

PvZ - CodeStrike

OOP Project

Advisor : Prof. Trần Thanh Tùng

TEAM MEMBERS



Nguyễn Thế Khoa
ITCSIU24042



Trần Khánh Bình
ITCSIU24015



Nguyễn Hưng
ITCSIU24030



Nguyễn Hữu Ninh
ITCSIU24066



Hoàng Triệu Nam
ITCSIU24059

CONTENTS

01.

Introduction

02.

Gameplay

03.

Demo

04.

UML/Class
Diagram

05.

Conclusion

Introduction

In our simplified version of Plants vs Zombie game, the architecture adheres to Object-Oriented Programming (OOP) principles to maintain modularity, scalability, and ease of maintenance. Key OOP concepts applied include:

- **Encapsulation:** Game elements such as **Plant**, **Zombie**, and **Lawnmower** are structured with private attributes and controlled access via public methods.
- **Inheritance:** The **Zombie** class serves as a base type, with specialized subclasses such as **BrickheadZombie** inheriting core functionality while introducing unique mechanics.
- **Polymorphism:** Entities like **Zombie** and **Lawnmower** share a common interface, enabling dynamic behavior changes based on context without requiring explicit type-checking.

Additional gameplay enhancements:

- **Lawnmower mechanic** added, functioning as a last line of defense that automatically clears approaching enemies upon activation.
- **Level selection system** integrated, allowing players to choose specific stages rather than following a linear progression.



Gameplay



01.

Characters

- Plants: Sunflower, Peashooter, Potato Mine, Walnut, and Cactus.
- Zombies: Basic Zombie, Conehead Zombie, Buckethead Zombie, and Brickhead Zombie.

02.

Gameplay

- Collecting and using the sun to plant the Plants to defend your home.

03.

Levels

- There are 4 levels to play through.
- After beating each level, you will achieve a new Plant.
- For the 4th level, this can be played by pressing “4” on the keyboard.

04.

Running platform

- Using Greenfoot platform to execute the game.

GREENFOOT

GREENFOOT APPLICATION

<https://www.greenfoot.org/overview>

Greenfoot is an educational tool for learning Java

- interactive
- user-friendly

Highly recommended for budding coders !



