Comput Game J (2018) 7:153–172
https://doi.org/10.1007/s40869-018-0062-y

Prototyping Tools for Game Writers

Henrik Engström[1] · Jenny Brusk[1] · Patrik Erlandsson[1]

Received: 11 April 2018 / Accepted: 30 May 2018 / Published online: 5 June 2018

Abstract A game is best evaluated by playing it and prototyping is therefore an important activity in game development. Game writers and narrative designers are responsible for the narrative structure of a game, which may have a varying degree of interactivity to it. The aim of this paper is to analyse the role of prototyping tools for game writers. There is a limited range of such tools available, of which Twine is one of the most established. Most of these tools have a text-based programming interface for modelling of game mechanics. This paper presents Deig—a prototyping tool for creating point-and-click adventure games. In Deig, game mechanics is modelled graphically using nodes from a set of primitives. We present an interview study where game writing students reflect on their experience of using Deig and Twine as prototyping tools. The result shows that both tools have their merits and complement each other. Deig was found to be intuitive for modelling of game mechanics, which lead students to create interactive narratives. Twine was found to be more useful for experimental writing. The conclusion of this work is that there is a need for a diverse set of prototyping tools to support game writing.

Keywords Computer game · Game writing · Narrative design · Pototyping · Tools

✉ Henrik Engström
henrik.engstrom@his.se

Jenny Brusk
jenny.brusk@his.se

Patrik Erlandsson
patrik.erlandsson@his.se

[1] School of Informatics, University of Skövde, Box 408, 541 28 Skövde, Sweden

1 Introduction

Game development is a collective activity, where individuals from a range of disciplines collaborate to create a complex product, and where the player is involved in the creation of the experience. Programming, graphics, audio and game design are well established disciplines in game development. In the last decades, game writing has been added to this list (IGDA no date; Dinehart 2011). Writing is mainly a solitary work using traditional tools such as pen, typewriter or word processor. A game writer however needs to collaborate with other disciplines, and the written material is integrated into a complex game production system. The game mechanics may be integrated with the narrative, which means writers need to handle branching narratives. Moreover, testing is essential in game development (Schell 2008). None of this is supported by traditional writing tools. The lack of a universal tool or format has been identified as the biggest challenge for game writers (Francis 2015).

The complexities of game writing have been handled in a range of different ways: from individual authors who claim they keep the model in their head (Despain 2008) to large team projects where writers and computer scientists create highly advanced AI-based interactive fiction (Koenitz 2016). An intermediate approach, commonly used, is to model branching narratives using Twine (Twinery.org 2018) or similar tools. In Twine, game mechanics can be modelled using a traditional scripting language. The programming environment provided in Twine lacks most features found in modern integrated development environments. Writers, who typically have a non-technical background, use Twine to write less logic-driven games (Friedhoff 2013).

The aim of this paper is to analyse the role of tools for game writers who want to prototype narrative and mechanical elements. This is done through a case study where Twine and Deig, a novel tool for dialog-based adventure games, are compared. A class of game writing students at the University of Skövde have used booth tools in different assignments. Their experiences and reflections have been analysed through semi-structured interviews. The result shows that both Twine and Deig were found to be useful prototyping tools but with quite different benefits. While Twine allows for more traditional authoring, and has a potential for innovative gameplay, the threshold to create new game mechanics was found to be high. Deig, on the other hand, is limited to a specific genre, but was conceived to be intuitive and very easy to learn. This made it possible for persons with limited programming proficiency to model game mechanics. The students reported insights into the complexities of interactive storytelling from working with Deig, which they had not got from working with Twine.

2 Background

Almost 20 years has passed since the "ludology versus narratology" debate dominated the scholarly discussions on computer games (Aarseth 2012). Today, it is hard to find anybody who claims that computer games cannot be studied from a narrative perspective or that the ludic dimensions of games make them different from non-interactive narration. Aarseth (2012), for example, presents a ludic-narrative design-space with four independent, ontic dimensions: world, objects, agents and events. He gives examples on how different games can be positioned at the pure story–pure game scale along these dimensions. Game mechanics is a central element in most games. There exist many definitions of game mechanics, used in different contexts such as game analysis and game design (Sicart 2008). One popular example of the latter is the *Mechanics Dynamics Aesthetics* (MDA) framework (Hunicke et al. 2004). In MDA, mechanics is defined in terms of the algorithms and data; dynamics is the run-time behavior of the mechanics; and aesthetics is the emotional response evoked in the player. A similar perspective is presented by Järvinen (2008) who emphasizes that mechanics can be seen as the verbs of the game. When dialogs are used in games, they provide *conversing* to the list of verbs.

2.1 Game Writing and Narrative Design

The International Game Developers Association (IGDA) describes game writers as "…those in the video game industry who write dialogue scripts, construct overarching narrative structure, produce documentation and articles, and a myriad of other writing tasks…" (IGDA, no date). IGDA does not define the role of a narrative designer but as the name indicates, rather than writing, their main responsibility is to design game narrative structures. Dinehart (2011) defines a narrative designer's role to "… champion story, craft compelling narrative elements, and define the systems through which they will be delivered to the player. Interactive Narrative Design is a new craft waiting to be further defined and explored". Given the large interest from scholars in the theoretical understanding of narratives in games, there is an appalling lack of interest to study game writing, as an applied activity. In the research database Scopus (Burnham, 2006) a search for "Game Writer" returns 5 publications and a search for "Narrative designer" returns 6. As a comparison, a search for "Game Designer" returns almost 1000 articles in Scopus. The scholarly interest has a strong bias towards theoretical reasoning and descriptive models rather than prescriptive (Koenitz 2016). Joyce (2015) presents a list of essential criteria that should be considered to best balance story and player agency. This list is based on previous scholarly discussions and experiences from playing games. The closest Joyce gets to applied writing is a brief discussion on the need for a "drama management system". This refers to studies from the AI research community where there is an interest in interactive fiction. These approaches are experimental, involves very advanced computational models, and do not represent a pragmatic view of game writing.

