**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 3](#_Toc444634553)

[2 Proposed System 3](#_Toc444634554)

[2.1 Overview 3](#_Toc444634555)

[2.2 Gameplay Elements 3](#_Toc444634556)

Analysis Report

JCrawl: 2D Top-down Adventure Game

# 1 Introduction

JCrawl is a purely Java based top down adventure game, the basic game mechanics/fundamentals of the game such as room exploration, movement, and combat will be inspired from the critically acclaimed game called Binding of Isaac, but with some design ideas borrowed from The Legend of Zelda, such as the UI design, art style and misc. non-functional related aspects.

**Binding of Isaac link:**

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

**Legend of Zelda link:**

<https://en.wikipedia.org/wiki/The_Legend_of_Zelda_(video_game)> (as the game was released in 1996, the link will be connected to a Wikipedia page about it)

The primary focus of this game will be heavily on Player-to-Object and more specially, Object-to-Object interactions like pressing a switch to turn on the lighting, killing enemies by shooting projectiles at them, moving blocks on a top of a pressure plate to keep the traps off and such.

# 2 Proposed System

## 2.1 Overview

The player controls a single character 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, the documentation containing required instruction on how to do so will also be shipped with the final product, and the only tool required to perform level editing will be any kind of text editor like Notepad or Word.

## 2.2 Gameplay Elements

1. **Player**

The player controlled character will possess variety of stats that the user can upgrade and utilize throughout the course of the game: Health, Movement Speed, Attack Speed, and Damage; Picking up certain upgrades will either permanently boost its stats. For example, picking up an item named “Rapid-fire Module” will permanently alter player’s interval between each attack so that it is more rapid.

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

1. **Enemies**

Enemies will have three different types of behavior set to them: Rudimentary AI, Hard coded behavior, 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, the player will be able to take it down with a single hit.

Behavior Type: AI

2. Wolf: Slightly stronger counterpart of Slime, it will move faster and deal more damage when coming in contact with the player, but it will still go down with single hit from the player’s attack.

Behavior Type: AI

3. Orc: This guy hits harder and he is harder to take down then the Wolf, but as a tradeoff, he moves much slower and has longer interval between movements. Takes around 5 hits to kill.

Behavior Type: AI

4. Goblin: This enemy is just like slime but it moves constantly in a destined patrol route. Only takes one hit to kill.

Behavior Type: Hard coded

5. Skeleton Archer: Stationary enemy that shoots arrows from a distance, only takes one hit to take down.

Behavior Type: Stationary

6. Magmatrum: A stronger, slower version of the slime, the special aspect of this enemy is that it leaves special terrain called “Scorched Earth”, which will damage the player if the player touches it. Takes 5 hits to take down.

7. Final Boss (Placeholder name): This enemy will move fast, and unlike all other enemies (except stationary types), will actively track the player and attack him from distance with projectiles just like the player. Takes 20+ hits to take down.

**Disclaimer:** Any specific numbers associated with the enemy’s stat is **not** final and will always be subject to change in the Testing phase.

1. **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: A hybrid between Dart Trap and Spike Trap, while the range isn’t melee, but it doesn’t exactly reach from the other end of the room to the other end as well. The range of this trap will be around 3~5 Tiles.

4. Automatic Doors: A door that closes and opens with set interval.

5. Monster Spawner: Spawns AI controlled enemies with long intervals between the spawn.

1. **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. Locked Door: If the player has a key, it will open, giving access to new room.

2. Light Switch: Turns on/off lights 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, or the player can move a block so that the pressure plate stays pressed without having the play standing on it.

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.

7. Door: Unlike Locked Doors, these can be opened by triggering mechanisms associated with this door to grant access to other parts of the current room.

1. **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. Purely Cosmetic.

6. Scorched Earth: Behaves exactly the same as lava, but the difference is that it will disappear after certain time.

7. Void: Another version of impassable terrain.

1. **Power-ups**

Power ups offer temporary boost to aid in player’s adventure during the play session by altering/modifying player’s stats in a positive way.

1. Restore Health: Restores the health of the player by 1.

2. Burst of Speed (Movement Speed): Increases movement speed by certain amount for certain duration.

3. Invincibility: Player becomes immune to any source of damage for certain duration.

4. Burst of Speed (Attack Speed): Increases attack speed by certain amount for certain duration.

5. Firepower Boost: Increases damage by certain amount for certain duration.

1. **Upgrades**

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

1. Rapid-fire Module: Permanently boosts the player’s shooting speed by certain amount.

2. Quad-direction Bolt Splitter Mk IV: Instead of firing towards the direction that the character is looking at only, player can now fire towards all directions at once, regardless of where the player is looking at.

