VIETNAM NATIONAL UNIVERSITY OF HO CHI MINH CITY  
INTERNATIONAL UNIVERSITY  
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Logo

Description automatically generated

**FINAL REPORT**

**Artifact Project**

***Course:*** *OBJECT-ORIENTED-PROGRAMMING- IT079IU*

ADVISOR:

*Assoc. Prof. Nguyễn Thanh Tùng*

**ABSTRACT**

Artifact game is a 2D platformer game that blends contemporary gameplay principles with aspects of vintage side-scrolling games. Players take control of a brave character in this colorful and lively game as they try to stop an evil sorcerer from destroying their community.   
  
There are several stages in the game, and each has its own special difficulties and barriers. To advance through these stages, players must acquire power-ups, vanquish opponents, and solve riddles. Because to the responsive and easy-to-use controls, players of all skill levels may enjoy this game.

The dynamic difficulty system of Artifact game is one of its main characteristics. The game offers a difficult yet equitable experience by adjusting to the player's skill level. The game's richness and replayability are further increased by the assortment of mini-games and side missions it offers.

To sum up Artifact game is an entertaining and captivating 2D platformer game that provides a distinctive fusion of traditional and contemporary gameplay components. Adventure Quest is likely to keep gamers of all ages entertained for hours on end with its simple controls, varied difficulty structure, and endearing artwork

# Group 6 Project’s Member

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Full Name** | **ID** | **Role** | **Contribution** |
| 1 | Hồ Ngọc An | ITITIU21146 | Leader | 40% |
| 2 | Phạm Minh Nguyên | ITITUN21033 | Member | 30% |
| 3 | Trần Viết Trung | ITITWE22116 | Member | 30% |

## *Table 1: The contribution and information of all project members*

## **TABLE OF CONTENTS**

[**Group 6 Project’s Member** 3](#_heading=h.gjdgxs)

[**Chapter 1. Introduction:** 5](#_heading=h.u3xpfzt73ez5)

[1.1. Plan 5](#_heading=h.1fob9te)

[1.2. Requirements 6](#_heading=h.3znysh7)

[**Chapter 2. Story:** 6](#_heading=h.6ot98nkuqutg)

[**[Chapter 3. Tutorial:](#_heading=h.26in1rg)** [7](#_heading=h.26in1rg)](#_heading=h.2s8eyo1)

[3.1 How to play: 7](#_heading=h.26in1rg)

[3.2 Goals 9](#_heading=h.35nkun2)

[**Chapter 4. Design and New features: 10**](#_heading=h.2s8eyo1)

[4.1 Design: 10](#_heading=h.35nkun2)

[4.2 New features 9](#_heading=h.35nkun2)

[4.3 Summary: 9](#_heading=h.35nkun2)

[**Chapter 5. 4 properties of OOP** 10](#_heading=h.2s8eyo1)

[5.1 Encapsulation 10](#_heading=h.35nkun2)

[5.2 Abstraction: 10](#_heading=h.35nkun2)

[5.3Inheritance 10](#_heading=h.35nkun2)

[5.4 polymorphism 10](#_heading=h.35nkun2)

[**Chapter 6. UML and Diagram** 11](#_heading=h.2s8eyo1)

[6.1 Audio: 11](#_heading=h.2jxsxqh)

[6.2 Enemy: 12](#_heading=h.2jxsxqh)

[6.3 Handlers 13](#_heading=h.2jxsxqh)

[6.4 Main: 14](#_heading=h.2jxsxqh)

[6.5 Entity: 15](#_heading=h.2jxsxqh)

[6.5.1. Artifact: 15](#_heading=h.2jxsxqh)

[6.5.2. Enemies: 15](#_heading=h.z337ya)

[6.6. GameStates: 19](#_heading=h.stryk4i9ylz9)

[**Chapter 7. Conclusion: 20**](#_heading=h.111kx3o)

[5.1. Future Work 20](#_heading=h.3l18frh)

[5.1.1 Implementing more User Interface (UI) 20](#_heading=h.206ipza)

[5.1.2 Complete the security system. 20](#_heading=h.4k668n3)

[5.1.3 Packaging 20](#_heading=h.2zbgiuw)

[5.1.4 Deployment 21](#_heading=h.1egqt2p)

