**Software Requirements and Design Document**

**For**

**Group <17>**

Version 2.0

**Authors**:

Aamir Baloch

Blake Barth

Jacob Copham

Joshua Krug

Ethan Lin

# Overview (5 points)

Our “system” is a single-player video game that combines elements of tower defense, platforming, puzzles, and resource management. Players navigate through a cyberspace-themed world, where they face waves of enemies attacking their base. The core mechanics involve defending the base by strategically placing and upgrading defenses while solving science-based puzzles to unlock new upgrades and abilities. The game also incorporates physics-based movement, allowing for engaging platforming challenges as players explore the world and gather randomly seeded resources.

The narrative unfolds as players progress, revealing an intriguing and mysterious story that adds depth to the gameplay. Resource management is key, as players must collect and utilize resources to improve their defenses and solve puzzles. The game’s aesthetic, driven by its cyberspace theme, is reinforced through dynamic music and futuristic visual designs, creating an immersive experience that balances fast-paced defense with thoughtful puzzle-solving.

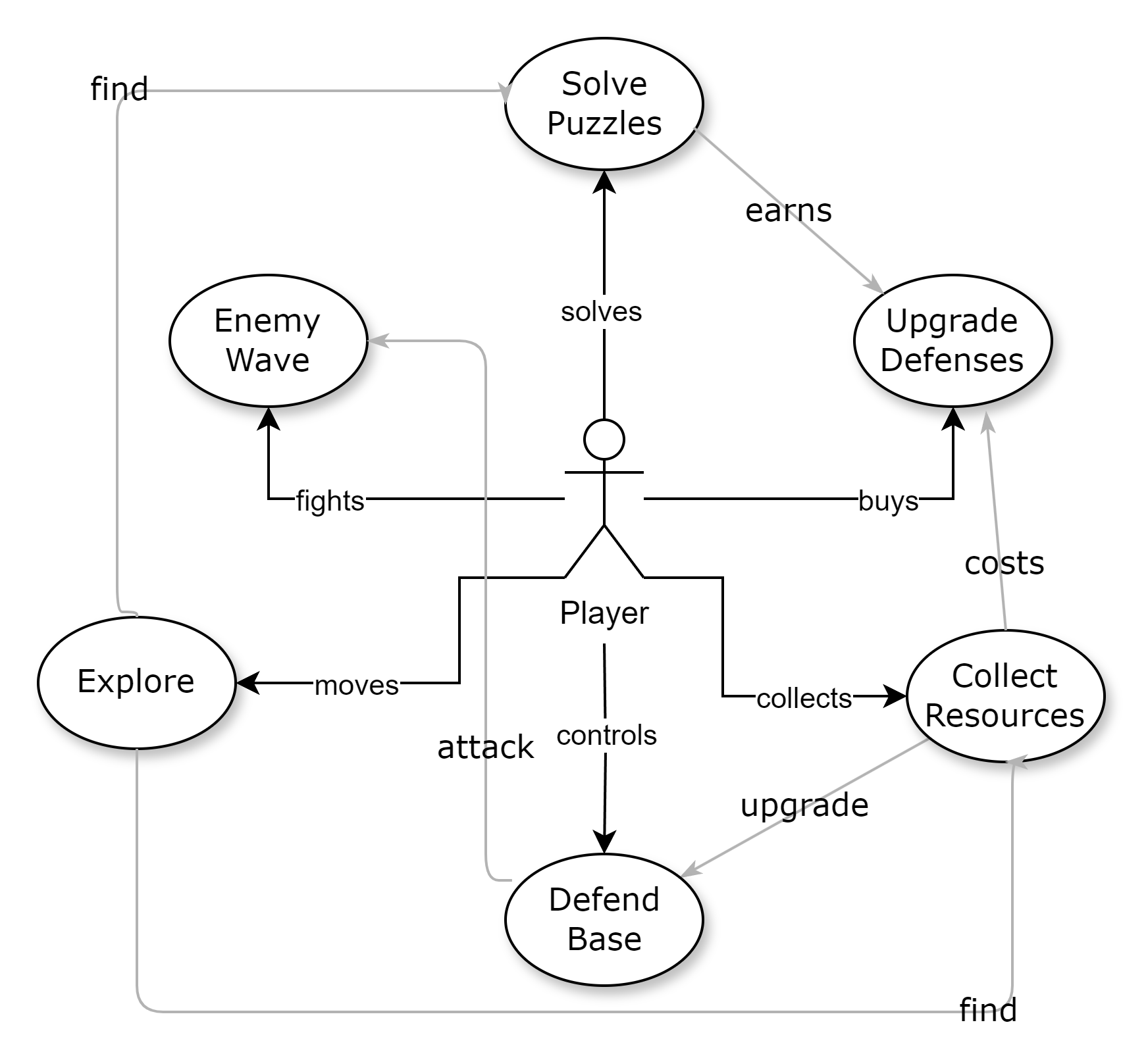
# Functional Requirements (10 points)

