[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.]

CE301 Final Report

Hypertext Game Project: HECC-IT

Project: **Hypertext Game Project**

Student Name: **Rachel Lowe**

Registration Number: **1804170**

Supervisor: **Dr. Richard Bartle**

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

Degree Course: **Computer Games BSc**

Contents

[Acknowledgements 2](#_Toc67931388)

[Abstract 3](#_Toc67931389)

[List of Symbols 4](#_Toc67931390)

[Literature Survey 4](#_Toc67931391)

[Technical Documentation 5](#_Toc67931392)

[Project Planning 5](#_Toc67931393)

[Intro 5](#_Toc67931394)

[Jira 5](#_Toc67931395)

[Usage of Jira 5](#_Toc67931396)

[Reflection on the project planning 5](#_Toc67931397)

[Conclusions 5](#_Toc67931398)

[Bibliography 6](#_Toc67931399)

# Acknowledgements

I would like to thank Dr Richard Bartle for the support he was able to provide for this project as my supervisor, and for dealing with my general incompetence over the past year or so. Without his guidance, this project would have ended up in a much worse state than it would have been in otherwise.

Thanks to my pastoral and study skills mentors, Michelle Carpenter and Michelle Paul, for putting up with me, and helping me to stay relatively sane and on track with my studies over the past three years.

I would also like to thank David (my brother) for the invaluable support he was able to provide after several components of my desktop PC died; without his help, I would not have been able to recover that computer, which would have made it much harder for me to get this coursework done. Thanks to my parents as well for putting up with me and helping me to stay alive whilst I was at home, and thanks to Betty for the moral support (even if she did eat my ethernet cable in February).

Special thanks to the contributors of *showdown.js* (*A bidirectional Markdown to HTML to Markdown converter written in Javascript*) [1], as this has been incorporated into the outputs produced by my tool.

Thanks to the academic and technical staff at the university for helping me to gain the necessary background knowledge to work on this project, and also for providing the software and other technical support which allowed me to complete this project relatively smoothly. In particular, I would like to thank Dr Dimitri Ognibene, for teaching the CE218 module last year, which was rather helpful for working out how to implement part of the final deliverable for this project.

# Abstract

HECC-IT (Hypertext Editing and Creation Code Integrated Toolkit) is a toolkit for authoring hypertext games, designed for indecisive people. Users may write/edit a raw .hecc file themselves, or use the 'OH-HECC' GUI provided by the tool to assist them in the editing process, before converting their .hecc file into a playable hypertext game. Unlike most existing hypertext game authoring tools, which require authors to exclusively write raw source code or exclusively use a GUI, HECC-IT has been designed to allow authors to edit their games however they want (with or without the GUI), without having to go through a convoluted process of converting their drafts into different formats before using the other editing method.

This tool has been used to produce several demonstration games (playable at <https://11belowstudio.itch.io/the-hecc-it-demo>), as well as a more fully-featured game: *Backblast*; a murder mystery where **you** are the victim (playable at <https://11belowstudio.itch.io/backblast>).

HECC-IT itself is available for free on my itch.io page, here: <https://11belowstudio.itch.io/hecc-it>.

(note: Depending on when you are reading this, the latter two links may not yet be publicly viewable. If this is the case, the password to view those pages is ‘301’)

# List of Symbols

* Hypertext Game
  + At the simplest level, a hypertext game is a game where the player is offered choices, which may lead to different sequences of events happening (or not happening). These can be presented as electronic html documents, but they don’t need to be.
* HECC-IT
  + Hypertext Editing and Creation Code Integrated Toolkit
  + This is the hypertext game authoring tool which I created. It will be explained in full throughout the rest of this document.
* HECC
  + Hypertext Editing and Creation Code
  + The intermediate scripting language used by HECC-IT, to store work-in-progress hypertext games (stored in .hecc files).
* OH-HECC
  + Optional Help for HECC
  + A GUI for editing .hecc files, included as part of HECC-IT.
* HECC-UP
  + HECC Ultra Parser
  + The part of HECC-IT responsible for turning .hecc files into playable hypertext games.
* HECCIN’ Game
  + HECC-Infused Nice Game
  + This is the ‘formal’ term for hypertext games produced with HECC-IT. ‘Nice’ in this context is not an indicator of the game’s quality, it’s an indicator of ‘oh nice you produced this hypertext game with HECC-IT’.
* HECCER/heccer.js
  + HECC Engine for Runtime
  + The (generic, pre-written) component of the HECCIN’ Game that works as the ‘engine’ for the game, responsible for all the backend logic.
* HECCED/hecced.js
  + HECC Exported Data
  + The component of the HECCIN’ Game that contains the exported game data (constructed from the .hecc file) produced by HECC-UP.

