Empowering Storytelling in Dash

Empowering Storytelling in Dash

I. INTRODUCTION

Dash, a collaborative hypermedia web-based application, allows users to store and manipulate all types of documents on a 2d unbounded canvas, as shown in figure 1. Among all these documents, the user can create links and annotations, either to specific parts of the document or the document itself. When an author adds and links multiple documents in the canvas, he or she creates a corpus which other users can naturally traverse and explore by following those links, marking the most simple storytelling experience Dash offers.

To allow the author to direct a specific narrative over his or her documents in a corpus, Dash has supported a simple, linear version trails, or a sequence of waypoints that zoom and highlight particular regions (see figure 2), similar to a Prezi presentation. However, unlike a Prezi presentation, the user can still "hop-off" the trail and explore the corpus at any point within the trail’s progression and return back to where he or she previously was before "hopping-off" the trail.

For a long time in Dash, this limited trail system has been the core way to offer author-controlled storytelling over his or her corpus, restricting the possible use-cases and value authors can get from their corpuses. Imagine, if you will, that an author in Dash would like to create a storytelling experience with her or her voice guiding moving elements, along with a final conclusion that was specifically tailored from the audience’s decisions while traversing the story. Like so, adding the following three features outlined below to trails - video and screen replaying, programmatic functionality to slides in trails, non-linear trails - will open up storytelling to more creative uses for both the author and users.
II. RELATED WORK

Within the space of data-driven storytelling, many pieces of literature thematically overlap in some way with Dash’s features: web-based application, information-rich environments, trails of visualization as bookmarks, annotations, highlighting, and reuse of elements [1][2][3][4]. Many applications in this field investigate interactive experiences within the process of storytelling, recognizing that “author-driven scenarios follow a linear structure”, and “reader-driven scenarios give control to the person receiving the information by providing an open system, allowing for free interaction” [1]. Reader-driven experiences allow for “inviting verification, new questions, and alternative explanations” [2]. Many studies have focused on reader-driven storytelling in data-rich domains - “such as finance and journalism” - or more opinionated journalism pieces that allow "the visualization to function in the place of a written story" [2]. Notably, the importance of hosting the storytelling experience as a web-app was cited as an important step to offer accessibility to all audiences [1][2].

While said, for making authoring more accessible, “a thorough understanding of the design space for narrative visualization has yet to emerge” [2], until fairly recently. In 2019, Donghao focused on “designing interactions and implementing user interfaces for visualization authoring”, to “enable non-programmers to create custom visualization for data-driven storytelling”. His focus on design enabled "larger audiences to create visual data-driven stories using different presentation media, leading to an overall enriched diversity of visualization designs" [3].

Overall, while there is a strong foundation for data-driven storytelling, there appears to be a weak development of design research that allows non-coders to create a storytelling experience that lies somewhere on the "spectrum" between an "author-driven" and a "reader-driven" interactive experience, especially in information-rich environments like Dash [1]. Relevantly, McPherson boldly claims that “new forms” of experimentation - specifically involving “multi-modal communication” are necessary “to to advance (and perhaps save) scholarly publishing in the humanities” [4], marking an extremely important implication that should motivate design research to allow anyone to create data-driven stories virtually.

III. ADDING VIDEO AND SCREENREPLAYING TO TRAILS

In the modern era of storytelling, we often incorporate temporal media (audio and video) into presentations, which also supports an important goal of offering more modes of communication within trails. We wanted to create an simple, natural, and quick way to spawn interactive presentations that includes the author’s webcam and corresponding progression through the corpus naturally within Dash. To do this, we based the design of this feature off of the "authoring by doing" philosophy. With this feature, an author, after clicking record, can naturally give a presentation into his or her
webcam as he or she pans and zooms throughout the corpus. The result from the recording becomes attached to a slide on the trail, where the users can watch the author’s presentation like a video moving the corpus as a part of the trail. However, like with a normal trail, they can still explore the corpus at any point within the trail, which will automatically pause the progression of the author’s presentation. At any point, the users can return to where they left off from the video by clicking play.