[5.2 Summary 21](#_heading=h.3ygebqi)

[**Chapter 8. GITHUB: 20**](#_heading=h.111kx3o)

**LIST OF FIGURES**

[Figure 3.1 Move 7](#_heading=h.fxskkxt5u1me)

[Figure 3.2 Attack 7](#_heading=h.fxskkxt5u1me)

[Figure 3.3 Dash 8](#_heading=h.fxskkxt5u1me)

[Figure 3.4 Enemies and Boss 8](#_heading=h.fxskkxt5u1me)

[Figure 6.1: Audio 11](#_heading=h.fxskkxt5u1me)

[Figure 6.2: Enemies 12](#_heading=h.h1xt1lvf52sm)

[Figure 6.3:Handlers 13](#_heading=h.mij89goqpsic)

[Figure 6.4: Main Class 14](#_heading=h.iisfbdoci3wo)

[Figure 6.5.1: Artifact 15](#_heading=h.1pxezwc)

[Figure 6.5.2: Enemies 18](#_heading=h.49x2ik5)

[Figure 6.6:GameStates 19](#_heading=h.8itg0an9h27d)

**LIST OF TABLES**

[Table 1: The contribution and information of all project members 2](#_heading=h.adnxzzbdmefu)

[Table 1.1: Table of timeline 5](#_heading=h.wx0lmiw0n8o1)

# Introduction

### 1.1. Plan

An overview of an artifact project is what this paper aims to convey. Stakeholders engaged in its creation, implementation, and maintenance are the target audience for this paper, which includes design, implementation, and assessment. For anyone who wants to know more about Artifact Project, the report is a useful reference.

## **Table 1.1: Table of timeline**

|  |  |
| --- | --- |
| **Week** | **Task** |
| 1 - 4 | Learn Basics of Javascript-ES6 and decide a type of game, Gameplay Mechanics, Art Style. |
| 5 - 8 | Design characters, level and use interface |
| 9 - 12 | * + - Implement the gameplay mechanics, character movements, collision detection, etc.   + - Create or source the art assets for your game, including characters, backgrounds, and UI elements. |
| 13 (Deadline) | Play the game ourself and make adjustments based on our experience. Have others play your game and provide feedback |

### 

### 1.2. Requirements

The process of making a 2D game is thrilling and starts with a well-defined concept and design that specify the genre, plot, characters, and aesthetic.

A thorough grasp of a programming language appropriate for game creation, such as Python and Pygame, JavaScript and Phaser, or C# and Unity, is necessary for this procedure. By offering necessary tools and frameworks, game engines such as Godot, Unity, or GameMaker may greatly expedite the creation process.

Characters, locations, and in-game items are all rendered in two dimensions utilizing graphic design tools, which adds to the game's visual attractiveness. Sound effects and music are added to further improve the immersive game experience.

Thorough testing is essential to guarantee seamless gaming and to find and fix any faults. When a game is flawless, it's prepared for release on an appropriate platform. This may be a web-based platform for browser games, a PC platform like Steam, or a mobile app store. Making a 2D game only takes time, patience, and a will to learn, but it can be a really fulfilling process despite the difficulties.

# Story

***Stage 1:*** Once upon a time, in a distant kingdom, there existed an evil sorcerer who sought to conquer the world. Using dark magic, he spread suffering and fear among the people. However, there was a young man named Arthur, born with a courageous heart and an indomitable spirit. Arthur decided to embark on a quest to find the legendary sword, which the legends said was the only weapon capable of defeating the sorcerer.

***Stage 2:*** On his journey, Arthur encountered many companions: a fierce female warrior, a wise monk, and a small, flying dragon. Together, they overcame numerous challenges, from dense forests and treacherous mountains to monster-filled caves. When they finally reached the sorcerer's fortress, Arthur and his companions faced deadly traps and fierce minions.

***Stage 3:*** In the end, with bravery and the unity of his friends, Arthur confronted the sorcerer head-on. Using the legendary sword and the support of his companions, Arthur defeated the sorcerer, freeing the world from the clutches of evil. The people rejoiced, the kingdom returned to peace, and Arthur and his friends became heroes remembered forever.

# Tutorial

**3.1 How to play:**

* “Play” to start the game.
* “Help” for knowledge about how to play game.
* “Quit” to close the game.