# Literature Survey

The creation of HECC-IT was heavily influenced by the research I performed on some existing hypertext game authoring tools, and on the topic of hypertexts in general. In this section of this report, I shall summarize my findings and explain how these findings influenced HECC-IT itself.

## The tool-based research.

I started by researching some existing hypertext game authoring tools. I did this because I wanted to gain an overview of the current state-of-the-art for the tools, to find a gap in the market that I could exploit. A full rundown of the tools (and most of the academic literature) I reviewed can be seen in the report on the background reading [2], so, to avoid reiterating those points again, I shall discuss the conclusions drawn from this reading.

The existing tools could be divided into several categories; some of them had a GUI, whilst others were all effectively just scripting languages (some with an IDE, some without an IDE), and most of these tools would require the author to exclusively use a GUI or a raw scripting language throughout the entire development process. In hindsight, this does make sense; it means that development time is not spent split between two ways of doing the same thing, only being concerned with one way of doing a task. There were two (pairs of) tools which did offer the user a choice between using a raw scripting language and a GUI, meaning that an author could, in theory, freely swap between whatever editing method they would currently deem more convenient, but, both of these did this with a caveat.

The *Inklewriter* [3] and *ink* [4] tools allowed some level of flexibility. *Inklewriter* is a server-side, GUI-based, authoring tool, but two of the obvious problems with it are how it’s server-side (meaning that if the company hosting it stops hosting it, this tool will cease to exist), and how authors need to make an account on the website to save/load their work (deterring authors who don’t want to deal with that). The *ink* tool is a client-side, scripting language-based authoring tool, offering the same functionality as *Inklewriter*, minus the GUI. Those two tools are somewhat interoperable, but in a rather inconvenient way. Whilst *Inklewriter* does have an option to export a game as an *.ink* file, this requires the author to manually copy and pastes the exported code into an .*ink* file. Then, to open an *.ink* file in the *Inklewriter* GUI, an author must first export that *.ink* file to JSON within the *.ink* tool, log in to *Inklewriter*, and then manually copy and paste the JSON into an ‘import from JSON’ option. This inconvenient process realistically means that no author would want to bother doing it.

*Twine* [5] and *Twee2* [6] were a bit less inconvenient in this regard. *Twine* is usable either via a web browser, or as a standalone executable, operates entirely client-side, and is a fully-featured GUI-based tool for producing hypertext games, offering plenty of flexibility for authors, even allowing authors to use different ‘formats’ (offering differing syntaxes/levels of functionality) for the games produced with that tool. Additionally, it presents a very helpful overview of games produced with it as networks of connected passages, making it very accessible for casual users. It saves the games in .html files, which can be opened in a web browser to be played, or opened within *Twine* for the actual game to be edited. *Twee2* is advertised as ‘Twine for power users’, effectively working as a pure code-based version of *Twine*, offering all of the options that *Twine* does (and then some), besides the GUI. The *Twee2* utility itself is a command-line program, which reads *.tw2* files, and exports them as .html files, as if those files had been made in *Twine*. Whilst this is more convenient than the *Inklewriter*/*ink* conversion, there are a couple of problems. If you are using Windows, you can’t use *Twee2* to convert from *Twine* format to *Twee2*. Additionally, the writer still needs to go out of their way to perform this conversion, so it still isn’t entirely convenient.

This exposed a clear gap in the market; a gap for a hypertext game authoring system that allowed users to freely choose if they wanted to edit their games using a GUI or by writing raw source code, so, if a writer wanted to make smaller edits to their game, they don’t need to use the GUI, and if they wanted to make larger changes to the overall structure, they can use the GUI.

