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.

* **Enemies**

1. *Slime:* Weak enemy that moves randomly. (AI)
2. *Wolf*
3. *Orc*
4. *Goblin:* Weak enemy that patrols in fixed route. (Hard coded)
5. *Skeleton Archer*
6. *Final Boss (Placeholder name)*

* **Objects (Independent)**

1. *Dart Traps:* Shoots darts at the fixed direction.
2. *Spike Traps*
3. *Fire Traps*
4. *Rolling Stone of Death*
5. *Automatic Doors*
6. *Monster Spawner*

* **Objects (Dependent)**

1. *Door:* If the player has a key, it will open, giving access to new room.
2. *Light Switch*
3. *Button*
4. *Pressure Plate*
5. *Lever*

* **Power-ups**

Power ups offer temporary boost, or act as consumables to aid in player’s adventure during the play session.

1. *Restore Health:* Restores the health of the player by 1.
2. *Burst of Speed (Movement Speed)*
3. *Invincibility*
4. *Burst of Speed (Attack Speed)*
5. *Firepower Boost*

* **Upgrades**

Upgrades, unlike power-ups, offer permanent boost to player’s capability, sometimes unlocking completely new gameplay mechanic with it.

1. *Rapid fire:* Permanently boosts the player’s shooting speed.
2. *Multi direction fire*
3. *Boots of Speed*
4. *Armor*
5. *Regeneration*
6. *Fire proof cape*
7. *Double shot*
8. *Triple shot*
9. *Power fist:* Allows players to push blocks.

## Functional Requirements

1. **Play Game**

The purpose of the game is to reach the final room where you will encounter the final boss which if you defeat it, you will win the game. Majority of the player’s stats are prone to permanent upgrade with the items you find during your adventure. The players will need to find keys to gain access to new rooms by completing certain objectives depending on the room type (combat/puzzle/both).

1. **Change Options**

The user can change few options:

-Texture quality

-Screen size

-etc.

Texture quality will alter the type of Spritesheet that will be used in the game for tiles/objects/enemies. Screen size is self-explanatory, although it will be recommended to use high definition textures for larger screen size.

1. **Display Credits**

Simple display of credits for any external sources we might use and list of our names/IDs.

1. **Display Instructions**

As the game does have some degree of depth, the users might want to see full list of controls before jumping into the main game.

1. **Open Bestiary**

There will be many different types of enemies and their behavior will be unpredictable/surprising for the first time players. Therefore, the Bestiary will contain information about all enemies the players will encounter in the game.

1. **Pause Game**

Sometimes users have to answer the call of nature. <- Subject to Change

## 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.