**Figure 3.1: Menu**

* Hold E and the button 🡪,🡨 to run.



**Figure 3.2: Move**

* You can kill all enemies by F in a straight line when using dash

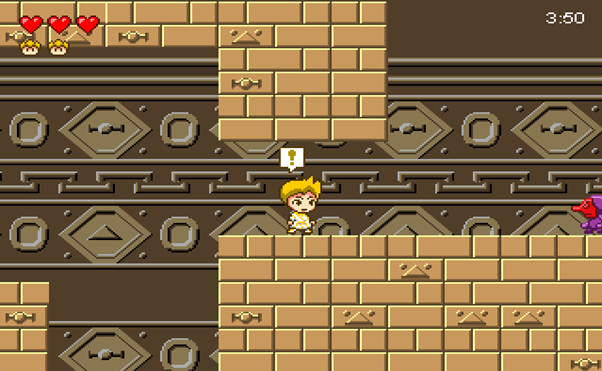
**Figure 3.2: Attack and defeat enemies**

* Fall into an abyss the game will start over

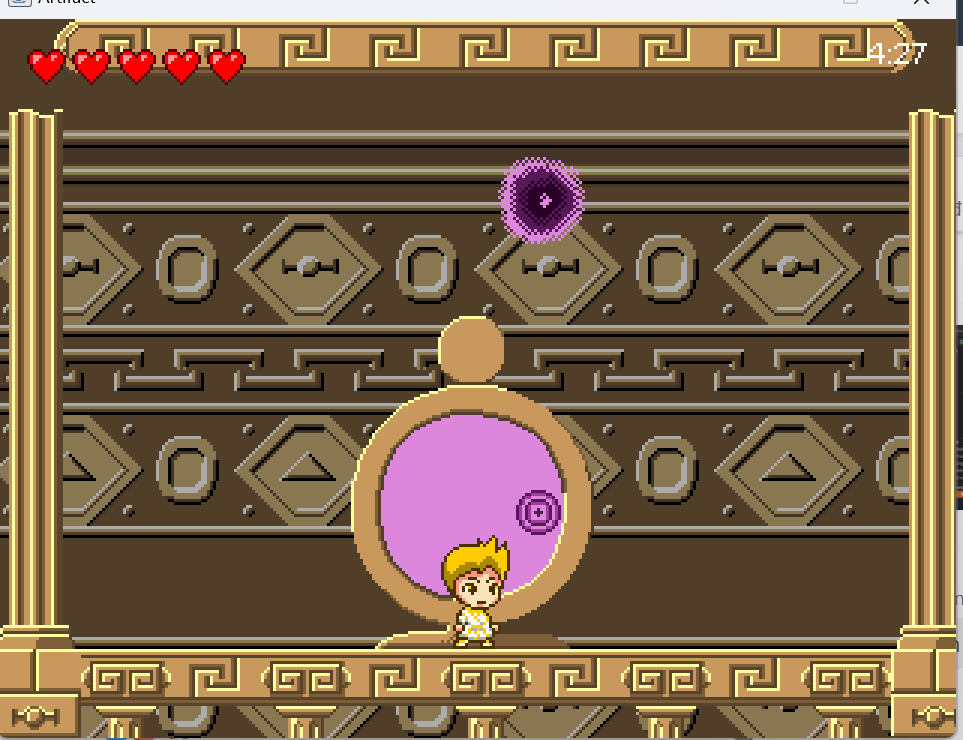


**Figure 3.3: Fall into**

**3.2 Goals:**

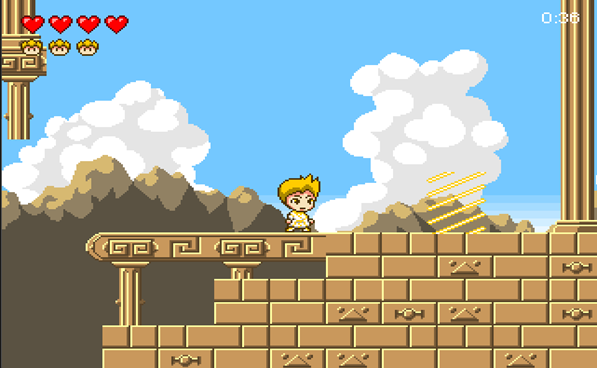
****

* Defeat enemies and boss



**Figure 3.4: Enemies and Boss**

* The place to move to the next stage.