Another key finding from this tool-related research was how many of the tools (*Twine* [5], *Inklewriter* [3], *Squiffy* [7], *Undum* [8], *eHyperTool* [9], *ChoiceScript* [10], *Inform* [11], *Quest* [12], *TADS3* [13], and *Ren’Py* [14]) are all capable of exporting games that can be played in .html format (even if, for some of these tools, some server-side legwork may be required by the author). This design choice makes these tools rather appealing from the perspective of a player of the games produced by the tools, as they won’t need to go out of their way to download/install anything beforehand, and can even play these games on their phones. When compared to *Storyspace* [15], which does not have this option, not including this option would very clearly discourage an author from using my tool. Even then, the games exported with some of these tools involve some server-side components, which can potentially discourage some authors from trying to distribute their games if they do not have a server which they can deploy their games on; therefore, I chose to make the games produced with HECC-IT consist of entirely client-side HTML and JavaScript code, for the sake of everyone’s convenience.

In terms of the tools themselves, the only ones which were usable via a web browser themselves were *Twine* [5], *Inklewriter* [3], *eHyperTool* [9], *Squiffy* [7], and *Quest* [12]; of these, only *Inklewriter* and *eHyperTool* could not be used as standalone applications, due to their inherent server-side nature. Therefore, I saw making the tool itself browser-based as not being a necessity, as the existing ‘standard’ for these tools did not extend to making the tools browser-based; in fact, it appeared more like the expectation was for the tools to be downloadable as standalone executable applications. Therefore, I chose to make HECC-IT a standalone application. This then begged the question of ‘what language should I write HECC-IT in?’. Ultimately, I chose to write in in Java. After noticing the operating system-related limitations of *Storyspace* [15] and *Twee2* [6], I knew that I wanted HECC-IT to not be plagued with such arbitrary platform dependence. I could have tried to find a language that has a cross compiler (allowing it to be compiled to multiple operating systems), but I chose Java, because the Java Virtual Machine already is available for multiple operating systems, so, I would only need to build HECC-IT once, and it would inherently run on any operating system. Java is also the language which I feel most confident in using, therefore, it was the obvious choice.

One other discovery I made during this initial research was about the existence of *The* *Treaty of Babel standard for Interactive Fiction Bibliography* [16]. The document itself specified some requirements for interactive fiction development tools that are signatories of this ‘treaty’, such that any works of interactive fiction produced with these tools can all be archived and identified in an appropriate way. The bare minimum requirements are that a tool should allow a title and an author name to be specified for the work produced, assign a unique ‘Interactive Fiction Identifier’ (or IFID) to the work, and for a C routine for the ‘babel’ utility to be contributed which can produce an ‘ifiction’ record for the work in question. The requirements for a tool which falls outside of the scope of the agreement are slightly different (the IFID must be the MD5 hash of the game file), however, during the development of HECC-IT, I chose to assign an IFID for my work as specified for a party which is a signatory to the agreement, and also produce an ifiction file for games produced with HECC-IT during the ‘parsing’ process that reads the input .hecc files and outputs the games. I justified this at the time because, at the time of the initial research, *Twine* [5], which was not currently a signatory to the treaty, assigned IFIDs to games produced with it as if it was a signatory, so I thought that there would be no problems if I were to do the same thing for HECC-IT. However, in January 2021, the tenth revision of *The Treaty of Babel* [17] was published, and *Twine* was now a signatory. Due to this, I am now not sure if the IFID-related components of HECC-IT would be considered ‘allowed’ or not. However, being realistic, I doubt that many people will find out about and use HECC-IT in the first place, so, for the time being, I suppose that I may not need to worry about this until HECC-IT somehow becomes widely used enough to justify asking if HECC-IT can become a signatory to the treaty.

