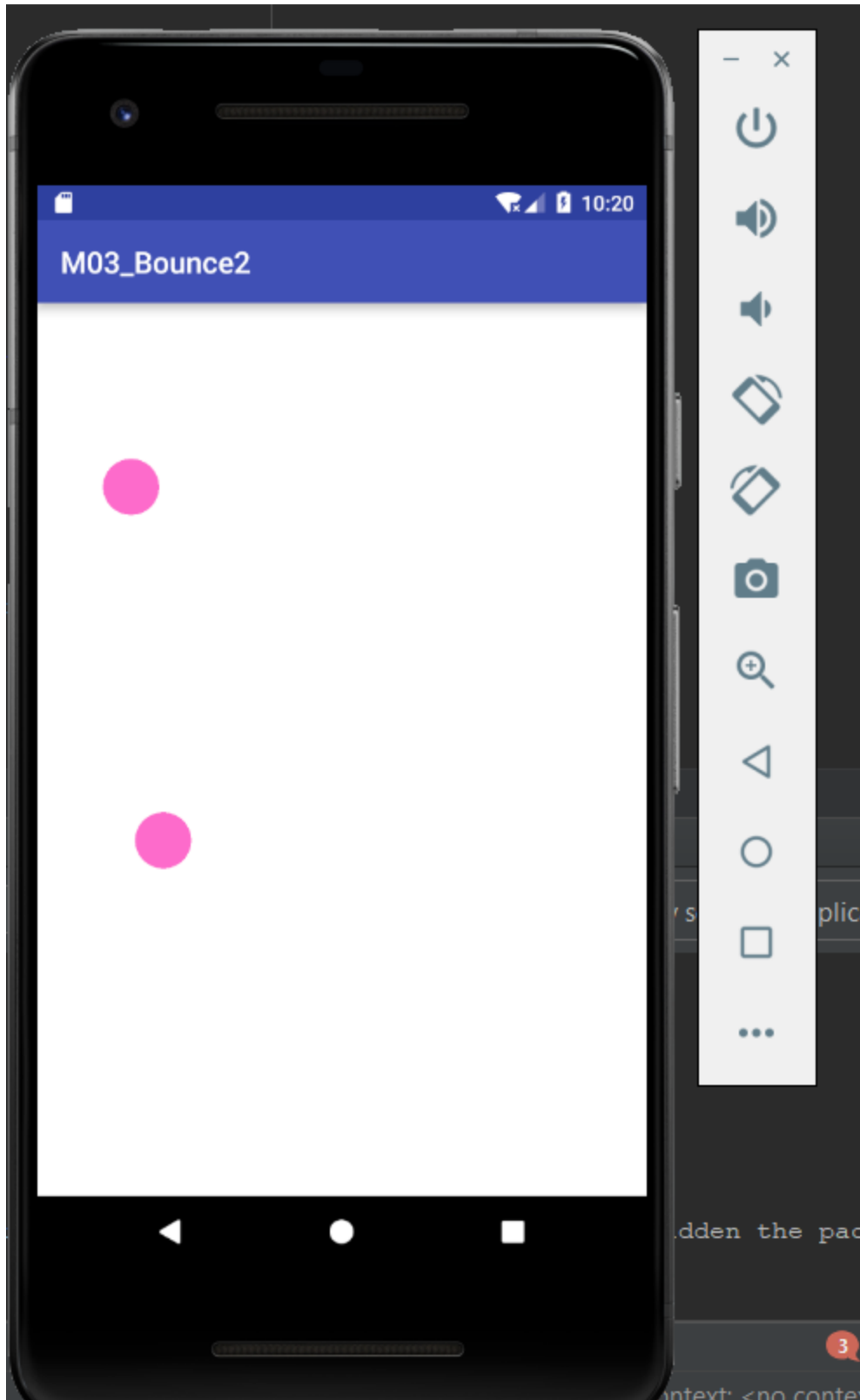


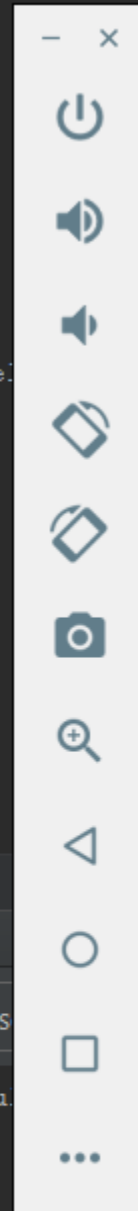
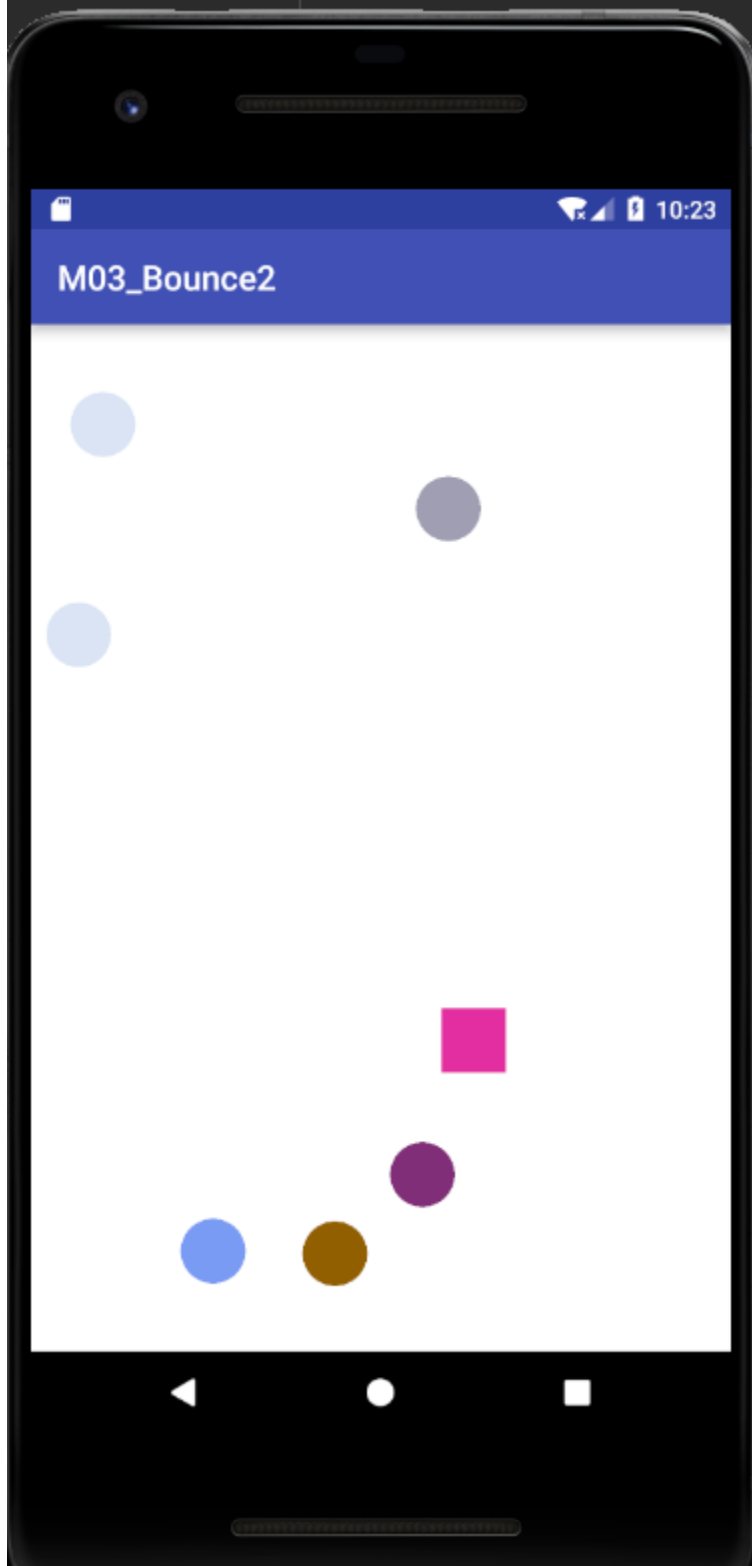
1. Change the box color... i.e. the background colour. What did you do?

```
52
53 // create the box
54 box = new Box(Color.WHITE); // ARGB
55 Random rnd = new Random();
56 int randomColor = Color.argb(alpha: 255, rnd.nextInt(0: 256), rnd.nextInt(0: 256), rnd.nextInt(0: 256));
57
58 //Create colors to pass as random colors.
59
60
```