3. Boots of Speed: Permanently increases movement speed by certain amount.

4. Assault Cuirass: Increases maximum health by 1.

5. Ring of Vitality: Passively regenerates lost life

6. Salamander’s Shroud: Grants immunity to fire based damage (Lava/Scorched Earth/Contact with Magmatrum)

7. Burst Fire Module Mk II: Shoots two bolts instead of one

8. Burst Fire Module Mk III: Shoots three bolts instead of one

9. Gauntlet of Ogre Strength: Allows players to push blocks.

1. **Miscelleneous**

Other objects which do not belong to any specific category will belong here, in this case: the Key.

1. Key: The key will spawn when the players satisfy specific conditions which depends from room to room at predetermined spot. If Player picks up the key, the Locked Doors will open.

## 2.3 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). Combat involves destroying all enemies in a room which will in turn spawn the key at the center of the room (if applicable), puzzle will involve interacting with certain dependent object which will trigger the spawning of the Key at certain spot.

**2. Change Options**

The user can change two options:

-Texture quality

-Screen size

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

**3. Display Credits**

Simple display of credits for any external sources we might use and list of our names/IDs will be displayed on the screen.

**4. 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. The system will list them out on the screen.

**5. Open Bestiary**

There will be many different types of enemies and their behavior will be completely unknown for the first time players. So, the Bestiary will contain information about all enemies the players will encounter in the game. They will be listed on the screen by the system.

**6. Pause Game**

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

## 2.5 Non-functional Requirements

• Game Performance

The game will require to have low latency (100 ms) 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 must 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 and allows easy reusal/expanding content for the future.

