CS478: Software Development for Mobile Platforms

Project #4

Due time: 11:59 pm on 4/10/2022

Total points: 100

Instructor: Ugo Buy

TAs: Sampath Gottikere Kumaraiah, Shahmeer Ahmed, Garima Chaudhary and Yash Kurkure

Copyright © Ugo Buy, 2022. All rights reserved.
The text below cannot be copied, distributed or reposted without the copyright owner's written consent.

You are to design and code a game of *gopher hunting* as an Android app. Imagine you are in a field occupied by a gopher. The gopher could be hiding in one of any number of holes in the field's ground. Two algorithms that you will design play against each other in an effort to find the hole that contains the gopher. (The game does not say what to do with the gopher, once it is found.)

The game is played by two worker threads contained in your app, running different algorithms and playing against each other. (There is no human input once the game is started.) There are exactly 100 holes in the field; the holes are arranged as a 10×10 matrix and equally spaced with respect to each other. The first thread to find gopher wins the game.

The app supports a continuous-play mode whereby the two threads play without interruption until one thread wins the game. Each time a player makes a guess at to the location of the gopher, it receives feedback indicating how close the guess was to the gopher's location, including whether the gopher's location was discovered correctly.

The UI thread is responsible for setting up the game, starting the worker threads playing against each other, showing the guesses of the two players on two separate tables.

Whenever a thread makes a guess, the UI thread provides one of four possible responses:

1. Success—The thread guesses the hole containing the gopher and wins the game.
2. Near miss—The thread guesses one of the 8 holes adjacent to the gopher's hole. The game continues with the other thread making a guess.
3. Close guess—The thread guesses a hole that's 2 holes away from the gopher's hole in any directions. In other words, it is possible to go from the guessed hole to the gopher's hole by making two moves to adjacent holes from the guessed hole. Each move could be in a horizontal, vertical or diagonal direction.
4. Complete miss—The thread gets this response in all other cases.

Here are some additional requirements on the app.

1. Your app should contain three threads, namely the UI thread and two worker threads playing against each other.
2. At the beginning of the game, the UI thread should choose a random location for the gopher in the 10x10 matrix. This location is not shared with the worker threads.
3. You are at liberty to design the app in the way that you find most appropriate, including the number and type of components in the app. (Hint: You should probably have at least two activities, one for starting a game, another to display the progress of the game and whether the game was finished.)
4. The main activity should show a button for starting the game and a button for stopping a game. When the user stops the game, the UI thread should send a message to the worker threads indicating that they should terminate themselves.

5. The activity showing the game's progress should show two 10x10 tables clearly indicating the progress of each player. Each guess should be clearly marked with a number showing the order of the guesses, with the first guess being marked 1, the second guess being marked 2, etc. In addition, the activity should show the (identical) location of the gopher in both tables.

6. Whenever a player thread makes a guess, it pauses (sleeps) for two seconds in order to show its new move on the display. The thead can resume deliberating its next move (unless the game ended in the meantime) after two seconds.

7. Each thread must have a job queue, a looper and a handler. The two worker threads should work in parallel to compute the best strategy for winning the game. The main thread should manage the game, e.g., starting the game, determining when the game is over, notifying the worker threads to stop playing and displaying the winner on the app's display.

8. The threads must communicate with the main thread using *handlers*. You must include both runnables and messages (at least one of each) in the job queue of the worker thread. (When adding jobs to handlers, make sure to post at least a *Runnable* and send at least a *Message* somewhere in your project code.)

9. The sequence of guesses produced by the two players should be different. You can achieve this goal by using the same algorithm, but using different starting locations (e.g., a random guesses and a fixed location) for the first guess of the two players.

10. The app is meant to be run in portrait mode, for instance, showing the two progress tables on top of each other. You need not worry about showing the app in landscape mode.

You must work alone on this project. For this project use a Pixel 3 XL device running the usual Android platform (API 30—Android 11). You are not required to provide backward compatibility with previous Android versions. Submit one Studio project as a zip archive using the submission link in the assignment's page on Blackboard. Code that does not use worker threads for the players will receive no credit. No late submissions will be accepted.