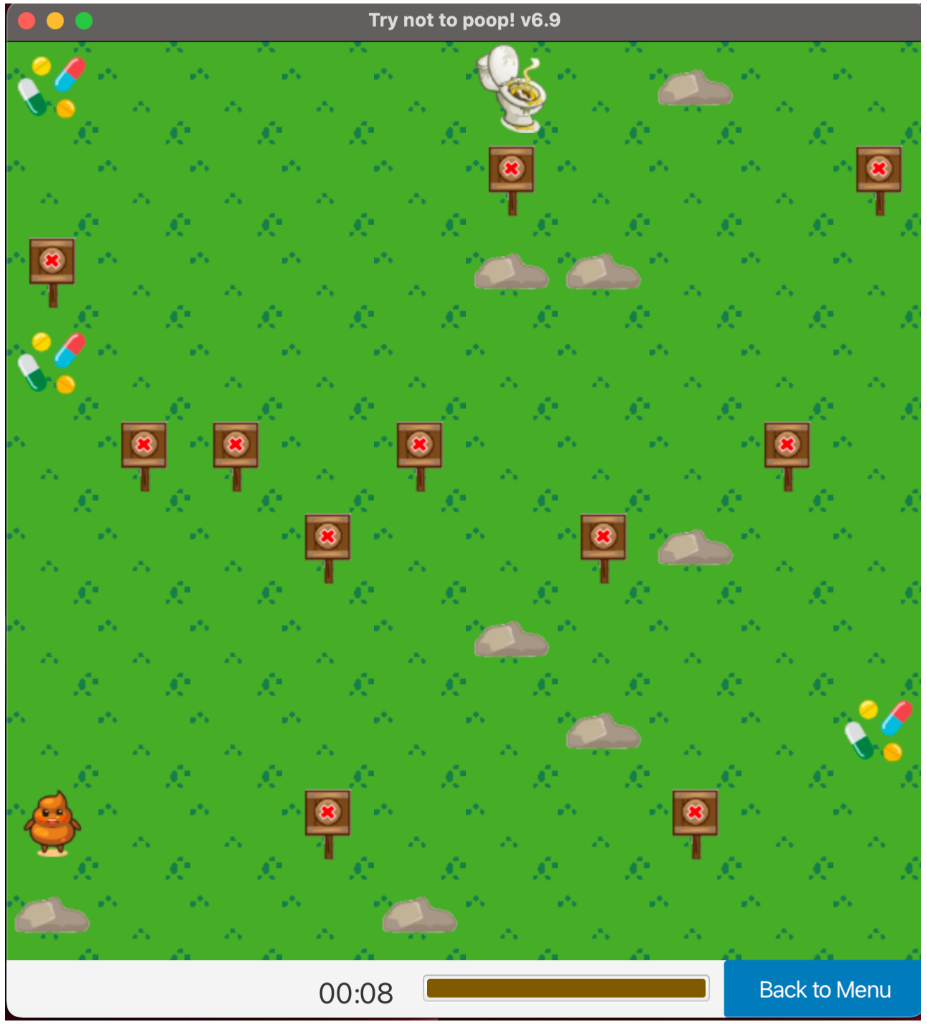


Try Not To Poop simulator (2D)

