6-2013

GUInform: Interactive Fiction for GUI Prototyping

Tesca Fitzgerald
Portland State University, tesca@cs.pdx.edu

GUInform: Interactive Fiction for GUI Prototyping

Tesca Fitzgerald
Portland State University
P.O. Box 751
Portland, OR 97207
tesca@cs.pdx.edu

ABSTRACT

There are many methods of rapidly prototyping a graphical user interface (GUI), the most prominent being paper prototyping and wireframe prototyping. In paper prototyping, the developer creates a physical GUI prototype using simple materials such as paper, pencils, and tabbed cards. Paper prototyping, while easy to implement, leads to unrealistic and awkward user interaction. Wireframe prototyping involves the use of simplified software tools to develop a primitive version of the proposed user interface. This method enables graceful interaction, but requires major implementation effort. I propose a middle ground: a method of rapid, interactive GUI prototyping using Interactive Fiction (IF) tools.

I have modified Gargoyle, an IF environment, to render and display SVG in the interactive console. I have also built libraries for the Inform 7 IF description language, enabling SVG rendering of GUI elements. The resulting system, GUInform, permits a GUI developer to use Inform 7 to define the elements of a GUI prototype and how they respond to interaction. A user can then interact with this prototype by typing textual commands and viewing the displayed responses.

Keywords

Graphical User Interface, Prototype, Interactive Fiction

1. INTRODUCTION

Prototyping is an essential step in the development of a software product. By prototyping their graphical user interface designs, software developers can test that their designs are intuitive and effective. The method that developers choose to prototype their graphical user interface (GUI) designs may impact the efficiency and effectiveness of the process. Choosing a method that is rapid while allowing for interactive testing can be difficult, as each method is associated with features and drawbacks that impact the prototyping process.

In this paper, I evaluate the uses, features, and drawbacks of paper prototyping and wireframe prototyping. I discuss the background of Interactive Fiction (IF) systems, followed by an introduction of a new method of interactive prototyping using the Inform 7 IF description language for rapidly prototyping GUIs.

Section 2 discusses the importance of user interface prototyping and review the general classification of prototyping methods. Sections 3 and 4 discuss the benefits and drawbacks of using paper prototyping and wireframe prototyping. Section 5 provides a summary of interactive fiction and its technologies. Section 6 describes the process involved in developing the GUInform prototyping system. Sections 7 draws conclusions and makes recommendations for future work.

I call this project the "GUInform" system for interactive prototyping. The main contributions of this work can be summarized as follows.

1. A method of rapid GUI prototype development using an IF platform
2. A system for textual interaction with GUI prototypes
3. The addition of SVG capabilities to Gargoyle, an IF environment
4. An extension for the Inform 7 description language that provides a toolkit of elements for creating GUI prototypes

2. BACKGROUND

Prototyping is an important stage in the software development process, and is used to incrementally improve a design prior to its implementation. While prototyping often takes significant time and effort to complete, its benefits far outweigh its cost. Prototyping allows the developers to correct ambiguities found during the requirements and specifications stage of the software engineering process [10]. Catching such ambiguities during the specifications process often requires less time and money to correct than if discovered during the development or production stages.

In applications with a graphical user interface (GUI), the prototyping stage involves viewing or interacting with a vi-

To appear as Portland State University Computer Science Department Technical Report #13-01, June 2013.

sual design of variable finality. In creating such a prototype, the developer is forced to consider the intended interface in more detail. An interface that may have seemed intuitive in concept may be revealed as too complex or difficult to navigate only once a visual, GUI prototype of the interface has been developed. As a result, time and effort can be saved if this realization can be made during the prototyping stage, rather than following the GUI's implementation. Interacting with a GUI prototype can be accomplished in multiple ways depending on the degree to which the user interface has been implemented. User interface prototyping can be accomplished using several approaches. Exploratory, experimental, and evolutionary prototyping are common methods [6].

2.1 Definitions

Several terms are recurrent throughout this paper. A developer is a person designing or implementing a GUI prototype, which is then evaluated by a test user. I refer to "interactive" prototyping as any prototyping process in which the GUI prototyping system provides the test user with real-time feedback. This feedback may be in the form of an updated image, pop-up window, change in page view, or other interaction between the prototype and the user. Each visual aspect of the interface, such as a button or drop-down menu, is referred to as an element. Finally, a "rapid" prototyping process allows a GUI prototype developer to easily create and modify their designs.

2.2 Prototyping Methods

The aim of exploratory prototyping is to develop a prototype that explores a potential solution to a posed problem. As a result, this prototyping method often results in a presentation prototype or functional prototype. A presentation prototype is used to demonstrate how a user interface design fits the user's specifications. A functional prototype is also used to demonstrate the user interface design, but also incorporates the usability and functionality of the proposed interface.

Experimental prototyping is focused on the usability of the interface. The most important result of this type of prototyping is a demonstration of how the interface is used, rather than how it appears to the user. As a result, a common deliverable of this type of prototyping is a functional prototype or breadboard. A breadboard represents the technical details of a prototype's functionality. Rather than represent the project as a whole, breadboards can be used to test and design particular portions of the project that must be evaluated for risk.

Finally, evolutionary prototyping is a method used to change and evaluate a usable prototype over time. Rather than a single project, evolutionary prototyping is focused on adapting a prototype to further its development. As a result, a common deliverable of this prototyping method is a pilot system. A pilot system is a refined prototype that can be incorporated into the final design, and represents an interface design that has already been refined and implemented.

Once a usable prototype, such as a functional prototype or pilot system, has been created, the developer may choose to present it to a test user, who may be a potential end-user of the product. After the developer has given the test user a task to complete, such as "Locate the product information page," the test user interacts with the prototype to complete the task. By reviewing or recording the test user's interactions with the prototype, the developer can learn whether the interface design is intuitive, effective, and meets the interface design criteria. Changes may then be made to enhance the prototype in response to this testing feedback.

