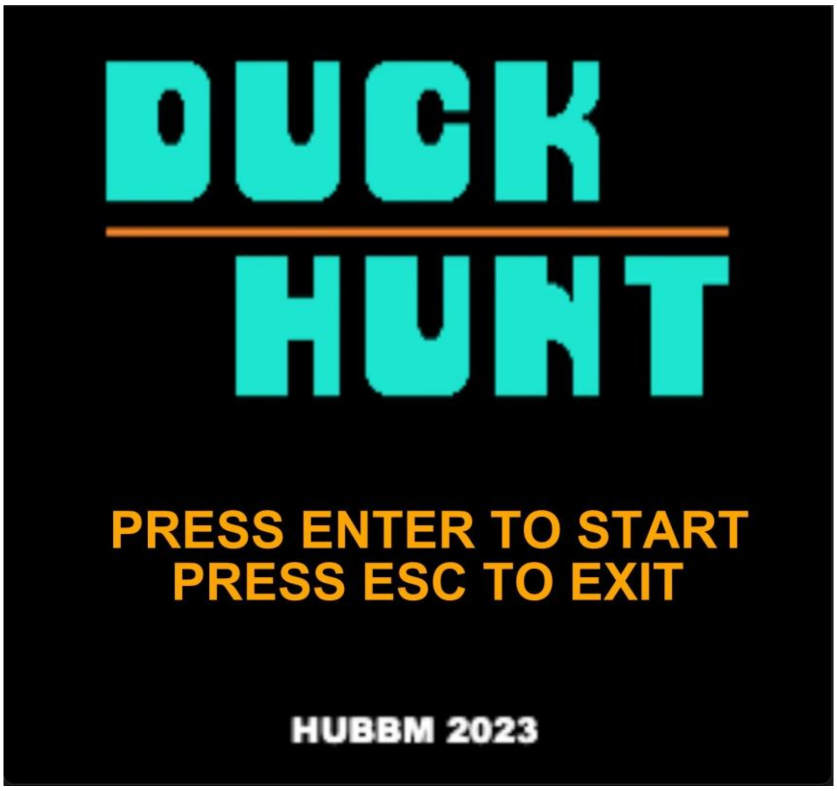
PROJECT

Programming Language : Java 8 (Oracle)
Software Platform : JavaFX



1 Introduction

In this assignment, you are expected to gain practice on developing a Graphical User Interface (GUI) application using Java programming language. As opposed to the command-line programs where the interaction between the user and the computer often relies on a string of text, a GUI program offers a much richer type of interface where the user uses a mouse and keyboard to interact with GUI components such as windows, menus, buttons, checkboxes, text input boxes, scroll bars, and so on. Because most people today interact with their computers exclusively through GUI, developing a GUI based application has become a must for the new developers.
There are many frameworks to develop a GUI application (such as Swing, SWT, AWT, and the like). In this assignment, you are to employ JavaFX framework to complete this assignment. JavaFX is a software platform for creating and delivering desktop applications and Rich Internet Applications (RIAs) that can run across a wide variety of devices. JavaFX supports desktop computers and web browsers on Microsoft Windows, Linux, and macOS.

In this project, a simplified version of game namely Duck Hunt is expected from you to develop through JavaFX framework. All details you need while designing the system are explained in the following section; you will also be given recordings of first two recitations.
Duck Hunt originally is 1984 light gun shooter video game developed and published by Nintendo for the Nintendo Entertainment System (NES) video game console and the Nintendo VS. System arcade hardware. In Duck Hunt, normally player is given some amount of ammo to hit ducks that appears at CRT television with NES Zapper. The ducks can appear in some amount and if the player shoots all of ducks with given amount of ammo, the player will advance to the next round; otherwise, the player will receive a game over. In this project, you are expected to design a version of Duck Hunt game with Java programming language by using JavaFX (Just pure Java code, any other concepts (CSS, FXML, Scene Builder, extra libraries etc.) are forbidden). Moreover, any other assets are forbidden, your program must be successfully compiled and ran with your pure code (*.java files) and given asset files. Note that, contents of asset files can be changed (such as change at sound effects, backgrounds etc.), so please do not code them in static manner, and also please do not submit any asset files as they are going to be given to your code when it is tested.

