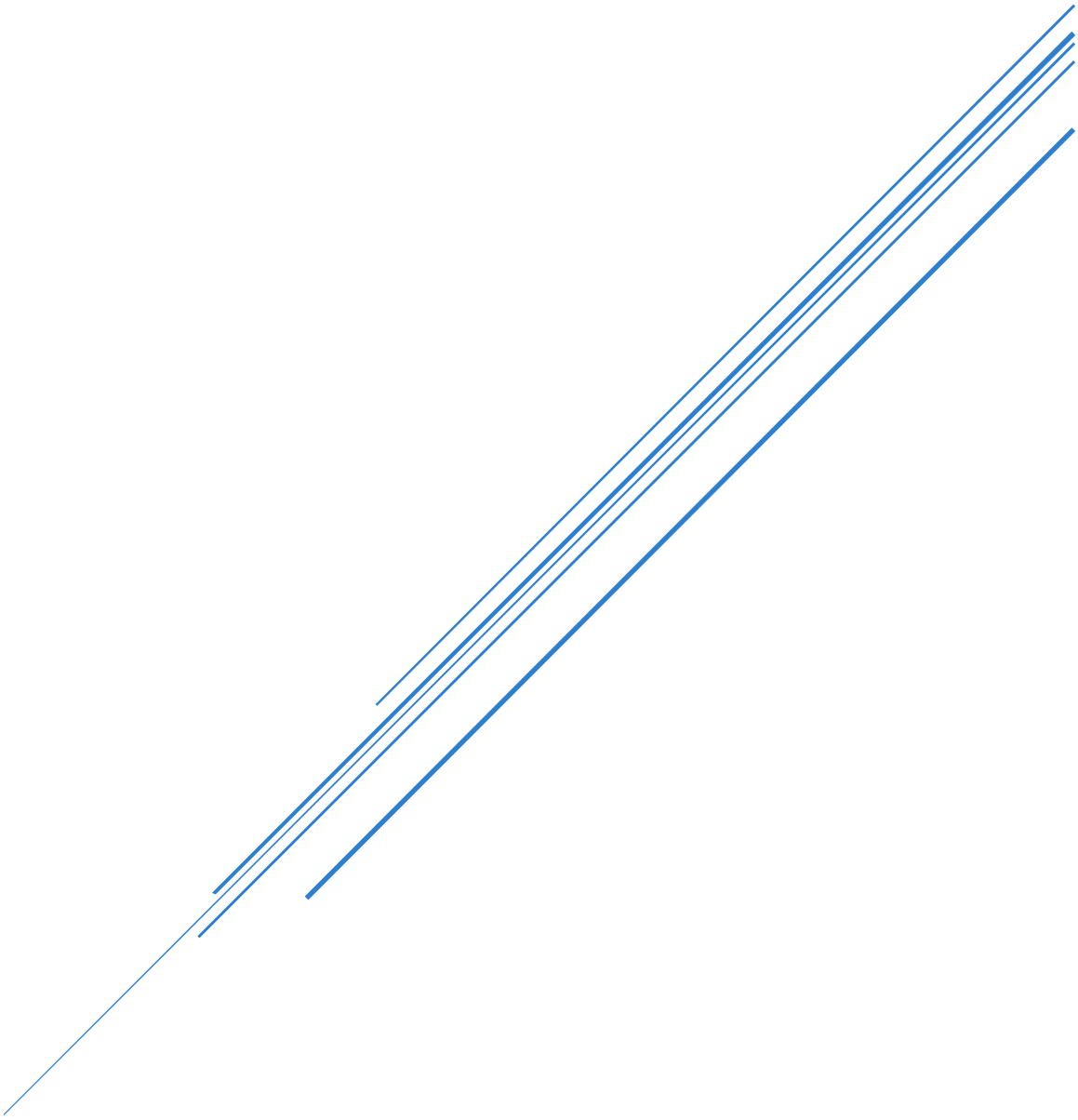
WATCH OUT, NANA!