The most natural, effective, and simple way to integrate video and audio into trails synchronizes the author’s movements on the corpus with what he or she is recording through a camera (e.g. webcam); this presentation is stored onto a slide, becoming a "videoslide" within the trail. Realizing that this chosen integration forces the author to do "one-takes", we wanted to choose a design that relieves the pressure off the author to do one whole perfect presentation in one sweep. Hence, the recording widget supports recording the overall videoslide in undo-able, removable, and exchangeable (in order) segments. For example, if an author wanted to make a 20 minute video presentation, he or she can split it into ten 2-minute chunks to avoid an unfavorable result from shooting the while video within a 20 minute block. Other considerations - voicing over a whole trail after its creation - were inefficient, by either an extra step or hindering the replay experience.

With these ideas in mind, we created a mockup recording box with options to, one include audio, and, two, track screen movements. It also has a record button, an undo button, and empty rectangular area to hold the segments. Segments can either be dragged around to manipulate order or dragged off-screen to be removed. While actively recording the video for a presentation trail, the panning and zooming actions of the author are captured for reply later (among multiple tabs). As expected, the final order of segments will match the final, continuous order of the video and replaying.

In this way, we were able to follow the principle of "authoring by doing" while maintaining failsafes for the associated mistakes that can come with the flaw of the design, with accessibility to all types of input devices.

Pertaining to the users progressing through these video slides, we wanted to design this so that the corpus will navigate to where the author began at the beginning of recording as a video widget appears. This vbox holds the recording of the author’s video (from the webcam), and the timeline of the video maps to where the author was within the corpus at those time codes. Seeking on the timeline will navigate the user to those corresponding points within the corpus, and clicking play will begin the guided presentation, playing the video and moving the screen to the author’s interactions (pan/zoom) with the corpus at that point while recording the video. During a replay, when the user interacts with the screen, it automatically pauses the video to allows users to explore the corpus freely. When the user resumes the video, it returns to where the user left off to begin exploring. The end of the video marks the end of the slide.

The user interface for the vbox corresponding to the slide corresponds to the UI for a normal vbox; however, being central to the flow of the narrative, we decided the vbox should be overlaid and easily draggable over all content on the screen to ensure it’s always seen and easily movable if it were to cover any important content.

Final implementation of the vbox and recording box matched the mockups, with slides having a recording icon on them to prompt the recording box turn them into a videoslide (see figure 3). In practice, authors usually liked the idea of WYSIWYG style of creating and presenting presentations, along with the segmentation of the videos. Also, the aspect of having their faces within a presentation suits more interpersonal and professional use cases, such as in "virtual lessons" (discussed in section VI).

IV. TRAILS AS PROGRAMS

To unlock the potential of trails, we wanted to allow the author to offer new unforeseen, interactions inside their trails to improve the creative possibilities of storytelling. Basic trails would only edit the user’s viewing position and zoom onto documents, with some options for "Ken-Burn's" style animations, given by a collapsible side panel. Rethinking the actions of trails as programs, in which the slides of trails act as an operation that can also edit documents - its size, position, content, etc. - inside the corpus, simplified our implementation and reduced possible conflicts that could occur with cyclic modification of data in such ways. The

4 The social media application TikTok inspired this recording structure, but it does not have as much actionable modifications like re-ordering, undoing - with the segments themselves.

5 Special thank you to Jenny Yu, who was also involved heavily in the design process of the recording box.

6 WYSIWYG literally stands for What You See Is What You Get, implying that when the author makes content, what the author sees while making the content is exactly what the users will see - versus a HTML, a markup language, for contrast. WYSIWYG principles better support non-coders. Dash by default employs WYSIWYG since the author and viewers see the canvas in the exact same way - there is no intermediate step the author must do to "compile" or "prepare" a presentation, since a trail just directs over certain parts of the shared canvas.
Figure 3. The final implementation of the overlayed (over the corpus) recording and videobox. The top 2 images show the features of the recording box. The top image shows the author actively recording a segment of a video. The middle image shows the state between recording segments and confirming the video; specifically, the author is moving the last segment in front of the other two. The bottom image shows the replaying of the recorded presentation. The screen replaying is hard to display statically, so, to view video examples of video and screen replaying, please click on one of the following youtube hyperlinks: [basic] [advanced].

The corresponding new implementation consisted of a script executing when the user traverses to the slide itself, which greatly expands the functionality and adaptability of trails in numerous ways, including offering the ability for non-linear trails (covered in the next section, for now, we focus solely on how an author can easily set the modification of content or layout of a document through a trail’s slide).