There is a need for more studies on game writing and on how game writers, in practice, can combine narrative and mechanics in games.

2.2 Tools in Game Development

Tools are vital parts of game development (O'Donnell 2011) and they play an important role in the creative process. In their study of creativity within the video game industry, Lê et al. (2013, p. 55) state that "[d]uring the creative process, technology first acts as a medium between ideas and their realization" and they add that, for games, ideas have to be implemented and tested before they make sense. It is for this reason Schell (2008) argues that developers should "pick a fast loop engine" to support early testing and iterative development. In the games industry, prototyping is commonly used in the early phases of a development process (Kasurinen et al. 2014; Schmalz et al. 2014) and companies expect their tools to support prototyping and allow for change (Kasurinen et al. 2013).

A distinction can be made between the full-fledged game engines used to develop the final game and lightweight prototyping tools that allow for rapid prototyping of ideas. In game production, the tool-chain used can be of pivotal importance (Murphy-Hill et al. 2014). Companies create and use pieces of software that glues together the various tools used by different disciplines. This pipeline can be highly specialized to specific company configurations and is typically not available when game engines are released (O'Donnell, 2011). Custom made tools for game writers may be part of this tool-chain. It is, however, notable that Excel is commonly mentioned as an interchange format between game writers and programmers (Despain 2008; Francis 2015). This indicates that there is a lack of specialized tools for game writers.

Prototyping tools are different to the production tools in that they allow for quick modelling of specific characteristics of a game and may not be part of the tool-chain. Nelson and Mateas (2009) present a requirement analysis of tools to support game design and the result of this study shows a variation in requirements from different designers. Some preferred to focus on the interface while others requested easy to use tools for modelling of game mechanics.

2.3 Prototyping Tools for Game Writing

There exist a large number of tools for rapid prototyping of games for specific game genres or limited subsets of games such as 2D, single player games etc. Popular examples of such tools include GameMaker: Studio, Construct 2, RPG Maker and Stencyl (Ciesla 2017). Most of these tools employ a combination of graphical modelling primitives and a scripting language to enable modelling of game mechanics. None of these tools are specifically targeting game writers and have very limited support for dialog writing or creating a narrative.

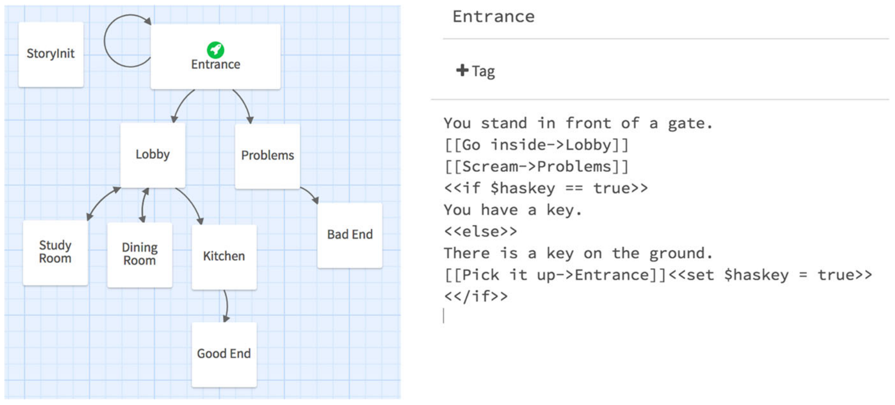
On the other side of the spectrum, there are general purpose tools that can be used to model narratives, game logic and user interfaces. Word processors and spreadsheets are used extensively, as well as writing tools focusing on managing text fragments, e.g. OneNote or Scrivener. For modelling of dependencies and flows, tools such as Visio may be used (Despain 2008). None of these tools support executing the modelled flow or testing the game.

There exist a few tools that are specifically targeted towards game writers. The most comprehensive is Articy:draft (Nevigo 2018) which support complex modelling of hierarchical state-charts. Another example is Chat mapper, which is targeted at dialog modelling. The mechanical behaviour in Chat mapper is modelled using Lua scripts. Articy:draft and the full featured version of Chat mapper are commercial products aimed at the tool-chain for game production. They are not primarily focused to support rapid prototyping of games. Among the freely available tools, Twine, Inklewriter and Ren'Py are examples of popular tools (Pixelles.ca 2017). Ren'Py (Renpy.org 2018) is a free tool focused on visual novels and its primary interface to create content is through scripting. Twine and Inklewriter (Inklestudios.com 2018) are both free tools for creating interactive stories, and both offer sufficient capabilities for prototyping such content. The graphical modelling of mechanics is limited in these tools, but both offer scripting languages, which can support complex mechanics. Twine is currently the primary tool used to teach interactive storytelling at the game writing program of the case presented in this paper.

2.4 Twine

Twine is aimed at interactive storytelling and was originally developed by Chris Klimas (Friedhoff 2013). In Twine, a game is created as a graph of connected passages (Fig. 1, left). Each passage is written in a text panel (Fig. 1, right) where prosaic text can be mixed with code segments and links to other passages. The core functionality of Twine is to create hypertexts. A passage can be seen as a page in a choose-your-own-adventure book. Links between passages are automatically added to the graph based on the links in text (using the "[[Label- > Target passage]]"-syntax).

The main purpose of Twine is to provide a graphical interface for creating branching stories. Twine does not, however, provide a smooth interface for the programming elements. Friedhoff (2013, p. 5) writes: "Twine has an ambivalent



Entrance

```
You stand in front of a gate.
[[Go inside->Lobby]]
[[Scream->Problems]]
<<if $haskey == true>>
You have a key.
<<else>>
There is a key on the ground.
[[Pick it up->Entrance]]<<set $haskey = true>>
<</if>>
```

Fig. 1 The passage graph of a Twine game (left) and the specification of a passage (right)

relationship with code" meaning that although Twine lacks much of the expected support to implement mechanics, there is a community who creates technically advanced Twine games. The main advantage of Twine, which has attracted a large user base, is that it provides a "code-free" environment for creating games (Salter 2016).

