Jacob Marra

Sarah Nolan

Kristin Johnson

COMP1050-11

"Hang Dude"

Description

Almost everyone has some sort of electronic device. Whether it is a phone, a laptop, or gaming console, not many of the owners know much about what lies beneath their screen. In order to educate people on just part of what goes on behind their devices screen, The Three Doges created their version of the classic game known as hangman. The game of hangman involves a player trying to guess a phrase, one letter at a time. Their version, "Hang Dude", allows the user to learn computer terminology while playing a game. The dictionaries, in which the phrases are taken from, are text files that can be edited or replaced so that the user can upload any dictionary they want, allowing them to learn anything that they want. With "Hang Dude" anyone, no matter what age, can learn a variety of computer terminology.

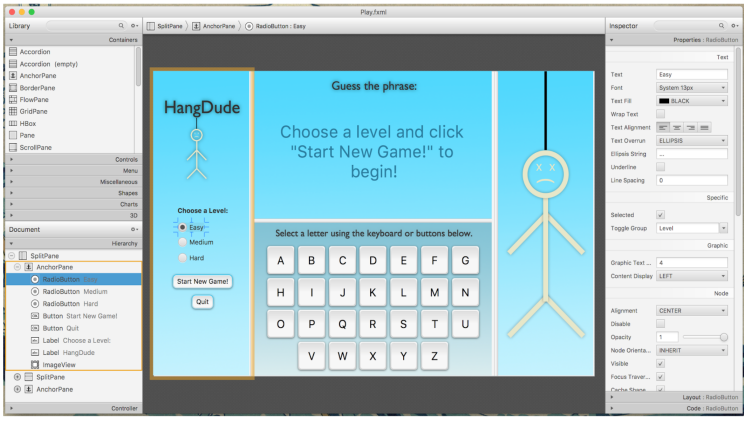
Analysis

The creation of "Hang Dude" included a lot of independent studying and trial and error. Besides using lectures and the textbook, The Three Doges spent time researching online and testing codes. One thing The Three Doges had to research was how to have a dialog window pop up while in the game. After countless attempts at different ways, JavaFX Alert was the best way to go about displaying the final screen. The Three Doges also found that the best way to get the gradient background was to use Photoshop to create image files of a gradient background, add

the image files to the project and use CSS to set the images as the background of the panes. To make the game easier for anyone to run, we wanted to turn the Eclipse project into a .jar file. We had trouble with this, because of the additional files in the project- like the stylesheet, image files, and text dictionaries. We eventually re-worked the file system to allow this, but still had trouble reading from the text files. Although they were in the src folder, using a Scanner would not read from the file. After some research, we had to instead use the text file as an input stream, and read it with a BufferedReader instead of a Scanner.

Implementation

In order to create "Hang Dude", The Three Doges used a variety of libraries and tools. The GUI is based on the JavaFX library. The Three Doges used Scene Builder to create the majority of the game's layout and appearance, and used a CSS stylesheet for any other characteristics such as the button gradients and shadowing. Other resources we used were String methods, the java I/O package, and utilities such as ArrayList, Collection, Iterator, and Random.



Creating the interface in Scene Builder

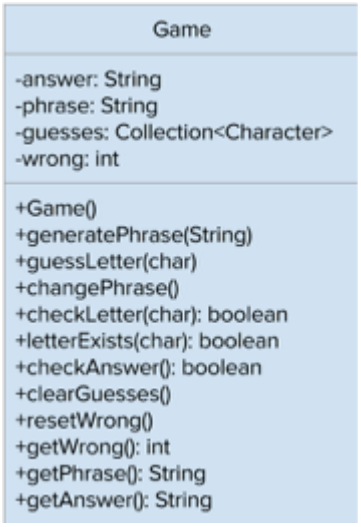
Play.java | GameController.java | Game.java | Play.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.effect.*?>
<?import javafx.scene.image.*?>
<?import javafx.scene.paint.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.text.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import javafx.scene.chart.*?>

