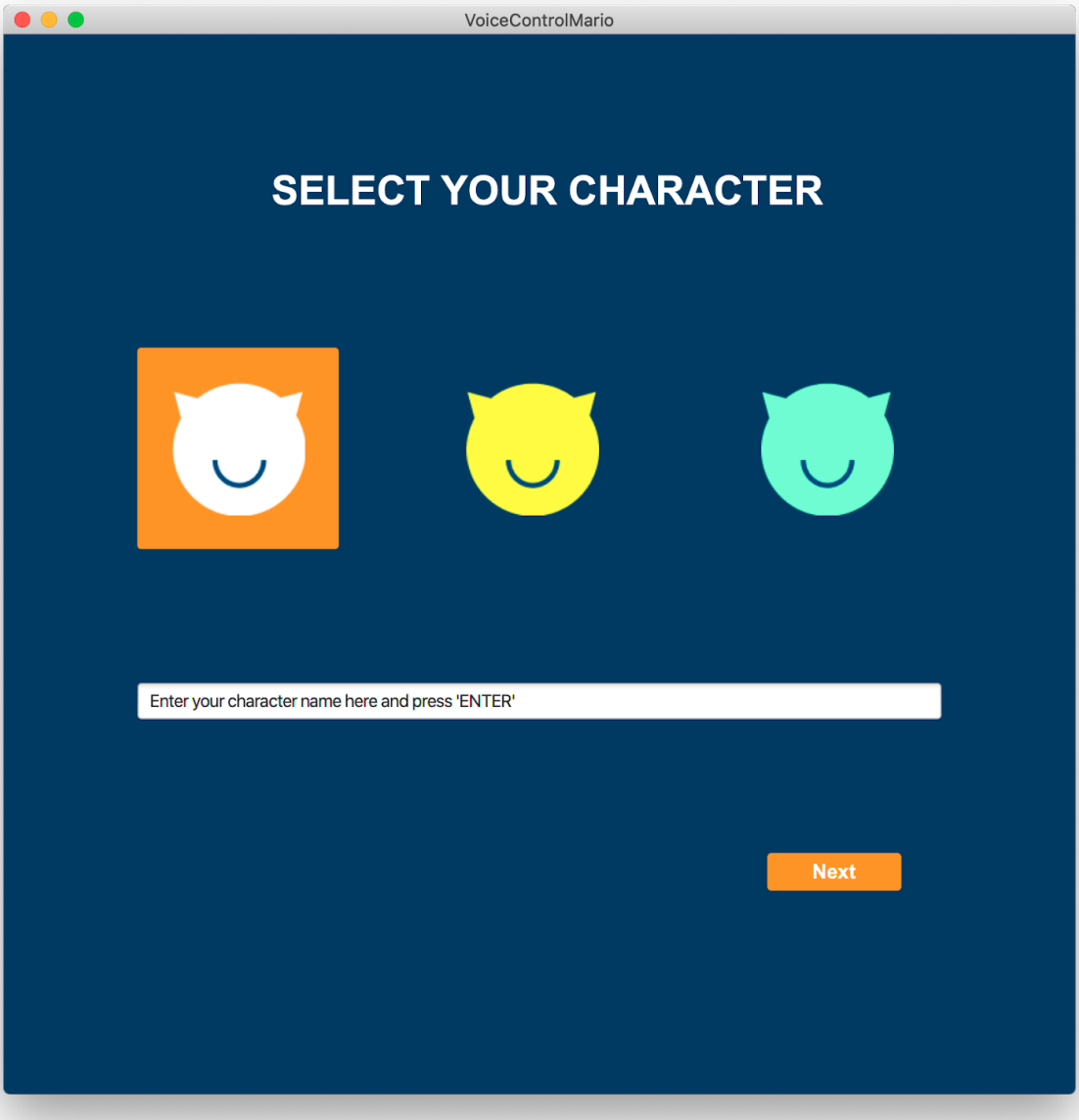
Voice Maze

G9
Jiayi Zhang
Tiannan Yang
Wan-Hsuan Chiang

This JavaFX game is a voice maze that the character moves in terms of the sound amplitude from user input. We practice the following interactions:

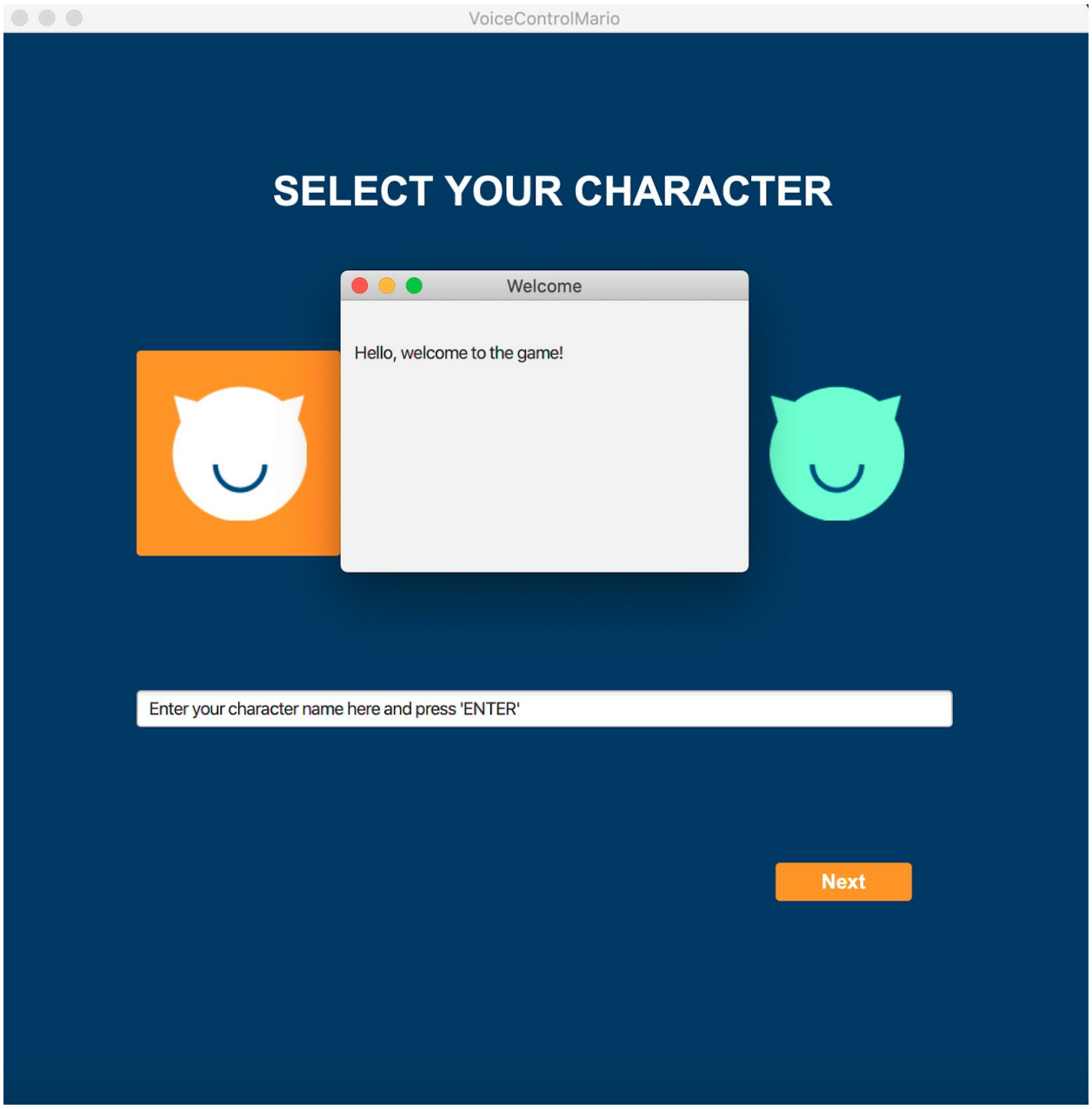
- Sound Control
- Keyboard Input
- Mouse Click
- Timer

