

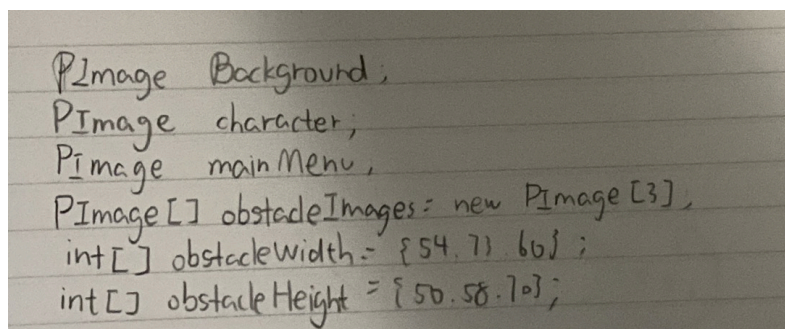
Implementation Log

1. Game Overview

This game is a simple action game based on jumping and avoiding obstacles. The player controls a character and jumps to avoid obstacles that keep appearing. Scoring one point is achieved by jumping over an obstacle. The game includes basic gravity, jump mechanics, obstacle generation, and a scoring system. The game ends when the player hits an obstacle, and the player can press R to restart the game.

2. Main Variables and save image

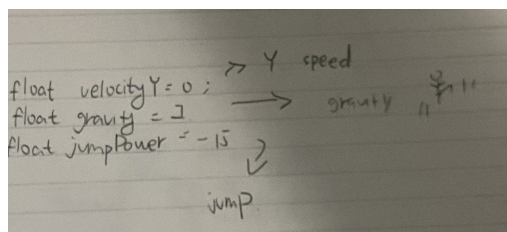
The key variables in the game include the position of the character, background, obstacles, and coins. These variables are defined as PImage type or numerical types for dynamic adjustment in the game. In the setup() method, the game canvas, background, character, and obstacle images are initialized, and the game state is reset using the resetGame() method.



```
PImage Background;  
PImage character;  
PImage mainMenu;  
PImage[] obstacleImages = new PImage[3];  
int[] obstacleWidth = {54, 71, 60};  
int[] obstacleHeight = {50, 58, 70};
```

3. Game Logic

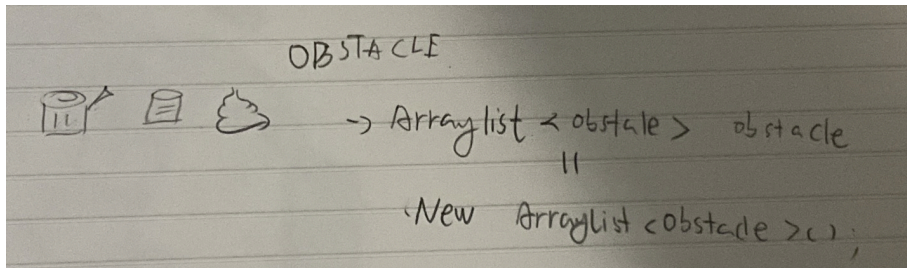
During the game, the player's character is affected by gravity and moves vertically on the screen. The character jumps by pressing the spacebar, and the power of the jump is controlled by the jumpPower variable. After each jump, the character's vertical speed is affected by gravity (gravity). The player can jump a maximum of two times in a row (maxJumpCount), and the jump count is reset every time the character touches the ground.



```
float velocityY = 0; // Y speed  
float gravity = 1; // gravity  
float jumpPower = -15; // jump
```

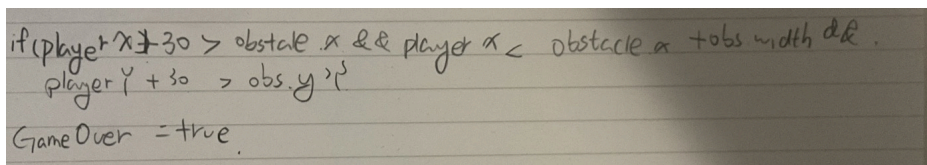
4. Obstacles

The obstacles in the game are implemented using the Obstacle class, and their position, width, and height will change continuously as the game progresses. When an obstacle completely leaves the screen, the player's score increases, and new obstacles will appear on the screen.



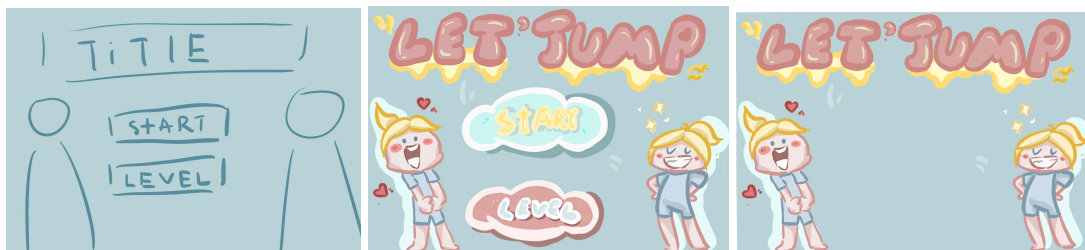
5. Game Over and Replay

When a player character collides with an obstacle, the game ends. At this point, the "Game Over" prompt is displayed, and the player can press R to restart the game. When the game is restarted, all variables will be reset to their initial values, and the game will restart.



6. Main Menu

When the game starts, a main menu is displayed, and players can start the game by clicking the "Start" button. In the `mousePressed()` method, the player's click position is detected, and if the click is within the range of the start button, the game screen is entered. I originally designed levels and start screens, but my ability was not sufficient to implement them, so I cancelled the level and start screen design and replaced it with a start button.



7. Character Design

I based my character design on myself, creating a jumping girl. Then pressing the space bar controls the character's jump.



8. Problems encountered in design

I had no problems importing photos into the game. I initially created a rough outline for the game without importing photos, using simple graphics to test whether my game idea could be implemented. However, I had no problems with the background, cover, or characters. But because I initially designed random obstacles, and I also designed three different types of obstacles. I kept trying to import my images, but I kept failing. I spent two days debugging. Finally, I asked my tutor, who was giving me private lessons, where my problem was. So I made the necessary changes.

9. Summary and outlook

Although this game is simple, it covers many basic elements of game development, including image loading, collision detection, scoring system, and game reset. Because time was tight this time, but the game can be enriched and made more challenging and interesting in the future by adding more game elements. But in fact, when I tried the game myself, it seemed simple, but it was very difficult for me to get a score of 10.