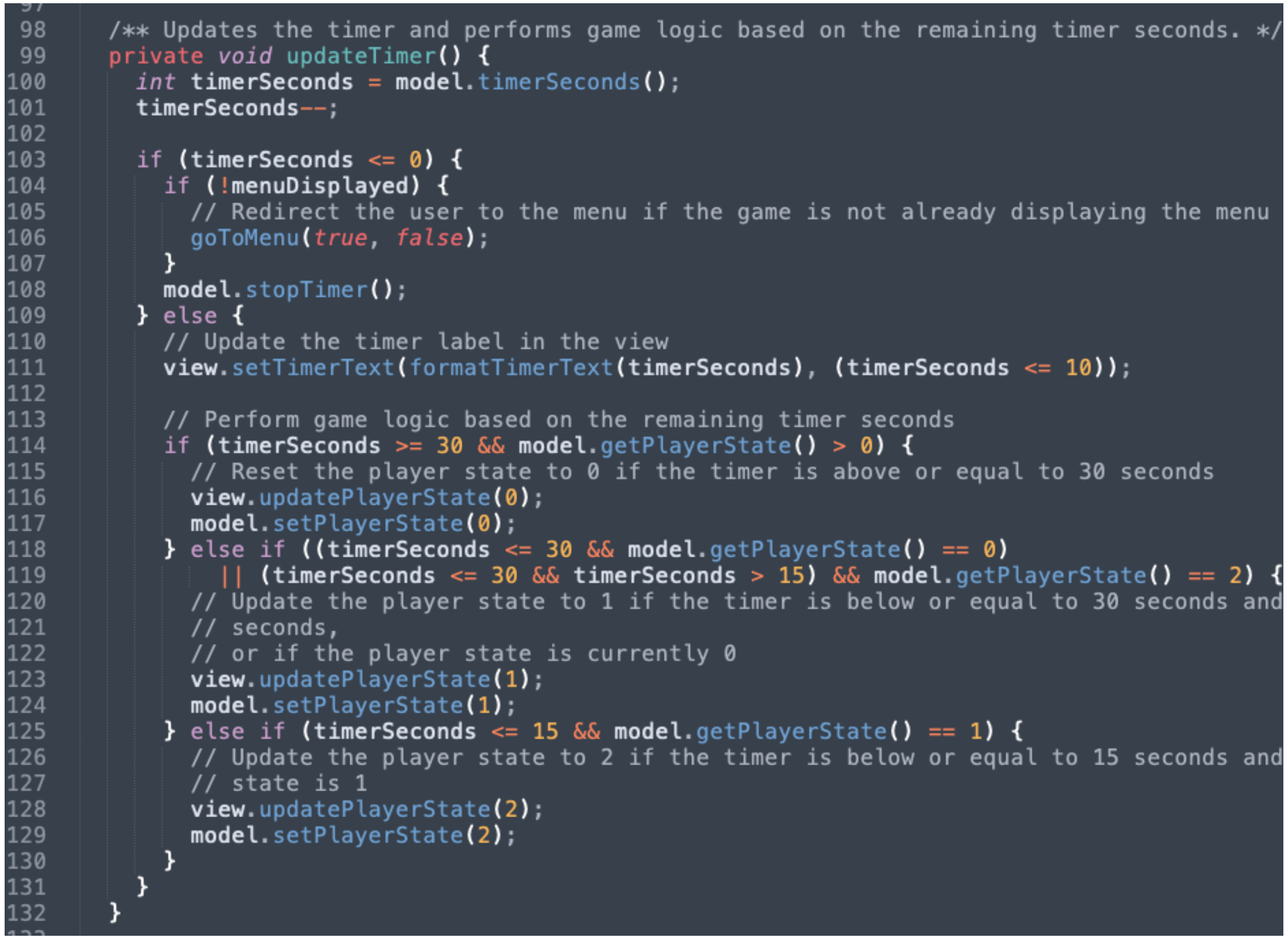
STEINBACH Matteo

LE DREZEN Aymeric

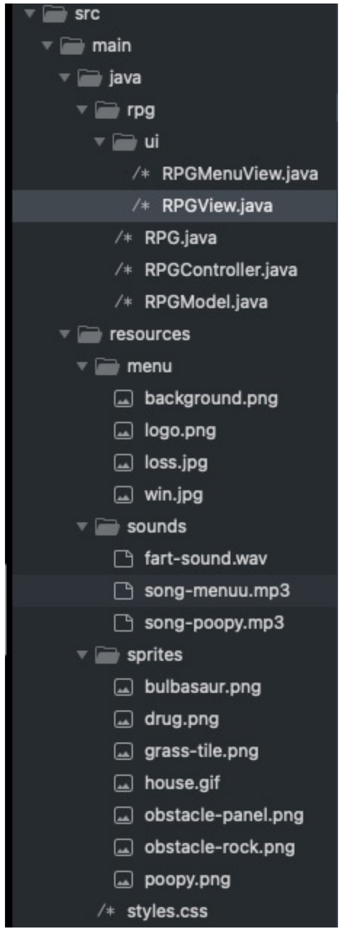
JACK Oliver

Timer and Game Progression

- Timer
  - Tracks the literal progression of the game
  - Bonus and Maluses affect game timer
- Game progression
  - Losing when there is no time left or poopy poops is pants
  - Winning when poopy reach the toilet