3 Deig

Deig is a standalone Java program that can be used to model logic, author dialog, debug and test dialog-based 2D adventure games. This allows for rapid prototyping of games. The tool has evolved through a research project primarily focused on including visually impaired players in a gaming experience (Engström et al. 2015). As part of this project, two mobile games have been developed and released. During the development of the first of these games, *Frekvens Saknas* (University of Skövde 2015), the developers identified a need for a tool to support the dialog authoring. A number of candidates were evaluated, including Twine, but none were found to feature the required capabilities. Instead, the first version of Deig was developed and used in the tool-chain. This first version was later extended to be used as a prototyping tool in game design and game writing courses. Deig has then been further extended and used in the development of *Marvinter* (University of Skövde 2017), a game produced in collaboration with the national Swedish Radio as part of their traditional, 24-episode advent calendar series. Marvinter was developed to include visually impaired players and hearing-impaired players. The game's narrative was intertwined with the narrative of the radio calendar. The total play time of the game was on average 7.2 h and it contained 10,000 recorded voice acting lines. As Deig aims to include visually impaired players, it uses TTS synthesis to enable rapid prototyping and testing of dialog. Deig games can be played directly in the editor but they can also be exported to Unity where prototyping assets (images and sounds) can be replaced with production quality assets (using Unity's systems for animations, lightning, 3D sound etc.).

The version presented and evaluated in this paper is Deig 1.1. This version has been extended and revised and is available for public download (http://deig.se). It can be run on Windows, OSX, and other desktop systems with a Java runtime environment. This free version does not include the proprietary text-to-speech (TTS) system used in the present study. Instead, it contains a number of free TTS alternatives and an API to add additional TTS modules.

3.1 Modelling Primitives

A game created in Deig is composed of a number of *chapters* (levels). Each chapter contains a number of *locations,* which in turn contains a number of *interactables*. When a game is played, the player is located in one of the locations and can be in one of two states: exploration state, where interactables can be found and activated; and dialog state, where the player can engage in a traditional menus-based dialog with the characters of the game. The logic of the game is modelled using a set of *flow nodes,* which are used to create a graph for each interactable. For each location, there are two special types of interactables: *enter interactable* and *exit interactable*. These interactables are triggered automatically when the player enters and exits a location respectively.

Figure 2 shows the main window of Deig. The window has three main panels: a list of locations (upper left), a list of interactables (lower left) and a main panel (right). The properties of a location can be edited in the list in the upper left corner. In Fig. 2, the location "Bedroom" is selected, and its graphical representation is shown in the main panel. There are two interactables in the Bedroom: Bed and Door. The positions of these interactables are marked with circles in main panel.

3.2 Modelling of Game Mechanics

The dialog and logic of an interactable is modelled using a set of *flow nodes* connected in a single-entry, directed cyclic graph (Fig. 3).

There are eight different node types that form the modelling language of Deig: Act, Dialog, Set variable, Condition, Fork, Transit, Dice, and Code (see Table 1).

Each node can have one or more input connections and zero or more output connections (Fig. 4). When a node graph is traversed and a node has more than one output connection, the output is selected based on user input (the dialog node) or the node's logical behaviour (all other nodes). If a node has no output connection, the game returns to exploration mode. In the case of the transit node, this is preceded by the execution of the current location's exit nodes (if any) followed by the new location's enter nodes (if any). The exploration then starts in the new location.

All node types, except the fork node, have inspector dialogs where the specifics of a node are modelled. For dialog authoring, the act node editor is used, shown in Fig. 5.

The dialog text is written as a sequence of lines. For each line, the character name and an emotion are specified. For the main character, the line can also be flagged to

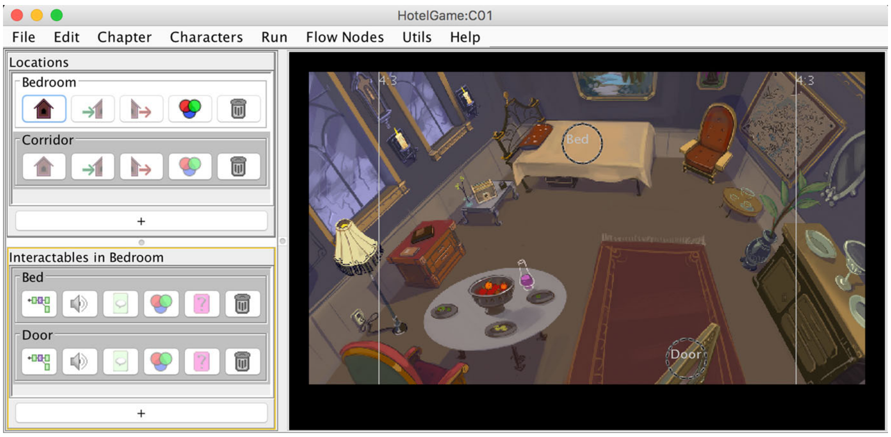


Fig. 2 The main window of Deig, with some sample game content

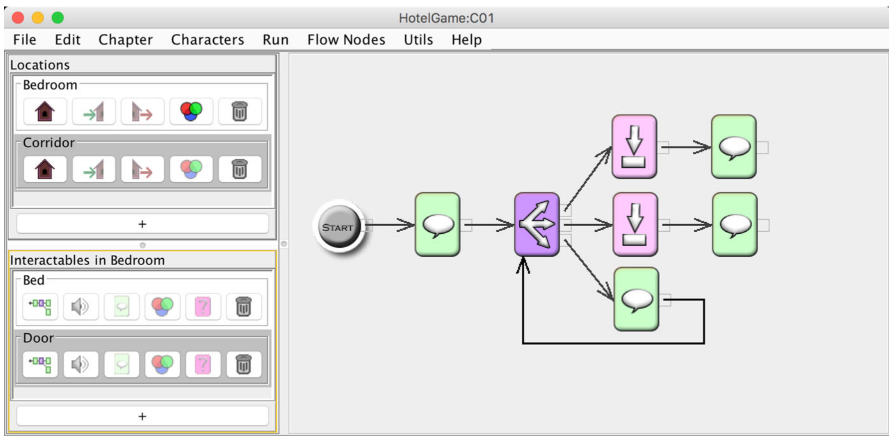


Fig. 3 A node graph used to model the dialogs and logic of an interactable

Table 1 The eight flow node types used to model dialog and logic in Deig

Icon	Description	Icon	Description
💬	Act node—a sequences of dialog lines	⑂	Fork node—selects the first output the first time, the second all other times
⇐	Dialog node—enables the user to select a dialog choice	⇨	Transit node—changes the active location for the player
⇩	Set variable node—sets the value of a named variable	🎲	Dice node—randomly selects an output.
?	Condition node—selects an output based on conditions using variables	}	Code node—miscellaneous actions related to game state (chapter transits etc.) and multimedia (sound etc.)