## 2.6 Pseudo Requirements

1. The code will be written in Java only

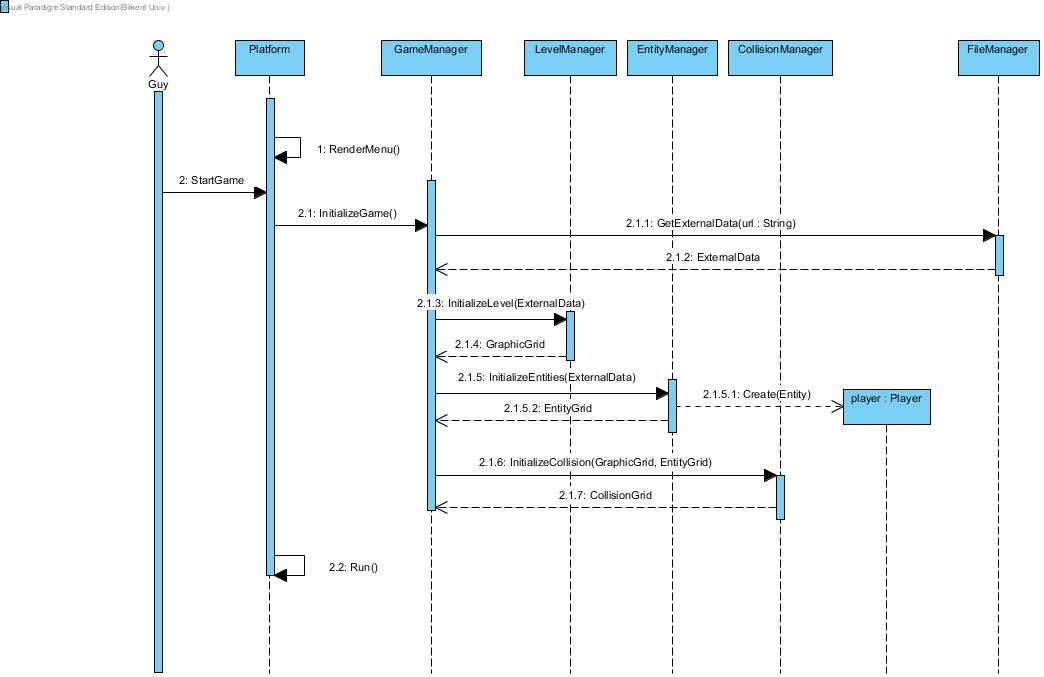
2. Desktop only

3. Level editing should be easy to learn.

## 2.7 System Models

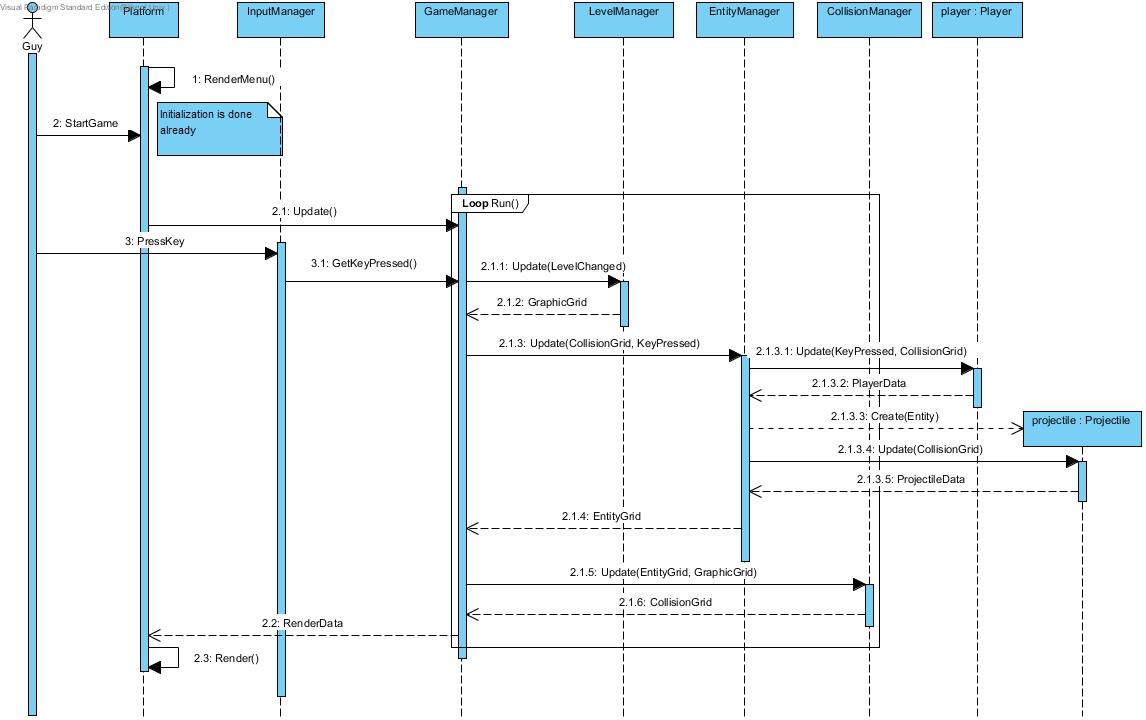
* Use Case Model
* Dynamic Models

1. **Scenario 1:** After Chris started the game, the system initialize graphics and objects according to the file of Chris. He forgot to place any other entities except for the player, so only the player was rendered and there was no other entities he could interact with.