CS 1083 Project Documentation

Group Members: Chukwunonso Ekweaga, Ashlee Muchirahondo, Alamin Adeleke,



Game Concept for CS 1083 Game Overview

Watch Out, Nana! is an engaging risk-vs-reward game where players help a grandmother cross a dangerous eight-lane highway while making strategic decisions about when to cash out their earnings. The game combines elements of probability management with decision-making, creating an entertaining experience for both streamers and their audience.

Core Gameplay Mechanics

Basic Concept

- Players guide Nana across an eight-lane highway
- Each successful lane crossing awards 125 VBucks to the player
- Risk increases with each lane crossed
- Players can choose to "cash out" at any time by calling a traffic warden
- Failure results in losing all accumulated earnings

Probability System

- Starting failure probability: 10%
- Probability increases by 3% per lane crossed
- Progressive risk structure:
    - Lane 1: 10% failure rate
    - Lane 2: 13% failure rate
    - Lane 3: 16% failure rate
    - Lane 4: 19% failure rate
    - Lane 5: 22% failure rate
    - Lane 6: 25% failure rate
    - Lane 7: 28% failure rate
    - Lane 8: 31% failure rate
- Probability of successfully crossing all the lanes is approximately 15.5%

Payout Structure

- Maximum potential payout: 1,000 VBucks (8 lanes × 125 VBucks each)
- Minimum payout:125 VBucks (single lane)
- Strategic decision points:

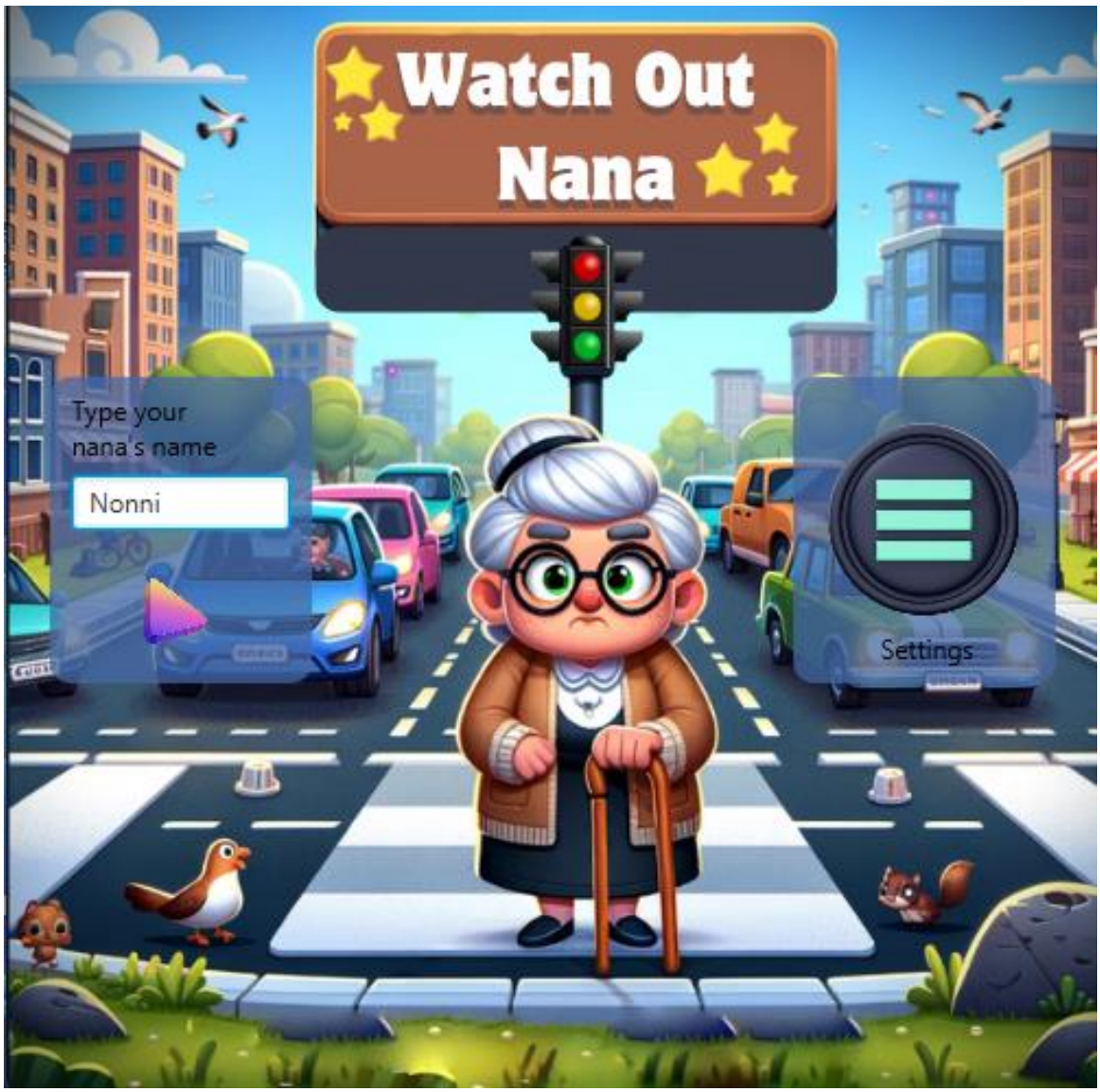
- Early cashout: Lower risk, probable small reward
- Full crossing attempt: High risk, potential for maximum reward

