Christopher Katnic  
894-68-7516

Project Summary and Design Report

**Introduction:** The project is called Dungeon Runner. The archetype can be described as a dungeon crawler with turn-based combat. The player controls an explorer traversing multiple floors of a dungeon, each of which contain stronger and stronger monsters to defeat. The player must utilize time in combat effectively, alternating between healing and attacking, in order to successfully navigate each encounter and subsequent floor. The objective is to travel down as far into the dungeon as possible before perishing.

**Design:** Upon starting the game, a player is greeted with a prompt to enter their name. The game shall store this name locally. Then, a splash screen will display, briefing the player about the rules and objective of the game. Finally, the game screen will initialize with one large display that encapsulates three smaller displays: a minimap, an encounter window, and a log window.

Once the game screen has been updated, the player shall be able to begin navigation of the dungeon, from top to bottom, using the WASD keys to control navigation. The player icon is denoted as the yellow square in the minimap. The navigation ambience music will begin.

Every step that the player takes through the dungeon, there is a 5% chance that the player will be engaged in an encounter with an enemy. The enemy that the player encounters depends on the floor that the player is on. Pixies will reside on the first floor, Lizardmen will reside on floors 2-3 and golems reside on the floors greater than 4. Pixies, lizardmen, and golems all will be generated as the same or lower level as the hero. Once the enemy is generated, the battle begins. The battle music begins.

Battle is turn based, always starting with the player. The player has two options for battle: attack, and heal. The enemy monster will only ever attack. Opponents alternate, performing one single action at a time, and then passing the turn. The battle ends when the first character’s health reaches 0.

When a character attacks, it does an integer amount of damage to the opposing character. The character defends the attack, which subtracts a small amount of the damage. Any other damage amount greater than 0 is then subtracted from the defending character’s health point total.

The amount of damage that a character does with an attack, as well as the damage prevented while defending, is based on the type of character that it is, and the level that that character is at the time. The player, pixies, lizardmen, and golems all have different attack and defense modifiers, and as a result, do a different range of damage. Additionally, every attack’s base value is picked randomly from a small range of values, depending on the character type.

Enemy characters have a 20% chance of scoring a critical hit, which is an attack with a higher base damage range, which usually results in a much stronger attack.

If the player’s health total reaches 0 first, the game ends, and the game over splash screen is displayed.

If the enemy character’s health total reaches 0 first, the player levels up, increasing the internal level by 1, and healing an amount based on the player level and a base amount. The victory sound will play, after which the player continues navigation.

A log window on the right side of the screen will display text that describes what is happening at every point during the game including every turn of combat, the beginning and end of each battle, when the explorer enters the next level of the dungeon, and at the beginning of the game to welcome the player.

An engagement window on the bottom of the screen will show images depicting the player in battle with each particular enemy, as well as an image when the player levels up, and an image for when navigation resumes.

At any time a player may exit the game by pressing the ESC key.

At a minimum, the user playing this game must be able to follow instructions on a screen, as well as basic mathematic skills. A player should consider a few turns in advance for planning each action, as well as take into consideration the increasing difficulty of each floor before advancing.

Generally, the game can last from 5 minutes to 50, depending on the amount of time that a player spends on each floor before progressing to the next. If the player attempts to descend floors as quickly as possible, they will be matched with monsters that are generally too difficult to defeat.

**Software Architecture:** The software architecture used was Model-View-Controller, and is based on two primary software design patterns: object oriented design pattern, and factory design pattern. Additionally, the singleton design pattern was implemented, but was not critical to the design.

The object oriented design pattern was implemented to reduce code complexity, increase maintainability, and reduce the impact of changes during rework. The Character class is the parent class of both the player class and the villain class. The villain class is the parent class of the pixie, lizardman, and golem class. The character class implements the generic methods that all characters perform. The player subclass includes methods for leveling up, healing, and overrides the attack method. The villain subclass includes a type function to return the type of villain, as well as implements a critical hit method. Each of the villain subclasses override the critical hit and type methods, as well as the character class initializer method. This design pattern made adding additional character types and changing current character types extremely easy; additional villain subclasses only required adding about 7 lines of code before they could be engaged in-game, and changing existing classes had no impact on the rest of the code structure.