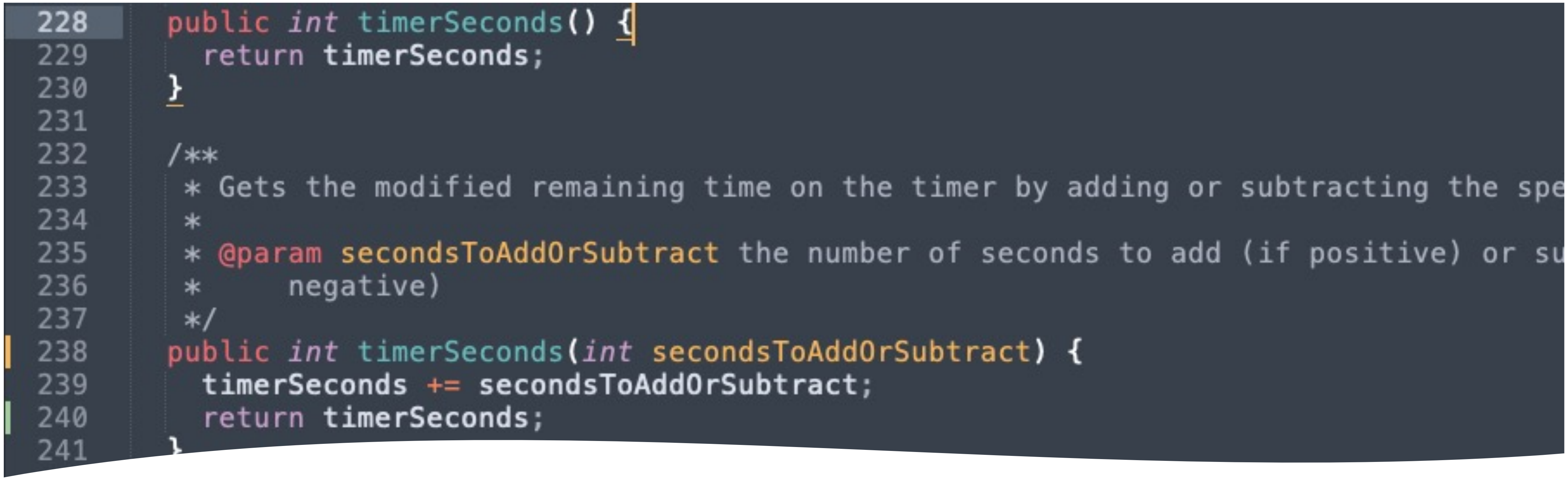
```
/** Updates the timer and performs game logic based on the remaining timer seconds. */
private void updateTimer() {
  int timerSeconds = model.timerSeconds();
  timerSeconds--;

  if (timerSeconds <= 0) {
    if (!menuDisplayed) {
      // Redirect the user to the menu if the game is not already displaying the menu
      goToMenu(true, false);
    }
    model.stopTimer();
  } else {
    // Update the timer label in the view
    view.setTimerText(formatTimerText(timerSeconds), (timerSeconds <= 10));

    // Perform game logic based on the remaining timer seconds
    if (timerSeconds >= 30 && model.getPlayerState() > 0) {
      // Reset the player state to 0 if the timer is above or equal to 30 seconds
      view.updatePlayerState(0);
      model.setPlayerState(0);
    } else if ((timerSeconds <= 30 && model.getPlayerState() == 0)
        || (timerSeconds <= 30 && timerSeconds > 15) && model.getPlayerState() == 2) {
      // Update the player state to 1 if the timer is below or equal to 30 seconds and
      // seconds,
      // or if the player state is currently 0
      view.updatePlayerState(1);
      model.setPlayerState(1);
    } else if (timerSeconds <= 15 && model.getPlayerState() == 1) {
      // Update the player state to 2 if the timer is below or equal to 15 seconds and
      // state is 1
      view.updatePlayerState(2);
      model.setPlayerState(2);
    }
  }
}
```



```
src
  main
    java
      rpg
        ui
          RPGMenuView.java
          RPGView.java
        RPG.java
        RPGController.java
        RPGModel.java
    resources
      menu
        background.png
        logo.png
        loss.jpg
        win.jpg
      sounds
        fart-sound.wav
        song-menuu.mp3
        song-poopy.mp3
      sprites
        bulbasaur.png
        drug.png
        grass-tile.png
        house.gif
        obstacle-panel.png
        obstacle-rock.png
        poopy.png
      styles.css
```

