Bilkent University

Department of Computer Engineering

CS 319 - Object Oriented Software Engineering Project

Bullet Drop

Final Report

Group 2F

Ulaş İş, Alperen Erkek, Ömer Faruk Ergün, Abdullah Seçkin Özdil

Course Instructor : Bora Güngören

Table of Contents

1. Introduction **2**

2. Setup of The Game **2**

3. User Guide **3**

4. Changes Done During the Implementation Process **12**

5. Incompleted Parts **13**

1. Introduction

Bullet Drop is a 2D shooting game for desktop environment which is implemented by using Java language. The main goal is to hit the target with the gun without seeing it at first time when shooting is done. After first attempt, user can understand the location of target and can do his next shoot according to this. There are 4 different map options with different difficulties which is caused by the counts of the external forces that is affected to bullet. There are three levels for every map. User open this maps one by one. There are also 5 different weapon options which user can choose. User can open these weapons also one by one while he/she is passing the levels. These different weapons have different bullet speeds which directly affects their reaction to external force. The main goal of the game is to improve memorization skills of the players by forcing them to remember the location of target and the counts/effects of external forces.

2. Setup of The Game

In order to play game, user's computer must contain Java. After completing Java updates, user needs to download game folder from github. (Available: https://github.com/allpino/2F.Bullet-Drop). From that, to play the game you can just compile and run the code by using an IDE. For this, the user will need NetBeans IDE, Eclipse or Intellij. Since most of our game code is implemented by using Intellij, we can recommend Intellij. Except all of these, player doesn't need any helper program. Moreover, if you want to play the game without running the code, you can download jar file (link is given in README.md file) and run the game with that jar file. Don't forget that you need to copy the file into the project folder, i.e where README.md file is.

3. User Guide

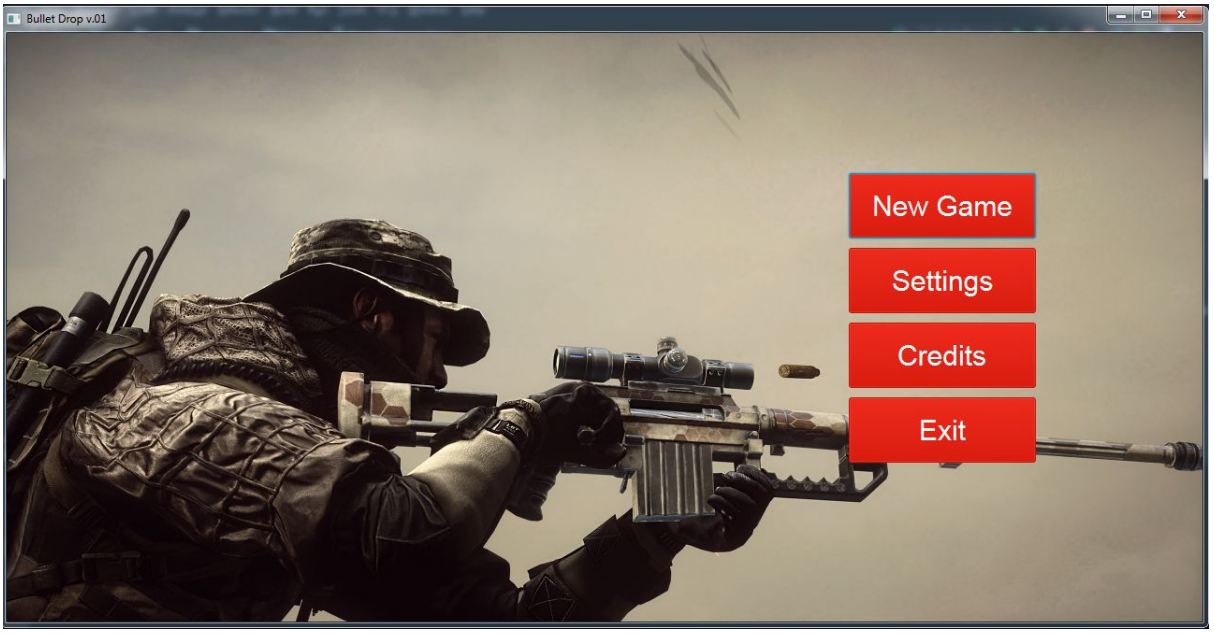

Figure 1.

