5-1-2015

Creating an Immersive and Entertaining Game for Raising Internet Privacy Awareness

April Suknot

April Suknot
Candidate

Computer Science
Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

Patrick Gage Kelley , Chairperson

Lydia Tapia

Joel Castellanos

CREATING AN IMMERSIVE AND ENTERTAINING GAME FOR RAISING INTERNET PRIVACY AWARENESS

by

APRIL SUKNOT

PREVIOUS DEGREE
B.S., Computer Science, University of New Mexico 2013

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Computer Science

The University of New Mexico
Albuquerque, New Mexico

May, 2015

CREATING AN IMMERSIVE AND ENTERTAINING GAME FOR RAISING INTERNET PRIVACY AWARENESS

by

April Suknot

B.S., Computer Science, University of New Mexico 2013
M.S., Computer Science, University of New Mexico 2015

ABSTRACT

The continuous evolution in the pervasiveness and connectedness of technology is a subject of growing concern. Databases, smartphones, tablets, laptops, wearables are all united through a single global network—the Internet. This growth has resulted in social networks and other data-collecting applications becoming a major part of life for many people. To address emerging threats and growing concerns, our goal is to present internet users with a simulation that helps them consider the actions that they take when online. To do so, we have created a narrative game, *Immaculacy*, which is intended to raise awareness in common privacy issues that many internet users encounter.

Immaculacy is an interactive story that is set in a slightly dystopian future with a world that is littered with privacy issues. We have completed the prologue and first act of the game to serve as a demonstration for testing purposes. The demo consists of twenty-five scenes, three mini-games, and features twenty-one unique characters. Events unfold within the game based on hidden scores that are maintained throughout gameplay. In total, there are four scoring scales that we maintain: data leaking, government suspicion, character morality, and reputations with other major characters. These scores are calculated based on specific decisions made by the player both in dialogue and in

interactions with the world. Throughout the narrative, we allow the player to experience many privacy issues through their explorations of a world filled with hyper surveillance and connectivity. To verify the entertainment value of *Immaculacy*, demonstrations and informal playtesting have been conducted in which feedback was received in improvements that can be made to the game mechanics and interface. Additionally, we are beginning formal user tests to test the educational value of the game and to further test how entertaining *Immaculacy* is. Ultimately, we aim to create an engaging environment which presents players with situations and choices intended to encourage them to consider their own personal behavior when using online services.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Internet privacy is a growing concern worldwide as new networked devices and software are introduced. There are several new harmful consequences of this growth. Some consequences result from threats that have emerged. Some of these threats can include malicious attacks, some of which require a certain level of ignorance or naivety by an individual that allow an attacker to gain access to sensitive data. In addition, some consequences are the result of an individual's own behavior within internet provided forums. No matter the threat, the results can severely impact an individual through professional, financial, personal, and even physical damage.

Immaculacy, which is introduced in *Immaculacy: A Game of Privacy* (Suknot et al.) was created as a way to combat these threats. The stories, choices, and mini-games that make up *Immaculacy* have the primary intention of raising awareness in the ways that attacks can be delivered. The game targets a broad audience of internet users. Some specific examples include:

- people who are interested in data privacy and security but hope for an experience that is more in-depth than a simple quiz,
- individuals who enjoy using the internet and sharing on social networks,
- teenagers,
- people who enjoy stories with a dystopian theme,
- those who spend a great deal of time reading digital comics,
- social networking educators at both the university and grade school levels, and

- people who simply enjoy playing games.

Ultimately, we did not attempt to limit the target audience in the hope that we can get people thinking about privacy-related attacks.

In reviewing previous work in the area of privacy education with delivery through a game, there was a noticeable lack of immersive games. The games *CounterMeasures* (Jordan et al.) and *CyberCIEGE* (Irvine et al.) are quiz-based games that give the player choices in different scenarios, offering them a chance to choose a correct or incorrect answer. With *Immaculacy* the approach taken is to give the player realistic scenarios in which the right answer may not be clearly defined. Moreover, *Immaculacy* offers the player decisions in which the more secure option may not seem like a "kind" or "nice" choice. This may place the player in a position in which he or she must make difficult decisions that may not be popular among interacting characters. By implementing the game in this way, the hope is that the player will have the opportunity to learn about privacy attacks and make mistakes with sensitive information in a safe environment. In addition, the player can use this environment to learn that privacy choices are not inherently bad choices but that there are often tradeoffs in the decisions that are made.

Games are discussed by Denning et al., as well as Irvine, Thompson, and Allen, and Jordan et al. Like *Immaculacy*, they have a story-based style. However, these games are tailored more to people in fields working in cyber-security. Due to the effectiveness of these games, as discussed in more detail in Chapter 2, a similar approach was taken when creating and designing the story and games making up *Immaculacy*. To test if *Immaculacy* is comparatively effective for its audience, user tests are being started. In addition, informal playtesting and demonstrations have been conducted on the first part

of the game to assist the iterative development of the game by giving insight into the entertainment value of the game, effectiveness of the game mechanics, and aesthetic value of the user interface.

Chapter 2

Related Work

As introduced in Chapter 1, there are several state of the art games that have features similar to *Immaculacy*. Some of the games have similar delivery or gameplay, while others have similar content. All games that were reviewed involved computer security or privacy subject matter, to varying extents. Section 2.1 discusses these games in detail and explains similarities and differences to *Immaculacy*.

Moreover, the study in computer privacy discussed by Felt et al., as well as the one by Young and Quan-Haase were reviewed in order to gain knowledge useable for creating the game content for *Immaculacy*. In addition to privacy content, some of the user studies and iterative design processes discussed by Kelly et al. and by Kelley, Cranor, and Sadeh were taken into consideration when approaching the design and testing for *Immaculacy*. Section 2.2 will cover these studies and influences that they had on the development process.

Section 2.1 – Related Games

With the intent of gaining perspective on the types of games that already exist for privacy education, several computer and card based games pertaining to this topic were reviewed. Section 2.1.1 discusses games that have a similar format to *Immaculacy*, in that the game is story-based and intends to immerse the player in a realistic environment. This enables the player character to complete tasks to maintain a secure environment. Section 2.1.2 discusses games that have similar content as they target a more broad audience of average internet users. Finally, Section 2.1.3 discusses games that are not necessarily meant for education or raising awareness, but have a similar tone to *Immaculacy*.

Section 2.1.1 – Format

CounterMeasures (Jordan et al.), is intended to teach computer security practices to professionals entering a cybersecurity field. *CounterMeasures* offers its players *training missions* and *live missions* through which they have to work through different security-related scenarios. For this game, the training missions assist the player in learning by focusing on one topic in computer security and instructing the player during the mission. Live missions concentrate on testing the players' skills with more complex missions involving several security topics. The live missions require that the player rely on lessons from previous missions and thus, do not provide instruction. The interface presented in the game is a console in which the player types in commands for missions, along with several other panes that display objectives and other information to help the player complete the mission like hints and learned commands. A similar game, *Virtual Werewolves* was developed by Ensafi, Jacobi, and Crandall with the intent of using the game to augment computer security courses taught at the university level.

To evaluate the effectiveness of *CounterMeasures*, a user study was conducted that involved assessing game scores of the players and a questionnaire given before and after gameplay to gauge an increase in the player's level of knowledge. Results of the testing showed that most of the users did experience a significant increase in score on the questionnaire (Jordan et al. 5). *Virtual Werewolves* was evaluated in the classroom and was found to provoke enthusiasm in students toward computer security and was able to challenge students at various levels of experience (Ensafi, Jacobi, Crandall. 5-8).

The game *CyberCIEGE* (Irvine et al.) is presented as using an environment in which characters act as virtual users in a professional environment. *CyberCIEGE*,

challenges the player to take on the role of a decision maker in the virtual organization's IT department. The simulation engine created for the game allows players to create scenarios that may be encountered in an IT department. Similar to *CounterMeasures*, *CyberCIEGE* is targeted toward people pursuing a career in information assurance (IA), rather than the typical internet user.

The game *Control-Alt-Hack* (Denning et al.) is presented as a tabletop card game that is aimed toward teaching computer security to its players. Similar to *CyberCIEGE* and *CounterMeasures*, *Control-Alt-Hack*, while appealing to a wide audience, is more appropriate for players who are pursuing an education in computer science and engineering. Also similar to the games already discussed, *Control-Alt-Hack* gives its players different scenarios to play through as missions described on each card. This approach allows players to encounter multiple missions each time they played the game, requiring them to build and use a variety of skills required for computer security.

For *Control-Alt-Hack*, a user study was conducted to test the effectiveness of the game. In addition, feedback surveys from educators who had copies of the game distributed to them were conducted (Denning et al. 6-10). Through the user studies, the authors were able to find that people surveyed felt that the game had the most value in the areas of awareness and social engagement. Some of the critiques received were in the area of complexity, length, and entertainment value of the game. The game received additional critiques questioning the educational value of the game for the specified target audience. Some players with some experience with computer science commented that they expected to be challenged more than they were by the game. Adam Shostack, one of

the creators of *Control-Alt-Hack* also took part in creating a card game for threat modeling called *Elevation of Privilege*.

Also immersing a player in a story, *CounterMeasures*, *CyberCIEGE*, and *Control-Alt-Hack* are similar to *Immaculacy*. *Immaculacy* employs a narrative interface, discussed further in Chapter 3. The studies performed in evaluating *CounterMeasures* and *Control-Alt-Hack* showed that a story-based format can be used to help people pursuing careers in computer security or IA gain knowledge in topics within that field. In addition, the topics in the questionnaires for these user studies were topics that could be considered when approaching development for *Immaculacy*. For example, regarding *Control-Alt-Hack*, some of the major critiques were toward the complexity, length, and entertainment value of the game. It can be seen how these categories can apply to *Immaculacy* as well. We were able to gauge *Immaculacy*'s delivery within these categories through internal and informal playtesting, which will be discussed in more detail in Chapter 4.

Section 2.1.2 – Content

While the format of the games presented in Section 2.1.1 are most similar to *Immaculacy*, the games *Anti-Phishing Phil* (Sheng et al.) and *Mission: Laptop Security* have a more similar subject matter. *Anti-Phishing Phil* is a game in which the player assumes the role of Phil, a small fish, who must identify potentially malicious web addresses in the form of worms floating in the water. *Anti-Phishing Phil*, like *Immaculacy*, is targeted at average internet users. Though, unlike *Immaculacy*, the subject matter of *Anti-Phishing Phil* is specific to phishing attacks.

Also similar to *Immaculacy*, an iterative design process was taken in developing *Anti-Phishing Phil* (Sheng et. al 7-9) which considered feedback in approaching each iteration of development. The first stage was a pilot test in which the authors evaluated their participants' ability to identify phishing websites both before and after playing the game. The authors found a decrease in false negative responses, though there was an increase in the false positive answers. From this study, the authors were able to modify the game based on the feedback received, then conduct a second user test.

The second user test conducted for *Anti-Phishing Phil* was intended to test the condition of the game's training material, tutorial, and the condition of the game itself. In this study, the authors recruited participants who could be considered "non-experts" in regard to knowledge of computers. Again, the testing was conducted using pre- and post-testing each participant with questionnaires containing a combination of legitimate web addresses and spoof addresses. The study showed that there was an improvement for most of the sample addresses, between pre- and post-testing, in correctly identifying real and spoof addresses. More importantly, the user study helped to identify areas in which the game could be improved. For example, the studies revealed that some users had a tendency to falsely identify real addresses as spoofs after playing the game. Though, overall the studies found that interactive games do have potential to be an effective means to educate people in being less susceptible to phishing attacks.

Mission: Laptop Security was launched by the US Office of Justice Programs. The web-based game in which the player assumes the role of an agent who has lost a laptop with sensitive information on it. As the agent travels with a new laptop, various multiple-choice questions related to physical security are asked. If all questions are

answered correctly, the agent is considered fit to be allowed back on the force. One question was on the topic of password security. Though, most of the questions involve storage or handling of the laptop itself while the agent is performing a certain task, like going through airport security or when using the laptop at a coffee shop.

