A Novel Design Pipeline for Authoring Tools

Daniel Green, Charlie Hargood, and Fred Charles

Bournemouth University, UK
{dgreen, chargood, fcharles}@bournemouth.ac.uk

Abstract. Interactive digital narrative research presents a diverse range of authoring tools [1,4,8,12,14]. Although our field often publishes the technology, it less often publishes a refined UX design pipeline for those tools' authoring experience. This is despite the UX of these tools long being identified as a key challenge [14] and UX design pipelines being an active area of research in adjacent technologies such as the games that sometimes deliver our stories [3,10,11]. We present a three-stage design pipeline targeting the creation of interactive narrative authoring tools that is informed by existing design pipelines that consider the user and their experience at all stages. We then detail our own application of this pipeline to the design of a new authoring tool, reporting on the methodologies, analyses, and findings of each step.

Keywords: Interactive Narrative · Authoring Tools · User Experience.

1 Introduction

Without a well-designed user experience (UX), accessibility is reduced, and systems can become restricted to those with appropriate technical knowledge. This is true for Interactive Digital Narrative (IDN) authoring tools as much as any other software where poor UX can act as a gatekeeper preventing creatives from using the medium [12]. In order to increase the usability of our authoring tools, we must not only design our authoring tools around the underlying narrative data models. We must also consider the authoring experience at all stages by incorporating them into established design and development processes.

Existing publications in this space detail features of the technology and sometimes an overriding philosophy behind the design of the tool, such as the support of specific patterns [12] or "language-based" principles [9]. However, detailed UX-centric design pipelines, such as in other software or games [10], are rarer, despite discussions in this space identifying key UX concerns with these tools for some time [14]. It follows that there is a gap in this domain that might be addressed by more research into refined UX design methodologies and pipelines.

In this paper, we present our own design pipeline for creation of IDN authoring tools informed by NNGroup's Product Design Cycle[1] and Hamm's *Wireframing Essentials* [6]. We then describe our own application of the pipeline for the design of an authoring tool that supports our *Novella 2* narrative model [5].

[1] https://www.nngroup.com/articles/ux-research-cheat-sheet/

By using the design of our own tool as a case study we highlight what can be learned from the application of this pipeline and how wider authoring tool design might benefit from such a process.

2 The Pipeline

Our pipeline consists of three phases — *Research*, *Discover*, and *Refine*. The *Research* phase includes creation of a persona(s) to represent a typical user of the target audience, analysis of where in a pipeline the tool would best fit, and a Minimum Viable Product (MVP) listing of required features per user requirements and expected functionality. This ensures that the target audience is known, the software is positioned for practical use, and early requirements are identified. The *Discover* phase involves creation of candidate designs in the form of static, low-fidelity wireframes, and involvement of potential users in an exploratory participatory design process, both based on the previous phase's output. After this phase, there should be a final candidate design ready to progress. The *Refine* phase takes the final candidate design, and from it creates a high-fidelity, interactive wireframe. This is then refined further using a RITE philosophy [10,11], ensuring that the final mockup has been tested with actual users to identify and fix any potential usability issues. These three phases are informed by industry-standard techniques, and after completion, will result in a refined mockup prototype that is suitable for use in development.

2.1 Research

Persona Creation Personas typically evolve from insights into actual target users, but for our field, access to professionals is limited, so we instead surveyed relevant job postings and literature to build up a profile. 14 job postings from a range of leading video game studios were used as a representative sample of expected skills and requirements for the target audience of our authoring tool. Using NNGroup's persona guidelines[2], the tabulated skills and requirements were combined with information on the role of a Narrative Designer present in *The Game Narrative Toolbox* [7] to help build our final persona.

Pipeline Analysis It is important for authoring tools development to not only know how, but *when* the tool will be used. For game narrative authoring tools, this is knowing where it fits within the game development pipeline. The point at which the narrative team are involved and their level of engagement with development differs not only between studios, but even between games of the same studio. We looked at the approaches existing game studios take to implementing narrative and considered upstream and downstream processes [7] to help frame the position in the pipeline, taking into account the intended function of our particular tool design. Combining these, we were able to build an understanding of where the tool would fit within a real game development pipeline.