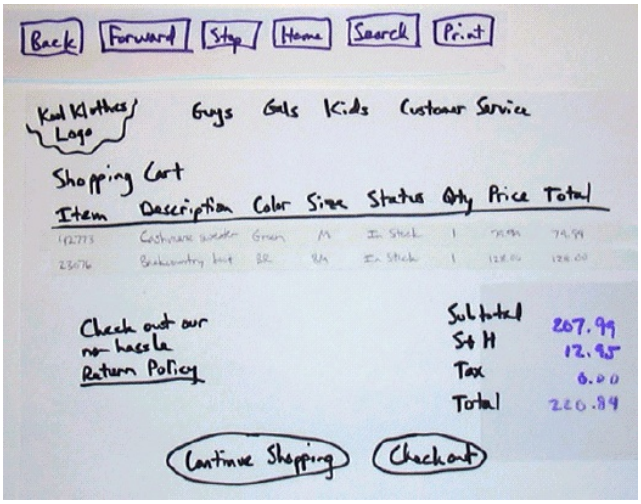

Figure 1: A Simple Paper Prototype [21]

3. PAPER PROTOTYPING

Paper prototyping is a common method of application prototyping for usability testing, and was first used in the early 1990's [23]. This method is a form of exploratory prototyping, as it is an easily implemented method of exploring potential user interface designs. The aim of this form of prototyping is to create a presentation prototype that may illustrate the proposed interface, but does not allow for easy usability or functionality testing. Creating a paper prototyping consists of using pen, paper, and other common craft supplies to create a prototype of the proposed user interface. Elements of the interface are drawn onto the paper. Figure 1 depicts an example of a paper prototype of a website's shopping cart page.

The complexity of paper prototyping depends on the developer's intentions and on the level of potential interaction with the prototype. For example, a set of tabbed index cards can be used to represent a tabbed interface, with each index card containing a drawing of its respective content in the interface. The content of drop down menus can be drawn directly onto the paper prototype, and text or graphical content can be written, drawn, or printed directly onto the paper prototype. Prototype elements can be drawn onto sticky notes and then rearranged to alter the prototype design. Figure 2 shows how tabbed cards can be used to represent the tabs in an interface design.

The ease of creating a paper prototype has led to it being the most commonly used prototyping method [9]. Two methods are common for testing a paper prototype. The first method involves asking a product user to draw how they

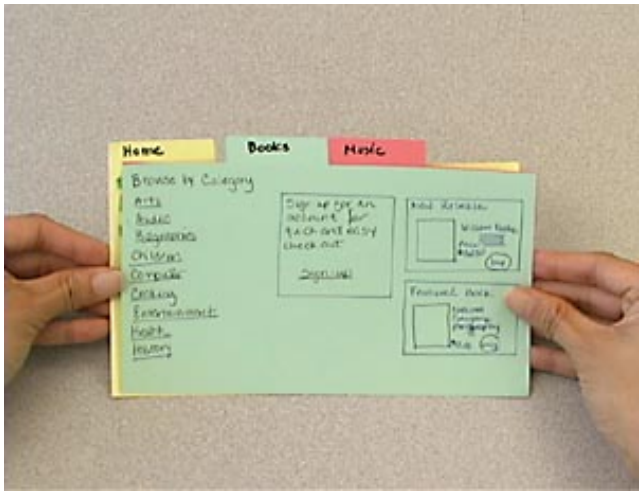

Figure 2: Using Tabs for Paper Prototype Interaction [24]

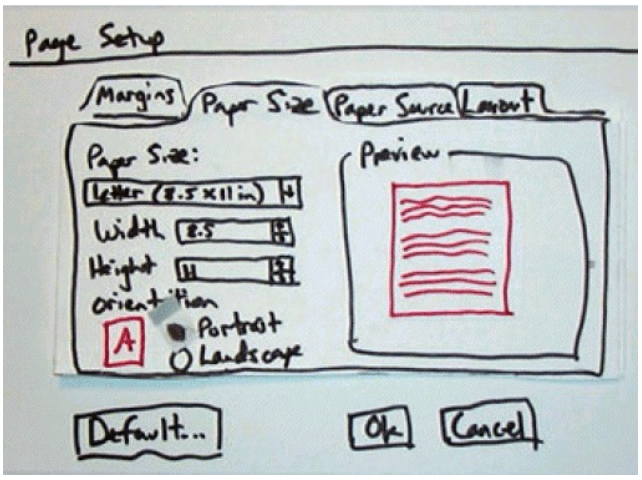

Figure 3: Using Static Tabs for Paper Prototyping [21]

would expect the interface to appear follow the completion of a specified action. This method allows the product developers and designers to understand how their users expect to use their product's interact, allowing them to design the interface accordingly.

The second method involves the creation of a paper prototype by the interface designer that is then presented to a product user. The user then interacts with the paper prototype in a way that corresponds to how they would interact with an actual interface of the same design [13]. Users perform tasks typical of the resulting interface product by interacting with a physical, paper version of the interface. To mimic a computerized interaction with the prototype, the paper prototype is manipulated using physical variants corresponding to interactions that would occur in the final interface [22]. To simulate clicking on a button drawn on the paper prototype in Figure 3, the user may tap the button drawing with a pencil or otherwise indicate 'clicking' on the button drawing.

This paper prototyping method allows developers to receive feedback regarding their user interface design by observing how the user manipulates the paper interface [23].

3.1 Advantages of Paper Prototyping

Paper prototyping has many benefits. First, paper prototyping is a method that is accessible to people without technical backgrounds who may find computer-based prototyping methods to be intimidating [11]. No programming experience is required to create prototypes, allowing designers of diverse backgrounds and roles to participate in the prototyping process. Second, multiple iterations of a prototype can be developed quickly. This is in contrast to the time cost associated with changing an implemented prototype between prototyping iterations, as the implementation form of prototyping requires that the implementation be potentially rewritten to account for changes in design. Paper prototyping allows the interface developer to completely recreate or edit a new interface with minimal effort. Finally, paper prototyping is an inexpensive form of prototyping, as low-cost materials are used in contrast to perhaps expensive prototyping software suites.

3.2 Disadvantages of Paper Prototyping

While paper prototyping has many benefits, there are also scenarios in which it is not a desirable prototyping method. Paper prototyping is useful for testing the visual aspects of an interface, but does not realistically simulate an interactive interface. Elements of interaction such as scrolling, text fonts, and graphics can be difficult to represent in a paper prototype. Additionally, it can be time consuming to create an interface in which design components are used repeatedly, as the design components must be re-drawn for each interface view. Elements of the interface that are common to multiple pages can be difficult to duplicate or reuse for multiple iterations of the same prototype. Finally, unless prototypes are drawn onto removable sticky notes, a new interface must be drawn when elements need to be changed or relocated during an iteration of the prototyping process.