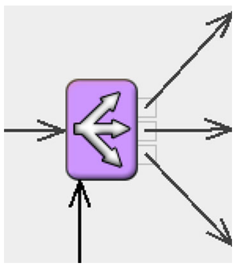
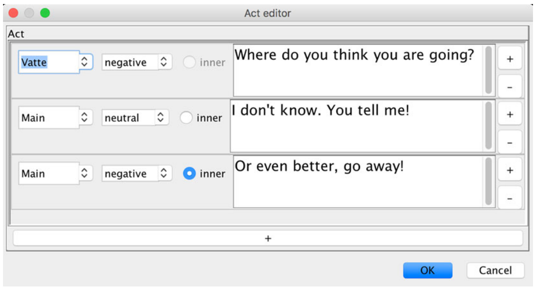


Fig. 4 A node can have one or more inputs (arrows leading to the node) and zero or more outputs (arrows leading from the node)



Act editor

Act

Vatte	negative	inner	Where do you think you are going?
Main	neutral	inner	I don't know. You tell me!
Main	negative	[✓ inner]	Or even better, go away!

Fig. 5 The act node editor where dialog lines can be added and edited

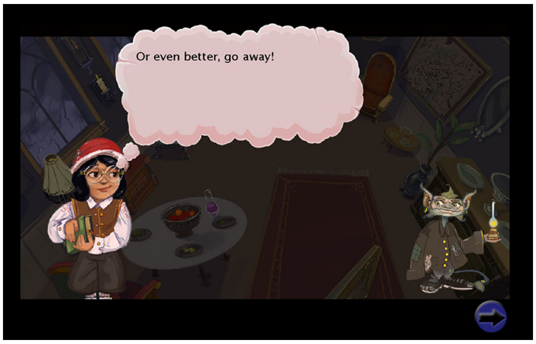


Fig. 6 Play testing in Deig

be inner monolog. When played, emotions and inner monolog are expressed visually through different speech bubbles (Fig. 6).

4 Case Study

The role of tools for game writers have been analysed through a case study with game writing students taking a course on dialog systems for games. In this course they used both Twine, which they had prior experience of, and Deig, which was new to them. Through semi-structured interviews their experiences and reflections have been analysed.

4.1 Respondents and Course Module

The game writer programme at the University of Skövde is one of very few dedicated programmes in this area in the world (Sheldon and Toftedahl 2016). It is one of six 3-year bachelor programmes within computer games development at the University. The other programmes major in programming, graphics, audio, music and game design. In addition to courses focusing on their major, all programmes contain collaborative game development projects where students from all disciplines develop a 2D game (first year) and a 3D game (second year). The students participating in this study were all second-year students, half-way through their education. They had taken an introductory course to procedural programming; an introductory course to game design and prototyping using GameMaker: Studio;

and they had used Twine in previous courses. They had participated in the first game project but not yet in the second-year project.

The course module studied in this paper is focused on dialog systems for game writers. The course contains two major assignments. The first is an individual assignment where each student develops a dialog-based game using Twine (version 2.2.1). In the second assignment, students in groups of 2–3 develop a dialog-based game using Deig. The students were given an introductory lecture followed by an individual 3-h practical exercise in using Deig. The group then had 5 days to complete the game. The assignment was finished with a play-test session where groups tested each other's games. The interviews were held in direct connection to this test session.

4.2 Method

The interviews were held with the groups that had been working together with Deig. The choice of the small group interview format was made to create a peer environment similar to a format already established in many courses. The interview followed a semi-structured protocol (DiCicco-Bloom and Crabtree 2006) with three main sections: subjects' background, their perception of the assignments and the tools, and specific elements of Deig (Appendix 1).

Questions regarding their perception of the tools and difference between Twine and Deig were gradually made more and more specific. This made it possible for subjects to raise observations regarding differences before they were explicitly asked about it. In the end of the interview, subjects were asked if they wanted to add anything.

All interviews were recorded and transcribed. The transcribed material has been coded and analysed using MAXQDA 2018 (Verbi 2018). A first step in the analysis of the interview material was to identify recurring themes, which were then used to code the transcripts. All interviews were held in Swedish. Cited quotes from interviews have been translated into English. Text marked with italics indicates that a clear emphasis was put on that word in the interviews. Ellipsis (…) marks longer pauses.

The respondents were informed that participation was voluntary, that they are anonymous, and that the recordings will be deleted after transcription. A written consent was signed by all participants.

5 Results

There were 19 students who were actively taking the course and they produced 16 Twine games individually (3 did not complete the assignment in time). All groups completed the Deig task and created 8 games.