When user first opens the game he will see directly Figure 1 which is the main menu of the game. In this menu, he/she can start to play game by clicking “New Game” button and go weapon selection screen or can go settings menu or can exit the game. Also, there is a ‘Credits’ button which shows the developer’s information. Both Credits and Settings buttons include ‘Back’ button. Users can turn back the main menu if they want.

Figure 2.

Here is the credits screen. Players can find the information about developers in this screen [Figure 2].

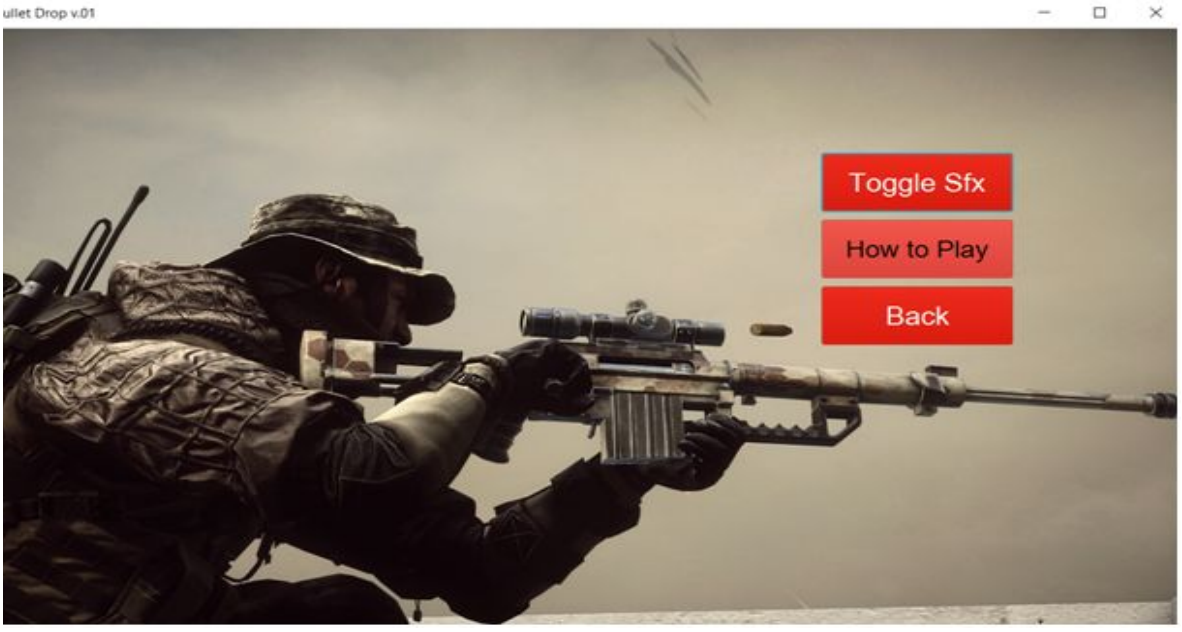

Figure 3.

When the player click on “settings” button in main menu he/she will see this screen [Figure 2]. The player can read the explanations about how to play the game, or turn back to main menu.

