WORD CANNON

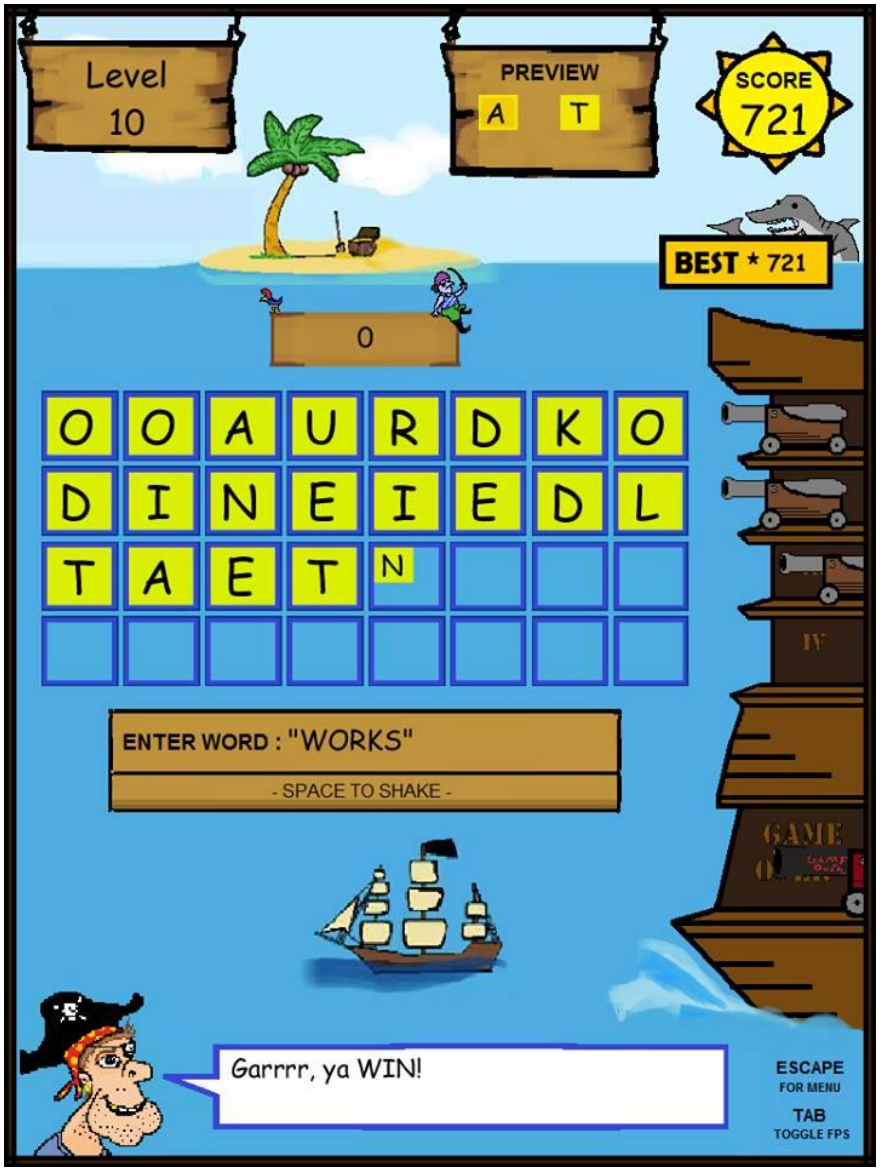
THE SWASHBUCKLING SPELLING GAME

A GAME BY

RYAN RUSHTON, ROBERT BLASZKOWSKI AND AIDEN BROWN

