[Draw your reader in with an engaging abstract. It is typically a short summary of the document. When you’re ready to add your content, just click here and start typing.]

Summary of Background Reading

This is the summary of my background reading.

Project: **Hypertext Game Project**

Student Name: **Richard Lowe**

Registration Number: **1804170**

Supervisor: **Dr. Richard Bartle**

Second Assessor: **Dr. John O'Hara**

Degree Course: **Computer Games BSc**

# The strategy I used for my background reading

This project (To create a hypertext game creation tool, and then to use this tool to create a hypertext game), being a more software-based project, appeared to facilitate an approach focused on researching existing hypertext game creation tools, and then using this insight to guide the design of my hypertext game tool. I spent the first half of August looking at tools, before spending the latter half of August creating a prototype of my tool. Then, during September, I looked at some academic literature on the topic of hypertext games, as well as some examples of hypertext games, to guide the creation of the game, and to guide the refinement of the tool.

# Researching existing hypertext game creation tools

I shall first explain my methodology, before going into the specifics about what I found out about each tool. I decided that I would attempt to answer the questions of ‘How does it work?’, ’How do I use it?’, ‘What options does it give me?’, ‘What features does it have?’, ‘What features is it missing?’, and ‘How does it work?’. I initially found five hypertext game/interactive fiction creation tools to look at: two of which I was aware of already (Twine[1] and Ren’Py[2]), two of which I’ve forgotten how I found out about (Storyspace[3] and TADS [4]), and one which only existed in specification form, which Dr. Bartle sent me a copy of, along with a link to a demo of what it might have produced (eHyperTool [5] [6]). Whilst researching Twine, I also found out about Twee2 [7] and the ‘Treaty of Babel’ standard for archiving works of Interactive Fiction [8]. Once I had finished researching the first five tools, Dr. Bartle also told me that it would be worth investigating Inform [9], Inklewriter [10], ChoiceScript [11], Quest [12], Squiffy [13], and Undum [14].

## Researching Storyspace

I started off by looking at Storyspace [3], mainly because I was not sure if I would be able to get much information about it anyway, as I knew I would not be able to use it myself, considering that it costs $149, and only works on Mac computers (I do not own a Mac). Storyspace appears to have been one of the first hypertext game creation tool to have been made, and is a somewhat complex tool, supporting text, graphics, audio, and video [15], a user interface based on ‘topographic writing’ (presenting the story as a network of linked ‘writing spaces’ to the writer), implementing features such as ‘guard fields’ to impose some limitations on how the reader can traverse the output hypertext, [16] and it has an architecture where the ‘links’ between ‘writing spaces’ are distinct objects themselves instead of being part of the ‘writing spaces’ they are linked to. Storyspace 3 was written in C++, making heavy use of pointers, and the tool has a rather complex syntax for guard conditions for links [17]. However, it does have some significant drawbacks: the complex architecture of the system means that there’s no way for the games it produces to be playable on the web, and the barriers to entry from it being proprietary/unusable on anything but a Mac, has significantly diminished its appeal, especially compared to modern tools which do not have these limitations [16].

I was able to find a guide to using Storyspace, which did allow me to gain some insight about how to use it. In short, you can double-click the ‘map’ view to create ‘writing spaces’, click on those to open the dialog to edit them, and drag your cursor from one ‘writing space’ to another to create a link between them (which can be edited by clicking on the link). Your work-in-progress hypertexts are saved in an XML-based format (.tbx) [18], and it appears that, to play them, you need to open the .tbx file with the Storyspace Reader application (which, unfortunately, also appears to be Mac-only) [3].

## Researching TADS

TADS (Text Adventure Development System), upon closer inspection, appeared to be a system for developing text adventure games (a la *Zork*) instead of hypertext fiction games [4], so I chose to keep my research on this tool rather brief. In short, it’s an object-oriented code-based tool for producing text adventures, where you write .t source code to define the objects for the game (‘rooms’, ’things’, ’NPCs’, etc.), optionally utilizing certain premade libraries to simplify the process of setting everything up for the game, and then outputting the playable game as a .t3 file [19]. The system itself is written in C++, and the .t3 games are played on the ‘T3 Virtual Machine’ (also written in C++), which is similar in concept to Java’s JVM, but with a completely different object model, uses runtime typing, and has options for the user such as undo/save/load. It is possible for a .t3 game to be played on a web browser, but doing so requires a server-side instance of the T3VM to be running on the website hosting the web-playable .t3 game [20]. However, due to TADS not entirely being relevant to the task at hand, I decided to not spend too long looking into it.