To design the way for authors to offer modification of content, as before, we employed the principles of "authoring by doing" and WYSIWYG to reduce overhead and complexity of generating the underlying scripts the slides contain. The design has three new buttons, leveraging the internal implementations of documents: the content and layout "docs" within a document. To easily embed a document change on a slide within a trail, the author can pin the initial state of the document to be edited as a slide to the presentation then modify the document. Then, the author can re-pin either that document’s new content, layout, or both as the next following slide in the trail (pinning shown in figure 4) with an example of making a trail that edits a text document in figure 5.

In many cases, an author would want to move the order of slides within a trail around after creating them.

---

7 Special thank you to Bob Zeleznik for implementing this new mental model of slides within trails.
8 We store documents in Dash by classifying their layout and content separately to allow for easy aliasing and cloning when viewing or manipulating the document in multiple, unnatural scenarios. It conveniently worked well as a mental model for how users can modify documents using trails.
9 In Dash, pinning refers to adding a document onto the trail as a slide, as shown in figure 4.
Figure 5. An example of making a simple trail that modifies a text document, following the WISIWIG principle. In the topmost image, the author pins both the layout and content to the trail using the final pin option to offer a starting state for the document. Directly under that, the author edits the text and negatively translates the text document’s Y value then re-pins the layout and content of the document. The image under that shows the creation of the slide that modifies the content. Finally, the author goes back to the first slide of the trail, which restores the initial state of the document. In this way, the author made a trail that modifies the layout (position) and content (text) of a document.

However, unlike before when the order of slides were fully independent, shuffling an individual slide that modifies content can now have adverse effects on the desired or original outcome of the overall trail. To help with this, we wanted to also have a mechanism to identify independent sections of content that can safely moved. For this, hovering over or traversing a slide that pertains to a layout or content change will be grouped with the total sequence of slides that affect it, as shown in figure 6.

The final implementation of trails that modify content fulfilled the functionality and seamless design ex-
plained above. While the mechanism to create the slides was well received, after some use, authors found it useful to have some way to update the layout and content of the modified documents after the fact, so offering buttons on the slides to update the layout and content of the slide that modifies documents allowed this in a simple way, as shown in figure 7. This allows for authors to easily create more interactive trails, without compromising the simple nature of creating both basic and advanced trails, as shown in the "riddle collector" use case.

Figure 7. The slide buttons for layout (with the one for content, marked by C, to the right of it) that allow the author to easily update the script on the slide that edits the document to match the newer version of that document displayed on the corpus.

V. NON-LINEAR TRAILS

To offer more interactivity into trails to better support "reader-driven" experiences, we wanted to the user’s decisions within trails to affect the overall progression of that trail. To do this, we investigated two types of non-linear trails: nested and branching trails. Nested trails provide associated content at a more in-depth level, for which the viewer can decide the fidelity of detail while progressing throughout the trail. Branching trails literally can go to many places, but they generally provide content tangential to the main subject of the presentation. Below discusses our design implementation of both of these types of non-linear trails, considering our approach to reduce complexities specifically brought on the fact that slides can edit a document’s content or layout.

A. Updating Nested Trails

Dash has supported a basic tree structure to support nested trails, in which the progression of trails matches exactly the slides shown in the trail window. Users can expand and collapse subtrails by clicking on the parent element, and, in this way, a basic nested slide structure allows users to decide the depth of subsections of their trail, as shown in figure 8. We needed to incorporate the programmatic translation of this, considering that subtrails can now represent subroutines since slides can now execute a script. To allow the author to control the default preference of whether to run a subroutine in a simple way, we decided to use a toggleable flag, initially selected by the author. If the button is set to active on a parent slide, its child processes will recursively run as the trail progresses to the next slide in the UI list. All parent slides with a child will have this toggleable button, so the author can control how deeply nested, by height, which subroutines will run, as shown in figure 9.

This feature vastly improves what the author can do with "trails as programs", as having collapsed nested subroutines will run multiple slides in succession pseudo-instantly with the progression of one (parent) slide within the trail. Without having this ability run multiple slides under the progression of one slide, many use cases that involve editing multiple documents at once, like "become a top500 in overwatch" below in section VI, would not be possible.

