Using JavaFX to create a Simple RPG Game

Felix Gabriel Montañez
College of Engineering
BS CPE
Manila,Philippines
felixgabrielmontanez@gmail.com

JavaFX is a powerful tool in developing applications with a Graphical user interface for better user interface and intractability. Given JavaFX's tools, creating a game will be very efficient as JavaFX has powerful objects that can be efficient for player interaction. RPGs or Role-Playing games is one of the most popular game genres because of the storytelling, fantasy elements and player choice. Creating a simple RPG game using JavaFX is very efficient because of the tools that JavaFX has in creating the atmosphere of storytelling and interaction for player choice to create an RPG themed game.

Keywords—Role-Playing game, JavaFX, fantasy, GUI, player interaction

I. INTRODUCTION (*HEADING 1*)

JavaFX is one of the best ways in implementing GUIs in java. JavaFX is a software platform used in creating desktop applications as well as rich internet applications or (RIAs). It was intended to replace swing as the standard GUI library for JAVA SE, but both are reported to be included for the future [1].

Role Playing Games are one of the most popular games genres of all time. It has been favored because it is in a fictional setting where games can be as creative as they can be because of the lack of the restriction of realism. Players take responsibility in acting within the story through acting out their character and deciding how they would decide in the situation giving a character development [2]. Role playing games are often set in a fantasy setting with a variety of creatures and characters that may be inspired from folklore or mythology with inclusion of magic.

The genre of Role-Playing Games or RPGs have evolved throughout the years. Its original form was initially in tabletop form. Tabletop form is where the game is usually played in a flat table where players discuss and their action through speech and spoken commentary. It also evolved into live Action role-playing games where the players themselves act their player's actions [3].

Role-playing games gaining popularity has shifted them to video games with massively multiplayer online role-playing games (MMORPGs) where a large number of players online come together to form a group and play together completing the game's objectives. Role-playing games also include single-player role-playing video games [4].

JavaFX can be a very useful tool in creating games because of the tools it has and the ease in creating layouts with the help and integration of Scene Builder. The game will take on elements of an RPG game like a fantasy setting with character choice and a turn-based gameplay.

II. METHODOLOGY

The Program will have three parts, the introduction, main gameplay and the outro. The game will revolve around finding monsters in location and beating them one by one using weapons that your player has. The program user interface will have two parts. The main screen for the graphics and the lower dialog box for text.

Before the game proper, there should be a dialog box that will ask for the user's name for the program to properly access the user's name.

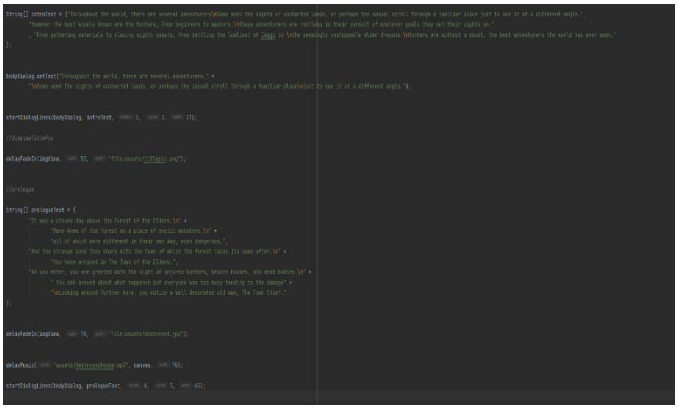
```
TextInputDialog textinput = new TextInputDialog();
Stage stage = (Stage) textinput.getDialogPane().getScene().getWindow();
stage.getIcons().add(new Image( s: "file:assets/dragon.png")); // To add an icon
textinput.setTitle("Enter name");

textinput.setHeaderText("Enter your name ");

textinput.showAndWait();

textinput.getDialogPane().setContentText("Enter name:");
TextField input = textinput.getEditor();

if (input.getText() != null) {
    userName = input.getText();
    System.out.println("User is " + userName);

} else {
    System.out.println("no input");
}
```

```
/**
 * Adds goto buttons to the map
 * @param firstPane canvas
 *
 */
private void createButtonsForMap(AnchorPane firstPane) {

    PauseTransition pauseTransition = new PauseTransition(Duration.seconds(170));
    pauseTransition.setOnFinished(e -> {

        gotoFR = createButton( btnText: "Flooded Ruins", family: "Centaur", textSize: 12, x: 613, y: 282);
        gotoHH = createButton( btnText: "Highland Hills", family: "Centaur", textSize: 12, x: 280, y: 414);
        gotoOWT = createButton( btnText: "Old Witch's Tower", family: "Centaur", textSize: 12, x: 878, y: 473);
        gotoRW = createButton( btnText: "Ragged Wood", family: "Centaur", textSize: 12, x: 769, y: 47);
        gotoFR.setOnAction((actionEvent -> checkButton( location: "flooded ruins")));
        gotoHH.setOnAction((actionEvent -> checkButton( location: "highland hills")));
        gotoOWT.setOnAction((actionEvent -> checkButton( location: "old witches tower")));
        gotoRW.setOnAction((actionEvent -> checkButton( location: "ragged wood")));


        Group btnGroup = new Group();
        btnGroup.getChildren().addAll(gotoFR, gotoOWT, gotoHH, gotoRW);


        firstPane.getChildren().add(btnGroup);

    });
    pauseTransition.play();

}
```