Overall, paper prototyping is an appealing method of prototyping when few interfaces need to be created and a limited degree of interaction is required to test the prototype. This method is useful and effective for quickly prototyping an interface, but is not a good indicator of usability since it is not fully interactive, and is not ideal for prototyping multiple, frequently changing interfaces. Paper prototyping may be appropriate for the first stage of interface development when developers want to test a potential interface design. However, a different prototyping system may be needed to present the proposed interface to a test user.

4. WIREFRAME PROTOTYPING

Another method of interface prototyping is the development of a rudimentary implementation of the interface. This involves the creation of a wireframe prototype in the target programming language or using interface building software. This method of prototyping may involve writing one or more programs that would allow the basic functionalities of the prototype to be tested. In doing so, this method allows the

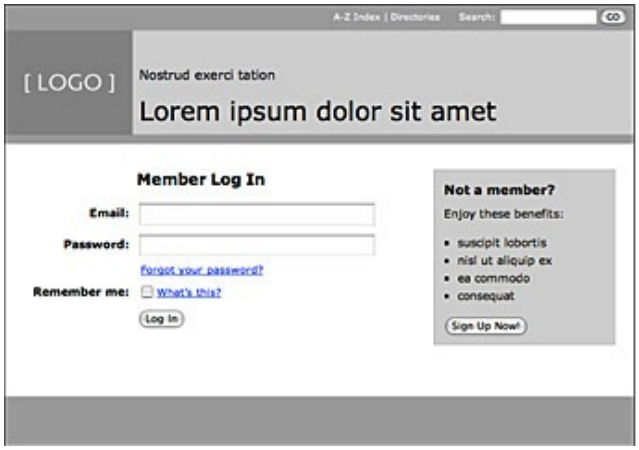

Figure 4: An HTML Wireframe Prototype [12]

Figure 5: A Balsamiq Wireframe Prototype [8]

developers to focus on the usability and interaction with components of the interface prototype, rather than specifics of its visual design. Basic shapes are used to represent functional elements of the eventual design, creating an interactive illustration of the proposed interface design [20], as shown in Figures 4 and 5.

A prototype developed using this implementation method would also be tested by a user guided by a test facilitator. Similar to the evaluation of paper prototypes, the facilitator and developers observe the user's attempts to complete a task specified by the facilitator. However, instead of simulating an interaction via a paper prototype, the user interacts with the prototype as if it were the final implementation of the product. The test facilitators can gain feedback from the user's interaction with the prototype by observing any difficulties the user has when attempting to complete the task. This allows the developers to view more realistic interactions with the interface prototype than if paper prototypes were used.

4.1 Advantages of Wireframe Prototyping

This method of implementing a more complete prototype allows the developers to determine whether the interactions associated with the prototype are intuitive. Figure 5 illustrates an wireframe prototype that can respond to user interaction such as button clicking.

If the user has difficulty using the interface to complete the task given by the test facilitator, the interface developers can adjust the prototype accordingly. Additionally, this prototyping method may reveal design constraints that could only be realized through the process of implementing the prototype, such as constraints of the target system that impact how the interface must be designed. When a change needs to be made to the prototype, the same prototype can be kept with minor edits to its source code, rather than recreating the entire prototype to incorporate a change. The prototype code can also be incorporated in the final implementation of the application should the prototype be successful.

4.2 Disadvantages of Wireframe Prototyping

While there are many benefits to wireframe prototyping, it is also a time consuming method. When creating a full or partial implementation of an interface, the developer must devote time to determining the implementation details of the interface design. This detracts from the purpose of interface prototyping, which is to create an intuitive design that allows the user to effectively use the software product rather than define the technical details of how the interface should be implemented. As a result, it is difficult to use this method to quickly design, test, and evaluate interface designs. Should a complete redesign of the interface be necessary, much development time is wasted, as the implementation of the interface is no longer usable for prototype design.

Overall, this method of prototyping by implementation may be best suited for testing interfaces that are already near their final form. Rather than being used in the initial design, creation, or testing phases of prototyping, this method would best be used in scenarios where the product is not at risk for a complete redesign and will require few edits.

4.3 Optimal Prototyping

After looking at these the paper prototyping and prototyping by implementation methods, two extremes of prototype development methods, it seems that the following aspects are desirable in a prototyping method:

- Fast initial implementation
- Easily altered to reflect changes in design
- Models interactions with the interface
- Accessible to those without technical backgrounds

An optimal prototyping method will cater to these needs of developers during the prototyping stage. While paper prototyping addresses the criteria of fast initial implementation and accessibility to those without technical backgrounds, it fails to accurately simulate interactions with the interface. Additionally, it is not as suitable for scenarios in which frequent alterations must be made to prototypes. Prototyping by wireframe implementation properly simulates interactions with the interface, but is extremely time consuming

to create initially and is not suitable for frequently changing designs. An optimal prototyping method will strike a balance between the two extremes of the paper and wireframe methods.

5. INTERACTIVE FICTION

Interactive fiction (IF) is a form of writing that involves the user by responding to the user's text statements or commands [14]. Text adventures, a form of IF, are a type of computer game that interacts with the player via text. In its standard use, an IF player simulates the exploration of a computerized environment without a visual representation, but rather though the use of textual dialogs [1]. The current situation is presented in the IF game as a textual, story-like description to which the user can respond with a variety of text commands. These interactions, both story dialog and text commands, occur through a console window that displays the situation description in the form of a text log and accepts input from the user via a text prompt. When the user enters a text command into the prompt, the IF game presents an updated situation story according to rules that are pre-defined by the game developer. Figure 6 demonstrates an interactive console as it would be used in an IF game.

The "Z-machine"[15] is the IF virtual machine developed by Infocom in 1979 to run large interactive fiction games on computers with limited available memory. The code it interprets is known as "Z-code", which is now the standard for running interactive fiction. Interactive fiction programs can be expressed compactly using Z-code, an object oriented language describing interactive fiction objects. A interpreter program on the user's personal computer must emulate the Z-machine and interpret Z-code as input to provide the game [3].

