EECE 435L - Software Tools Lab
Fall 2021

Project Description

1 Objective

A project that involves most important software concepts to enable the student to build and apply software skills while working in a team. The students will be guided by a set of lab experiments, where they will be introduced to helpful methodologies and tools.

2 Project Description

A gaming framework with educational purposes. Every game serves an objective. The first game helps learning the best practices and approaches in software and the second game helps enhancing memory and tactical skills. It tracks performance via saved scores and levels. Thereby, every user can have an account through which she/he can access the different games at the last reached level, and can check the performance and the history scores in every game.
Below is a description of two games that can be expanded. Other games serving the main purpose of the framework are also welcomed and given extra credits.

3 Accounts

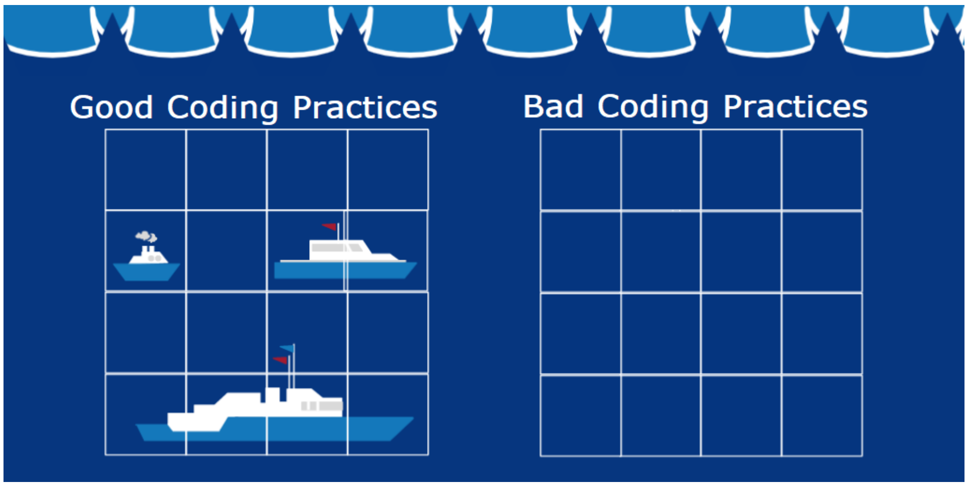
- A user should be able to sign up, sign in or play as guest.
- In the *Sign up* form, the user is asked to enter: his/her username, password, first name, last name, date of birth, gender, phone, and a profile picture.
- The username is a unique identifier.
- When the user logs in, his/her name, profile picture and current date should be displayed.
- When the user logs in, his country name and flag should be displayed according to the country code in the phone number (Up to 5 countries in the data file is sufficient).
- The user should be able to view the history of the scores of the games he/she has previously played. His/Her score should be compared to the global best score.
- The password should consist of at least 8 characters and contain at least one number, upper and lower case letters.
- Phone number should have the following format: countryCode - phoneNumber (*eg.+961-76616248*)
- You can use *txt* or *json* files to store your data.

Further details related to the accounts are up to you.

3.1 Game One: Battleship (Do's v.s Don'ts)

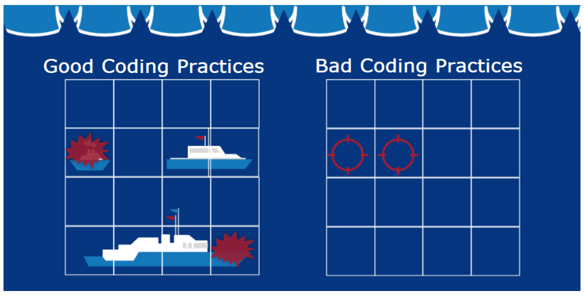
Objective: Apply some strategy thinking and learn the dos and donts of software development. The goal is to get the majority of correct answers at the end of the game.

Description: The goal is to find and strike the battleships presenting bad coding practices by answering questions about software development. The artwork teaches best practices in software development such as documenting code, using version control, testing, debugging, etc.. On the other hand, a student skipping the unit testing part with a consequence of suffering with a buggy very long code is considered a bad practice in programming.

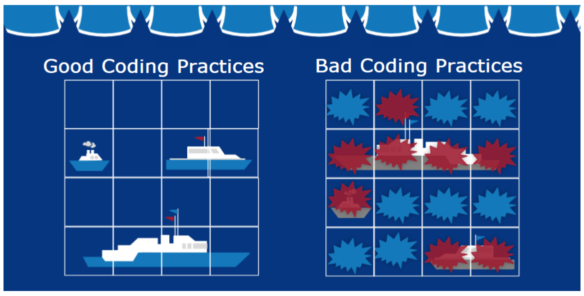


Detailed Description and Rules:

- **Game overview:**
  - The game is played on 2 4x4 gridded board.
  - The player starts after clicking the play button or pressing F1.
  - The user clicks on coding bad practices cells to find the ships. If a ship is under the chosen box, the player answers the question correctly to hit the ship.
  - If the question is answered correctly, the enemy ship (bad coding practices) is hit.
  - If the answer is incorrect, your shot is missed and your enemy fires back, hitting one of your ships. The enemy's hit can be randomly set - at any location as per the below figure.



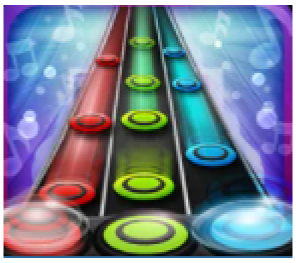
  - If no question is found in the box, the trial is missed and the total number of attempts is reduced by 1. The cell is hence marked by any color or shape as per the below figure.
  - The player can have up to 16 trials to uncover all the cells - all trials must be consumed.
  - The player wins if 7 out of 10 questions are answered correctly.
  - The player fails if 4 questions are answered incorrectly.



