VIETNAM NATIONAL UNIVERSITY – HCM

INTERNATIONAL UNIVERSITY

SEMESTER 2 (2024-2025)
ALGORITHMS DATA STRUCTURE

PACMAN PLUS

Members:
Lê Phan Hồng Huế - ITDSIU23032
Lê Thị Ngọc Tuyền - ITDSIU23028

Table of Contents

CHAPTER 1 : INTRODUCTION

We created a game using basic algorithms and data structure for our final project in the Algorithm & Data Structure course, so we chose to create a PacMan game with more creation and challenge.

The only coding language we utilized for this project was Java. Java is a based-class. In this report, we will go over this game in greater detail, including how it may be created, what type of game it is, and how players can participate. We also talk about how we apply algorithms and data structure in this game.

CHAPTER 2: RULE AND GAME PLAY

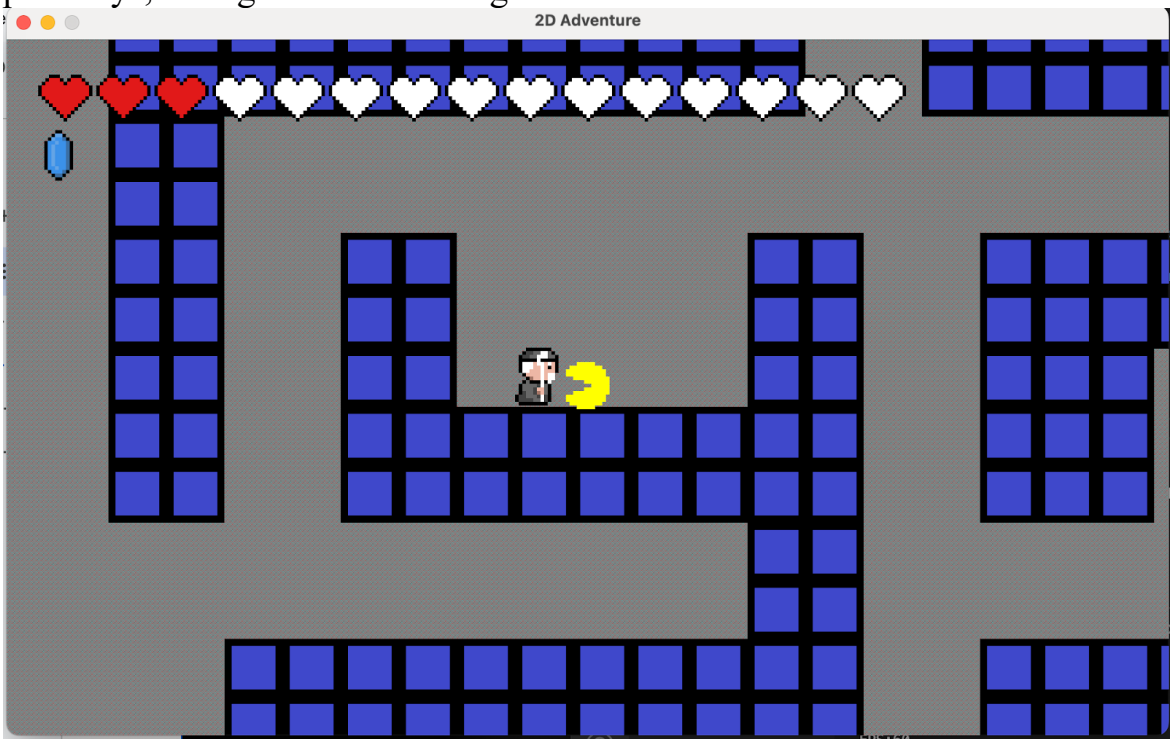
The game we make is a top-down 2D-platformer game inspired by youtuber name roySnow and Kenny Yip Coding. In the normal Pac Man, to win the game , we need to control Pac-Man to find the chest while avoiding the ghosts or defeating the ghost.However, we do not avoid the ghosts forever, we decided that our Pac-Man could attack against these ghosts and receive hearts, after defeating the final ghost, Pac-Man would receive a chest and win the game.To make the game more fair, Pac-Man can only see a small area around him. Since the ghosts only appear at night, Pac-Man's vision is limited due to the darkness.