A. *Introduction*

Like modern Role -playing games, introduction serves to introduce the player to the world and the atmosphere of the game. Here the game tells the user about the lore of the game or the background story of the game so that the user knows relevant things about the game's story. An intro class was created which will facilitate the intro sequence of the game. In this class will be created essential methods to facilitate an intro sequence like adding an image, fade in and out of image and a delay music function. Main game functions are also put into another class gameFunctions where the program will store all the methods which will be used by all other classes and use inheritance for ease of use. Inside the intro class, there will be the array of sentences which will be the narration of the intro. With playmusic method and start dialog lines method it starts the intro inside startintromethod. This is where the game displays information about the world and the game in general so that the player can have information going into the game . After the title sequence it will proceed to the prologue which will tell the player the current problems and the present situation in the game. Which will also use the startdialoglines method form game functions and playmusic functions.

At the end of the intro the game will display the main map of the game which will be a main interaction for the player. Here there will be buttons to go to a specific location where the monster can be found. At the end of the prologue it will display the current quest the user has and set its quest accordingly.



B. *Main Gameplay*

There are 4 main monsters each one has their own statistics and are more powerful than the previous one. The player has a standard set of stats and 4 weapons and 3 potions. These 4 weapons will be used to attack the monsters while the potion will help the player if needed extra health. Potions give an additional 50 hp or health points to the player.

Each weapon will be effective for only one monster so during the main gameplay, the user must use the most effective weapon. The monster has only one attack but has varying percent multipliers. These multipliers have their respective chance percent which will be dictated by the probability class. The monster will also have an evasion stat which will help the monster avoid damage. Each monster will have a higher stat percentage.

```
/**Gets number probability
 *
 * @param arr Selection array
 * @param freq Respective Frequency
 * @return double result
 */
public double getProbability(int[] arr, int[] freq){

//      int[] arr;
//      arr = new int[]{1, 2, 3, 4};
//      int[] freq = {5, 5, 1, 2};
    int  n = arr.length;

    return myRand(arr, freq, n);
}
```

Both the monsters and the player will have their respective classes. These classes will contain their stats and their attacks. In the player class, it will contain the critical multiplier for the player's base damage, name of the user, hp, base damage, potion count, miss chance and frequency. The monster class will be an abstract class and each monster will have their own separate classes. The abstract monster class will contain the hp, dmg, critical multiplier, name, musicpath which is the sound the monster makes, critical, and freq. it will also contain

