Software Design Description

<An Endless Night>

<Jory Alexander, Estephanie Gonzalez, Aaron Knobloch, Caleb Sears >

# 1.0 Introduction

## 1.1. Purpose

< Clearly state the purpose of this *document* and its intended audience. >

*The purpose of this document is to outline nuances on how our game will be built. An Endless night is standalone software created for a user to interact and traverse through for the purpose of making it to the last room to beat the game. This document will give details on the UI, data storage, and subsystems.*

## 1.2. Scope

< State the dominant design methodology. Overview the architecture of the product briefly. Describe the external systems with which this system must interface. >

*An Endless Night is a text-based game, the application is free to download for any operating system that has latest java jre installed. Players are able to traverse the game, fight monsters, and solve puzzles through the use of keyboard commands. Administrators have access to back-end functionality, and can change the contents and aspects of the game.*

## 1.3. Glossary

< Define the technical terms used in this document. Do not assume the experience or expertise of the reader. >

* UI: User Interface, refers to the window that the User will interact with and enter in information to.
* Hero: Is the player object within the game that will be traversing through rooms and getting health adjustments throughout.

## 1.4. References

< List here any references to other documents cited anywhere in this document including references to related project documents, especially the SRS. Add references here when other project documents are created. This is usually the only Bibliography in the document. >

## 1.5. Overview of Document

< Describe the contents and organization of the rest of this document. Since there is already a Table of Contents, this overview will be less formal but more informative. Describe the remaining sections. >

# 2.0. Deployment Diagram

# 3.0. System Design

Subsystem Descriptions

•***Combat Subsystem:*** *The combat System is responsible for handling the interaction between hero and monster in which both character types will be able to attack one another. Users will be able to enter commands for the combat system through the interface subsystem.*

*•* ***Room Management Subsystem****: The room management subsystem will be responsible for the generation of rooms.*

*•* ***Interface Subsystem****: The Interface subsystem will be responsible for the interpretation of user input and the interaction between subsystems. Beginning tasks including but not limited to combat, artifact manipulation, and room interaction.*

*•* ***Inventory Subsystem****: The inventory subsystem will be responsible for keeping track of all artifacts available within the game system.*

Design Goals

# 4.0. Data Structure Design

< Design in detail and specify the persistent data structures to be used in the implementation. If these include databases, define the table structure of all databases including full field descriptions and all relations. Graphical languages are appropriate. Note that a database is an object and may have been fully described in the previous section. Each subsystem in your program should be a subheading. >

## 4.1 Subsystem 1

4.1.1 Combat Subsystem (Aaron Knobloch)-

The combat subsystem is responsible firstly for containing the data associated with the

Hero and the monsters, collectively known as characters. The combat subsystem will also

be responsible for handling the interactions between the characters through combat. combat involves calculating damage between characters by first accessing their damage, defense and health attributes and then adjusting their health accordingly.

4.1.2 Description of Each object with attributes and methods described

* Character:
  + CharacterID – The unique identifier of this character.
  + Health – The value associated with the current health of this character. Once this reaches zero, the game will end.
  + Strength – The value associated with the attack power of this character, or the amount that the character can reduce from the health of another character.
  + Name – The String representation of this character’s identification.
  + Location – The current room ID that the character is located in.
  + DefenseValue – The value associated with the defense of this character
  + getCharacterID – returns the CharacterID
  + getHealth – returns the health of the character.
  + doDamage(int damage) – substracts the damage amount from the health.
  + Heal(int amount) – adds the amount specified to the health
  + getStrength – returns the strength of character.
  + setStrength(int amount) assigns the specified amount of strength to the character
  + getDefense – returns the amount of defense associated with this character
  + setDefense(int amount) – assigns the specified value to this characters defense
  + getName – returns the name of the character
  + getLocation – returns the ID of the room this character is located in
* Hero:
  + statusConditions – An array of the status conditions inflicting this character
  + equippedWeapon – The weapon that the hero currently has equipped
  + equippedArmor – The armor that this hero currently has equipped
  + playerInventory – an array of all the items equipped in this character’s inventory
  + getEquippedWeapon – returns the weapons the hero has equipped
  + setEquippedWeapon(Weapon weapons) – equips the weapon specified
  + getEquippedArmor – returns the armor the hero has equipped
  + setEquippedArmor(Armor armor) – equips the armor specified.
  + getStatusConditions – returns an array of the status conditions afflicting this hero
  + addStatusCondition(StatusCondition) – adds this condition to the hero
  + removeStatusCondition(StatusCondition) – removes this condition from the hero
  + getPlayerInventory – returns an array of all of the items in a player inventory
  + addArtifactToInventory(Artifact item) – adds an item to inventory
* Monster:
  + ProbabilityOfAppearing – the probability that this monster has of appearing in a room
  + DroppedItem – an item that this monster drops.
  + getProbabilityOfAppearing – returns the ProbabilityOfAppearing
  + getDroppedItem – returns DroppedItem
  + setDroppedItem – cycles through possible items and sets it

## 4.2 Subsystem 2

4.1.1 Room management Subsystem (Estephanie Gonzalez)- more detail than requirements analysis and system design.

4.1.2 Description of Each object with attributes and methods described

## 4.3 Subsystem 3

4.1.1 Interface Subsystem (Jory Aexander)-

4.1.2 Description of Each object with attributes and methods described

## 4.4 Subsystem 4

4.1.1 Inventory Subsystem (Caleb Sears)-

4.1.2 Description of Each object with attributes and methods described

## 4.5 Subsystem 5

4.1.1 Object Model - more detail than requirements analysis and system design.

4.1.2 Description of Each object with attributes and methods described

### 

# 5.0 Flow of Control

This section consists of any new Sequence Diagrams and/or Flow Diagrams detailing various scenarios/use cases and how they occur using the objects described in section 4.0. DO NOT INCLUDE THOSE SEQUENCE DIAGRAMS PROVIDED IN REQTS ANALYSIS.

n/a

# 6.0 User Interface Design

< Describe the user interface. Prototype screenshots are acceptable. You should discuss the expected effectiveness of your design. >

*The user will be presented with a variety of input options on the console. The user will enter their command into the console and new information will be displayed based on the command.*

# 7.0 Help System Design

< Describe the **structure** of the help system and how it is to be accessed. Will it be context sensitive? Will there be a system reference capability? >

*At any time the user will be able to enter in the word help. The system will recognize the word regardless of case. The user will be given a list of general commands, relative to the context of their current menu, followed by a brief description of each command.*

# Index

< Anything not included in this document that helps you get started on the code. >