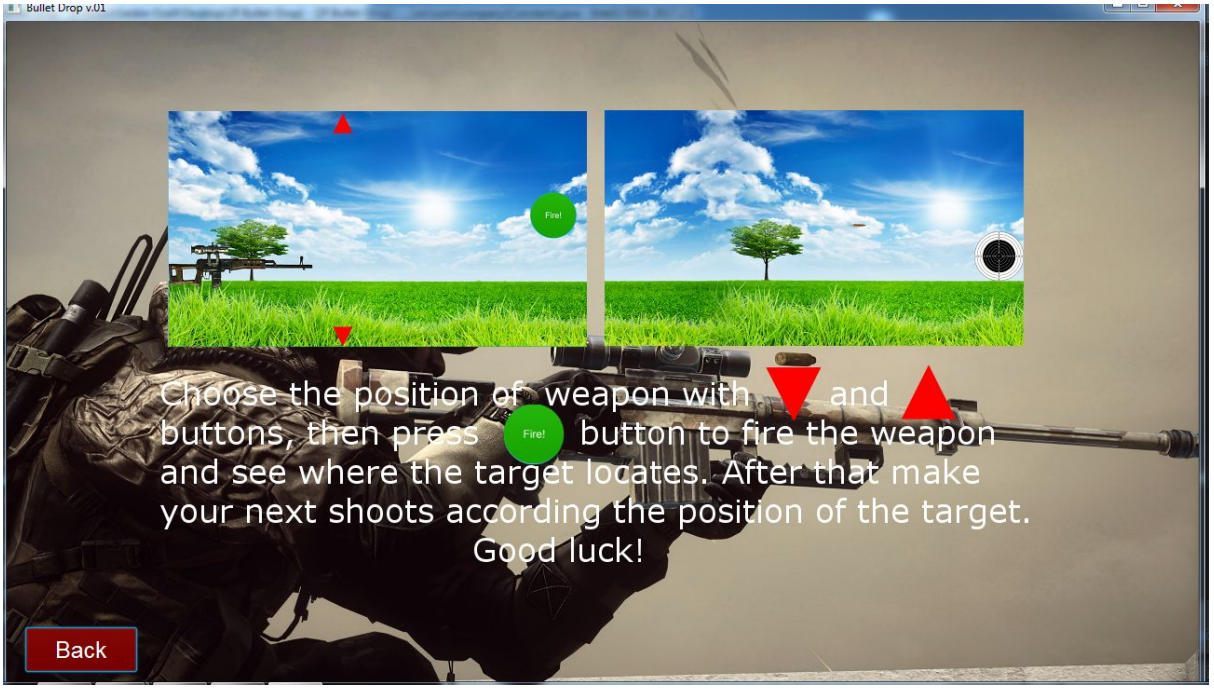

Figure 4.

Figure 3 shows ‘How to Play’ screen. When users want to know process of gaming and purpose, they can click ‘How to Play’ button on main menu and read the guidance of the game.

Figure 5.

When user click the new game button in main menu, this page is will appear to him [Figure 5]. The player can go back to main menu from the “back” button at left bottom of the screen. As a main duty of this page, the player choose the weapon that he/she want to play with. These weapons will be opened through the time player pass the levels. By clicking the next/previous buttons player can change it. Below 4 picture you can see different weapon options in this page.