UI/UX Design

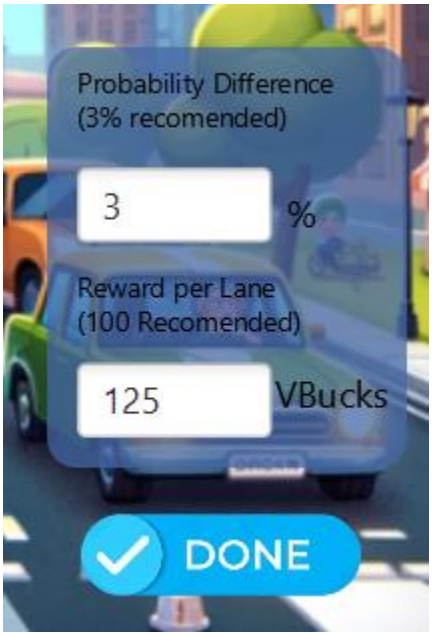
There are two scenes to the game, the home screen and the game screen.

1. The Home screen is made up of three central elements, A name input, a payout input, a probability input, a play button and settings button.

   The image shown below is the home screen:



This is the settings part of the home screen after the settings button is pressed:



- The name input is a labelled text field, the label indicating to input their grandma's name. This field is auto populated with a random name. This is explained in the Random Loss Messages and Grandma Names section.

- The assets on the left (the text fields and labels) are not visible unless the settings button is clicked, i.e. they are not seen by the audience unless the streamer clicks the settings button.

- The payout input is a labelled text field, the label asking to input the payout amount per lane, this is set to 125 by default (as seen in probability system).

- The probability input is a labelled text field, the label asking to input the difference in probability per lane, this is set to 5% by default (As seen in probability system).

The done button clears the settings panel and hides the input fields, returning the home screen to its default state.

Upon clicking the Play button, the inputted settings (grandma name, payout amount, and probability difference) are saved in the program for later use.

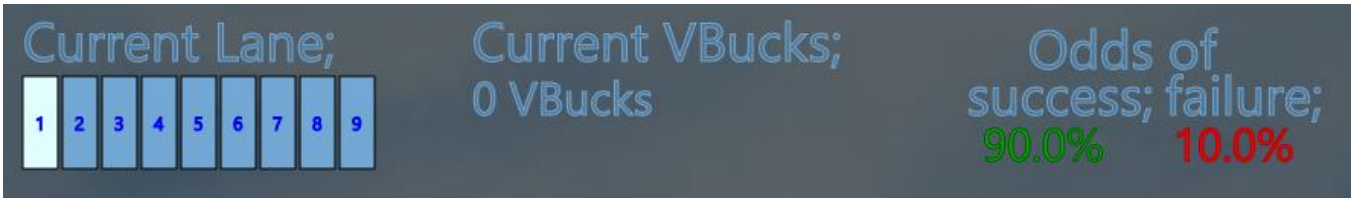
2. The GameScreen is made up of three parts, the output pane, the input pane, and the manView. These will be in the bottom, left, and center part of a border pane respectively.