2 Title Screen

This is the first screen of the game, "assets/effects/Title.mp3" will be played in loop during that screen. User can either go to the background selection screen by pressing "ENTER" key or exit by pressing the "ESC" key. There is a centered flasing text in two lines as follows "PRESS ENTER TO PLAY" and "PRESS ESC TO EXIT" to inform user about what can he/she do to play the game. Title of the game must be "HUBBM Duck Hunt", and also favicon of your game must be "assets/favicon/1.png". Title and favicon must be preserved at each screen and during the game play.

3 Background Selection Screen

This is the options screen of the game, user can either change the background by selecting with left and right arrow keys, and crosshair by selecting with up and down arrow keys. There is a centered text in three lines as follows "USE ARROW KEYS TO NAVIGATE", "PRESS ENTER TO START" and "PRESS ESC TO EXIT" to inform user about what can he/she do to play the game. Moreover, user also can go back to the title screen by pressing the "ESC" key and start the game with selected configurations by pressing the "ENTER" key. Sound effect of the title screen must be still playing without any interrupt during the background selection screen and also it must be still playing if user goes back to title screen. If user presses to "ENTER" and starts to game, sound effect of the title screen gets stopped and "assets/effects/Intro.mp3" gets played for once, after that sound effect ends, game must start from first level. Note that, game must not start until that sound effect finishes. Also note that, options at background selection screen must be reset if some goes to background selection screen again from title screen.

4 Game

Mouse cursor must be changed with selected crosshair icon during the game, note that mouse must be return to system's default cursor icon when user moves his/her mouse to outside of the game window, and also it must return the system's default cursor icon if user returns to the title screen. Game must consist of at least six levels and six directions (going left, right, across the screen -from each corner to its opposite corner-) for the three ducks must be covered. Levels must be sorted according to their difficulty. Ducks must be seen not just as moving but also as flying. Ammo amount for each level must be three times of the duck amount at that level, ammo cannot be transferred to next levels which means it does not matter how many ammo does the user have from previous level, he/she will have just ammo of current level. Ducks must be in between background and foreground, which means if a duck goes to the a foreground object, it will be behind of that instead of being in front of it. Ducks must be reflected in the opposite of the direction that they are coming from whenever they hit to the edges (including corners) of the window (Note that, they are not supposed to be reflected from foreground objects, they just must be reflected whenever they reach the edges of the game window). There must be a centered text at the top of the window that states which level it is and how many total levels there are, sample format for first level of the game which contains six levels as follows, "Level 1/6". Moreover, there must be a text that is at the right corner of the page which states how much ammo is left, sample format for two ammo left is as follows, "Ammo Left: 2". While user shoots (independent from hitting to duck or not) "assets/effects/Gunshot.mp3" must be played once, moreover, if ammo hits more than one duck at same time, it is considered as all the ducks that got hit have been killed, and also ammo are assumed to hit to the ducks that are behind the foreground objects (including floor). Ducks have to fall with animation which starts from "assets/duck_{COLOR_OF_DUCK}/7.png" and followed with "assets/duck_{COLOR_OF_DUCK}/8.png" (images must be flipped if duck is going from right to left when it got hit. Then it has to fall with last mentioned asset. Note that, also "assets/effects/DuckFalls.mp3" must be played once while duck starts **falling**. If user runs out of ammo and at least one of the ducks is still alive, it means that it is game over as loss, so "assets/effects/GameOver.mp3" gets played for once and user gets informed about game is over and he/she can either start again from the first level by pressing the "ENTER" key or go to the title screen by pressing "ESC" key. This information will be done by centered text in three lines (second and third are as flashing) as follows "GAME OVER!", "Press ENTER to play again" and "Press ESC to exit". If user finishes the level successfully (finishing successfully means that killing all the ducks with given amount of ammo), then he/she completes the current level and there are two different situations for completing the level, finishing the last level or finishing any level (which is not last). Finishing the last level will be mentioned later. If user just finishes a level (but not last), as game is not completed yet, just "assets/effects/LevelCompleted.mp3" gets played for once and user gets informed about he/she successfully completed the game. This information will be done by centered text in two lines (second one is as flashing) as follows "YOU WIN!" and "Press ENTER to play next level". So, as informative text says that, user can play the next level by pressing the "ENTER" key, when user presses to "ENTER" key, next level gets loaded. If game is completed (which means completing the last level) "assets/effects/GameCompleted.mp3" gets played for once and user gets informed about game is completed and he/she can either start again from the first level by pressing the "ENTER" key or go to the title screen by pressing "ESC" key. This information will be done by centered text in three lines (second and third are as flashing) as follows "You have completed the game!", "Press ENTER to play again" and "Press ESC to exit". Sound effects of that level must be immediately stopped if user skips to next level, returns the title screen (title screen's sound effect must be played for sure) etc. Moreover, user must not be able to shoot when game or level is completed or game is over even if he/she has ammo left. Note that you must not play intro sound effect if user just starts the game from end of the game or completing the game just via pressing the "ENTER" key (It means that intro sound effect will not be played if user does not start the game from background selection screen.).