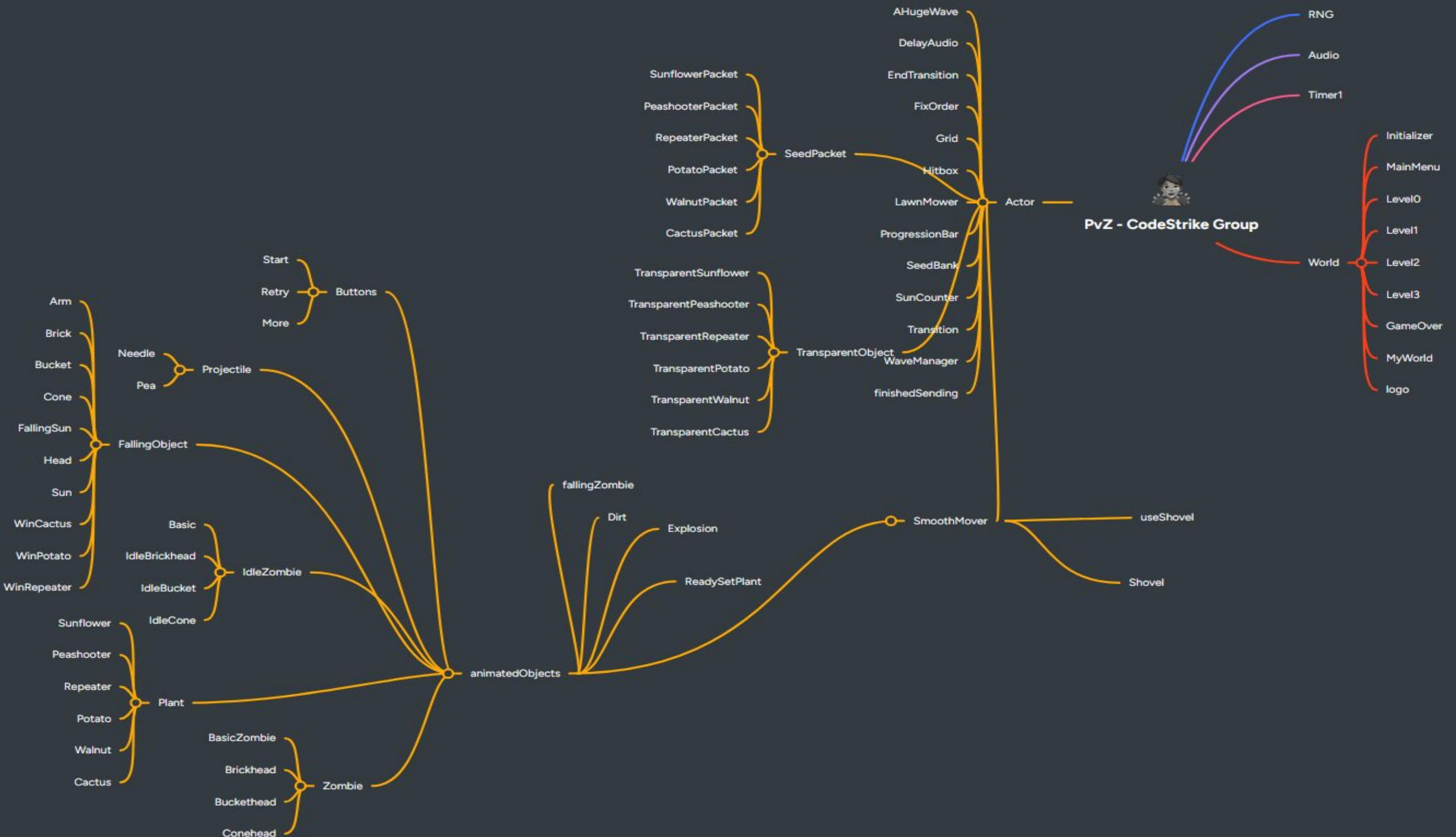
DEMO

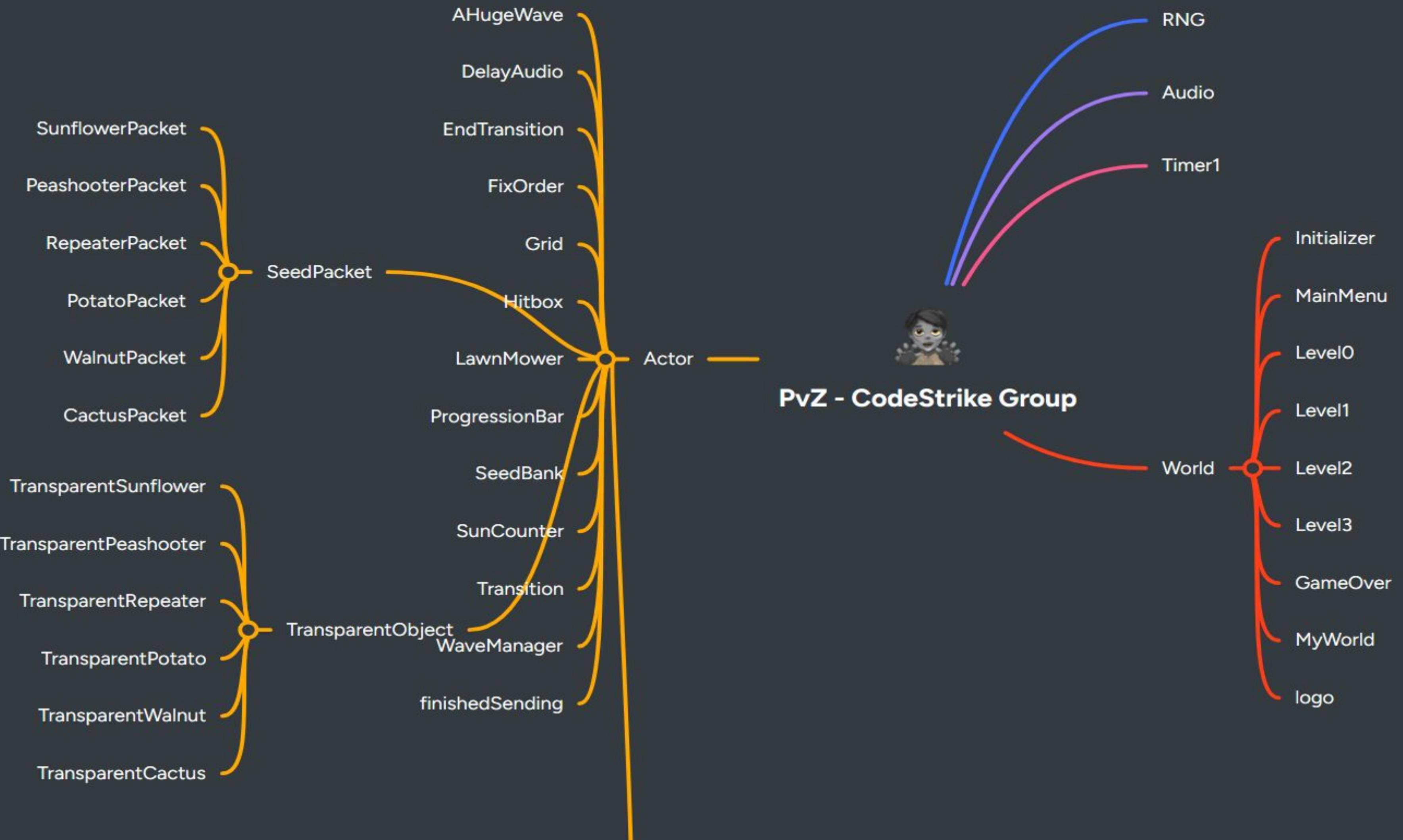
<https://www.youtube.com/watch?v=vKbh2-2IUVM>