a) The output pane – It is an object created from the "Output" class, and it will consist of the following data;

- Current lane indicator – It is the image shown below and will change when the cross button is clicked. It is made in its own class LaneIndecator that takes in x,y coordinates, which contains 8 rectangles coloured dark, but at least 1 coloured a different colour. The class has only one method, it takes in one parameter(an integer) which is the index of which rectangle is to be coloured differently, the other rectangles are coloured the same color.

- There are also text objects which contain

The Output pane will consist of the current Lane indicator, Accumulated VBucks counter and a text display for the probability of crossing successfully. The output pane will be a pane, and all its elements will be arranged manually with x, y coordinates (a grid pane or HBox would have consistent spacing, which is not the desired result as seen in the Image below).



b) Input Pane - The input pane is created using the Input class and includes three buttons: "Cross," "Call the Warden," and "Reset." Below is a detailed explanation of each button's functionality:

- A “Cross” button to attempt to cross to the next lane. The button will be disabled unless the conditions for player to be able to cross are met (If the game has not been either won or lost).

- A "Call Traffic Warden" button, to cash out all the current earnings made so far. TThe button will be disabled unless the conditions for player to be able to cross are met (If the game has not been either won or lost):

- A “Reset” button, for the Streamer’s convenience to return to the home screen after current game. This will not be clickable until the game has either been won or lost.

Each button in the input pane is a customized image button that glows when the mouse hovers over it. This process is handled by the glow method.

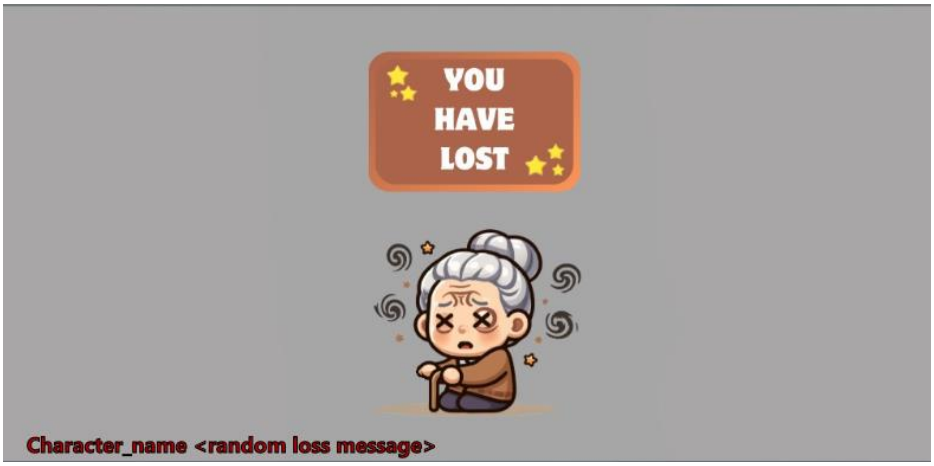
The input pane will be pane that contains a of all the buttons placed top of each other and will look like the image shown below:

It will be contained in the right part of a border pane (in the GameScreen class)