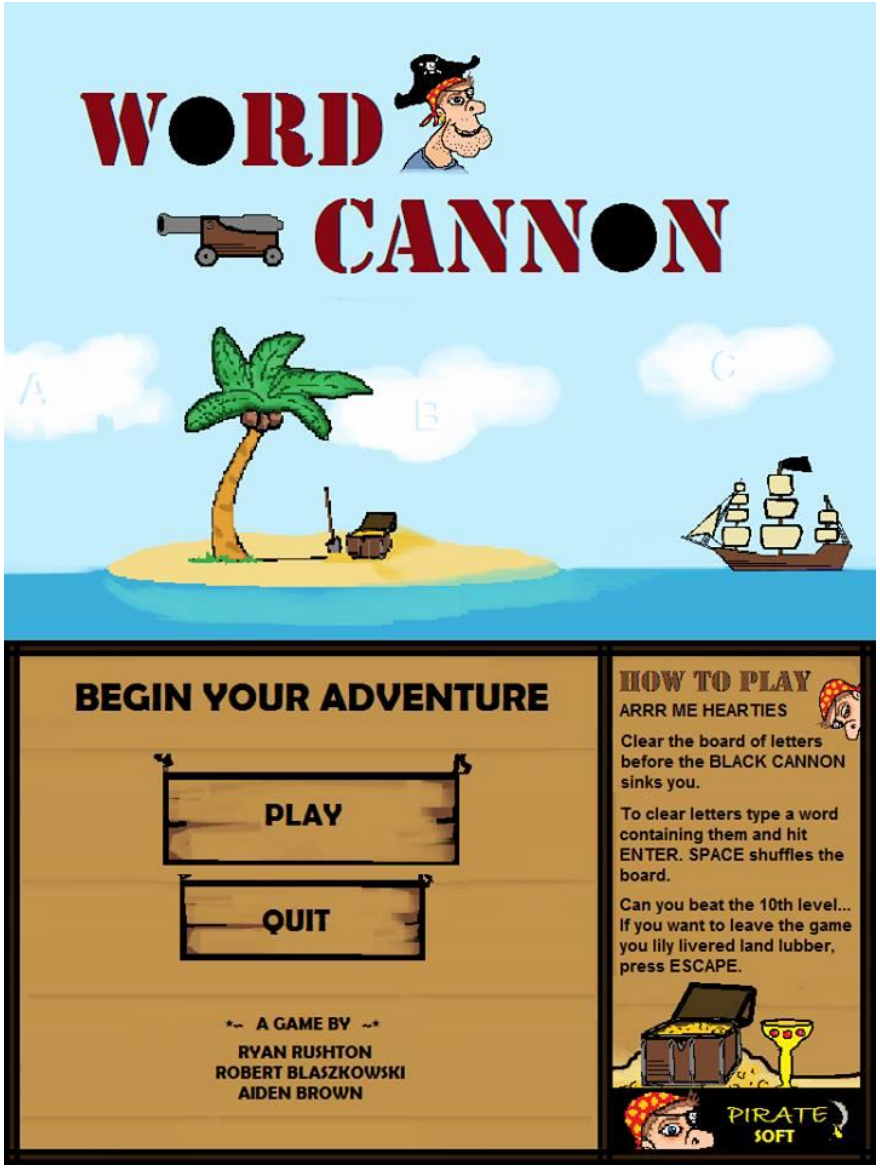
DESIGN

- SIMPLE
- ENGAGING
- COLOURFUL
- FLAVOURFUL
- CORRECT SIZE



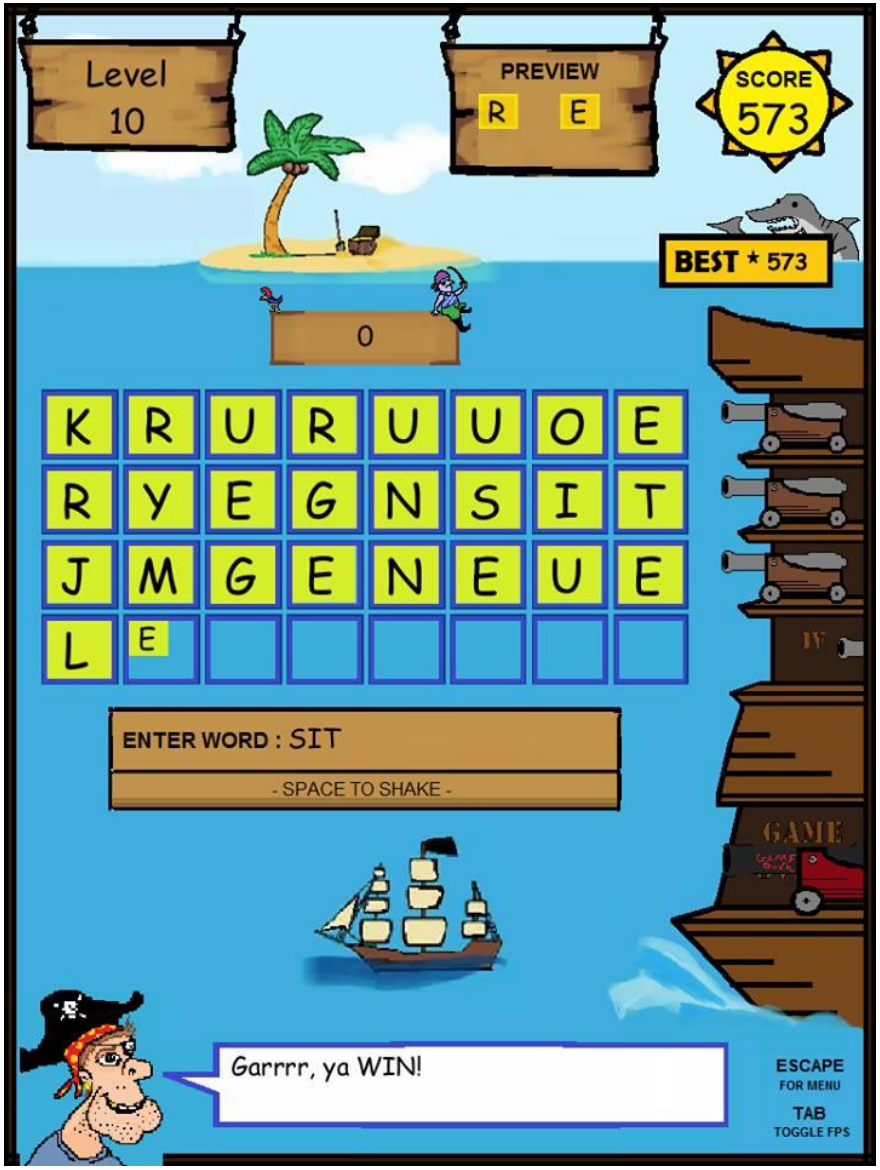
DESIGN