the method for attack which uses the probability object to get the critical multiplier and will return the damage. Each monster class will have a constructor which sets the stats respectively based on the monster. The player will also have an attack function which will process the button inputs. It will first check the current monster and set the probability chances that the player will have respective to the monster.

```
public abstract class Monster extends gameFunctions {
        double hp;
        double dmg;
        double criticalMultiplier;
        String name;
        String difficulty;
        String musicPath;
        /**
         * Critical percentages
         */
        int[] critical;
        /**
         * REspective frequencies
         */
        int[] freq;



        //constructor
        public Monster(String difficulty) { this.difficulty = difficulty; }

        /**moster attacks and returns a multiplier
         *
         * @param canvas Grid pane canvas
         * @return double multiplier
         */
        public double attack(GridPane canvas) {
                Probability probObj = new Probability();
                playMusic(musicPath,canvas);
                System.out.println("Entered main attack, dmg is "+ dmg);
                criticalMultiplier = probObj.getProbability(critical,freq);
                return dmg*(criticalMultiplier*0.01 + 1);
        }
```

```
//miss chance
System.out.println("Weapon is " + weapon +"\nMonster is " + quest);
switch (quest) {
    case "lyngruel" -> {
        missChance = new int[]{0, 1};
        missFrequency = new int[]{20, 80};
    }
    case "forrek" -> {
        missChance = new int[]{0, 1};
        missFrequency = new int[]{30, 70};
    }
    case "merl" -> {
        missChance = new int[]{0, 1};
        missFrequency = new int[]{50, 50};
    }
    case "azazel" -> {
        missChance = new int[]{0, 1};
        missFrequency = new int[]{60, 40};
    }
}
```

This sequence will happen with each monster. When the player enters the correct location based on the monster, it will enter the battle phase.

```
/**
 * Starts lvgruel battle
 */
private void startLvgruelBattle() {

    //play battle music
    battleMusic = playMusic( musicPath: "assets/battle1.mp3", canvas);
    //create objects
    player = new Player(userName, difficulty);
    currentMonster = new lyngruel(difficulty);

            setStatLabel(player, currentMonster);

    String[] monsterDialog = {
            "What are you doing here??\nHow did you find me?", ""
    };

    String[] name = {
            "Lyngruel the Giant",
    };

    fadeInImage(imgView, path: "file:assets/bg1.jpg", time: 5);
    bodyDialog.setText("You have found Lyngruel the Giant!");
    setMapButtonVisibility(false);
    monsterImgView.setLayoutX(347);
    monsterImgView.setLayoutY(95);
    monsterImgView.setFitWidth(385);
    monsterImgView.setFitHeight(427);

    firstPane.getChildren().add(monsterImgView);
    fadeInImage(monsterImgView, path: "file:assets/lyngruel.png", time: 5);

    startDialogLinesWithName(bodyDialog, nameDialog, monsterDialog, name, cps: 1, wpl: 1, sequenceTime: 1);

    createAttackButtons(player);


}
```

During the battle the main screen will change displaying the monster with the background of the location. On the top left corner will be the monster's stats with its name, hp and dmg. On the bottom left corner will be the player's stats and weapons. The stats will contain the player's name, hp, dmg, and potion count. The buttons will contain the player's 4 weapons and the potion. If the player loses (player's health is 0 or lower), the prompt will say that you have failed the quest and will exit the program. If the player won, the game will go back to menu and update the quest.

```
/**
 * Goes back to main quest map
 */
private void backToMenu() {
    musicFadeIn(ambient);
    //hide gameplay objects(monster, attack buttons,stats label )
    monsterImgView.setVisible(false);
    setAttackButtonVisibility(false);
    setStatLabelVisibility(false);
    //change game map
    fadeInImage(imgView, path: "file:assets/map.jpg", time: 13);

    PauseTransition pauseTransition = new PauseTransition(Duration.seconds(33));
    pauseTransition.setOnFinished(e -> setMapButtonVisibility(true));
    pauseTransition.play();



}
```

After displaying the updated quest, the player can now enter into the next quest. This will continue on until the player manages to defeat the final boss.