1. The system shall provide physics-based movement mechanics for the player.  
   Priority: High  
   This is essential for platforming and resource exploration, contributing to the core gameplay experience.
2. The system shall implement randomly seeded resources in the game world, allowing players to explore and gather items.  
   Priority: High  
   This supports the exploration and resource management aspects, ensuring it can be replayed through randomized resource locations.
3. The system shall generate waves of enemies that attack the player’s base at timed intervals.  
   Priority: High  
   Core to the tower defense aspect, defending against enemy waves is a key part of the game loop.
4. The system shall allow players to set up and upgrade defensive structures to protect their base from enemy waves.  
   Priority: High  
   Defensive strategies and upgrades are fundamental to tower defense mechanics.
5. The system shall implement science-based puzzles that players must solve to unlock new upgrades and progress through the game.  
   Priority: Medium  
   These puzzles add variety and depth to the gameplay, offering players rewards for critical thinking.
6. The system shall include unique upgrades based on solving puzzles that alter gameplay mechanics or provide significant advantages in combat or resource management.  
   Priority: Medium  
   These upgrades motivate players to engage with the puzzle mechanics and create progression paths.
7. The system shall integrate a cyberspace theme with music and visual designs that fit the aesthetic of a futuristic digital world.  
   Priority: High  
   A consistent theme is crucial for immersion and reinforcing the narrative.
8. The system shall support resource management mechanics, including collecting, storing, and utilizing resources to upgrade defenses and solve puzzles.  
   Priority: High  
   This feature ties the various gameplay elements together, as resources are central to both defense and puzzle-solving.
9. The system shall feature dynamic enemy AI that increases in difficulty as the player progresses through waves.  
   Priority: Medium  
   Dynamic AI creates a more challenging and engaging experience as the game advances.
10. The system shall include a narrative-driven progression, revealing the mysterious story through exploration, puzzles, and defense stages.  
    Priority: Medium  
    The narrative gives context and meaning to the player’s actions, enhancing the overall game experience.
11. The system shall provide a save and load feature, allowing players to resume their progress later.  
    Priority: High  
    A basic quality-of-life feature necessary for longer play sessions and user convenience.
12. The system shall include a tutorial to guide players through the basic mechanics of physics-based movement, puzzle-solving, and resource management.  
    Priority: Low  
    Although not essential for experienced players, a tutorial improves accessibility for new players.