The Deig games have a playtime between 10 and 20 min. All games have meaningful gameplay in that there are clear goals for the player to achieve and there are consequences to the actions made. On average, a game has 300 dialog lines and

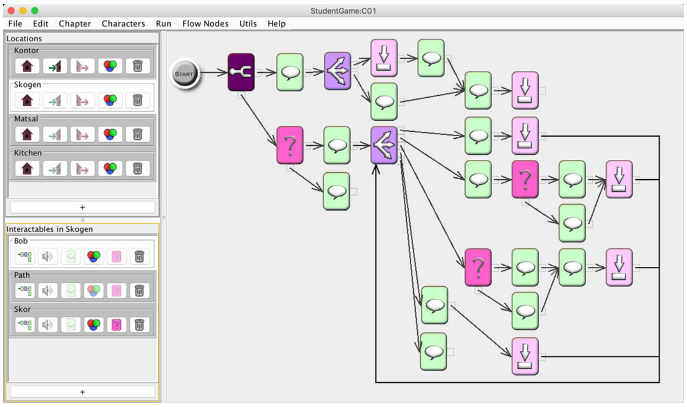


Fig. 7 An example of a logic model for an interactable

involves 4 characters in addition to the protagonist. Figure 7 shows the logical model of an interactable in one of the produced games.

The Twine games produced in the module are more varied than the Deig games. A majority has a linear progression without any repeated passages. Figure 8 shows the simplest (left) and most complex (right) structure of the 16 games. Note that these are models for complete games while the model in Fig. 7 is for one of 20 interactables in that game.

The produced prototyped were play-tested by primarily other students and instructors on the course. All prototypes were high fidelity with respect to the interactivity but there were differences in terms of visual design fidelity. The Twine prototypes were in general low fidelity with the major interaction through a text-based interface. These games were played in a web-browser on a desktop computer. These games resemble game which Friedhoff (2013) claims is typical in the Twine community. The Deig prototypes contained both graphics and audio and were tested in two different versions: initially these games were tested in the low-fidelity player

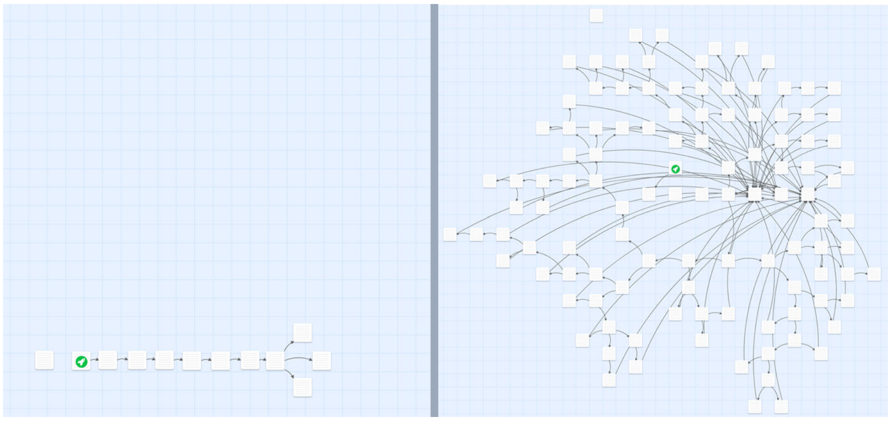


Fig. 8 Examples on the structure of two produced Twine games

of Deig on a desktop computer. The final games were tested in high fidelity versions on tablets. In this version, many graphics and audio elements have production quality. This type of prototype has been used extensively in play tests conducted in the production of Marvinter (University of Skövde 2017).

In total 16 students (7 females, 9 males) chose to participate in the study. The average age of subjects were 23 years. All eight Deig teams were represented in the material. One of the interview sessions was held with a single respondent, another with a team of three respondents. The remaining six sessions were held with two respondents. The duration of interviews ranged from 16 to 25 min (on average 20 min).

5.1 Themes

A general observation from the initial analysis of the material is that the eight groups gave a relatively uniform picture. Saturation was reached after approximately two-thirds of the groups. A number of recurring themes have been identified in the material. These themes can be coarsely clustered where the first four themes are more geared towards the comparison between Twine and Deig, while the last three themes are more general reflections. The first four concern: the author experience; modelling of structure; intuitiveness; and usability. The three remaining themes relate to: game dynamics; implications of audio-visual elements; and programming proficiency.

5.1.1 Author Experience

All teams expressed that there was a clear difference between Twine and Deig when it came to the freedom to write. Twine was perceived as more open where Deig is focused on a specific genre. As one subject put it: "… I want to create an adventure game… I don't know how to program but I don't give a shit about it, then [Deig] is the way to go" (S14). Deig was appreciated for the simplicity to write dialogs but several subjects expressed that they lacked the freedom to write long prosaic texts. Several subjects expressed that they found Deig to be more oriented towards games and Twine more towards creative writing. One subject stated that "… you use it in two completely different ways. This [Deig] was much more towards games but when I write Twine projects the feeling is much more that I try to write… something that could be in a book" (S7). This is in line with another subject who stated: "Twine is also somehow a bit more serious and this leads you to write some kind of short story, and you just write straight away and you know that the player will actually read this later" (S11).

At the same time, Deig was appreciated for its focus on dialog and several subjects found that it had helped them to develop their dialog writing skills. All groups reported that they felt that they had created a story in the Deig task. Some groups discussed how the authoring process had been affected by the nature of the tool and that it guided their creativity in certain directions. "The story came to life by itself in some way, I think" (S9).

5.1.2 Modelling of Structure

It was very clear from the interviews that Deig was perceived to be very intuitive for modelling the structure of the story and game. All groups expressed this. For example, when asked for spontaneous reactions to the Deig-exercise, one subject replied: “Actually, my first reaction to this program was: shit this is actually *intuitive…*” (S3). Another subject responded to the same question: “I found [Deig] to be super agile. Especially considering we have worked with Twine previously, which was much more complicated for creating trees and such things… so this has been a much easier tool to work with… to understand… to see the structure of everything…” (S7). Another subject was asked to describe the difference between Twine and Deig when it comes to modelling the logics of the game: “*much* easier in [Deig] *I think…* to have a passage with five, six if-statements becomes *super complex* [laugh] in Twine… and… in [Deig] it feels—simpler to keep track of” (S6).