[2] www.nngroup.com/articles/persona

MVP For our prototype tool, the MVP was building a core list of constraints and requirements that consider the persona, pipeline position, satisfy the underlying narrative model functionality, and cater for expected tool functionality.

2.2 Discover

The *Discover* phase is about creating and iterating upon candidate designs based on the output of the *Research* phase, preparing them to be further developed in the *Refine* phase. Taking into account outputs of the *Research* phase, several candidate designs should now be drafted. In our case, we created two high-level design variations as static wireframes. We used an approach based on participatory design [13] allowing us to identify the first impressions formed by users and to uncover any faults or desires early on in the design process.

Methodology The participant demographic was students of IDN who have any level of experience with existing authoring tools. Participants, in individual moderated sessions, were firstly shown an introduction document that covered the persona, a high-level vision of the tool, its required functionality based on the MVP, and a detailed description of the underlying model. Participants then used think-aloud protocol to sketch and annotate the way that they intuitively envisioned an interface supporting the descriptions they had just read. It was critical to our approach that the detailed description in the introduction contained no leading or suggestive phrasing to bias this process. These sketches were explored in an unstructured interview seeking to understand their motivation and to expand upon their ideas. Following this, they were shown our own candidate designs and engaged in discussion about their interpretation, as well as the advantages and disadvantages of each approach.

Results & Discussion The experiment was run with three participants, all with previous experience in authoring IDN. In participants' own designs, all intuitively defaulted to graph-based systems with nodes connected with lines. Another trend between all participants was the use of an outliner panel on the left of the interface to show hierarchical relationships to the user. Property editing was implemented either as a popup window or a sliding panel, both being used to allow for non-intrusive editing in a way that did not permanently take up screen real-estate. All participants highlighted the importance in their approaches of using distinct shapes and colors to differentiate between elements of the model, with the intention being to aid quick identification.

Our first design was described as in intuitive and fluid due to the familiarity of the flowchart-like visuals. However, there was mixed feedback about how the design provides hierarchical context to the user. A suggested solution was the addition of an outliner, something that all participants had in their own designs.

Our second design was preferred by all participants. All participants spoke positively about the inclusion of an outliner due to its ability to provide an overview of the story and given context as to the current position when editing. The most discussed part of the design was the multilayer 'artboard' system which provides a window to see the contents of a node without explicitly travers-

ing into it. This was praised for providing extra context and therefore a better understanding of the story at a given point without the need for further traversal.

Taking into account the feedback from participants, it was decided that the second candidate design was to be chosen, with feedback from participants being considered for inclusion in the design.

2.3 Refine

The final phase is *Refine* which involves taking the chosen candidate design and feedback from the *Discover* phase and information from the *Research* phase and using them to refine the prototype. As the chosen candidate design is an approach rather than a prototype, it first needs to be expanded upon, which involves enhancing its features and visual quality, transitioning into a high-fidelity mockup with interactivity. In our case, features were added inline with user feedback, constraints, and requirements outlined in previous phases. A notebook was created alongside the new prototype to have a clear understanding of what everything should do, which sometimes is unclear due to limited capability of prototypes. Examples within the prototype were also contextualized to reflect an actual game story rather than placeholder content. The methodology used in this process is based upon the philosophy of the RITE method [11], which positions itself as a discount usability test, and has extensive use in the games industry [10]. The goal is to produce a prototype with a refined user experience by detecting and fixing miscommunications and misunderstandings that participants encounter.