Code Structure and Organization

- MVC Organization.
- Separation of the UI package from the main RPG package.
- Code
  - A modular structure enhances maintainability by allowing easier identification and isolation of specific components for modification or troubleshooting.
  - Each module is loosely coupled, allowing developers to extend the application's functionality without disrupting existing code for good scalability.
  - The modular design promotes code reusability, as individual components can be reused in different parts of the application or even in other projects.
  - Adherence to programming principles and thorough documentation for improved code maintainability and collaboration.

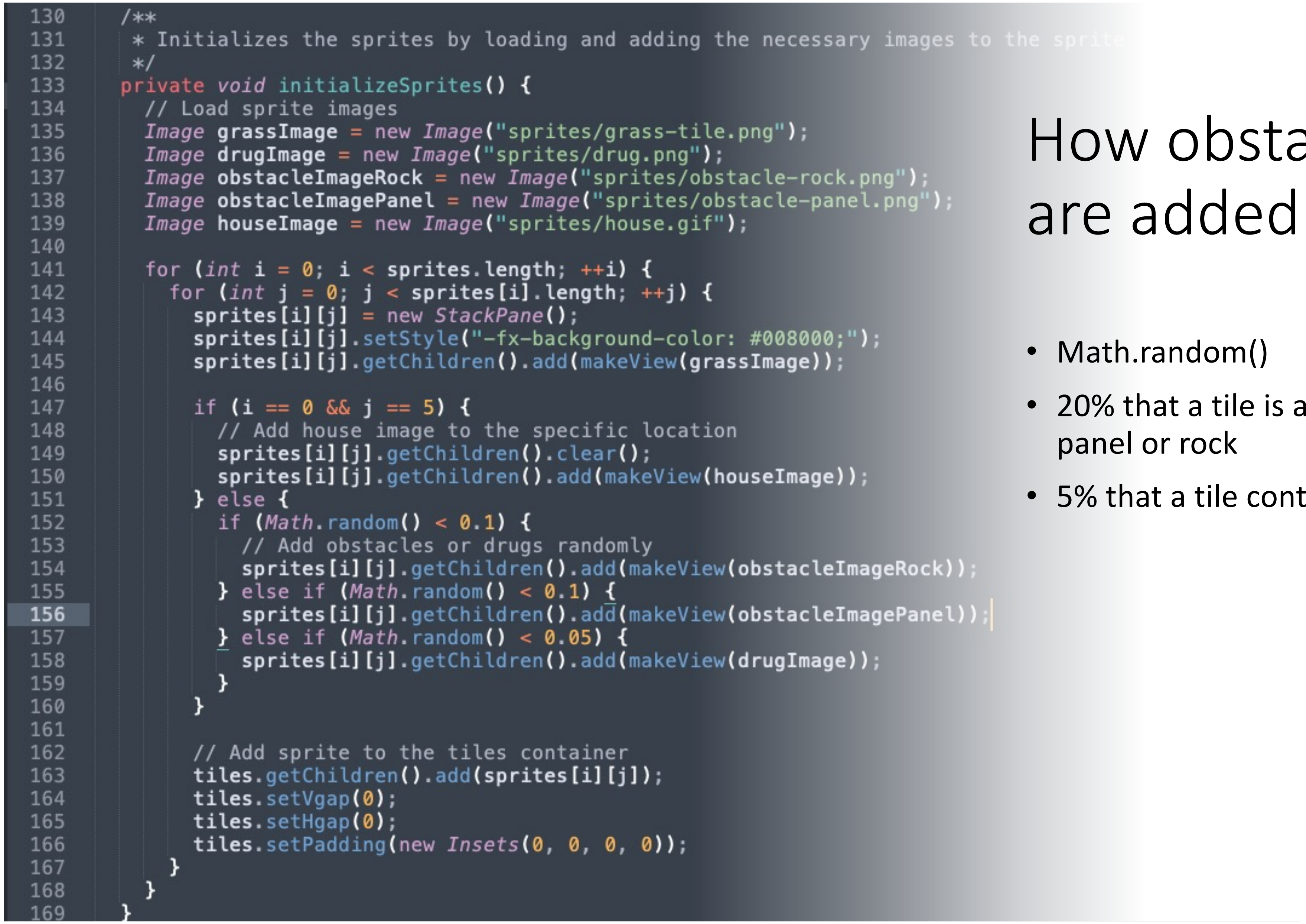


```java
public int timerSeconds() {
  return timerSeconds;
}

/**
 * Gets the modified remaining time on the timer by adding or subtracting the spe
 *
 * @param secondsToAddOrSubtract the number of seconds to add (if positive) or su
 *     negative)
 */
public int timerSeconds(int secondsToAddOrSubtract) {
  timerSeconds += secondsToAddOrSubtract;
  return timerSeconds;
}
```