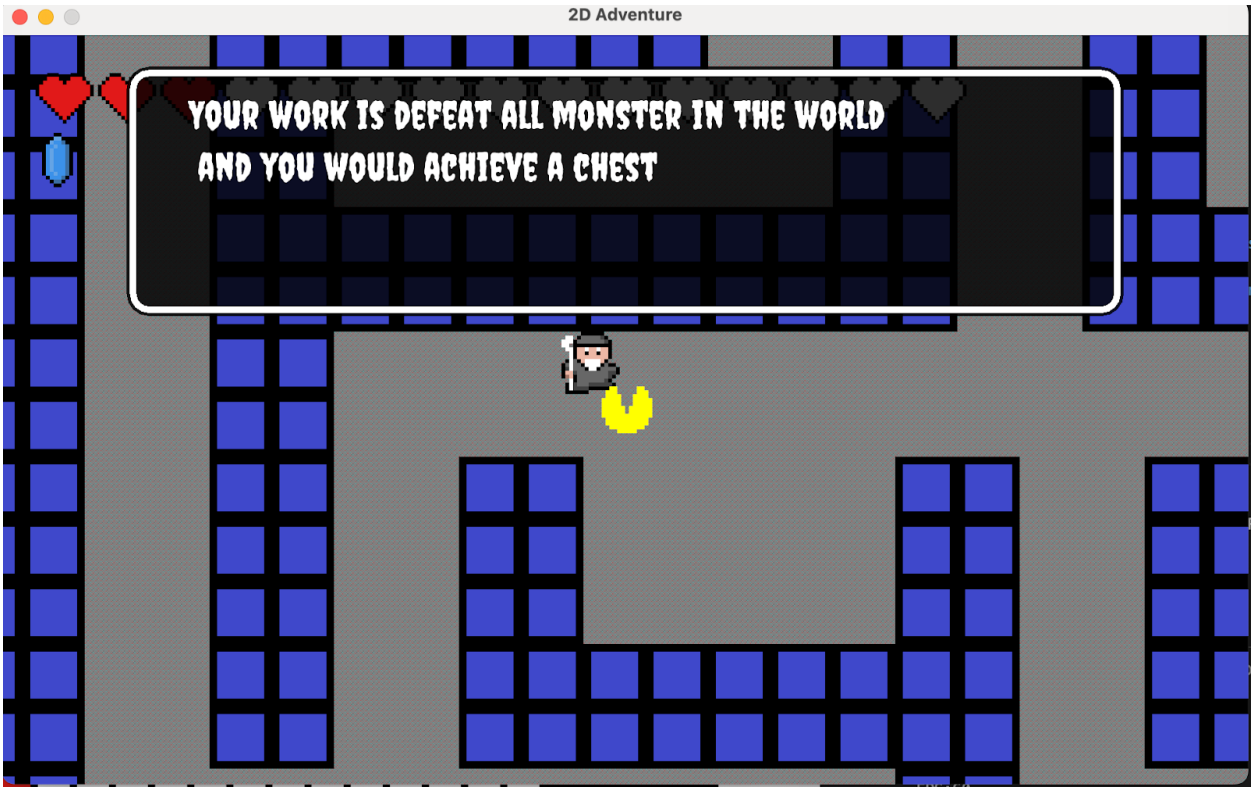
The game we made is a top-down 2D- platformer game inspired by youtuber named ryoSnow where our character is moving around in different levels to find objects that need to kill some monster, to increase its blood.

To play this game, we always use four main push buttons: W to go up, A to turn left, S to move down, and D to turn right and SPACE to attack the monster. The other push button is ENTER, which the user uses when they want to talk with the npc that we created to talk with the character. Another one is ESC which is often used to open the setting and turn off the menu of the game when we turn it on.