First Screen: Select Characters

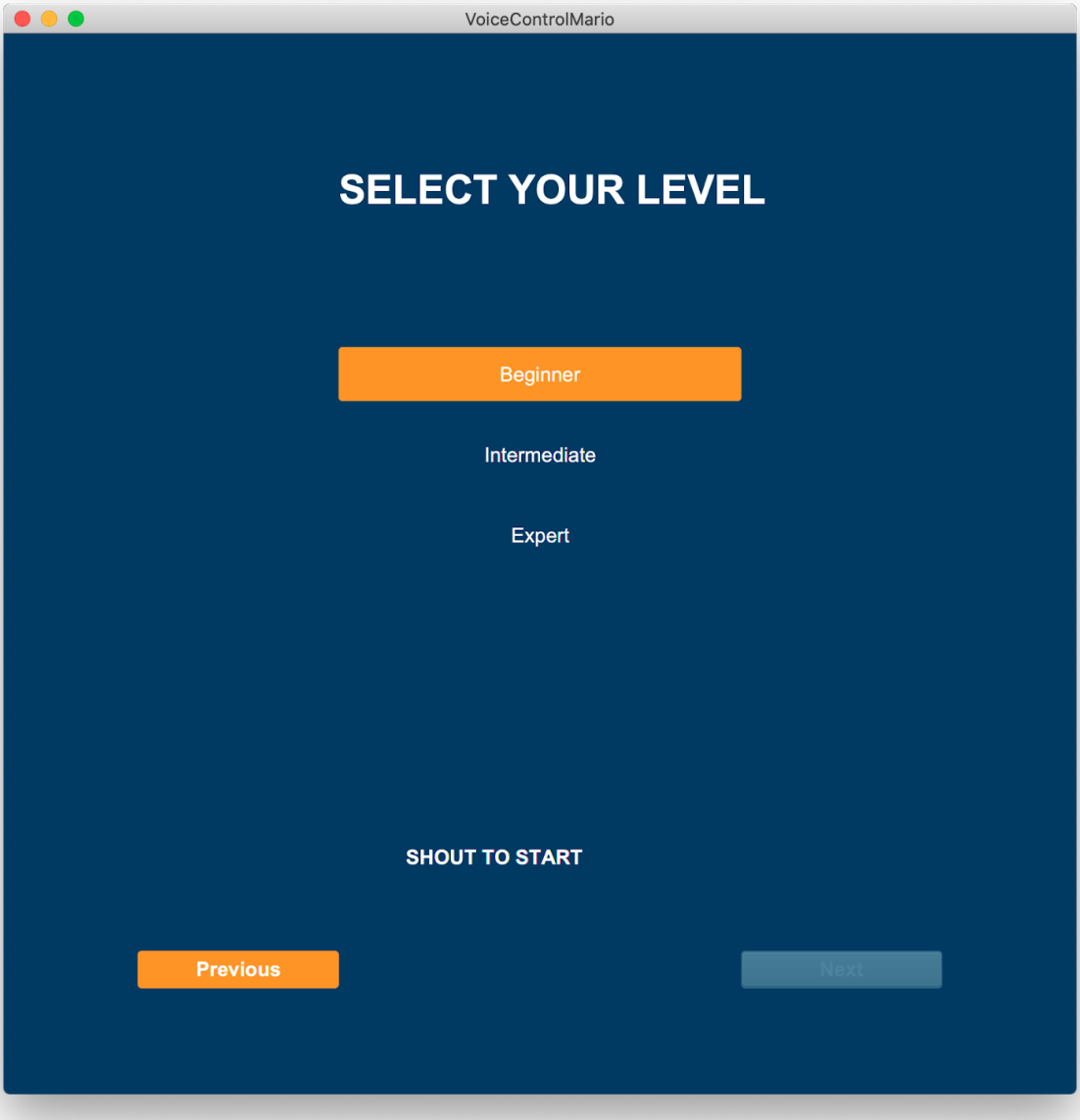


In this game, we create three characters. Users can click on the toggle button to select characters. Also, they can name the character by typing in the text field. After entering the content in the text field, users are asked to press “Enter” on the keyboard, and a small welcome window will pop out.

Popout Window



Second Screen: Select Level



This screen contains the following interactions:

- Button Click
- Sound Control

In this game, we use a unique way to start the game. We combine the microphone test and "start" together since valid sound input is critical to play the game. Therefore, by shouting into

