Tomato Platformer - Beginner Friendly Documentation

# 1. Introduction

Tomato Platformer is a 2D platformer game made using Python and the Pygame library. In this game, you control a tomato character that can run, jump, collect coins, defeat enemies, and reach the goal to complete levels. This document explains the code step by step, in a way that even beginners can understand.

# 2. How the Code Works

The game code is divided into smaller parts called functions and classes. Each part has its own job, and together they make the game work.

## 2.1 Constants and Colors

At the beginning of the code, we define constants like SCREEN\_WIDTH, SCREEN\_HEIGHT, GRAVITY, PLAYER\_JUMP, and PLAYER\_SPEED. These are values we use again and again, so instead of writing numbers everywhere, we give them names.  
  
We also define colors like BLACK, WHITE, RED, GREEN, and SKY\_BLUE. These are stored as RGB values.

## 2.2 Helper Functions

- get\_image: Takes a small part of a sprite sheet (a big image with many frames) and returns it as a separate image for animation.  
- load\_scores and save\_scores: These functions read and write the best scores to a file called scores.json. This way, even when you close the game, your best scores are saved.

## 2.3 Classes

Classes are like blueprints to create objects in the game. Each class represents something in the game.

- Platform: Creates the ground or blocks that the player can stand on.  
- MovingPlatform: A special platform that moves left and right.  
- Hazard: Creates dangerous spikes. If the player touches them, they lose.  
- Collectible: These are the coins the player collects for points.  
- Enemy: Red square enemies that move back and forth. If the player jumps on them, they are defeated.  
- Goal: The flag that marks the end of the level.  
- Player: The tomato character you control. It handles movement, jumping, gravity, and animations.  
- Level: A container that holds platforms, enemies, hazards, coins, and the goal for each stage.

## 2.4 Game Flow

The game has different screens and states:  
- main\_menu: Shows the title screen with Start and Quit buttons.  
- level\_select\_screen: Lets the player choose a level to play. Shows best scores.  
- game\_screen: Runs the main game loop where the player controls the tomato.  
- pause\_screen: Freezes the game with Resume and Main Menu options.  
- reset\_level: Restarts the level whenever the player loses.  
  
All of this is managed by the main() function, which keeps switching between screens depending on what the player does.

# 3. Development Challenges and Solutions

When making the game, we faced some problems. Here are the main ones and how we solved them:

- Sprite Cropping: At first, the tomato sprite looked broken. We fixed it by correcting the coordinates in get\_image.  
- Enemies Falling: Enemies were falling off platforms. We added gravity and edge detection to Enemy.  
- Score Not Resetting: The score didn’t reset properly when the player died. We created reset\_level to fix this.  
- Level 4 Problems: Platforms and hazards were placed badly. We fixed their positions by testing and adjusting.  
- Packaging Errors: PyInstaller couldn’t find images. We fixed this by using a helper function to find the correct path.

# 4. Future Enhancements

- Add power-ups like shields or speed boosts.  
- New enemies with flying or shooting abilities.  
- Add a health/lives system so the game is less punishing.  
- Create more levels with harder challenges.  
- Add sound effects and background music.

# 5. Big Picture

In simple words, the game is like a story:  
- The player (tomato) is the hero.  
- Platforms are the roads.  
- Enemies and hazards are the dangers.  
- Coins are rewards.  
- The goal is the finish line.  
  
The code is like a director of a movie. Each function and class is like an actor with a specific role. Together, they create the full gameplay experience.