With physical security of passwords, documents, and other items being addressed in *Immaculacy*, *Mission: Laptop Security* provides an interesting example of how physical security topics, and security topics in general, can be approached. While the game presents the questions in a multiple choice format and there is no real dialogue in the game, it does present each question as part of a larger story. Moreover, *Mission: Laptop Security* is also targeted at average computer users. In those ways, the game is similar to *Immaculacy.*

Section 2.1.3 – Tone

Two other games were reviewed that are not intended to be educational games but have a similar tone and entertainment value that was aimed for in the development of *Immaculacy*. These games, *Blackbar* by Mrgan and Moore and *TouchTone* by Mikengreg both deal with surveillance by a large government.

In *Blackbar* the player character (PC) is the friend of a girl who is living in a community that has constant surveillance in order to censor information that is sent in and out of the city. The object of the game is to fill in the blanks in each letter of the friend's correspondence to unravel events in the story. Similarly, the object of *TouchTone* is to solve visual puzzles that increase in difficulty. Then, for each puzzle solved, part of the story in the game is uncovered. The story eludes to an agency that is conducting surveillance on people.

Blackbar and *TouchTone* are reviewed by Welch and Squires, respectively. The reviews for both touch on the dark and dystopian themes in the games. These themes, which leave the player feeling vulnerable or like they are being watched, is partly the feeling that we hope to capture with *Immaculacy*. In Chapter 4, Section 4.3, there will be a description of how we plan to achieve this tone in the game by gaining feedback through user tests. A summary of the major similarities and differences between *Immaculacy* and all games discussed in this section can be seen below in Table 1.

Compared Games	**Major Similarity**	**Major Difference**
CounterMeasures *CyberCIEGE* *Control-Alt-Hack*	These games are given a more immersive, story environment to present the material that is being taught.	The content of these games is tailored to a more specialized audience, either within cybersecurity fields or at least with a strong technical background.
Anti-Phishing Phil *Mission: Laptop Security*	The content of these games is in the area of privacy that are applicable to a very broad audience.	The format of these games is presented in a more quiz-like environment in which the player is immediately told whether or not a correct choice was made.
Blackbar *TouchTone*	The tone of these games is focused on some level of an invasion of privacy and are meant to give the player an eerie feeling.	These games are meant for entertainment purposes, as opposed to training their players, or raising awareness in some way.

Table 1 - Summary of the games compared to Immaculacy in this section.

Section 2.2 – Related Studies

In addition to privacy and security related games, studies in privacy were assessed. The intent was to begin identifying topics that should be covered and to gain perspective on concerns of users in regard to computer privacy. Studies used to influence the subject matter of the game are discussed in Section 2.2.1. Papers reviewed that aided the design and development process are covered in Section 2.2.2.

Section 2.2.1 – Subject Matter

The studies conducted by Felt et al. and Young et al. surveyed users about privacy concerns regarding various types of devices and networking environments. In *I've Got 99 Problems, But Vibration Ain't One: A Survey of Smartphone Users' Concerns*, Felt et al. were interested in learning about concerns that smartphone users have regarding app privileges. Through a user test, they questioned smartphone users about their level of concern for a series of different application privileges. From this study, the types of privileges were rated either low, medium, or high. An example of a low-risk application privilege would be vibrating the phone. A high-risk privilege involves actions like editing contacts on the phone (e.g. deletion) or reading and sharing text messages found on the phone.

In *Information Revelation and Internet Privacy Concerns on Social Network Sites: A Case Study of Facebook*, Young et al. conducted a case study of the social networking site. Through the case study, they hoped to learn about how younger Facebook users conduct themselves on the site. Information revealed by this study helped in identifying concerns that Facebook users have regarding privacy in addition to actions that they take online that may be compromise their privacy. In the area of user concerns, some topics include unwanted audiences and privacy concern in general. Regarding behavior on the site, the survey revealed trends in level of information revelation, profile visibility, personal network size, and frequency of Facebook use among surveyed users.

Both studies discussed in this section helped us gain a great deal of perspective on how to approach many of the quests or situations that are presented in *Immaculacy*. The privileges and risks discussed by Felt et al. helped with approaching choices within the

game, in addition to the functionality used for the main character's own smartphone, discussed further in Chapter 3. The concerns identified by Young et al. assisted in identifying story elements that can be added to help the player feel more like the main character's privacy is highly vulnerable. In addition, the information presented by Young et al. regarding user behavior on these social networking sites provided insight into how consequences of certain behaviors, like oversharing information, can be addressed.

Section 2.2.2 – Design and Development Process

We also reviewed studies conducted by Kelly et al.: *A "Nutrition Label" for Privacy* and by Kelley, Cranor, and Sadeh: *Privacy As Part of the App Decision-Making Process* used an iterative process to design privacy policy labels and application permission screens, respectively. Through user tests, the authors were able to come up with effective designs to enhance readability and understanding of the information presented. We were also able to use the participant feedback in these studies when approaching the design of our user interface as some of the feedback pertained to design in general. Furthermore, the results provided by Kelley, Cranor, and Sadeh gave some insight to user mindset when given decisions on devices that may affect the user's privacy.

We were also able to take a similar approach to these in designing the user interface for *Immaculacy* through feedback received during informal playtesting and demonstrations, as discussed further in Chapter 4, Section 4.2. Additionally, as will be discussed further in Section 4.3, we hope to use formal user tests to aid the development process regarding the educational value of the game.

Chapter 3

Methods

Immaculacy is a narrative game that can best be described as an interactive graphic novel. Choices that the user makes at any time in the game can affect the way that events unfold within the storyline. Section 3.1 fully describes the game including the plot of the game and features that can be found within. The design and development process is discussed in full detail in Section 3.2.

Section 3.1 – Game Description

The development process of this game began with choosing the format in which the game should be presented. These choices are discussed in detail in Sections 3.1.1 and 3.1.2. In order to immerse the player into the role given in *Immaculacy*, an entertaining and engaging story must be presented. An overview of the story is given in Section 3.1.3. Finally, to give the game entertainment value, the game's mechanics and features play a very important role. The mechanics and features chosen for *Immaculacy* will be discussed in Section 3.1.4.

Section 3.1.1 – Genre

Since the intent behind *Immaculacy* is to immerse the player in a relatable story, a role-playing genre was chosen for the game. To allow simplicity in the interface, the game was designed to appear like a comic book. However, unlike a book, *Immaculacy* allows interactivity with the world in the game through various features to be discussed further in Section 3.1.3.

The genre chosen for the story is closest to science fiction, taking place in a somewhat dystopian society. We chose to set the game in the near-future, rather than

present day or in past years. This decision was made to allow the technology in the story to be more prevalent and advanced than has yet been seen in society. However, we did not want to set the game in the far future because we still wanted the technology in the game to be possible or near possible, thus enabling the game to be relatable.

Section 3.1.2 – Platforms

Because many of the issues tackled by *Immaculacy* are intended to reach a broad audience of average internet users, the intent in the development was to have several means of distribution. The primary target platform for the game is mobile devices like smartphones and tablets. This *LibGDX* mobile game engine facilitated the environment with which this game was developed. *LibGDX* is primarily used for Android development and thus, uses the *Android SDK* development environment and the Java programming language.

Section 3.1.3 – Story

Immaculacy begins in the hometown of the story's protagonist and player character, Sydney Carlisle, pictured in Figure 1. Being from the small town of Danville, Sydney and her lifelong best friend Beth have always dreamed of moving to the big city, New Washington. Now that the two have graduated high school, the dream is becoming a reality. However, as other characters in the game will point out, rumors about New Washington's overbearing city government give Sydney cause for concern.

When Sydney receives a job offer that she cannot refuse, she finds herself in New Washington, bearing witness to the truth behind the rumors. In the name of maintaining the "immaculacy," the New Washington exercises many intrusive actions against citizens that provoke a certain level of suspicion. The surveillance used by the city's government to keep watch on its citizens leaves all of Sydney's information open for others to see. While the city government seems threatening, Sydney soon realizes that they are not the only ones with whom she needs to practice caution.

The storyline produced for *Immaculacy* provides the opportunity for many privacy-related issues to be addressed. The infringement on the information of citizens by the government leaves a great deal of personal data vulnerable in careless or malicious hands. In addition, other consequences of the government's behavior will prove to affect Sydney's privacy and safety, like citizens rebelling. At many points throughout the story, the player will be confronted with decisions that can affect Sydney and the world around her.

Some of the characters introduced in *Immaculacy* serve a major purpose in Sydney's life and thus, are able to provide the story with many options for choices that can be presented to the player. For example, the prologue introduces Sydney's boyfriend, William and her best friend, Beth. These two characters are placed at odds regarding certain decisions that Sydney has to make, which requires the player to make personal considerations when making decisions. Sydney's reputation with these non-player characters (NPCs) and their actions toward her are one form of consequence of the player's decision making.

One major part of the story is Sydney's move to New Washington following her job offer. Rather than have the entire story play out in Sydney's hometown, or having her already be a resident in New Washington, we decided to require that she relocate from this job. The primary motivation behind this decision was to have the player experience, along with Sydney, a transition to an unfamiliar and even scary new environment. Moreover, by making New Washington such a large and technologically advanced place when compared to Sydney's hometown, we were able to maximize the strangeness of Sydney's new home.

Section 3.1.4 – Game Mechanics and Features

In order for the choices made in the game to change elements of the story and the world as the player perceives it, a scoring system was created. In *Immaculacy*, there are two primary types of scores, security scores and personal scores.

Section 3.1.4.1 – Hidden Scores

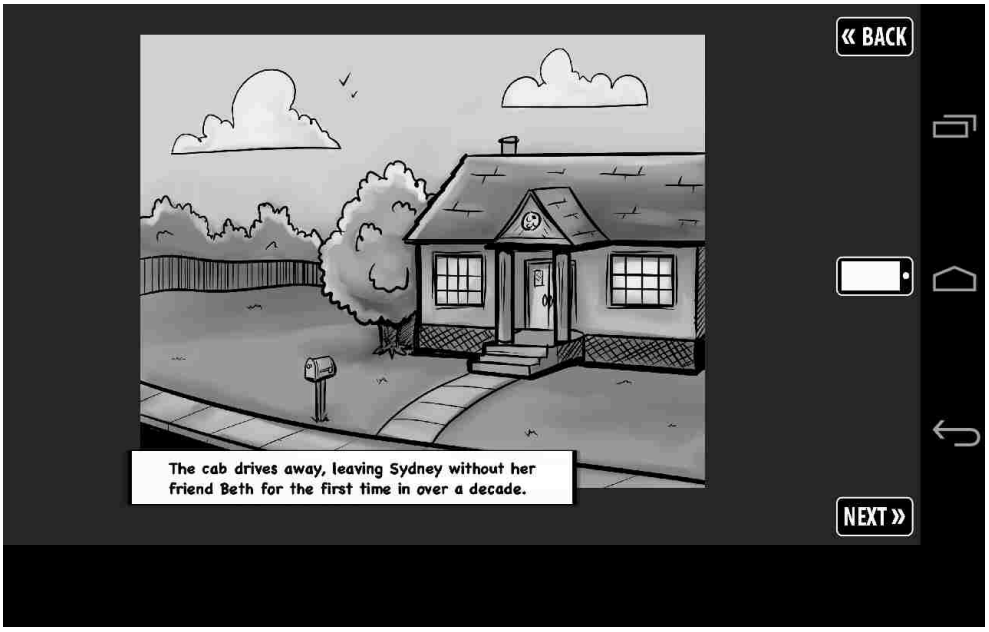
Security scores include: data leaking, determined by the amount of sensitive information the player reveals in dialogue, and city suspicion, determined by how much information the player withholds when speaking with authority figures. The security scores affect the extent to which Sydney is watched by the government and her susceptibility to privacy-related crimes. One example of a choice that effects Sydney's data leaking and suspicion scores is when she is being interviewed upon her arrival in New Washington. During the interview her encounter with the woman is addressed. If Sydney chooses to not help the woman, she will be asked why she did not help. If Sydney is honest about her intentions, her data leaking score will be increased. However, if Sydney is vague with the response, her city suspicion score will be increased.