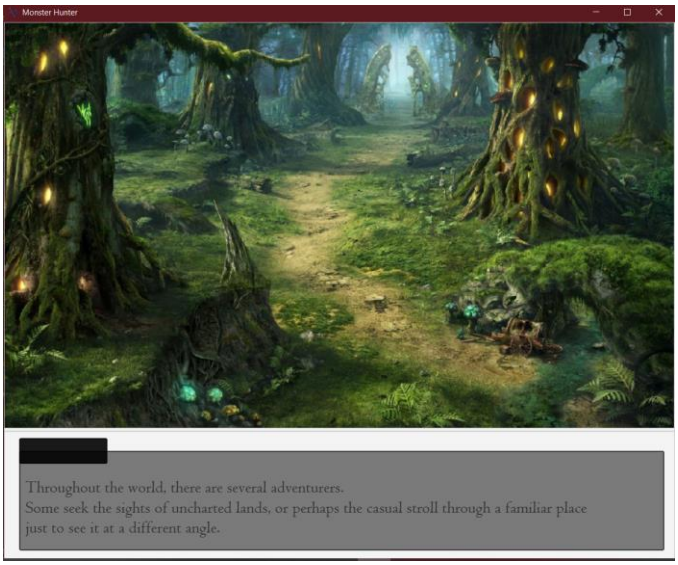
C. Outro

After beating the final boss, the outro will be played. Similar to the intro class, the outro will display an end sequence and will also display the credits. In outro class will be the start outro function which will initiate the outro sequence. The game will display the ending dialogue which will also relate to the lore of the game. This will also use the same function used for dialog lines in intro. After the end dialog, it will display the title image and proceed to the credits. The credits will be in a black rectangle with half opacity.

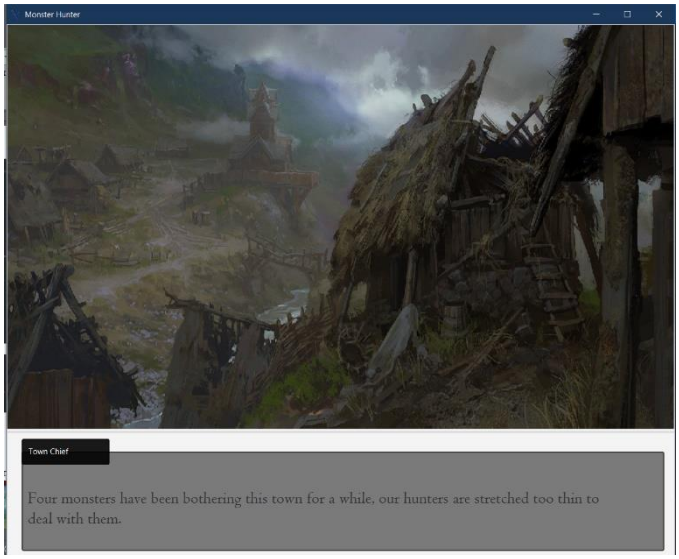
```
delayFadeIn(imgView, wait: 2, path: "file:assets/destroyed.jpg");

startDialogLinesWithName(bodyDialog,nameDialog,body,name, cycle: 6, wait: 2);

//display title pic
delayFadeIn(imgView, wait: 62, path: "file:assets/end.png");

//play credits

//create bg for text

Rectangle bg = new Rectangle( v: 252, v1: 31, v2: 494, v3: 541);
firstPane.getChildren().add(bg);
Color c = Color.web("rgba(0,0,0,0.5)");
bg.setFill(c);

bg.setVisible(false);


PauseTransition pauseTransition = new PauseTransition(Duration.seconds(83));
pauseTransition.setOnFinished(e -> bg.setVisible(true));
pauseTransition.play();


delayFadeIn(imgView, wait: 80, path: "file:assets/skyrimbg.jpg");




Label creditLabel = new Label();
creditLabel.setLayoutX(302);
creditLabel.setLayoutY(43);
creditLabel.setPrefSize( v: 393, v1: 459);

creditLabel.setTextFill(Color.WHITE);
```

```
creditLabel.setFont(Font.font( s: "Centaur", FontWeight.BOLD, v: 15));

firstPane.getChildren().add(creditLabel);

startDialogLines(creditLabel,credit, index: 0, cycle: 2, wait: 75);
```

