Project Proposal: Cat Adventure Game

Marnilla Metwaly
Andrew ID: mmetwaly

November 19, 2024

Contents

Acknowledgment

I would like to acknowledge that ChatGPT has assisted me in formatting and refining the structure of this report, as well as helping with the LaTeX code for writing equations and improving the overall presentation of the document. I want to verify that all thoughts and Database organization are mine.

1 Project Description

The project is titled **"Cat Adventure Game"**, a platform game where the player controls a cat character to collect coins or fish while avoiding enemies, such as dogs. The game will feature multiple levels with varied environments, including forests and seas. Players can choose between different cat heroes and enjoy unique power-ups, such as double jumping, a coin magnet, and a protective shield. The game is designed to be engaging, fun, and unique by adding a cat-themed twist to traditional platform game mechanics.

2 Competitive Analysis

Platform games such as **"Super Mario Bros."** and **"Sonic the Hedgehog"** have inspired this project. These games are renowned for their side-scrolling mechanics, engaging visuals, and simple but addictive gameplay. While the **"Cat Adventure Game"** takes inspiration from these classics, it introduces unique elements that set it apart.

First, the game focuses on cat-themed mechanics, such as double jumping for agility and a coin magnet, catering to players who love cats. Additionally, the ability to choose a hero and use creative power-ups adds an innovative layer to gameplay. Unlike most traditional platform games, which focus primarily on linear challenges, this game emphasizes exploration and customization through varied environments and cat-specific features.

3 Structural Plan

The project will be divided into modules, functions, and objects as follows:

1. **Main File**: Handles game initialization, screen rendering, and overall game logic.
2. **Hero Class**: Manages the cat's movements, power-ups, collision detection, and animations.
3. **Enemy Class**: Includes behaviors for different types of enemies, such as walking and randomizing their direction.
4. **Platform Class**: Creates platforms for the cat to traverse, including collision detection.
5. **Collectible Class**: Handles coins and fish that the player collects.
6. **Environment Module**: Generates level-specific environments, such as backgrounds, obstacles, and decorative objects.
7. **Power-Up System**: Implements special abilities (e.g., double jumping, shield, magnet).
8. **Utilities Module**: Contains helper functions for drawing and collision detection.

4 Algorithmic Plan

The **trickiest part** of this project will be implementing the **power-ups**. Here's the detailed algorithm for the **coin magnet**:

1. **Activation**:
   (a) When the coin magnet is activated, iterate through all collectibles.

   (b) For each collectible, calculate the distance between the cat and the collectible.
   (c) If the distance is less than a predefined threshold, adjust the collectible's position incrementally toward the cat's position, simulating a magnetic pull.

2. **Update**:
   (a) Update the collectible's position each frame until collected or the magnet's effect expires.

3. **Deactivation**:
   (a) Deactivate the magnet after a fixed duration, resetting collectible behavior to normal.

This will require efficient distance calculations and smooth animations for a seamless experience.

5 Timeline Plan

Week 1:

- Design environments for different levels.
- Implement hero selection and basic hero movement.
- Create basic platform generation logic.

Week 2:

- Add power-ups (double jump, magnet, shield) with corresponding methods in the Hero class.
- Develop enemy behavior and collision mechanics.

6 Version Control Plan

To ensure the code is backed up and version-controlled:

- I will use **Google Drive** for regular backups of the project files.
- Additionally, I will use **Git** with a repository on **GitHub** for version control.
  - Commits will be made after completing each feature or module to track progress and revert to previous versions if needed.

Here is the backup flow:

```
Develop locally --> Commit changes to GitHub --> Sync with Google Drive
```

7 Module List

The project will use the following modules and technologies:

- **cmu_graphics**: For drawing and rendering game elements.
- **random**: For generating randomized elements like platform gaps and collectible placements.
- **math**: For distance calculations and angles in the game physics.
- **PIL (Optional)**: For processing images to incorporate real cat characters.

8 Preliminary Code

Below is an early snippet of the Hero class, which manages the cat's movements and power-ups:

```
class Hero(Sprite):
    """
    Represents the player's character.
    """
    def __init__(self, x, y):
        super().__init__(x, y, color="gray")
        self.speed = 5  # Movement speed
        self.jumpStrength = -15  # Jump velocity
        self.lives = 3  # Number of lives
        self.score = 0  # Player's score
        self.currentImageIndex = 0  # For animation frames
        self.stepsPerImage = 5  # Controls animation speed
        self.stepsSinceLastImage = 0  # Counts steps for animation
        self.powerUps = {
            'double_jump': False,
            'coin_magnet': False,
            'shield': False
        }
        self.powerUpTimers = {
            'double_jump': 0,
            'coin_magnet': 0,
            'shield': 0
        }

    def activatePowerUp(self, powerUpType, duration):
        """
        Activates a power-up and sets its timer.
        """
        if powerUpType in self.powerUps:
            self.powerUps[powerUpType] = True
            self.powerUpTimers[powerUpType] = duration

    def deactivatePowerUp(self, powerUpType):
        """
        Deactivates a power-up.
        """
        if powerUpType in self.powerUps:
            self.powerUps[powerUpType] = False
            self.powerUpTimers[powerUpType] = 0

    def onStep(self, app):
        # Update position and check collisions
        super().onStep(app)
        # Update animation frame
        self.updateAnimation()
```

```
        # Update power-up timers
        for powerUp, timer in self.powerUpTimers.items():
            if timer > 0:
                self.powerUpTimers[powerUp] -= 1
                if self.powerUpTimers[powerUp] == 0:
                    self.deactivatePowerUp(powerUp)
```

This code sets the foundation for managing power-ups, allowing the hero to activate and deactivate abilities like double jumping and coin magnets.