5 Scaling

There must be a scale parameter namely SCALE (as in double format) at main class (which is namely DuckHunt) and it must scale the game. It works as follows, say that the object is x by y in manner of pixel, and scale factor is s, then that object must be s times x by s times y in manner of pixel. Say that object's width is 30, height is 40 and scale factor is 3, then object's width must be 90 and height must be 120 as result. Note that default value for scale factor must be 3 when you submit your work as it is suitable for my working environment, but you can use any other scale factor when you are recording your demo video, it is up to you and your screen resolution. Note that, you can still get partial point if you cannot implement scale as changeable with any global variable but code scale as three by static.

6 Adjusting Volume

There must be a volume parameter namely VOLUME (as in double format) at main class (which is namely DuckHunt) and it must adjust volume of sound effects. It must be in range of [0, 1]. One corresponds to maximum volume where zero corresponds to no sound. For example, if volume value is 0.5, it means that volume is at 50%. Note that default value for volume value must be 0.025 when you submit your work as it is suitable for my working environment, but you can use any other volume value when you are recording your demo video, it is up to you and your sound system. Note that, you can still get partial point if you cannot implement volume value as changeable with any global variable but code volume value as 0.025 by static.

7 Bonus: Horizontal Scrolling

For getting a 15 points of bonus, you must implement a horizontal scrolling (only horizontal, not vertical) mechanism for your game (just for the game, not for the background selection and title screens), the game must start from the center of the whole scene, and user can navigate to left and right by putting mouse icon near to the left and right edges of the screen. If this functionality implemented, the vertical edges must be changed with edge of the map of the game instead of edges of the window while handling ducks reflection at the edges. Note that, please do not break the original concept while trying to add this functionality, the game must work with given rules and this rule is just an extra over them.

8 Restrictions

- Your code must be able to compiled and executed on Java 8 (Oracle).
- Your code must be clean, do not forget that main method is just driver method that means it is just for making your code fragments to run, not for using them as main container, create classes and methods in necessary situations but use them as required. Moreover, use the four pillars of Object-Oriented Programming (Abstraction, Encapsulation, Inheritance, Polymorphism) if there is such a need, remember that your code must satisfy Object-Oriented Programming Principles, also you can benefit from exceptions and even if create your own exception class if you need any.
- You are encouraged to use lambda expressions which are introduced with **Java 8**, list structure of Java (and JavaFX) such as LinkedList, ArrayList etc., and your own exception classes.