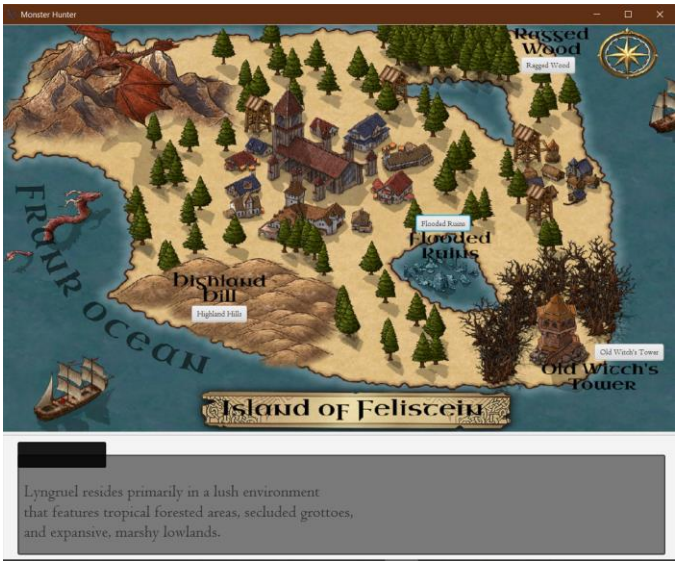
III. RESULTS



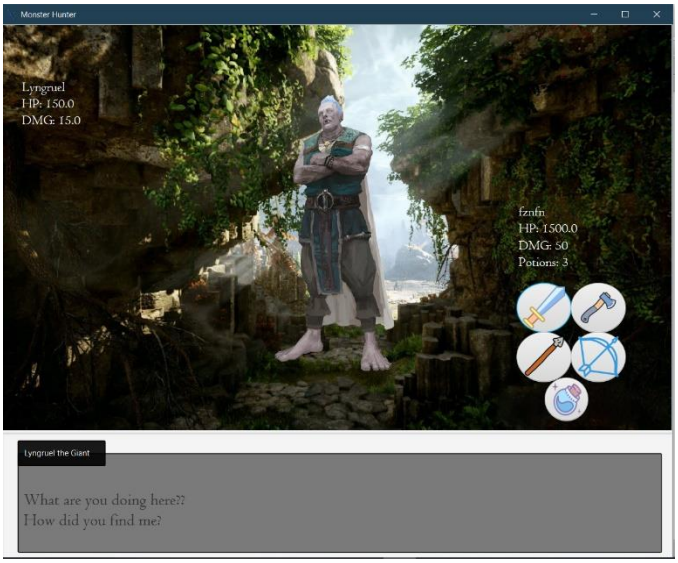
After the text prompt to get name, the intro class will start the intro. The intro class played the intro sequence very smoothly with a great transition using fadeTransition to transition to the main map.



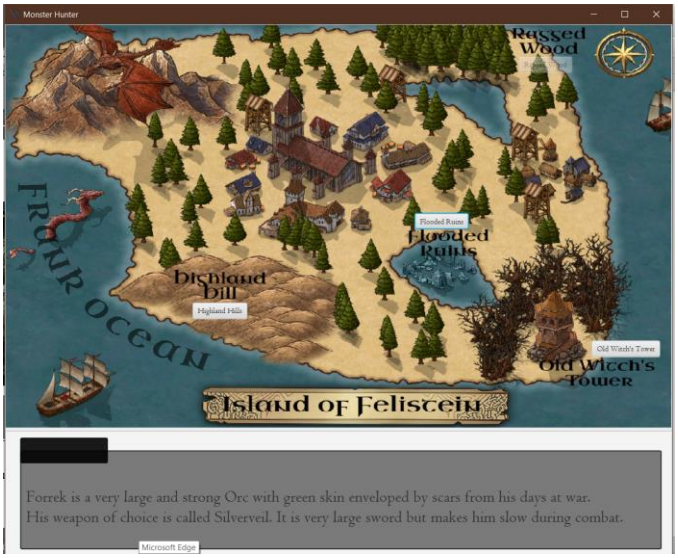
Using button object to create a go to option for the map made for an easier way to activate the battle sequence.



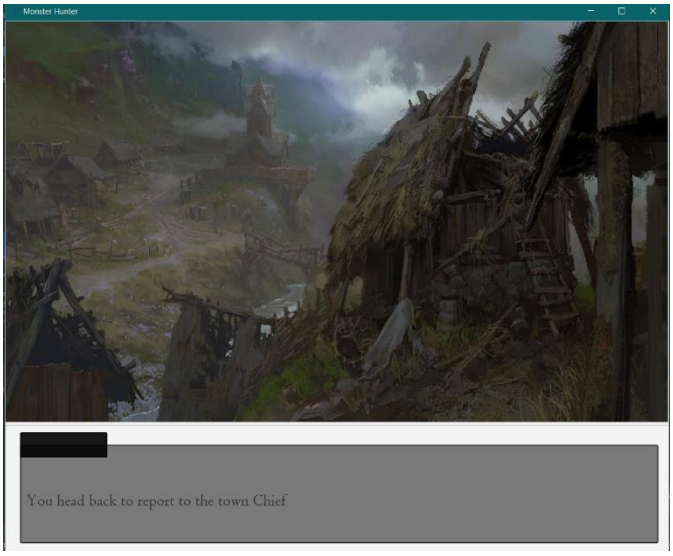
The main battle sequence proved to be exciting with the use of probability leading to the variety of possible turns during a battle. playmusic function was also very effective in creating a tense atmosphere for the battle.