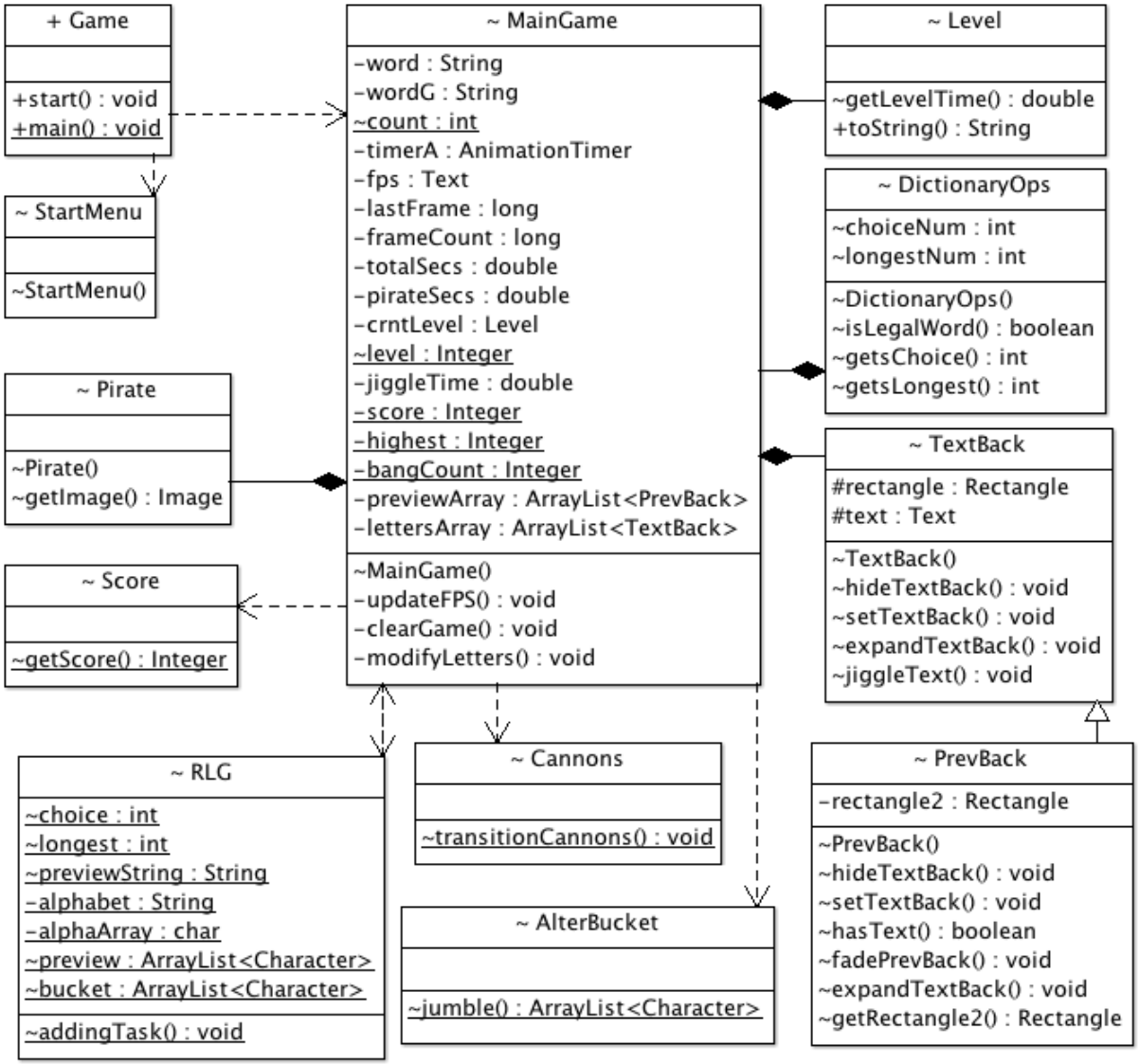
- Menu system
- Separate window
- Restart the game without leaving the application
- Keeps best score for current session