<SplitPane fx:id="pane" dividerPositions="0.0, 0.0" maxHeight="600.0" maxWidth="900.0" minHeight="600.0" minW
  <items>
    <AnchorPane id="sidePane" fx:id="levelPane" maxHeight="600.0" maxWidth="200.0" minHeight="600.0" minWidth
          <children>
             <RadioButton fx:id="rdoEasy" layoutX="51.0" layoutY="298.0" mnemonicParsing="false" selected="tru
                <toggleGroup>
                   <ToggleGroup fx:id="Level" />
                </toggleGroup>
             </RadioButton>
             <RadioButton fx:id="rdoMedium" layoutX="51.0" layoutY="327.0" mnemonicParsing="false" text="Mediu
             <RadioButton fx:id="rdoHard" layoutX="51.0" layoutY="358.0" mnemonicParsing="false" text="Hard" t
             <Button id="gameButtons" fx:id="startButton" layoutX="40.0" layoutY="400.0" mnemonicParsing="fals
             <Button id="gameButtons" fx:id="quitButton" layoutX="79.0" layoutY="440.0" mnemonicParsing="false
             <Label layoutX="50.0" layoutY="266.0" text="Choose a Level:">
                <font>
                   <Font name="System Bold" size="13.0" />
                </font></Label>
             <Label alignment="TOP_LEFT" layoutX="22.0" layoutY="49.0" minWidth="100.0" text="HangDude">
                <font>
                   <Font name="Gill Sans" size="36.0" />
                </font>
                <effect>
                   <DropShadow color="#17595eba" />
                </effect>
             </Label>
             <ImageView fitHeight="131.0" fitWidth="73.0" layoutX="56.0" layoutY="92.0" pickOnBounds="true" pr
                <image>
                   <Image url="@HangDude.png" />
                </image>
                <effect>
                   <DropShadow color="#00000085" />
                </effect>
             </ImageView>
          </children></AnchorPane>
```

```
#buttons {
    -fx-background-color:
        linear-gradient(#f2f2f2, #d6d6d6),
        linear-gradient(#fcfcfc 0%, #d9d9d9 20%, #d6d6d6 100%),
        linear-gradient(#dddddd 0%, #f6f6f6 50%);
    -fx-background-radius: 8,7,6;
    -fx-background-insets: 0,1,2;
    -fx-text-fill: black;
    -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0
}
buttons:hover {
    -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0) , 0, 0 ,
}

#gameButtons {
    -fx-background-color:
        linear-gradient(#00c6ff, #b5eeff),
        linear-gradient(#fcfcfc 0%, #d9d9d9 20%, #d6d6d6 100%),
        linear-gradient(#dddddd 0%, #f6f6f6 50%);
    -fx-background-radius: 8,7,6;
    -fx-background-insets: 0,1,2;
    -fx-text-fill: black;
    -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0
}
#gameButtons:hover {
    -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0) , 0, 0 ,
}