Since the Z-machine file format was designed for use on computers with limited memory, it does not allow the creation of games larger than 128K. The number of objects, attributes, and properties are also limited to keep the game size to 128K. Additionally, the Z-machine does not support the use of graphics, sound, various typefaces, and additional I/O features that are desirable to interactive fiction developers.

Glk is an API developed to handle I/O interactions with the text interfaces associated with interactive fiction [16]. It is designed to handle input from the user and output as designated by its associated virtual machine. The purpose of Glk is to provide a common I/O API for multiple versions of interactive fiction virtual machines, including the Z-machine and with multiple libraries.

The Glulx virtual machine specification[18] was created by Andrew Plotkin and Graham Nelson to address the limitations of the Z-machine specifications. Glulx is a lightweight virtual machine designed for use with the Inform language, a language specifically designed for creating interactive fiction. Glulx relies on the Glk I/O API and overcomes the memory constraint of the Z-machine by using 32-bit memory rather than 16-bit memory [17]. As a result of being lightweight and utilizing more available memory than the Z-machine, the Glulx virtual machine is advantageous for use in modern computers.

```
Gate Waiting Area A-1
You can see a Gate A-1 Door, an insurance
salesman, a flight attendant, a red suitcase
(empty), a blue table and a cookie here.

> examine suitcase
It's the salesman's suitcase, colored red. As you
look at it, the salesman begins to glare at you.

> go north

Terminal A-1 Commons
A airport terminal common area with the usual
restaurants, tables, vending machines, information
desks, restrooms, etc. To the south is the gate
waiting area.

You can see a television and an oak desk (on which
is a blue key) here.

> take key
Taken.

> go south

Gate Waiting Area A-1
You can see a Gate A-1 Door, an insurance
salesman, a flight attendant, a red suitcase
(empty), a blue table and a cookie here.

> unlock door
(with the blue key)
You unlock the Gate A-1 Door.

>
```

Figure 6: An Interactive Fiction Game Transcript

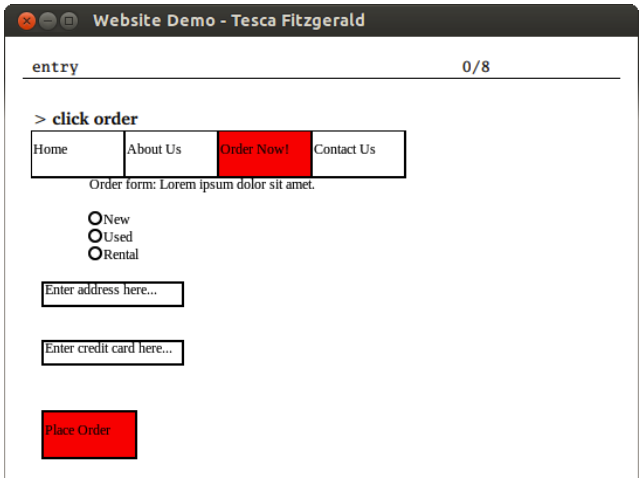

Figure 8: A web page prototype prior to receiving a text command

The Inform 7 language[19] is designed for the creation of interactive fiction using natural language programming. As a result, it allows a user without a technical background to create interactive fiction. The Inform compiler is designed to create code for both the Z-machine and Glulx virtual machines. Programs written in Inform 7 are created using the Inform IDE shown in Figure 7.

6. METHODS

My goal was to create a prototyping system that allows for the easy implementation of prototype designs while still providing an interactive experience to the user. I chose to use Inform 7 to implement this prototyping system because it allows for such interactions between the user and an interactive console, which could present the interface prototype. Additionally, Inform 7 does not require a technical background to use, increasing the number of potential users. It is also able to be used on multiple operating systems and does not use a large amount of memory.

As interactive fiction responds to text commands, the prototyping system would use text input from the user to interact with the prototype design. The idea, proposed by Bart Massey, was to define GUI elements using Inform and display them in an interactive fiction text-interface window as shown in Figure 8.

6.1 Architecture

The GUI prototype's description and interaction rules are implemented in Inform 7. These interaction rules contain references to commands defined in a custom Inform 7 extension that I developed. The layers of the system include the following: the GLK specification; the implementation of the GLK specification in an Inform 7 interpreter; and the Inform 7 extension that provides a definition of commands that call for SVG rendered in the interpreter. To support commands that result in SVG drawings, changes are needed for each of these layers. Figure 9 illustrates the architecture of the GUInform system.

