**Bilkent University**

Department of Computer Engineering

**CS 319 Project**

*JCrawl: 2D Top-down Adventure Game*

Design Report

Group Members

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

Contents

[1. Introduction 3](#_Toc445236848)

[1.1 Overview 3](#_Toc445236849)

[1.2 Design Goals 3](#_Toc445236850)

# 1. Introduction

## 1.1 Overview

JCrawl’s system primary goal is of course, entertainment. In order to achieve this, the system must have well-designed, streamlined gameplay, user friendly interface, along with little to no bugs. One of the key feature of JCrawl is that the levels will be customizable by the user. In order to achieve this, the game will be packaged with simple, easy to learn documentation to help users learn how to do so.

## 1.2 Design Goals

* **Performance**: The nature of the game that is going to be built demands high performance and low latency rates between each keystrokes. Optimizing data structures and misc. code will be the key to decrease the time taken for each update to take place.
* **Robustness**: There is nothing more frustrating than a game-breaking bug from end user’s perspective. The system should be prepared to handle myriad of unexpected/unwanted inputs from the user without causing the game to crash.
* **Extendibility**: Since the system will be designed with an object oriented language, five main principles will be kept in mind while designing the system: S. O. L. I. D
  + **1**. **S**ingle Responsibility Principle – All classes should have single responsibility (A class should not have multiple responsibility)
  + **2**. **O**pen Closed Principle – All classes should be open for extension, but avoid unnecessary edit/revision (extend existing methods instead of writing new ones or editing existing methods)
  + **3**. **L**iskov Substitution Principle – Child Class and Parent Classes should be interchangeable
  + **4**. **I**nterface Segregation Principle – An interface should not have redundant methods that are never used.
  + **5**. **D**ependency Inversion Principle – Parent Class should not be dependent of Child Class
* **Usability:** The system will be user-friendly, easy to use. For example, due to the relatively high learning curve of learning custom level design, default level layout will be provided when the system is delivered. However, documentation will be provided to aid in user’s pursuit to learn it as easily as possible.

**Trade Offs**

* **Size vs Performance** – Since our system will be more focused on using the best algorithms to give us the lowest time complexity (big O), memory space usage optimizations will be at lower priority. However, there will be efforts to avoid pointless memory waste.
* **Reusability vs Performance** – In order to squeeze out the best possible performance, sacrifices must be made from reusability portion of the design. Also, if the issue of reusability can be outside of the scope of our design, it will allow for certain methods that highly specializes our system to be used so that performance can be at its best level.
* **Functionality vs Robustness** – Having a lot of gameplay content is appealing for the user, however, more toys usually means more things can break. There will be several restrictions on expanding new game content for the sake of system security and robustness.