timerSeconds Method

- Used for returning the timer time
- Overloading for when updates are needed regarding bonusses/ malusses

How obstacles are added

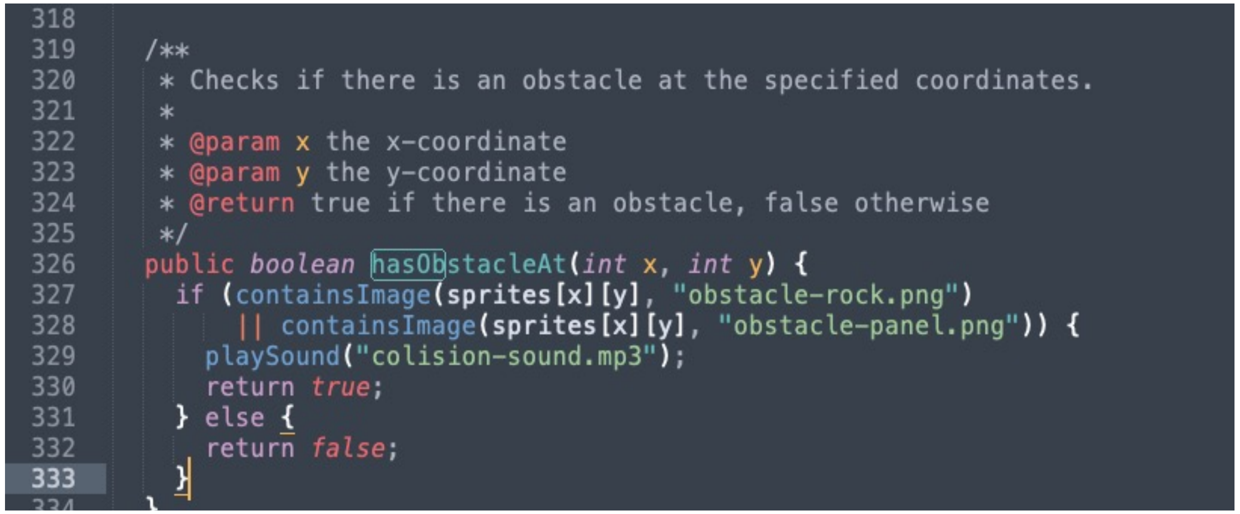


```java
/**
 * Initializes the sprites by loading and adding the necessary images to the sprite
 */
private void initializeSprites() {
  // Load sprite images
  Image grassImage = new Image("sprites/grass-tile.png");
  Image drugImage = new Image("sprites/drug.png");
  Image obstacleImageRock = new Image("sprites/obstacle-rock.png");
  Image obstacleImagePanel = new Image("sprites/obstacle-panel.png");
  Image houseImage = new Image("sprites/house.gif");

  for (int i = 0; i < sprites.length; ++i) {
    for (int j = 0; j < sprites[i].length; ++j) {
      sprites[i][j] = new StackPane();
      sprites[i][j].setStyle("-fx-background-color: #008000;");
      sprites[i][j].getChildren().add(makeView(grassImage));

      if (i == 0 && j == 5) {
        // Add house image to the specific location
        sprites[i][j].getChildren().clear();
        sprites[i][j].getChildren().add(makeView(houseImage));
      } else {
        if (Math.random() < 0.1) {
          // Add obstacles or drugs randomly
          sprites[i][j].getChildren().add(makeView(obstacleImageRock));
        } else if (Math.random() < 0.1) {
          sprites[i][j].getChildren().add(makeView(obstacleImagePanel));
        } else if (Math.random() < 0.05) {
          sprites[i][j].getChildren().add(makeView(drugImage));
        }
      }

      // Add sprite to the tiles container
      tiles.getChildren().add(sprites[i][j]);
      tiles.setVgap(0);
      tiles.setHgap(0);
      tiles.setPadding(new Insets(0, 0, 0, 0));
    }
  }
}
```