## Researching eHyperTool

Unlike all the other tools mentioned here, this one wasn’t so much a ‘tool’, more a specification [5] and a demonstration [6]. The specification states that this would have been a cloud-based (in the sense of ‘hosted on a private server, accessed via browser, and you need to log in to use it’) GUI for creating hypertext games, displaying games as a network of nodes (supporting text, images, and audio), linked by ‘options’ at the end of the node text (with each option having its own text, timer, guard conditions etc), allowing writers to move between nodes either by clicking them on the overview network, or by navigating to them via a tab, also allowing writers to mark the status of nodes (how complete they are), add notes to nodes, and keeping a record of when a node was last edited (and by whom). It even supports some meta content, such as a title/author name/cover art etc. The specification also states that it would support several output formats: plain text/book, raw HTML, HTML with extra functionality provided by JavaScript, and a format that could make it playable on Kindle. However, it does appear to be rather inflexible in terms of formatting, requiring all options to explicitly be after all the node contents [5]. The output demo is implemented in a way which uses a separate .html page for each node, and takes advantage of HTTP GET requests to keep track of what choices the user has made, but obfuscated/decoded in a way which keeps the length of the URL rather short. However, this approach does have the disadvantage of requiring a lot of .html files to be made/stored/downloaded for the game to work [6].

## Researching Twine

Twine is a GUI for creating hypertext games, which can be used either in browser on the Twine website (running client-side), or downloaded as a standalone executable, either option still working the same way as each other. Basically, it shows the story as a network of linked ‘passages’ (where links are defined in the passage content, via a certain syntax), more ‘passages’ can be added either by pressing an ‘add’ button or by creating a new link within the contents of a passage, and the game can be tested at any point in time. This is because a twine game is stored in a single .html file (with no distinction between a work-in-progress twine game and a complete twine game), containing all the HTML, JavaScript, and CSS needed to run the game, along with the contents of the game itself [1] [21]. The system offers a choice of ‘story formats’, each with their own particular syntaxes and functionality, such as Harlowe (the default format, which the Twine GUI supports syntax highlighting for) [22], SugarCube [23], Snowman [24], Chapbook [25], and PaperThin (a plaintext format intended for proofreading purposes) [26]. However, some common pieces of functionality for each of these formats include variables, some method of saving progress, and conditional statements. Additionally, as a twine game is stored in a single .html file, if you want to host a twine game on a web server, all you need to worry about is the single .html file. It does still have a couple of drawbacks, however. Twine has no native support for images and audio; however, you can still reference external resources like you would in any normal website. Additionally, the game is effectively interpreted instead of compiled, meaning that, if there are any syntax errors within the game, the writer will need to look through the game themselves in order to ensure that it works correctly [26].

## Researching Ren’Py

Ren’Py is a python-based visual novel engine. Whilst hypertext games and visual novels aren’t exactly the same thing, seeing as visual novels still tend to have a similar basic choice-based branching narrative structure to hypertext games. Ren’Py games are written as directories of .rpy files (An object-oriented python-based language), containing declarations both for the text content, and also for manipulating the sprites/background images/GUI/etc that constitute the ‘visual’ part of the ‘visual novel’ (along with the audio as well). It builds and output games in executable format for a range of platforms (including HTML5 format). It also has some support for variables etc. However, one limitation of this is that it’s entirely code/IDE; there’s no visual editor of any variety, which might be off-putting to any potential casual users [2].

## Researching the Treaty of Babel

I found out about the Treaty of Babel whilst researching Twine, after noticing a rather curious piece of metadata within Twine games labelled ‘IFID’. In short, this is a standard to facilitate archiving works of interactive fiction (not just of the ‘hypertext game’ variety), by including metadata within the file of the work to identify it, in a manner which can be easily read via a command line C utility. The bare minimum pieces of metadata required by the treaty are the game title, the author’s name, and an ‘IFID’ (a unique identifier for a game, which coincidentally happens to be in the same format as a UUID). This standard is mainly used for purposes of archiving works of interactive fiction on the Interactive Fiction Database (IFDB), however, if I can make the outputs my system comply with it, it might make my system a little bit more attractive to use [8].

## Researching Inform

Inform, like TADS, is a system used to create text adventure games, not hypertext games, so I didn’t look too deeply into this system. The games for this system are written by first ‘creating the world’ (essentially written in the form of assertions about the starting state of everything in it), and then by ‘specifying the rules of play’ (defining the ‘rules’, written as instructions, which define the behaviours of objects, and defining the ‘actions’, written as participles, which the player can perform). Whilst all of that isn’t entirely relevant to the task at hand, one aspect of Inform that is of interest is how the writer is prompted to define where to save the game/what the title will be/what their (pen)name is when creating a project (with the option to change it later), ensuring that these pieces of bibliographic metadata will have been defined [9].