2. Change the color of newly made balls to a new random color for each new ball. What did you do?

```
public boolean onTouchEvent(MotionEvent event) {  
    Random rnd = new Random();  
    int randomColor = Color.argb(alpha: 255, rnd.nextInt(n: 256), rnd.nextInt(n: 256), rnd.nextInt(n: 256));  
    balls.add(new Ball(randomColor, previousX, previousY)); // add ball at every touch event  
}
```



3. Make the newly made balls go super-fast and super-slow. What did you do?

```
public Ball(int color, float x, float y) {  
    bounds = new RectF();  
    paint = new Paint();  
    paint.setColor(color);  
  
    // random position and speed  
    this.x = x;  
    this.y = y;  
    speedX = r.nextInt( n: 100) + 1;  
    speedY = r.nextInt( n: 100) + 1;  
}
```

Above is new constructor that takes r (previously defined random()) and sets speed x & y to random speeds

4. Try different approaches for invalidate() (different code locations, methods, ...):

1. Does the program still work each time?

From where I tried, it needs to be precisely where it was located in order to work.

2. What does invalidate() do? What happens when it isn't called at all?

I did a bit of reading about this, and from what I can tell it invalidates the previous draw on frame, and requires the program to redraw. When I removed it, it only displayed one frame and then stopped

3. What are the times that onDraw() is called?

onDraw is called every time invalidate is called. I did a small amount of reading about androids library, and from what I understand invalidate is what refreshes ondraw.

5. Add another shape class...fast swipes makes the new shape, slow swipes make circles.

```
if (deltaX > 10 || deltaY > 10){  
    squares.add(new Square(randomColor, previousX, previousY));  
}else {  
    balls.add(new Ball(randomColor, previousX, previousY)); // add ball at every touch event  
}
```

Above code snippet is from onTouchEvent, which takes deltaX and y as the speed of a touch event. If either X or Y surpasses ten, a square is made, otherwise a ball is made.

6. Add a rectangle shape...any time a shape collides with that rectangle you increment a score count (show score on logcat). My square shape is essentially the same as the ball class, just with drawrect. I added this method to check for collisions:

```

public boolean collisionCheck(Shape s){
    // Get new (x,y) position
    x += speedX;
    y += speedY;

    // Add acceleration to speed
    speedX += ax;
    speedY += ay;

    // Detect collision and react
    if (this.x == s.x) {
        return true;
    } else if (this.y == s.y) {
        return true;
    }

    return false;
}

```

I also created a Shape parent class for ball and square, so that it is easy to compare both in collision check.