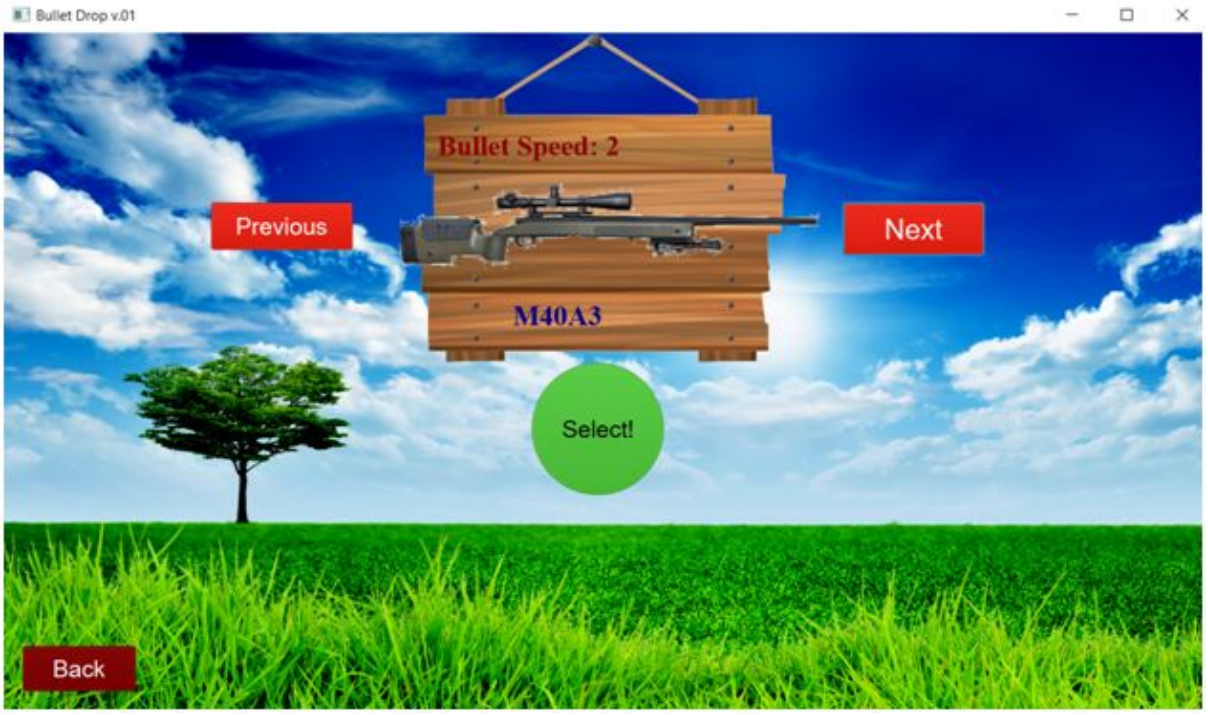

Figure 6.

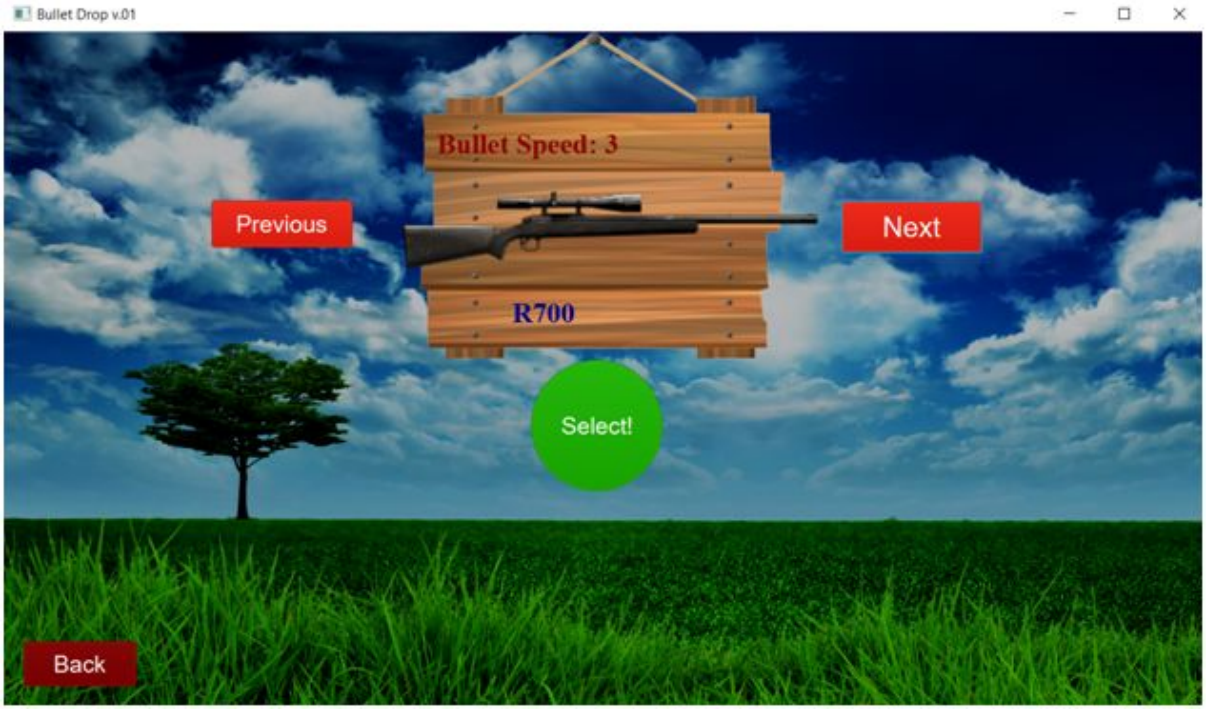

Figure 7.