## Researching inklewriter

inklewriter, like eHyperTool, is a cloud-based server-side GUI for writing hypertext games, hosted by inkle on the inklestudios.com website, requiring users to create an account on the website to save their work (and only saves work on the server, not locally). Each node is referred to as a ‘stitch’, with links to other ‘stitches’ being at the end of a ‘stitch’. Variables are supported, and links may have guard conditions, requiring certain stitches to be (un)visited, or requiring conditional statements involving variables to be met. Additionally, you can define ‘sections’ (for ease of navigating through sections of the story when editing it), there are some formatting options (but limited to inserting images and using bold/italic text), and you may navigate through the story you are editing either in a view that resembles the structure of the final output, as a simple list of stitches, or via a map overview. You need to create an account on the website to save your game, or to export it (either as a playable game on the server, in JSON form to copy it into another game on the server, or in .ink format for use with ‘ink’) [10].

ink is an alternative to inklewriter (also by inkle), albeit in script/IDE form (not GUI form) and is entirely client-side, but it still allows writers to produce games with roughly the same functionality as inklewriter games, despite not having the same tools inklewriter provides (but having a few more options, such as styling options). It is possible to transfer a game between inklewriter and ink, however, it is a bit complicated. To transfer it from inklewriter to ink, you can just use the previously-mentioned .ink export option, but, to transfer from ink to inklewriter, you need to save your .ink game in .json format, open the .json file, copy the full json string, then paste it into the ‘import from json’ input on inklewriter (after logging in to inklewriter) [27].

## Researching ChoiceScript

ChoiceScript is a scripting-based system for creating hypertext games (with no official GUI or IDE, but there is an unofficial IDE called ‘Chronicler’), however, even though you need to download it and run it locally, it’s written as a server-side Node.js utility, meaning that, in order to play/test any games written with it, you need to launch a ChoiceScript server using a batch file supplied with the tool, and keep it running whilst playing your games. This is probably due to the creators of ChoiceScript, Choice of Games, commercially distributing ChoiceScript games, and I suspect that the serverside nature of this system is intended for DRM purposes. Games are written in .txt files in a directory holding all the other files needed for the game to operate, following the ChoiceScript language. One of these .txt files, startup.txt, is used for declaring the overall structure of the game: defining all of the variables for the game, referencing all the other .txt files which constitute the ‘scenes’ of the game, and holds all the important metadata for the game. Unlike most other tools, ChoiceScript does provide some automatic testing functionality; ‘Quicktest’ (a code coverage test which attempts to exhaustively test every single option/conditional statement, returning information about unreached code/syntax errors), and ‘Randomtest’ (which tests the game by randomly going through it a given number of times (although it will start from a constant seed every time for consistency, unless explicitly given a new seed to start from), reporting how frequently each line was encountered after all the tests, and reporting syntax errors etc) [11]. Whilst the testing functionality it provides is nice, I feel that the extra hassle of the server-side architecture (especially considering that this is supposed to be used to make games locally) could deter many casual users from trying to use this system, so I do not intend to use a similar architecture for my tool.

## Researching Quest

Quest is, first and foremost, a tool for creating text adventure games, however, upon opening it up, it offers users a choice between producing a text adventure or a ‘gamebook’ (essentially a hypertext game). It’s an object-oriented tool, and saves games in ‘.aslx’ format (an XML-based format, declaring all of the objects in the game world, facilitating the use of this system as a text adventure system) [12]. However, despite giving writers the option to create ‘gamebooks’ with this system, the documentation for ‘gamebooks’ is hidden in the documentation, only linked within part 2 of the tutorial, and starts by actively discouraging users from attempting to make a gamebook with Quest, encouraging writers to use Squiffy instead for this purpose [28]. The gamebook implementation within Quest still has the same basic structure as the text adventures, but this significantly limits the functionality the writer has access to, so I decided to take the advice of the documentation and look at Squiffy instead.

