STUDIO
SHODWE

# LINE ADVENTUR
# GAME

TEAM 3:

- TRẦN NAM ANH
- NGUYỄN ĐỨC HẢI
- NGÔ THỊ ÁNH DƯƠNG
- ĐẶNG MINH PHÁT

STUDIO
SHODWE

# **TABLE**
# CONTENTS



STUDIO
SHODWE

# INTRODUCTION

STUDIO
SHODWE

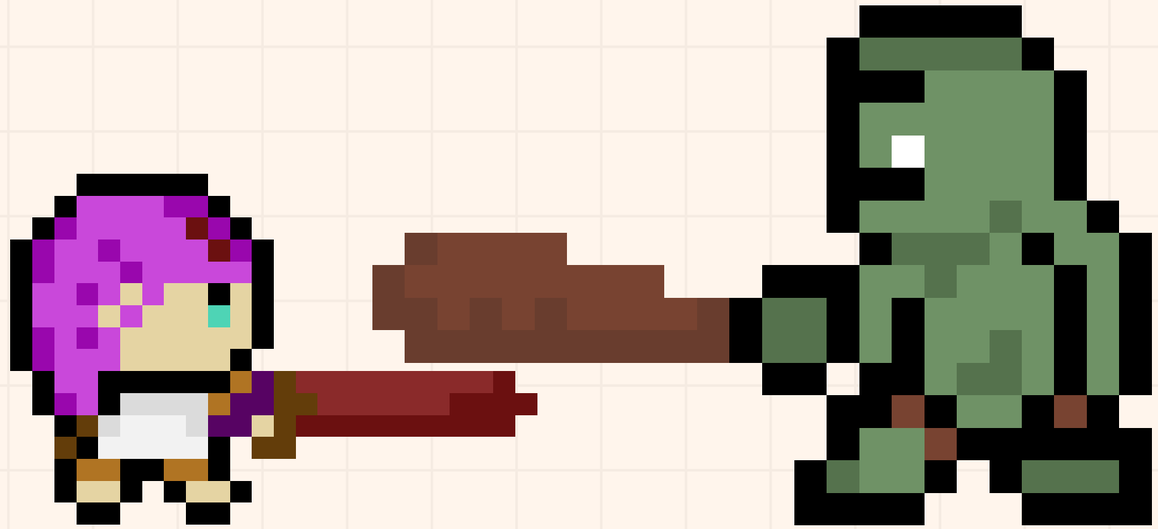
# INTRODUCTION

- **About the Game:**
  - **Genre**: Adventure, player as a discoverer.
  - **Gameplay**: Combines interactive scenarios & narrative-driven elements.
  - **Experience**: Mentally stimulating and emotionally relaxing.
- **Development Objectives (for Team):**
  - **Apply OOP Concepts:** Inheritance, Encapsulation, Polymorphism.
  - Learn & Adhere to **SOLID** Principles.
  - Practice **Common Design Patterns**.
  - Strengthen **theoretical knowledge** & practical **software engineering skills**.

STUDIO
SHODWE

# **PROJECT**
# OBJECTIVES

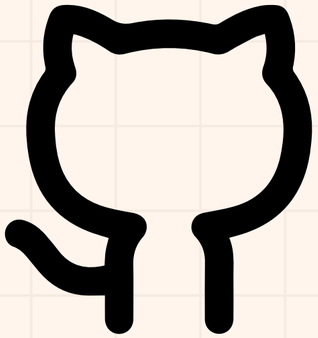
- A creative, thought-provoking and interesting game
- The Object-Oriented Programming course to the real-life scenarios
- Program collaboration

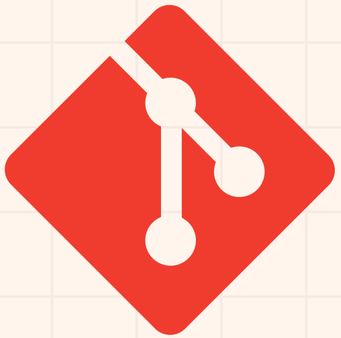


STUDIO SHODWE

# **TECHNIQUES** & TOOLS USED

- Programming Language: **Java**
- Integrated Development Environment (IDE): **IntelliJ**, **VSCode**.
- Version control and collaboration: **Git** and **GitHub**
  - Git: used to track all versions of change.
  - GitHub: used for creating, storing, managing and sharing code.
- **Piskel**: draw pixel images.