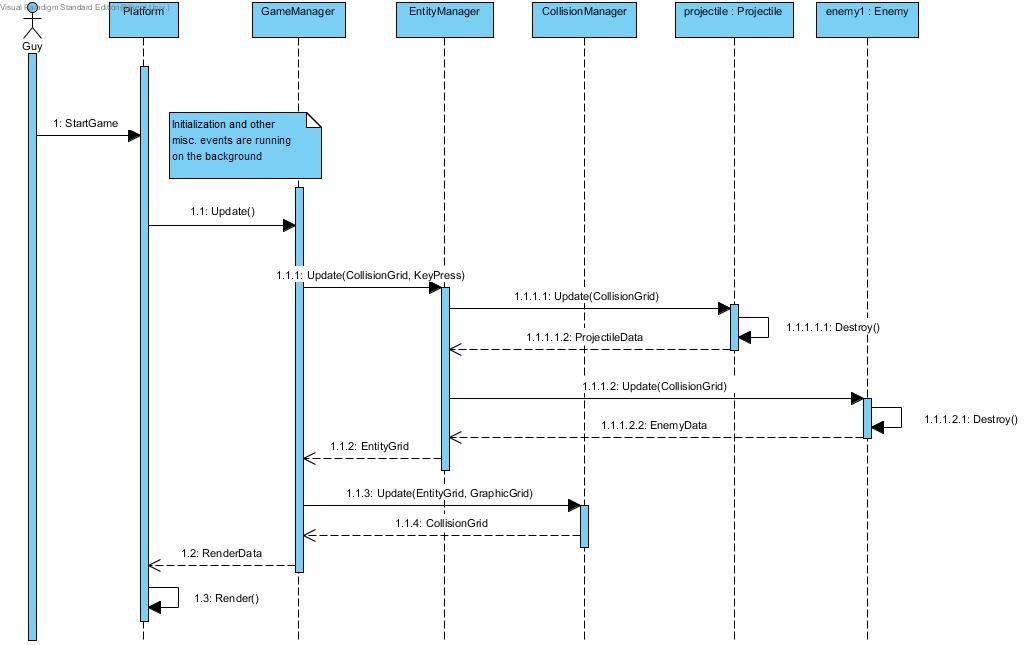


Here, the Platform renders the menu of the game using RenderMenu() method, then the player presses on the “start game” which causes the Platform to call the InitializeGame() method, which in turn calls the related initialization methods of all relevant managers (InitializeLevel(), InitializeEntities()…) All these initializations return their corresponding data which in turn will be used by the Platform later on to render the screen (which is not shown here yet).

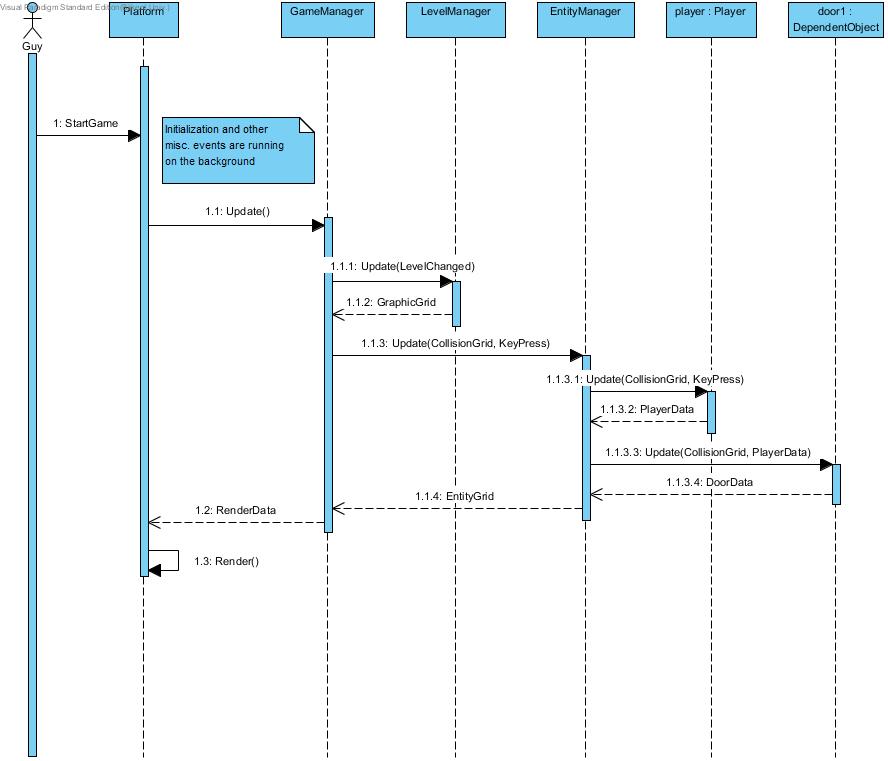
1. **Scenario 2:** Jennifer started the game, the system loaded first level with blocks, independent and dependent object and enemies, she decided to test some of the functionalities of the game before moving onward to the next room, she fired some projectiles at the wall.