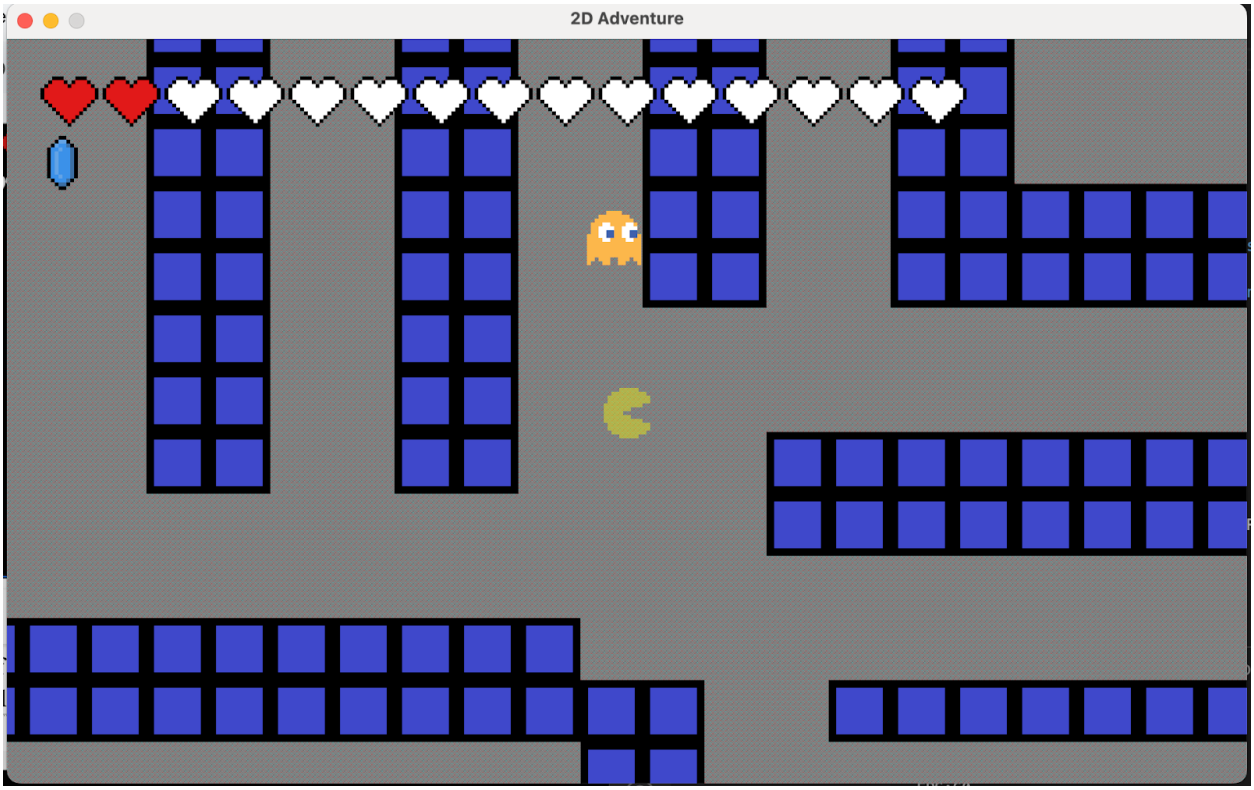
When you open the game, the first appearance will be look like this

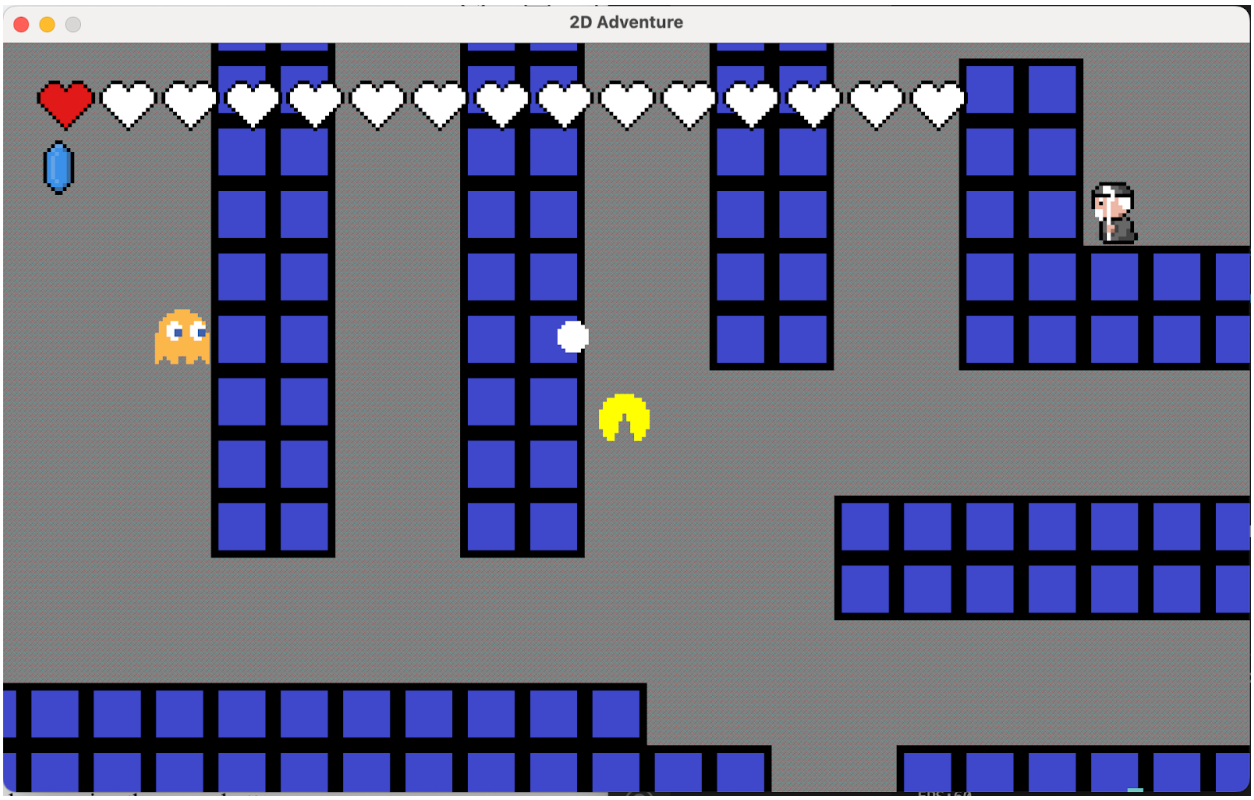
Then, click "New game" to begin the game or "quit" to end it. After clicking "New game," the application will take you to the map, which represents the level of the "PacMan Plus game". Here, your journey unfolds as you navigate through intricate pathways, facing diverse challenges and obstacles.