STUDIO
SHODWE

# GAME DESCIPTION

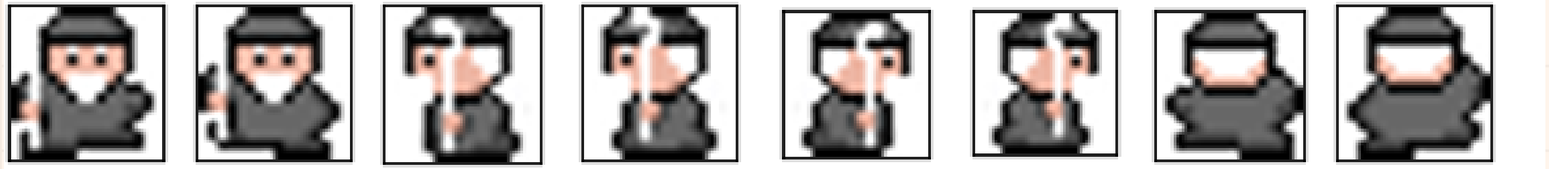
STUDIO
SHODWE

# DESIGN

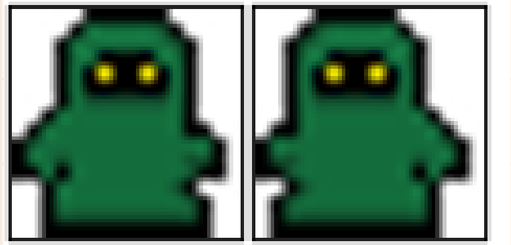
## 1, Character

MAIN
CHARACTER

NPC
OLD MAN



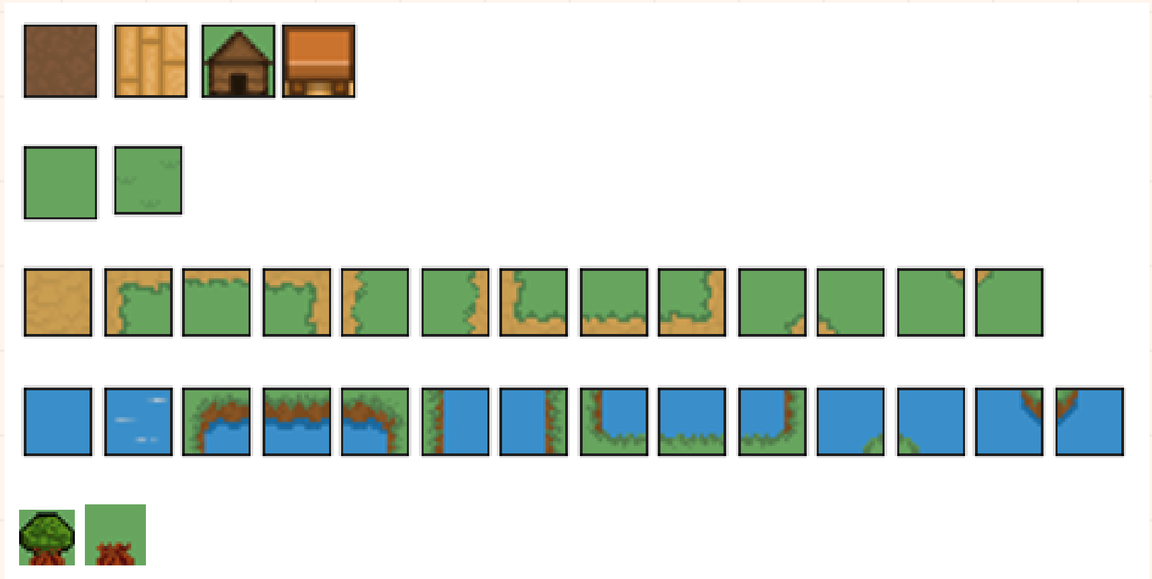
NPC
MERCHANT



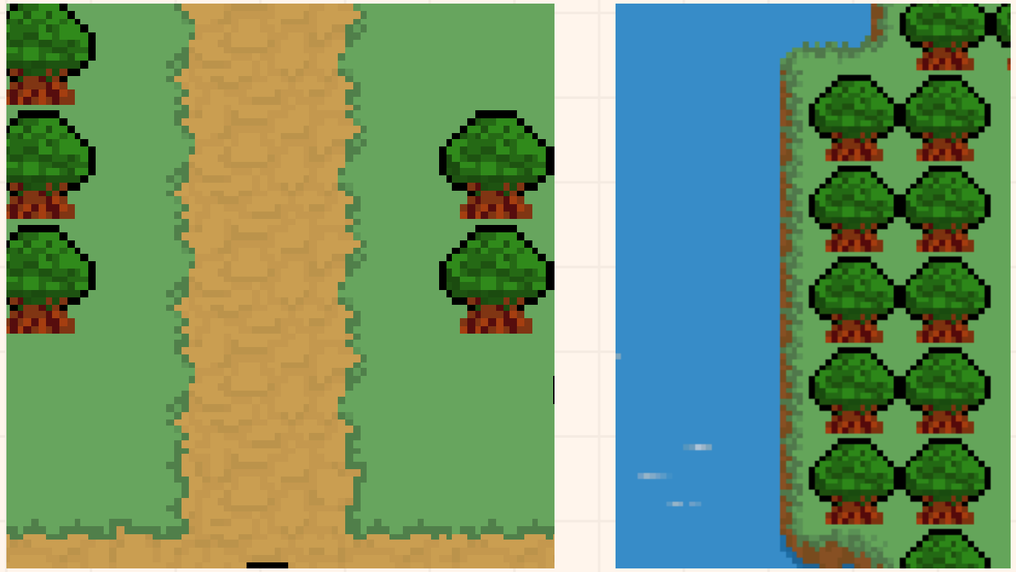
STUDIO
SHODWE

# DESIGN

## 2, Background design



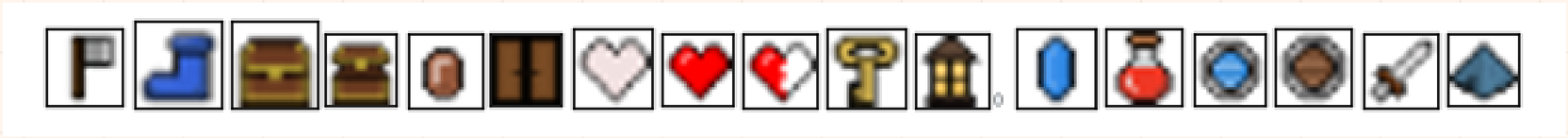
TILES



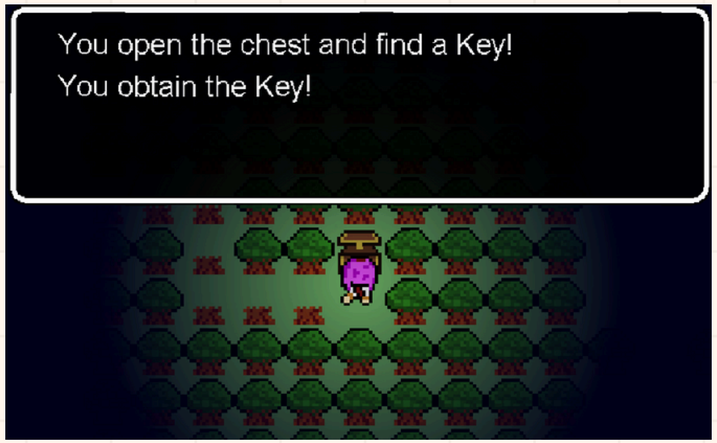
MAPS

STUDIO
SHODWE

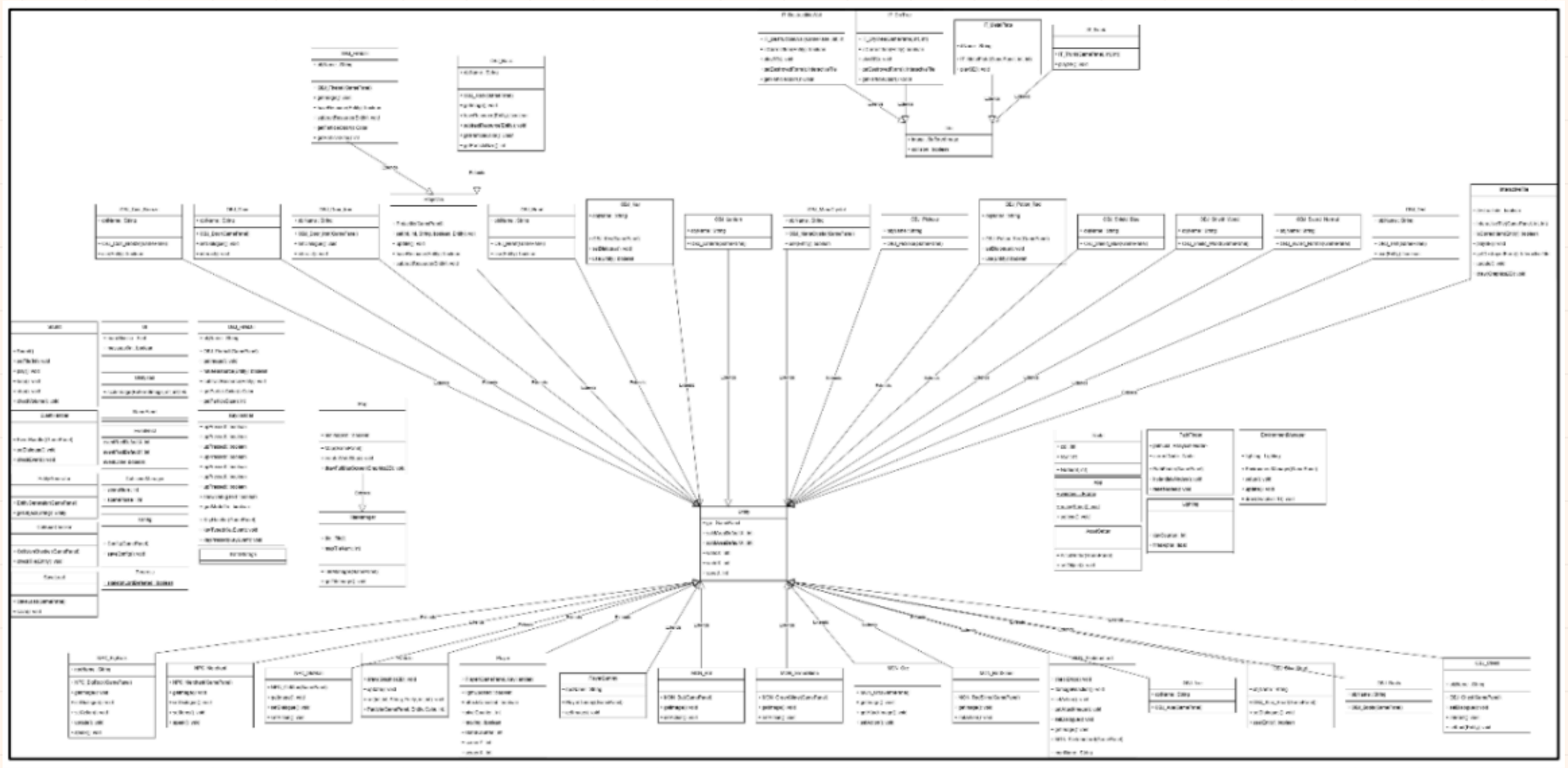
# 2, Background design



OBJECTS THAT THE PLAYER WILL INTERACT WITH

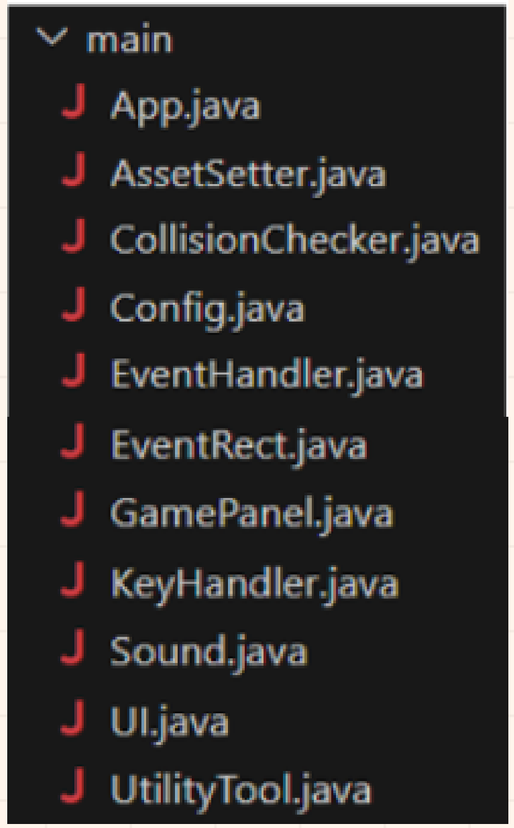
# UML



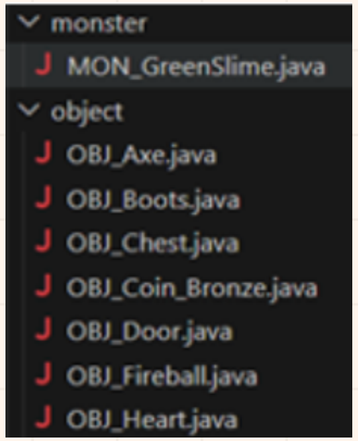
STUDIO SHODWE

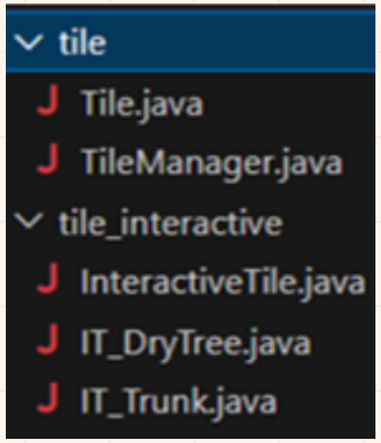
# CORE GAME

## 1. GAME DESIGN






- Package: **9 folder**
- Class: **60 file**
- **13 main class** → control and combine others
- App.java is the **main run** of this game

STUDIO SHODWE

# CORE GAME

## 1. GAME DESIGN

```
window = new JFrame();
window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
window.setResizable(resizable:false);
window.setTitle(title:"2D Adventure");
```

Window Setup

```
GamePanel gamePanel = new GamePanel();
window.add(gamePanel);
gamePanel.config.loadConfig();
if (gamePanel.fullScreenOn == true){ ···

window.pack();
window.setLocationRelativeTo(c:null);
window.setVisible(b:true);
```

Add GamePanel

STUDIO SHODWE

# CORE GAME

## 1. GAME DESIGN

```
window = new JFrame();
window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
window.setResizable(resizable:false);
window.setTitle(title:"2D Adventure");
```

Window Setup

```
GamePanel gamePanel = new GamePanel();
window.add(gamePanel);
gamePanel.config.loadConfig();
if (gamePanel.fullScreenOn == true){ ···

window.pack();
window.setLocationRelativeTo(c:null);
window.setVisible(b:true);
```