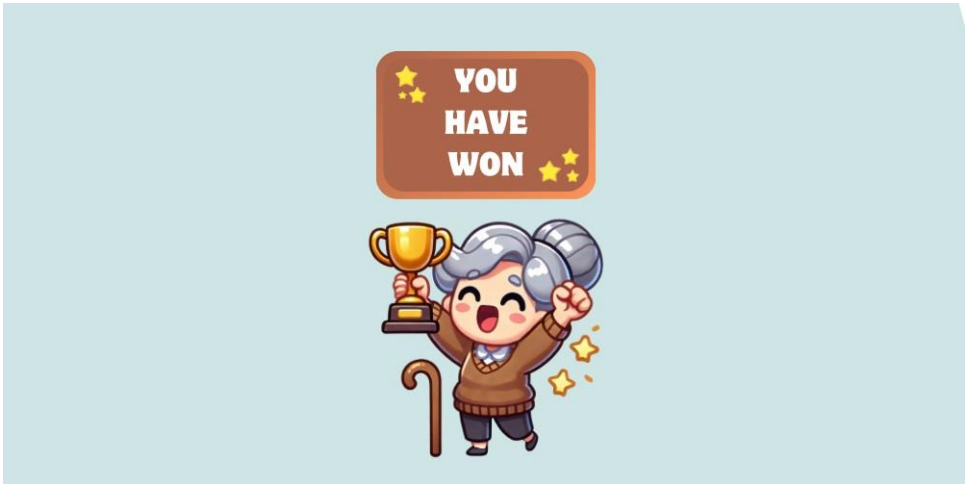
c) Main game: It will be in a class called “MainView” ,that is a pane that sits in the center part of the Border pane in the GameScreen class, that consists of 8 different lane objects (See Lane description), the nana character, and it will also be able to change into a Winning screen and losing screen with the use of certain methods.

There is also a walk method which animates the nana character to move horizontally the distance of one lane. Along with a reset method to return the nana character to her starting position.

- The Losing screen – The image shown below is displayed when the player has lost and it displays the name the user chose for the character alongside a random loss message as shown in the image shown below. When the loss screen is displayed a womp womp sound is played all this is done through the “setLose” method in the MainView class, the “listen” method in the GameScreen class and the LossMessages class.



- The Victory Screen – It is displayed when the player has won the game or has decide to cash out the earning from the game. It displays the image shown below and plays a victory sound. When the player wins a victory sound is played. This is done using the “setWin” method in the MainView class and the “listen” method in the “GameScreen” class



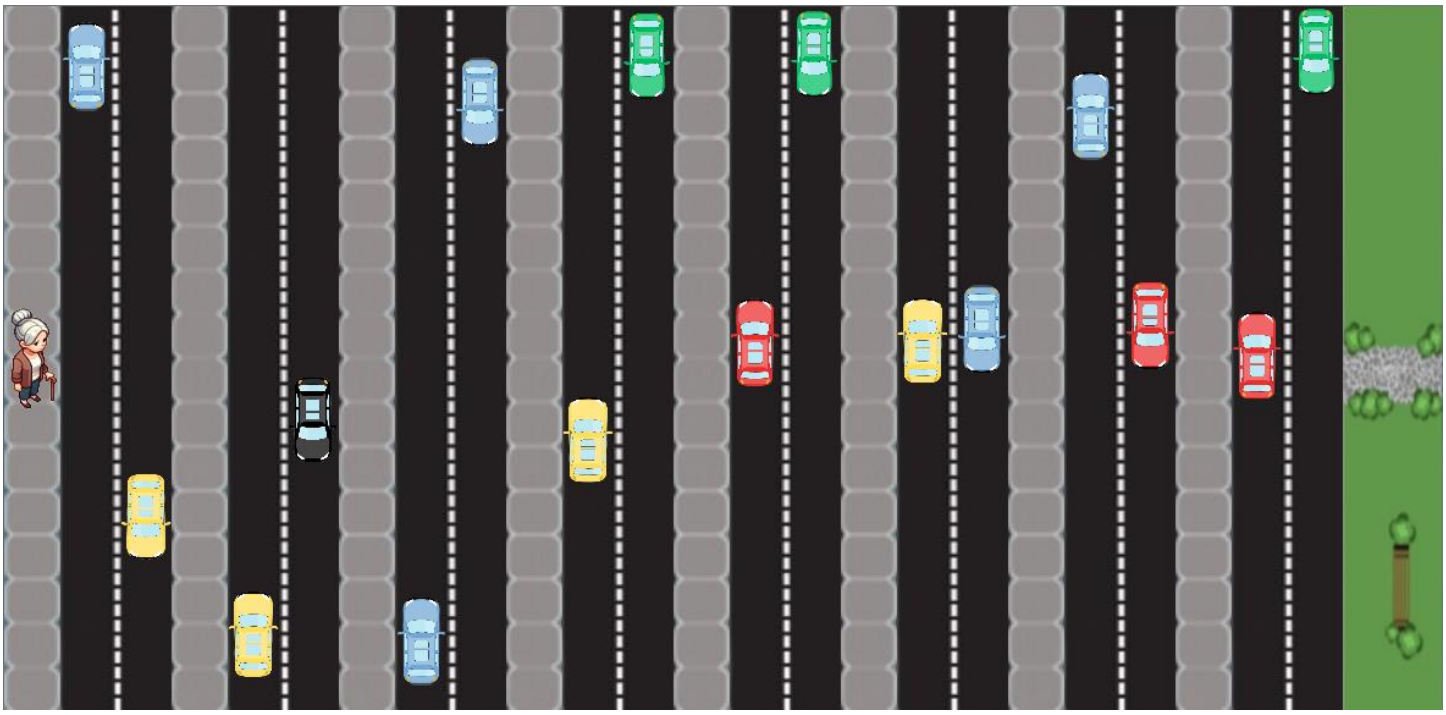
- Lane description – the lane object is a class with 3 main instance objects, the road, sidewalk, and car images. The sidewalk image lies beside the road and is designated to hold the nana character.

