COMP 3710 - Wireless Software Engineering
Spring 2018
Term Project

Assignment Build an Android app of your choosing.

Objective The term project is hands-on application of the skills you learned this semester. My expectation is that you will approach building an Android app in a disciplined fashion, following sound engineering principles.

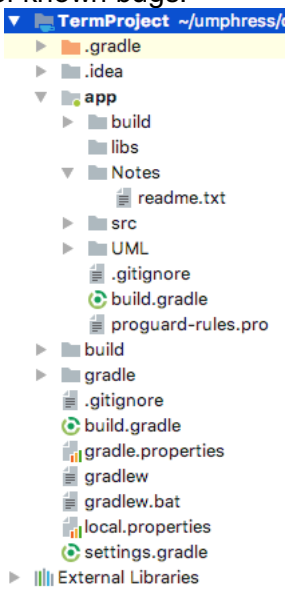
Requirements

- Development Logistics:
  - You may work in a team of 1, 2, 3, or 4 people. The team roster must be listed in Canvas.
  - Your app must have features commensurate with the size of the team.
  - Your app must have demonstrable usefulness.
  - Your app must be built with targetSDKVersion=25 (Android 7.1.1 Nougat). It must be capable of running on a Nexus 6 (resolution:1440x2560).
- Deliverable material:
  - Engineering documentation:
    - Use case diagram
    - Domain model
    - System sequence diagram
    - Design class diagram(s)
    - Optional:
      - Individual use case descriptions
      - Design sequence diagram(s)
      - Activity diagram(s)
      - State diagrams(s)
  - Project source and accompanying files
  - Release notes.
- Baseline software specifications*:
  - Your project should be structured as follows:
    - Project name: a meaningful name of your choosing.
    - Application name: a meaningful name of your choosing
    - Package name: edu.auburn.eng.csse.comp3710.spring2018.teamxx where *teamxx* is the name of your team as shown on Canvas.
    - Test device: Nexus 6 (resolution:1440x2560)
    - Android 7.1.1; API Level 25
    - Include a subdirectory titled “UML” directly under the “app” subdirectory (see below for a sample screen shot) for your UML diagrams. The diagrams must be in PlantUML format.
    - Include a subdirectory titled "Notes" directly under the "app" subdirectory (see below). Place there a file named "readme.txt". The file should contain the following information:
      - The name of the app
      - A 1-to-5 line description of what the app does
      - Special instructions needed to build the app (if none,

* Please make explicit arrangements with me if you need to deviate from these specifications.

please indicate "none").

- A list of known bugs.



- The app should be capable of being paused without loss of state. In other words, the application must handle the receipt of a phone call; another application being started while your application is running; your application being paused and subsequently killed to conserve resources; etc.
- The app should handle changes in screen orientation, unless there is a justifiable reason to lock the app in a specific orientation and prevent screen rotation. Justification should be provided in the release notes.
- The app must have a prolog screen (also known as a splash screen) that shows
  - a logo for the application
  - the name of the application
  - the name of the members of the team
  - "Auburn University COMP3710 Spring 2018"
- The app must have a unique launch page icon.

Deliverables

- Milestone 1:
  - Form a team of 1, 2, 3, or 4
- Milestone 2:
  - Submit the name of your project, a brief description of your project, and a list of major features you plan to implement.
- Milestone 3:
  - Upload to Canvas a single zip file exported from Android Studio. The steps are:
    - From Android Studio's main menu, select File --> Export to Zip File... . A dialog box will appear.
    - Name the zip file “teamxx.zip”, where *teamxx* is the name of your team as shown on Canvas.
    - Upload the file to Canvas.

Evaluation

Your project deliverables will be evaluated along the following criteria:

Delivery

- Are release notes complete
- Can be rebuilt without assistance

Use Case Diagram

- Is free of design and implementation issues
- Expresses user desires
- Communicates clearly

System Sequence Diagram

- Shows black-box level of detail
- Appropriately identifies actors
- Is syntactically correct
- Corroborates other diagrams

Domain Diagram

- Depicts objects within the domain
- Has appropriate relationships, relationship names, multiplicity
- Is at appropriate level of detail
- Is free of design and implementation issues
- Corroborates other diagrams

Design Class Diagram

- Reflects delivered code
- Reflects objects of discernment and representation
- Demonstrates grasp of cohesion and coupling
- Corroborates other diagrams

Other Diagrams (optional)

- Reflects delivered code
- Illustrates behavior at the appropriate level of abstraction
- Is syntactically correct
- Corroborates other diagrams

Application Functionality

- Supports requirements
- Has usable interface

Application Stability

- Is robust
- Has no obvious problems on the emulator[†]

Application Architecture

- Demonstrates engineering forethought
- Employs MVC
- Employs Fragments

Overall

- Suitability as a sample to distribute to prospective students

Projects will be scored along the rubric below.

1. Outstanding (90-100)
   The app is a superior sample of work; suitable for distribution; completely implements requirements; provides accurate information and/or services; expresses functionality and terms in the vernacular of the user; appropriately protects privacy (as appropriate); is well-engineered; has no obvious defects.

