Willy and the Wumpus Post-Mortem 10/16/2015

Willy and the Wumpus is a point and click, text-based adventure game where you must enter a cave and slay the monster known as a Wumpus. The game is very simple and was intended to be, as my focus was primarily on the application state management system, user input system, and user feedback systems the game is built on top of. Every game should have intuitive methods of navigating the application, easy to use controls, and excellent visual and audio cues that keep them informed on the current state of the game and application. A lack of a well implemented back-bone consisting of these elements can make an application frustrating to use and take attention away from the game itself. Overall, the resulting application is very easy to use and is much more appealing than a terminal based Hunt the Wumpus game because of the systems it uses, despite it having an almost identical set of rules as a normal Hunt the Wumpus implementation.

The main challenge I faced in creating Willy and the Wumpus was implementing a basic OpenGL based user interface, as I never have before. Having read the entire Game Programming Patterns book (http://www.gameprogrammingpatterns.com/contents.html) a week before implementing the game, I chose to use the Observer Pattern discussed in the book to allow user interface objects to observe the input class for events. Each user input object would just have to request to observe input, and then execute a command when the input happened. At this point I ran into my next major issue. I had not figured out how to make different buttons execute different commands when clicked, but I knew there must be a way to pass functionality to an object. After some research on function pointers and Lambda functions, I decided that Lambda functions were powerful and exactly what I needed. Each button is passed in a function for various events such as pressed, released, and mouse movement that get executed when the event occurs, giving my user interface a lot of flexibility. From there I was able to implement a text box class that utilizes buttons for its scroll bar, which gave me everything I needed to implement a point and click, text-based adventure game.

Before implementing the game, there was one more problem I wanted to address. In my previous game, Gravity Pong, the game implementation relied on a few global dependencies such as graphics, audio, and input classes that were net handled well and caused the code to slowly become a chaotic mess. I decided to solve some of these issues by implementing a Service Locator Pattern that I also read about in the Game Programming Patterns book. The pattern allows these services to still be globally accessed, but abstracts them so that the service implementations can easily be switched out without the game implementation needing to change. Overall, this did a lot to clean up my code and make it more sustainable.

Willy and the Wumpus has a very limited set of rules and was not difficult to implement. A large portion of my time was spent designing the state management system and the various different screens needed for the application, including the splash screen, main menu screen, game screen, rules screen, and exit screen. I animated the splash screen so that launching the game would be memorable, and spent time designing screen layouts for each screen so that they could easily be identified as different components of the application. Navigation between states either is automatically caused from the passing of time or by clicking on a transition button on the screen. One issue I had to worry about when transitioning between states was making sure that only the current state was listening to input, so interactive states had to make sure to stop observing when they went out of view and start observing again if they came back in view. Below is a state diagram for the Hunt the Wumpus Application.