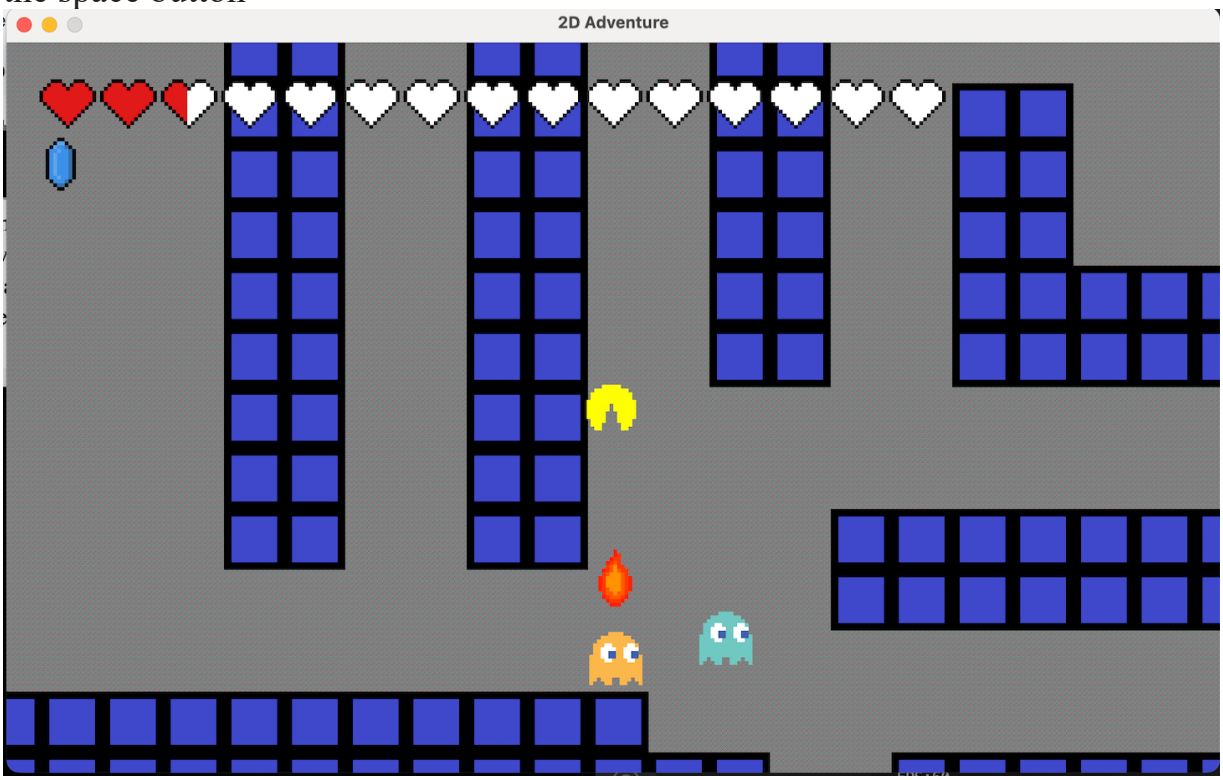


The player meet NPC who guide you understand the meaning and how to win the adventure