Figure 8.

Figure 9.

After clicking the “select” button in weapon selection page, the player faces with this page to determine the position of the weapon [Figure 6,7,8 and 9].

Figure 10.

After the choice of position he makes the first shoot and see the position of the target [Figure 10 and 11]. Then player makes the next shoots according to this target position to hit and success.

Figure 11.

After the player shoot, he/she will see the result screen that provide an option to try again if shoot was a failure or give an option to pass to next page if shoot was successful [Figure 12 and 13]. In both case the player can also exit and turn back to main menu.

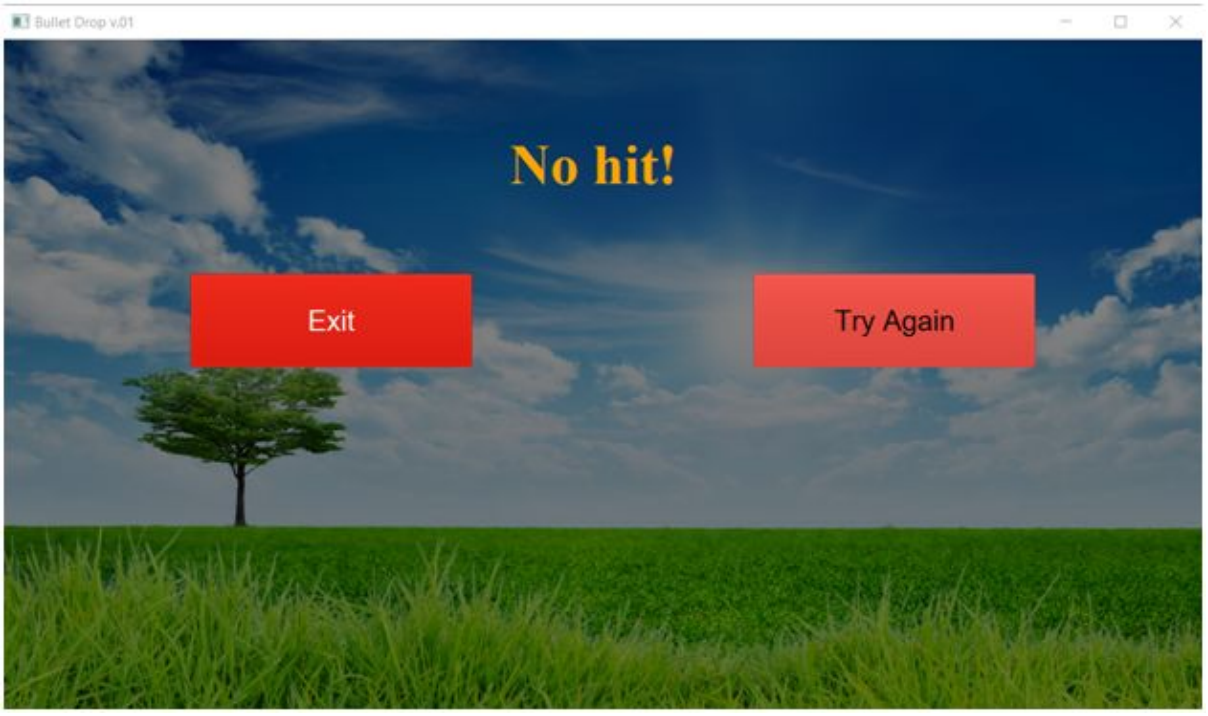

Figure 12.