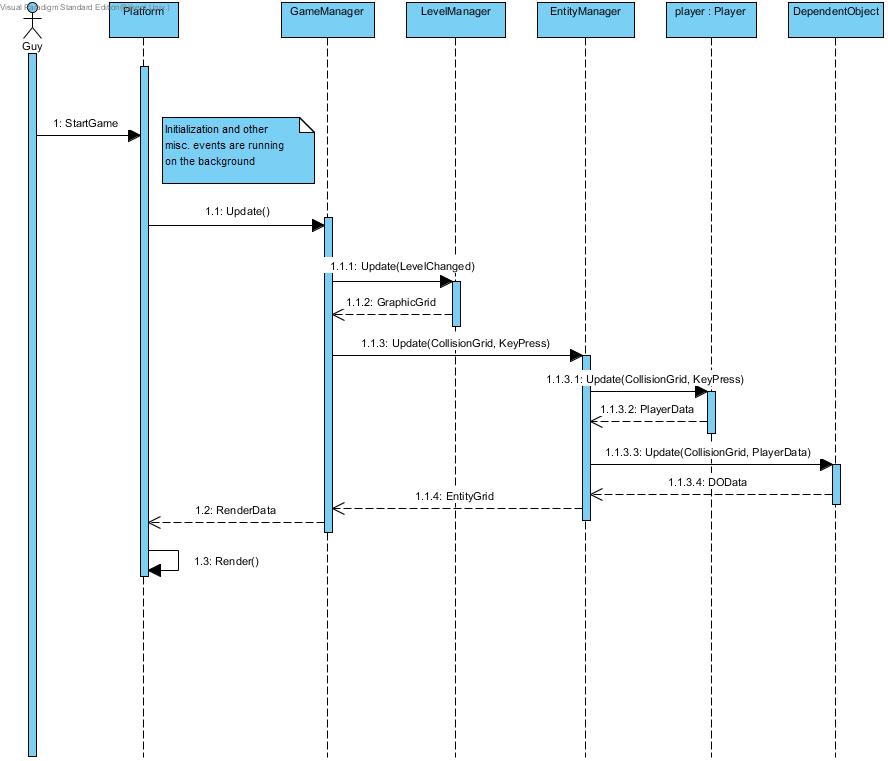
1. **Scenario 3:** Ali performed a test in the custom level. He created a Slime, he shot some projectiles into the Slime and he destroyed Slime.



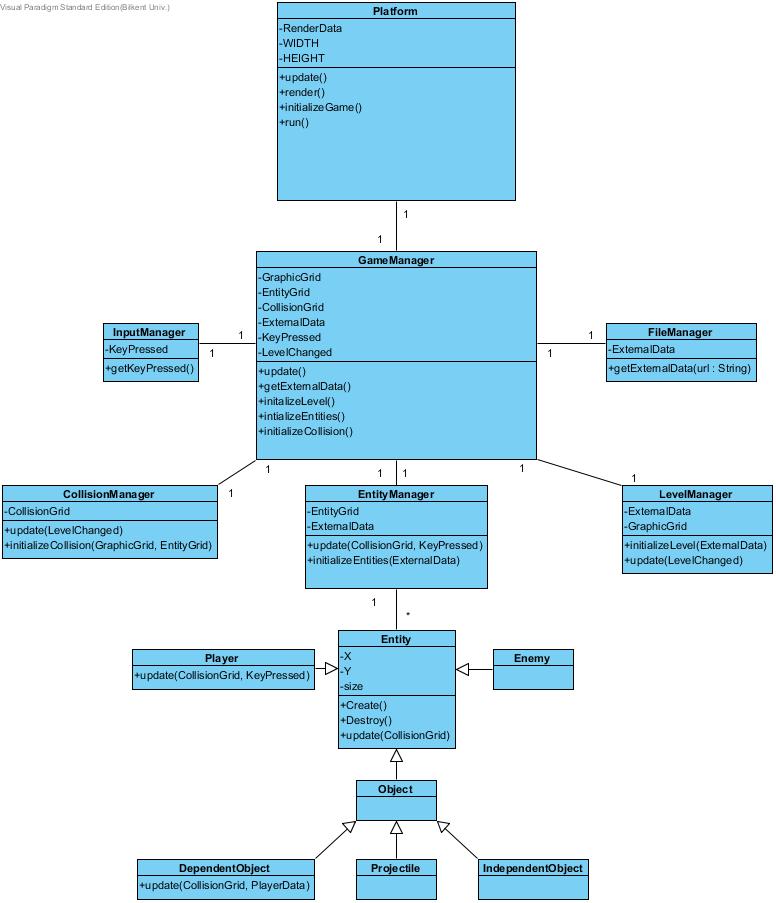
1. **Scenario 4:** Hande already took care of multiple enemies which was roaming the room, and she already picked up the Key from the center of the room. She then proceeded to the Locked Door and because she had the key, the door opened and she was able to proceed to the next room.



1. **Scenario 5:** Hande found a switch next to her character right after she moved to the next room. Triggered, she decided to interact with it and apparently it actually didn’t do anything.



* Object and Class Models



# 3 Glossary

# 4 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.