Personal scores include: individual non-player character scores maintained for several main characters in the game and representing Sydney's reputation with them, as well as a character morality or karma score determined by the player's overall treatment of other characters. The scores affect the status of relationships with other characters, as well as her employment and community status. The scoring system for *Immaculacy* is further discussed in Section 3.3.2. An example of a reputation score with Sydney is encountered in the first scene of the game. In this scene, William and Beth are arguing about New Washington. Sydney is presented with several options in the argument. If she takes William's side, her reputation with him increases and her reputation with Beth decreases. If she sides with Beth, the reputation scoring is reversed. Regarding Sydney's character morality score, there is a question in the interview regarding who Sydney knew in the city, shown in Table 2 below. If Sydney chooses the third option, it is considered a lie and Sydney's character morality score is lowered as a result. This actual dialogue conversation is also pictured in Section 3.2.1 in Figure 5.

Character	**Line**	**Score Impact**
Interviewer	Do you know anyone in the city?	none
Sydney (First Option)	My friend Beth and the man who recruited me for the advertising agency, Mr. Kyle.	Data Leaking +1 City Suspicion -1
Sydney (Second Option)	My recruiter, Mr. Kyle...oh, and some girl who I went to school with for a few years.	City Suspicion +2 Character Morality -1
Sydney (Third Option)	Just Mr. Kyle, he recruited me for my position at the advertising agency.	Data Leaking -1 City Suspicion +5 Character Morality -2

Table 2 – An example of a dialogue option that the player has as Sydney, along with the score impact which remains hidden until the end of gameplay.

The mechanics for dialogue, narrations, and item interactions make up the majority of the game and are displayed similarly. Each scene is broken up into *panels*

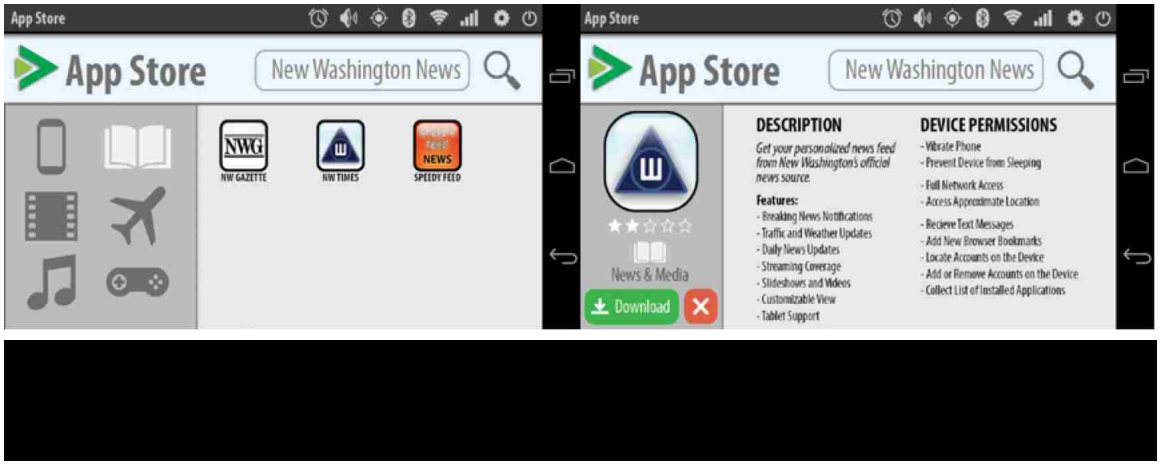


which contain either a line of dialogue with Sydney's response or choice, an item interaction with Sydney's response or choice, or a narration. The concept and design of the panels are further explained in Section 3.2.1. In the panel interface, the player is given buttons that allow them to move to the next panel or reset the current scene, as is shown in Figure 2.

For each panel containing a choice, the player must select a choice before being able to select the option to move to the next panel. In many cases there is enough room so that the full text of all choices can not be displayed at once. As a result, the player is given the ability to view the full text of each choice by tapping the box in which the choice is displayed. The player has the ability to view each choice as many times as desired and can move to the next panel once a choice is made.

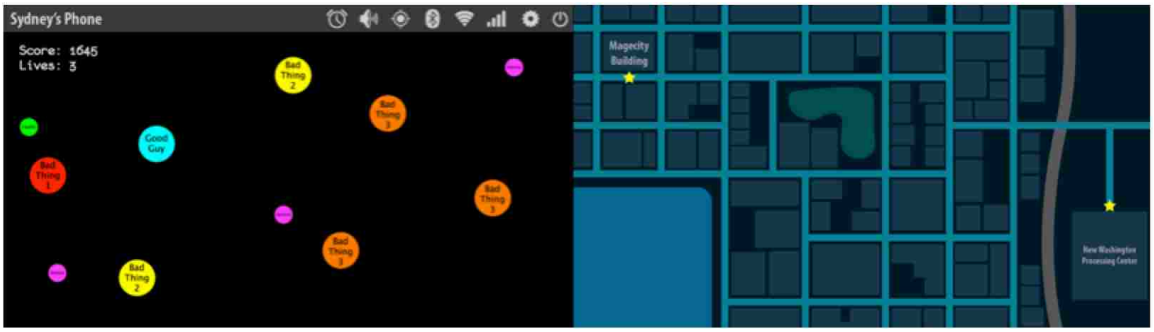
Section 3.1.4.2 – Smartphone

Throughout the game, the player also has the ability to interact with Sydney's personal smartphone. The functionality of the smartphone is similar to a typical



smartphone that one may encounter in reality. There is a home screen with icons for each app, including a simulated app store. The apps made available in the smartphone interface, either pre-installed at the beginning of gameplay or acquired through the app store, are created for *Immaculacy*. While they are not real applications found on current smartphone markets, they are modeled after real apps that the player may already be familiar with. Furthermore, some apps are only available at defined points in the storyline or as a result of certain choices made by the player. Screenshots of an application used within the smartphone interface can be found in Figure 3.

The player can view the smartphone at any time throughout gameplay through a third button on the narrative interface. Though, there are points within the story in which the character will be scolded and even punished for cell phone use. On the other hand, there will be instances in which an interaction with the smartphone is part of the narration and the smartphone interface will automatically be launched. In this case, the player will have instructions of actions to perform on the phone. After the player performs the instructed actions, the game will return to the narrative interface automatically. At this point, the player can continue the scene. When not part of the scene, the player must manually access and exit the smartphone screen via provided icons.

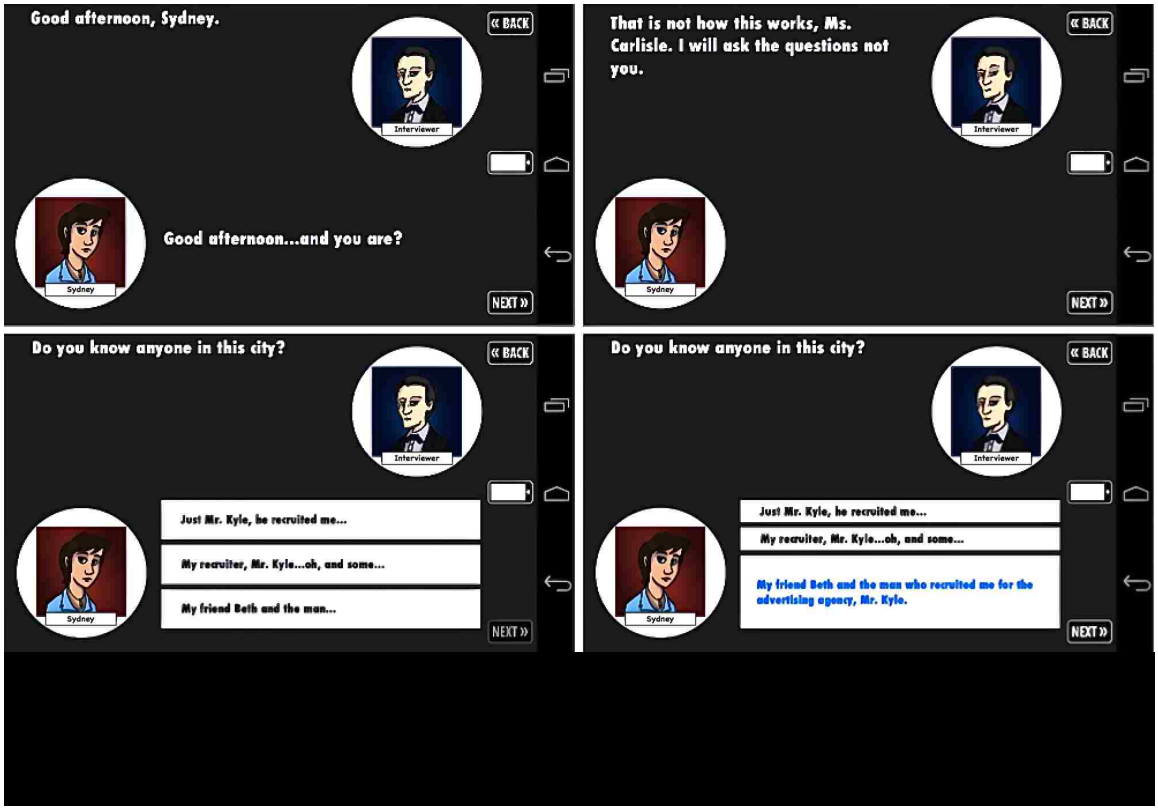


Section 3.1.4.3 – Mini-Games

The final and newest type of features present in parts of *Immaculacy* are mini-games that support the story in some way. The intent behind including mini-games is to increase the entertainment value of the game. Each mini-game has its own controls and interface type, which are introduced when the mini-game is launched. Each mini-game is also given an end condition to avoid the player being stuck in the game too long. One example mini-game is a game that Sydney plays on her smartphone while she is waiting in a long line. Other mini-games in development include a navigational game for finding a building in the city, as well as a memory test given at a point in the story where Sydney is at a doctor's office. A screenshot of one mini-game and art used for another mini-game can be found in Figure 4.

Section 3.2 – Game Interface Design

One major part of the design process for *Immaculacy* was designing and evolving the user interface. The interface is broken up into three major components: narrative interface, smartphone interface, and mini-game interface. These components and design decisions made when creating them are described in detail in Sections 3.2.1, 3.2.2, and 3.2.3, respectively.



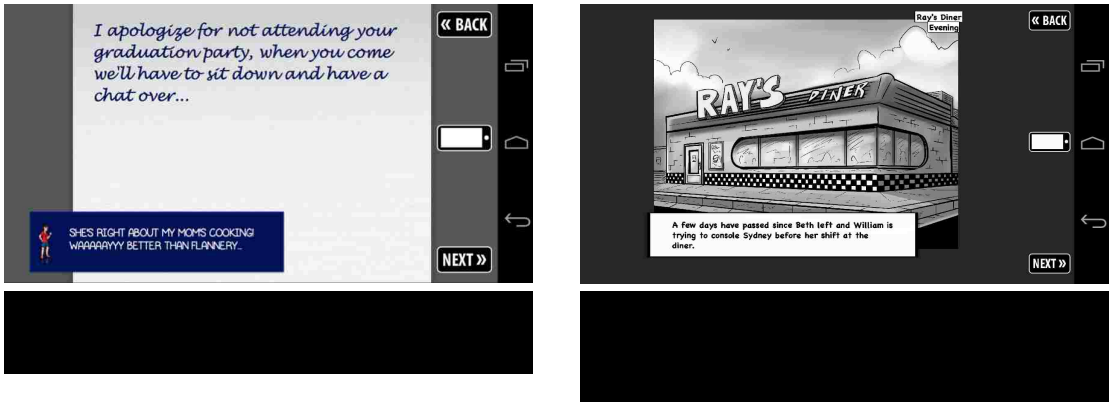
Section 3.2.1 – Narrative Interface Design

The primary interface in the game is the narrative interface which includes dialogue between characters, screens representing item interactions, and narrative descriptions. Each individual screen in the narrative interface is considered a single panel and the display of each panel is often determined by the scores maintained within the game. For further distinction, conditional statements are attached to certain lines of dialogue that should only be displayed if they meet specific requirements. For example, in the first scene in the game, the player is given a choice between a joking response and sentimental responses in one panel. For the non-player character to which Sydney is talking, the dialogue displayed in the next panel is determined by whether or not the player chose the joking response. The conditional statement can be as simple as a true or false flag, or they can be a more complex statement with thresholds on the value of a

specific score or multiple scores. Scoring and flags are discussed in detail in Section 3.3.2.