DESIGN

- Pirate Pete helps
- Interesting animation
- Original artwork
- Original Music
- High Score System




+ Game
+start() : void
+main() : void
~ StartMenu
~StartMenu()
~ MainGame
-word : String
-wordG : String
~count : int
-timerA : AnimationTimer
-fps : Text
-lastFrame : long
-frameCount : long
-totalSecs : double
-pirateSecs : double
-crntLevel : Level
~level : Integer
-jiggleTime : double
-score : Integer
-highest : Integer
-bangCount : Integer
-previewArray : ArrayList<PrevBack>
-lettersArray : ArrayList<TextBack>
~MainGame()
-updateFPS() : void
-clearGame() : void
-modifyLetters() : void
~ Level
~getLevelTime() : double
+toString() : String
~ DictionaryOps
~choiceNum : int
~longestNum : int
~DictionaryOps()
~isLegalWord() : boolean
~getsChoice() : int
~getsLongest() : int
~ Pirate
~Pirate()
~getImage() : Image
~ TextBack
#rectangle : Rectangle
#text : Text
~TextBack()
~hideTextBack() : void
~setTextBack() : void
~expandTextBack() : void
~jiggleText() : void
~ Score
~getScore() : Integer
~ RLG
~choice : int
~longest : int
~previewString : String
-alphabet : String
-alphaArray : char
~preview : ArrayList<Character>
~bucket : ArrayList<Character>
~addingTask() : void
~ Cannons
~transitionCannons() : void
~ AlterBucket
~jumble() : ArrayList<Character>
~ PrevBack
-rectangle2 : Rectangle
~PrevBack()
~hideTextBack() : void
~setTextBack() : void
~hasText() : boolean
~fadePrevBack() : void
~expandTextBack() : void
~getRectangle2() : Rectangle