## Researching Squiffy

Squiffy is a script/IDE-based, client-side tool for producing hypertext games, with the nodes being ‘sections’, which may contain ‘passages’ (extra optional text that still remains in the same section). The IDE allows you to write .squiffy code, offering a testing preview of the game you are writing to the right of the code input area (allowing you to easily test it by pressing the ‘run’ button at the top, refreshing this preview to reflect the changes you’ve made to the squiffy code), allowing writers to easily navigate to sections/passages via some drop-down option menus above the code panel, and incorporating syntax highlighting to make it easy to tell if your code is valid or not. It supports links at arbitrary positions within the text content, variables, conditional statements/interactive elements that are parsed at runtime, and HTML/markdown formatting of the text contents. However, you need to use the command-line version of Squiffy to import external files, and, if you want to edit the CSS/HTML page holding the game, you need to edit them manually in the game output folder. Upon building a squiffy game, it’s output in a folder containing copies of a premade index.html, style.css, and jquery.min.js file, along with a story.js file; this contains a prewritten definition of a ‘squiffy’ object (essentially defining all the functionality the system has), followed by definitions of the game content (built from the .squiffy file). This is honestly a rather nice method of exporting a working game, and I’m probably going to use a similar approach to this for mine (as in the ‘define premade components, and export copies of those, but build the definitions of the game contents dynamically from the script’ approach) [13] [29].

## Researching Undum

Undum isn’t really a tool, more so a JavaScript framework for creating hypertext games. Whilst Squiffy outputs a JavaScript-based game engine along with declarations of the game contents from the .squiffy file, Undum skips the intermediate scripting stage and basically requires the writer to write the declarations of the game content directly in JavaScript. The framework provides support for things such as variables (albeit only numeric variables), animations, various formatting options, etc. The nature of it, being entirely JavaScript, probably does lower the barrier to entry for people who are already familiar with JavaScript (seeing as they don’t need to learn a whole new syntax), however, this probably doesn’t appeal to casual users without any JavaScript knowledge, partially because they’re expected to write raw JavaScript, and there’s no form of IDE/GUI to help them out. However, the git repo for Undum does provide some template Undum code, which might simplify the process [14].

## Researching Twee2

Twee2 is the last tool I researched. In short, it’s like Twine, but entirely script-based/command line instead, geared towards power users instead of casual users. Users write their game as .twee code, then use the command-line Twee2 tool to convert the .twee code into a Twine game. Twee2 can also decompile Twine games into raw .twee code, meaning that it is theoretically possible to create a game using Twine and Twee2, however, this decompilation functionality currently doesn’t work on Windows systems, so not all users will be able to take advantage of this. Twee2 supports all the same story formats as Twine, and has the same linking syntax as Twine, but the passage declaration syntax resembles that of Squiffy. The lack of any sort of GUI/IDE can be excused, seeing as it’s explicitly presented as ‘Twine for power users’, but, as a standalone tool, this is a bit of a disadvantage [7].

# Creating a prototype version of my tool

This is probably outside the scope of what this report is supposed to cover, however, I started work on my tool shortly after I looked at those other tools. Firstly, I made a ‘wishlist’ for functionality that the program might need, before working out how feasible/unfeasible they would be, before working out the core functionality I would need to include. I decided that a good starting point would be a code-based tool where users write declarations for the game content in an intermediate language, which would then be parsed into a playable client-side HTML/JavaScript game (in the same vein as ink, Squiffy, and twee2), and, after the MVP stage, extending this with a GUI which can produce the intermediate language in a more user-friendly way (presenting the WIP games in a topological structure similar to that of Storyspace, eHyperTool, and Twine, but saving/loading from the intermediate format). The games it will output shall consist of a prewritten index.html page (as an interface), a prewritten JavaScript engine, and some JavaScript constructed from the contents of the parsed intermediate code to provide definitions of the game contents to the engine. I started off by making a prototype version of the output, then a rather procedural implementation of the parser, before refactoring the parser to be more object-oriented (because I’d need to perform that refactoring at some point, so I decided to do it sooner rather than later). This topic will be covered in further detail in my challenge week presentation.

# Researching literature on the topic of hypertext games

At the start of September, I realized that I had forgotten to look at any academic literature on the subject of hypertext games, or to properly play/take notes on some hypertext games. Dr. Bartle sent me .pdf copies of a couple of articles he had on the topic (*H. Koenitz, "What Game Narrative Are We Talking About? An Ontological Mapping of the Foundational Canon of Interactive Narrative Forms,"* [30] and *H. K. Rustad, "A Four-Sided Model for Reading Hypertext Fiction,"* [31]), and suggested that I read *E. J. Aarseth, Cybertext: Perspectives on Ergodic Literature* [32]. I found several other articles myself via Google Scholar, after searching up works on the topic of ‘Hypertext Games’.

