Bilkent University

Department of Computer Engineering



CS 319 Project

JCrawl: 2D Top-down Adventure Game

Analysis Report

Group Members

* Arda Yücel
* Cheol Woo Park
* Fatih Taş
* Mustafa Fidan

Contents

[1 Introduction 1](#_Toc254263637)

[2 Proposed System 1](#_Toc254263639)

[1.1 Overview 1](#_Toc254263640)

[1.2 Functional Requirements 1](#_Toc254263641)

[1.3 Non-functional Requirements 1](#_Toc254263642)

[1.4 Pseudo Requirements 2](#_Toc254263643)

[1.5 System Models 2](#_Toc254263644)

[1.1.1 Scenarios 2](#_Toc254263645)

[1.1.2 Use-Case Model 2](#_Toc254263646)

[1.1.3 Object and Class Model 2](#_Toc254263647)

[1.1.4 Dynamic Models 2](#_Toc254263648)

[1.1.5 User Interface 2](#_Toc254263649)

[3 Glossary 2](#_Toc254263650)

[4 References 2](#_Toc254263651)

Analysis Report

JCrawl: 2D Top-down Adventure Game

# Introduction

JCrawl is a purely Java based top down adventure game, inspired by adventure games like Binding of Isaac, while borrowing some gameplay elements from classics like Legend of Zelda.

**Binding of Isaac link:**

<http://store.steampowered.com/app/113200/>

The primary focus of this game will be heavily on Player-to-Object and more specially, Object-to-Object interactions.

The gameplay will be fusion between Binding of Isaac and Classic Zelda, although it will be important to shoot at the enemies and kill them to unlock other rooms, some rooms will incorporate puzzles instead of combat and some will just contain both.

# Proposed System

## Overview

The player controls a single character with variety of skills to explore myriad of rooms filled with dangers, loot and puzzles. The players will be able to move freely in all directions, but shoot at the direction they are looking at only initially. There will be many kind of items which will further enhance player's capabilities (Temporary power ups/ Permanent upgrades). Some enemies will be stationary, some enemies will have hard coded behavior, and some enemies will have independent basic AI. There will be two types of objects: Independent/Dependent; Independent objects (e.g. Fire traps, Automatic doors) will perform their intended behavior regardless of whether the player input is present or not. Dependent objects (e.g. Light switch, Doors) will perform their intended behavior IF it receives a player input.

The level design will be editable by user, allowing for further replayability.

## Functional Requirements

1. **Play Game**
2. **Change Options**
3. **Display Credits**
4. **Display Instructions**
5. **Open Beastiary**
6. **Pause Game**

## Non-functional Requirements

* Game Performance
* Graphics Performance
* User-friendly Interface
* Reusability

## Pseudo Requirements

1. The code will be written in Java only
2. Desktop only

## System Models

### Scenarios

A hyperlink is here.

### Use-Case Model

A code segment is below:

for (i=1; i<=5; i++)

System.out.println(“report to write”);

If you need to inline code, use “this” style.

### Object and Class Model

Table 1 is an example table.

Table An example table

|  |  |
| --- | --- |
| Key | Value |
| key | Value |

### Dynamic Models

### User Interface

# Glossary

Glossary for any domain-specific terms you use in your report.

# References

1. Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.