# Non-functional Requirements (10 points)

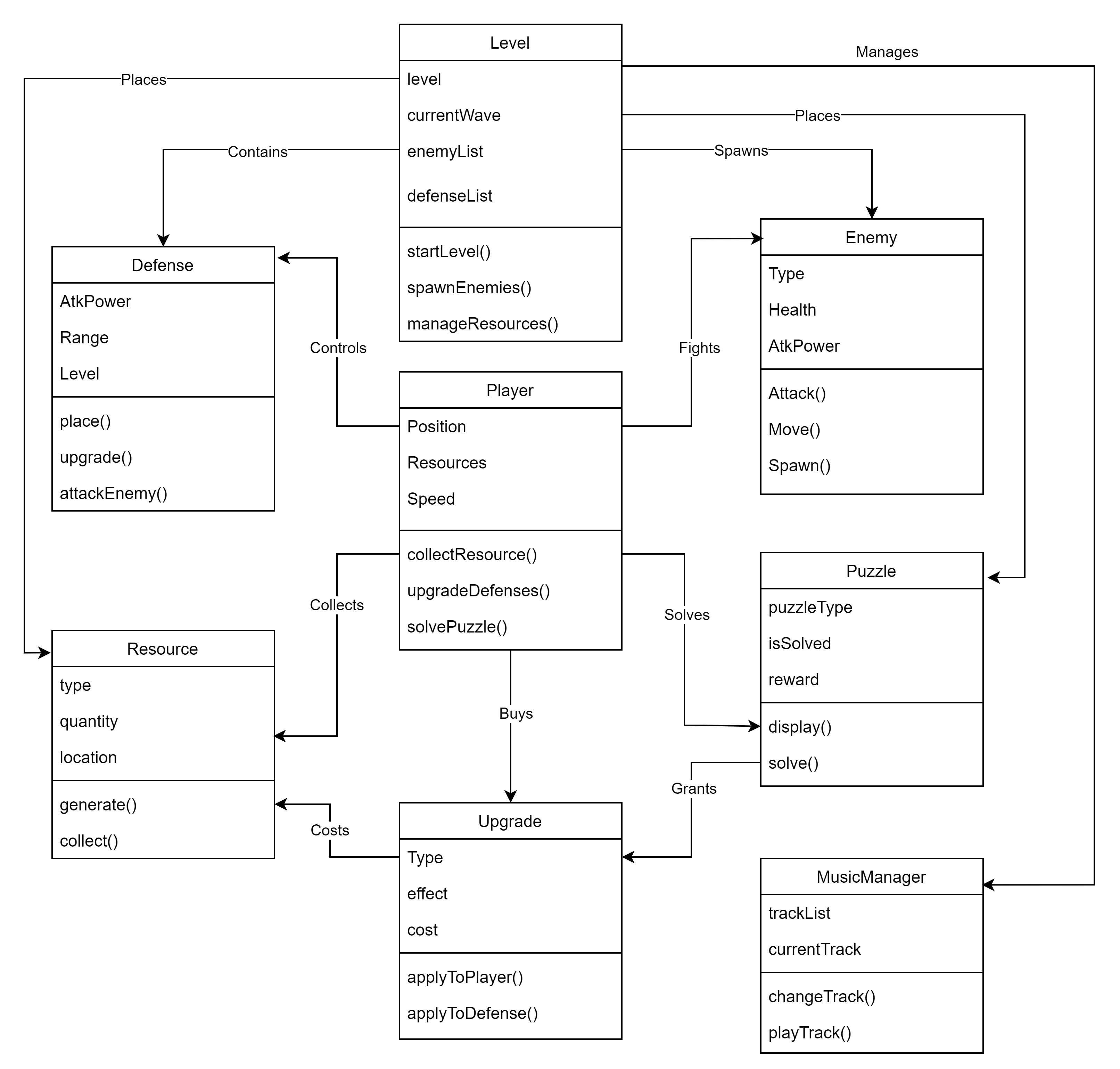
Our main nonfunctional requirement is that the game can run on a variety of browsers. As long as the browser supports the canvas tag our game should be designed to run on it. We require no more than the requirements of Phaser JS.