**Figure 3.4: Place to move**

# Design & New features

**4.1 Design:**

* BufferedImage for image files and pixel color manipulation.
* Pixel colors for game components with RGB channels.
* Level class for level data and game elements.
* Google resources for optimization.

**4.2 New Features**

* Keyboard control for movement and attack.
* Slashing move until hitting tiles.
* Tutorial at the start.
* Bosses at the end.
* Jumping costs energy

**4.3 Summary:**

* State pattern for modes and transitions in gamestates package.
* Constants class for game constants.
* Single Responsibilities pattern applied.
* Design principle for readability and maintainability.

# 4 properties of OOP

* 1. **Encapsulation:**
* Encapsulation in our game project using classes like Player, Enemy, and Game.
* Attributes and methods are hidden.
* For example, the Player class has specific attributes.
* Getters and setters.

**5.2 Abstraction:**

* Create subclasses for code reuse and hierarchy.
* Enemy class.
* The GameState interface.
* Abstraction enhanced modularity and maintainability of shared features.

**5.3 Inheritance:**

* Create subclasses for code reuse and hierarchy.
* For example, MapObject and GameState
* Reducing code duplication.
* Promoting modular code

**5.4 polymorphism:**

* The EnemyManager class contains 2 checkEnemyHit methods.
* Depend on the type of collision.

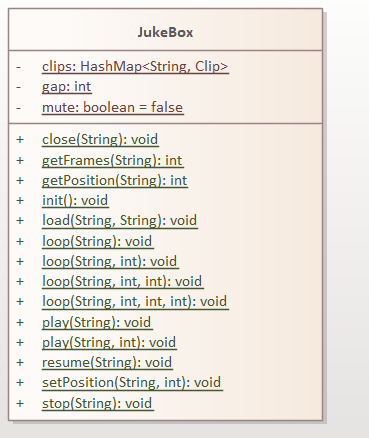
# UML and Diagram

**6.1 Audio:**

***JukeBox class*** is designed to handle audio functionalities such as playing, stopping, and looping sound clips. It includes various attributes and methods that manage these audio operations.

***JukeBox class*** is responsible for managing audio playback within the application. Its primary functions include:

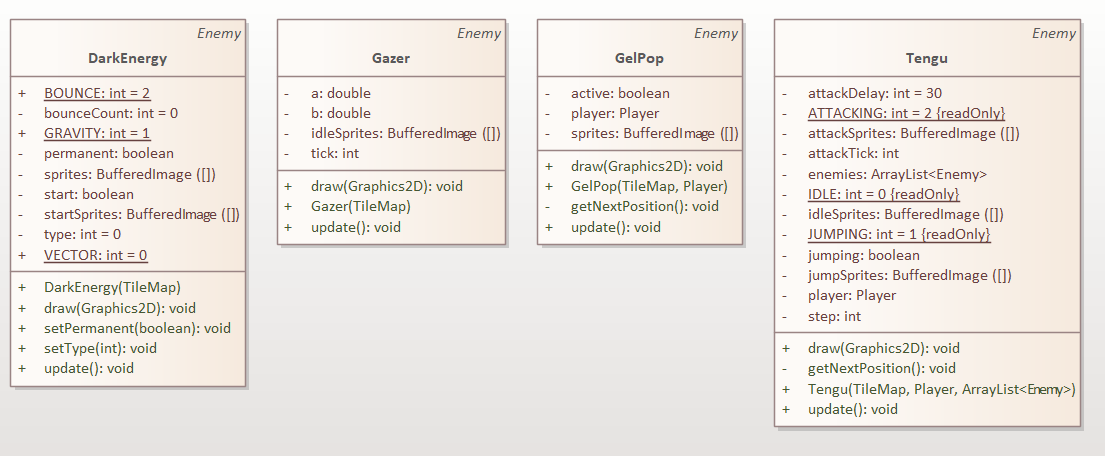
* ***Loading and Storing Audio Clips***: New clips are loaded into this collection using the load method, and all audio clips are stored in the clips property.
* ***Paying Control:*** The ability to start, stop, and resume the playing of audio clips is provided via controls like play, stop, and resume.
* ***Looping:*** With control over start and finish positions, the loop techniques offer many ways to loop audio clips for a predetermined number of times or forever.
* ***Position Management:*** You may precisely control where playback begins and ends by using methods like getPosition, setPosition, and getFrames to manage the playback position of audio clips.
* ***Muting:*** The audio playing can be muted or unmuted using the mute property.