The factory design pattern was implemented for the window and subwindow generation. The factory design pattern is concerned with the creation of similar objects with different configurations. The screengenerator class is responsible for configuring each one of the Surface objects that is displayed at runtime. The screengenerator creates the main window, the mnimap subwindow, the enouncter subwindow, the log subwindow, as well as any text fields (including every message that scrolls through the log subwindow), and returns them as Surface objects, which are then displayed on screen. This was an extremely useful design pattern to implement for a number of reasons. First, by decoupling the Surface object instantiation from the main code, the code structure looks much cleaner. Second, any changes that need to be made to the structure of Surface objects needed only be changed in one place. Third, future Surface configuration requirements can be easily implemented without needing to touch the main code. Fourth, simple configuration variances that need to be applied to the same subwindow can simply be passed as a single argument into the same method, reducing the amount of complexity in

In the MVC, the main function acted as the controller/delegate, responsible for switching between the views (Surfaces), and calling methods that acted on the models (objects like the characters, surfaces, etc.) which were displayed on the views. This was a very simple implementation of MVC, unlike what is normally seen in complex GUI based software titles such as iOS/Android applications, but still functions in a similar fashion.

The singleton design pattern was implemented to enforce a single, globally available instantiation of the screengenerator and musics classes. In doing so, the two resulting objects were accessible at any point inside of the controller (the main function) which often requires access to these two objects at many different points during runtime.

**Game Demonstration:** The below pictures show examples of the implementation of the design outlined in the second section. They show the user name, splash screen, main display, and battle