The tree structure harnessing these subroutines allows authors to properly organize deeply nested subroutines, with the ability to quickly name and toggle off certain subroutines under a parent. Many similar presentation softwares, such as Microsoft PowerPoint, support running multiple animations of elements (that have motion paths or fade in content), but its structure for denoting groups of animations to run simultaneously is very primal and limited in (linear) nature; it only uses numbers to convey to the author structurally which animations will run in-tandem (see figure 10). Also, no concept of subnested trees that can be easily toggled on or off for experimentation exists in PowerPoint. From this comparison, Dash’s design is more effective and clearer for authors as it offers stronger organizational semantics and hierarchy.

Lastly, just as the author can share his or her preference as to whether particular subroutines run, those who will use the trail can also use this toggleable button to decide if the subroutines run, allowing these users

10 We refer to a routine as one of these programmatic slides that edits a document when traversed in a trail. A subroutine is a series of routines organized under a parent slide when using Dash’s trail tree structure.
to form different outcomes from the same trail, based on which subroutines ran. This implication reflects the first time in Dash where decisions in fully following a trail can cause a different overall output state of the corpus.

B. Branching Trails

We wanted to add support to navigate and explore branching trails, or entering another trail while currently progressing a trail, which came as a byproduct from revamping trails to be programs. That is, now that traversing any link pointing to a slide will cause the user to "hop onto" the trail, clicking links on documents within the corpus while on a trail will cause the user to enter a different trail. In this fashion, the user has decisions while on a trail (or just exploring the canvas) to explore different trails. Simultaneously, the author has the ability to manufacture branching trails, and the user’s actions in following those will determine the overall resulting path of his or her exploration, in ways the authors may or may not predict.

11 There is also an option the clicking the document itself will act like a button to enter a new trail.
Figure 10. Setting two (or more) animations in Microsoft PowerPoint to run simultaneously with one progression of the presentation requires those two animations to share the same sequence numbers. The numbers indicate the animations sequence number, and one must physically set these numbers to be the same, versus dragging and dropping under a parent in Dash.

The UI for this, at minimum, must, one, tell users where they are (in their path while exploring branching trails), and, two, allows users to return to a previous part of their exploration history. An application of breadcrumbs\textsuperscript{12} would work well to achieve these goals; however, the crumbs must offer enough detail to help users remember the slide from which they diverged. For this, our initial design incorporated a visual image and hover-ability on that visual cue to offer more helpful, memorable information on the place from which they diverged. If they click on the visual image, it would bring them back to the trail from which they diverged. This "deck of cards" design also would naturally allow space for deeply nested exploration, as the visual images would appear stacked on top of each other and expand when hovered.

In the final implementation, users can view an overlaid breadcrumbs UI with buttons that show the history from switching between trails, as shown in figure\textsuperscript{11}. The buttons, only including trail name and slide name, will revert the users back to where they once were within in that previous trail. Edits (made by users or the trail’s routines) to the documents affected by the trail’s routines will also be reverted (more on this in the following subsection)!

The concept of branching trails greatly expands the sheer number of use cases and creative storytelling the author can employ by allowing authors to have a stronger control of the narrative without making the users switch trails themselves. Also, this functionality allows for the directing to commonly-used trails from the canvas itself, allowing users to reuse or repurpose those trails within other trails.

C. On the Complexity of State Management

When discussing trails as programs and their corresponding branching trails, many complex, difficult considerations boggled our thinking of how traversing slides that edit documents can lead to tricky, unintuitive states of the corpus. In other words, the corpus’s state itself is now dynamic, based on where a user is within a trail; therefore, edits done at certain points on the trail force the question of unsynchronized version control. Our generalized, most-intuitive solution to these situations is that the content displayed to the viewer on the trail matches exactly what the author saw when making the trail, in as many cases as possible. To achieve this, first, if the user jumps to a slide on a new trail, the preceding slides of a trail must be ran to ensure the content that would have been edited by traversing those slides. Second, most content and layout shifts done to documents on the most recent version of the corpus that a trail’s progression will edit will not actually be produced when the trail is run. That is, the trail slides themselves must be edited, updated, or recreated to force a change to those documents when traversing that trail. This also applies to users edits to those specific documents while browsing branched trails; if an explorer returns to a previous trail after making edits in a different trail, those edits will be effectively undone.