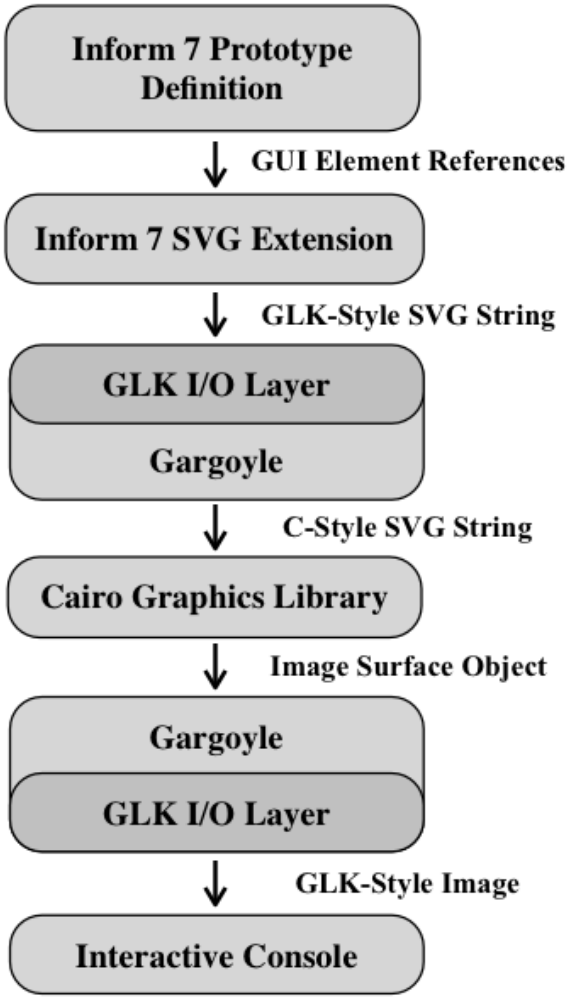

Figure 9: GUInform System Architecture

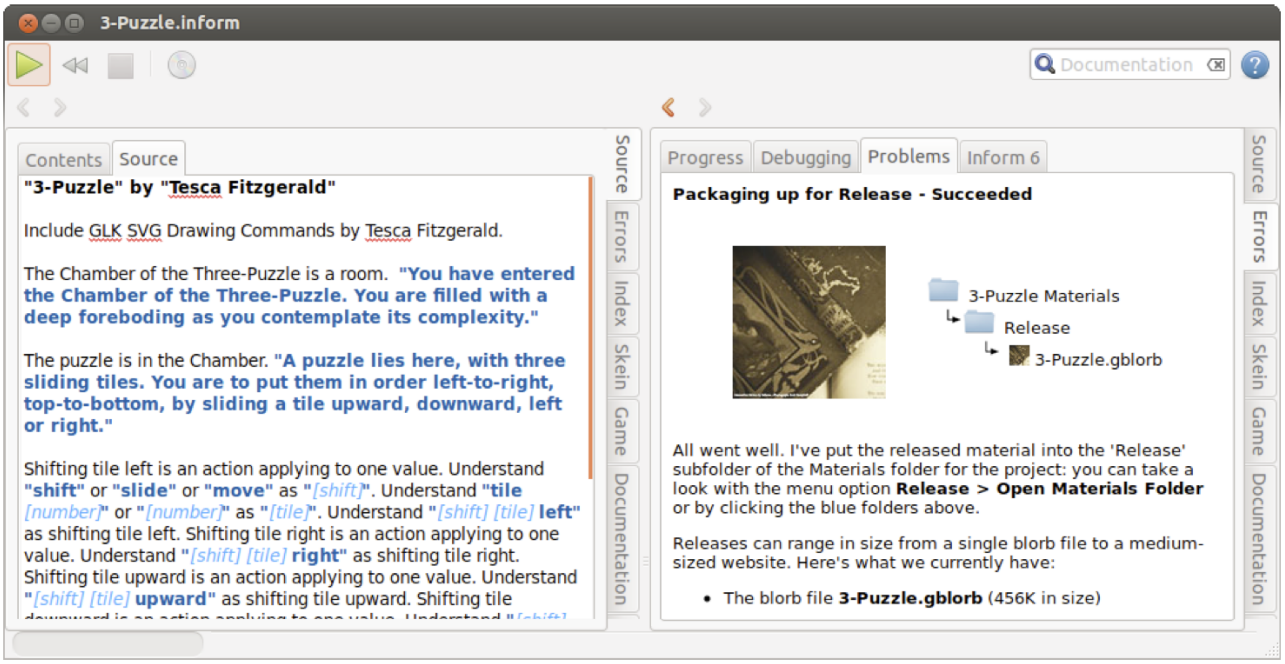

Figure 7: The Inform 7 IDE

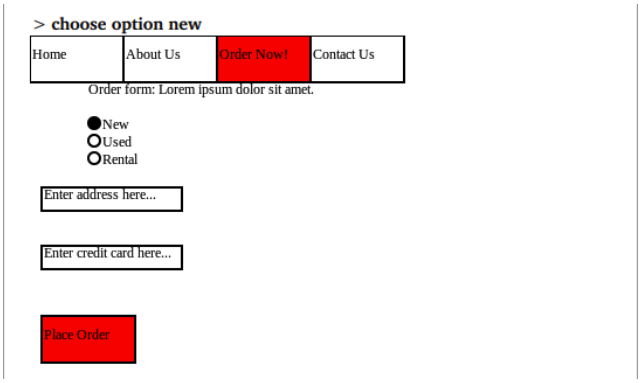

Figure 10: Updated Prototype in Response to a Text Command

6.2 Inform 7 Prototype Definition

Text commands are used to simulate actions performed on the prototype. As commands are entered into the interface window, the GUI elements are changed to reflect the results of interactions with the prototype. Figure 10 shows the same prototype as in Figure 8, but updated after a text command is entered.

To use the prototyping system, the developer must determine the interactions that can be performed on the prototype. These interactions are modeled as action rules in Inform, which allow an action to result from a text command entered by the user. Actions dictate how the prototype responds to various commands from the user. The Inform 7 code in Figure 11 demonstrates how an action may be defined for an interface prototype. This action responds to the text commands ”click home” and ”click home button” by

```
Clicking home is an action applying to nothing.
Understand "click home" or "click home button" as
clicking home.

Carry out clicking home:
        Display the home panel.
```

Figure 11: An Inform 7 Action Definition

displaying the home panel, which is defined elsewhere in the Inform 7 code. This home panel definition, along with multiple other panel definitions, comprise the visual elements of the prototype's interface design. Each panel contains a description of various visual GUI elements. The Inform 7 code in Figure 12 demonstrates the definition of a panel.

6.3 GLK I/O Layer and IF Interpreter

An interpreter is necessary to run Inform 7 programs as IF. Some currently available IF interpreters have graphics capabilities, enabling them to display a static image stored on disk. However, none of these interpreters support SVG rendering. This limitation led me to modify an existing IF interpreter to support SVG. I chose to modify Tor Andersson's implementation of the Gargoyle interpreter due to its portability on multiple operating systems, current support for static graphics, and availability as an open-source project [2].

The GLK specification is a standard held universally for all Inform 7 interpreters to ensure that basic functions are implemented. While this standard requires the implementation of various functions, it does not require every interpreter to support all features. For example, each interpreter

```
Section - The Basic Panel

The basic panel is a panel.

The basic-background is a white background. The x
is 0. The y is 0. The width is 140. The height is
210.

The result is a text field. The x is 10. The y is
10. The width is 120. The height is 30. The
content is "0".

To make the basic panel:
  Include the basic-background in the basic panel;
  Include the result in the basic panel;
```

Figure 12: A Panel Definition Example

may support static graphics display, or instead implement a function stating that the graphics feature is unavailable. While graphics are a part of the GLK specification, support for SVG is not. As a result, I edited the GLK specification to include the standard for a function that renders SVG text and displays the rendered image in the interactive interpreter window. The revised specification files are available in the GUInform project repository [7].