TextBack

```java
class TextBack extends Group{

    protected Rectangle rectangle;
    protected Text text;

    /*Creates a yellow rectangle with text centered in it*/
    TextBack(int x, int y){
        this.rectangle = new Rectangle(57,51);
        this.rectangle.setFill(Color.YELLOW);
        this.rectangle.setOpacity(0.8);
        this.rectangle.setX(x);
        this.rectangle.setY(y);
        this.getChildren().add(rectangle);
        this.text = new Text();
        this.text.setX(x);
        this.text.setY(y);
        this.text.setFill(Color.BLACK);
        this.text.setTextAlignment(TextAlignment.CENTER);
        this.text.setTextOrigin(VPos.TOP);
        this.text.setOpacity(1.0);
        this.getChildren().add(text);
    }
```

```java
    /*This makes the TextBack's expand out into the screen when they are added to the bucket*/
    void expandTextBack(double secs, Level level){
        double fadeTime = Math.exp((secs-level.getLevelTime()));
        rectangle.setWidth(57*( fadeTime));
        rectangle.setHeight(51*( fadeTime));
        text.setWrappingWidth(rectangle.getWidth());
        text.setX(rectangle.getX()+rectangle.getWidth()/2-fadeTime*31);
        text.setY(rectangle.getY()+rectangle.getHeight()/2-fadeTime*30);
        text.setFont(Font.font("Comic Sans MS", FontWeight.NORMAL, 42*(fadeTime)));
    }

    /*This makes the letters bounce around in the yellow rectangle.
     *Rather then defining boundaries and having if statements to change direction
     *it uses the fact that a sin function is bounded by 1 and -1 to keep the
     *letters straying too far from center.
     *Also note the exponential decay is used to make them settle back down to center.*/
    void jiggleText(double secs, Level level){
        double x1 = rectangle.getX()+rectangle.getWidth()/2-31;
        double y1 = rectangle.getY()+rectangle.getHeight()/2-30;
        double fadeTime =  Math.exp((-secs));

        text.setX(x1 + 5*fadeTime*Math.sin(30*secs)*Math.random());
        text.setY(y1 + 5*fadeTime*Math.sin(30*secs)*Math.random());
    }
```

MainGame

- Uses Event Handlers.
- Uses Animation Timers.
- Incorporates all other classes.

Random Letter Generator

```
static int choice;
static int longest;
static String previewString;
private static String alphabet = "aeioubcdfghjklmnpqrstvwxyzeeeeeeeeeeeeaaaaaaaaiiiiiiiiiooooooonnnnnrrrrrtttttlllsssuuudddggbcmpfhvwy";;
private static char[] alphaArray= alphabet.toCharArray();
static ArrayList<Character> preview= new ArrayList<Character>();
static ArrayList<Character> bucket= new ArrayList<Character>();
```

- Followed the letter distribution of Scrabble
  - Provides a stochastic implementation of letter delivery.
- More natural letter occurrence

Random Letter Generator

```
if(preview.size()==2 && preview.get(0) == preview.get(1)){
    ArrayList<Character> reducedList = new ArrayList<Character>();
    char[] reduced = alphabet.toCharArray();
    for(int i = 0; i < reduced.length; i++){
        reducedList.add(reduced[i]);
    }
    while(reducedList.contains(preview.get(1))){
        reducedList.remove(preview.get(1));
    }
    preview.remove(1);
    preview.add(reducedList.get(rand.nextInt(reducedList.size())));
}
```

```
if(bucket.isEmpty()||
        bucket.contains(alphaArray[0])||
        bucket.contains(alphaArray[1])||
        bucket.contains(alphaArray[2])||
        bucket.contains(alphaArray[3])||
        bucket.contains(alphaArray[4])||
        preview.contains(alphaArray[0])||
        preview.contains(alphaArray[1])||
        preview.contains(alphaArray[2])||
        preview.contains(alphaArray[3])||
        preview.contains(alphaArray[4])){
    preview.add(randlet);
}
```

- Makes sure that there are vowels within your RLG
- Makes sure that letters are not repeated