**Figure 2.1: Audio**

**6.2 Enemy:**

* ***DarkEnergy Class:*** Depicts an energy-based adversary that may be either transient or permanent and travels in a bouncing pattern under the effect of gravity.
* ***Gazer Class:*** Describes an adversary with particular timing and sprite modifications that may employ beams or gaze assaults.
* ***GelPop Class:*** This type of opponent resembles a blob and is capable of interacting with the player and changing its state in response to actions.
* ***Tengu Class:*** Possibly modeled after a mythological monster, this class depicts a multifaceted adversary capable of both attacking and jumping.



**Figure 2.2: Enemies**

**6.3 Handlers:**

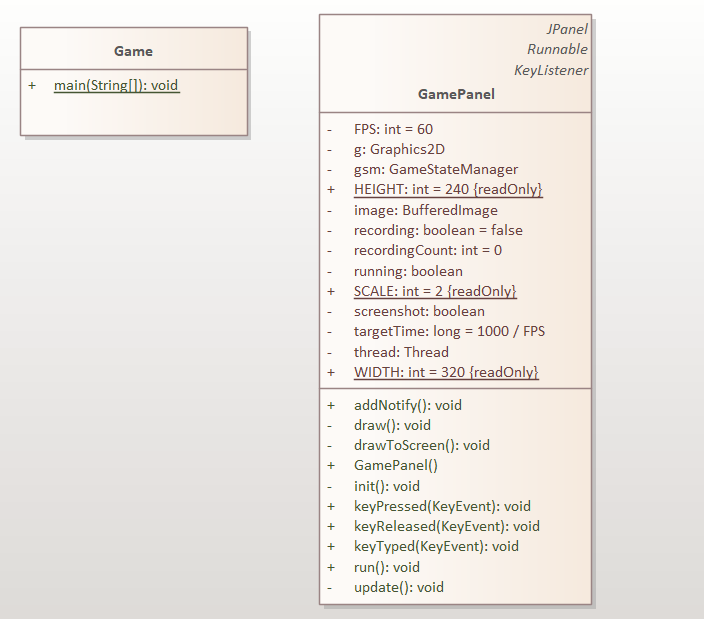
* ***Content Class:*** in charge of controlling the loading and storing of picture assets within the application.
* ***Keys Class:*** this class handles updating and preserving key input statuses so the program can respond to user keyboard and mouse inputs.

****

**Figure 2.3: Handlers**

**6.4 Main:**

* ***Game class***: serves as the starting point of the application with its main method.
* ***GamePanel class***: is the core component that handles the game loop, rendering, and input. It contains various attributes for managing the game's state and methods for initializing, updating, drawing, and handling keyboard events.

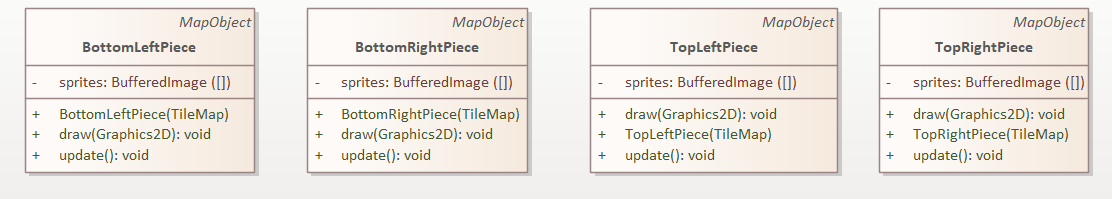


**Figure 2.4: Main Class**

**6.5 Entity:**

**6.5.1 Artifact:**

* ***BottomLeftPiece:*** This symbol designates a section of the map situated in the lower-left corner. This particular section of the map is rendered and updated by this class.
* ***BottomRightPiece:*** This element of the map denotes the area at the bottom-right corner. This particular section of the map is rendered and updated by this class.
* ***TopLeftPiece:*** This symbol designates a section of the map situated in the upper-left corner. This particular section of the map is rendered and updated by this class.
* ***TopRightPiece:*** This symbol designates a section of the map situated in the upper-right corner. This particular section of the map is rendered and updated by this class.