Methodology The participant demographic was students of IDN who have any level of experience with existing authoring tools. Participants, in individual moderated sessions, were shown a document that covered a summary of the persona and an overview of the details about the tool. Following this, they were walked through three short videos from *Mass Effect*[3], which were simplified and recreated in the prototype. A total of 36 tasks were created based on mandatory functionality that users should be able to perform in the tool, mostly phased as 'show me how' or 'walk me through' due to limited interactivity of prototypes. Participants completed each task using the prototype and think-aloud protocol. This was semi-structured in that further questioning could take place based on how the participant responds. If participants require minimal assistance when completing a task (clarification of behavior, pointed in a broad direction) then the RITE spreadsheet is marked with **X** representing error as miscommunication of a feature. If they are unable to continue without assistance, a **Z** is marked, representing failure as a misunderstanding of a feature. If a mark is made, further questions were asked to identify the cause which aids generating solutions. For each task, the participant's expected solution was compared with the intended solution of the design. Participant answers were rephrased and repeated back to them to resolve potential ambiguity, allow them to expand upon their answer, and to give time to take more accurate notes [2]. Tweaks and fixes were made to the prototype between participant sessions based on RITE spreadsheet entries,

[3] Mass Effect. BioWare, 2007.

Task	Problem Description	P1	P2	P3	P4	P5	P6	P7	Fix 1	Fix 2	Fix 3
T4	Couldn't figure out how to create a Frame.			X					Note 1		
T8	Struggled to identify the nodes in the Canvas.		X						Note 2		
T6	Thought that double clicking a node edited its label.		X	X					Note 3		
T10	Unable to navigate into a node in Artboard mode.	X							Note 4		
T13	Unable to add Node Templates to the Canvas.	X							Note 5		
T13	Unable to create Node Templates from the editor.	X							Note 6		
T17	Unable to add links to nodes.		X	Z					Note 7		
T28	Scripts window contained scripts unclear.	X	X	X					Note 8		
T32	Couldn't tell that colored variables relate to the outputs.		Z								
T33	Struggled with states in the Simulation Mode.		X	X					Note 9	Note 10	Note 11

Fig. 1. RITE spreadsheet showing changes due to errors (**X**) and failures (**Z**). Columns P1–P7 are participants. **Note 1:** Added a Frame icon to the toolbar. **Note 2:** Added 'Main' Group to the Outliner. **Note 3:** Double clicking a node edits the label. **Note 4:** Replaced expand icon and darkened color. **Note 5:** Changed '+ Node Template' to 'Insert Node Template', moved buttons to right, darkened text. **Note 6:** Added labeled headers to lists in all Editor panels. **Note 7:** Connection panels animate a + button. **Note 8:** Added a label to the left of the script dropdown and hidden the help text in a popup. **Note 9:** Added an explicit save button and moved delete behavior into a popover no longer requiring applying states to delete them. **Note 10:** Renamed delete button to read 'Delete…'. **Note 11:** Renamed delete button to read 'Manage'.

feedback from participants, and moderator observations. Logs and copies of the prototype were stored per participant to help track change over time.

Results & Discussion Seven participants took part in the experiment. Each participant interacted with an *iPad Pro* acting as a real-time preview of the prototype, controlled by the moderator. Fig. 1 shows the problems that participants P1–P7 encountered and changes made to mitigate them. If a cell is empty, the problem was not encountered by the participant. Shaded cells represent a corresponding change being used. A summary of key changes follows with a sample of the final refined interface shown in Fig. 2.

Artboard button In T10, P1 incorrectly assumed the icon button on Artboards was for closing and only clicked after guessing. The icon was updated to be less ambiguous and no others experienced troubles.

Node creation Creating nodes, tested in T3, was done via the Canvas' context menu. P3 suggested adding physical buttons, which was implemented by adding an array of icon buttons to the Toolbar. These buttons could be clicked on or dragged from into the Canvas, with their different interaction method being distinguished by dotted outlines. All following participants used the buttons over the context menu, and all correctly identified the drag-drop behavior.

Frames Frames, tested in T4, were originally created using the Canvas' context menu and didn't take into account selections. P1 suggested autofitting to selection, which was added to the prototype and the task text updated accordingly. P3 required assistance with creation but no change was made as their thought process greatly increased complexity and was contradictory to estab-

lished paradigms. P4 suggested adding a button for Frames to the toolbar, after which all participants used this button over the context menu.

Adding links Adding links to nodes could originally be done only through a context menu, and was tested in T17. P2 and P3 struggled with adding links. A fix was introduced to reveal a hidden '+' button on the node's connection panel upon hover. All following participants defaulted to the new hover functionality.