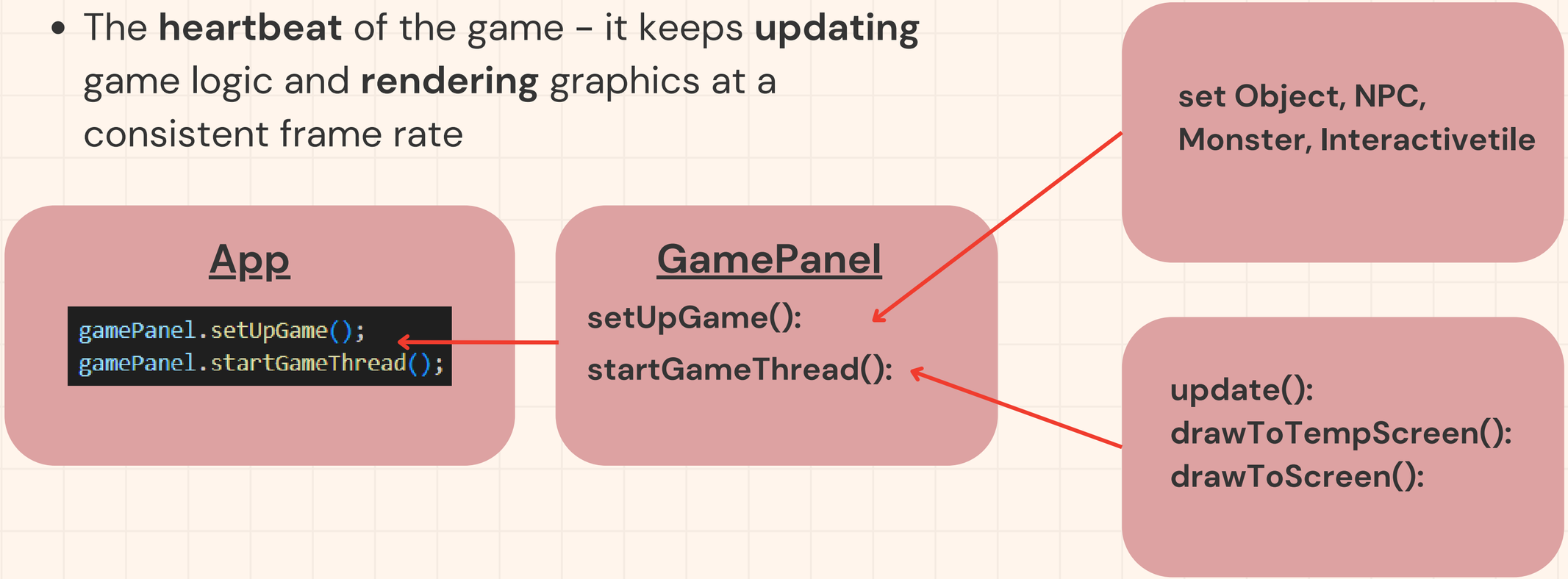
Add GamePanel

STUDIO SHODWE

# CORE GAME

## 1. GAME DESIGN

- The **heartbeat** of the game – it keeps **updating** game logic and **rendering** graphics at a consistent frame rate



STUDIO SHODWE

# CORE GAME

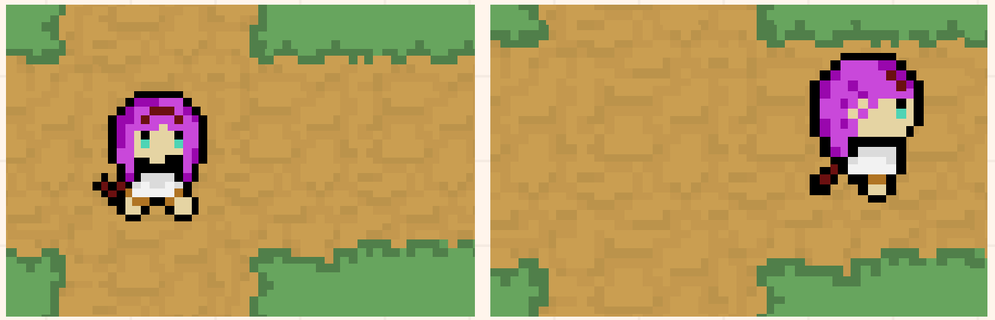
## 2.ENTITY AND PLAYER

### 2.1 Entity: A. Update

```
// CHECK ENTITY COLLISION
    if (collisionOn == false) {
        switch(direction) {
            case "up": worldY -= speed; break;
            case "down": worldY += speed; break;
            case "left": worldX -= speed; break;
            case "right": worldX += speed; break;
        }
    }
```