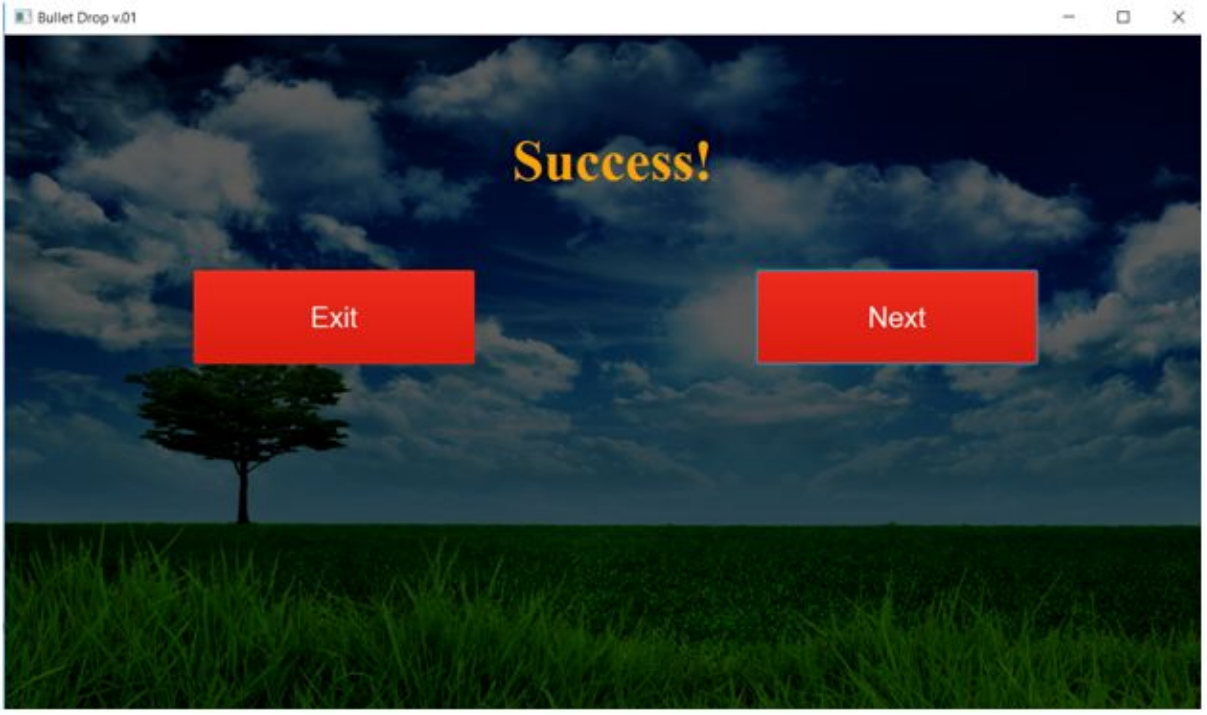

Figure 13.

The main difficulty of the game comes from map differences. We have four different map and they have different count or frequency of external forces at these levels.

Figure 14.

Figure 15.

Figure 16.