6.4 SVG Rendering

To render SVG strings, an SVG rendering library was needed. After considering multiple libraries, I decided to use the Cairo library because of its portability and implementation in C. Cairo is an open source project and provides a C API, which was necessary to render SVG directly from the Gargoyle code. Many other SVG rendering libraries contain extraneous image editing abilities that would not be needed in this project, such as image manipulation or rendering of multiple image formats. Additionally, Cairo is also heavily documented and easily available for any users who would be interested in downloading my version of Gargoyle to render SVG images. Cairo is available for download online [4].

Cairo works by creating a "surface object" from the SVG, which can then be equated to a GLK picture object's data. One option was to have Cairo convert the SVG to an equivalent PNG file which would then be saved in the game's directory. The Inform program would then read the PNG file and display it as it would any other image in its directory. This works in testing mode, but in the release mode, any figures to be used in the Inform game must be packaged into the release version game. Due to the way that interactive fiction games are packaged, any images to be displayed must be in the project at the time of packaging. This prevents the system from being able to dynamically create and subsequently reference PNG files created from SVG. Rather, I needed a way to directly access the rendered SVG object in memory and handle it using the default GLK methods. To do this, a new GLK picture object was created with a bitmap extracted from the Cairo surface object. This GLK picture object was then displayed using the originally provided graphics functions. These provided graphics functions display a given GLK picture object in the interactive interpreter window. This code is available in the GUInform project repository [7].

```
To draw a SVG rectangle of dimension (width - a
number) by (height - a number) at position (x - a
number) by (y - a number):
   render "<rect width='[width]' height='[height]'
   x='[x]' y='[y]'/>" as SVG.
```

Figure 13: Command to Draw an SVG Rectangle

6.5 Inform 7 SVG Extension

Having edited the GLK standard and the Gargoyle code, I created an Inform 7 extension designed to specifically handle SVG rendering requests and call the GLK function. This extension contains Inform 7 rules, such as the one in Figure 13, to call the GLK function for drawing SVG.

The purpose of the Inform 7 SVG extension was to provide a function that could call the GLK SVG function without forcing the Inform 7 programmer to be knowledgeable in writing SVG. Rather, the extension serves as an API for the GLK SVG function, allowing the programmer to write SVG commands using natural text in Inform 7. The "indexed text" data type is used to send dynamic text to the GLK function. Inform 7 indexed text is converted character-by-character to a C-style string, and then used as a parameter for the GLK SVG function [5]. This SVG handler extension is available for download at the GUInform project repository [7].

The GUI elements demonstrated in previous sections are not native to Inform 7. Rather, I created an Inform 7 extension that contains SVG strings that, once rendered, illustrate the intended GUI element. For example, in the previous Figure 12 example, the "white background" and "text field" references are GUI elements that are defined in my Inform 7 extension. The following elements are defined in the Inform extension: radio buttons; checkboxes; buttons; labels; drop-down menus; backgrounds; and text fields. The latter portion of the example, starting with "To make the basic panel", attributes the background and text field elements to the "basic panel" object.

Figure 14 illustrates how panels can be used multiple times within a single prototype. This example contains a panel of basic calculator buttons that is displayed when the calculator is in "basic" mode, and an additional panel containing buttons that are specific to the scientific calculator mode. Both the scientific button panel and basic button panel are displayed when the calculator is in "scientific" mode. An action is defined for this prototype such that both panels are displayed when a "choose scientific mode" command is entered, and only the basic button panel is displayed when the "choose basic mode" command is entered.

To interact with the prototype, the developer or test user enters text commands into the interactive console. When the user enters a command into the console, the prototype is updated according to the actions defined in Inform 7 by

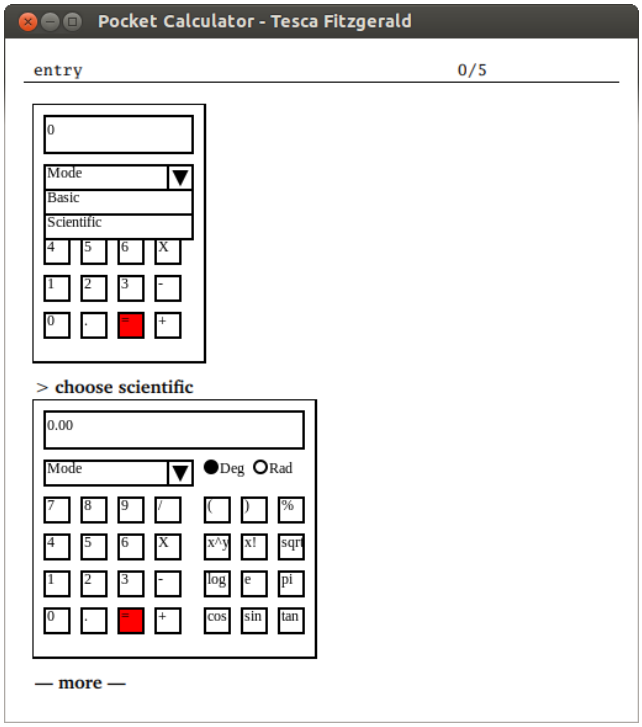

Figure 14: The prototype changes in response to a "choose scientific" command

the developer, as described in Section 6.2. The rendered GUI prototype and text commands from the user are not erased from the prototype window. This results in a log of commands and resulting prototype states that is created in the prototype window. This log can be used to record a prototype testing session for use in developing later revisions of the prototype. Figure 15 illustrates a log created by interacting with the timer prototype. The Inform 7 definition for this prototype is provided in the appendix.