  - At least one ship must span up to 3 cells. Others can fit in 1 cell.
  - Questions and answers should be saved and read from a txt/json file.
  - Game can be customized to reflect other areas. That is the user can at any time change all the questions and the corresponding answers according to the new topic.

- **Valid Hits:**
  - Inside the boxes of the right grid.
- **Game Ends:**
  - When the total number of trials is 0.
- **Time Limit (Bonus):**
  - To add more pressure to the game, a timer can be added.
  - Time limit ranges from 5 minutes up to 10 minutes.
  - When the time expires, the scores will be calculated to identify whether the player wins or looses the game.
- Check the following link for the game basic concepts and strategies: `https://www.asu.edu/lib/tutorials/storyline/academic-integrity/story_html5.html`

3.2 Game Two: Shooting discs

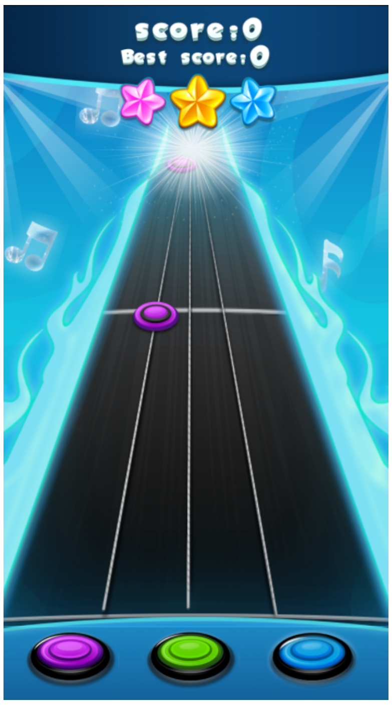


Objective: Apply some strategy thinking of software development.

Description: ShootingH is a classic shooting game. Shoot as much as you can and hit the objects. The original game involves arbitrary shooting with a counter. The winner is the first player to reach the target. We will introduce some changes to tweak the objective of the game and add an algorithmic purpose.

Detailed Description and Rules:

- The game is played on a rectangular grid board.
- A number of discs is placed randomly on a scrolling board. When the disc is hit, it turns into a grey disc and the counter is increased. Otherwise, it disappears.



- The objective of the game is to hit high number of discs.
- There are 3 types of discs: red, green and blue. Each disc type is represented by a distinctly colored game piece token.
- There are 3 counters for the different types of viruses and a total score. Red discs are worth 3 points; Green discs are worth 5 points; Blue discs are worth 7 points.
- The player starts after clicking the play button or pressing F1. The player collects points when he successfully hits the target. The score is incremented according to the type of the disc being hit.

- To win the game, the player must collect 150 points regardless of the counter. If the total score is more than a 150, the player wins and the additional points are considered as a bonus and saved separately.
- The rolling speed starts slowly. The speed is doubled after collecting 30 points. For instance, if the starting speed is 1x, the speed becomes 2x after the first 30 points, then 4x after 60 points and so on. The speed should not exceed 16x (which occurs at 120 points).
- The player looses if he misses hitting 3 discs.
- You are free to design the game front end. You can check the online game to get ideas: `https://lagged.com/play/1151/`. You can also check other online games for more ideas, for example: `https://www.crazygames.com/game/snakes-and-ladders`.
- Generate the grid with discs dynamically. The position of the discs can be generated inside the algorithm directly or read from the data file. The data file should also contain the winning score (150 in this example) and the number of missed discs to loose the game (3 in this example).

Bonus points:

- Extra Features (ex: Sound effects when shooting, when hitting the disc, ..).
- Option to change the game level using the GUI interface (Easy, Medium, Hard).
- Picking a background image/color.
- Additional creative ideas are also considered as a bonus.

4 General Requirements

Note that the different games have similar features. You are encouraged to re-use logic you already implemented in another game whenever possible. In this case, the logic should be designed as generalized methods, or objects, to be called, or instantiated, properly from different code locations. Below are some general requirements for the overall framework.

I Player Accounts:
Enable the user to create an account
Save history scores and allow the player to access his account for playing, viewing scores, or continuing a previous game.
Generate a performance graph for the player according to the history scores in each game and as overall.

II Necessary code for score computation, level tracking, and such other details.

III A GUI for the overall framework with:
1. Menu for setting/accessing account, instructions and options
2. Game interface: Where different games can be accessed for more information and instructions, and then to start the game.
The design is free as long as it satisfies all required interactions for using the framework and playing the games.

IV Quality code with learned software concepts and tools. (Object Oriented Programming, source control, testing, ...)

V You should use version control extensively in your project.

VI The code should be fully documented (classes and methods mainly).

VII Automatic Build of the project.

5 Deliverables and Deadlines

Phase 1	Implementation of Accounts and framework menu	Due October 05
Phase 2	Implementation of Game 1	Due October 28
Phase 3	Implementation Game 2	Due November 25
Phase 4	Final Submission Merge of all work in one working framework Addition of other requirements as user accounts and history. Presentations will be within a week from the final submission.	Due November 28

Note that for every phase submission, you are expected to use the tools that we are taking in the lab - whenever submission occurs after the corresponding labs.

6 Grading Criteria

- 20%: Phase 1 (Accounts) - Main menu navigation and framework navigation
- 40%: Phase 2 (Game 1)
- 20%: Phase 3 (Game 2)

  (Evaluation for the games include use of version control and documentation).
- 10%: Use of tools: profiling, automatic build and packaging
- 10%: Overall completion of the framework+ final report+ presentation