A few respondents expressed that they were aware of the potential of Twine to create advanced game mechanics through scripting but they also expressed that it was a complex task: “… if you have an idea then it is *possible* to do it in Twine. It may not be *easy*. You *should* probably not do it in Twine. But you *can* [laugh]” (S14). Others argued that Deig emphasised a different kind of creative expression than Twine: “… you can focus *more* on how to structure… mechanics… compared to Twine and similar [programs] where the focus is much more on… like write and write…” (S5).

Half of the groups mentioned limitations in Deig with respect to modelling of structure. One subject found that Deig lacked timing-based mechanics; two others expressed concerns regarding the layout of flow-nodes when the graph grew big. Finally, one subject expressed concerns that logical disjunctions could not easily be expressed.

5.1.3 Intuitiveness

All groups expressed clearly that they appreciated the intuitive interface of Deig and that they quickly had learned to use it. One subject spontaneously stated that “… it feels as in each course we take we get some new program to learn… but this one [Deig] was surprisingly… it was surprisingly *quick* to learn.” (S3). This opinion was expressed by a large number of subjects. They also found Deig to be easy to work with and that they quickly could create prototypes: “It is possible to do something very quickly without any real programming experience” (S14). It was furthermore perceived to be even quicker than Twine: “… if you compare with like… the game that we actually made now and if you were to do it in Twine. So *this* was made in a couple of days” (S6). The same subject concluded: “… and if we had made it in Twine it would for sure take *more* than a week” (S6).

Finally, one clear trend in the material is that subjects enjoyed working and playing with Deig. This is an important quality of a game prototyping tool; it should support playfulness even in the construction phases. As one subject stated: “I enjoyed the program [Deig] very much, it was very fun to play in” and he continues

"… I even sat and played with it in my spare time" (S9). It is likely that the perceived swiftness when modelling games in Deig contributes to the fun. Another contributing factor may also be the TTS, which makes it possible to hear the written dialog with different voices.

5.1.4 Usability

Many subjects reported that they lacked features in Deig that they were used to have in similar applications. This can be seen as usability issues. The most commonly mentioned missing feature was support for an undo-function. Other features that were mentioned were spell-checker, support for object renaming, support for snapping in the node graph and better integration between Deig and the exported Unity game with respect to text-layout. In most cases subjects explicitly remarked that they did not consider these limitations to cause any major obstacles: "… small sources of irritation that might grow. Nothing that is game breaking. Nothing that makes it impossible to continue… at least not that I have noticed, but sources of irritation, yes" (S12).

5.1.5 Game Dynamics

Game dynamics is the run-time behaviour that emerge when the game mechanics is acting on player input (Hunicke et al. 2004). A majority of the groups reported that the Deig assignment had given them an understanding for game dynamics. This is something that can be gained from working with Twine as well, but it appears that the characteristics of Deig made game dynamics more apparent to them. The comments regarding dynamics can be divided into two types. The first type of comments related to how introduction of game mechanics and user choice lead to complex logical models. When questioned on what they had learned from the Deig module, one subject stated: "I would say how quickly it gets complicated. How quickly it… kind of… a few choices make everything… *extremely* so much more… it not just… like writing a script for a movie where it's just *straight*" (S7).

The second type of comments are insights on how to write for games where the dialog is affected by game mechanics. For example, one subject stated: "It is possible to develop Twine to be strongly… explorative too, but it is more… it is more complex to do it, while in [Deig] it comes naturally to have an explorative perspective on it" (S11). This subject uses the term *explorative* to characterize the freedom of players to give input to the system.

Another subject (S3) discussed the situation where a player returns to a character, that she has interacted with previously. The subject expressed that it made a huge difference to the player experience to have variations in the dialog. This could, for example, be to have three or four different greeting lines that the game alters between, when the player returns several times.

Play testing had an important role for subjects to realise the implications of game dynamics. The games had been subject for frequent testing, including internal play testing within the group as well as tests by instructors and other students: "… it is relatively quick to kind of go through what you have, and kind of test and test again, and then invite the others" (S16).

5.1.6 Implications of Audio-Visual Elements

One difference between Twine and Deig that was highlighted by all groups is the presence of graphical assets and multi-voiced TTS. There were remarks that this had both positive and negative implications to the creative process. The TTS was generally conceived to be constructive in the writing process in that it helped to analyse the length and quality of spoken lines. One subject explained: "… even if it sounds very cracked and robot-like. So you still notice if it is too much or if it is too slow or… 'God this is boring to listen to.'" (S3). Several groups also reported that they were inspired by the characteristics of the voices and that it affected how they perceived their own characters: "… that the voice also affects how you write to that character, so I found it interesting to reflect on this kind of interaction" (S15). Although most comments regarding TTS were positive, some groups also expressed worries that it could have a negative impact on the writing and that there were limitations to it: "… it can also be a problem that… you tend to… it can be the reason many choose to do comedy games just because they laugh so much at it [the TTS]. It could be a bit harder to do more serious games in this manner." (S12).

Several subjects noted that the presence of graphics made it unnecessary to write descriptions of the environment. It was appreciated that the graphics made it possible to quickly prototype a "real game", but some subjects did not appreciate how the available graphics guided the stories towards a certain genre. At the same time, they were aware that the alternative would be to create their own graphical assets, which would be much more time consuming.

5.1.7 Programming Proficiency

All subjects participating in the study had taken a course in introductory programming and a course in game design and prototyping. Still, many subjects revealed an inability to analyse and discuss programming concepts. Boolean variables were frequently referred to as "true or false" and several subjects had problems characterizing the node graph and talked about it as "the grid" or as "dots". Another example is a subject who, when talking about conditional dialog options, expressed: "If all conditions for showing is running at the same time, then you will never have the chance to trigger the condition to remove the alternative…" (S5). Although it may be possible to follow the line of argument, the concepts of *running* and *triggering* a condition are not well defined. This indicates a low programming proficiency. Despite this, all subjects stated that they felt that they mastered the mechanical modelling in Deig to a high degree. The subjects who exposed greater programming proficiency expressed that Twine allowed for flexible modelling of mechanics, but they also acknowledged that it may be a complex task to achieve, as exemplified in the quote from S14 in the section *modelling of structure* above.