7. CONCLUSIONS

Overall, I found that using the interactive fiction framework for GUI prototyping is an effective method of quickly creating interactive prototypes. The use of SVG to create these visual prototypes allows a prototype designer to create dynamic prototypes that react to a given text command. The main contributions of this work are as follows:

- A method of rapid GUI prototype development using an IF platform
- A system for textual interaction with GUI prototypes
- The addition of SVG capabilities to an IF environment
- An extension of the Inform 7 description language that provides a toolkit of elements for creating GUI prototypes

A low-level GUI implementation is not required for developers to interact with their prototypes. Rather, the developer can define action rules in the GUInform system that dictate

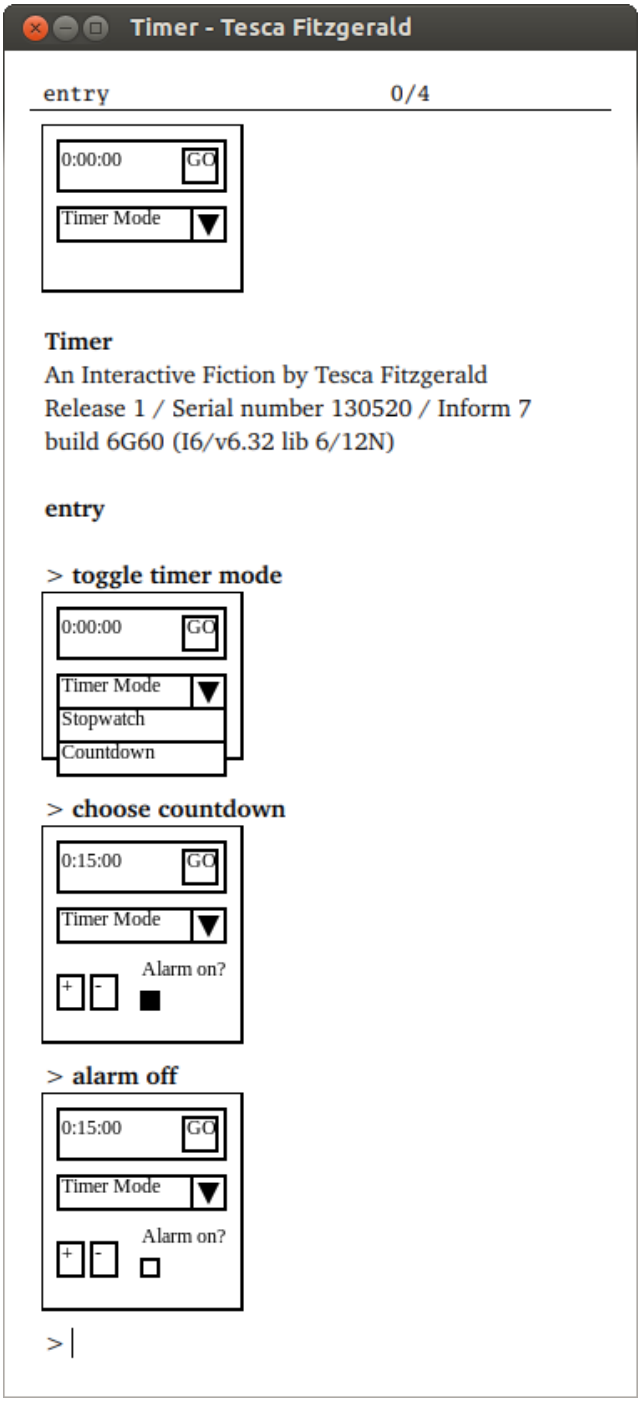

Figure 15: A Prototype Interaction Log

how the prototype responds to user interaction. Additionally, prototypes can be designed quickly without the use of the paper prototyping method. While paper prototyping is a simple and effective method, it does not allow for a realistic interaction between the user and the prototype. The GUInform system allows the developer to experience a more realistic interaction with the prototype that incorporates drop-down menus, radio buttons, and other standard interface elements. Finally, this work demonstrates how interactive fiction platforms can be used for purposes other than creating text adventure games. The incorporation of SVG into the interactive fiction platform allows developers to use Inform 7 for interactive graphics purposes.

8. FUTURE WORK

Future work on this project includes further testing and incorporation of the GUInform system into a realistic software engineering process. Additionally, I would like to incorporate layouts and other methods specifying a GUI prototype element's location, as the current method of specifying each elements' coordinates is tedious.

I would like to measure the efficiency of GUInform prototyping compared to other methods. I plan to receive feedback regarding this prototyping method from software developers who have also had experience using other prototyping method, namely paper and wireframe prototyping. One efficiency test could be performed by requiring three developer groups to prototype the same design using the paper, wireframe, and GUInform prototyping methods. The development time and quality of the prototypes resulting from each method could then be compared.

9. ACKNOWLEDGMENTS

Many thanks to my advisor, Professor Bart Massey, for proposing the GUInform concept, and for his continual feedback throughout this project. This project would not be possible without his guidance and support.

I also thank Professor Lois Delcambre for her suggestions regarding ways I could test the GUInform system in the future. Thank you Brady "EmacsUser" Garvin for providing the Inform 6 code that converts SVG strings from the Inform 7 indexed text format into GLK C-strings, making the GUInform system possible. I would also like to thank Graham Nelson for developing the Inform 7 description language, Andrew "Zarf" Plotkin for developing the GLK specification and Glulx virtual machine, and the developers of the Gargoyle Interactive Fiction interpreter.

10. REFERENCES

[1] About interactive fiction. http://www.inform7.com/if. Accessed 2013-05-25.

[2] garglk - a cross-platform io layer for an interactive fiction player. http://code.google.com/p/garglk/. Accessed 2013-05-27.

[3] How to fit a large program into a small machine. http://www.csd.uwo.ca/Infocom/Articles/small.html, Jul 1995. Accessed 2013-01-03.

[4] Cairo. http://www.cairographics.org, March 2012. Accessed 2013-05-28.

[5] Indexed text to glk function using if6. http://www.intfiction.org/forum/viewtopic.php?t=5518&p=40129, August 2012. Accessed 2013-05-28.

[6] Dirk Baumer, Walter R. Bischofberger, Horst Lichter, and Heinz Zullighoven. User interface prototyping–concepts, tools, and experience. In *Proceedings of the 18th international conference on Software engineering*, ICSE '96, pages 532–541, Washington, DC, USA, 1996. IEEE Computer Society.

[7] Tesca Fitzgerald. http://github.com/TescaF/GUInform. Accessed 2013-05-28.