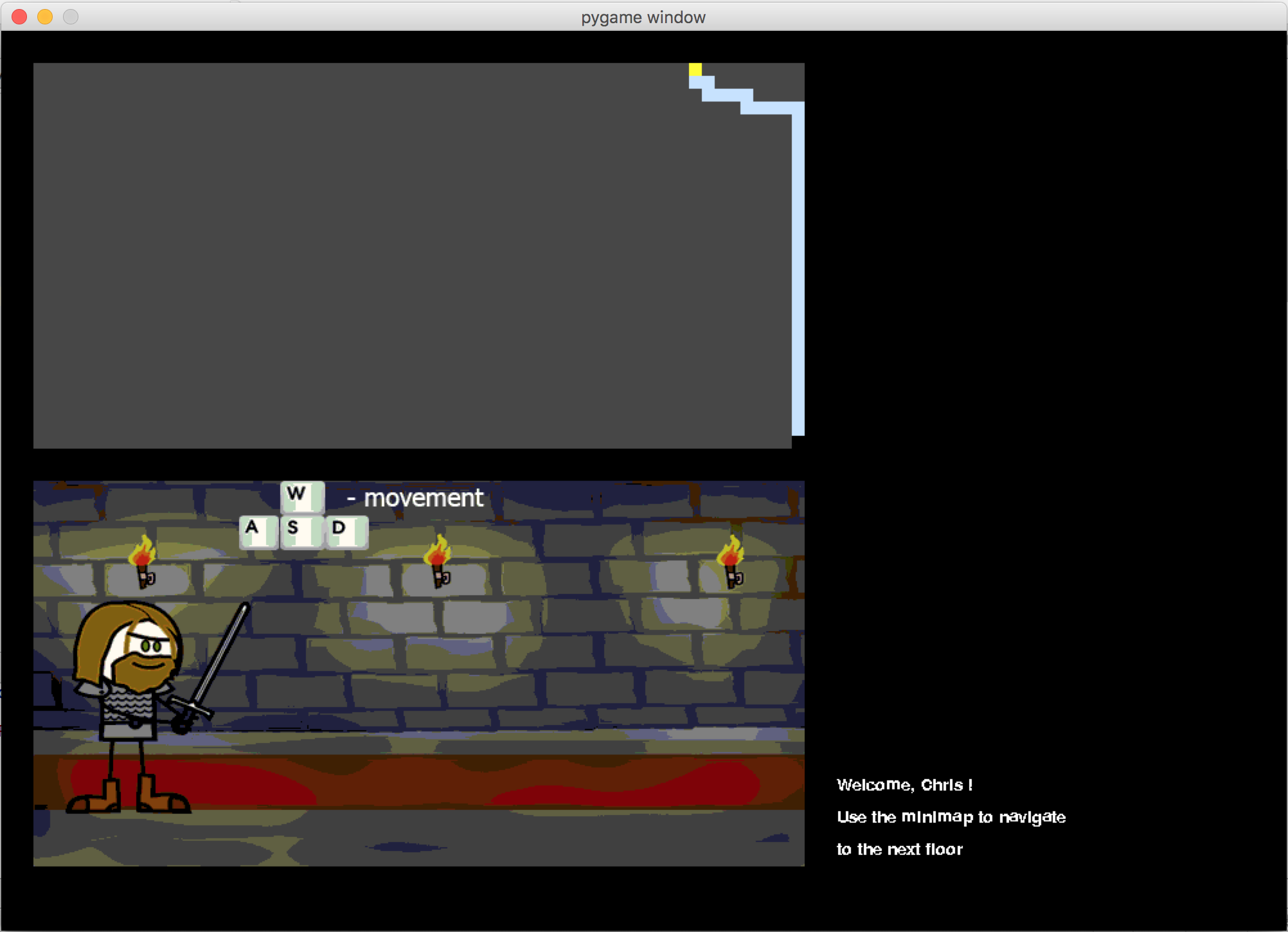
**First Screen: User enters name**

****

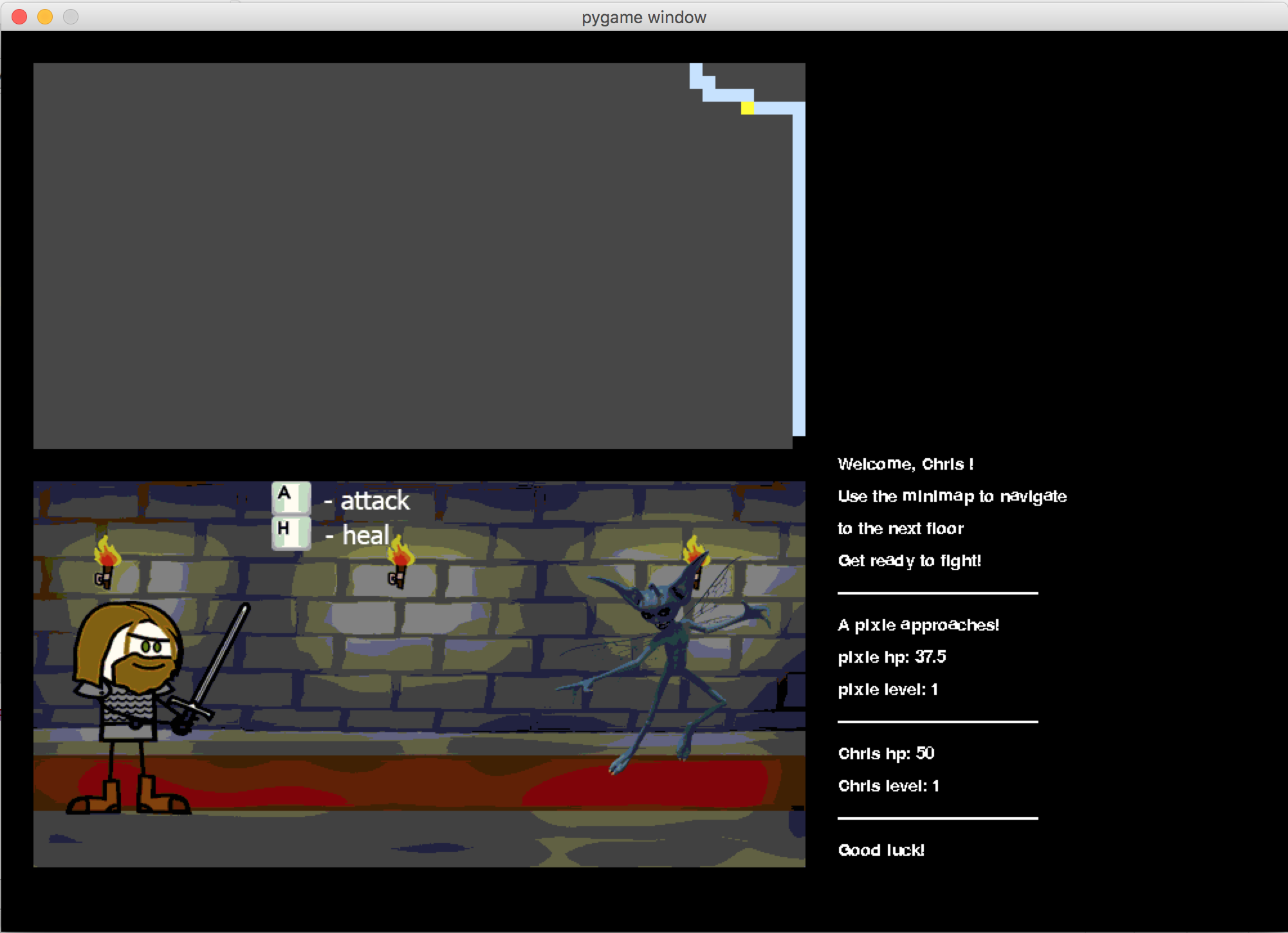
**Splash Screen: quick explanation before game begins**