False

Entity can go through



True

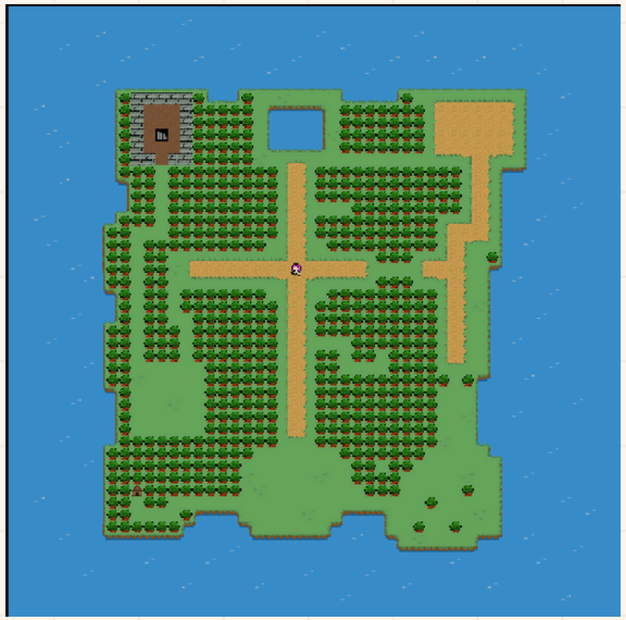
Entity can’t go through

STUDIO SHODWE

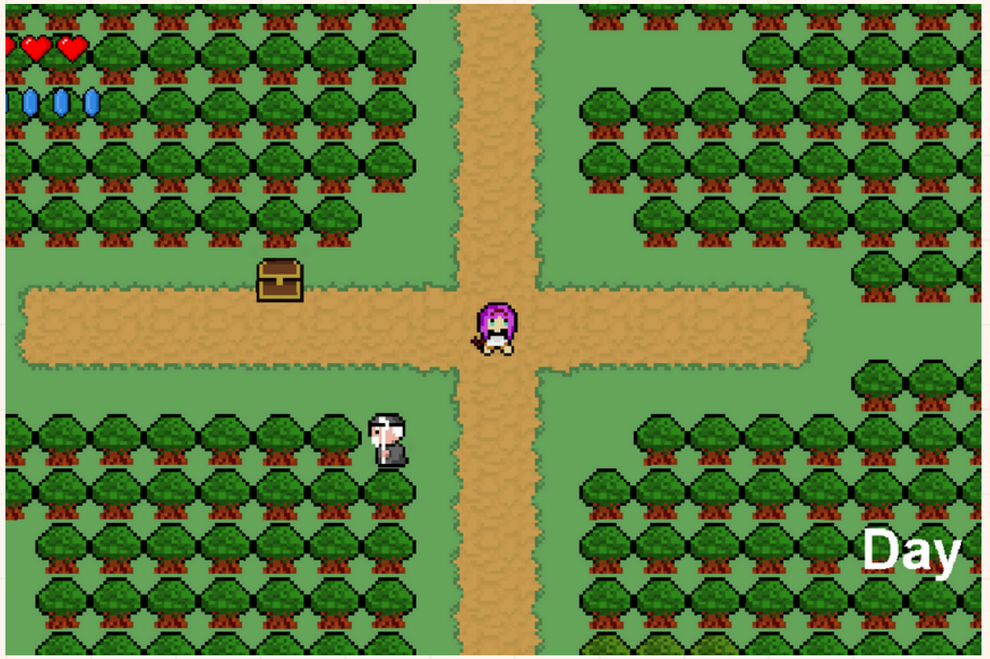
# CORE GAME

## 2.ENTITY AND PLAYER

### 2.1 Entity: B.Draw (step 1: draw the map)



Instead of showing a whole map




show only 5 tiles from player

STUDIO SHODWE

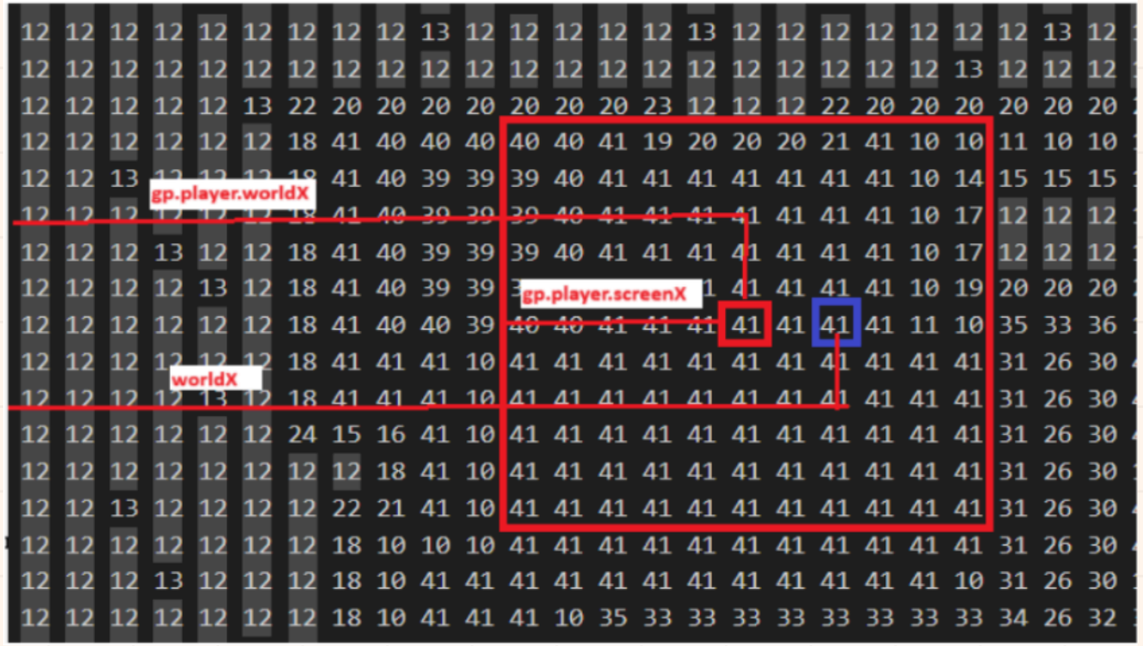
# CORE GAME

## 2.ENTITY AND PLAYER

### 2.1 Entity:

B.Draw (step 2: set default position)

```
int screenX = worldX - gp.player.worldX + gp.player.screenX;
int screenY = worldY - gp.player.worldY + gp.player.screenY;
```



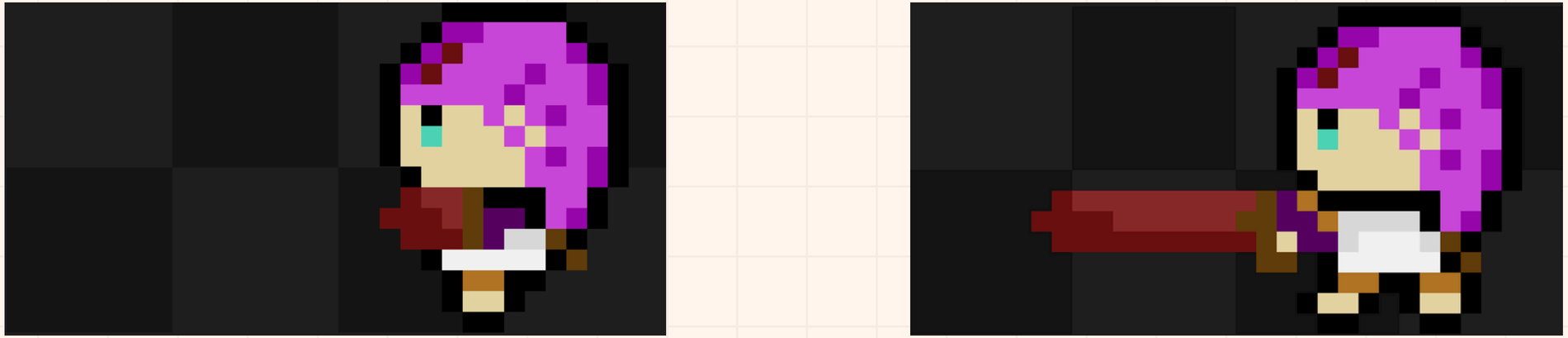

Choosing place to display objects

STUDIO SHODWE

# CORE GAME

## 2.ENTITY AND PLAYER

### 2.1 Entity: B.Draw (step 3: draw player)



Draw player action

STUDIO SHODWE

# CORE GAME

## 2.ENTITY AND PLAYER

### 2.1 Entity:

### B.Draw (step 3: draw player)

```
if (worldX > gp.player.worldX - gp.player.screenX - gp.tileSize
    && worldX < gp.player.screenX + gp.player.worldX + gp.tileSize
    && worldY > gp.player.worldY - gp.player.screenY - gp.tileSize
    && worldY < gp.player.worldY + gp.player.worldY + gp.tileSize) {

        switch(direction) {
            case "up" -> {
                if (spriteNum == 1) {image = up1;}
                if (spriteNum == 2) {image = up2;}
                }
            case "down" -> {
                if (spriteNum == 1) {image = down1;}
                if (spriteNum == 2) {image = down2;}
                }
            case "left" -> {
                if (spriteNum == 1) {image = left1;}
                if (spriteNum == 2) {image = left2;}
                }
            case "right" -> {
                if (spriteNum == 1) {image = right1;}
                if (spriteNum == 2) {image = right2;}
                }
        }
```