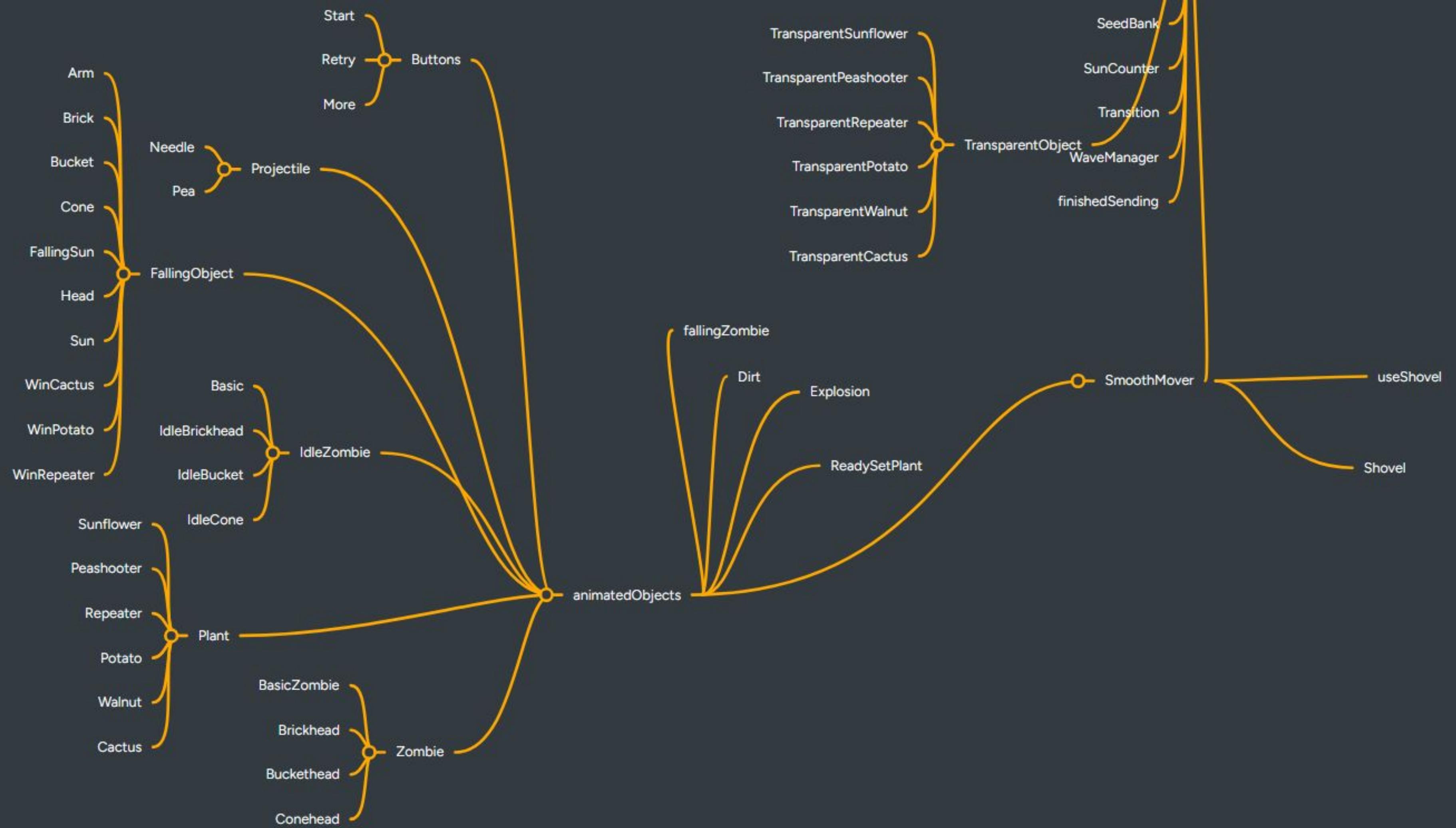


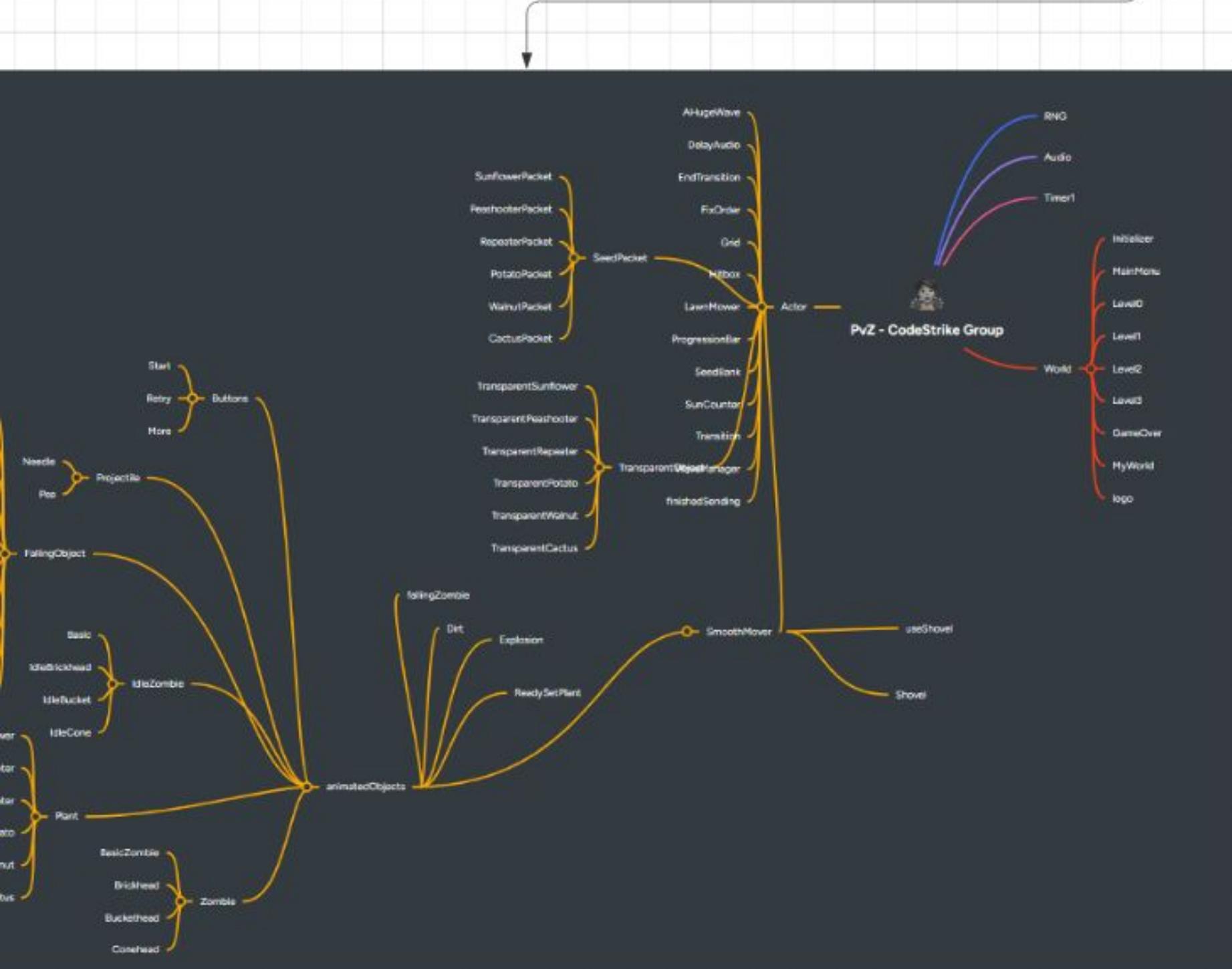
UML/Class Diagram











World

World is **the world that Actors live in**. It is a two-dimensional grid of cells. All Actor are associated with a World and can get access to the world object. The size of cells can be specified at world creation time, and is constant after creation.

Greenfoot
<https://www.greenfoot.org/files/javadoc/World.html>

- worldWidth : int
- worldHeight : int
- cellSize : int
- bounded : boolean
- background : GreenfootImage

```
+ World(worldWidth: int, worldHeight: int, cellSize: int)
+ World(worldWidth: int, worldHeight: int, cellSize: int, bounded: boolean)
+ addObject(object: Actor, x: int, y: int) : void
+ removeObject(object: Actor) : void
+ removeObjects(objects: Collection<? extends Actor>) : void
+ getWidth() : int
+ getHeight() : int
+ getCellSize() : int
+ getObjects() : List<Actor>
+ getObjects(cls: Class) : List<Actor>
+ showText(text: String, x: int, y: int) : void
+ setBackground(image: GreenfootImage) : void
+ setBackground(filename: String) : void
+ repaint() : void
+ act() : void
+ started() : void
+ stopped() : void
```

```

SeedBank
<<extends Actor>>

+ MyWorld: MyWorld
+ suncounter: SunCounter = new SunCounter()
+ bank: SeedPacket[]
+ selectedPacket: SeedPacket = null
+ image: TransparentObject = null
+ transparent: TransparentObject = null

+ static final x1: int = 182
+ static final x2: int = 702
+ static final xSpacing: int = Grid.xSpacing
+ static final y1: int = 62
+ static final y2: int = 417
+ static final ySpacing: int = Grid.ySpacing

+ SeedBank(bank: SeedPacket[])
+ act(): void
+ addedToWorld(world: World): void

```

```

LawnMower
<<extends Actor>>

- activated: boolean
- speed: int = 7
- lawn mowerImage: GreenfootImage

+ LawnMower()
+ act(): void
- activate(): void