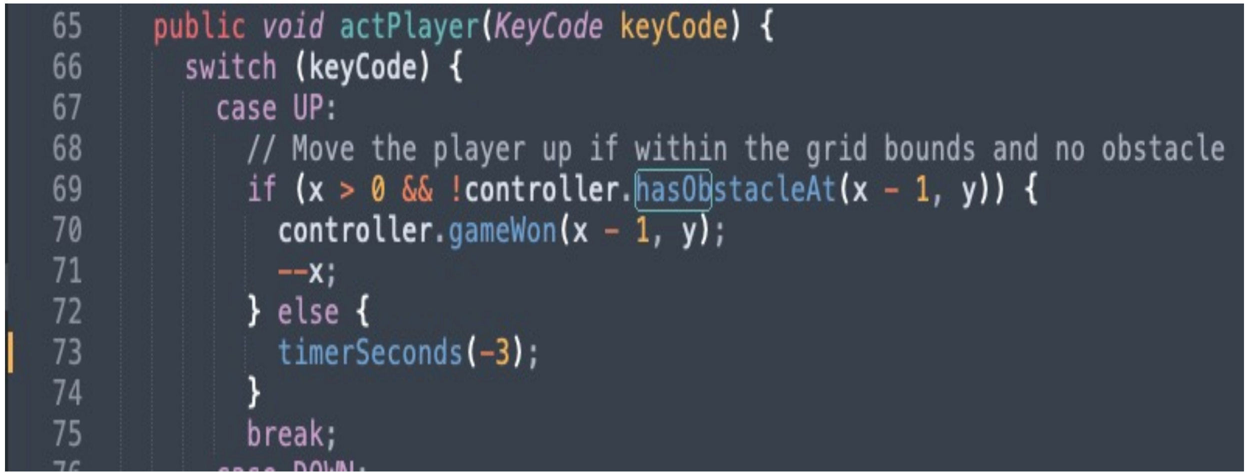
- Math.random()
- 20% that a tile is an obstacle panel or rock
- 5% that a tile contains drugs

Player Movement and Obstacle Handling

- actPlayer() called on event, switch with keyCode
- Collision with rocks and panels
- Drugs bonus encountered when the playerImage goes on a drug tile

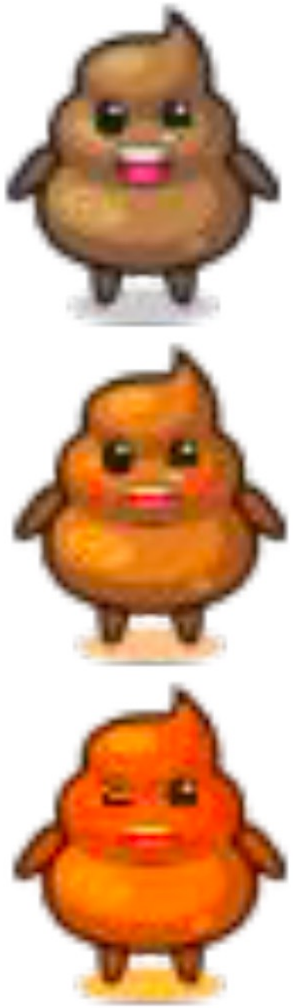


```java

/**
 * Checks if there is an obstacle at the specified coordinates.
 *
 * @param x the x-coordinate
 * @param y the y-coordinate
 * @return true if there is an obstacle, false otherwise
 */
public boolean hasObstacleAt(int x, int y) {
  if (containsImage(sprites[x][y], "obstacle-rock.png")
      || containsImage(sprites[x][y], "obstacle-panel.png")) {
    playSound("colision-sound.mp3");
    return true;
  } else {
    return false;
  }
```



```java
public void actPlayer(KeyCode keyCode) {
  switch (keyCode) {
    case UP:
      // Move the player up if within the grid bounds and no obstacle
      if (x > 0 && !controller.hasObstacleAt(x - 1, y)) {
        controller.gameWon(x - 1, y);
        --x;
      } else {
        timerSeconds(-3);
      }
      break;
```