Scoring

```
for(int i =0; i< word.length(); i++){
    if(onepoint.contains(word.charAt(i))){
        currentscore = currentscore + 1;
    }
    else if(twopoint.contains(word.charAt(i))){
        currentscore = currentscore + 2;
    }
    else if(threepoint.contains(word.charAt(i))){
        currentscore = currentscore + 4;
    }
    else if(fourpoint.contains(word.charAt(i))){
        currentscore = currentscore + 4;
    }
    else if(fivepoint.contains(word.charAt(i))){
        currentscore = currentscore + 5;
    }
    else if(eightpoint.contains(word.charAt(i))){
        currentscore = currentscore + 8;
    }
    else if(tenpoint.contains(word.charAt(i))){
        currentscore = currentscore + 10;
```

```
String onestring = "eaiounrtlsu";
ArrayList<Character> onepoint = new ArrayList<Character>();
for(int i = 0; i < onestring.length(); i ++){
    onepoint.add(onestring.charAt(i));
}

String twostring = "dg";
ArrayList<Character> twopoint = new ArrayList<Character>();
for(int i = 0; i < twostring.length(); i ++){
    twopoint.add(twostring.charAt(i));
}

String threestring = "bcmp";
ArrayList<Character> threepoint = new ArrayList<Character>();
for(int i = 0; i < threestring.length(); i ++){
    threepoint.add(threestring.charAt(i));
}

String fourstring = "fhvwy";
ArrayList<Character> fourpoint = new ArrayList<Character>();
for(int i = 0; i < fourstring.length(); i ++){
    fourpoint.add(fourstring.charAt(i));
}

String fivestring = "k";
ArrayList<Character> fivepoint = new ArrayList<Character>();
for(int i = 0; i < fivestring.length(); i ++){
    fivepoint.add(fivestring.charAt(i));
}

String eightstring = "jx";
ArrayList<Character> eightpoint = new ArrayList<Character>();
for(int i = 0; i < eightstring.length(); i ++){
    eightpoint.add(eightstring.charAt(i));
}

String tenstring = "qz";
ArrayList<Character> tenpoint = new ArrayList<Character>();
for(int i = 0; i < tenstring.length(); i ++){
```

- Uses a scoring system similar to Scrabble.
- Scoring is competitive. Players rewarded for advanced words
- High Score gives replay value.

The Brains of Pete

- Displays the Choice and Longest functions

```
DictionaryOps operation = new DictionaryOps();
choice = operation.getsChoice(bucket);
longest  = operation.getsLongest(bucket);
previewString = preview.toString();

final Text stats = new Text(200,944, ("Pick one of 0 words, the longest is 0 ya scurvy dog"));
stats.setFont(Font.font("Comic Sans MS", FontWeight.NORMAL, 18));
stats.setFill(Color.BLUE);
stats.setWrappingWidth(390);
stats.setOpacity(1.0);
root.getChildren().add(stats);
```

```
int getsLongest (ArrayList<Character> bucketlist) {
    longestNum = 0;

    InputStream dict = getClass().getResourceAsStream("dictionary.txt");
    BufferedReader dictionary = new BufferedReader(new InputStreamReader(dict));
    String line;
    try {
        while((line = dictionary.readLine()) != null){

            ArrayList<Character> wordlist = new ArrayList<Character>();

            for(int i = 0; i < line.length(); i++){
                wordlist.add(line.charAt(i));
            }

            if(bucketlist.containsAll(wordlist)){
                for(int i = 0; i < bucketlist.size(); i++){
                    wordlist.remove(bucketlist.get(i));
                }
                if(wordlist.size() == 0 && line.length()> longestNum){
                    longestNum = line.length();
                }
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    try {
        dictionary.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return longestNum;
}
```

```
int getsChoice (ArrayList<Character> bucketlist){
    choiceNum = 0;

    /*Scans in the dictionary file*/
    InputStream dict = getClass().getResourceAsStream("dictionary.txt");
    BufferedReader dictionary = new BufferedReader(new InputStreamReader(dict));
    String line;
    /*While the file has a new line it adds each character of the word in the next
    try {
        while((line = dictionary.readLine()) != null){
            ArrayList<Character> dictionaryList = new ArrayList<Character>();

            for(int i = 0; i < line.length(); i++){
                dictionaryList.add(line.charAt(i));
            }

            /*If the ArrayList above is contained within the bucket we remove each
             * the ArrayList (note if the ArrayList does not contain a char it doe
             * ArrayList is empty after this we know that each letter of the bucke
             * we can increment the number of choices by 1
             * After this has been done for each word in the file we return choice
            if(bucketlist.containsAll(dictionaryList)){
                for(int i = 0; i < bucketlist.size(); i++){
                    dictionaryList.remove(bucketlist.get(i));
                }
                if(dictionaryList.isEmpty()){
                    choiceNum++;
                }
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    try {
        dictionary.close();
```

Project features

- Correct sizing within JavaFX
- RLG is not simply random
- Modified scoring systems
- Level system
- High score system
- Menu system
- Engaging game play with animation, music and a consentient themes
- Runs at ~60fps