In terms of functionality, the bare minimum supported by all tools was some method of linking between ‘passages’, some method of keeping track of what actions the player had previously taken, and some form of ‘guard conditions’/conditional statements. Some tools (such as *Quest* [12], *Inklewriter* [3], *eHyperTool* [9], and *ChoiceScript* [10]) implemented ‘links’ in the form of ‘pick one of these options’ at the end of each ‘passage’ in the text, always after the main passage content. In these tools, ‘guard conditions’ were implemented such that, depending on certain criteria, each of these ‘links’ would be selectively hidden/shown (with this condition being explicitly for the link), and in turn, could allow greater automatic validation of the game, at the cost of flexibility for the author. Conversely, *Twine* [5] (and the various story formats available for it), *Squiffy* [7], *Storyspace* [15], and *Undum* [8] all have the links defined ‘within’ the content itself. *Storyspace* is the odd one out here, as its links are considered ‘objects’ within the games it produces, and can each be individually configured. However, for the others, ‘guard conditions’ may be implemented in the form of conditional statements, which can be used to conditionally show/hide parts of the ‘content’ of the passages, and, in turn, conditionally show/hide links. This does mean that there is less scope for automatic checking of these ‘guard conditions’, however, due to the additional flexibility it offers the writer in terms of formatting, I chose to use this approach.

## The literature-based research

I studied a range of literature for this project as well, both on the topics of hypertext itself, hypertext games, and some on the topic of producing hypertext games.

The article which had the biggest impact on HECC-IT was S. Kitromili, J. Jordan and D. E. Millard’s paper on *What Authors Think about Hypertext Authoring* [18], which, at the time of first reading it, was a very recently-published paper (published in July 2020), and was the ‘Best Student Paper’ at the conference it was presented at, giving it some credibility. It highlighted several key points about the process of authoring hypertexts, from the initial idea to the finished products, and several complaints that some authors had about existing tools. When making HECC-IT, I decided to address the points raised about unclear documentation (by aiming to make it completely clear what HECC-IT could/couldn’t do), debugging tools (by making HECC-UP refuse to produce a game with an obvious error, giving details about it, and also indicating the states of the passages within OH-HECC), some ‘separation of content and behaviour’ (through explicitly-defined ‘comment’ areas, separate from passage contents), and several of the post-lifecycle complaints (making ‘distribution’ easy via the internet, simplifying ‘maintenance’ because only the ‘hecced.js’ file would need updating if the game is updated, offering some avenue for ‘profitability’ due to the control an author can have over the ‘hecced.js’ file with the data, and some level of ‘curatability’ via the *Treaty of Babel*).

From the approach of a literature review, however, I probably should have looked at some other literature on the topic of hypertext game authoring, as that paper was the only one I looked at on this topic. I’m not entirely sure if these findings which I applied to HECC-IT actually will turn out to have been properly useful or not, and, if I had looked at more research on this topic, I may have been able to make even better-informed decisions regarding HECC-IT’s overall design from the perspective of an author. But, I didn’t, so this part of the research is a bit questionable in hindsight.

The rest of the research had fewer obvious problems. On the topic of hypertext itself, I started with E. J. Aarseth’s *Cybertext: Perspectives on Ergodic Literature* (or, at very least, the first chapter of it, as I was unable to legally obtain a copy of the full book). Despite not having access to the full text, it was rather enlightening, explaining that a ‘cybertext’ is a text which, effectively, can be seen as some sort of ‘machine’, where there is a ‘textual feedback loop’, such that ‘the cybertext reader is a player’. It also covered ‘ergodic literature’, where ‘non-trivial effort is required to allow the reader traverse the text’ [19]; this made it very clear that the game I would need to produce with HECC-IT would need to contain these things to qualify as more than just ‘text’.

M. Bernstein’s article *On Hypertext Narrative* was also rather informative, discussing things such as how hypertexts must ‘offer links, but the selection of links must be significant and consequential’, and that ‘the cycle, not the branch, goto, or jump, is the central hypertext structure’ [20]. The former appeared rather obvious, but the latter was rather unexpected, and did start to give me a few ideas about the game I would ultimately produce, even if the game produced didn’t quite include the form of ‘cycles’ discussed by Bernstein in this article. H. K. Rustad’s article on *A Four-sided model for reading hypertext fiction* provided more insight into specific reading methods of hypertexts, from ‘semantization’ (effectively a ‘search for meaning’), ‘exploration’, ‘self-reflection’ (where players ‘play a role’), and ‘absorption’ (where the player is ‘in a condition of confusion’) [21]. These, along with the information about how to invoke these modes of reading, gave me a few ideas about how I could potentially structure the hypertext game I was going to produce, and was re-consulted during the development process, in an atempt to induce the desired effects on the player.