## Notes from *H. Koenitz, "What Game Narrative Are We Talking About? An Ontological Mapping of the Foundational Canon of Interactive Narrative Forms,”*

This article was more of a piece of analysis about existing articles on the subject of interactive narrative, specifically on what the authors of these articles treat as ‘narrative’ (due to a lack of a consistent definition for it across academia). Koenitz identifies two main definitions of narrative by mapping these articles on scales of how they consider the narrative to be affected by ‘Media Specificity’ and ‘User Agency’; these two definitions, coming from the two clusters on this mapping, are “high specificity and user agency” (coming from a cluster ‘at the maximum value of both dimensions’), with the other being ‘low on media specificity and high on user agency’ [30]. When I looked at this article, I wasn’t sure if there was any information from this article I could apply to my project, however, in hindsight, I probably should have tried looking at the other pieces of literature which this article was analysing. Unfortunately, I don’t think I have enough time left in the summer break to look at those other articles, so I’ll probably have to just record that as an identified shortcoming with my research.

## Notes from *H. K. Rustad, "A Four-Sided Model for Reading Hypertext Fiction,"*

In this article, Rustad identifies four ‘modes’ of reading works of hypertext fiction, discusses how each of these can be facilitated in a game, and draws comparisons to player archetypes. The first mode, the ‘semantic mode of reading’, is characterised by a ‘quest for meaning’, facilitated by games which invite players to exhaustively search every single node in them, appealing towards ‘achievers’. The second mode, the 'explorative mode of reading’, is characterized by the reader trying ‘to gain experiences through exploring the hypertext games’, facilitated by games which ‘encourages or opens up for random exploration’, appealing to ‘explorers’. The ‘self-reflective’ mode of reading is characterized by the reader becoming ‘aware of his own codes, his own experiences and expectations’ facilitated by giving ‘the reader the opportunity to play a role’, and this role-playing interactive aspect links to the behaviours of ‘socializers’. Finally, ‘absorption’ is characterized by keeping ‘the reader in a state of confusion’, facilitated by overwhelming the reader, such that they have 'no other option than to slip into an in-between position'. The outcome of this mode is ‘similar with the outcome enforced by killers’[31]. Reading this article has given me some insight into different reading styles for hypertext fiction, who these modes appeal to, and how to invoke these modes through the design of a hypertext game; I will be taking this into account when creating the hypertext game with my system for the final deadline/submission.

## Notes from *E. J. Aarseth, Cybertext: Perspectives on Ergodic Literature*

I will admit that I was unable to fully read this book, because I could only find a preview of it (only containing the introduction) online, I did not have access to a physical copy, and I did not want to resort to legally dubious methods of viewing the full text. However, I still was able to get some information from the introduction. The main concepts Aarseth discusses in this book are those of ‘Cybertext’ and ‘Ergodic Literature’. Cybertext, in short, considers 'the intricacies of the medium as an integral part of the literary exchange' and treats the 'consumer, or user, of the text, as a more integrated figure', and 'must contain some kind of textual feedback loop'. This definition doesn’t restrict itself to electronic texts, as non-computerized texts can have cybertextual features as well. Ergodic Literature is the concept of literature where ‘nontrivial effort is required to allow the reader to traverse the text’; for example, requiring a ‘specialized ritual of perusal’, incorporating choices, and basically requiring the reader to actively take part in it. The reader of a cybertext is to be considered a player, not a mere spectator, and has a somewhat interventionist role in the narrative[32]. This has given me a few more ideas for what the game I’ll need to produce should involve to properly be a hypertext ‘game’, namely making the reader into a ‘player’, with the game having a ‘textual feedback loop’, and ensuring that there are consequences for the reader’s choices.

## Notes from *D. E. Millard, "Games/Hypertext,"*

This article is mostly about the relationship between narrative games and literary hypertext, and Millard attempts to analyse this relationship. His conclusion was that 'literary hypertexts appear to be a sub-type of narrative games', but 'narrative games are fundamentally hypertextual', and that it is worth 'linking the worlds of hypertext and game design... apply known hypertext theory to new and modern contexts, and build a better theoretical understanding of electronic literature and games'[33]. In hindsight, this article probably isn’t entirely relevant to the task at hand (being about the value of linking hypertext theory and game design), however, it does still point out that it might be worth applying some game design methodology to hypertext games, so I will try to bear that in mind when creating the hypertext game itself.