Outliner The Outliner originally displayed nodes made by the user only. In T8, participants had to identify types of nodes in the Canvas, with P2 using the Outliner hierarchy to help them, but incorrectly identified some. To help distinguish between the types, a parent 'Main Group' was added to the Outliner. After adding this extra context, participants were able to identify all types without trouble, using the Outliner as a reference.

Node Templates Creation and insertion of Node Templates was refined in tasks 13 and 14. The insert button, originally labeled '+ Node Template', was identified by P1 as creation. The label was changed to 'Insert Node Template' and a context menu alternative was added to the Canvas. After these changes, no participants had trouble inserting them. Creation was done in an editor window using a context menu on an item list. This was problematic for P1 as the purpose of the list was unclear. The solution involved adding labeled headers to all item lists, and as a suggestion from P2, adding an explicit '+' button to the headers as an alternative to context menus. The labeled headers were then added to every list for all editor windows. After these changes, no participants had trouble creating elements, and most defaulted to the '+' buttons in the headers.

Node properties Editing of node properties, refined in T6, was originally done by double clicking a node to show a temporary floating window. P1 suggested that the popup should be detachable, remaining permanent unless closed. P2 and P3 instead expected double clicking to edit a node's label in-place, so the behavior was changed and opening properties was demoted to the context menu.

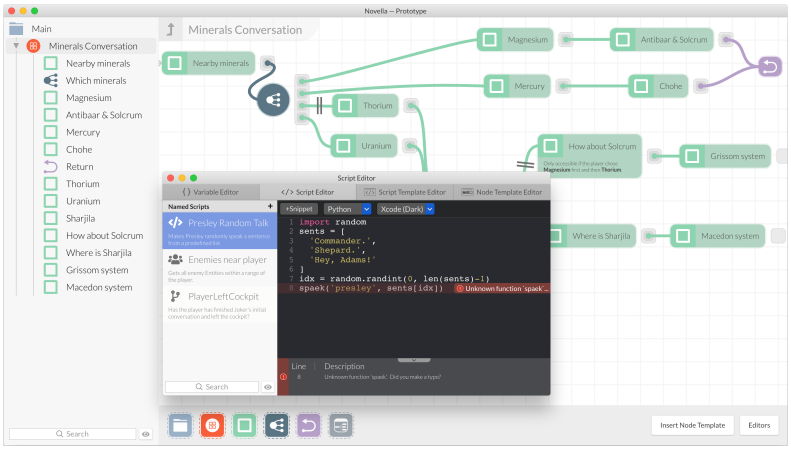

Fig. 2. A snapshot of the refined authoring tool design with an editor window open.

Half of the remaining participants double clicked the node to edit its label, with the other half preferring to use the properties panel.

Scripts All script editor panels have a collapsible error list, looked at in T23. P1 suggested that the clarity could be increased by adding a colored bar and icon to reinforce that it shows errors. After this change, all participants identified its function correctly. T28 looked at a node's custom scripts, which are edited in a dedicated window for that node. P1–P3 were unable to change which script was being edited without assistance. This was fixed by better labeling the dropdown used to select scripts, and by expanding upon yet counter-intuitively hiding the help text behind an info button. With these changes, all following participants read the help text and had no trouble selecting and assigning custom scripts.

Simulator Variables The Simulation window highlights variables that impacts outputs or sibling triggers, and was tested in T32. P2 was unable to identify this relationship, but upon discussion they decided it was misinterpretation of colors. It was decided to not make changes until further occurrences, of which none happened, and as such this was treated as an outlier.

Simulator Variable States T32 had participants complete a number of steps relating to variable state management in the Simulator window using a small desktop application that mimicked the states panel in the prototype. Initially, this was done with a single editable combobox and delete button, with deletion requiring assignment of a state first. P2 and P3 found the behaviors confusing. With their feedback, an explicit save button was added (although placebo, as it functioned identically to submitting in the combobox) and deleting was changed to show a popup to select what to delete rather than deleting the current state. All following participants were able to complete the tasks, even though the underlying functionality didn't change. However, P4 still hesitated with deleting, and after discussion, the delete button was suffixed with ellipsis to suggest further action. P5 and P6 hesitated less, so the label was changed one last time to read 'Manage'. This change appeared successful, as P7 did not hesitate at all, and the terminology used allows for further expansion of state management rather than only deleting.