#sidePane {
    -fx-background-image: url("Panel.png");
}

#topPane {
    -fx-background-image: url("TopPane.png");
}

#bottomPane {
    -fx-background-image: url("BottomPane.png");
}
```

FXML document and StyleSheet

Design

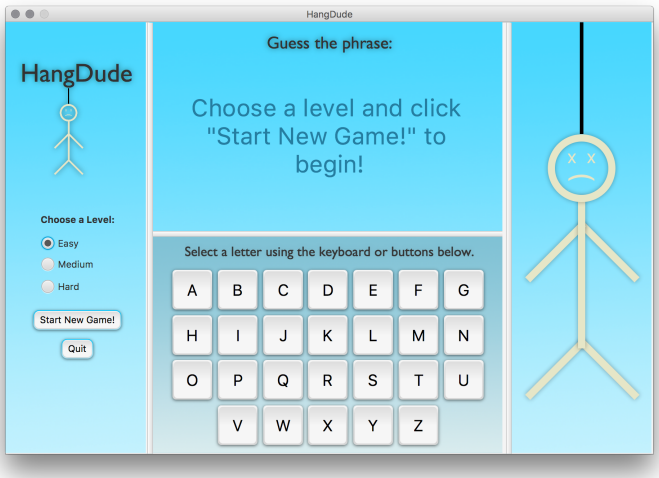


HangDude contains three Java classes. The first, “Play,” is used to load the GUI and launch the game. The second, “GameController,” is used to control the GUI and handle UI events. The last, “Game” is the ‘brains’ of the game, and controls the game itself. Hang Dude’s “Game” class contains four private variables, String “answer”, String “phrase”, Collections <Character> “guesses” and int “wrong”. The game also includes one constructor, Game(), and eleven public methods, generatePhrase(), guessLetter(), changePhrase(), checkLetter(), letterExists(), checkAnswer(), clearGuesses(), resetWrong(), getWrong(), getPhrase(), and getAnswer().

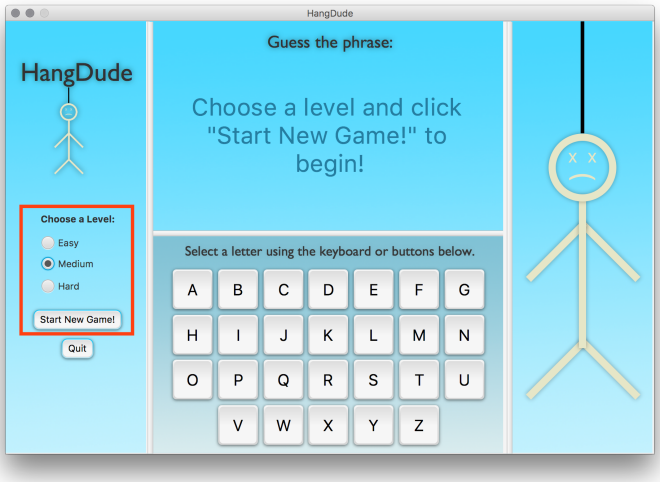
To determine if the letter selected is in the current word, a group of if/else statements were used. If the user selected a letter from the keyboard, a KeyEvent is invoked on the pane, and the source is ran through an if/else to determine the letter that was pressed. This