[8] Gus Fraser. Balsamiq vs powerpoint storyboarding with vs 2012. http://techblurt.com/2012/09/13/balsamiq-vs-powerpoint-storyboarding-with-vs-2012/, September 2012. Accessed 2013-05-19.

[9] Tracy Frayne. Tools of i.a. focus on prototyping. http://csusap.csu.edu.au/~tfrayn01/paperprototyping.html, June 2010. Accessed 2013-05-27.

[10] Hassan Gomaa and B.H. Scott Douglas. Prototyping as a tool in the specification of user requirements. *IEEE*, pages 333–342, 1981.

[11] J.A. Landay and B.A. Myers. Sketching interfaces: Toward more human interface design. *IEEE Computer*, 34, 2001.

[12] Patrick Lynch and Sarah Horton. The design process. http://webstyleguide.com/wsg3/10-forms-and-applications/4-design-process.html. Accessed 2013-05-28.

[13] Shawn Medero. Paper prototyping. http://alistapart.com/article/paperprototyping, January 2007. Accessed 2013-05-23.

[14] Stuart Moulthrop and Nancy Kaplan. Something to imagine: Literature, composition and interactive fiction. *Computers and Composition*, 9(1):7–23, Nov 1991.

[15] Graham Nelson. The z-machine standards document. http://www.gnelson.demon.co.uk/zspec/, June 1997. Accessed 2013-01-03.

[16] Andrew Plotkin. Glk api specification. http://eblong.com/zarf/glk/glk-spec-074.txt. Accessed 2013-01-03.

[17] Andrew Plotkin. Glulx: A 32-bit virtual machine for if. http://www.eblong.com/zarf/glulx/, October 2012. Accessed 2013-01-03.

[18] Andrew Plotkin. Glulx specifications. http://www.eblong.com/zarf/glulx/glulx-spec_0.html, October 2012. Accessed 2013-01-03.

[19] Aaron Reed. *Creating Interactive Fiction with Inform 7*. Course Technology Press, Boston, MA, United States, 1st edition, 2010.

[20] Garr Reynolds. The importance of wireframes in web design and 9 tools to create wireframes. http://www.onextrapixel.com/2009/07/15/the-importance-of-wireframes-in-web-design-and-9-tools-to-create-wireframes/, July 2009. Accessed 2013-05-27.

[21] Caroline Snyder. Paper prototyping. http://www.snyderconsulting.net/article_paperprototyping.htm. Accessed 2013-05-28.

[22] Caroline Snyder. Using paper prototypes to manage risk. http://www.uie.com/articles/prototyping_risk/. Accessed 2013-01-02.

[23] Carolyn Snyder. *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interface*. Morgan Kaufmann, 1st edition, April 2003.

[24] Terry Winograd and Bill Verplank. Paper prototyping - climate control. http://hci.stanford.edu/courses/cs247/2009/handouts/paper-2009-exercise.html, 2009. Accessed 2013-05-28.

APPENDIX

[Prototyping Example]

The following is an example of a GUI prototype written in Inform 7 using the GUInform system.

```
"Timer" by "Tesca Fitzgerald"

Include GUI Library by Tesca Fitzgerald.

Section - Setup

The entry is a room.
The current panel is a text that varies.

When play begins:
        Make the stopwatch panel;
        Make the countdown panel;
        Now the current panel is "
            stopwatch";
        Display the stopwatch panel.

Section - The Stopwatch Panel

The stopwatch panel is a panel.

The stopwatch-background is a white
    background. The x is 0. The y is 0.
    The width is 120. The height is 100.

The display-field is a text field. The x
    is 10. The y is 10. The width is 100.
    The height is 30. The content is
    "0:00:00".

The go-button is a blue button. The x is
    85. The y is 15. The content is "GO".
    The width is 20. The height is 20.

The mode list is a dropdown. The x is 10.
    The y is 50. The width is 100. The
    height is 20. The content list is {"
    Stopwatch", "Countdown"}. The title is
     "Timer Mode".

Toggling mode is an action applying to
    nothing. Understand "toggle mode" or "
    click mode"  or "mode" as toggling
    mode.

Carry out toggling mode:
        Toggle the mode list;
        If the current panel matches the
            text "stopwatch", display the
            stopwatch panel;
        Otherwise display the countdown
            panel.

Selecting mode is an action applying to
    one topic. Understand "select [text]"
    or "choose [text]" as selecting mode.

Carry out selecting mode:
        If "[the topic understood]"
            matches the text "stopwatch":
                Toggle the mode list;
                Now the current panel is "
                    stopwatch";
                Now the content of the
                    display-field is
                    "0:00:00";
                Display the stopwatch
                    panel;
        Otherwise if "[the topic
            understood]" matches the text
            "countdown":
                Toggle the mode list;
                Now the current panel is "
                    countdown";
                Now the content of the
                    display-field is
                    "0:15:00";
                Display the countdown
                    panel.

To make the stopwatch panel:
        Include the stopwatch-background
            in the stopwatch panel;
        Include the display-field in the
            stopwatch panel;
        Include the go-button in the
            stopwatch panel;
        Include the mode list in the
            stopwatch panel.

Section - The Countdown Panel

The countdown panel is a panel.

The countdown-background is a white
    background. The x is 0. The y is 0.
    The width is 120. The height is 130.

The plus-button is a white button. The x
    is 10. The y is 90. The content is
    "+". The width is 15. The height is
    20.

The minus-button is a white button. The x
    is 30. The y is 90. The content is
    "-". The width is 15. The height is
    20.

The alarm-label is a label. The content is
     "Alarm on?". The x is 60. The y is
    90.

The alarm option is a checkbox. The x is
    60. The y is 100. The toggle is 1.

Checking box is an action applying to
    nothing. Understand "toggle checkbox"
    or "check" or "uncheck" or "alarm on"
    or "alarm off" or "click checkbox" as
    checking box.

Carry out checking box:
    Toggle the alarm option;
    Display the countdown panel.

To make the countdown panel:
        Include the countdown-background
            in the countdown panel;
        Include the plus-button in the
            countdown panel;
        Include the minus-button in the
            countdown panel;
        Include the alarm-label in the
            countdown panel;
        Include the alarm option in the
            countdown panel;
        Include the stopwatch panel in the
             countdown panel;
        Remove the stopwatch-background
            from the countdown panel.
```