel. 2010. Get Around 360$^{o}$ Hypervideo. In *Proc. MindTrek*. 119–122.

[29] Netflix. 2017. Puss in boots: Trapped in an Epic Tale (screenshot). (2017). https://www.netflix.com/title/80151644

[30] N. Sawhney, D. Balcom, and I. Smith. 1997. Authoring and navigating video in space and time. *IEEE Multimedia* 4, 4 (1997), 30–39.

[31] U. Spierling, S. A. Weiß, and W. Müller. Dec. 2006. Towards Accessible Authoring Tools for Interactive Storytelling. In *Proc. ACM TIDSE*.

[32] T. Stockhammer. 2011. Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles. In *Proc. ACM MMSys*. 133–144.

[33] L. Toni and P. Frossard. 2017. Optimal Representations for Adaptive Streaming in Interactive Multiview Video Systems. *IEEE Trans. om Multimedia* 19, 12 (Dec. 2017).

[34] VideoLAN. [n. d.]. x264 (software library). ([n. d.]). https://www.videolan.org/developers/x264.html

[35] M. Wijnants, J. Leën, P. Quax, and W. Lamotte. 2014. Augmented video viewing: transforming video consumption into an active experience. In *Proc. ACM MMSys*. 164–167.

[36] M. Wijnants, P. Quax, G. R. Ruiz, W. Lamotte, J. Claes, and J.-F. Macq. 2015. An optimized adaptive streaming framework for interactive immersive video experiences. In *Proc. IEEE BMSB*. 1–6.

[37] Wikipedia. 2019. (2019). https://en.wikipedia.org/wiki/Black_Mirror:_Bandersnatch

[38] Y. Zhao, D. L. Eager, and M. K. Vernon. 2007. Scalable on-demand streaming of nonlinear media. *IEEE/ACM Trans. on Networking* 15, 5 (Oct 2007), 1149–1162.