Stanford University, CS 193A
Homework 5: 2D Graphics; Swiping; Maps

The purpose of this assignment is to practice the material from the past few weeks, such as apps that draw 2D graphics and apps that use maps. We will list several suggestions, but you **only need to write one app** to submit. As always, if you have a different idea for a program to make, please feel free to do so.

Please turn in a **.ZIP file** containing the entire contents of your Android Studio project folder, named **hw5-*sunetid-description*.zip**, such as, **hw5-jsmith12-Pong.zip**. Your submission will be graded quickly by simply running it and evaluating its functionality. It does not need to be perfect or bug-free to receive credit. If you want help, please feel free to show your code to others or ask for help in our online **Piazza forum**. If you work on your solution for roughly **2 hours** and are still not done, you can turn it in and we will award you credit.

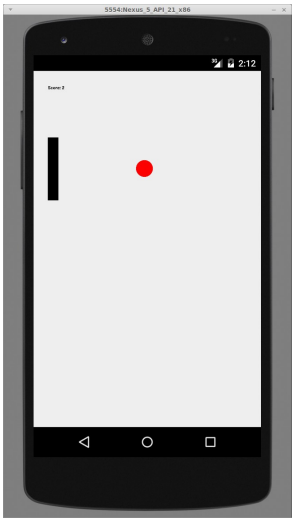
Assignment Ideas and Suggestions:

Suggestion 0: Make Up Your Own

If you prefer something of your own, please feel free to do so. We prefer an app that has the following qualities:

- Your app should be set up as an **Android Studio** project, so it can easily be opened/run/graded by others.
- Your app should use some of the last two weeks' **new material** in some way.
- Along with your app (inside the ZIP), turn in a file named `README.txt` that contains your name and email address along with the name of your app and a very brief description of it, along with instructions to run it.

Suggestion 1 (light)*:* 1-Player Pong



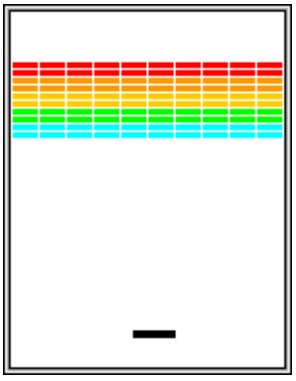
Write an app that draws a custom `View` for a very basic one-player Pong game. This can be heavily based on the "bouncing ball" program that we wrote in lecture.

Paddle: The main change that you'd need to make is to provide a **paddle** that the player can control. The paddle can be at the left edge of the screen, and the user can tap the screen to move it. If the user taps above the paddle, it starts to move up. If the user taps below the paddle, it starts to move down. You can implement whatever control scheme you like, or put the paddle on whatever side of the screen you like.

Scoring: If the ball gets past the paddle and hits the left edge of the screen, the player loses 1 point. If the ball hits the right edge of the screen, the player gains 1 point.

Please feel free to use the library classes shown in class, such as `DrawingThread` and `Sprite`. Also feel free to use any of the `BouncingBall` code from class as a basis for this program.

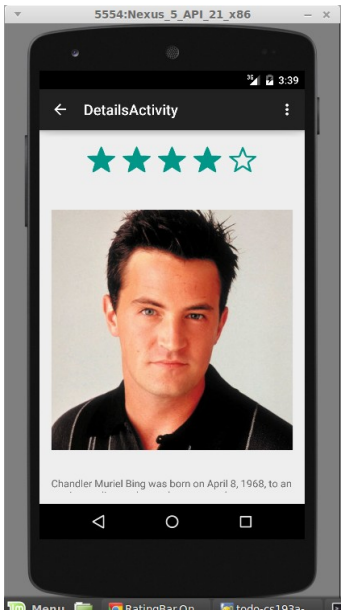
Suggestion 2 (heavy): Breakout



Convert your CS 106A Breakout game into an Android app. This would be a bit of work to complete, but the result is an impressive game. A lot of logic can be copied from your old CS 106A code if you still have it. Here are a few implementation notes:

- Remember that the Stanford Java libraries, such as `GRect`, `GObject`, `RandomGenerator`, and `getElementAt` are not available in Android.
- In Breakout there are a large number of bricks. In this version, you could store them as an `ArrayList` of `Sprite` objects. You'd need to loop over the list to see if the ball collides with any one of them.
- You might want to modify the `Sprite` class to be more similar to the `GRect` class from 106A here.

Suggestion 3 (light): Friendsr with Swiping



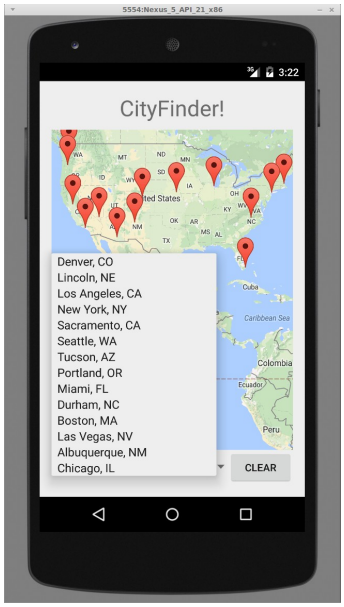
If you did the Friendsr dating app for HW3-4, modify it to use **swiping** so that when the user is viewing the details about a given person, they can swipe left on that person's image to "reject" that person or swipe right to "accept" that person. Consider modifying your activities so that the swipe is remembered and leads to a thumbs-up or thumbs-down icon next to that person in the overall list, or something of that nature to indicate who has been accepted and who has been rejected.

If you already made a previous version of this app using a 5-star rating bar, you could interpret a left-swipe to be a "1-star" rating and a right-swipe to be a "5-star" rating for simplicity if you like.

If you like, you can use the `OnSwipeListener` helper library class shown in lecture to help you implement this feature. The listener has several options you can set up such as whether the image should drag along with the user's finger during swiping, whether it should "snap" back into its original position after the swipe, and so on. See the documentation of that class for more details.

Suggestion 4 (medium): Add Features to CityFinder

For this suggestion, you'll modify the CityFinder app shown in lecture and add some features to it. First you'll need to download the CityFinder app and get it to run using your own Maps API key. This is non-trivial and requires several steps as outlined in the lecture slides.



Change the app to add functionality, such as some combination of the following:

- **More Cities:** Add more cities to the `cities.txt` file; make sure that each one gets a marker on the map.
- **City Picker:** Add a drop-down menu (such as a `Spinner`) where the user can pick any one of the cities by name, and the map will shift its camera to center on that city.
- **Draw/Clear Paths:** Make it so that the user can draw a path of lines between cities by clicking on a series of cities, and click a Clear button to remove the path from the map.
- **Click to make a marker**: Make it so that when the user clicks on the map on an otherwise unoccupied location, a marker is placed there that shows that position's latitude and longitude (as its "title").
- **Other:** Anything you want! Play around with the map and see what neat features you can come up with.