Recordings After P6, a Recordings feature was added to the Simulation window without disturbance of other features to allow capture and playback of story traversals to better support users in their testing and to reduce human error. Its design largely copies that of variable states, which had already been greatly refined. Corresponding tasks to test this were also added. The RITE method is flexible enough to allow for modifications like this, as long as there is a followup to the changes. P7, who went through the new tasks, had no trouble explaining and operating the feature, and explicitly commented upon its similarity to variable states, saying that they found this intuitive as a result.

3 Conclusions

In this work we have identified a gap in existing authoring tool research concerning formal UX design pipelines. We present our own pipeline for authoring

tool UX design and show through a case study of our own tool design how it has helped us to refined the UX. Following our design pipeline will cover identification of potential users and requirements of the tool, exploration and ideation of early designs, and refinement of the designs to be ready for implementation, at all stages considering the user. We demonstrate the potential of the pipeline by walking through our own application of it in the creation of our own authoring tool design. We were able to identify, report on, and fix usability problems that we otherwise would not have noticed, especially if creating the design without involving users. For example, the Scripts window in our tool is core to the model, but the RITE process demonstrated that users found the initial design confusing. Without this we may have mistakenly assumed that this was clear.

Our contributed design pipeline is composed of various user research methodologies that we believe are best suited to our needs, yet remain the most flexible for the design of other future authoring tools. However, as established earlier the specific methodologies used will depend on the context of the tool being developed. Our intent with this contribution is not to insist on a single UX design pipeline but rather to confirm their value to IDN authoring tools, and how one might be developed in a similar context.

References

1. Bernstein, M.: Storyspace 1. In: Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia. pp. 172–181. ACM (2002)
2. Bromley, S.: Interviewing Players. In: Games User Research. Oxford University Press (2018)
3. Desurvire, H., El-Nasr, M.S.: Methods for Game User Research: Studying Player Behavior to Enhance Game Design. IEEE CG&A **33**(4), 82–87 (2013)
4. Green, D., Hargood, C., Charles, F.: Contemporary Issues in Interactive Storytelling Authoring Systems. In: ICIDS. pp. 501–513. Springer (2018)
5. Green, D., Hargood, C., Charles, F.: Novella 2.0: A Hypertextual Architecture for Interactive Narrative in Games. In: Proceedings of the 30th ACM Conference on Hypertext and Social Media. HT '19, ACM (2019)
6. Hamm, M.J.: Wireframing Essentials. Packt Publishing Ltd (2014)
7. Heussner, T., Finley, T.K., Hepler, J.B., Lemay, A.: The Game Narrative Toolbox. CRC Press (2015)
8. Koenitz, H.: Three Questions Concerning Authoring Tools. In: AIS, ICIDS (2017)
9. Martens, C., Iqbal, O.: Villanelle: An Authoring Tool for Autonomous Characters in Interactive Fiction. In: Interactive Storytelling. pp. 290–303. Springer (2019)
10. Medlock, M.: The Rapid Iterative Test and Evaluation Method (RITE). In: Games User Research. Oxford University Press (2018)
11. Medlock, M., Wixon, D., Terrano, M., Romero, R., Fulton, B.: Using the RITE Method to Improve Products: A Definition and a Case Study. UPA **51** (2002)
12. Millard, D.E., Hargood, C., Howard, Y., Packer, H.: The StoryPlaces Authoring Tool: Pattern Centric Authoring. In: AIS, ICIDS (2017)
13. Muller, M., Kuhn, S.: Participatory Design. Communications of the ACM **36**(6), 24–28 (1993)
14. Spierling, U., Szilas, N.: Authoring Issues beyond Tools. In: Interactive Storytelling. pp. 50–61. Lecture Notes in Computer Science, Springer (2009)