Poopy changes appearance



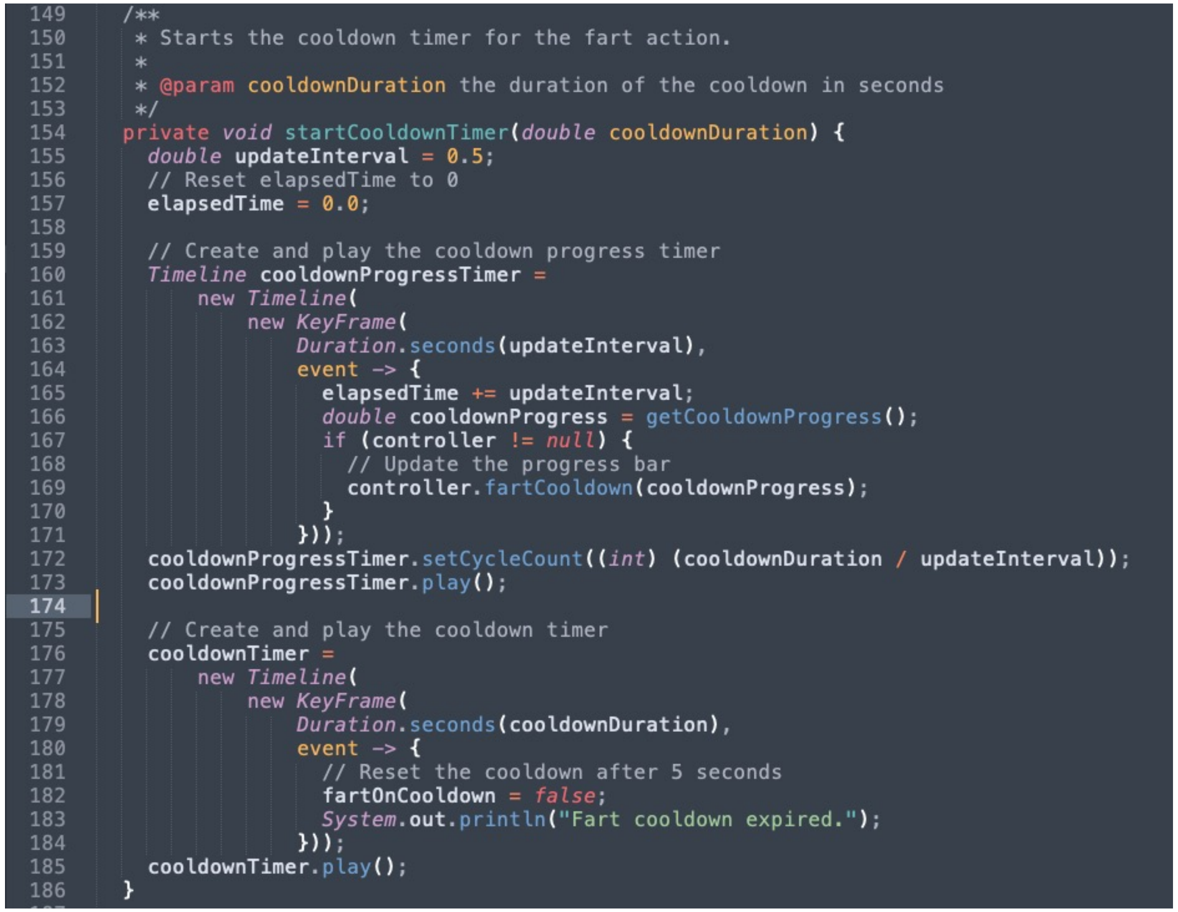
```java
playerImage.setViewport(new Rectangle2D(0, 0, 50, 50));
```

```java
public void updatePlayerState(int state) {
  if (state > 2 || state < 0) {
    return;
  }
  playerImage.setViewport(new Rectangle2D(0, state * 50, 50, 50));
}
```