The dialogue part of the narrative interface was given minimal features to keep the user engaged in the dialogue. The screen is slightly different when a choice is available where all decisions are shown in separate boxes and each box can expand to display the full text for the choice. An image showing the difference between the two screens can be found in Figure 5 on the previous page. A comic-style image of characters is displayed to show characters currently participating in the conversation. Since Sydney is part of every conversation and for the sake of consistency, her image and response is always located in the bottom half of the screen while other characters in the conversation and their lines are located in the top of the screen. Thus, each panel consists of a non-player character's line and Sydney's response. If Sydney does not have a response to the line, for example, when there are three people in the conversation and the two NPCs are addressing each other, Sydney's image will still appear at the bottom of the screen, with no response.

The item interaction displays facilitate parts of game scenes in which the player interacts with various items. To offer some variety in the interface and further explanation of the type of interaction being performed, images have been added to the item interaction screens. These images act as backgrounds to represent the interface or item with which Sydney is interacting. For example, if Sydney is reading a handwritten letter, the background and font changes from the dialogue interface to make the screen look more like a handwritten letter. An example of the handwritten letter screen is shown



above, in Figure 6. Similar view changes occur for computer or email interactions and journal interactions.

Like the dialogue views, panels in which a narration is setting up a scene or describing a part of a scene are also designed to have a comic-book appearance, as depicted above in Figure 7. For the narrative panels setting a scene, the location and time of the scene is displayed in captions at the top of the screen. All narrative panels have the narrative description in a caption at the bottom of the screen, with an illustration representing the scene being described.

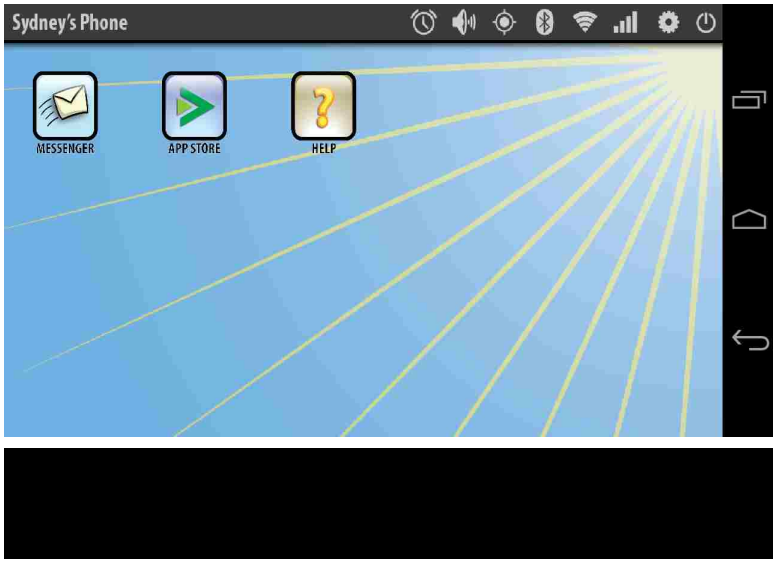
Because the main part of *Immaculacy* is the story, it is important that the interface displaying the story be immersive. While trying to maintain some diversity in the types of displays shown between the different types of panels, a simplicity was still pursued to avoid distracting from the story. This was achieved, in part, by moving some of the typical game menu functionality to the smartphone interface which will be explored further in Section 3.2.2. In addition, small animations like words scrolling in on dialogue screens and narrative captions sliding onto the screen are added to further encourage player engagement. For some of the characters, several images of the character were created to display different emotions. This helped in a small way to contribute to the realism of the game by portraying the characters in a way that supports the tone of a

scene or line of dialogue. Some of these design choices were informally tested in playtesting or demonstration sessions. Feedback and influences regarding these choices are discussed in Chapter 4, Section 4.2.

Section 3.2.2 – Smartphone Interface Design

As was mentioned in Section 3.1.4, the smartphone interface was designed to look like a real smartphone. This functionality was added both to allow extra functionality in order to maintain the simplicity in the narrative interface, as well as to present another medium through which the player can interact with the game world. The smartphone also represents another potential vulnerability for Sydney's private information.

The home screen of the smartphone interface is always available through the narrative interface. This screen, shown in Figure 8, is where all other apps, including the app store are accessible through their respective icons. The app store is added to the smartphone primarily for story events through which Sydney is prompted to download an app either for personal use, or by another character. All apps installed on the smartphone via the app store will be available in the smartphone from when they are installed through the remainder of the game. Moreover, the app store is available throughout the entirety of the game but the apps available will be dependent on the story timeline.



Section 3.2.3 – Mini-Game Interface Design

When compared to the different narrative and smartphone interface screens, the mini-games have a much freer form regarding user interface design. Currently, there are three mini-games designed for *Immaculacy*. The first game is a top-down enemy avoidance game, which is available in the smartphone part of the game. The second is a tiled memory game in which the player must remember the order of tiles lighting up. Finally, the third game is a navigational, maze-like game that takes Sydney from one point in the New Washington to another point.

Very different approaches were taken in designing the interfaces for these three games. This was mostly due to the major differences between the subject matter and types of each of these games. This especially includes the information that needs to be presented or displayed for each mini-game. The top-down game requires a display of the score and number of lives left, as well as the actual game screen on which the characters are being animated. On the other hand, the memory game need only display tiles that display images in a certain order to be remembered. Moreover, the top-down game is intended to be hosted on the smartphone, and thus, the interface must include the smartphone header bar. However, the memory game is intended to be hosted on a terminal in a doctor's office, which must be shown as part of the interface. Thus, it can be expected that there will be less uniformity in the mini-game interfaces, when compared with the other types of interfaces.

Section 3.3 – Representing Scenes

The dialogue and scoring system is the component of the game that drives each scene and maintains the game state through each transition. As will be described further

in Section 3.4, the actual use and communication of game data happens through the code itself. However, an internal scene representation scheme was developed to manage the dialogue and assets used within the game.

Section 3.3.1 – Scene XML Representation

The first step in the implementation of the dialogue driver was finding a way to represent the script within the codebase. The decision was made to convert the script into XML and to construct a decision system within the schema. The next few paragraphs will describe the structure for the XML system. Though, an actual schema of the XML structure can be found in Appendix A.

When creating the scene structure in XML, the first real challenge was deciding how to model the dialogue path. First, the decision was made to have each scene be represented by its own XML file. Each XML file begins with a declaration of flags and scores that will be used or updated within the scene and whether the flags or scores exist globally or if they are only used for the single scene.

After scores and flags, characters that appear in the scene are declared. Each character appearing in the scene is given a unique key that is used for the character in that particular scene. In addition, the default asset key for the character must be declared. This allows the code for the game to retrieve the asset for the specified character when it must be displayed. This functionality is facilitated by having a master file with character image keys being linked to actual image.

Once the header information is defined within the scene file, the dialogue itself can be represented. The approach taken was to use a graph structure to represent the dialogue, where each node represents a single line of dialogue. Examples of the graph

representations for the dialogue in each scene can be found in Appendix B. In addition to Sydney sometimes having choices for dialogue lines, *Immaculacy* has lines for both Sydney and NPCs that are only displayed if they meet certain conditions for a specific score or flag. For example, in an interview scene with Sydney and a city official, the questions asked are based upon choices that Sydney has made. At a point in the game before this interview, Sydney is given the choice to help an NPC. In the interview, one of the questions is in relation to this situation. However, the specific question that is asked, along with Sydney's response options, are dependent on whether or not Sydney helped the NPC. This presented a need for the dialogue driver to only consider dialogue lines that meet a certain condition.

Section 3.3.2 – Scoring System

Implementing checks for score conditions and flags for the dialogue system also presented a simple way to represent the graph structure. For the most part, each scene's dialogue has defined start and end points, with other branch choices in the middle. Thus, the structure is like a tree only in that each dialogue file can be seen as having levels. However, unlike a tree, nodes can point to parent nodes and there is no guarantee that there will be only a single node at the root level. Each level contains one or more nodes, where levels with one node represent a single dialogue line, a narration, or an item description. In the case that the line is a single comment by a character, that line is the only option at that level, under any condition. A level with more than one node does have options at that level, conditional alternatives, or sometimes both if it is a line spoken by Sydney. Thus, each dialogue node has a level attribute which is implemented by creating a level score in each scene that is updated each time a node is visited in a graph. The

value for which to update the level is an attribute within all nodes pointing to that level. If the dialogue reaches that level, any node residing at that level is selected only if it meets any conditions specified for it.

It was previously mentioned that there are two types of flags and scores: local and global. Local data is specific to one scene and is destroyed at the end of the scene. Global data refers to flags and scores that are used across the entire game. Local flags and scores are used when a choice is selected at one point in a scene, then later in the scene, the flag or score is used as a condition for displaying particular dialogue options. Global flags and scores work the same way as local, except they can be accessed across multiple scenes. The character morality, city suspicion, data leaking, and character reputation scores are global scores. The level score, while appearing in each scene, is considered a local score as it is reset between each scene. Regardless of whether a flag or score is global or local, it is stored in the game state, covered more in the next section, and is used for determining what dialogue lines or events to select throughout each scene. In addition, scores and flags can be used as conditions for updating other scores and flags as well.

Updates are a data type assigned at the node level and are executed if a node is selected as part of the dialogue. A node is selected as part of the dialogue if it is displayed on the screen, and for player options, it must be the option selected when the next button is pressed. Each node has at least one update, which is the update for the level score. Recall that a node can point to any other node in the graph, but it must be specified what the node points to. For nodes that take place at the end of the scene, the level score is updated to -1 to signal for the dialogue driver to move on to the next scene. Other updates include score and flag changes. Like the nodes themselves, updates can also have

conditional values that constrain cases in which the specified flag or score should be updated. For example, one case in which this happens is when there are more than one possible levels that a node can point to, and the level that should be assigned at runtime is dependent on the value of a certain flag or score at that point in the game.

Section 3.3.3 – Handling Assets

The flags, scores, and updates discussed above dictate the flow of the game and the actual dialogue that takes place. For the dialogue, at the node level, several attributes are assigned that are dependent on the type of node. All node types can specify an image key but what type of image is displayed depends on the type of node. For narrative nodes, the image tag specifies the image setting the scene that the narrative line is describing. In addition, the narrative nodes can have a time and location specified that are displayed on the screen, as described in the Section 3.2.1. For item nodes, the image attribute is the only additional attribute to updates and dialogue line that can be assigned. The image tag for an item node defines the key for the background image to be used, like the journal book background for Sydney's journal entries. Similar to the item nodes, dialogue nodes also only have an image attribute. In this case, the attribute describes the emotion that the character is showing. Each character has a default emotion which is displayed anytime the image tag is not used. In this case the key is simply the character's name. For example, Sydney's default emotion is a neutral pose. The key for that image would simply be "sydney." However, if Sydney should be depicted as happy for a certain dialogue line, the image tag would be present with the emotion being conveyed, in this case "happy." Then, in the lookup file, the driver would search for "sydney_happy" to retrieve the image of Sydney with a happy expression.

All data that is represented within the XML files is accessed by the code for the game to be parsed and distributed to the appropriate parts of the game structure. Since data defined in the XML scene files dictates how events unfold based on game state and dictates how and what data should be visible on the screen at a given time, it was important to design the a game architecture to support the data in the XML files.