To achieve this programatically, once a trail is hopped onto, it must be run first, fully in reverse so that that documents match exactly how the author intended.\textsuperscript{13} Therefore, for slides that specifically edit documents, those corresponding documents will be shown exactly to the version that the author saw when making the trail. With this implementation, however, documents that are not affected by slide edits will still reflect the

\textsuperscript{12} Breadcrumbs, a user interface used for strong navigation, lets users 1) know where they are by showing their history while traversing many links and 2) allows them to return to a previous link to which they have been. Users can find breadcrumbs most effective when they traverse hypermedia systems, such as a file hierarchy or the World Wide Web.

\textsuperscript{13} This may seem complex, but running the trail backwards will actually update the content to exactly what that trail had at the time it was created, since the slides that edit content stored the before and after of the change. We can swap those values when run in reverse.
VI. USE CASES

A. Riddle Collector

For those who love to collect riddles, Dash is perfect for storing and sharing those riddles, as now it’s possible to hide the answer of the riddle in the trail (and not reveal its answer in the corpus), as shown in figure 12. This ability to allow the author to initially hide content until unlocked or found by traversing the trail offers more opportunity for interactions for the explorer. This is a simple use case, made possible by allowing trails to edit the content of documents.

B. Virtual Lesson

For those teachers who need to make interpersonal, organized, and asynchronous content, Dash’s can allow a personalized, effective way for students to learn. Consider figure 13, in which the author has prepared a trail with videoslides, discussing documents on the canvas like in a normal teaching style. However, the author has also prepared some additional, optional content that delves deeper into the discussed topics, as indicated by some key video slides that have nested trails with more video and non-video slides under them. The teacher’s students can decide on if they want to explore the content based on interest, skill, or time by expanding child slides in the tree structure the trail follows. For this specific example, students can choose to watch an embedded video the teacher added to the trail that walks through a specific example of Dijkstra’s algorithm, if a particular student would find it helpful, as shown in figure 14.

At any point while following the teacher’s videoslide, the students can also stop the video by interacting with the corpus, allowing them to digest the content at their own pace - being able to create their own links and annotations on the documents - then return to where they were.

While not perfect, this works well in all non-overly complex situations. A whole paper alone can be written on how to handle reverting state within trails and corpuses. In general, the best solution would give the author and explorer power to choose how they prefer to see it, but it’s hard to do that in a non-overwhelming fashion.

content displayed in the most recent version of the corpus. In this way, the state of the corpus while branching trails follows an intuitive, effective merging heuristic that offers a healthy balance between the number of forward-compatible documents and the number of the ones that must revert to the state of trail 14. Over time, the author can understand this method and leverage it to be precise with exactly what content will be reflected in the future within his or her trail, but, in the creation of trails for novices, this should still allow most trails to be fairly independent with future edits to the corpus.
stopped the video. In this way, the learning process becomes personalised, matching the students interests and desires, arguably offering a better learning experience than in-person, non-dynamic traditional lectures.

This use case bases off the video and screen-replaying features of Dash’s trails, allowing for a seamless authoring and viewing experience, along with the basic multi-linear tree structure to offer organization into optional, more-detailed content for the students. The authoring experience for the videoslides matches the virtual translation of teaching: a recorded lecture with the teacher interacting with the documents on a screen. Also, the authors can incorporate and organize (with good semantics under the tree structure) as much information as necessary without time restrictions onto the trail or canvas, and the students who need the more detailed content can choose to delve into it.

Disregarding the focus on trails alone, using Dash in this use case centralizes all material for a course into one corpus (with the trails going over it), benefiting both students and other (future) teachers. With Dash’s collaborative nature, anyone can add links and annotations to any part of the corpus. For example, linking material in a presentation from Week 1 to material in a presentation for Week 11 allows students to easily follow that link to reacquaint themselves with previous content - or even branch onto the trail of that Week 1 presentation and start exploring it. A full-fledged learning environment for a course within Dash, if came to light, could offer one of the best, most-inclusive, and collaborative ways to learn virtually.
C. Concentration Advising

For many college students, deciding which courses to take to fulfill their concentration, or major, can be a difficult process. In this case, a central place holding peer-to-peer advice, reviewing concentration requirements, and overviewing previous iterations of courses could help provide extremely helpful information to make selecting courses easier. An advisor can spearhead a collaborative corpus with associated branching trails to offer this information in an interactive, specialized experience for student exploration and addition of their own opinions.