STUDIO SHODWE

# CORE GAME

## 2.ENTITY AND PLAYER

### 2.2 Player:

**A. Unattacking Mode**

- Check collision
- Check interaction with the objects

```
// COLLISION CHECK
collisionOn = false;
gp.cChecker.checkTile(this);

int objIndex = gp.cChecker.checkObject(this, player:true);
pickUpObject(objIndex);

int npcIndex = gp.cChecker.checkEntity(this, gp.npc);
interactNPC(npcIndex);

int monsterIndex = gp.cChecker.checkEntity(this, gp.monster);
contactMonster(monsterIndex);

// CHECK INTERACTIVE TILE
gp.cChecker.checkEntity(this, gp.iTile);
gp.eHandler.checkEvent();
```

STUDIO SHODWE

# CORE GAME

## 2.ENTITY AND PLAYER

### 2.2 Player: A. Unattacking Mode

```
// MOVE IF NO COLLISION
if (!collisionOn) {
    switch(direction) {
        case "up" -> worldY -= speed;
        case "down" -> worldY += speed;
        case "left" -> worldX -= speed;
        case "right" -> worldX += speed;
    }
}
```

After interactions

STUDIO SHODWE

# CORE GAME

## 2.ENTITY AND PLAYER

### 2.2 Player:

B. Attacking Mode

```
// Save the current worldX, worldY, solidArea
int currentWorldX = worldX;
int currentWorldY = worldY;
int solidAreaWidth = solidArea.width;
int solidAreaHeight = solidArea.height;

//Adjust player's worldX/Y for the attackArea
switch(direction) {
    case "up" -> worldY -= attackArea.height;
    case "down" -> worldY += attackArea.height;
    case "left" -> worldX -= attackArea.width;
    case "right" -> worldX += attackArea.width;
}
```

Before interaction

STUDIO SHODWE

# CORE GAME

## 2.ENTITY AND PLAYER

### 2.2 Player: B. Attacking Mode

```
int monsterIndex = gp.cChecker.checkEntity(this, gp.monster);
damageMonster(monsterIndex,attack, currentWeapon.knockBackPower);
int iTileIndex = gp.cChecker.checkEntity(this, gp.iTile);
damageInteractiveTile(iTileIndex);
int projectileIndex = gp.cChecker.checkEntity(this, gp.projectile);
damageProJectile(projectileIndex);
```

While interaction

STUDIO SHODWE

# CORE GAME

## 2.ENTITY AND PLAYER

**2.2 Player:** B. Attacking Mode

```
worldX = currentWorldX;
worldY = currentWorldY;
solidArea.width = solidAreaWidth;
solidArea.height = solidAreaHeight;
```