Section 3.4 – Game Architecture

All of the assets, dialogue, scores, and features are tied together by the code used to run the game. Several classes from *LibGDX* are extended and utilized to facilitate the functionality of the game. The design documentation including UML and diagrams describing the flow of data throughout the game can be found in Appendix C. A basic model-view-controller approach was taken in creating the structure for the game. The model portion of the structure includes the classes that manage game scores, the game assets, and other data used. The view portion of the game involved the implementation of the narrative, smartphone, and mini-game interfaces which were discussed earlier in this section. Finally, the controller is made up of the driver classes that handle user input and use the input to update scores, update dialogue, update the view.

Section 3.4.1 – Model Component

The model component of the game structure involves the classes that manage the data and assets in order to maintain a consistent game state across all components. There are three main classes that provide this functionality, the *GameStateManager*, *UIStateManager*, and *AssetsManager*. The *GameStateManager* class keeps track of all scores and flags in the game, both global and local. Anytime a score or flag is created by the dialogue driver or updated through an update attached to a dialogue node, the data is

passed to and stored in the *GameStateManager* class. In addition, the *GameStateManager* class keeps track of character data as well, including which characters are present in the current scene and their display names, which are both hashed using the character's scene key. The *GameStateManager* also has the capability to evaluate the boolean statements used for dialogue lines or updates that have conditions dictating whether or not they are to be used. This information is used by the drivers to decide when an update should be executed and which dialogue lines to display.

The *UIStateManager* is responsible for keeping track of all data used by rendering classes. This includes constant and variable values. The constant values represented in the *UIStateManager* set screen geometry variables that help to gauge where on-screen objects should be placed and how long animations should last, based on the screen resolution of the device being used. Additionally, there are constant instances of the Stage, *ShapeRenderer* and *SpriteBatch* provided by *LibGDX* for rendering. Variables in the game state involve values that keep track of how long animations have been running to test when they should terminate.

The *AssetsManager* keeps track of the image and dialogue file locations used for each scene. Within the assets manager, the image and dialogue master files are parsed which contain the file locations for their respective files. The dialogue file locations are accessed in order, based on which scene was just finished. The image file locations are indexed by keys that are used within the dialogue scene files to reference said images. For saving and loading games, the data values within the *GameStateManager*, *UIStateManager*, and *AssetsManager* are written to a file which can be used to load a saved game state at a later time.

In addition to the managers, classes creating the structure of each scene, *DialoguePath* and *DialogueNode* also belong to the model component. These classes are largely used by the dialogue driver class to determine which panels should be displayed at certain points in the game, given game state. The utilization of these classes by the driver is further discussed in Section 3.4.3.

Section 3.4.2 – View Component

The user interface creation began with classes that implement the *Screen* and *GestureListener* interfaces, provided by LibGDX. For *Immaculacy*, there are three types of screens implemented. The first two, *MainScreen* and *ScoreScreen*, are simple, static screens with little input and no animations. The *MainScreen* is the opening screen of the game and gives the player the option of starting a new game, continuing the game, or quitting. The *ScoreScreen* is available after the game has been played through and displays the values that the player achieved for each of the scores that remain hidden throughout the game. The third and most complex screen implementation is the *GameScreen* which entails the entirety of the game itself. In other words, the game screen handles all interfaces described in Section 3.2: the narrative, smartphone, and mini-game interfaces.

For an entire playthrough, there is only one instance of the *GameScreen* class. The *GameScreen* class maintains instances of classes extended from the *Display* abstract class which was created to facilitate rendering for the narrative, smartphone, and mini-game interfaces. This design choice was made so that each of the different interface types can be rendered quickly with the ability to have only one instance of the dialogue driver classes to be accessed through the screen instead of being passed around to different new

instances. In addition, the interface types have some major similarities at different levels. All interface displays have the implementation for most of the methods inherited by and implemented in the *GameScreen* class. The *Display* abstract class also provides utility methods that are used by all display types as well, like centering text in a rectangle or checking if a rectangular area is pressed. This leaves room in the individual display classes to cover their differences in their own rendering and user input methods.

For each part of the game: dialogue and narration, smartphone interaction, and mini-games, a different display abstract class is created which extended by fully implemented subclasses. At the level of these abstract classes, there is also functionality that the subclasses share. For example, there is an abstract class *PanelDisplay* for showing the context of the scenes, which is extended by the classes: *DialogueDisplay*, *ItemDisplay*, and *NarrationDisplay*. Each of these displays include some of the same graphics, like the buttons on the side of the screen to navigate through the scene, in addition to other similar functionality. Then, these classes each have their own render and input methods, as well as other methods for view setup and rendering that are specific to the individual display.

The *GameScreen* then maintains a single instance of *Display* and the instance created each time the input from the player causes a display change, this instance is assigned to a new instance of the appropriate display type. The approach of using a single *Display* instance within the *GameScreen* allows for more organization within the codebase and gives the *GameScreen* class the ability to concentrate on communicating with the *DialogueDriver* and setting up the *Display* for each screen change. Then, the display subclasses themselves focus on animations and rendering the correct objects in

the correct locations. Doing so is feasible because the *GameScreen* class only utilizes methods that are implemented or at least declared in the *Display* superclass, which are the user input and rendering methods. These methods are used within the *GameScreen*'s own user input and rendering methods. Other methods that are defined within the *DialogueDisplay*, *ItemDisplay*, and *NarrationDisplay* abstract classes and their subclasses are only used within said subclasses.

Section 3.4.3 – Controller Component

All of the data passing, including XML parsing, is handled by the controller component. The central part of the controller component is the *DialogueDriver*, which acts as a bridge between the input from the user via the view component, and the data within the game, maintained by the model component. Within the *DialogueDriver* itself, the functionality provided involves updating the user interface when player input is received. The three main update functions involve updating dialogue when the next button is pressed, resetting dialogue when the back button is pressed, and saving the game state when quit game is pressed. When dialogue is updated, the updates for the current dialogue nodes are run, then the next dialogue panel is created and the screen updated. The panel creation either involves using the next nodes in the dialogue path or nodes created when reading in a new scene file if the end of a path is reached. To support this functionality *DialogueDriver* utilizes the *DialogueXMLHandler* for parsing the XML scene files and a *DialoguePath* instance to traverse the structure of the dialogue tree currently in memory. An instance of *GameScreen* is used within the *DialogueDriver* to update the screen after player input is received by calling the correct display setup method within the *GameScreen*.

The *DialogueXMLHandler* is responsible for actually creating the dialogue tree structure for each XML scene file that is parsed. Flags, scores, and asset information defined within the scene file is added to the *GameStateManager* when encountered during parsing. Anytime a *response* tag is reached during parsing, a new *DialogueNode* is created. Since each XML is parsed from top to bottom, all other tags: *update, image, location, time,* and *line* can be added to the existing dialogue node within the XML handler. Then, once the *response* end tag is reached, the existing node can be added to the *DialoguePath* via the *DialogueDriver* class. The *DialogueDriver* then adds the node to the dialogue graph structure maintained by the *DialoguePath* class.

Within the *DialoguePath* class, a structure of *DialogueNodes* are maintained and organized based on the level for which the node is assigned. As explained in Section 3.3.2, there can be more than one node at any level and some nodes can only be selected under certain conditions. These checks are completed within the *DialoguePath* class when a new *Panel* is requested by the *DialogueDriver*. When a new *Panel* is requested, the *DialoguePath* grabs all eligible nodes at the new level and adds them to a new instance of *Panel*. If the node at the new level is a narration node or the new level involves nodes in which only Sydney is speaking, the new *Panel* can be returned to the dialogue driver. For single nodes at the new level of type NPC dialogue or item interaction descriptions, the *Panel* must have a response node for Sydney added as well. This is achieved by first checking the level at which the node at the current level is pointing. If the node at that level is a node with a response for Sydney, then that is the node added to the panel for Sydney's response. Otherwise, a node is created to be added to the *Panel* with a silent response for Sydney.

Section 3.5 – Discussion

The design described in this chapter was reached through several iterations of development. As the game began to take form and grow, new approaches and design patterns emerged out of necessity, in order to keep the codebase at a manageable size and avoiding duplicate work. In addition, we conducted informal tests and presentations on demonstrations of the game which influenced the design of the user interface and some of the game mechanics. Evaluations that were performed on the game will be discussed in Chapter 4, along with future plans for conducting more formal user tests.

Chapter 4

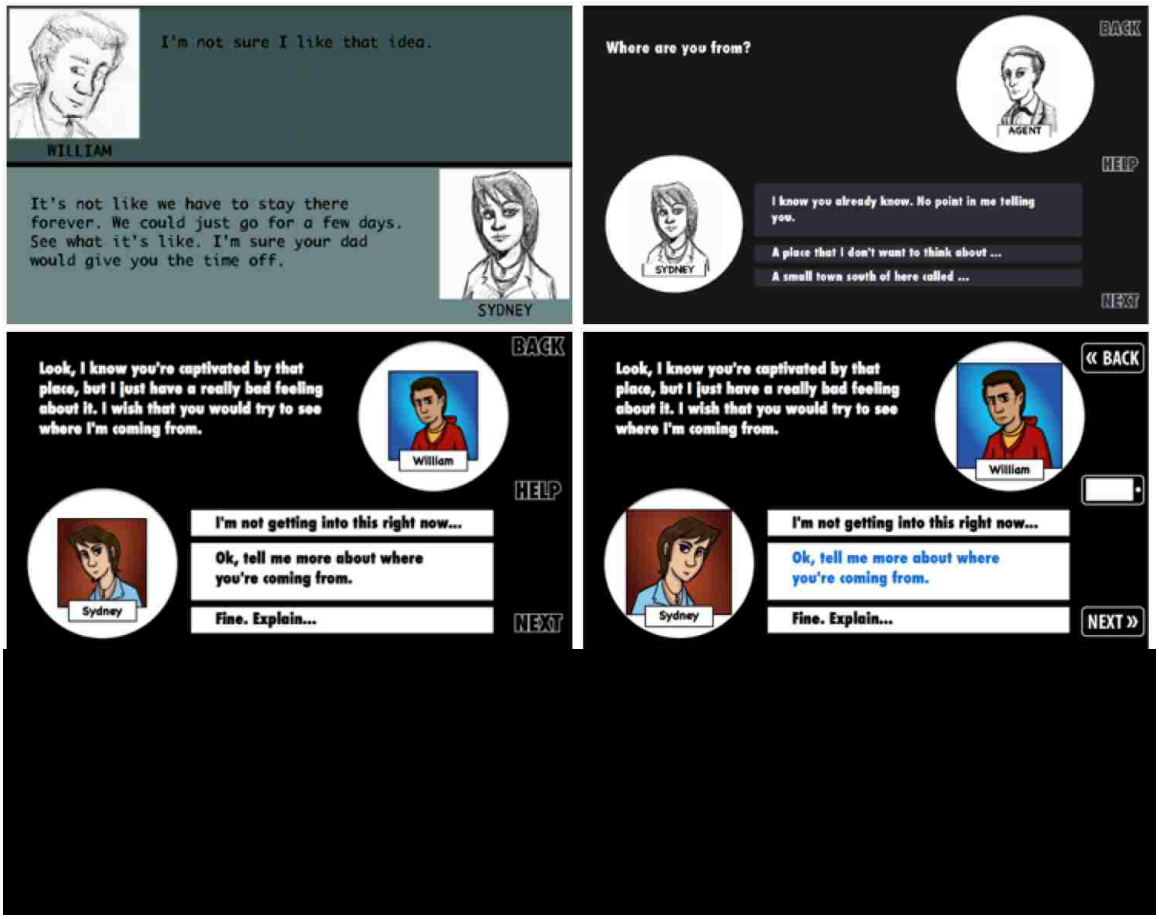
Game Evaluation

As was briefly described in previous chapters, an iterative approach was taken in the game design process for *Immaculacy*. The first few iterations involved prototyping certain ideas and testing them ourselves. This process is discussed in further detail in Section 4.1. Section 4.2 moves on to informal playtesting and feedback received from game demonstrations. Lastly, Section 4.3 will cover current and future work for formal user testing which is intended to gauge the effectiveness and entertainment value of the game.