In this use case, the main trail alone outlines what is necessary to complete a degree, with each slide offering a broad overview of each of these components. Consider the slide shown in figure 15 for fulfilling the pathway requirement\(^\text{15}\) which overviews all of the pathways and their core courses. Clicking on a course while traversing this trail will put the user on another trail specific to that course, offering an overview of the course, then ending with student opinions of the course, as shown in figure 16. Due to the collaborative nature of Dash, students can post their own advice for courses onto the trail by creating text documents in the review group and adding them as a nested slide within the trail.

\(^{15}\) Specific to the Brown CS degree, we have required thematic pathways, which cover many related courses students must take within that pathway to complete it.

\[\text{Figure 15. Trail representing the requirements for a CS major to earn a ScB degree, as outlined by the trail structure. The explorer has progressed to the 5th slide represented the required pathways, with the child slides overviewing each pathway in more detail.}\]

\[\text{Figure 16. The explorer, starting with the main "ScB Concentration Requirements" trail has branched into the "cs1300" trail, from clicking a link while on the design pathway slide in the previous trail. The user then progressed to the student review slide, in which students have added text documents to the group and added some of them as nested slides within the "cs1300" trail. The overlayed "Branching Trail" breadcrumbs UI (title highlighted) allows the user to return back to the main trail, back to the design pathway slide, upon click of the "SCB CONC: DESIGN PATHWAY" button.}\]
Branching trails are preferred to do this, since many pathways share courses, so the advisor (or students) do not have to recreate trails - they would only have to link the slide that starts the trail, as shown in figure 17. Also, students may be viewing similar courses that overlap content and may want to branch hop onto another course’s trail. Also, semantically, the main trail is an overview of all the requirements, so this allows separation between the content outside of the main trail and what the main trail itself represents. Lastly, nesting so many trails under one main trail can become overwhelming for both the author and user.

In conclusion, the advisor can use the structure of trails to his or her advantage while also crowd-sourcing opinions on courses and professors for students in a decentralized way, resulting in a helpful, community-led corpus to allow students to efficiently find the information they need to select courses.

Branching trails allow for the users to commit to decisions that can put them on new trails. The coach can create buttons to allow the student to have a choice, indicating to which trail the user should jump, as sequenced in figures 18, 19, and 20. If the students commit to the correct decision, the main trail continues to the next sequence of the fight, where the map will update based on the result of the corresponding actions, using the nested subroutines to do multiple updates at a time (see figure 19). If the students choose an incorrect decision, then the user will be guided to a new trail, in which unfavorable events occur and the students must return to where they made an incorrect decision, as shown in figure 21.

D. Become a Top500 in Overwatch

To become among the top 500 players in Overwatch, it requires amazing gamesense and strong mechanics. While improving mechanics is purely a matter of time, a coach can use Dash to teach others how to improve gamesense - a much harder concept to grasp, built through experience - through creating an interactive trail where the students are forced to make decisions that can affect the outcome of individual fights within the game. Choosing a wrong decision in a fight will result in a short, unsuccessful trail, in which the explorer, playing a character, could be eliminated from the game and learn which decision was better by returning to a previous point and making different decisions. A series of correct decisions will guide the students and their team to victory.

In this use case, students can learn in an interactive way how to improve their gamesense by forcing them to make decisions on the fly then explaining the correct decision. The breadcrumbs of the interface allow for quick reversal of the a bad decision, so they can return to a previous point and make a better decision to win the fight. In this way, students can learn through following branching trails how to best position their characters on the map to achieve their goals of reaching top 500. In this way, the coach can teach his or her knowledge in an interactive way by putting his or her students in these light simulations virtually.

With so many maps and characters in Overwatch, an overarching trail can contain multiple parent slides, representing a fresh simulation like this for every map. Hence, a trail’s structure can also offer an organizational

---

16 The organization, structure, and naming of slides within a trial alone can offer a significant amount of meaning to the trail itself.
Figure 19. The trail presents the user with the first decision. Once again, buttons will allow the user to progress the simulation via jumping to a different trail. For demonstrational purposes, the trail list UI is revealed, showing the normally collapsed subroutines in red that update multiple elements (all at once) under the main parent slides to update the corpus.