Upon beating the monster, the game will automatically go back to the map and update the quest. The buttons will also be updated.

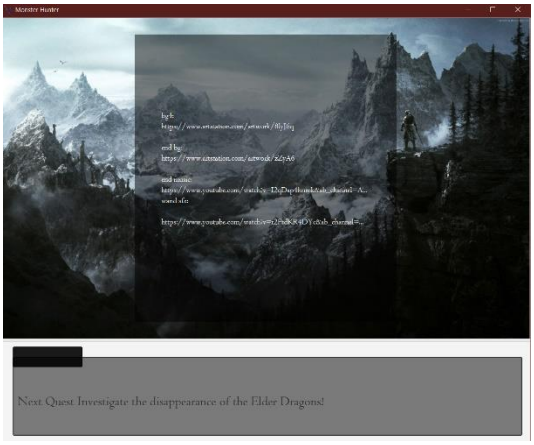


Upon beating all the monsters with the final boss, the outro class will automatically call its functions to start the outro sequence.



This will change the background and display the opening location and start the end dialog lines. Afterwards it will display the end title screen with the use of pauseTransition.

After the end title it will show the credits of all the files used.



IV. Conclusion

JavaFX is a very powerful and effective tool in creating multiple apps for various reason. It has tools like buttons, animations and other objects that makes it easier to create efficient graphics. Interactive mediums like video games are very effective to be created using JavaFX because of the tools available. RPGs are one of the most exciting game genres and a favorite for many people that plays games. Not only is exciting but the lore, story and exciting mechanics of the game make it really enjoyable and entertaining. The game proved to be very challenging especially in timing the pause Transitions for the dialogues and the fade transitions for the images as well. Creating a gameplay mechanic that was both engaging and with a level of difficulty also proved to be challenging but workable with OOP. Creating classes for each monster and for the player and adding move sets were also challenging as the player needs to interact with the monster. The probability aspect of the game was also challenging but very fruitful in creating a more exciting gameplay experience with a chance of missing and dealing critical damage. As a simple RPG game this proved very effective and enjoyable given the use of transition for images and music, the story within the game and the randomness of the gameplay. The story, gameplay and settings are unique to this game and it serves to be a pocket experience for users that want to start with an RPG game through a simple yet exciting introduction to the genre of RPG.

Acknowledgment *(Heading 5)*

I would like to thank Sir John Anthony Jose, for guiding the project and throughout the whole course and for effectively teaching the concepts of java and OOP effectively even with online learning.

I would like to thank Bethesda Softworks and Todd Howard for leading the team in creating the elder scroll series which was a big inspiration for this project

I would also like to thank Elijah Ng for helping creating this project in a python console program format who primarily wrote the dialog of the game.

References

[1] "JavaFX FAQ". Oracle.com. Retrieved 2020-03-10.

[2] Grouling, J. (2010). The creation of narrative in tabletop role-playing games. Jefferson, NC: McFarland &.

[3] Tychsen, A. (2006). Live Action Role-Playing Games: Control, Communication, Storytelling, and MMORPG Similarities. *Games and Culture, 1*(3), 252-275. doi:10.1177/1555412006290445

[4] Crawford, C. (2003). *Chris Crawford on game design*. Indianapolis, IN: New Riders.

[5] Skeleton of a Town - Icewind Dale 2 soundtrack [Advertisement]. (2009, June 14). Retrieved from https://www.youtube.com/watch?v=aEqDtbC3bL8&t=18s&ab_channel=HalcyonSyzygy

[6] Pervukhin, A. (2014). Medieval Village [Digital image]. Retrieved from https://www.artstation.com/artwork/zavw

[7] I. (n.d.). Dragon [Digital image]. Retrieved from https://www.flaticon.com/free-icon/dragon_2602964

[8] TES V Skyrim Soundtrack. (n.d.). *Silence Unbroken*.