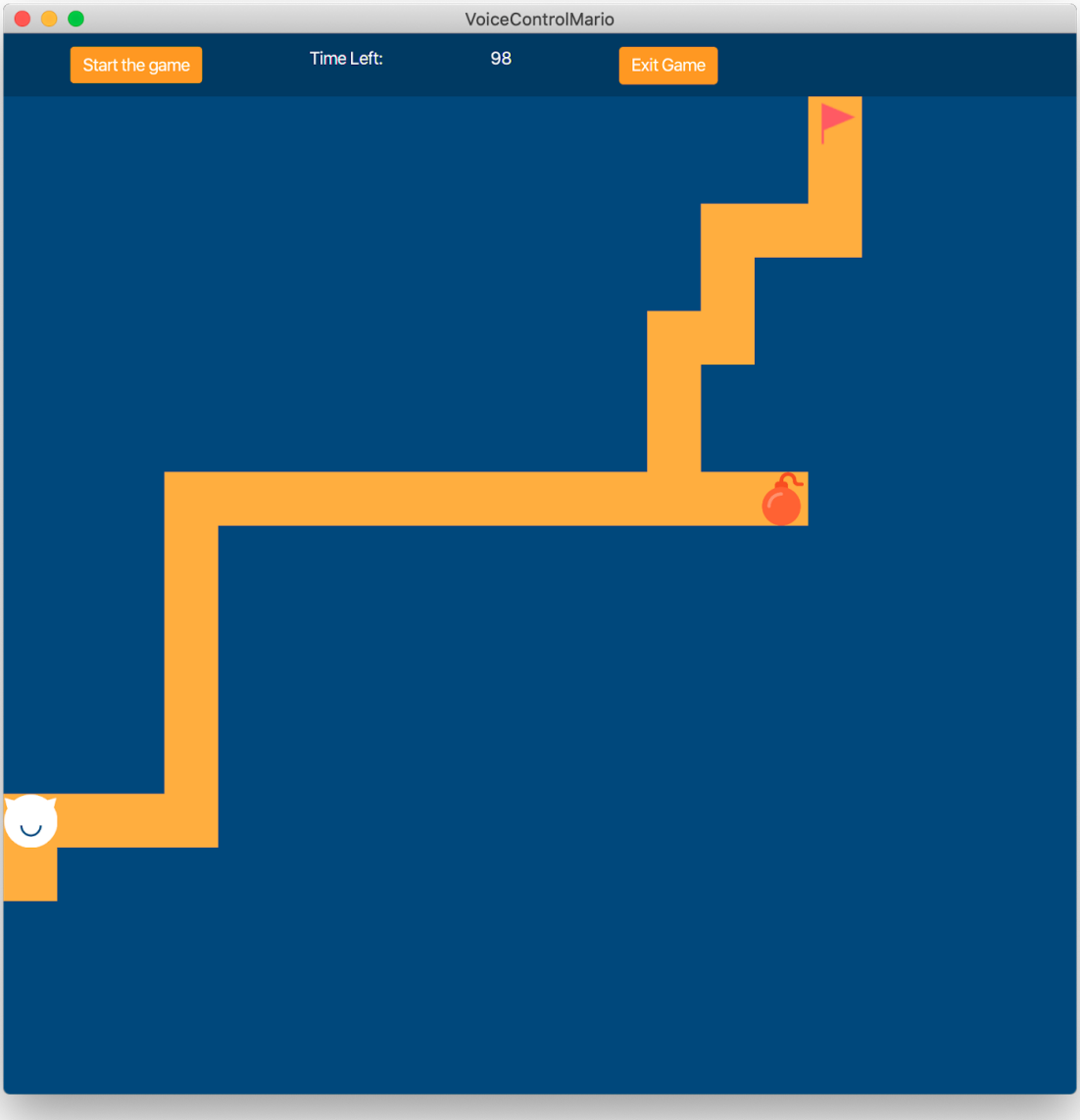
the microphone, the app will switch to the next screen in two seconds. The corresponding classes are as follows:

- `CaptureSoundController.java`
- `SoundController.java`
- `SoundModel.java`

We refer to Stackoverflow (https://stackoverflow.com/a/26576548) to create `CaptureSoundController`, which implementing `Java Sound API` to capture the sound. While keeping the entire code, we adjust the code that letting this controller return the sound amplitude to `SoundModel`. `SoundController`, implementing `AnimationTimer`, get the amplitude from `SoundModel`, and frequently check the value by each keyframe. If `SoundController` detects the sound which is louder enough, users will move to the next screen.

`SelectLevelView` creates this interface. We have three levels.

Third Screen: Map View (Game View)



The related classes are as follows:

- `MapView.java`
- `MapModel.java`
- `CharacterView.java`
- `App.java`

`MapModel` determines the maze. By calling `MapModel`, `MapView` can initiate the interface. Meanwhile, `CharacterView` paints the character and specifies the coordinates, in terms of the user's previous selection.

`App.java` plays a vital role on this screen. The controller inside has a similar idea to `SoundController`. This controller is an `AnimationTimer` that determines the movement of each keyframe. If it is in a low voice, the character will move forward to the right. If the voice input is louder enough, the character will start to move upwards. While the character reaches the top-right corner, the player succeeds.

UI components:
By clicking “Start the game”, the timer will begin from 100.
By clicking “Exit Game”, the window will be closed.