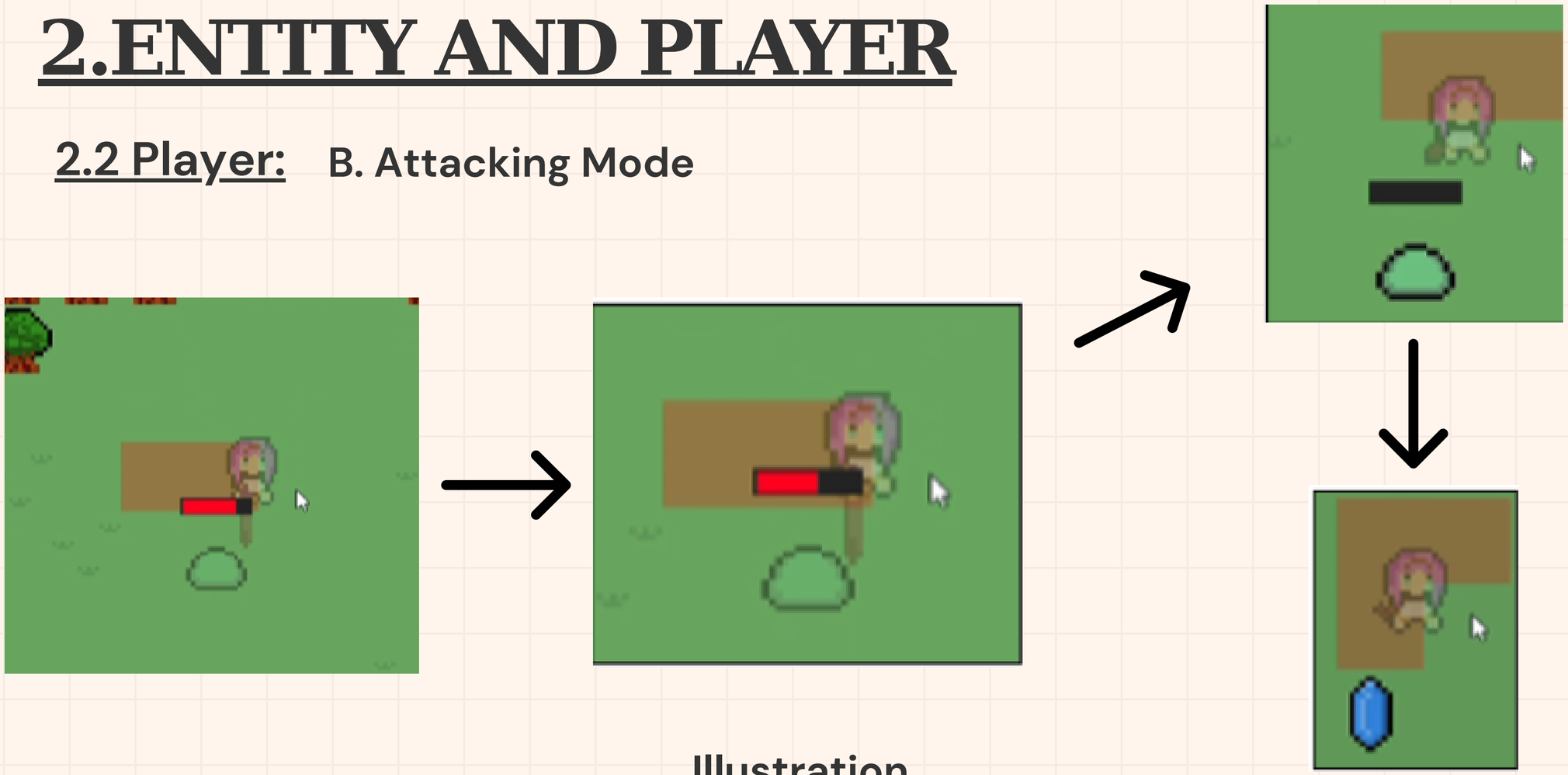
After interaction

STUDIO SHODWE

# CORE GAME

## 2.ENTITY AND PLAYER

### 2.2 Player: B. Attacking Mode



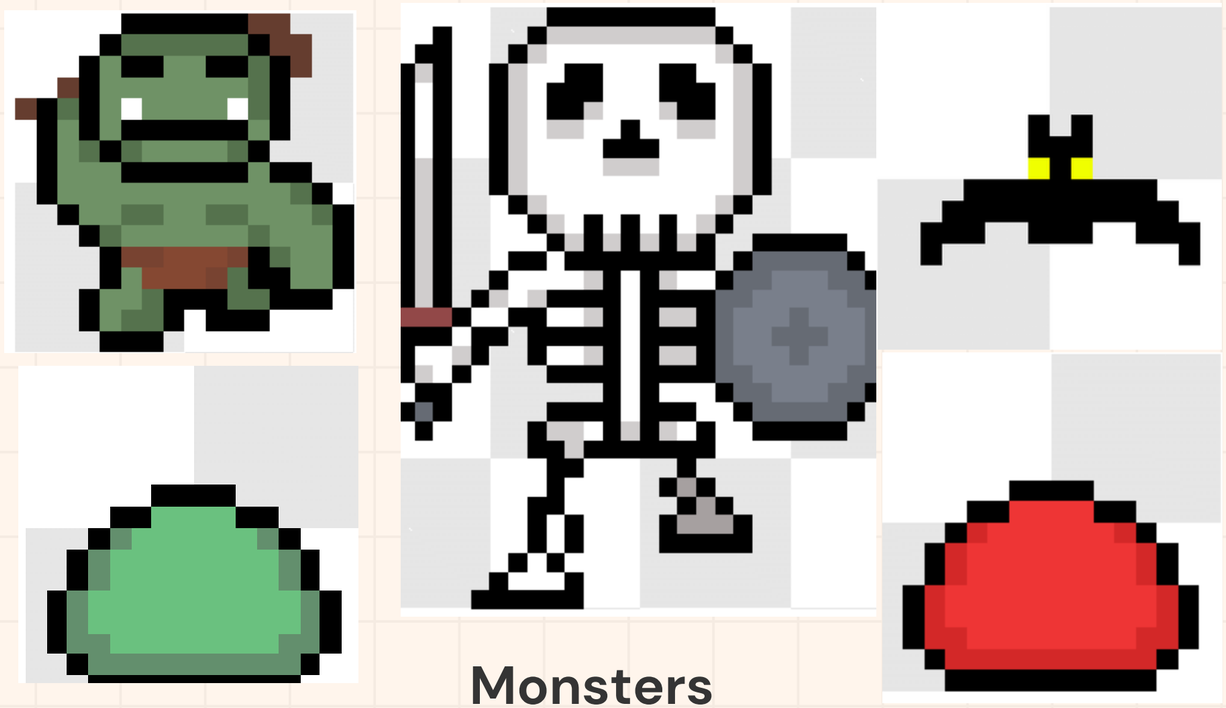
Illustration

STUDIO SHODWE

# CORE GAME

## 2.ENTITY AND PLAYER

**2.3 Features:** A* algorithm - pathFinding features



Monsters

NPC Old Man

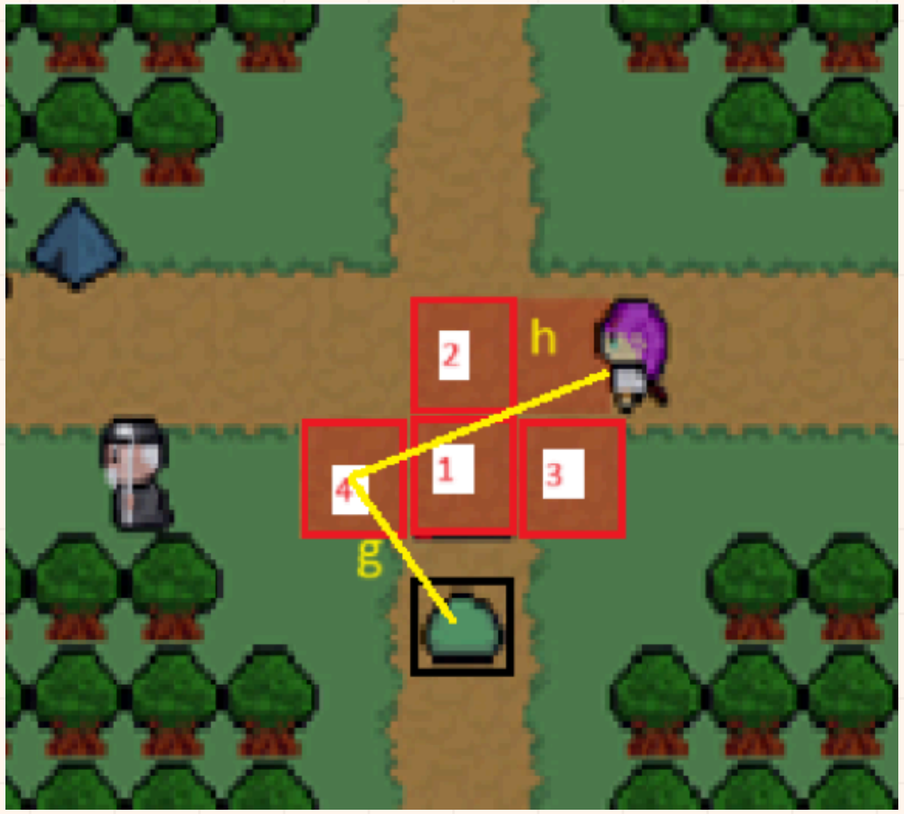
STUDIO SHODWE

# CORE GAME

## 2.ENTITY AND PLAYER

### 2.3 Features:

**A* illustration**




**H cost**: distance from the gold node to the checking cell

**G cost:** distance from the starting node to the checking cell

→ Set **F cost = H cost + G cost**

STUDIO
SHODWE

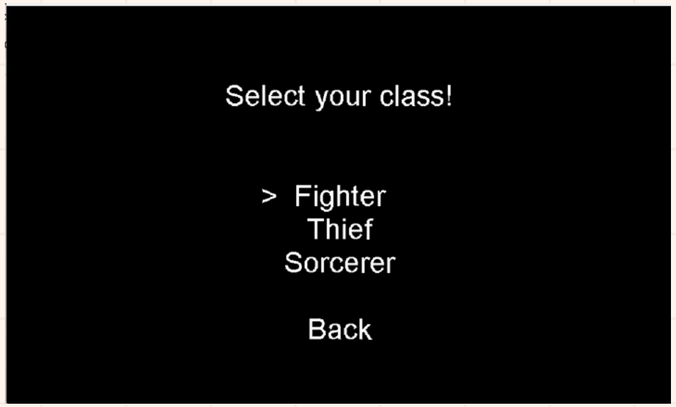
# GAME RULE

STUDIO SHODWE

# Basic guide and rules of the game

- **Game Concept**: In this **monster hunting game**, players take the role of **an adorable girl** who embarks on a journey to **train**, **collect** useful items, and **defeat** monsters to gain **experience** and **grow stronger**.

