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 player 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.

* **Player**

The player controlled character will possess variety of stats that the user can upgrade and utilize through out the course of the game: Health, Movement Speed, Attack Speed, Damage.

The player can also obtain completely new game mechanics with certain items, e.g. ability to push around light blocks, immunity to flame damage.

* **Enemies**

Enemies will have three different types of behavior set to them: Rudimentary AI, Hard code behavior (So they look like they are patrolling), or they are stationary. Rudimentary AI will cause enemies to roam around aimlessly, hard coded behavior will make them move in predetermined manner (so they look like patrolling certain areas for example), or they will be stationary and attack the player from range.

1. Slime: Weak enemy that moves randomly. (AI)
2. Wolf: Slightly stronger counterpart of Slime (AI)
3. Orc: Slightly stronger counter of Wolf (AI)
4. Goblin: Weak enemy that patrols in fixed route. (Hard coded)
5. Skeleton Archer: Stationary enemy that shoots arrows from a distance (Stationary)
6. Final Boss (Placeholder name): Killing this enemy wins the game (Hard coded)
7. Magmatrum: Challenging enemy which burns the ground it has passed (AI)

* **Objects (Independent)**

Independent objects execute their behavior with/without player’s intervention. For example, dart trap will constantly shower certain area with darts regardless of whether the player is in the area or not. However, this does not mean player cannot interact with the independent objects in a way that changes their behavior (e.g. pressing a button shuts down all the traps in the room)

1. Dart Traps: Shoots darts at the fixed direction.
2. Spike Traps: Melee version of Dart Trap
3. Fire Traps: Toastier version of Spike Trap
4. Rolling Stone of Death: Challenging trap that will instantly kill any entity on its path
5. Automatic Doors: A door that closes and opens with set interval
6. Monster Spawner: Spawns AI controlled enemies with long intervals between the spawn

* **Terrain**

Terrain types are usually for determining the graphics that a tile will use, but some terrain might have special perks attached to them, the main example being lava, which is still a walkable floor, but it will damage the players when walked on.

1. Floor: Standard terrain type with nothing special about it.
2. Wall: Standard impassable terrain.
3. Bridge: Different graphic version of what is essentially a floor.
4. Lava: Will damage the player when the player is standing on it. Can be rendered harmless with certain item.
5. Light Blocks: Lights up when the light switch is pressed.

* **Objects (Dependent)**

Dependent objects, contrary to independent objects, will not execute their predetermined behavior unless player gives any kind of input to them. However, they are the only way to change the existing behavior of the independent objects as well.

1. Door: If the player has a key, it will open, giving access to new room.
2. Light Switch: Turns on a light which will only have purely cosmetic value
3. Button: Triggers some other object (Could be a trap or a door)
4. Pressure Plate: Same as button but it has to be stepped on
5. Lever: Similar to button but unlike the button which trigger an event once, having the lever down will provide constant stream of event (The door stays open / Bridge stays extended / vice versa)
6. Block: Can be pushed around by the player if the player has obtained the prerequisite upgrade.

* **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): Increases movement speed
3. Invincibility: Player becomes immune to any source of damage
4. Burst of Speed (Attack Speed): Increases attack speed
5. Firepower Boost: Increases damage

* **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: Instead of firing towards the direction that the character is looking at only, player can now fire towards all directions at once
3. Boots of Speed: Permanently increases movement speed
4. Armor: Increases maximum health
5. Regeneration: Passively regenerates lost life
6. Fire proof cape: Ability to walk on lava
7. Double shot: Shoots two pellets instead of one
8. Triple shot: Shoots three pellets instead of one
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**

Allows players to pause the game in the case they need to take a break, or just tend to other matters.

## Non-functional Requirements

* **Game Performance**

The game will require to have low latency from each keystroke to make the gameplay as smooth as possible. Object management/rendering will also be a point of concern for performance as late game will offer higher number of objects on the screen simultaneously.

* **Graphics Performance**

As explained above, late game will feature higher numbers of objects, so rendering them as efficiently as possible is a notable concern to take note of. Also, it would give reason to add texture changing option as this game could be ran on low-end PC or laptop.