# Use Case Diagram (10 points)

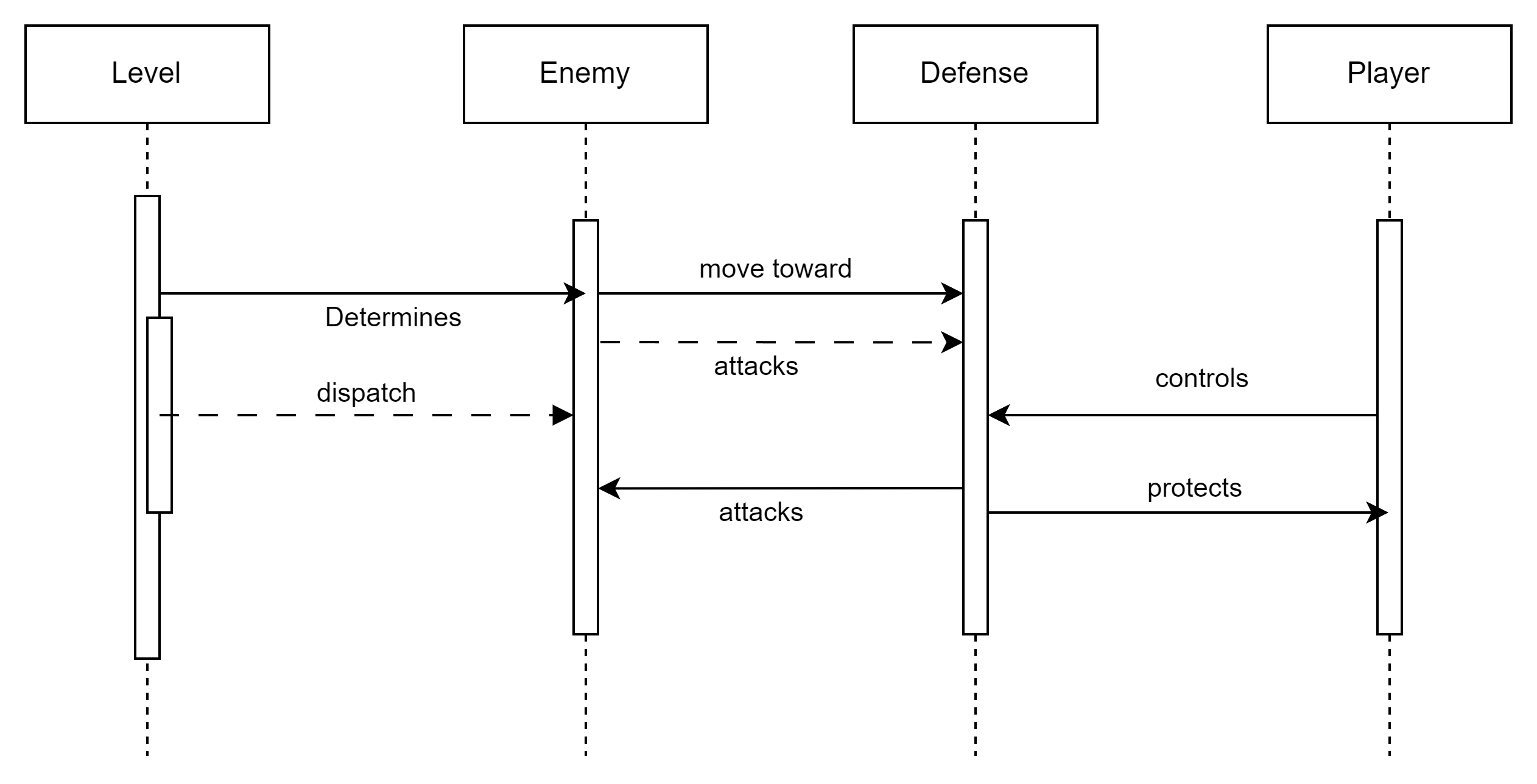


# Class Diagram