The road image holds the car images (made with a random car image in the assets file at instantiation). And are placed on either side of the road image (the second is rotated 180 degrees).

Car Animation

There are “animUp” and “animDown” methods that take in an image (the car) and animates it to move repeatedly up or down respectively,

There is also a class, “EndLane” which extends Lane but does not contain any cars. It is a specialized lane meant to signify end point of the game in the mainView.



(MainView with 8 Lane objects and an EndLane at the end.)

Nana Animation

The main view class (which contains the nana character) has a walk method, which animates (using the JavaFX Translate Transition class) the nana character to move horizontally the space of one lane and stores the distance nana has moved to a variable. The class also has a reset method which returns the nana character to the position of the first lane.

Application

Firstly, the scene of the app is the “changingScene” object that extends scene. It has a gameScreen and homeScreen instance data and sets its root to the homeScreen by default. It also has methods to swap its root pane between them.

The application (Driver) is simple and concise. It sets the title, icon, resizability and sets the scene to be the changingScene. It also has static instance data (name, pay, and odds), along with static accessor and mutator methods for them. the instance data are used throughout the program

Music and Sound Effects

Music and Sound Effects enhance the quality and experience of any game. The following paragraphs will explain how the implementation of the media class will enhance the enjoyability of the game and the reception of the game from the streamer’s audience.

The Media class is used to add music to the game’s home screen. It uses a media player to play the background music, and it has methods for loading, playing and stopping the music. The loading method is called in the default construct to load the music from the specified file path. It also has a method that switches between the tracks in the folder containing the background music. This ensures a smooth transition between each of the songs in the background music folder. The home screen class also contains a method that ensures that when play button is clicked the home screen background music stops and the background for game screen starts playing.

The GameScreen class manages sound effects during gameplay. It utilizes multiple MediaPlayer instances for different sound effects (e.g., button clicks, win, and lose sounds) and a Timeline for playing random horn sounds at intervals. The class contains methods that start and stop sound effects. It is programmed to play winning sounds when the player wins, losing sounds when the player loses, and click sounds that are heard throughout the game.

The audio components are designed to be easily customizable and reusable, allowing for easy modification of sound effects and music tracks. This will be important for the streamer because they will be able to pick the sounds they want, creating a more personalized gaming experience. Additionally, this flexibility ensures that the game can adapt to future updates or changes, making it easier to keep the content fresh and engaging for both the streamer and their audience.

Random Loss Messages and Grandma Names

The RandomGameText class controls all randomized text generation for the game. It contains two static arrays: one with over 100 creative grandma names (e.g., "Yaya-Belle", "Mamita-Lou") and another with 100+ humorous loss messages (e.g., "got hit by a rogue forklift!"). The class provides two key methods:

- getGranName(): Returns a random grandma name, which is used to auto-populate the name field in the HomeScreen constructor.
- getLossMessage(): returns a random failure message, combining it with the player-chosen name.