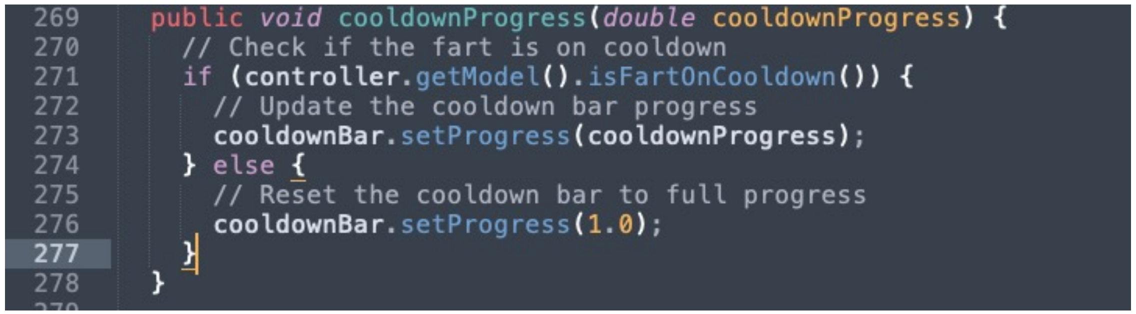
Poopy is stored in one image containing its three states. Depending on the timer, the view slides over and crops the new state.

Cooldown and Fart Action

- Performing the fart
  - Keycode “F”
  - fart() method
  - When above 10s left, 1/5 to poop and lose and 4/5 to add 5s to the timer this is done with Math.random()
- Cooldown on fart
  - fartOnCooldown
  - Timeline objects
  - cooldownProgress



```java
/**
 * Starts the cooldown timer for the fart action.
 *
 * @param cooldownDuration the duration of the cooldown in seconds
 */
private void startCooldownTimer(double cooldownDuration) {
  double updateInterval = 0.5;
  // Reset elapsedTime to 0
  elapsedTime = 0.0;

  // Create and play the cooldown progress timer
  Timeline cooldownProgressTimer =
    new Timeline(
      new KeyFrame(
        Duration.seconds(updateInterval),
        event -> {
          elapsedTime += updateInterval;
          double cooldownProgress = getCooldownProgress();
          if (controller != null) {
            // Update the progress bar
            controller.fartCooldown(cooldownProgress);
          }
        }));
  cooldownProgressTimer.setCycleCount((int) (cooldownDuration / updateInterval));
  cooldownProgressTimer.play();

  // Create and play the cooldown timer
  cooldownTimer =
    new Timeline(
      new KeyFrame(
        Duration.seconds(cooldownDuration),
        event -> {
          // Reset the cooldown after 5 seconds
          fartOnCooldown = false;
          System.out.println("Fart cooldown expired.");
        }));
  cooldownTimer.play();
}
```



```java
public void cooldownProgress(double cooldownProgress) {
  // Check if the fart is on cooldown
  if (controller.getModel().isFartOnCooldown()) {
    // Update the cooldown bar progress
    cooldownBar.setProgress(cooldownProgress);
  } else {
    // Reset the cooldown bar to full progress
    cooldownBar.setProgress(1.0);
  }
}
```

Other Features

- Javafx dependency javafx-media, sound effects with media player
- All styles for ui are done in a separate sheet styles.css add they are used using
  - getStyleClass().add(”classname"); on the javafx object
  - The style sheet is added to the scene using getStylesheets().add(getClass().getResource("/styles.css").toExternalForm());
- RPGMenuView
  - First view shown
  - Can go back to it with the go to menu button
  - Or when a lose or win happens it takes in the parameters gamePlayed and gameWon