Midterm Exam #1

CMSC 436
Programming Handheld Devices
Fall 2011

November 16, 2011

Guidelines

Put your name on each page before starting the exam. Write your answers directly on the exam sheets, using the back of the page as necessary. If you finish with more than 15 minutes left in the class, then bring your exam to the front when you are finished and leave the class as quietly as possible. Otherwise, please stay in your seat until the end.

If you have a question, raise your hand and I will come to you. Note, that I am unlikely to answer general questions however. If you feel an exam question assumes something that is not written, write it down on your exam sheet. Barring some unforeseen error on the exam, however, you shouldn't need to do this at all, so be careful when making assumptions.

Question	Points	Score
1	40	
2	30	
3	30	
Total	100	

1. Short answers (40 points). Give very short (1 to 2 sentences for each issue) answers to the following questions. **Longer responses to these questions will not be read.**

   (a) Android supports drawing 2D graphics to Views and to Canvases. Under what conditions would you choose to draw to a View? Under what conditions would you choose to draw to a Canvas?

   Answer:

   Draw to Views when the graphics are simple and don't require dynamic updating. Draw to Canvases when the graphics are more complex and require regular updating.

   (b) Android distinguishes Internal Memory Files from External Memory Files. Suppose you are creating an application that writes an External Memory Filed. Name 2 issues that you must address in this situation that you would not have to address if you were writing to an Internal Memory File instead.

   Answer:

   You must check that the External Memory is mounted & writable.
   Your app must request WRITE_EXTERNAL_STORAGE permission.

   (c) The View class has multiple invalidation methods, including one called invalidate(). What is the purpose of an invalidation method?

   Answer:

   It tells the system that some or all of the View needs to be redrawn. Calling an invalidation method will cause the View's onDraw() method to be called.

(d) The View class has a post method with the following signature, “public boolean post (Runnable action)”. What does this method do and why is it important?

Answer:

This method causes the Runnable parameter to be run on the UI/Main thread. It's important because it allows threads other than the UI/Main thread to update the UI on the UI/Main thread.

(e) Suppose that you're writing an application that needs to receive touch input. Within your onTouch() method, how can you tell that the user is touching with 2 fingers rather than 1 finger.

Answer:

You receive a MotionEvent action of type such as ACTION_POINTER_DOWN, rather than ACTION_DOWN.

(f) Android allows you to send Messages or post Runnables to an instance of the Handler class. Give one reason why you might choose to send a Message rather than post a Runnable to a Handler.

Answer:

The action taken in response to a sent Message is implemented by the Handler. The action taked in response to a posted Runnable is implemented by the poster of the Runnable.

(g) In Android, SQL queries are returned as an instance of the Cursor class. What is a cursor?

Answer:

An iterator over the set of result matching the query.

(h) Name 3 properties of a graphic element that can be controlled by a Paint object.

Answer:

Any 3 of Text size, Text color, Text typeface, Text style, border color, border width, border dash, background fill color.

2. Threads & Messaging. (30 points). The code below creates a simple game in which the screen in divided up into 4 quadrants. Your code will create 4 Threads; one for each quadrant. Each Thread will send Messages to a Handler to 1) set the Thread's quadrant's background to green and to 2) toggle the Thread's quadrant's visibility at random intervals. Fill in the code below to complete this application. You may assume that the MyImageView class has two methods: toggleVisibility() that toggles the ImageView's color between Green (visible) and Transparent (invisible), and setColor(int Color) that sets the MyImageView's background color to the specified Color.

```
public class SimpleThreadingExample extends Activity {
  public final static int TOGGLE = 0, SET_VISIBLE = 1;
  private final static MyImageView[] mImageViews = new MyImageView[4];

  static Handler mHandler = new Handler() {
    public void handleMessage(Message msg) {
      switch (msg.what) {


        case TOGGLE: {
          mImageViews[msg.arg1].toggleVisibility();
          break;









        }
        case SET_VISIBLE: {
          mImageViews[msg.arg1].setColor(Color.GREEN);
          break;








        }
      }
    }
  };

  public void onCreate(Bundle savedInstanceState) {
    ...
    mImageViews[0] = (MyImageView) findViewById(R.id.imageView1);
    mImageViews[1] = (MyImageView) findViewById(R.id.imageView2);
    mImageViews[2] = (MyImageView) findViewById(R.id.imageView3);
    mImageViews[3] = (MyImageView) findViewById(R.id.imageView4);
```

cont. on next page ...

```
    final Button button = (Button) findViewById(R.id.startButton);
    button.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
        new Runnable() {
          public void run() {
            for (int i = 0; i < 4; i++) {
              final int index = i;

              // Implement Threads here
              new Thread(new Runnable() {
                public void run() {









                  sleep();
                  Message msg = mHandler.obtainMessage(SET_VISIBLE, index, 0, null);
                  mHandler.sendMessage(msg);
                  sleep();
                  while (true) {
                    sleep();
                    msg = mHandler.obtainMessage(TOGGLE,index, 0, null);
                    mHandler.sendMessage(msg);
                  }







                }
              }).start();
            }
          }

          private void sleep() {
            // asssume this sleeps current Thread
          }
        }.run();
      }
    });
  }
}
```

3. Touch Events. (30 points). Extend the code from the previous question to process touch events on each quadrant and to keep track of the game's score. Score the game as follows: if the user touches a visible (Green) quadrant they get one point; if the user touches an invisible (Transparent) quadrant they get no points.

```
public class SimpleThreadingExample extends Activity {
  public final static int TOGGLE = 0, SET_VISIBLE = 1, UPDATE_SCORE = 2;
  private final static MyImageView[] mImageViews = new MyImageView[4];
  private static TextView scoreView;
  private static int mHits, mTouches;

  static Handler mHandler = new Handler() {
    public void handleMessage(Message msg) {
      switch (msg.what) {
        case TOGGLE: {
          // as before
        }
        case SET_VISIBLE: {
          // as before
        }
        case UPDATE_SCORE: {





          if ((Boolean) msg.obj)  mHits++;
          mTouches++;
          scoreView.setText(mHits + "/" + mTouches);
          break;










        }
      }
    }
  };

  public void onCreate(Bundle savedInstanceState) {
    ...
    scoreView = (TextView) findViewById(R.id.scoreView);
    scoreView.setText("0/0");
    ...

    // remainder as before
```

```
public class MyImageView extends ImageView {
  private int mColor;

  public MyImageView(Context context, AttributeSet attrs) {
    super(context, attrs);
  }

  public boolean onTouchEvent(MotionEvent event) {






    Message msg = null;
    if (MotionEvent.ACTION_DOWN == event.getAction()) {
      if (Color.GREEN == mColor) {
        msg = SimpleThreadingExample.mHandler.obtainMessage(
                              SimpleThreadingExample.UPDATE_SCORE, true);
      } else {
        msg = SimpleThreadingExample.mHandler.obtainMessage(
                             SimpleThreadingExample.UPDATE_SCORE, false);
      }
      SimpleThreadingExample.mHandler.sendMessage(msg);
    }
    return true;







  }

  void setColor(int color) {
    mColor = color;
    setBackgroundColor(mColor);
  }

  void toggleVisibility() {
    if (Color.GREEN == mColor) {
      setColor(Color.TRANSPARENT);
    } else {
      setColor(Color.GREEN);
    }
  }
}
```