2. Good (80-89)
   The app is an adequate sample of work; close to being suitable for distribution, given some polishing; implements the requirements deemed the most important; provides accurate information and/or services; addresses privacy (as appropriate), although may not fully protect it; expresses functionality and terms in the vernacular of the user; is suitably engineered; has no defects that detract from the overall functionality.

3. Adequate (70-79)
   The app represents a prototype of the intended concept; implements some requirements considered useful; provides accurate, although possibly incomplete, information and/or services; makes minor developer assumptions; uses software developer jargon; may have minor defects, but no “not responding” errors or unexpected exceptions; demonstrates some understanding of software engineering concepts.

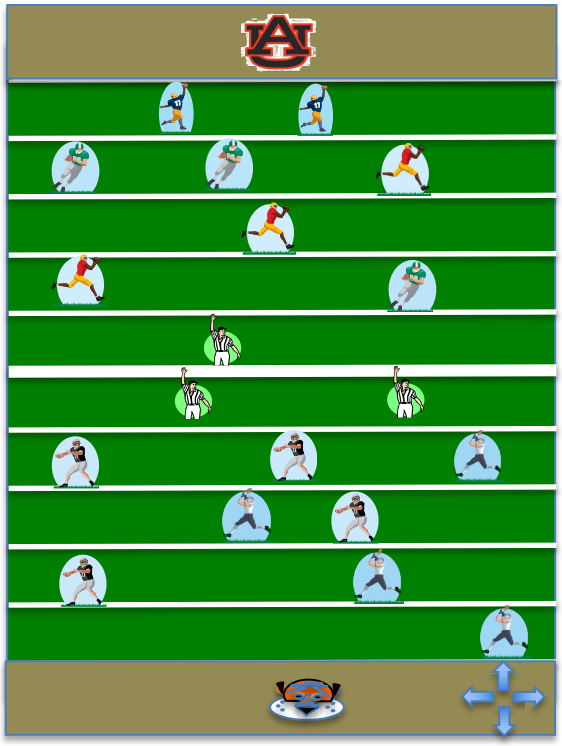
4. Marginal (60-69)
   The app represents an initial sketch of the intended concept; implements some requirements but would not be considered especially useful after an initial installation; provides information that may be incomplete; makes developer assumptions; has obvious defects; demonstrates only a weak understanding of software engineering concepts.

5. Unsatisfactory (<59)
   Deliverables don’t represent a sincere effort.

Here are some project ideas, if you are at a loss for what to do:

Project

Watch Out Aubie!. This is an adaptation of the classic "Frogger" arcade game. The object of the game is to direct Aubie from one end zone of the football field to the other, avoiding football players along the way. The player uses on-screen buttons to move Aubie up, down, right, or left.



Ideas

Basic

- The game has two selectable levels of play:
  - Easy - there are four lines of players: two running along each side of the 50 yard line. Players move at a relatively slow speed and are spaced at wide intervals.
  - Hard - six lines of players run across the field: three on each side of the 50 yard line. Players move at a challenging speed and are spaced at challenging intervals. Referees join football players in running across the field. The user gets an extra "life" if Aubie is hit by a referee. No line can have more than one referee; at most three referees can be on the screen at any point in time.
- The time to get Aubie from end-zone to end-zone is recorded. Statistics are kept on the minimum, average, and longest time. Statistics can be reset.
- Aubie is animated when he is in motion.
- Football players run in lines across the field. Lines move at different speeds; players in the same line move at the same speed.
- Football players in the lower half of the screen move from left to right; players in the upper half of the screen move right to left.

Challenging

- Each line consist of a random number of players.
- The player has four attempts to get Aubie from one end-zone to the other without being hit by a football player. A sickening crunching sound results when a football player hits Aubie.
- The game ends with a thumbs-down if Aubie does not make it to the end zone after four attempts, and with a thumbs-up and the AU fight song if Aubie gets to the end zone without being hit.

Coolness

- Instead of players moving linearly across the field, five players and one referee are placed in random locations in the half of the field opposite Aubie. When Aubie starts to move, the players move toward Aubie as a swarm (see Wikipedia's "swarm behavior").

Aubie's Memory. This application is patterned after the Milton Bradley game, *Simon*. "The game unit has four large buttons, one each of the colors red, blue, green, and yellow. The unit lights these buttons in a sequence, playing a tone for each button; the player must press the buttons in the same sequence. The sequence begins with a single button chosen randomly and adds another randomly-chosen button to the end of the sequence each time the player follows it successfully. Gameplay ends when the player makes a mistake or when the player wins (by matching the pattern for a predetermined number of tones)" [Wikipedia].

Basic

- The application has four buttons: red, blue, green, and yellow. Each button has an image of Aubie.
- The longest correctly-repeated sequence of button presses is displayed as the score.
- The player can configure the game to enable/disable sound.
- The player can configure the difficulty level of the game by specifying how quickly the sequence of tones/lights is displayed. The "beginner level" is slower than the "intermediate level", which is slower than the "expert level".