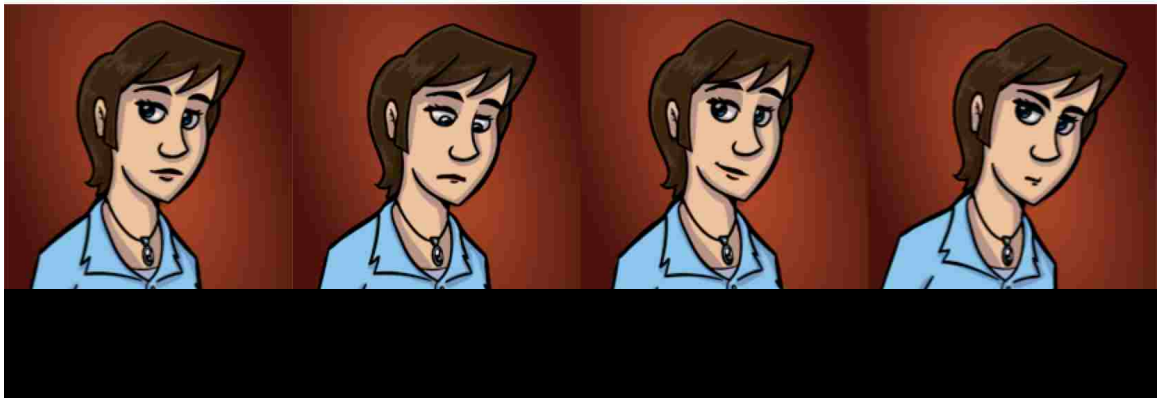
Section 4.1 – Initial Implementations and Internal Testing

The first few iterations of implementation involved prototyping different ideas and testing them within our research group. This approach helped in choosing the genre and mechanics for the game. *Immaculacy* began as a point-and-click adventure with dialogue and choices. However, when testing a version of the game with dialogue only, it was found that it was easier to focus on the story. As a result, the decision was made to have the game's main functionality be a narrative and interactive story.

The choice was made to have a smartphone interface implemented within the game for messages and menu options. This decision was made prior to the decision to use the narrative interface. So, we decided to give the smartphone a larger role to add to the story. This was achieved by adding more functionality and tying the smartphone use in with the dialogue at times. Then, to add more entertainment value, the decision was made to begin adding in mini-games. The mini-games are also tied in to the dialogue in many cases, though sometimes they can be accessed and replayed later.



Once the decision was made to use a narrative interface, the testing process began to focus more on interface design and how the game mechanics would work. One particular challenge was deciding how to display dialogue or narrative choices within the game. We first decided that we want to display as much information as possible to the user at one time. However, we knew that there often was not enough room on the screen to display all of the text at once. We also hoped to maintain a consistent layout between the panels with choices and those without. Thus, the decision was made to still limit the space taken up by the choices to the space allotted for Sydney's responses without choices. To show that choices exist, boxes are displayed with part of the first line of text in them. When a box is touched or clicked, it will expand to reveal the full text. We also found that it was important to ensure that the game does not allow the player to move on



to the next panel until a choice is selected. This was handled by disabling the next button until the player makes a choice.

When testing the interface itself, sketches of characters and scenes were used in each panel. Through this process, we were able to identify that a comic style was likely ideal for the art within the interface. The evolution of the interface can be seen above in Figure 9, on the previous page. It was also noted during this testing that it would be appropriate to create several images for some characters to convey different emotions so that characters did not have an inappropriate expression on their face for lines that they were delivering. For example, if two characters are having a conversation with a sad tone, images in which the characters are smiling would not match such a tone. An example of how the emotions are conveyed is shown in Figure 10. Lastly, for the narrative interface, initial testing helped to identify points in the interface where animations would be appropriate. Doing so helped the narrative interface feel more immersive. Some of the animations added include: scrolling text in dialogue and item interactions, having the narrations slide onto the screen on narration panels, and having choice boxes gradually expand and contract when narrative choices are presented.

The most recent internal testing was conducted on the smartphone interface to determine how it should look and work. The first challenges faced were deciding what

functionality to add to this part of the interface and how it would be accessed. It was decided that the smartphone could be accessed at anytime from the narrative screen via a button on the side panel between the back and next buttons. In an earlier version of the game, this was the location of the help button. Thus, it was decided to move the help button and its respective screen to the smartphone interface. This help screen, the game menu, and the other applications on the smartphone can be accessed through icons on the phone's home screen.

While we were able to identify many interface and mechanical issues and design ideas, we could learn much more by demoing the game to people outside of our research group and allowing people to play the game. The next section will describe in detail what we were able to learn from these demos and the feedback that we were able to use.

Section 4.2 – Informal Playtesting

Throughout the course of development of *Immaculacy*, we have had several opportunities to demo the game in a presentation and to conduct informal playtesting. *Immaculacy* has been presented in several seminars at which students and faculty who were also working on game projects could view and play the game to give feedback. *Immaculacy* was also accepted to be featured in the CHI Play 2014 student game design competition. At this venue, dozens of conference attendees who work in game development and research played the game and were able to give some very valuable feedback.

For each venue at which presentations and informal playtesting was conducted, the process was fairly similar. During presentations, the game subject matter and gameplay are discussed before a demo of the game is played in front of the audience.

This approach helped us receive a great deal of feedback regarding interface design. However, it is hard for people to critique the game mechanics without playing the game themselves. Thus, during these seminars and at CHI Play's student game design competition, people are also given a chance to play through as much of the prologue and first act as desired. While they were doing so, we were able to gain both verbal feedback from the players, as well as having a visual perspective of potential design flaws noticed by parts of the interface either overlooked or misunderstood by players.

During the presentations and informal playtesting, feedback was obtained regarding interface design, complexity of the game, length of the game, and overall entertainment value. Given this feedback, we were able to make some very important decisions in the design of the game. Some suggestions helped with interface design, while others helped us to address mechanical issues that we may not have noticed.

- **Having interactions with items appear with different backgrounds.** For example, having a letter or journal in the background, rather than text on a solid background.
- **Sydney's portrait facing away from other characters during dialogue.** Some people pointed out that the flow of the conversation would be more obvious if Sydney's portrait was flipped vertically to have her facing the portrait of the character with whom she was speaking.
- **Touch target size and placement.** Some touch targets for icons were too small and users had to press the icon several times before having a response. This was solved by making touch targets larger.

- **Back button in smartphone interface causing players to quit the game altogether instead of returning to narrative interface.** Many users inadvertently exited out of the game itself by pushing the physical device's back button to return to the narrative interface. To mitigate this issue, we chose to override the back button to simply return to the narrative screen.

By considering the feedback, performing another iteration of development, and conducting follow up playtests, we did receive comments that the parts of the game addressed contributed to how immersive the game was. We plan to take a similar approach with formal user testing. However, in addition to learning more about how to make the game more immersive, we hope to gauge *Immaculacy*'s effectiveness at encouraging players to consider privacy more often. The next section will further discuss how we plan to conduct the formal user testing and exactly what we hope to learn.

Section 4.3 – Formal User Testing

For *Immaculacy*, the intent is to assist average internet users in gaining knowledge in privacy-related topics and how to better protect themselves. From previous work, we are aware that it is important to avoid extremes regarding length and difficulty of the game. In addition, we must ensure that *Immaculacy* fulfills its purpose of raising awareness on many prevalent privacy issues. Through formal user testing, the hope is that *Immaculacy* can be optimized in educational value as well as entertainment value.

While the participant plays through the game, we intend to observe the participant's interactions with the interface in general. From this, we hope to further identify any existing interface issues that may impede gameplay or distract from the game content. In addition, we will be watching for what decisions are made throughout

the game. For key privacy-related choices, we may verbally ask the participant what his or her motivation was behind the decision made. With these questions, we hope to gain several different types of insights. First, we want to ensure that instructions given in the game are understood by the player and that they are not simply trying options to see what works. In addition, we hope to learn what average internet users think about when confronted with privacy-related decisions, like app downloads, to try to evaluate whether or not we are taking an effective approach with the presentation of such decisions within the game. Finally, we hope to assess if decisions made early in the game demo have an effect on how a player thinks about choices that they make later in the game demo.

Finally, when the user completes the demo, we will first take note of the scores that he or she received for each of the categories defined in Chapter 3, Section 3.1.3. We will then follow up with several questions about the game and the scores. First, we will try to get an idea of how the participant thinks he or she did regarding the different scores and if they were surprised by any of their scores. Next, we will discuss the actual gameplay experience with questions about how disturbing or intrusive some of the scenes or characters were. With this, we hope to gain an idea of the entertainment value of the game, as well as the overall impact of the game. In addition, the participant will be asked about their feelings regarding the amount of decisions they were given in the game and the consequences of certain decisions. We would also like to learn more about the smartphone and mini-game interfaces and whether they contribute to the experience or if they distract from the point of the game. Finally, we hope to learn about the effectiveness of *Immaculacy* at raising privacy awareness. To do so, we will ask the participant if any new concerns regarding online privacy have been raised for them, or if the game has

changed the way he or she views their online conduct. Overall, our intent as developers is to give our audience a game that entertains them but also provokes thought on internet conduct. By conducting the user study described here, we hope to gain enough feedback to reach that goal.

Chapter 5

Conclusions and Discussion

Immaculacy is an interactive story that utilizes a comic-style interface to immerse its players into a future where a person's privacy is much more vulnerable than even in modern times. The intent behind *Immaculacy* is to raise awareness and encourage players to consider privacy implications when they meet privacy-related decisions in real life. To do so, *Immaculacy* covers a broad range of topics that are often of concern to people who use smartphones, social networks, and other internet accounts like email, banking, and healthcare.

Like many of the games described in Chapter 2, *Immaculacy* has been created for educational purposes in the area of online privacy and safety. While *Immaculacy* was influenced to varying degrees by these games, the approach, interface, and content within *Immaculacy* are unique both in style and delivery. Though, these studies did contribute in helping to identify development strategies and pitfalls in story and gameplay, in addition to showing different approaches for user testing and iterative design.

Early in the design process, the decision was made to style *Immaculacy* with an interface that is primarily narrative, with mini-games, item interactions, and a smartphone interface to break up some of the redundancies experienced in a fully narrative game. As was mentioned in reviews for *Immaculacy: A Game of Privacy* (Suknot et al.), the simplicity of the game allows the player to be fully immersed in the game and to give focus to the storyline and choices presented in the game. Implementing such an interface, while challenging, was still easy to organize into a model-view-controller design pattern. From there, the main challenge involved representing the script for each scene and

corresponding assets in a way that could be easily accessed and used by the game. This was achieved by representing each scene in XML and having the assets be easily accessed through locations defined in a master file and indexed with unique keys. The final challenges existed in designing and implementing an intuitive smartphone interface and mini-games that add to the story.

Even though *Immaculacy* is purposed for education, we have spent much of the development process prioritizing the player experience. The primary reason behind this is that we feel that if a game is not entertaining, it will not be able to maintain a player's attention. Though, a player must pay attention to the game in order to have a chance to learn from it. As a result of prioritizing the entertainment value, we have been able to make strides in the creation of a game that can be fun and enlightening to a broad audience, consisting of: average internet or smartphone users, digital comic fans, and role-playing game fans, among other groups. This has been achieved through an iterative design and testing process that is currently moving into final phases.

The design and development process for *Immaculacy* also yielded a framework for developing narrative software. As is shown in Chapter 3 and in Appendix C, the scene representation system was abstracted in a way that could be used with any story and assets by creating XML files that fit into the schema described in Appendix A. One part of future work that can be applied to this project would be to create a graphical interface to help people with little technical experience create their own games or stories by automatically generating XML files based on the user input.

The majority of current and future work is preparing for and conducting formal user tests that are intended to further assess the entertainment value of the game. In

addition, this iteration of testing will help us to assess the impact of *Immaculacy* and its effectiveness at motivating its players to evaluate their own privacy habits. Through interviews that we will be conducting before and after gameplay, we intend to get a more thorough understanding of privacy concerns that average internet users have, in addition to common mistakes made regarding privacy. Our hope is that this iteration of testing can assist the design and development of the remainder of the game by providing ideas and guidelines to consider during story writing, addition of smartphone features, and mini-game creation.