**

**Figure 2.2.1: Artifact**

**6.5.2 Enemies:**

1. **Player Class:**

***Function:*** Acts as a representative of the player character and controls its actions, interactions, and status in the game environment.   
***Main Duties:***

* Manages player activities including dashing, jumping, and attacking.
* Controls player states, such as health, score, and other states (such as dead or flinching).
* Refreshes the player's location, interacts with other objects, and looks for collisions.

1. **Enemy Class:**

***Function:*** Acts as a representative for antagonistic figures in the game, controlling their activities and interactions with the player.   
 ***Main Duties:***

* Controls adversary conditions, including health, location, and assault patterns.
* Manages player interactions, updates opponent position, and looks for collisions.
* Regulates enemy-specific behaviors and conditions (such as explosions upon death).

1. **MapObject Class:**

***Function:*** Acts as the foundational class for all objects on the game map that communicate with their surroundings.  
***Main Duties:***

* Offers fundamental characteristics for location, size, and motion.
* Controls the game map's collision detection and reaction.
* Describes changes in state and movement behaviors (e.g., falling, leaping).

1. **Tile Class:**

***Function:*** Stands in for individual tiles that comprise the game map.  
***Main Duties:***

* Keeps track of tile attributes including type and blocking status.
* Provides tools for managing the attributes of tiles and drawing them.

1. **HUD Class:**

***Function:*** Organizes and maintains the Heads-Up Display (HUD), which presents player-specific data like score and health.  
***Main Duties:***

* Sketches HUD components into the screen.
* Adjusts the display according to game events and player status.

1. **EnergyParticle Class:**

***Function:*** Describes particles produced by certain game events or activities (e.g., assaults' consequences).  
***Principal Duties:***

* Controls the motion and location of particles.
* Updates the status of particles and draws them.
* Establishes the appropriate time for particle removal.

1. **Explosion Class:**

***Function:*** Acts as the game's representation of explosive effects.  
***Main Duties:***

* Controls animation frames and explosion position.
* Creates and updates the effect of an explosion.
* Manages an explosion's lifespan, such as by eliminating it after it's finished.

1. **EnemyProjectile Class:**

***Function:*** Stands in for enemy projectiles.  
***Main Duties:***

* Controls projectile location, motion, and condition.
* Responds to impacts with other things, such as players.
* Draws the projectile and updates it.

1. **Portal Class:**

***Function:*** Stands in for portals that the player may utilize to get between the game's many sections.

***Main Duties:***

* Controls the portal's status (open, closed, etc.).
* Creates and modifies the animations for portals.
* Manages the player's portal interactions.