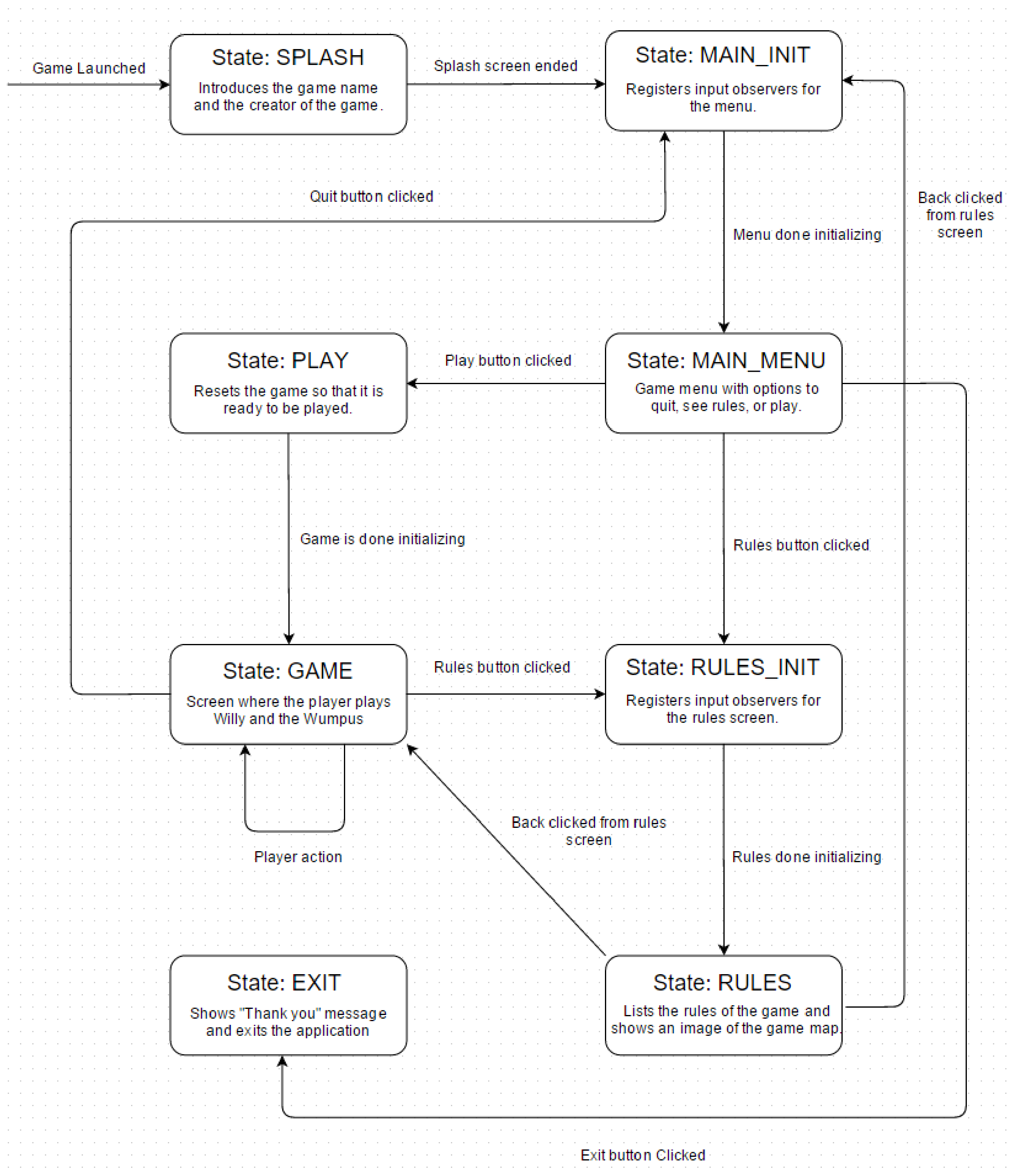
Game Launched
State: SPLASH
Introduces the game name and the creator of the game.
Splash screen ended
State: MAIN_INIT
Registers input observers for the menu.
Quit button clicked
Back clicked from rules screen
Menu done initializing
State: PLAY
Resets the game so that it is ready to be played.
Play button clicked
State: MAIN_MENU
Game menu with options to quit, see rules, or play.
Game is done initializing
Rules button clicked
State: GAME
Screen where the player plays Willy and the Wumpus
Rules button clicked
State: RULES_INIT
Registers input observers for the rules screen.
Back clicked from rules screen
Player action
Rules done initializing
State: EXIT
Shows "Thank you" message and exits the application
State: RULES
Lists the rules of the game and shows an image of the game map.
Exit button Clicked

Figure 1: Willy and the Wumpus state diagram.

After the game was fully functional, I focused my attention on improving user feedback. The first thing I did was add click sound effects to buttons so that the user received audio feedback upon valid interaction. I then modified the text box of the game so that it supported multiple colors, and made player actions, informative events, positive events, and negative events all display in different text colors. I then improved the user experience further by added sound effects to different events in the game. For example, a shriek plays when a bat is nearby and you sniff the air when you are close enough to smell the Wumpus. Finally, to improve the atmosphere of the entire game, I added in some eerie background music. Overall, Willy and the Wumpus was a good learning experience, even if the game itself is not very interesting. If I were to continue working on it, I would look into ways of putting input events on a separate thread, optimize my text rendering method, and improve upon the game’s visuals.