The other pieces of academic literature I read during this research didn’t leave much of an impression that translated into an actual impact on the development of the tool itself. However, some of these papers did identify a few examples of hypertext games, which I did look into a bit further.

## Reviewing the research into actual hypertext games

# Technical Documentation

The technical documentation for this project is held on Gitlab, and can be seen [here](https://cseegit.essex.ac.uk/ce301_2020/ce301_lowe_richard_m), on the project’s Gitlab repository [22]. They are held within the README.md files in the repository, and are organized with a table of contents on the top-level README.

# Project Planning

## Intro

## Jira

## Usage of Jira

## Reflection on the project planning

# Conclusions

# Bibliography

|  |  |
| --- | --- |
| [1] | E. Santos, P. Deschênes, C. Innis, R. Sharp, K. Käfer, R. Braun, D. Tarr, C. Chen, T. Stone, R. Sutherland, P. Lang, B. Combee, A. Backstrom, H. Wolfe, A. Courtiol, K. Balakrishnan, rheber, J. Gruber and J. Fraser, "showdownjs/showdown: A bidirectional Markdown to HTML to Markdown converter written in Javascript," 2 November 2019. [Online]. Available: https://github.com/showdownjs/showdown. [Accessed 19 Janaury 2021]. |
| [2] | R. Lowe, "The 'Summary of Background Reading' report I produced," 25 September 2020. [Online]. Available: https://cseegit.essex.ac.uk/ce301\_2020/ce301\_lowe\_richard\_m/-/blob/master/Reports%20n%20such/Summary%20of%20background%20reading.docx. [Accessed 25 September 2020]. |
| [3] | inkle Ltd., "inklewriter," inkle Ltd., [Online]. Available: https://www.inklestudios.com/inklewriter/. [Accessed 11 August 2020]. |
| [4] | inkle Ltd., "ink - inkle's narrative scripting language," inkle Ltd., [Online]. Available: https://www.inklestudios.com/ink/. [Accessed 11 August 2020]. |
| [5] | Interactive Fiction Technology Foundation, "Twine / An open-source tool for telling interactive, nonlinear stories," [Online]. Available: https://twinery.org/. [Accessed 5 August 2020]. |
| [6] | D. Q, "Twee2 | Interactive Fiction for Power Users," 2015. [Online]. Available: https://dan-q.github.io/twee2/. [Accessed 17 August 2020]. |
| [7] | textadventures.co.uk, "Squiffy - A simple way to write interactive fiction," 2020. [Online]. Available: https://textadventures.co.uk/squiffy. [Accessed 13 August 2020]. |
| [8] | 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]. |
| [9] | Beck & Bartle Limited, *eHyperTool Design Specification Version 1.02 (unpublished),* 2011. |
| [10] | 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]. |
| [11] | 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]. |
| [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] | 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]. |
| [14] | T. Rothamel, "The Ren'Py Visual Novel Engine," [Online]. Available: https://www.renpy.org/. [Accessed 6 August 2020]. |
| [15] | Eastgate Systems, Inc., "Storyspace: Storyspace," 2020. [Online]. Available: http://www.eastgate.com/storyspace/. [Accessed 3 August 2020]. |
| [16] | Interactive Fiction Technology Foundation, "The Treaty of Babel (Revision 9)," 24 October 2014. [Online]. Available: http://babel.ifarchive.org/. [Accessed 10 August 2020]. |
| [17] | Interactive Fiction Technology Foundation, "The Treaty of Babel (Revision 10)," 23 January 2021. [Online]. Available: https://babel.ifarchive.org/babel\_rev10.html. [Accessed 5 March 2021]. |
| [18] | 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. |
| [19] | E. J. Aarseth, Cybertext: Perspectives on Ergodic Literature, Baltimore, MD: Johns Hopkins University Press, 1997. |