* **User-friendly Interface**

The user know exactly what is going at all times, which means the UI should not be cluttered with useless information and make it easy for users to figure out what is going on at their screen while they are playing the game.

* **Reusability**

It is planned to have modular system for level design so that users themselves can create their own content for replayability value. Also, this will reduce the amount of hard coding that would be necessary for level design. (More focus on tech rather than game design)

## Pseudo Requirements

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

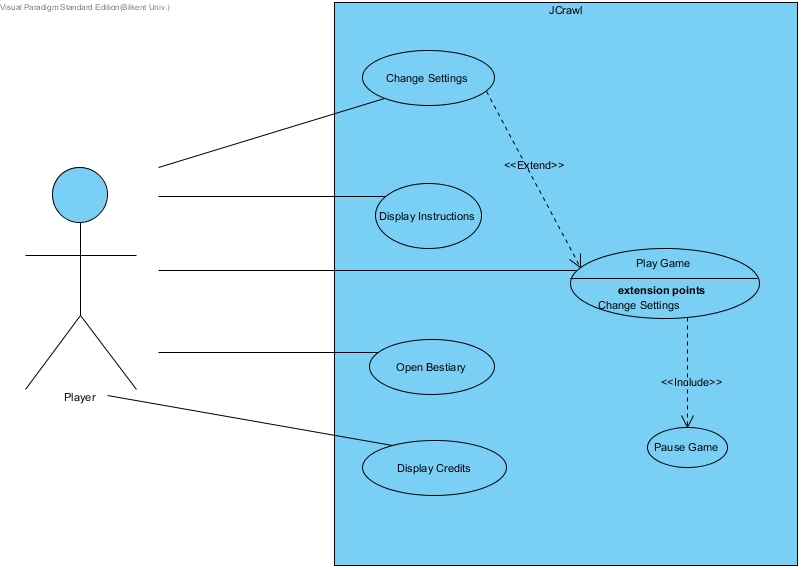
## System Models

### Scenarios

1. John, the user, had been playing the game, he passed the first level. Then, he remembered he could change texture quality to get better user experience. Despite the fact that he would lose his current game if he left the game, he changed the options and started again from the beginning. He was facing with the final boss with one health, when his phone rang. He paused the game and answered the phone. After a while, he looked too nervous and resumed the game, yet he lost his last health in the final room. Although he lost the game, he had lots of fun while playing the game. Therefore, he opened “credits” section to browse developers of the game.
2. Julia, the user, decided to play JCrawl, it would be her first experience. She directly had started the game, yet she could not figure out controls of the game. Therefore, she returned to the main menu and browsed instructions to know more about gameplay and controls. After that, she started game again, but her character died at second level. She realized there were different enemy types, so se decided to take a look bestiary to get more information about enemy types. Then, Julia began the game with enough information to beat the enemies.

### Use-Case Model

Use case diagram of JCrawl



**Figure 1.1.2.1**

1. **Change settings**

**Primary Actor:** Player

**Interests:** The Player wants to change some settings like texture quality or screen size.

**Pre-condition:** - Player has to be in Main Menu

**Post-condition:** -If player has changed any setting, new settings will be applied on the gameplay

**Entry Condition:** Player clicks on "Change Settings" in the main menu.

**Exit Condition:** Player clicks on ““Return to Main Menu” in the settings screen.

**Successful Scenario Event Flow:**

1. Player wants to change texture quality
2. Player gets the settings screen
3. Player changes game settings as he wishes
4. Player clicks on "Return to Main Manu"
5. System renders the main menu screen.

**Alternative Flows:**

1. Player wishes to return to main menu
   1. Player clicks on “Return to Main Menu”
   2. System renders the main menu screen.

2. Player wishes to see settings

a.Player gets the settings screen

b.Player checks the settings and doesn't make any change

c. Player clicks on “Return to Main Menu”

d. System renders the main menu screen.