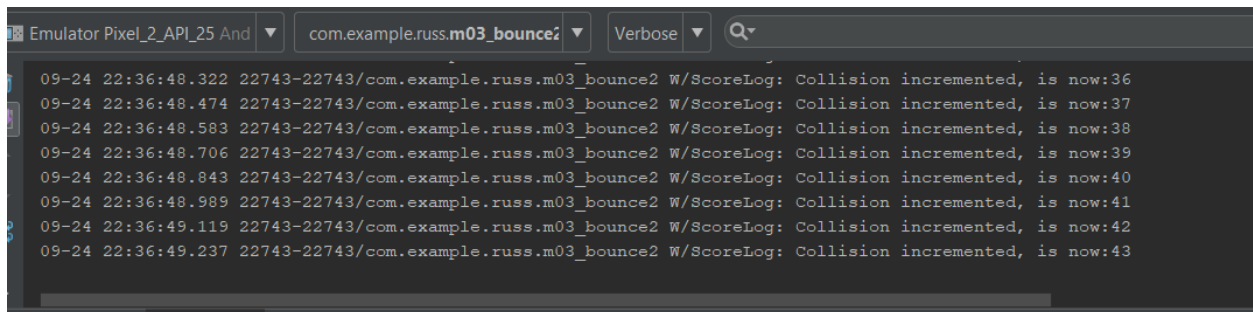
```

for (Ball b : balls) {
    b.draw(canvas); //draw each ball in the list
    b.moveWithCollisionDetection(box); // Update the position of the ball
}
for (Square s : squares){
    s.draw(canvas); //draw each ball in the list
    s.moveWithCollisionDetection(box); // Update the position of the ball

    if(s.collisionCheck(s)){
        s.count+=1;
        Log.v( tag: "ScoreLog", msg: "Collision incremented, is now:" + s.count);
    }
    for (Ball b : balls) {
        if(s.collisionCheck(b)){
            s.count+=1;
            Log.v( tag: "ScoreLog", msg: "Collision incremented, is now:" + s.count);
        }
    }
}
}

```

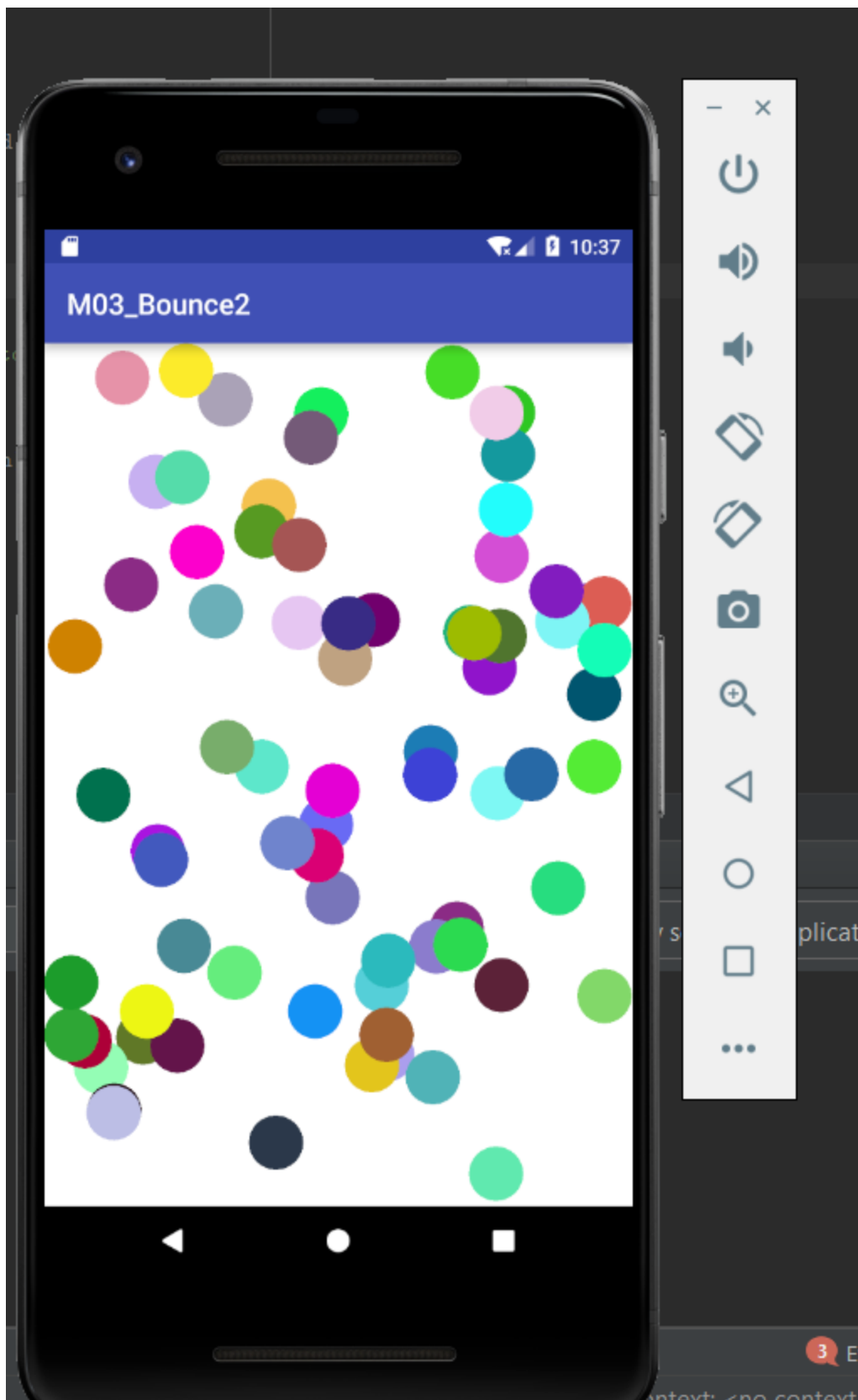
Above is my complex bit of code. This is in `onDraw()` in `BouncingBallView.java`. For each square, it checks if it collides with other squares in the array, then checks if it collides with any balls in the ball array. It then increments the square classes count, and logs it. Below is catlog:

A screenshot of the Android Studio logcat window. The top bar shows the emulator name 'Emulator Pixel_2_API_25 And', the package name 'com.example.russ.m03_bounce2', and the log level 'Verbose'. The log output shows a series of messages from 'W/ScoreLog' indicating collisions. Each message includes a timestamp, a PID, the package name, and a log tag, followed by the message text 'Collision incremented, is now:36' through 'is now:43'.

```
09-24 22:36:48.322 22743-22743/com.example.russ.m03_bounce2 W/ScoreLog: Collision incremented, is now:36
09-24 22:36:48.474 22743-22743/com.example.russ.m03_bounce2 W/ScoreLog: Collision incremented, is now:37
09-24 22:36:48.583 22743-22743/com.example.russ.m03_bounce2 W/ScoreLog: Collision incremented, is now:38
09-24 22:36:48.706 22743-22743/com.example.russ.m03_bounce2 W/ScoreLog: Collision incremented, is now:39
09-24 22:36:48.843 22743-22743/com.example.russ.m03_bounce2 W/ScoreLog: Collision incremented, is now:40
09-24 22:36:48.989 22743-22743/com.example.russ.m03_bounce2 W/ScoreLog: Collision incremented, is now:41
09-24 22:36:49.119 22743-22743/com.example.russ.m03_bounce2 W/ScoreLog: Collision incremented, is now:42
09-24 22:36:49.237 22743-22743/com.example.russ.m03_bounce2 W/ScoreLog: Collision incremented, is now:43
```

7. Think of another change (...and do that change) yourself, ...What did you do?

I changed the amount of balls possible on screen at once. It was quite the fun sight to see.




```
// A way to clear list when too many balls
if (balls.size() > 3000) {
    // leave first ball, remove the rest
    Log.v( tag: "BouncingBallLog",  msg: "too many balls, clear back to 1");
    balls.clear();
    balls.add(new Ball(randomColor));
    ball_1 = balls.get(0); // points ball_1 to the first (zero-ith) element of list
}
```