1. **FireBall Class:**

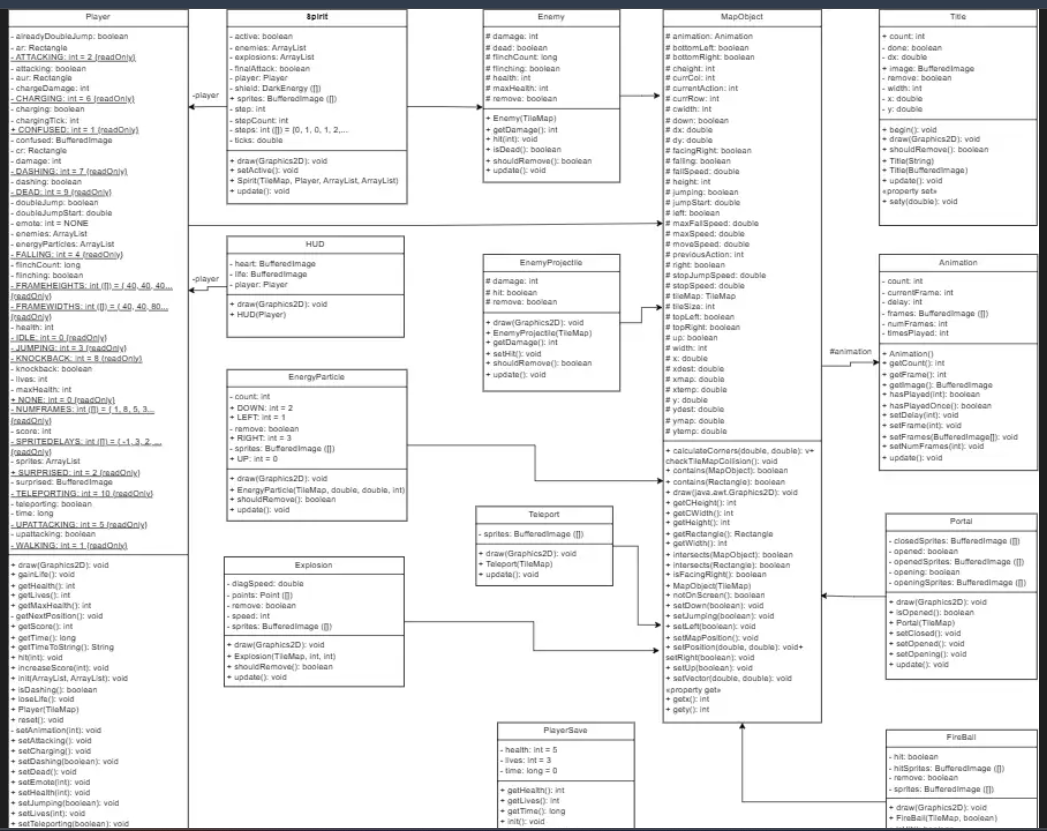
***Function:*** Stands for fireball projectiles that the player or adversaries can utilize.  
***Main Duties:***

* Controls the location, motion, and collision detection of fireballs.
* Draws the fireball and refreshes it.
* Manages how fireballs behave when they hit (e.g., delivering damage, removing after contact).

1. **PlayerSave Class:**

***Function:*** organizes the loading and storing of player status information.  
***Main Duties:***

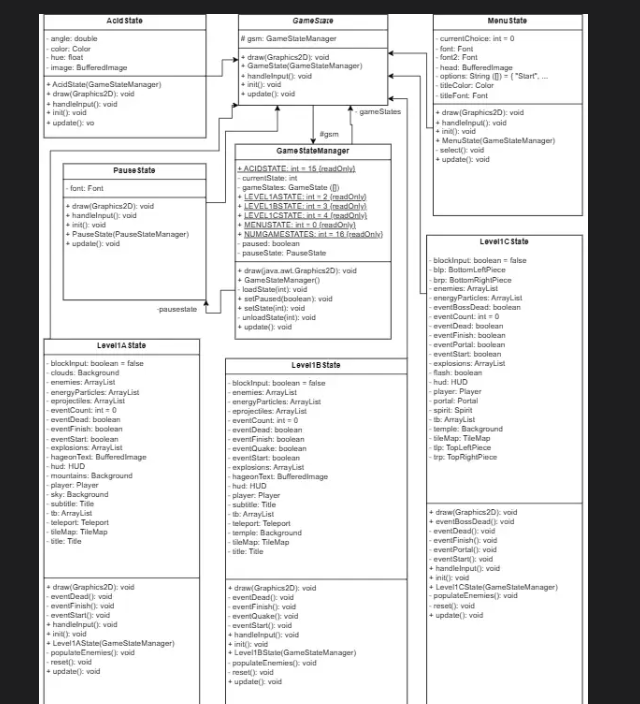
* Keeps track of player information, such as score, lives, and health.
* Allows for the saving and loading of this data, enabling game persistence.



**Figure 2.5.2 Enemies:**

**6.6 GameState:**

1. ***AcidState:*** Manages the state where the game includes acid-based mechanics or visuals.
2. ***PauseState:*** Manages the game state when the game is paused.
3. **GameStateManager:**Manages all game states, handling transitions and updates between them.
4. ***MenuState:*** Manages the main menu of the game.
5. ***Level1State:*** Manages the first level of the game.
6. ***Level2State:*** Manages the second level of the game
7. ***Level3State:*** Manages the third level of the game

****

**Figure 2.6 GameState:**

# Chapter 7. Conclusion

### 5.1. Future Work:

Our current application still has vital problems that the workers are hoping to address in future implementation. Here are some of them:

### 5.1.1 Implementing more User Interface (UI):

The user interface (UI) is a bit off since we had not got a user setting site, adding ticket, commenting page to function properly. We modified the login and sign up forms to store the user email in the database, had we not had abundant time to do everything we originally planned. Our team wanted to use ReactJS **(5)** Framework to make features like forms, buttons, and menus for easier user interaction.