****

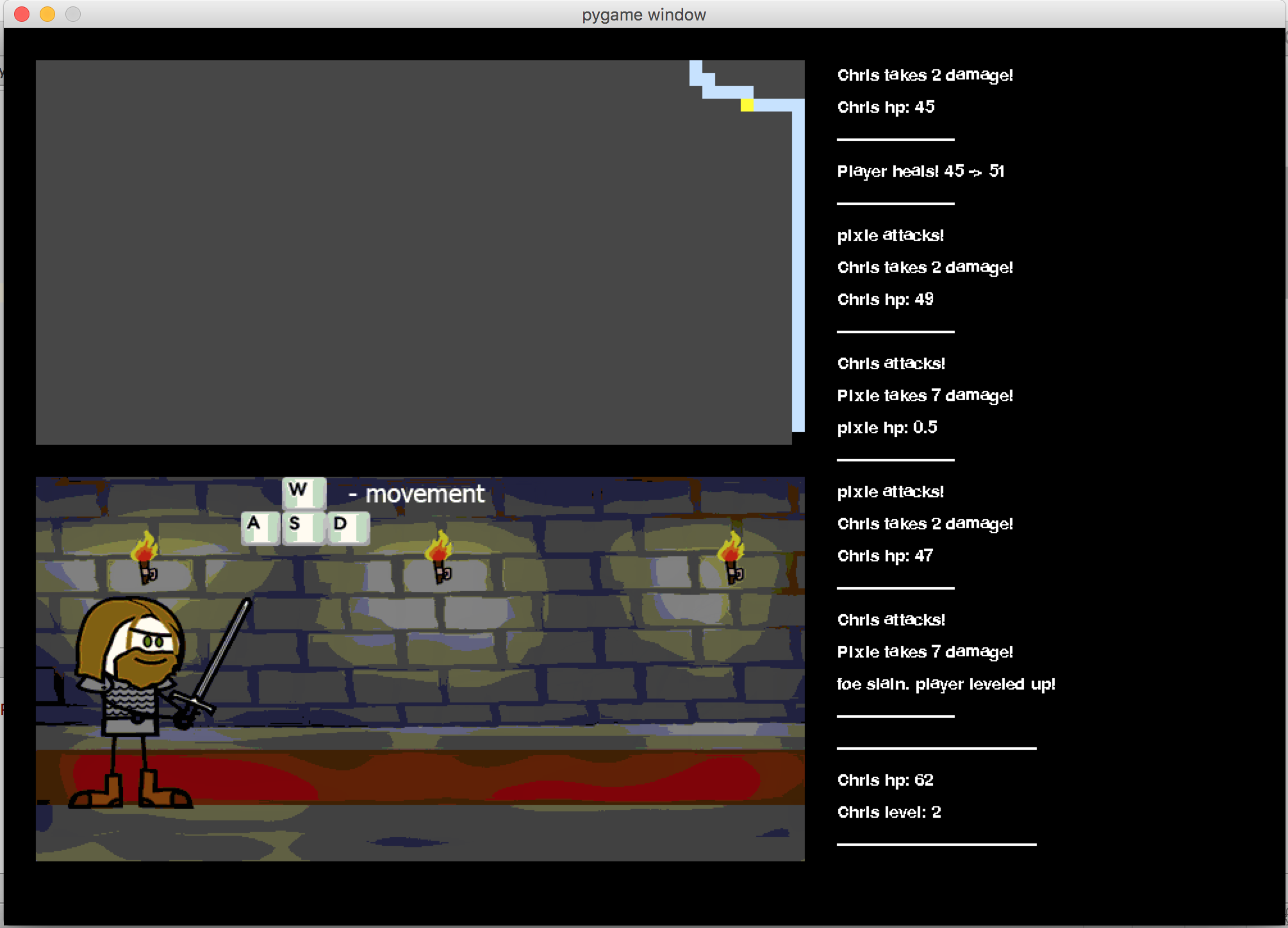
**Main screen: Start navigation**

****

**Navigation and begin encounter with pixie 1st floor**

****

**Post battle: victory/level up and continue navigation**



**Post battle: game over**

****

**Bibliography**

**Protagonist Image:** Generic Warrior 1 – Thormag on Deviantart: thormag.deviantart.com/art/Generic-Warrior-1-128317333

**Golem Image:** Golem sprite – Akira Toriyama, Bird Studio, Dragon Quest Series (1986-2015)

**Lizard Image:** Lizardman sprite – Atlas/Vanillaware Studios, Dragon’s Crown (2013)

**Pixie:**  Wolverine041269 on Deviantart: wolverine041269.deviantart.com/gallery/41877817/Evil-Pixies-Stock

**Dungeon background:** Dungeon Fight Background – JerfyBawss on Moddb.com: www.moddb.com/games/cakequest/images/dungeon-fight-background

**Battle Music**: Encounter Song – Pokemon Red Version, Junichi Masuda, Game Freak (1996)

**Victory Music:** Victory Anthem, Final Fantasy game series, SquareEnix (Squaresoft at the time of production) (1986)