and/or Sequence Diagrams (15 points)

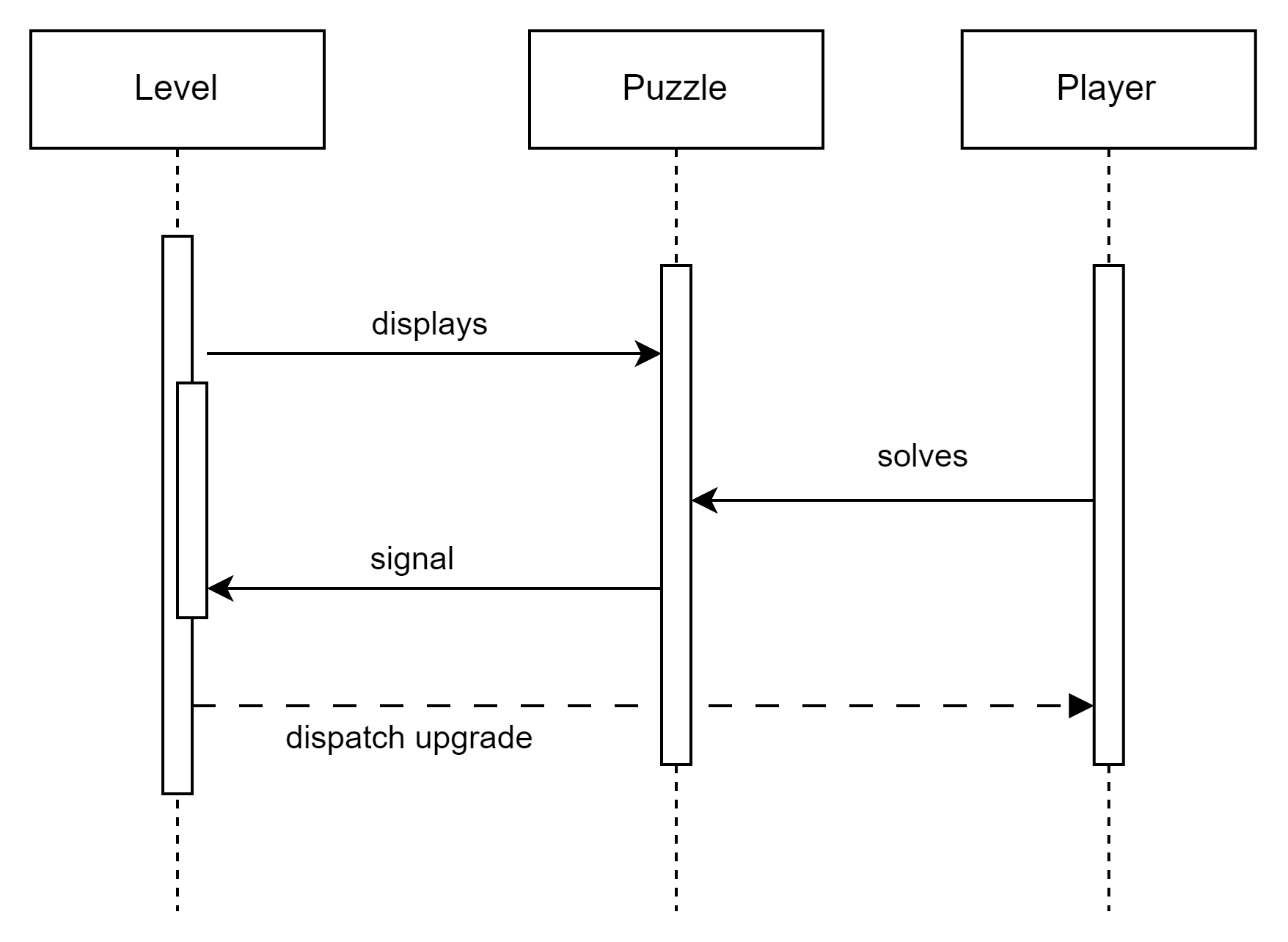
Class Diagram:



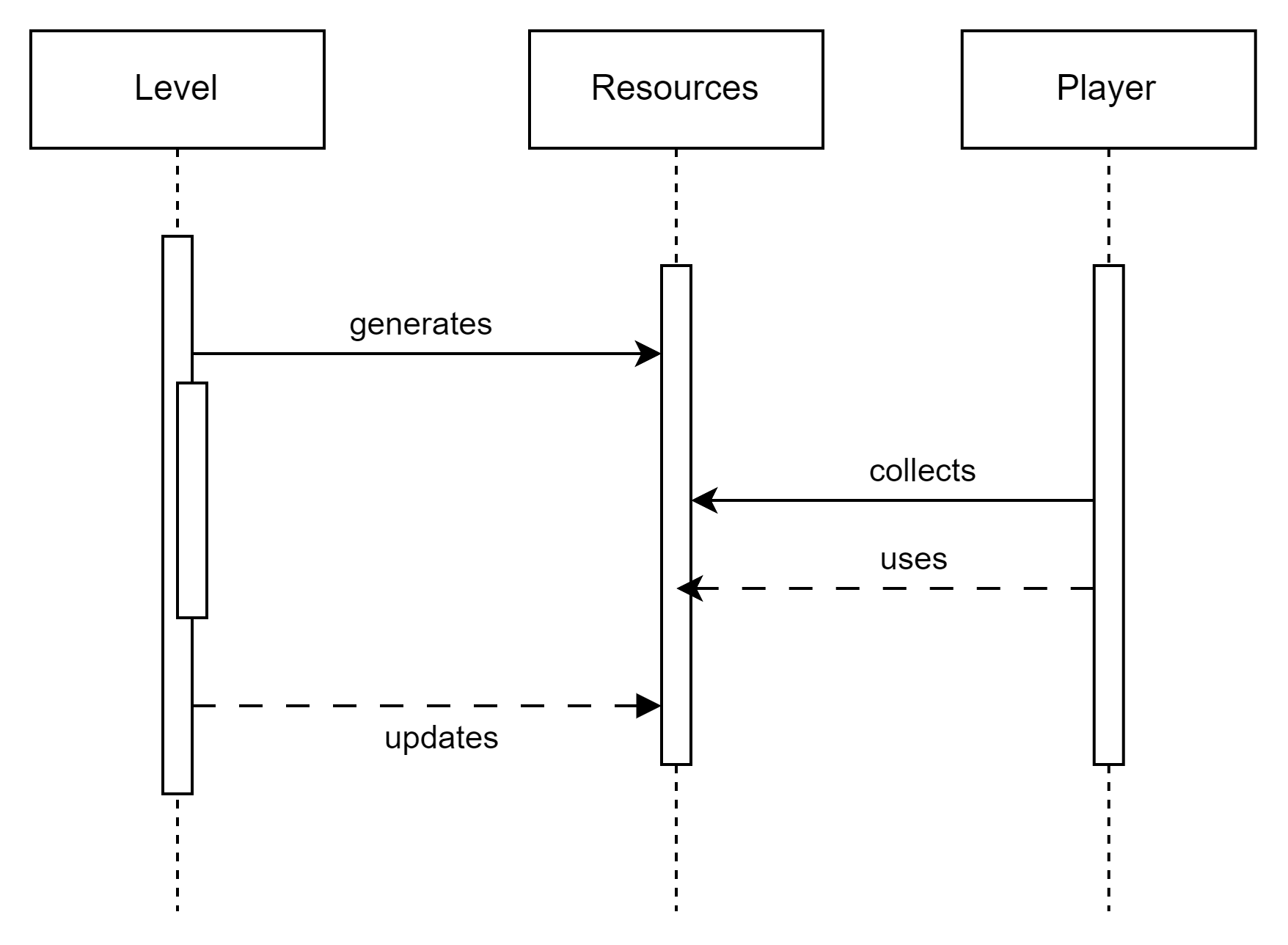
Sequence Diagram: Use Case: Defending Against Enemy Wave



Sequence Diagram: Use Case: Solving a Puzzle



Sequence Diagram: Use Case: Collecting and Using Resources



# Operating Environment (5 points)

The software will operate in a web-based environment, designed to be accessible on modern desktop and laptop devices via web browsers. The game should perform well on devices with at least 4GB of RAM, a mid-range processor, and basic integrated or dedicated graphics cards.

The game will run on the latest versions of major web browsers that support WebGL, HTML5, and JavaScript, specifically Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari. The game is designed to run directly in a browser without requiring additional plugins or external applications. However, network connectivity may be required if the game needs to download certain assets dynamically during gameplay, though this feature depends on the final implementation.

# Assumptions and Dependencies (5 points)

Assumptions:

1. **Third-Party Components**: We assume that Phaser.js libraries, plugins, and other third-party assets from online repositories will be available and compatible with the version of Phaser.js being used. If these assets become unavailable or have compatibility issues, game development may be delayed or require alternative assets.
2. **Browser Compatibility**: The game is being developed with the assumption that modern web browsers will fully support Phaser.js. Any changes to browser functionality, JavaScript performance, or security features could affect gameplay or overall performance.

Dependencies:

1. **Phaser.js Framework**: The game is dependent on the Phaser.js framework for rendering, physics-based movement, and game mechanics. Any issues with Phaser.js updates or bugs could impact the implementation of core features like physics interactions or enemy behavior, requiring workarounds or additional development time.
2. **JavaScript Engine**: The game relies on the performance and stability of the JavaScript engine in the player's web browser. Any significant differences in the way browsers interpret JavaScript (for instance, varying frame rates or input lag) could affect the user experience, requiring additional testing and optimization for cross-browser compatibility.
3. **Asset Reuse**: We are dependent on external graphic assets, sound effects, and music that we plan to reuse from free or purchased sources. If licensing terms change or the assets become unavailable, we will need to find replacements, which could delay development or affect the game’s aesthetics.
4. **Physics and Animation Libraries**: The game depends on the Phaser.js framework’s built-in libraries for animations and physics. Any changes in these libraries or limitations in Phaser.js could affect the quality and consistency of movement mechanics, enemy interactions, or puzzle physics.