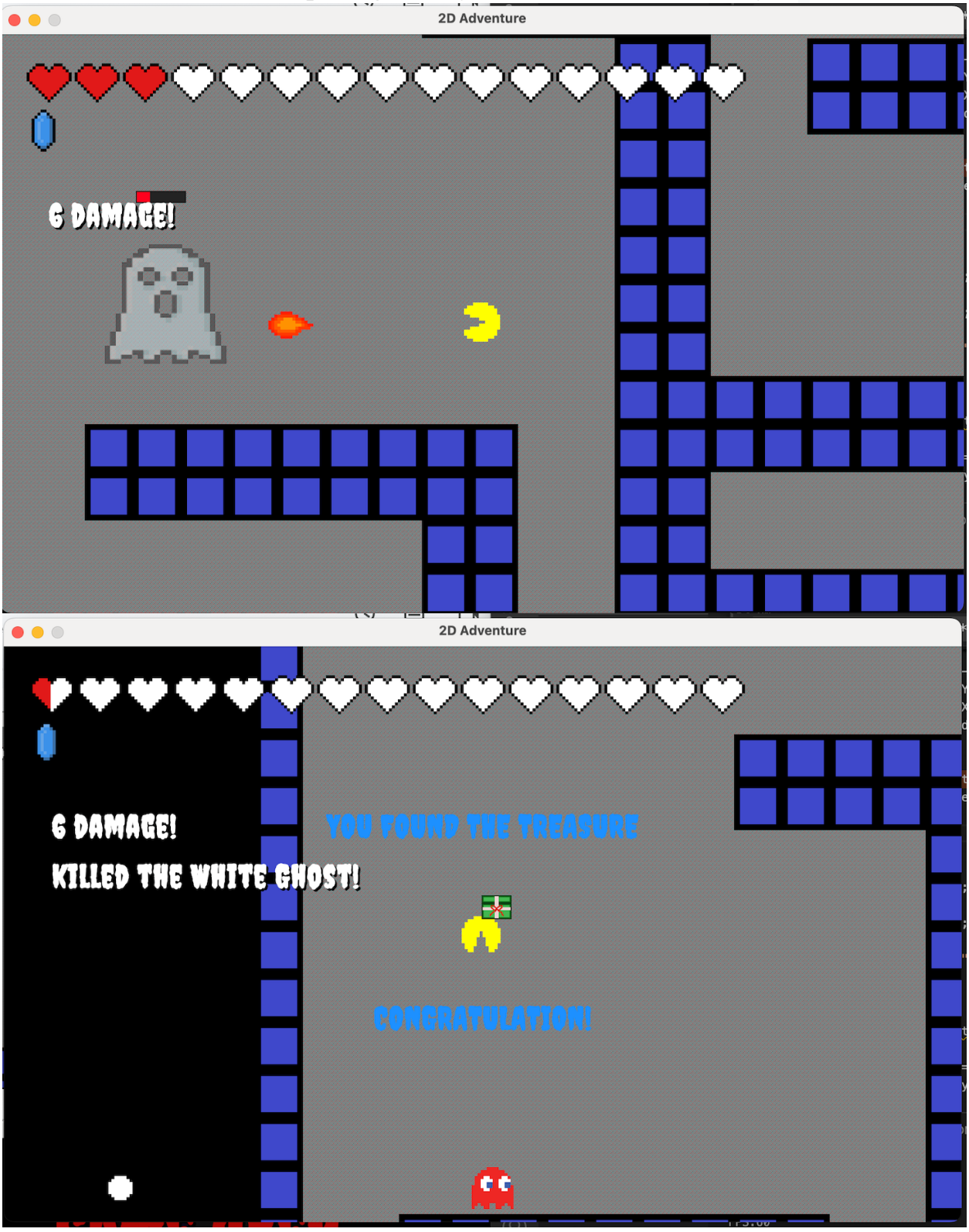




There are many ghosts attacking the player. When you touch or are hit by a ghost bullet, you will lose half of your heart. If you lose all his heart, you will lose. If you want to get more lives to survive, you need to kill the ghost with a fireball by pressing the space button



After the little ghost is killed, you get more lives to defeat the boss. When the white boss dies, it will drop the gift for you and touch it will win you.