Figure 20. The user decides to go left from figure 19, and a new scenario is presented after a subroutine moves the elements and slides the map upwards.

Figure 21. The user decides to go up from figure 20, resulting in the enemy eliminating the user’s player. A prompt appears, revealing why the incorrect decision caused him or her to lose the fight, and the user must use the breadcrumb UI to return to a previous trail to reconsider previous decisions.

Figure 22. An overview of the main trail, showing how it and the corpus can be adapted to support multiple maps and scenarios.

VII. FUTURE WORK

These use cases demonstrate specific, interactive ways authors can storytell their passions using Dash. While this paper explains a good first attempt to empower trails in this fashion, future related work on this subject can further extend the ease of use and functional possibilities of trails to further improve storytelling in Dash.

With respect to video and replaying slides in trails, the current implementation of the videobox segmentation can only support around 15 or more slides before becomes too cluttered to use effectively. Also, the numbers themselves do not offer any indication of the video segment they represent. Both of these faults in design limit the segmentation feature to a small number of segments. Possibly designing a scroll feature for the
segment bars after a certain number of them has been reached and offering a video preview on hover can help make the videobox authoring more scalable and easier to use.

With respect to the breadcrumbs UI for branching trails, making accurate thumbnails for visual cues is a strong limitation of web-based applications like Dash. Discussing ways to work around or redesign this to offer a better sense of where the user was previously before he or she diverged off the main trail in will offer a stronger support for those who deeply nest into multiple branching trails or need a reminder of their exploration path after a long break.

While discussing programmatic trails with routines, we found a neat use case which targets the ability to generate trails that match the exact history of an author’s actions, which would allow for convenient, useful access to all stages of version history. Doubled with branching trails, the author could then branch off and build a similar corpus based a previous state of the corpus. More generally, researching more applications of "authoring by doing" (making a trail just by modifying the corpus) like this can offer new, simple ways to create interactive trails for non-coders and non-power users.

Lastly, better understanding and resolving the best way to handle merging the status of the corpus with the status of the corpus when an author built a trail would allow for a clearer mental model for both authors and explorers, especially in cases where editing a document - like a typo on a text document, eventually edited by the trail - in the corpus should bleed through to all stages of that document within the trail, especially if that specific aspect of the document isn’t edited by the trail. Further, as mentioned above, any edits done (on most documents) by the explorer while traversing branching trails would be effectively erased; however, many use cases can include permanent user interactivity with the corpus within a branching trail. One can attempt remove this limitation with designing intuitive UI or rethinking how version control of corpuses work in general, especially when accessed by other users.

VIII. CONCLUSION

With incorporating these three main ideas to empower trails, an author can offer a more tailored virtual storytelling experience in ways that normal, linear presentations could not previously do in Dash, as exemplified by the use cases, without losing the simplicity of creating trails for non-coders. As our society relies more on technology to share information, as in the context of developing this paper during the COVID pandemic, people needed to translate many in-person experiences to a virtual platform, generally accessible to all. Dash - especially now with these empowered trail features - targets that audience, allowing a space for people to create and share interactive experiences in the way they want, with still offering freedom for the user to digest and explore the content in his or her own personal way. It’s exciting that these new trail features will offer the foundation for a much larger range users to share their passions virtually in more personal, creative manner.

Outside of trails alone, a strong 2D unbounded web-based hypermedia system like Dash, if it garnered enough users, could revolutionize the way we store and share information with all. Compared to the competition like Google Drive, with separated programs to open documents and the absence of a trail-like feature, Dash does not offer any obstacles to authors’ creative processes to share their documents stored in the system. There isn’t quite an information-rich, centralized, accessible platform like Dash that allows non-power users to creatively storytell over that information. The essence of these empowered trails mirrors a quintessential goal in this era of information distributed by internet-based technology.


17 Final thank yous to Andy Van Dam (advisor), Norm Meyrowitz (reader), Bob Zeleznik (project manager), and Geireann Roberts (student leader). Could not be more grateful for the amazing support and culture of Andy’s graphics research group here at Brown!