```

```

ProgressBar
<<extends Actor>>

- level: WaveManager
- frameWidth: int = 458
- frameHeight: int = 34
- fillAreaWidth: int = 438
- fillAreaHeight: int = 14
- fillOffsetX: int = (frameWidth - fillAreaWidth) / 2
- fillOffsetY: int = (frameHeight - fillAreaHeight) / 2
- frame: GreenfootImage
- bar: GreenfootImage

+ ProgressBar(level: WaveManager)
+ act(): void

```

```

SeedPacket
<<extends Actor>>

+ deltaTime: long
+ deltaTime2: long
+ lastFrame: long = System.nanoTime()
+ lastFrame2: long = System.nanoTime()
+ rechargeTime: long
+ currentFrame: long = System.nanoTime()
+ sunCost: int
+ recharged: boolean = false
+ selected: boolean = false
+ doneRechargeTime: boolean = false
+ recharge: GreenfootImage
+ image1: GreenfootImage
+ image2: GreenfootImage
+ name: String
+ myWorld: MyWorld

+ SeedPacket(rechargeTime: long, recharged: boolean, sunCost: int, name: String)
+ addedToWorld(world: World): void
+ act(): void
+ handleRechargeProgress(): void
+ startRecharge(): void
+ setRecharged(charge: boolean): void
+ setSelected(selected: boolean): void
+ getCharge(): boolean
+ getSelected(): boolean
+ addImage(): TransparentObject
+ getPlant(): Plant

```

```

Actor

- image: GreenfootImage
- world: World
- x: int
- y: int

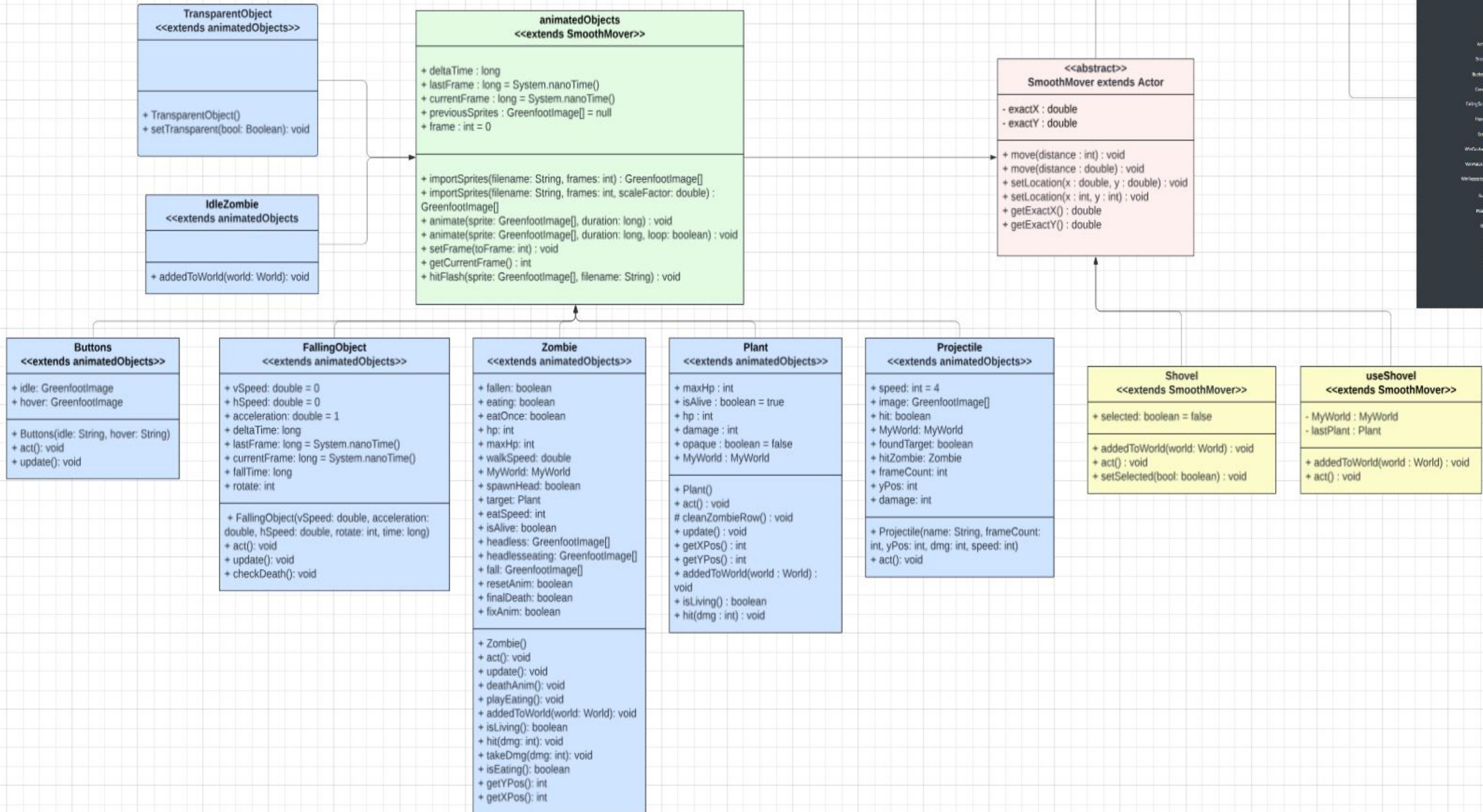
+ act(): void
+ getWorld(): World
+ setLocation(x: int, y: int): void
+ move(distance: int): void
+ turn(angle: int): void
+ getX(): int
+ getY(): int
+ getImage(): GreenfootImage
+ setImage(image: GreenfootImage): void
+ getRotation(): int
+ setRotation(angle: int): void
+ isAtEdge(): boolean