## Notes from *S. Kitromili, J. Jordan and D. E. Millard, "What Authors Think about Hypertext Authoring,"*

This article is about the opinions of authors towards hypertext authoring tools, identifying the steps involved in the authoring process, identifying problems which authors encounter during this process, and suggests some solutions for these problems. The steps, in order, are ‘ideation’ (coming up with ideas), ‘training/support’ (where authors learn how to use the tool), ‘planning’, ‘visualising/structuring’ (working out the structure of the game), ‘writing’, ‘editing’, ‘compiling/testing’, and ‘publishing’. The problems fall into three umbrella categories; ‘pre-production’, ‘production’, and ‘post-production’. ‘Pre-production’ problems stem from ‘misalignment’ between the user and the tool (mismatch between user intent and tool functionality), and poor documentation (mainly from when a user doesn’t know how to do something that is possible, or doesn’t know whether or not something is possible). ‘Production’ problems stem from ‘complexity’ (difficulties in using the tool as the game increases in scale), and from the ‘programming environment’ itself (poorly implemented/missing usability features). Finally, ‘post-production’ problems relate to the lifecycle of a game after it’s been completed, considering ‘distribution’, ‘profitability’, ‘platform support’, ‘curation’, and ‘maintenance’ issues[34].

I will be taking these identified problems into account when creating my tool, and, even though I know I won’t be able to conclusively address all of the issues, I shall attempt to address as many as possible. I shall attempt to avoid most of the ‘pre-production’ issues by making the documentation as clear as possible for my tool; stating what functionality it has, what functionality it doesn’t have, and, for the functionality it has, stating exactly how to utilize that functionality. ‘Production’ will be a bit more difficult; I’m not sure how to best address the ‘complexity’ problems, however, I suppose the problems involving keeping track of variables could be avoided by requiring variables to be explicitly declared in advance (which, in turn, could lend itself towards some automated error detection with variables, addressing that ‘programming environment’ issue), and some means for allowing authors to take notes will probably be appreciated as well. Finally, for the ‘post-production’ issues, I know I don’t have the resources needed to address ‘distribution’ and ‘profitability’. I can somewhat address ‘curation’ by complying with the Treaty of Babel, ‘maintenance’ might be facilitated by the intermediate language format, and ‘platform compatibility’ should hopefully be facilitated by having the games be simple client-side JavaScript games, playable on browser. This should hopefully make the tool somewhat appealing to use from an author’s perspective.

## Notes from *M.-L. Ryan, "From Narrative Games to Playable Stories: Toward a Poetics of Interactive Narrative,"*

In this article, Ryan defines a ‘narrative game’ as being a game where a ‘story is meant to enhance gameplay’, where ‘narrative meaning is subordinated to the player’s actions’, and ‘the player plays to win, to beat the game, and the story is a lure to the game world’. Ryan also defines a ‘playable story’ as having ‘gameplay… meant to enhance story’, where the ‘player’s actions are subordinated to narrative meaning’, and having ‘no winning or losing: the purpose of the player is not to beat the game, but to observe the evolution of the storyworld’. She also identifies two approaches to designing these interactive narratives; a ‘bottom-up’ approach and a ‘top-down’ approach. The ‘bottom-up’ approach relies on 'emergent systems that create stories during the run of the program', stemming from interactions within a 'world full of things and characters', but this approach tends to have a ‘lack of closure’, and is most suitable for playable stories. The ‘top-down’ approach relies ‘on pre-scripted content’, and ‘is a journey along a path which has already been traced and leads to a fixed destination, or to several’, which can be started ‘from the design of gameplay’, or ‘from a specific storyworld’, and is ‘typical of narrative games’. Finally, she identifies four types of immersion that [35]

## Notes from *M. Bernstein, "On hypertext narrative,"*

[36]

## Notes from *J. Pope, "A Future for Hypertext Fiction,"*

[37]