Code Implementation

```
// Auto-populate name field

nameField.setText(RandomGameText.getGranName());


// Display failure message

message.setText(RandomGameText.getLossMessage());
```

Methods in the GameScreen Class

CanReset() – returns a boolean based on whether the game should be able to reset, if the game has either been won or lost. It is mainly used in the reset button listener methods

Reset() – a method to set the mainview to the default (return from the win/lose screen, and reset character's position), reset the internal data(payout amount, current lane), reset the values of the output pane, and enable any previously disabled buttons.

disableAllButtonsExceptReset() - This is a method that disables all the buttons except the reset button. This method is called when the game has ended. This could be if the player lost, won or cashed out early.

listen() - This is a method that sets the function for all the buttons in the input pane. It is called once in the constructor. It is responsible for the click sound when any of the buttons are pressed, the victory sound heard when the player wins , and the loss sound heard when the player loses and the switch to the losing screen. It calls methods from the mainView class that handles the switch from the game screen to the victory screen if the player wins, and the switch from game screen to loss screen if the player loses.

update() - This method is responsible for the accurate display of the player's progress in the output pane. It updates all elements in the output pane: Current lane indicator (e.g., LaneIndicator class), Accumulated VBucks counter, Probability of success (e.g., 10% → 13% per lane).

getInput() - This method is responsible for adding the input pane to the game screen. It is called in the default constructor of the GameScreen class.

loadBackGroundSounds() - This method organizes all the background sounds into an array and puts them into a for each loop that sets the volume for each sound.

startPlayingSounds() - This method plays a random background sound every 5 seconds.

stopPlayingSounds() - This method stops the background sounds

Methods in the HomeScreen Class

draw() - This method creates the layout of the home screen. It is called in the default constructor of the HomeScreen class.

setFunc() - This method is run when the settings button is clicked. It reveals the payout input and probability difference input. It is also responsible for the done button's function

The rest of the methods in the home screen class are explained in the music and sound effect section

glow(ImageView img) - This method accepts an imageView as its parameter and makes the image glow when the mouse is hovered on top of it and reverts it to its original state when the mouse is no longer on top of the image. This method is used on all the buttons in the input pane

setValues() - This method is used to save the settings selected by the streamer. It saves the settings to static variables (as mentioned in the "Application" section of the document).

canGoToGameScreen() - This method determines whether the streamer can start playing the game or not. It allows the streamer to if the streames has entered a name longer than 2 characters and the payout per lane and odds to be subtracted fields contain at least one digit.

playMusic(String filePath) - This is method is responsible for the seamless transition between each track in the background music folder. It loads the files from the specified file path as the parameter and makes sure it plays the songs properly.

loadMusicFiles(String directoryPath)- THis method loads the music from the file directory and filters for files ending in".mp3" then it plays the first file it finds using the playNextTrack() method.

playNextTrack()- This method is used to play a song from the background music folder. It is called in the loadMusicFiles() method.

stopMusic() - This method stops the background music from playing. This method is run when the canGoToGameScreen() is run. THis is to ensure none of the background music plays at the same time as the background sounds in the Game Screen class.

Streamer Customization Guide

1. Text Assets

- **Modify GRANDMA_NAMES array in RandomGameText.java for inside jokes.**
- **Add edgy messages to LOSS_MESSAGES array.**

2. Sound Effects

- **Replace files in /assets/sounds/ for custom audio.**

3. Music

- **Add .mp3 files to background music folder for automatic rotation.**

Streamer-Viewer Interaction

1. Streamer controls the game using on-screen buttons and text fields
2. Viewers can participate by:
    - Suggesting when to cross or stop
    - Discussing risk/reward trade-offs.
    - Celebrating wins or commiserating losses
    - Predicting outcomes
3. Large, clear interface allows viewers to easily follow the action
4. Quick game rounds (under 5 minutes) to maintain engagement
5. Immediate visual and audio feedback keeps viewers involved