**HA NOI UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**

**A red sign with yellow text

Description automatically generated**

**Final Report  
Bomberman**

**Faculty: Information Technology**

**Course: Java Software Development**

**Teacher: MSc. Ngo Van Quyen**

**Students: 1901040002 – Vu Van An**

**1901040155 – Le Anh Phan**

**Date: November 10, 2024**

Table of Contents

[I. Introduction 2](#_Toc182094750)

[II. Related Work 2](#_Toc182094751)

[III. Methodology 2](#_Toc182094752)

[IV. Implementation 4](#_Toc182094753)

[V. Results 4](#_Toc182094754)

[VI. Conclusion 5](#_Toc182094755)

[VII. Contributions 5](#_Toc182094756)

[VIII. References 5](#_Toc182094757)

# Introduction

The goal of this project is to create a game called Bomberman, a traditional action-adventure game in which players demolish obstacles and vanquish adversaries by placing bombs. Using the Java programming skills acquired in the course, the objective is to produce a fully working version of Bomberman that enables users to control the main character to finish stages. Real-time interactivity and graphically captivating gameplay are essential for the game. Beside problems related to GUI, handle events in gameplay, AI enemies are also a problem but since the Breadth-First Search (BFS) technique "explores graph in increasing order of distance from source" and is certain to identify the shortest path in unweighted graphs, we utilize it to find the shortest path between points for the enemies AI to track down player. [1]

# Related Work

Project Tank 1990, which has been studied during a semester in the Java Software Development course, is a popular game in which players control a tank to defend their base and destroy enemies. Tank 1990 serves as a game that fully converges the elements of this course, including multi-threaded programming, meta-programming, GUI development with Swing, network programming, functional programming, and JDBC. Tank 1990 and Bomberman share gameplay features including AI combat and destructible obstacles. To defeat opponents in a limited space, players in both games need to have quick reflexes, agile movement, and smart strategy. Tank 1990 may have been the inspiration for the Bomberman project's gameplay strategies and player-obstacle interaction, but much modification was needed to create a distinctive Bomberman-style experience.

# Methodology

To deliver a productive and organized process, the Bomberman game development technique incorporates a number of ideas, tools, libraries, and programming frameworks. This approach delineates the technologies utilized throughout development and breaks the steps down into distinct phases. Comprising the algorithm, game structure, tools and libraries, and game design

1. Game design

Bomberman is a classic game developed in the 80s of the 20th century, they started with 2D pixel graphics. The game will be implemented on a 16x16 pixel grid, where each cell represents an individual unit of the game.

* Game Board: A grid that shows the player, opponents, obstacles, bombs, and power-ups. Every grid cell will include attributes like whether it has a power-up, bomb, or obstacle.
* Player and Enemies: The Bomberman, the player character, has four directions of motion, and the enemies will follow basic pathfinding algorithms to find and approach the player.
* Explosions & Bombs: The player can set off bombs that will detonate after a certain time, impacting enemies and destructible bricks in their blast radius.
* Power-ups: Items that improve the player's skills, including speed, bomb count, or explosion radius.

1. Tools and Libraries used

* Java Programming Language: Java will be used to develop the game, making use of its portability and object-oriented features. Java's cross-platform flexibility and resilience make it the perfect choice for desktop apps like this one.
* Tools: IntelliJ IDEA to develop projects, MySQL Workbench as a database, and Photoshop to create and modify images, icons, or buttons.
* Libraries and Frameworks: JDBC libraries mysql-connector-j for connection to the database, which its main function is to store the player’s score

1. Game structure and Algorithms

* Classes for game entities: Including map, obstacles, animated entities, power-ups, characters, bombs, etc,.
* Movement and Pathfinding Algorithm: The player’s movement and action are based on keyboard inputs (arrow keys or WASD; X or Space for placing bomb). A method listens for key events and updates the player's position on the grid. Enemies will use a pathfinding algorithm BFS to move toward the player. The enemies are divided into three types of difficulties, which are a heritage from BFS, they may randomly type enemies based on their difficulties and change their movement direction to avoid bombs, which will increase the difficulty.
* Bomb and Explosion Logic: Within a certain range, nearby cells are impacted by the explosion. Each neighboring cell (up, down, left, and right) is examined by the explosion logic to see if it can be destroyed or interacted with.

1. Rendering graphics: Java Swing Components: Java Swing renders the game board using JFrame, JPanel, and Graphics objects. The complete game state, including the grid, player, opponents, and bombs, is rendered by overriding the JPanel class's paintComponent(Graphics g) method.
2. Optimizing Performance: The game uses efficient rendering techniques, such as rendering only the parts of the screen that need updating, to prevent unnecessary computation and improve performance. Also, we implement multi-thread to make sure the game doesn't affect too much on PC resources

# Implementation

The game was implemented using the following Java techniques:

* Multithreading: This technique ensures responsive gameplay even when several interactions are taking place at once by controlling actions like character movement, opponent behaviors, and other game aspects.
* Meta-programming: Standardized methods for different game aspects were developed, enabling flexibility in game growth and modification.
* GUI Development with Swing: The game's interface was designed using Java Swing, which gave each level's characters, obstacles, and explosions a distinct visual arrangement.
* Functional Programming: Used the concepts of functional programming to effectively manage data processing and repetitive operations.
* JDBC: Player scores were stored via JDBC, with a database component added for storing high scores across gameplay sessions.

# Results

A fully playable game is the final product of the Bomberman gaming project. Interfaces, sounds, effects, in-game character interactions, bomb planting, barrier destruction, and engagement with enemies are all available to players. I had a lot of problems handling collisions between objects like placing a bomb but do not let it affect the tile or the ground. And even finding a way to make AI smarter, in addition to using algorithms to suit the AI in the game. Through iterative testing, essential features—such as competitor AI and graphical responsiveness—have been improved. By putting BFS into practice, opponents have been able to move efficiently, chase players through the fastest path, and react quickly to barriers.

# Conclusion

The Bomberman project was an interesting and demanding experience that improved my understanding of the Java programming concepts covered in this Java Software Development course. I gained knowledge about effective code organization, handling unforeseen testing situations, and performance optimization from this project. There are lot of work that can be done to improve this project since it just played as a foundation, I could improve the user experience in the future by using JavaFx as framework, implement network programming, introducing more elements to the game, like a multiplayer mode, gameplay, powerups or harder stages.

# Contributions

To complete this Bomberman project, my team initially consisted of 4 members, but due to some problems, the project ended with only 2 people, including:

I, Vu Van An, am the main programmer and team leader. I was responsible for conceptualizing and designing the game. I write DOC (Final Report), GUIDE (How to install and use), RUN (Sample run images), SLIDE (Presentation Slide).

Accompanying me was Le Anh Phan, graphic designer, responsible for handling the interface, images and sounds. In addition, Phan is also in charge of testing the game, ensuring that the game has no errors during the game, as well as that the sound and images in the game match what the game shows on the screen.

# References

[1]: [CS/ECE 374: Algorithms & Models of Computation](https://courses.physics.illinois.edu/cs374/sp2021/scribbles/A-2021-03-30.pdf)

<https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>