# Research on existing hypertext games

# Bibliography

|  |  |
| --- | --- |
| [1] | Interactive Fiction Technology Foundation, "Twine / An open-source tool for telling interactive, nonlinear stories," [Online]. Available: https://twinery.org/. [Accessed 5 August 2020]. |
| [2] | T. Rothamel, "The Ren'Py Visual Novel Engine," [Online]. Available: https://www.renpy.org/. [Accessed 6 August 2020]. |
| [3] | Eastgate Systems, Inc., "Storyspace: Storyspace," 2020. [Online]. Available: http://www.eastgate.com/storyspace/. [Accessed 3 August 2020]. |
| [4] | M. J. Roberts, "TADS - the Text Adventure Development System, an Interactive Fiction authoring tool," tads.org, 16 May 2013. [Online]. Available: https://www.tads.org/. [Accessed 4 August 2020]. |
| [5] | Beck & Bartle Limited, *eHyperTool Design Specification Version 1.02 (unpublished),* 2011. |
| [6] | R. Bartle, "Demo 1," February 2011. [Online]. Available: https://www.youhaventlived.com/cbdemo/. [Accessed 4 August 2020]. |
| [7] | D. Q, "Twee2 | Interactive Fiction for Power Users," 2015. [Online]. Available: https://dan-q.github.io/twee2/. [Accessed 17 August 2020]. |
| [8] | Interactive Fiction Technology Foundation, "The Treaty of Babel," 24 October 2014. [Online]. Available: http://babel.ifarchive.org/. [Accessed 10 August 2020]. |
| [9] | G. Nelson, "Inform 7 | Inform is a natural-language-based programming language for writers of interactive fiction.," 2015. [Online]. Available: http://inform7.com/. [Accessed 10 August 2020]. |
| [10] | inkle Ltd., "inklewriter," inkle Ltd., [Online]. Available: https://www.inklestudios.com/inklewriter/. [Accessed 11 August 2020]. |
| [11] | Choice of Games LLC, "Introduction to ChoiceScript - Choice of Games LLC," Choice of Games LLC, 2020. [Online]. Available: https://www.choiceofgames.com/make-your-own-games/choicescript-intro/. [Accessed 11 August 2020]. |
| [12] | textadventures.co.uk, "Quest - Write text adventure games and interactive stories," 2020. [Online]. Available: https://textadventures.co.uk/quest/. [Accessed 11 August 2020]. |
| [13] | textadventures.co.uk, "Squiffy - A simple way to write interactive fiction," 2020. [Online]. Available: https://textadventures.co.uk/squiffy. [Accessed 13 August 2020]. |
| [14] | I. Millington, A. Yakovlev, A. Plotkin, B. Dias, D. Fabulich, D. Eyk, D. Eliseev, I. Narozhny, J. Grams, J. Leinonen, M. N. Tenuis, Selene and Zonnah, "Undum – A client side framework for hypertext interactive fiction games," 2010-2018. [Online]. Available: https://idmillington.github.io/undum/. [Accessed 14 August 2020]. |
| [15] | C. Keep, T. McLaughlin and R. Parmar., "The Electronic Labyrinth," 1993-2001. [Online]. Available: http://www2.iath.virginia.edu/elab/elab.html. [Accessed 3 August 2020]. |
| [16] | B. Barnet, "Machine enhanced (re)minding: the development of storyspace," *Digital Humanities Quaterly,* vol. 6, no. 2, 2012. |
| [17] | M. Bernstein, "Storyspace 3," in *HT '16: Proceedings of the 27th ACM Conference on Hypertext and Social Media*, Halifax, 2016. |
| [18] | H. Oakley, "Getting started with Storyspace 3 - The Eclectic Light Company," 8 December 2015. [Online]. Available: https://eclecticlight.co/2015/12/08/getting-started-with-storyspace-3/. [Accessed 3 August 2020]. |
| [19] | M. J. Roberts and E. L. Stauff, "TADS 3 Library Reference Manual," 5 16 2013. [Online]. Available: https://www.tads.org/t3doc/doc/libref/index.html. [Accessed 4 August 2020]. |
| [20] | M. J. Roberts, S. Breslin, E. Eve, M. Nizette and A. Sewe, "TADS 3 Technical Manual," September 2006. [Online]. Available: https://www.tads.org/t3doc/doc/techman/toc.htm. [Accessed 4 August 2020]. |
| [21] | Interactive Fiction Technology Foundation, "Twine Specifications," 2019 February 2020. [Online]. Available: https://github.com/iftechfoundation/twine-specs. [Accessed 5 August 2020]. |
| [22] | L. Arnott, "Harlowe 3.1.0 Manual," 24 September 2019. [Online]. Available: https://twine2.neocities.org/. [Accessed 5 August 2020]. |
| [23] | T. M. Edwards, "SugarCube," 2020. [Online]. Available: https://www.motoslave.net/sugarcube/2/. [Accessed 5 August 2020]. |