```

An Actor is an object that exists in the Greenfoot world. Every Actor has a location in the world, and an appearance (that is: an icon). An Actor is not normally instantiated, but instead used as a superclass to more specific objects in the world. Every object that is intended to appear in the world must extend Actor.

 Greenfoot
<https://www.greenfoot.org/files/javadoc/Actor.html>
Actor (greenfoot API)





GITHUB

Our github link: <https://github.com/trankhanhbinh/pvz-project-OOP.git>

The screenshot shows the GitHub repository interface for the 'pvz-project-OOP' repository. The 'game' folder is selected. The commit history for this folder is displayed, showing the following commits:

- Noobovich1 hotfix (Last commit message: merge)
- .. (Create classes and upload some images)
- doc (Upload images and sound effects and)
- images (Update)
- sounds (Update classes, Create other necessary)
- AHugeWave.class (clean code)
- AHugeWave.ctx (fix progress bar and lawn mower)
- AHugeWave.java (fix progress bar and lawn mower)
- Arm.class (fix)
- Arm.ctx (fix)
- Arm.java (fix)

The screenshot shows the README.md page for the 'pvz-project-OOP' repository. The page title is "Plants vs Zombies (PvZ) – CodeStrike Group Project". The content includes a welcome message, a table of contents, and an "About the Project" section.

Plants vs Zombies (PvZ) – CodeStrike Group Project

Welcome to the Plants vs Zombies project developed by the CodeStrike Group! Built using Java and Greenfoot, this project is a modern twist of the classic strategy game, aiming to deliver engaging gameplay with plans to expand themes and levels. Currently, the game features **four levels**, and we have exciting plans to introduce additional content in upcoming releases.

Table of Contents

- [About the Project](#)
- [Supported By](#)
- [Repository & Collaboration](#)
- [Inspirations](#)
- [Development Team](#)
- [Design & Documentation](#)
- [Media](#)
- [Future Plans](#)
- [Acknowledgments](#)

About the Project

This project is a reimagining of the popular Plants vs Zombies game, developed using Java with the support of the Greenfoot platform. Our goal is to create an engaging, strategy-filled game experience that builds on classic mechanics while incorporating unique thematic elements and dynamic levels.

**THANK
YOU!**