- **Starting Point**: The game begins **at the center of the map**. Players must **explore** the world, **level up** by defeating monsters, and ultimately **defeat** the **final Boss** to **complete the game**. Before starting, players **choose** one of three character **classes**: Fighter, Thief, or Sorcerer.




# MONSTER HUNGTING SYSTEM

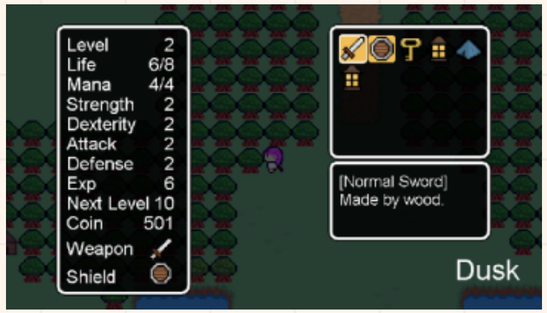
+ Players **control** character by using **WASD** to **move** around the map and **push** the giant rock.

# LEVELING AND EXP SYSTEM

- Once the **character** gains **experience** by **defeating monsters**, the character's level will **level up automatically**.

- When **leveling up**, the player will have **higher character stats**

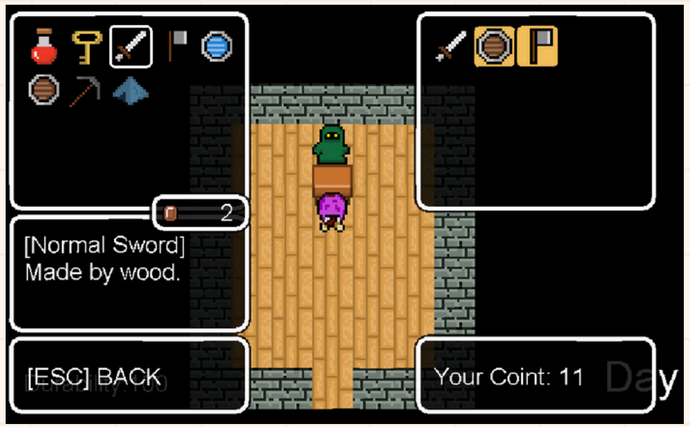


# ITEM EXCHANGE SYSTEM

- Players can **sell** received items to NPC to get **coins** and **buy** valuable **items** at a list price in the system.

- The price of the item the **player sells** will be lower than that of the item the **player buys** from the NPC.

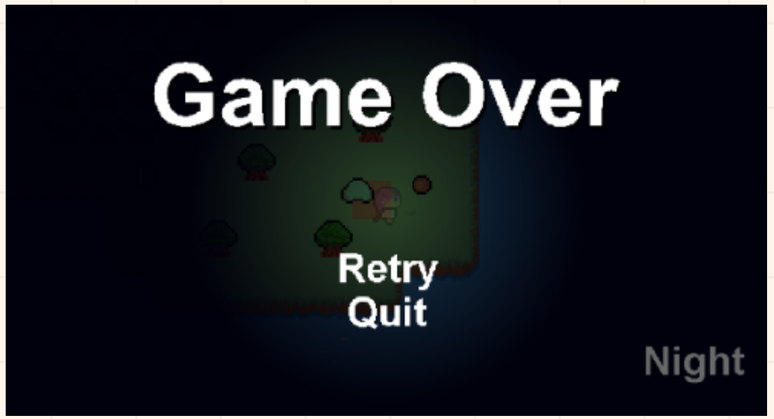



# HEALTH SYSTEM AND GAME OVER CONDITION

The character has a visible Health Bar (HP) displayed on the screen.


The character's HP will decrease when they take damage from monsters


If HP reaches zero, it's Game Over, and the player must restart from the beginning



# SETTING GAME SYSTEM

- Player can **adjust** the game's **volume**, **music** to fix with individual's interest
- The side of game's window can be **full screen** or **not**
- Players can **stop** the game and **save** it. one index and everything will be saved at the time the player saves the game

# THE TIME SYSTEM

THE TIME IS DIVEDED TO 4 PHASES




STUDIO
SHODWE

# GAME DEMO

STUDIO
SHODWE

# Link demo clip

**Scan here**

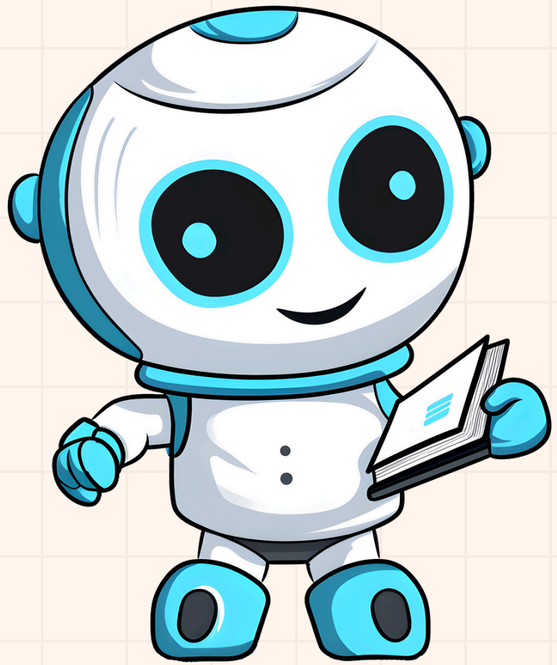
STUDIO
SHODWE

# Future Work

STUDIO SHODWE

# Future Work

- **Content Expansion:** More maps, enemies, NPCs, quests & items.
- **Gameplay & AI:** Advanced skills, improved AI, deeper combat, boss battles.
- **Technical & UX:** Enhanced visuals & sound, UI/UX polish, optimization, robust save.
- **Advanced Features:** Multiplayer, procedural content, modding support.



STUDIO
SHODWE

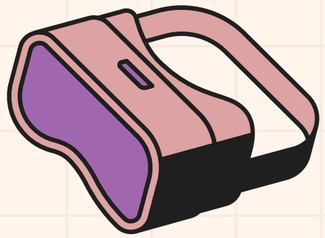
# CONCLUSION

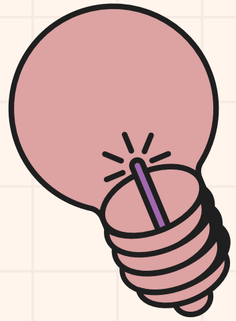
STUDIO
SHODWE

# CONCLUSION

- **Success**: effectively applying OOP principles (inheritance, polymorphism, encapsulation, abstraction).
- **Key Features:** Engaging gameplay, maintainable/scalable architecture, character interaction, combat, A* pathfinding.
- **Skills Gained**: Enhanced programming, fostered collaboration, understood software dev workflows.
- **Overall:** Blended creativity & technical skill, producing an educational & entertaining software grounded in modern programming.

STUDIO
SHODWE





# THANK YOU