5.2 Discussion

Both Twine and Deig were perceived to be useful tools for prototyping games but subjects expressed that they saw fundamental differences between them. Deig was perceived to be easier to learn and use for modelling of the mechanics of the game, while Twine offers more freedom for authoring text-centred interactive stories.

The group of students participating in the study had experience from a programming course and from a game design course using GameMaker: Studio. Still, most subjects revealed a limited familiarity with programming concepts and many of them expressed that they found it challenging to use the scripting functions in Twine. Even though Twine offers rich capabilities through scripting, it is in practice unavailable to game writers unless they have sufficient understanding of programming fundamentals. In Deig, all subjects were able to fully comprehend the mechanism to model game mechanics in the course of a week. As a consequence, many subjects reported that they had realised how complex interactive writing can be and that they had realised what the consequences can be when a player is given control of the order of actions. These are very important insights for game writers, and it is notable that many subjects had not got it from their previous work with Twine.

An interesting observation from the interviews is that the presence of graphics and audio, including TTS, affected the writing process, even in a prototyping stage. This emphasizes the importance of integrating the writing process with the other disciplines of game development. Writing in isolation may produce an expression that makes audio and graphics redundant or dissonant.

A general observation from the analysis of the interviews is that all groups made comparisons between Twine and Deig at an early stage of the interview, before they were explicitly asked to compare the tools. In several teams there was no need to explicitly ask for a comparison. This indicates that the similarities and differences identified were not far-fetched but came naturally from experience with both tools. The general impression from the interviews is that most subjects appreciated both Twine and Deig for different reasons.

5.3 Limitations

The results presented in this study are based on experiences from a relatively homogenous group of students. They do not represent actual game development practices. It is, however, important to note that that Deig has been used by developers with professional experience to produce games that have reached a large audience and been well perceived (Riis 2017). The results from the interviews resembles, to a large extent, anecdotal observations made during the development of these games. For example, the swift creation of prototypes made possible through Deig was found to be invaluable in the production of the 24 episodes of *Marvinter*. The majority of these episodes went through several major revisions and restructurings.

A limitation in the presented study is that the students had prior experience of using Twine when they approached Deig. This will affect the learning curve and their perception of Deig. The assignments for the respective tools also differed, which may impact on the comparison.

The interview was conducted by an instructor in the course, which may affect the respondents, even though they were clearly informed that the instructor had no role in the remaining examination of the course. The students were moreover aware that the researcher conducting the study had developed Deig. This may limit subject's openness to express criticism. The choice of the peer small group interview format will however likely have increased the subjects' openness to express their opinions and share their thoughts. This format has been found to have this effect when interviewing children (Mauthner 1997).

6 Conclusions

In this paper we present Deig—a prototyping tool for creating dialog-based point-and-click adventure game. It has been used in a course module for second year game writing students. The students have been interviewed about their experiences after working with Deig in comparison to Twine, which they also used in the same module, as well as in previous courses.

The results showed that students perceived the two tools to have similarities, but a number of fundamental differences were also observed. Deig was found to be very easy to use for modelling of game mechanics, rapid prototyping and play testing. The subjects enjoyed working and playing with the features of the editor and they reported insights on the nature of interactive narratives. Twine, on the other hand, was found to be more open-ended with respect to writing, but also that modelling of game mechanics was perceived to be complex. Theoretically, Twine can support practically any game through its scripting language, but it was not perceived as a viable option for the game writer students in this study. Those few subjects that indicated that it may be possible to achieve were well aware that it would be time consuming and complex. It appears that both tools were found to be useful but in different ends of the ludo-narrative design space (Aarseth 2012). Deig supported rapid prototyping of ludic-oriented concepts while Twine supports rapid prototyping of narrative-oriented concepts. The recommendation from this study, for game writers with little programming experience that wants to develop games that has dialog closely connected to game mechanics, is to use Deig. The graphical scripting capabilities of Deig has a much lower threshold than the text-based scripting of Twine.

An observation that can be made from the presented work is that the prototyping tool has an effect on the creative process. Twine invites to write long passages of text where Deig and its use of speech synthesis emphasises short dialogs and interactivity. Game writers may use any of these tools to create interesting gaming experiences but these may be drastically different. For some games, the lack of choice and interactivity may even be central to the experience (Salter 2016). Traditionally, games are designed to provide players with meaningful choices and agency (Salen and Zimmerman 2004). Which tool to use may depend on personal preference (Nelson and Mateas 2009), but it may also be affected by the target context for the final game. For solitary game writers, the prototyping tool may be the same as the production tool. For development teams, though, the target game engine may be more or less compatible with the prototyping tool. As was noted in this study, the presence of audio and graphical assets affected the direction of game writing and geared the process towards a gaming experience where other disciplines produce elements that contributes to the narrative. Text based tools such as Twine offers a space for individual game production (Salter 2016), but the main stage for game writers is in team-based production of 2D and 3D games. For those products, audio and graphics plays an important part in creating and conveying the narrative.

A future study will focus on the role of speech synthesis in the creation of dialog-based games. Another future work is to extend this study to include other tools used by game writers to prototype games. Chat Mapper, for example, is a tool similar to Deig, but where Lua scripting is used to model logic and with a different interface to the dialog modelling and play testing. A possible approach for this comparison would be to use heuristic evaluation and similar theoretical methods. Another important future work is to study how professional game writers in the game industry are working and what tools they are using. Within such a scope it would also be important to study the programming proficiency and *computational thinking* (Wing 2006) among professional game writers. Game writing and narrative design is an under-researched area despite the large scholarly interest in the descriptive understanding of narratives in games.

Compliance with Ethical Standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Informed consent Informed consent was obtained from all individual participants included in the study.

Appendix 1: Interview Protocol

Background

What is your prior experience of: game development; game engines; programming?

How do you perceive your role as a game writer? What tools do you think you will use?

Reflections on the Assignments and Tools

What are your spontaneous reactions?

What are the most important insights this module has given you?

What are the strengths and weaknesses of Deig?