After losing, you have two options, one is "RETRY" to be revived at the original position when entering the game with all the items and levels available before being defeated. The other is "QUIT" will help you exit the game completely and return to the NEW GAME lobby.

CHAPTER 3: ALGORITHM AND DATA STRUCTURE

1. ArrayList:

- a dynamic list in Java which allow:
  - add elements(objects) flexibly
  - remove element easily
  - Grow or shrink in size automatically
- In this game, we have many “ entities”
  - Player
  - NPC[non-player characters]
  - Ghosts
  - ProjectileList
  - ParticleList
  - Object
  - EntityList

- Why we use “ArrayList”:

a) projectileList: Managing projectiles

When the player shoots a fireball, the protiles flies disappears after a few seconds or upon hitting an enemy

If using a regular array, i’d would have manually find an empty slot,and shift elements when removing - inconvenient

With Arraylist :

```
projectileList.add(new Fireball(..)); // Add new
projectile

if(projectile.alive == false) {
   projectileList.remove(i); }   // remove it when it’s
no longer
```

No need to know the number of projectile in advance. It grows/shrinks as needed

b) particleList: Managing visual effects

Used to manage short-lived effects in the game, to make game feel more dynamic and alive.

When a monster dies, fades away after a few frames.

With ArrayList:

```
particleList.add(new Particle(..));
if(particleList.alive == false ){
   particleList.remove(i);
}
```

If using a regular array, i’d would have manually find a slot to add new particle and shift elements when removing - inconvenient

c) entityList: Temporary list used for drawing

Games need to draw entities in the correct order(from top to bottom on the screen) to avoid:

Ghosts behind the player appearing in front

Bullets flying underneath characters

Combine all drawable entities into one list

```
entityList.add(player);
entityList.add(npc[i]);
```

```
entityList.add(ghost[i]);
entityList.add(projectileList.get(i));
entityList.add(particleList.get(i));
Sort them by worldY
Collections.sort(entityList, new Comparator<>() {
   public int compare(Entity e1, Entity e2) {
       return Integer.compare(e1.worldY, e2.worldY);
   }
});
```

In top-down 2D games, entities lower on the screen (higher worldY) should appear in front of entities that are higher (lower worldX), just like in real life.

Without sorting, an object that's supposed to be "in front" may be drawn behind another object, making it look wrong.

2. **Breath-First Search(BFS)**
   which is a graph or grid traversal algorithm that explores all neighboring nodes at current level before moving to the next level.

Queue<Node>

- Used in Breadth-First Search (BFS) to explore map tiles level by level

Node Class:

- Represents each tile with coordinates and a link to its parent

3. **Breadth-First Search(BFS)**
   - Find the shortest path from the ghost to the player
   - Convert the ghost's and player's position from pixels to tile units
   - Initialize a visited map and queue with the starting node
   - Use BFS to explore the map(up,down,left,right)
   - Stop when the player's tile is found
   - Track back from target to ghost using the parent links to find next move

CHAPTER 4: TIME COMPLEXITY

1. ArrayList

```
add(E element)
```

Time complexity : O (1)
Add new element into the end list. When reach to the limit size, ArrayList would create new bigger array size, copy the data - in this case time complexity is O(n)

```
remove(int index)
```

Time complexity : O(n)

Convert all element behind the index which be removed

```
get(int int index)
```
Time complexity : O(1)

Access to exactly index

```
size()
```
Time complexity : O(1)

Release the numbers index

```
Collections.sort()
```

Combine merge sort and insertion sort

Time complexity:

- In best case: List have been arrange: O(n)
- In worse case: O(nlog(n))

2. Breadth-First Search(BFS)

```
public Queue<Node> queue = new LinkedList<>();
boolean[][] visited = new boolean[rows][cols]
```

Time complexity: O(V+E)

- V: numbers tiles in map
- E : numbers edge- every tile have 4 edges (up, down,left, right)

Space complexity : O(V) : visited, queue, parent