Our long-term goal is to complete the story and primary development for *Immaculacy*, in order to achieve an eventual release on the platforms for which the game has been developed. While educators and digital comic fans have already shown interest in *Immaculacy* we hope to gain an even broader audience by integrating the feedback received during each iteration of testing. While *Immaculacy* may not be considered a game that can formally teach a person how to handle any given situation on the internet, the game does provoke thought that can help to form a player's instincts by presenting comparable scenarios. One feature of *Immaculacy* that is unique to other privacy related games is that the player is presented with choices that, instead of always being addressed immediately, have the chance to sometimes sneak up on the player later in the game. This approach allows the game to present the player with a realistic chain of actions and consequences that can be related to real-life scenarios. By taking this approach, we hope to eventually release a game that has the ability to provoke thought on internet privacy within a fun and safe environment.

APPENDIX A

Scene XML Schema

Element Descriptions

```
<scene>
  <flag key="flag key" type="local/global">            </flag>
  <score key="score key" type="local/global">            </score>

   <character key="pc/npc(#)">
     <asset>                    </asset>
     <disp>           </disp>
  </character>

  <dialogue>
    <response key="character/item/narr key" level="#" cond="flag/score key" val="score/T/F">
      <update key="flag/score key" type="flag/score">           </update>
      <update key="level" type="score" cond="flag" val="true">          </update>
      <update key="level" type="score" cond="flag" val="false">          </update>
      <image>                              </image>
      <location>                       </location>
      <time>          </time>
      <line>                                    </line>
    </response>
  </dialogue>
</scene>
```

- Scene Tag: Signals the beginning of the new scene and beginning of XML
    - Attributes: none
    - Accepted Values: other elements
        - flag
        - score
        - character
        - dialogue

- Flag Tag: Declares local or global flag that will be used in a scene - there should be one tag for each flag within the scene.
    - Attributes:
        - key: the key used to look up the flag
        - type: the type of flag
            - "local": local flags are flags that are only used for a single scene
            - "global": global flags are flags used over more than one scene in the game
    - Accepted Values: default value – boolean

- Score Tag: Similar to flag tag, declares local or global score that will be used in a scene - there should be one for each score within the scene. (NOTE - the "level" score must be declared here with a default value of 0 in each scene file)
    - Attributes:
        - key: the key used to look up the score
        - type: the type of score
            - "local"
            - "global"
    - Accepted Values: default/start value – integer

- Character Tag: Adds a character to the scene.
    - Attributes:
        - key: the key used to identify the character in the scene. Will either be "pc" for Sydney or "npcX", where X is a number (1, 2, 3,...).
    - Accepted Values: other elements
        - asset
        - disp

- Asset Tag: The name with which the asset for the character is looked up.
    - Attributes: none
    - Accepted Values: text - name of the master file, where the master file is the file with the character's default emotion.

- Disp Tag: The display name for the character.
    - Attributes: none
    - Accepted Values: text - name of the character
- Dialogue Tag: Signals the beginning of the dialogue Attributes: none
    - Accepted Values: other elements
        - response

- Response Tag: Represents a single line of dialogue, item description, or narrative line a any associated data or updates pertaining to the line. NOTE: very long lines, narrations, descriptions can be broken up into several responses.
    - Attributes:
        - key
        - level: the level at which the line is located in the dialogue
            - the very first line of dialogue in the scene is at level="0"
            - to end the scene, the level must be updated to "-1" in the final response in the scene
            - The level given to a response should be viewed as an ID or name for the level, rather than the actual position in the dialogue path (i.e. in some files level 10 may come before level 3). This is for ease of adding new responses/levels at arbitrary points in the file.

                - If there are multiple choices (for PC) or possible responses (for NPC) to line of dialogue, they must be located at the same level.
            - cond: If there is a condition associated with one of the flags or scores in the scene, the value for this attribute should be the key for the flag or score.
            - val: IFF there is a cond attribute defined, the val tag represents the value that the flag or score should be set to in order for this line to be displayed.
    - Accepted values: other elements
        - update
        - image
        - location
        - time
        - line

- Update Tag: Defines an update that must be made to a global or local score when a specific response is selected.
    - Attributes
        - key: the key for the score or flag
        - type:
            - score
            - flag
    - Accepted Values
        - For incrementing a score: positive integer (i.e. 9)
        - For decrementing a score: negative integer
        - For setting a score to a specific value: character 's', followed by the integer value for the score (i.e. s9 or s-2)
        - For setting a boolean to a specific value: true or false
        - For setting a boolean based on a threshold: equality or inequality (i.e. suspicion<6 or tries==3)

- Image Tag: Notifies the parser of which image to associate with the current line.
    - Attributes: none
    - Accepted Values - text
        - character emotion IFF not the default emotion for the character (available emotions for each character are listed on the character assets page).
        - asset name for narrative or scene image

- Location Tag: IFF in a narrative scene, the location can be specified with this element.
    - Attributes: none
    - Accepted Values - text: location (Ray's Diner, Sydney's Room, etc.)

- Time Tag: IFF in a narrative scene, the time can be specified with this element.

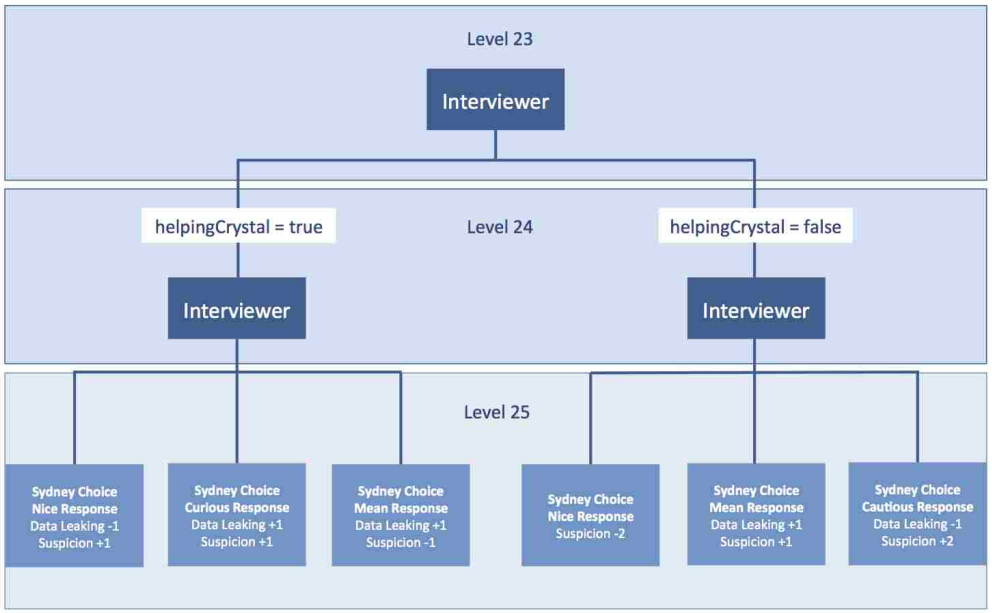
    - Attributes: none
    - Accepted Values - text: time of day (lunch time, evening, etc.)

- Line Tag: Element containing the actual line of dialogue.
    - Attributes: none
    - Accepted Values - text

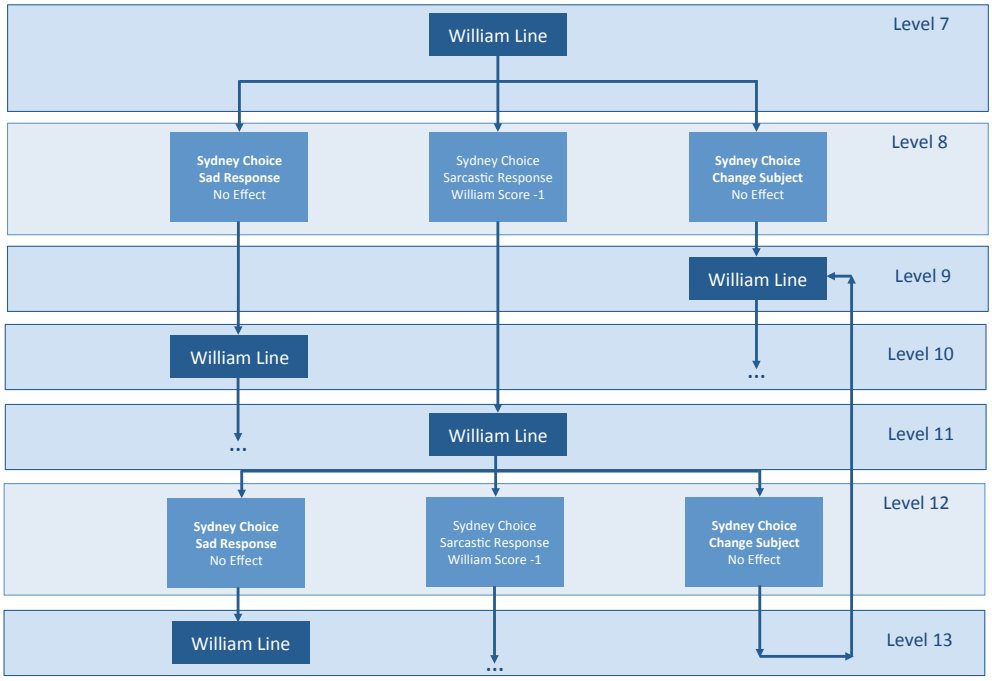
APPENDIX B

Scene Dialogue Diagrams

Example with conditional NPC lines and corresponding PC response options:



Example showing graph structure of dialogue:

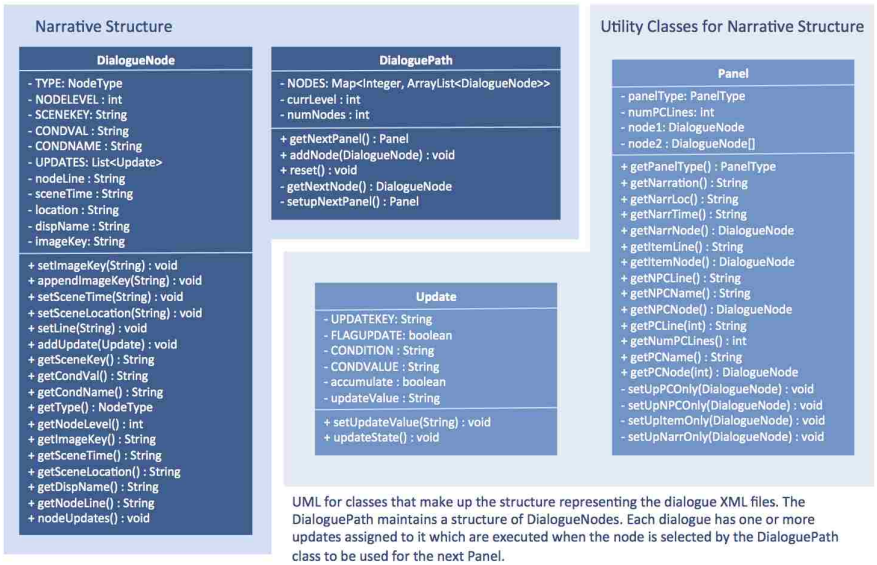


APPENDIX C

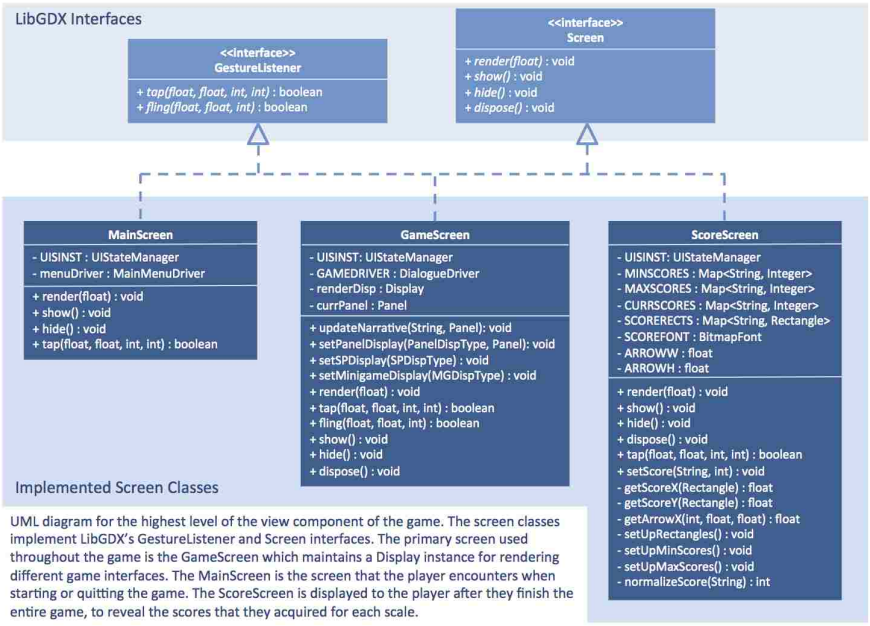
Game Architecture Diagrams

Model Component UML: Data State Maintenance

Model Component UML: Dialogue Structure

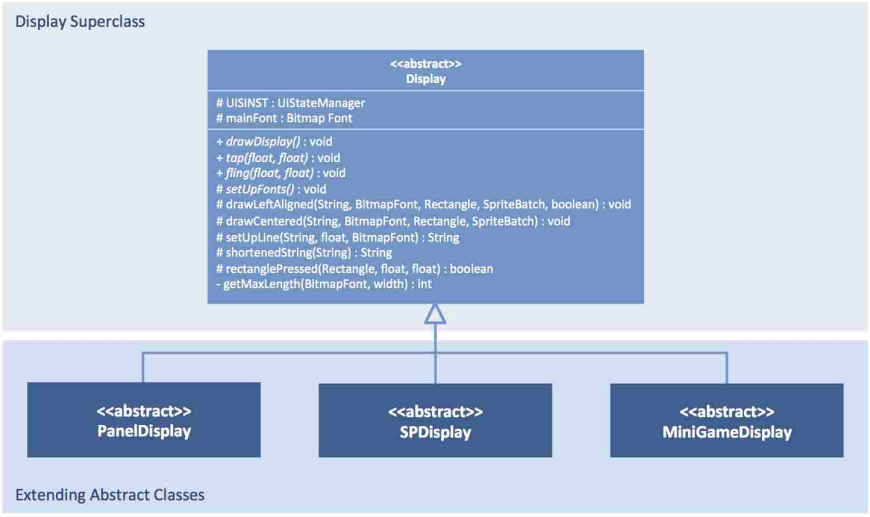


View Component UML: Screen Classes



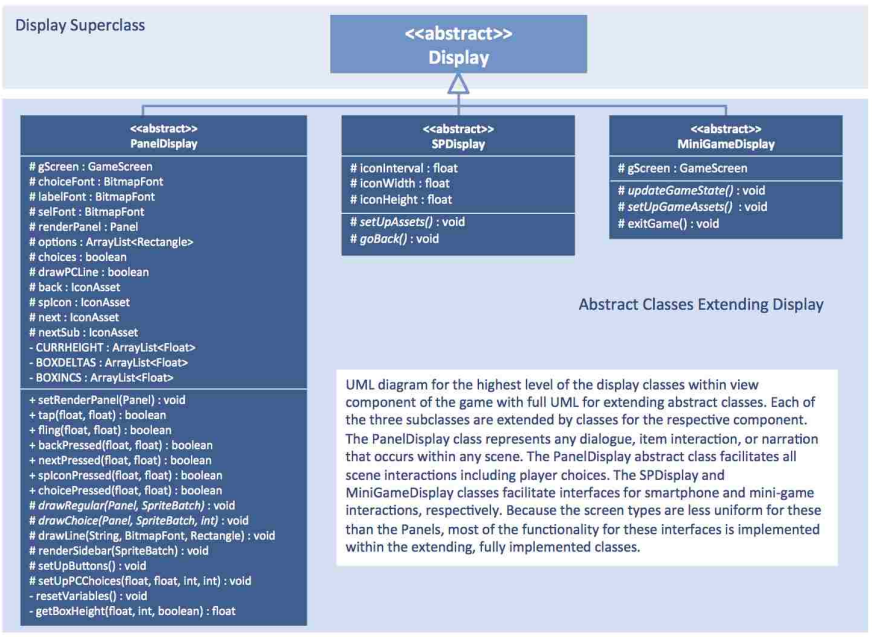
UML diagram for the highest level of the view component of the game. The screen classes implement LibGDX's GestureListener and Screen interfaces. The primary screen used throughout the game is the GameScreen which maintains a Display instance for rendering different game interfaces. The MainScreen is the screen that the player encounters when starting or quitting the game. The ScoreScreen is displayed to the player after they finish the entire game, to reveal the scores that they acquired for each scale.

View Component UML: Display Abstract Classes

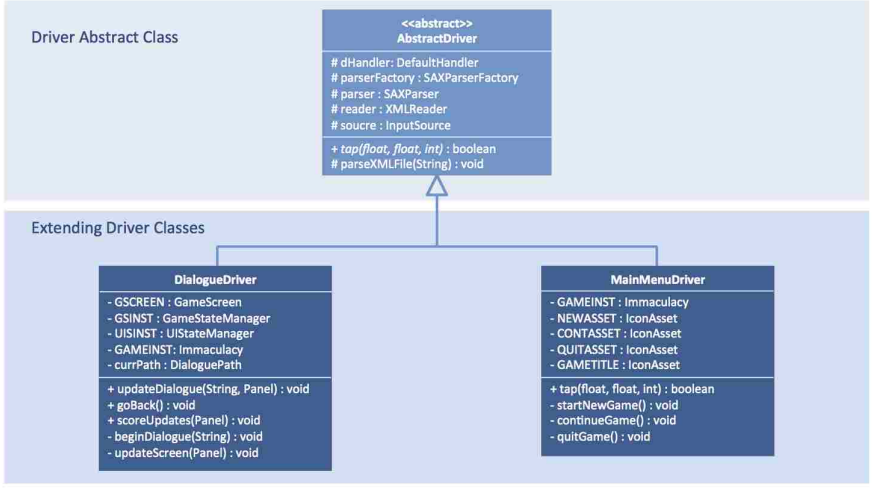


UML diagram for the highest level of the display classes within view component of the game with full UML for Display superclass. An instance of Display is maintained in the GameScreen class through which the rendering is facilitated. However, the functionality and design of each interface is implemented within the classes extending the PanelDisplay, SPDisplay and MiniGameDisplay abstract classes.

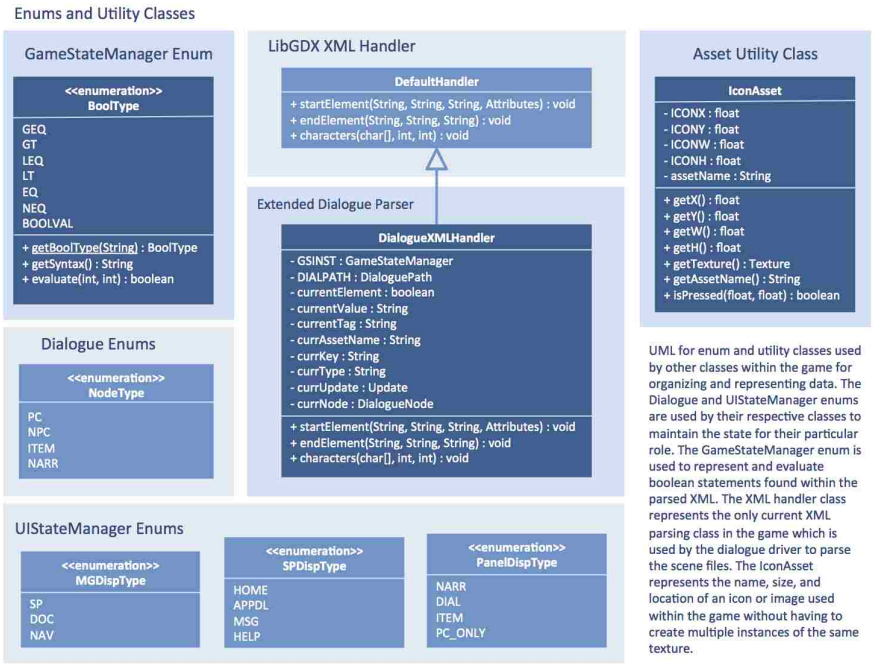
View Component UML: Display Abstract Classes



Controller Component UML: Driver Classes



UML diagram for the driver classes which act as the primary bridge between the user interface in the view component and the data that is maintained in the model component. The AbstractDriver provides its extending classes with XML parsing functionality as well as the ability to relay player input from the view component to the controller component. The DialogueDriver facilitates all interactions for the GameScreen and ScoreScreen classes, whie the MainMenuDriver provides interaction with the MainMenuScreen class.



UML for enum and utility classes used by other classes within the game for organizing and representing data. The Dialogue and UIStateManager enums are used by their respective classes to maintain the state for their particular role. The GameStateManager enum is used to represent and evaluate boolean statements found within the parsed XML. The XML handler class represents the only current XML parsing class in the game which is used by the dialogue driver to parse the scene files. The IconAsset represents the name, size, and location of an icon or image used within the game without having to create multiple instances of the same texture.

REFERENCES

Android. Android SDK. http://developer.android.com/index.html.

Denning, Tamara, Tadayoshi Kohno, and Adam Shostack. "Control-Alt-Hack TM: A Card Game for Computer Security Outreach, Education, and Fun." (2012).

Ensafi, Roya, Mike Jacobi, and Jedidiah R. Crandall. "Students Who Don't Understand Information Flow Should Be Eaten: An Experience Paper." *CSET*. 2012.

Felt, Adrienne Porter, Serge Egelman, and David Wagner. "I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns."*Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*. ACM, 2012.

Irvine, Cynthia E., Michael F. Thompson, and Ken Allen. "CyberCIEGE: gaming for information assurance." *Security & Privacy, IEEE* 3.3 (2005): 61-64.

Jordan, Craig, et al. "CounterMeasures: a game for teaching computer security." *Proceedings of the 10th Annual Workshop on Network and Systems Support for Games*. IEEE Press, 2011.

Kelley, Patrick Gage, et al. "A nutrition label for privacy." *Proceedings of the 5th Symposium on Usable Privacy and Security*. ACM, 2009.

Kelley, Patrick Gage, Lorrie Faith Cranor, and Norman Sadeh. "Privacy as part of the app decision-making process." *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2013.

LibGDX. LibGDX SDK. http://libgdx.badlogicgames.com.

Microsoft. Elevation of Privilege (EoP) Card Game. microsoft.com/en-us/sdl/adopt/eop/aspx.

Mikengreg. *TouchTone*. touchtonegame.com. 2015.

Mrgan, Neven and James Moore. *Blackbar*. mrgan.com/blackbar. 2013.

Office of Justice Programs. Mission: Laptop Security. http://www.onguardonline.gov/media/game-0008-mission-laptop-security.

Sheng, Steve, et al. "Anti-phishing phil: the design and evaluation of a game that teaches people not to fall for phish." *Proceedings of the 3rd symposium on Usable privacy and security*. ACM, 2007.

Squires, Jim. *TouchTone Review: Puzzling Is Your Civic Duty*. www.gamezebo.com/2015/03/20/touchtone-review-puzzling-is-your-civic-duty/. 2015.

Suknot, April, et al. "Immaculacy: a game of privacy." *Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play*. ACM, 2014.

Welch, Chris. *'Blackbar' is an iOS puzzler and a commentary on the NSA's love of redacted text*. www.theverge.com/2013/8/23/4651450/blackbar-an-ios-puzzler-and-commentary-on-nsa-redacted-text. 2013.

Young, Alyson L., and Anabel Quan-Haase. "Information revelation and internet privacy concerns on social network sites: a case study of facebook."*Proceedings of the fourth international conference on Communities and technologies*. ACM, 2009.