### 5.1.2 Complete the security system:

The existing application is susceptible to security vulnerabilities due to the absence of decentralized security measures. This means that a customer's token, once acquired, grants unrestricted access and modification privileges to all data in the database, posing a significant risk if the token falls into the wrong hands.

In simple terms, unauthorized possession of a user's token could enable malicious manipulation of the database, resulting in potential property damage and complications. This flaw may lead to the exploitation of crucial information, precipitating serious legal and financial consequences. To address these security concerns, upcoming versions of the database will implement a stringent user access control policy, mitigating the risk of unauthorized access and manipulation.

### 5.1.3 Packaging:

The current project faces compatibility challenges across platforms, requiring users to manually set up and install necessary dependencies. This process is error-prone and time-consuming, leading to complications and unreliability during the setup. Running the application in diverse settings becomes challenging due to these issues. To streamline this process, we aim to leverage Postman**(3)** to bundle our application.

By utilizing Postman(3), we intend to simplify the setup process and enhance the application's compatibility across various platforms. This approach aims to provide users with a more user-friendly and consistent experience, reducing the likelihood of errors and saving time during the installation phase. Postman's capabilities will contribute to a more seamless and reliable deployment of the application across different environments.

### 

### 5.1.4 Deployment:

Currently, the application is limited to running on localhost, restricting users from accessing it from different locations. To address this limitation and enhance accessibility, there are plans to deploy the application on Amazon Web Services (AWS). By leveraging AWS, we aim to make the application accessible over the internet, providing users with the flexibility to use it from various locations. This deployment strategy not only expands the reach of the application but also contributes to its scalability and availability by utilizing cloud infrastructure and services provided by AWS.

### 5.2 Summary:

To conclude, despite some shortcomings, we have achieved the main goal of the project. We have followed all the use cases in our Use Case Diagram and created a Database that meets the Third Normal Form. If you want to learn how to use our application for your own project, please contact us!

# Chapter 8. GITHUB

**This is out github code of progress in developing product:**

<https://github.com/KyungUwU/Artifact-II>

**REFERENCES:**

[https://longbaonguyen.github.io/courses/platformer/platformer.html](https://longbaonguyen.github.io/courses/platformer/platformer.html?fbclid=IwZXh0bgNhZW0CMTAAAR05t0EtDJpe2od5sPw5sLk2GMxbcD0ycbRuJPctY-Uei95JPMHcXUnTZ-c_aem_AfYUpwBb3ePUevOf4TA-1OhNev6ehuVu7mh-UYr29gRm30Tdfz8cwlMNiwihyjc3rBE4qK63nfH9DRUToK-mJoDB)

[https://github.com/topics/platformer?l=java](https://github.com/topics/platformer?l=java&fbclid=IwZXh0bgNhZW0CMTAAAR1X7F0tNnDvknHsgIcZbbfhA4V1PiOnU_wQriA80iEfXBdcHpZF9YKqd6E_aem_Afa_z1BrYnEGbhOYV2fBV-_n7sg7Pk4izO1zWUDwfiC9pjbBAP5-65KyG8Ae14wxha-UTc7U9a7Wyo1F3icpw-h0)

<https://github.com/topics/2d-platformer-game?l=java>

[https://www.youtube.com/watch?v=9dzhgsVaiSo](https://www.youtube.com/watch?v=9dzhgsVaiSo&fbclid=IwZXh0bgNhZW0CMTAAAR1Dhfal462yFbCMK9ygFRi8z3e81PEaWa3HQLatJ9QNEBBlJIqsRK_BADo_aem_Afbz-S3UeG54R_dybTceHU5mxnYxWxA5OyMCaZENqoY9Lo8BwpSk-bm2vz8X-YB-6vl6AW4DJ3-0M-TtTtRRhcE_)

# 