To what extent did you feel you created: a narrative; an interactive game?

To what extent did you feel that you master the features of Deig for creating game logic?

Describe how you perceive the differences between Deig and Twine when it comes to:

- creating a narrative;
- creating game logic.

Elements of Deig

How did you perceive the possibilities to: test and debug; play on tablet; use speech synthesis?

Other

Do you want to add anything else?

References

Aarseth, E. (2012). A narrative theory of games. In *Foundations of digital games conference* (pp 129–133). ACM.

Burnham, J. F. (2006). Scopus database: A review. *Biomedical Digital Libraries, 3*(1), 1–8. https://doi.org/10.1186/1742-5581-3-1.

Ciesla, R. (2017). *Mostly codeless game development*. Berkeley, CA: Apress.

Despain, W. (2008). Interactive script formatting. In W. Despain (Ed.), *Professional techniques for video game writing*. Boca Raton: CRC Press.

DiCicco-Bloom, B., & Crabtree, B. F. (2006). The qualitative research interview. *Medical Education, 40*(4), 314–321. https://doi.org/10.1111/j.1365-2929.2006.02418.x.

Dinehart, S. (2011). *What is a narrative designer?* http://narrativedesign.org/2011/09/what-is-a-narrative-designer-3. Accessed 30 March 2018.

Engström, H., Brusk, J. & Östblad, P. A. (2015). Including visually impaired players in a graphical adventure game: A study of immersion. *IADIS International Journal on Computer Science and Information System*, *10*(2), 95–112.

Francis, B. (2015). *Writing, honest tales from the trenches of AAA game*, *Gamasutra*. https://www.gamasutra.com/view/news/250657/Honest_tales_from_the_trenches_of_AAA_game_writing.php. Accessed 1 April 2018.

Friedhoff, J. (2013). Untangling twine: A platform study. In *DiGRA conference*.

Hunicke, R., LeBlanc, M. & Zubek, R. (2004). MDA: A formal approach to game design and game research. In *AAAI workshop on challenges in game AI* (pp. 1–5).

IGDA. (no date). *IGDA game writing SIG: About the game writing SIG*. http://www.igda.org/members/group_content_view.asp?group=121051&id=421185. Accessed 30 March 2018.

Inklestudios.com. (2018). Inklewriter. Inklestudios.com. Accessed 1 Apr 2018.

Järvinen, A. (2008). *Games without frontiers: Theories and methods for game studies and design*. Tampere: University of Tampere.

Joyce, L. (2015). Creating collaborative criteria for agency in interactive narrative game analysis. *The Computer Games Journal, 4*(1–2), 47–58.

Kasurinen, J., Maglyas, A. & Smolander, K. (2014). Is requirements engineering useless in game development? In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (pp. 1–16).

Kasurinen, J., Strandén, J. P. & Smolander, K. (2013). What do game developers expect from development and design tools? In *ACM international conference proceeding series* (pp. 36–41).

Koenitz, H. (2016). Interactive storytelling paradigms and representations: A humanities-based perspective. In R. Nakatsu, M. Rauterberg, & P. Ciancarini (Eds.), *Handbook of digital games and entertainment technologies*. Singapore: Springer.

Lê, P. L., Massé, D., & Paris, T. (2013). Technological change at the heart of the creative process: Insights from the videogame industry. *International Journal of Arts Management, 15*(2), 45–59.

Mauthner, M. (1997). Methodological aspects of collecting data from children: Lessons from three research projects. *Children and Society, 11*(1), 16–28. https://doi.org/10.1111/j.1099-0860.1997.tb00003.x.

Murphy-Hill, E., Zimmermann, T. & Nagappan, N. (2014). Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development? In *International conference on software engineering* (Vol. 2568226, pp. 1–11). ACM.

Nelson, M. J. & Mateas, M. (2009). A requirements analysis for videogame design support tools. In *International conference on the foundations of digital games* (pp. 137–144). https://doi.org/10.1145/1536513.1536543.

Nevigo. (2018). Articy:draft. https://www.nevigo.com. Accessed 1 Apr 2018.

O'Donnell, C. (2011). Games are not convergence: The lost promise of digital production and convergence. *Convergence, 17*(3), 271–286.

Pixelles.ca. (2017). *An extensive list of free narrative game engines*. https://pixelles.ca/2017/04/an-extensive-list-of-free-narrative-game-engines/. Accessed 1 April 2018.

Renpy.org. (2018). Ren'Py. renpy.org. Accessed 1 Apr 2018.

Riis, J. (2017) *SR Christmas calendar game Marvinter released*, *Nordic Game News*. https://nordicgame.com/sr-christmas-calendar-game-marvinter-released/. Accessed 1 April 2018.

Salen, K., & Zimmerman, E. (2004). *Rules of play: Game design fundamentals*. Cambridge: MIT Press.

Salter, A. (2016). Playing at empathy: Representing and experiencing emotional growth through twine games. In *Serious games and applications for health* (pp. 1–8). IEEE.

Schell, J. (2008). *The art of game design: A book of lenses*. Amsterdam: Elsevier.

Schmalz, M., Finn, A. & Taylor, H. (2014). Risk management in video game development projects. In *Proceedings of the annual Hawaii international conference on system sciences* (pp. 4325–4334).

Sheldon, L. & Toftedahl, M. (2016). Beginning with the word: Building a game writing program. San Fransisco: game developers conference. https://www.gdcvault.com/play/1023044/Beginning-with-the-Word-Building.

Sicart, M. (2008). Defining game mechanics. *The International Journal of Computer Game Research*, *8*(2).

Twinery.org. (2018). Twine. Twinery.org.

University of Skövde. (2015). Frekvens Saknad. https://itunes.apple.com/se/app/frekvens-saknad/id969079265. Accessed 1 Apr 2018.

University of Skövde. (2017). Marvinter. https://itunes.apple.com/se/app/marvinter/id1299687281. Accessed 1 Apr 2018.

Verbi. (2018). *MAXQDA*. Berlin: Verbi Software GmbH.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.