1. **Display Instructions**

**Primary Actor:** Player

**Interests: -**Players want to see different keybindings and general information on how to play the game

**-**System displays all of these on the screen

**Pre-condition:** - Player has to be in Main Menu

**Post-condition:-**

**Entry Condition:** Player clicks on “Display Instruction” in the main menu.

**Exit Condition:** Player clicks on “Return to Main Menu” in the instruction screen.

**Successful Scenario Event Flow:**

1. System renders all the instruction text on the screen.

**Alternative Flows:**

1. Player wishes to return to main menu
   1. Player clicks on “Return to Main Menu”
   2. System renders the main menu screen.
2. **Open Bestiary**

**Primary Actor:** Player

**Interests:**

**-**Player wants to get information about enemy types in the game

**-**System displays Bestiary panel on the screen

**Pre-condition:** Player has to be in Main Menu

**Post-condition:**

**Entry Condition:** Player clicks on “Open Bestiary” in the main menu

**Exit Condition:** Player clicks on “Return to the Main Menu” in the Bestiary screen

**Successful Scenario Event Flow:**

1. Player clicks on “Open Bestiary” in the main menu

2. System shows enemy types on the screen

**Alternative Flows:**

1. Player wishes to return to main menu
   1. Player clicks on “Return to Main Menu”
   2. System renders the main menu screen.
2. **Display Credits**

**Primary Actor:** Player

**Interests: -**Player wants to see information about developers of the game and external sources used in game development.

**Pre-condition:** - Player has to be in Main Menu

**Post-condition: -**

**Entry Condition:** Player clicks on “Display Credits” in the main menu.

**Exit Condition:** Player clicks on “Return to Main Menu” in the credits screen.

**Successful Scenario Event Flow:**

1. System displays all the credits text on the screen.

**Alternative Flows:**

1. Player wishes to return to main menu
   1. Player clicks on “Return to Main Menu”
   2. System renders the main menu screen.
2. **Pause Game**

**Primary Actor:** Player

**Interests: -**The Player wants to pause the game either to 1. Quit the game or 2. Pause it to deal with other issues or look at other things.

**Pre-condition: -**The player is in the game

**Post-condition: -**

**Entry Condition: -**The player presses the corresponding key that opens the Pause Menu

**Exit Condition: -**The player presses “Resume Game” OR “Quit Game”

**Successful Scenario Event Flow:**

1. The system shows the Pause menu.

**Alternative Flows:**

1A. Player wishes to return to main menu

1. Player clicks on “Quit Game”
2. System renders the main menu screen.

1B. Player wishes to continue the game

* 1. Player clicks on “Resume Game”
  2. System resumes the main game loop.

1. **Play Game**

**Primary Actor:** Player

**Interests:-**The Player aims to beat the game by defeating the final boss.

**Pre-condition:-**The system will use default settings unless the player has changed the settings beforehand.

- Player has to be in Main Menu

**Post-condition:-**

**Entry Condition:** Player clicks on “Start Game” in the main menu.

**Exit Condition: -**Player clicks on “Quit Game” in the pause menu.

-Player defeats the final boss

-Player loses all life

**Successful Scenario Event Flow:**

1. Game is initialized by the system.
2. Player starts from the first room.
3. Player explores/combats/solves puzzles within the room.
4. Player finds the key.
5. System unlocks the door allowing player to access new room.

*Repeat step 3 to 5 until player slays the final boss or loses all health*

1. System displays congratulation screen and returns to main menu.

**Alternative Flows:**

3A. Player wants to pause the game.

1. Player opens pause menu
   1. Player clicks on “Quit Game” to return to main menu
   2. Player clicks on “Resume Game” to exit from pause menu and continue the game.

3B. Player picks up an item

* 1. If Player picked up a power-up, boost the player’s corresponding stat according to the power up temporarily.
  2. If Player picked up an upgrade, boost the player’s corresponding stat permanently according to the upgrade picked up.

3C. Player collides with the Enemy or any kind of damaging element.

a. The System recalculates the Player’s health

a. If the Player has 0 life after recalculation

i. The system displays “You have died” and sends the user back to the main menu.

b. If the Player has more than 0 life after recalculation

i. Continue the game.

3D. Player completes a puzzle or kills all enemies

a. The system spawns the key at the predetermined spot.

3E. Player defeats the final boss

a. Move to successful event flow no. 6

5A. Player moves to previous room

1. System respawns all enemies in the previous room.

### Object and Class Model

Table 1 is an example table.

Table 1 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.