Challenging

- The application has four buttons: red, blue, green, and yellow. Each button has an image of Aubie.
- The longest correctly-repeated sequence of button presses is displayed as the score.
- The player can configure the game to enable/disable sound.
- The player can configure the difficulty level of the game by specifying how quickly the sequence of tones/lights is displayed. The "beginner level" is slower than the "intermediate level", which is slower than the "expert level".

Coolness

- Instead of pressing buttons in a particular sequence, the user draws patterns. The application displays a 3 x 3 matrix of buttons. The application draws a line with two buttons as end points, playing a tune for each line; the player must draw a line with the same end points. The sequence begins with a single line and adds another line to the end of the sequence each time the player matches the application's sequence. Gameplay ends when the player makes a mistake or when the player wins by matching a predetermined number of lines.

AU Puzzle. This is the classic plastic puzzle consisting of a frame encasing an (n x m)-1 set of tiles. A tile adjacent to the empty space can be shifted. Each tile has a portion of an image. The tiles, when placed in correct order, form a complete image. See below (but imagine a picture, not digits):

Basic

- The puzzle consists of a minimum of 4 x 4 squares (15 tiles and one empty space).
- Tile movement is animated, meaning, tiles follow the movement of a fingertip.
- The player selects from three AU pictures that are stored in the application's *res* subdirectory.
- The number of times moved is displayed along with the puzzle itself.

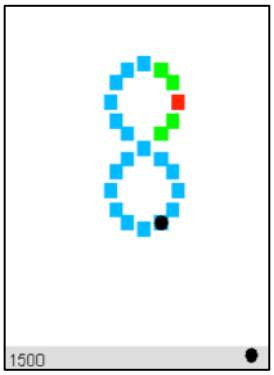
Challenging

- The player can select the the size of the puzzle (i.e., the number of tiles).
- The player can select the picture from a provided set of AU pictures or from the user's gallery of pictures.
- The time to complete the puzzle is recorded. Statistics are kept on the average, minimum, highest number of tiles moved; and the minimum, average, and longest time. Statistics can be reset.

Coolness

- Includes an animated "solve it for me" mode that moves tiles without user intervention.
- Makes sounds when a tile is moved. The user has the option of enabling/disabling sounds anytime while working the puzzle

AU Light Chase. This arcade game is a staple at places such as Chuck E Cheese. It consists of light bulbs arranged in a pattern. The bulbs light up in sequence, given the player the impression a light is running around a track. The player hits a button when a light at a designated part of the track illuminates. Hitting the button at the correct point results in a 5-point reward. 4 points are awarded if the button is hit one light either side of the target, 3 points if two lights away, and so forth. In the illustration below, the black dot represents the illuminated light, the red dot represents the five-point light, and the green dots represent lower-value lights. The black dot in the lower right shows the number of plays left.; the number shows the score.



Basic

- A sound is made each time a light is illuminated. The sound corresponds to the number of points associated with the light. Zero-point lights have a low-frequency tone, one-point lights emit a tone that has a slightly higher frequency, up to the five-point light which has the highest frequency tone.
- The player gets five attempts to score. Score is calculated by the number of points that correspond to the light that is illuminated at the time the select button is pressed.
- If the player's score is greater than a threshold set in the game's configuration settings, the player is presented with a more challenging track.

Challenging

- Statistics are kept on the average, minimum, and maximum duration of each play. Statistics can be reset.
- The application randomly reverses the sequence in which the lights are lit. It also randomly varies the speed at which the lights are lit.

Coolness

- The player can construct a track of lights by outlining it with his/her finger

AU Silver Eagle Cell Phone. Most so-called "senior citizen" cell phones are full-featured phones, but with oversized number keys. While this may be helpful to older people who see well and are cell-phone literate, these cell phones are not particularly usable for people who have trouble seeing and only want to make/receive phone calls. The purpose of this project is to develop a replacement home screen from which a blind senior citizen can make and receive phone calls.

Basic

- The home screen allows three basic functions:
  - Regular dialing mode: The buttons traditionally associated with a cell phone (including backspace and call) are displayed in large format. The phone voices the name of each key when pressed.
  - Speed dialing mode: Large buttons display the names of people who will be dialed when the button is pressed. When pressed, the button voices the name of the person.
  - Emergency mode: Sends a text with a customized message to a specific number, then dials that number.
- Upon receipt of a phone call, the phone voices the name and number of the caller.
- Dead-man switch: A large red button appears and a siren sounds upon receipt of a text containing “Are you OK?” received from a prescribed number. The user has 3 minutes to press the button to silence the alarm. Pressing the button responds with a text saying “Yes, I am OK”.

Challenging

- The user can dial the phone by drawing the numbers on the screen.

Coolness

- Recognizes voice commands.
- Calls the emergency number above based on a specific motion of the phone, such as waving the phone around or scooting it across the floor.