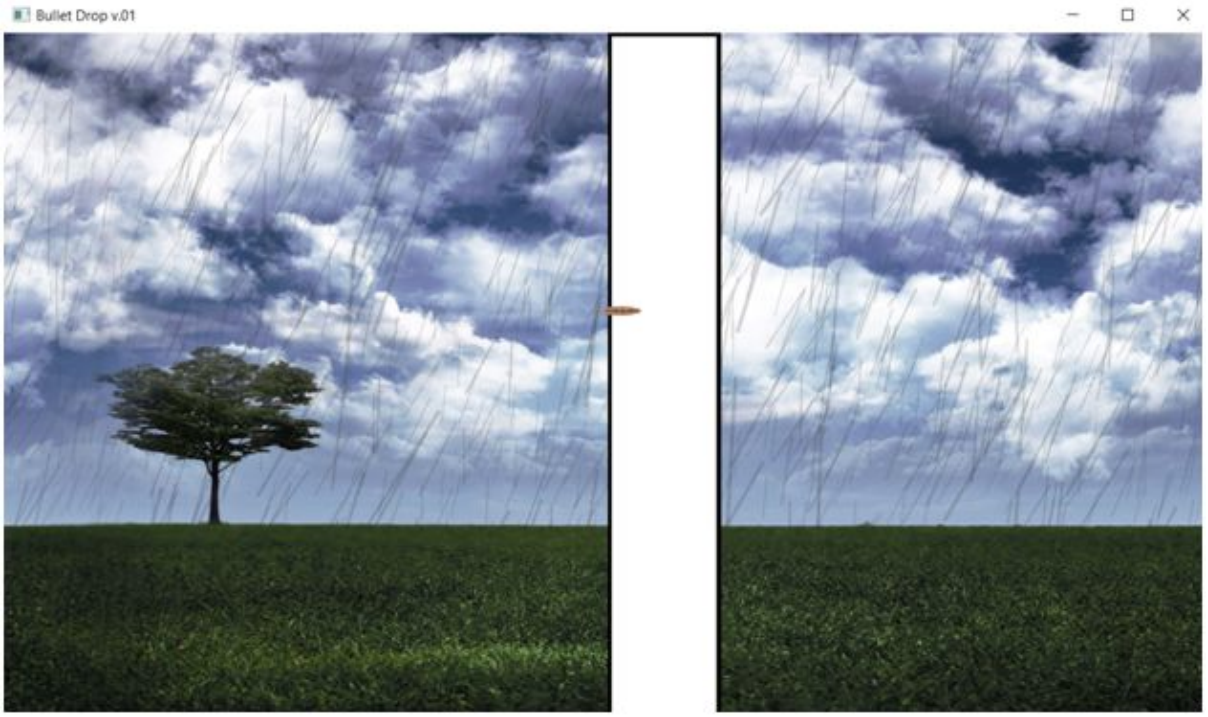

Figure 17.

When the bullet enter the rectangular area in the map (of course we don't have rectangle in release version, this is in order to show it and debug purposes), external force affects the bullet and it goes up or down. For each four different maps we have different

[Figure 14, 15, 16 and 17] count of rectangular area in other words the force that changes location of the bullet and gets him away as much as possible from the target.

Finally, after the user completes the four different maps with three different levels, he/she finishes the game and program returns him/her to main menu back.

4. Changes Done During the Implementation Process

Biggest difference was, instead of using Swing, we decided to use JavaFx. This was due to creating a screen manager and changing screens without an effort was tricky to create in Swing. If also think adding a tick/update method, this was unnecessary effort Fortunately, we came across with JavaFx. Because of that, every screen we'll have will extend Group and and our main class that is "BulletDrop" class has the all the necessary variables to create the stage. Our screen manager will manage the change of screens and their interactions with user. In other words, it will be responsible for current screen. One final information here, we managed to create our "tick-update" system by using JavaFx's AnimationTimer. It's being created in our main class, "Bullet Drop" and connected to our screens and screen manager.

Other than changing our graphic component, there are also some additional classes to helps us create animated image and their controller. These are new classes that helps us to create animated images without an effort.

Additionally, we have added new "Sprite" class to control interaction between our game objects. This class is basically how we define game objects. For example, bullet, target is a game object. Our sprite method is also responsible for these object's speed and movement, in other words position of these objects in every update. There's also collision methods that we use for whether bullet is in a force or bullet hit the target or not.

Finally, during our second iteration, we realised that it's much more flexible for us to do the logic part in screen classes. Normally, in a large scale production, we would not do that. However, when we wanted to that- that is making screen classes give information to one main logic class that controls components, we saw that we came across with some thread problems. For example, in some of situations our bullet were passing by some of the forces.

Additionally, this type of implementation forced us to create lots of unnecessary classes, for example classes that has purpose of passing objects to main controller. This was not a problem of design but JavaFx's lack of flexibility. Considering we only had logic part in our "fireScreen", we thought it's more preferable doing this way in terms of trade-offs between type of designs.

5. Incompleted Parts

There can be a little more optimisation between weapons and the map forces. It can be configured so that in every different map, there will be need for different type of weapon. Of course this was not our goal at the beginning. However, it will be more enjoyable and absorbing experience for the player.

Finally, we did not implement Input manager because user can handle all of the necessary inputs through game's GUI.