| [20] | M. Bernstein, "On hypertext narrative," in *HT '09: Proceedings of the 20th ACM conference on Hypertext and hypermedia*, Torino, 2009. |
| [21] | H. K. Rustad, "A Four-Sided Model for Reading Hypertext Fiction," *Hyperrhiz: New Media Cultures,* vol. 6, 2009. |
| [22] | R. Lowe, "ce301 / ce301\_lowe\_rachel\_m · GitLab (The Gitlab Repository for this project)," 2021. [Online]. Available: https://cseegit.essex.ac.uk/ce301\_2020/ce301\_lowe\_richard\_m. [Accessed 2021]. |
| [23] | 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. |
| [24] | 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]. |
| [25] | 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]. |
| [26] | 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]. |
| [27] | J. Pope, "A Future for Hypertext Fiction," *Converg. Int. J. Res. New Media Technol.,* vol. 12, no. 4, pp. 447-465, 2006. |
| [28] | M. Pavić and S. Sofrenovic, "The Glass Snail," Word Circuits, August 2003. [Online]. Available: http://www.wordcircuits.com/gallery/glasssnail/. [Accessed 17 September 2020]. |
| [29] | 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]. |
| [30] | D. E. Millard, "Games/Hypertext," in *Proceedings of the 31st ACM Conference on Hypertext and Social Media*, Virtual Event, 2020. |
| [31] | 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]. |
| [32] | 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]. |
| [33] | 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. |
| [34] | 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]. |
| [35] | 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]. |
| [36] | M. Joyce, "afternoon, a story," Eastgate Systems, Watertown, MA, 1987. |
| [37] | M. Heyward, "of Day, of Night," Eastgate Systems, Watertown, MA, 2004. |
| [38] | 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]. |
| [39] | N. Gibbins, "Telling Tales - EdShare Southampton," 6 November 2019. [Online]. Available: http://edshare.soton.ac.uk/20145/. [Accessed 21 September 2020]. |
| [40] | T. M. Edwards, "SugarCube," 2020. [Online]. Available: https://www.motoslave.net/sugarcube/2/. [Accessed 5 August 2020]. |
| [41] | D. Cox and C. Klimas, "Snowman 2.0 Documentation," September 2019. [Online]. Available: https://videlais.github.io/snowman/2/. [Accessed 5 August 2020]. |
| [42] | M. Bernstein, "Storyspace 3," in *HT '16: Proceedings of the 27th ACM Conference on Hypertext and Social Media*, Halifax, 2016. |
| [43] | R. Bartle, "Demo 1," February 2011. [Online]. Available: https://www.youhaventlived.com/cbdemo/. [Accessed 4 August 2020]. |
| [44] | B. Barnet, "Machine enhanced (re)minding: the development of storyspace," *Digital Humanities Quaterly,* vol. 6, no. 2, 2012. |
| [45] | L. Arnott, "Harlowe 3.1.0 Manual," 24 September 2019. [Online]. Available: https://twine2.neocities.org/. [Accessed 5 August 2020]. |
| [46] | 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]. |
| [47] | 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]. |
| [48] | Interactive Fiction Technology Foundation, "Twine Wiki," Interactive Fiction Technology Foundation, 2019. [Online]. Available: https://twinery.org/wiki/start. [Accessed 5 August 2020]. |
| [49] | Interactive Fiction Technology Foundation, "Twine Specifications," 2019 February 2020. [Online]. Available: https://github.com/iftechfoundation/twine-specs. [Accessed 5 August 2020]. |
| [50] | textadventures.co.uk, "Squiffy - Documentation," textadventures.co.uk, [Online]. Available: https://docs.textadventures.co.uk/squiffy/. [Accessed 13 August 2020]. |
| [51] | textadventures.co.uk, "Quest 5 - Documentation," textadventures.co.uk, 31 December 2018. [Online]. Available: http://docs.textadventures.co.uk/quest/. [Accessed 11 August 2020]. |