letter is then passed to the letterGuess method in the Game class. If the user selected a letter from the on-screen keyboard, an ActionEvent is invoked, and its source is ran through an if/else to determine the letter that was pressed. The letter is then passed to the letterGuess method in the Game class.

User Instructions for gameplay:

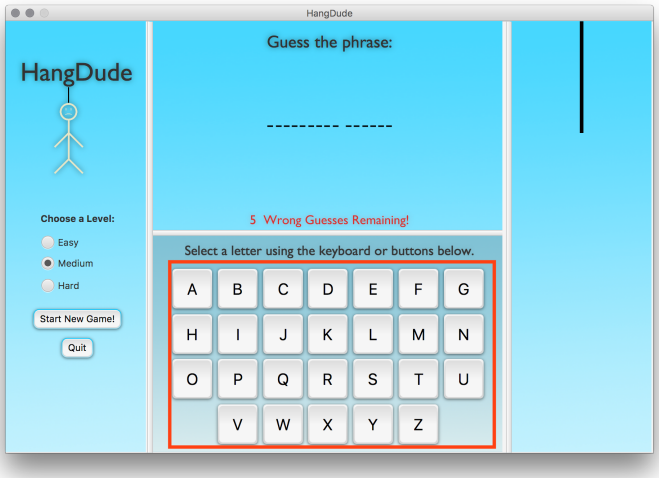
The game can be run as an Eclipse project or through the provided .jar file.



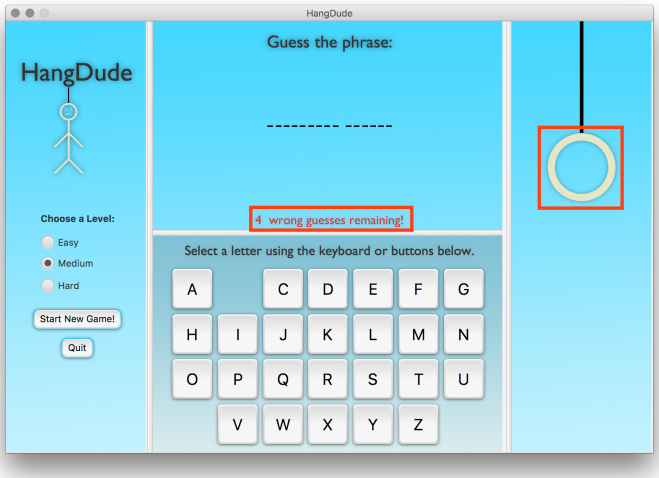
Opening Screen



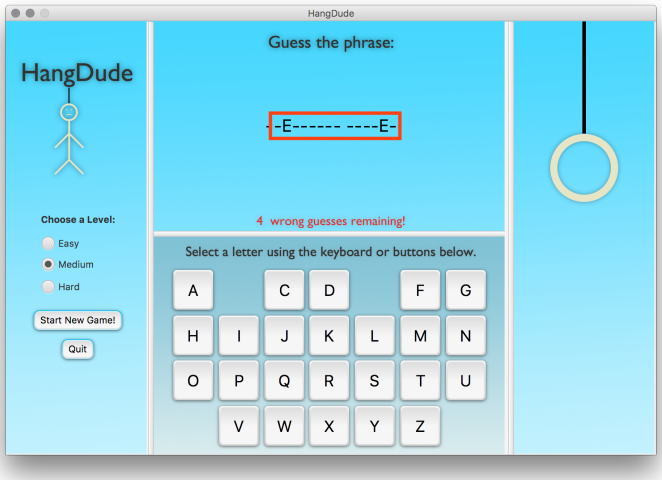
1) Choose a level and then click 'Start New Game!'



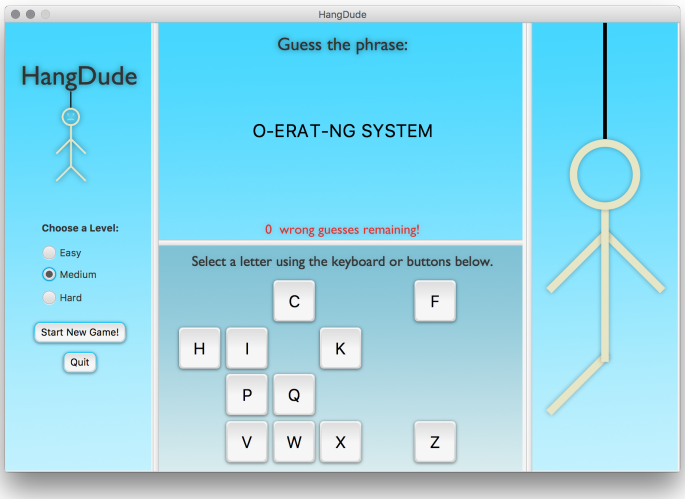
2) Using either the buttons or the keyboard, guess a letter



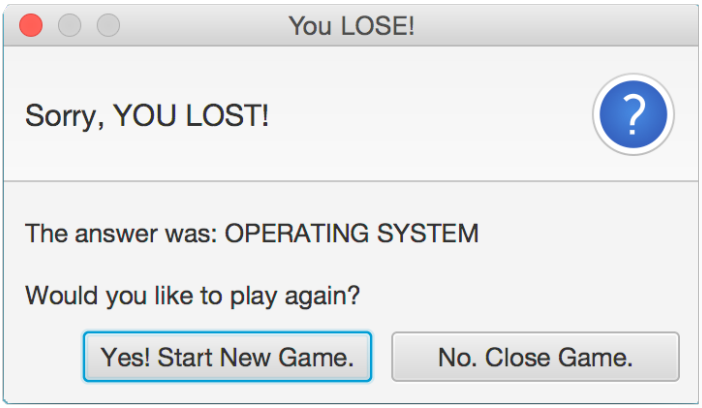
3) If the guess is wrong, part of the man will appear and your number of guesses will decrement by 1

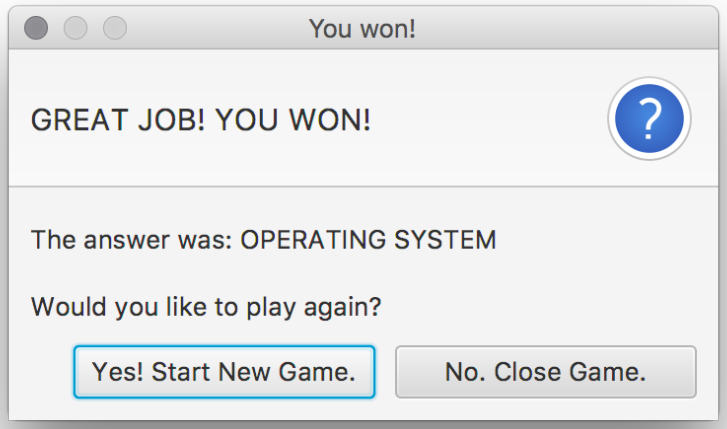


4) If the letter is there, it will appear above the keyboard



5) Keep guessing until you either win or you run out of guesses





6) Depending on whether you win or lose, one of the pop ups will appear. Click “Yes! Start New Game” to select a level and play again or click “No. Close Game.” to exit the window.