

420-SF3-RE PROGRAM DEVELOPMENT IN A GRAPHICAL ENVIRONMENT

ZOMBIED

Nagat Drawel

Gajjar, Zeel

Jain, Vedika

September 12, 2025

Task Description

Our application is called Zombied.

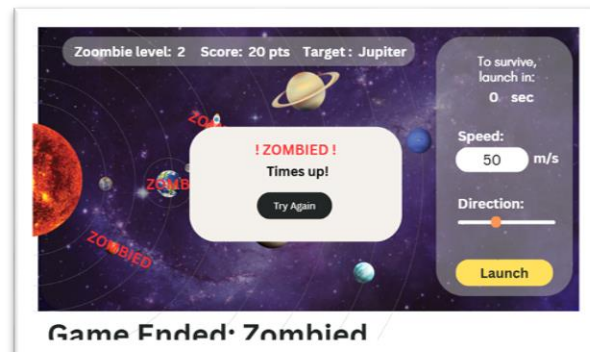
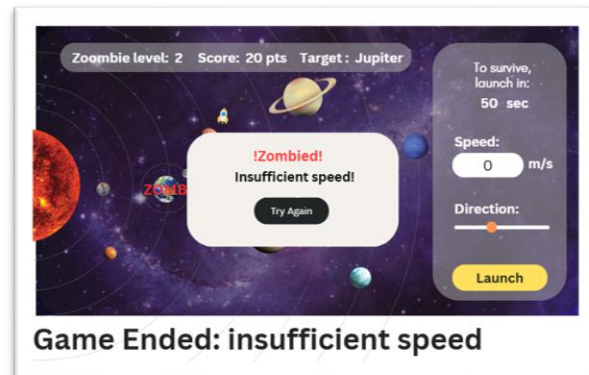
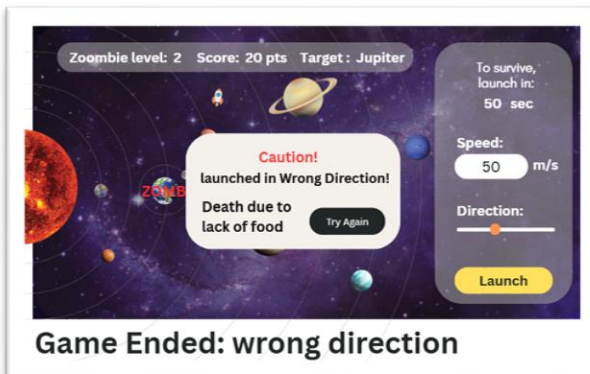
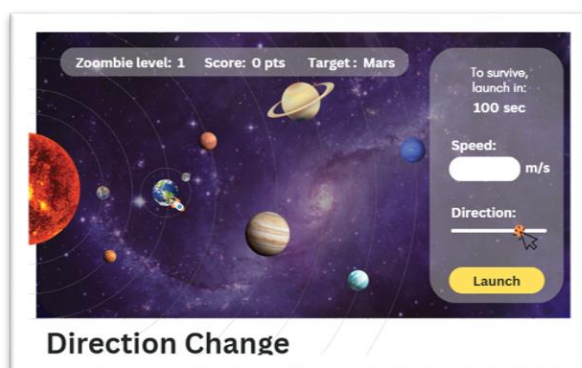
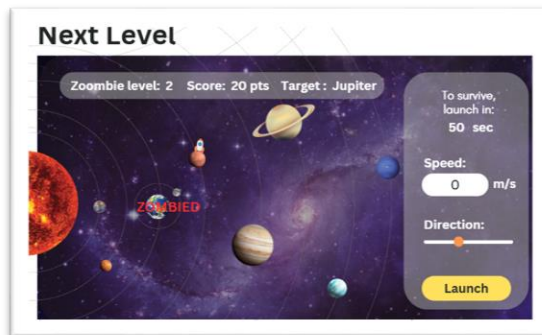
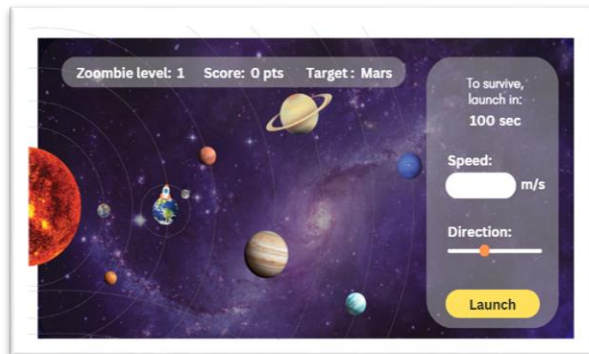
It's an educational game where players launch a projectile through space and try to hit a target planet. To do this, they set the speed and direction of a projectile, practicing basic physics concepts along the way.

Zombied has multiple levels and keeps score as players progress. After entering the values and clicking "Launch," the projectile travels according to the player's input.

- If it reaches the planet, the score and level go up.
- If it misses, the game explains why, helping players learn from the outcome.

With a space-themed background and planet targets, Zombied turns physics practice into an engaging simulation.

Interface Visualizations



Proposed Implementation Approach

1. Programming Language and Framework

- **Language:** Java
- **Framework:** JavaFX (for GUI, animation, and event handling)

2. Libraries, APIs, and Tools

- **JavaFX:** For GUI components, 2D graphics, animation, and user interaction.
- **JavaFX AnimationTimer/Timeline:** For rocket movement and countdown timer.
- **JavaFX Media:** For adding simple sound effects (launch, explosion, zombie warning).
- **IDE:** NetBeans or IntelliJ IDEA (for development).
- **Version Control:** GitHub/Git for version tracking and collaboration.
- **Organization:** Trello for organizing the tasks among team members.

3. Project Structure

- **Main Application**
 - ZombiedApp → Entry point, initializes JavaFX application, sets up scenes.
- **Controllers**
 - MenuController → Handles the main menu (start, instructions, exit).
 - GameController → Manages gameplay logic, user input (speed/direction), and launching projectiles.
 - ResultController → Displays outcomes, explanations, and learning feedback.
 - SoundController → Plays sound effects (launch, hit, miss).
- **Models**
 - Projectile → Stores physics values (speed, angle), calculates trajectory.
 - Planet → Represents the target with position and hit detection.
 - GameState → Tracks score, level progression, and attempts.
- **Utilities**
 - PhysicsUtil → Contains methods for projectile motion and trajectory calculations.
 - CollisionUtil → Determines whether a projectile collides with a planet.
- **Views (FXML Layouts)**
 - menu.fxml → Menu screen.
 - game.fxml → Main gameplay interface.
 - result.fxml → Feedback and learning screen.

Trello Link:

<https://trello.com/invite/b/68c30d964704cec08b1ecd08/ATTI7396e035fbf17e8f70d5e5f886a14a40226D801D/sem-3-programmingfinalproject>

GitHub Link:

https://github.com/ZeelDGajjar/Sem3-Final_Project.git