[9] Forrest [Digital image]. (n.d.). Retrieved from https://wallhere.com/en/wallpaper/712853

[10] Cloud Giant [Digital image]. (n.d.). Retrieved from https://criticalrole.fandom.com/wiki/Cloud_Giant

[11] Random number generator in arbitrary probability distribution fashion. (2020, February 28). Retrieved October 05, 2020, from https://www.geeksforgeeks.org/random-number-generator-in-arbitrary-probability-distribution-fashion/

[12] GFX sounds. (2020). *Bow and Arrow - Sound Effect [HD]*. Retrieved from https://www.youtube.com/watch?v=kiKOo6HQFRw&ab_channel=GFXSounds

[13] HD Sound Effects. (2020). *Sword Slash (Sound Effects)*. Retrieved 2020, from https://www.youtube.com/watch?v=c_RjEKLacwo&ab_channel=HDSoundEffects

[14] The Elder Scrolls V. (2011). *Skyrim - Combat #1*. Retrieved 2020, from https://www.youtube.com/watch?v=Dicrp9rYfmU&list=PL6F112y8X_M5F7_DFgeIbSI5Tb9Cx7CA4&index=1&ab_channel=Dagre

[15] Orc Barbarian [Digital image]. (2020). Retrieved 2020, from https://www.pinterest.ru/pin/562950022164006755/?nic_v2=1a5PchXnE

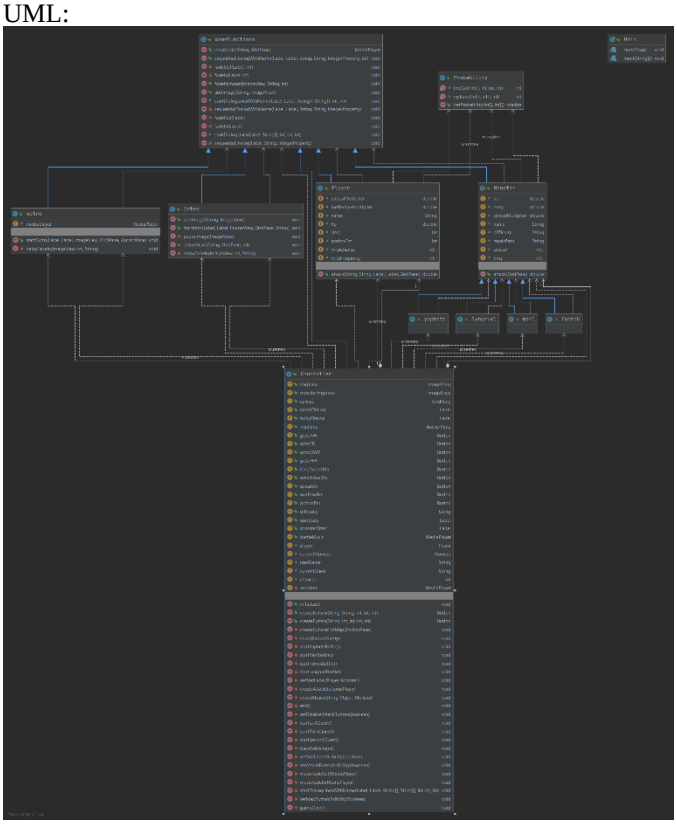
[16] Kay, V. (2018). SkyForge [Digital image]. Retrieved 2020, from https://www.artstation.com/artwork/0AGww

[17] Comentale, J. (2019). Three Pillars [Digital image]. Retrieved 2020, from https://www.artstation.com/artwork/8lyJ6q

[18] Sarrailh, S. (2018). Forest of Liars : Traveling with clouds [Digital image]. Retrieved 2020, from https://www.artstation.com/artwork/zZyA6

Appendix

UML:



CRC:

Controller	gameFunctions
• Controls the graphical user interface • facilitates the main game	• intro • outro • gameFunctions • player • Monster

Intro	gameFunctions
• Facilitates the intro	

Outro	gameFunctions
• Facilitates the outro	

gameFunctions	
• Contains the main game functions which will be used by most classes	• Controller • intro • outro • monster • outro • player

Player	gameFunctions
• Facilitates the user stats • Contains user attacks and weapons	• Controller • Monster

Abstract Monster	gameFunctions forrek, lyngruel,merl,yaphets
• Facilitates the monster stats • Facilitates the monster attacks	• Player • Controller

Probability	
• Gets the probability for monster attacks • Gets the probability for player hits	• monster • player

Main	Application
• contains the main function to run the program	• Controller