| [24] | D. Cox and C. Klimas, "Snowman 2.0 Documentation," September 2019. [Online]. Available: https://videlais.github.io/snowman/2/. [Accessed 5 August 2020]. |
| [25] | C. Klimas, "Chapbook, a story format for Twine 2," 6 July 2020. [Online]. Available: klembot.github.io https://klembot.github.io/chapbook/. [Accessed 5 August 2020]. |
| [26] | Interactive Fiction Technology Foundation, "Twine Wiki," Interactive Fiction Technology Foundation, 2019. [Online]. Available: https://twinery.org/wiki/start. [Accessed 5 August 2020]. |
| [27] | inkle Ltd., "ink - inkle's narrative scripting language," inkle Ltd., [Online]. Available: https://www.inklestudios.com/ink/. [Accessed 11 August 2020]. |
| [28] | textadventures.co.uk, "Quest 5 - Documentation," textadventures.co.uk, 31 December 2018. [Online]. Available: http://docs.textadventures.co.uk/quest/. [Accessed 11 August 2020]. |
| [29] | textadventures.co.uk, "Squiffy - Documentation," textadventures.co.uk, [Online]. Available: https://docs.textadventures.co.uk/squiffy/. [Accessed 13 August 2020]. |
| [30] | H. Koenitz, "What Game Narrative Are We Talking About? An Ontological Mapping of the Foundational Canon of Interactive Narrative Forms," *Arts,* vol. 7, no. 4, p. 51, 2018. |
| [31] | H. K. Rustad, "A Four-Sided Model for Reading Hypertext Fiction," *Hyperrhiz: New Media Cultures,* vol. 6, 2009. |
| [32] | E. J. Aarseth, Cybertext: Perspectives on Ergodic Literature, Baltimore, MD: Johns Hopkins University Press, 1997. |
| [33] | D. E. Millard, "Games/Hypertext," in *Proceedings of the 31st ACM Conference on Hypertext and Social Media*, Virtual Event, 2020. |
| [34] | S. Kitromili, J. Jordan and D. E. Millard, "What Authors Think about Hypertext Authoring," in *HT '20: Proceedings of the 31st ACM Conference on Hypertext and Social Media*, Virtual Event, 2020. |
| [35] | M.-L. Ryan, "From Narrative Games to Playable Stories: Toward a Poetics of Interactive Narrative," *Storyworlds: A Journal of Narrative Studies,* vol. 1, pp. 43-59, 2009. |
| [36] | M. Bernstein, "On hypertext narrative," in *HT '09: Proceedings of the 20th ACM conference on Hypertext and hypermedia*, Torino, 2009. |
| [37] | J. Pope, "A Future for Hypertext Fiction," *Converg. Int. J. Res. New Media Technol.,* vol. 12, no. 4, pp. 447-465, 2006. |
| [38] | Z. Quinn, P. Lindsey and I. Shankler, "Depression quest: An interactive (non)fiction about living with depression," 14 February 2013. [Online]. Available: http://www.depressionquest.com/. [Accessed 17 September 2020]. |
| [39] | M. Pavić and S. Sofrenovic, "The Glass Snail," Word Circuits, August 2003. [Online]. Available: http://www.wordcircuits.com/gallery/glasssnail/. [Accessed 17 September 2020]. |
| [40] | M. Heyward, "of Day, of Night," Eastgate Systems, Watertown, MA, 2004. |
| [41] | M. Heyward, "Live traversal of ‘of day, of night’ Nov 8, 2019 at WSUV, USA | spaces in between time," 23 December 2019. [Online]. Available: https://www.creativecultural.com/meganheyward/?p=1750. [Accessed 17 September 2020]. |
| [42] | M. Lutz, "the uncle who works for nintendo by ztul," itch.io, 6 December 2014. [Online]. Available: https://ztul.itch.io/the-uncle-who-works-for-nintendo. [Accessed 17 September 2020]. |
| [43] | A. A. Anthropy, "Queers in Love at the End of the World by anna anthropy," itch.io, 12 March 2016. [Online]. Available: https://w.itch.io/end-of-the-world. [Accessed 17 September 2020]. |
| [44] | J. Anderson, "afternoon, a story - Michael Joyce," youtube.com, 26 April 2012. [Online]. Available: https://www.youtube.com/watch?v=djIrHF8S6-Q. [Accessed 17 September 2020]. |
| [45] | T. Memmott, "Lexia to Perplexia," eliterature.org, 2000. [Online]. Available: https://collection.eliterature.org/1/works/memmott\_\_lexia\_to\_perplexia.html. [Accessed 17 September 2020]. |
| [46] | M. Joyce, "afternoon, a story," Eastgate Systems, Watertown, MA, 1987. |
| [47] | N. Gibbins, "Telling Tales - EdShare Southampton," 6 November 2019. [Online]. Available: http://edshare.soton.ac.uk/20145/. [Accessed 21 September 2020]. |