Laboratory Eight  
Smartphone Apps

## Basic Concepts

Today we will look at writing a software application for the Android smartphone hardware we will be discussing in the lecture. Our application is an arcade game called **breakoutGate**, shown at right. The game uses a software game engine called **AndEngine** and the game is written in **Java**. Your lab instructor will discuss the following concepts:

1. Software applications may be developed on the Android through an Android software development kit (SDK), which is based upon the Java development kit (JDK).
2. To write Java code for the Android, using the Android development tools (ADT), an interactive development environment (IDE) called **Android Studio** (an editor plus more) is provided. All the Android software tools are available at: [http://developer.android.com/sdk/installing.html](http://developer.android.com/sdk/installing.html)
3. Games are typically developed through the use of special software add-ons called engines that provide physics and animation. We will be using code from **AndEngine**.

There is no prelab exercise associated with this lab.

## Task One: Load the App and Go

The first thing to with the Android Studio is to load the breakoutGate app into the emulator.

1. **Run Android Studio.** Go to the home directory and find the Android Studio folder. Run Android Studio.
2. **Load the code into Android Studio.** The breakout project can be downloaded from github.com, otherwise your lab instructor will tell you where to find the source code.
3. **Run breakoutGate.** Go to Android Studio and press **SHIFT F10** to start the game. It should show you the **NAND** bricks forming **inverter** circuits like shown above.
4. **Download the app.** Try downloading the app into your own Android phone (you need version 2.3.3 or better):
   a. Connect your Android with the micro USB cord
   b. Find breakout.apk in the bin folder on your PC.
   c. Drag it into your Dropouts Android folder. Click it.

### Android Studio Short cuts:

1. **CTRL F10** to begin code execution, i.e. Run
2. **CTRL F** to search for a word. (**Never use the replace option. Use Refactor instead.**)
3. **CTRL ALT L** to reformat the code into proper Java form
Task Two: Some things to try

Here is a list of some things you can modify in the software to alter the game. Please refer to the emulator screen dimensions and variables shown at right.

1. Alter the velocity of the ball. Change the value of DEMO_VELOCITY in the beginning of the code.
2. Enter your name on the screen. Search for nameBox (CTRL F) and retype “Cody Herring” (original idea maker student) with your name. Center it on the line.
3. Change the color of the NAND gate brick. Search for “NANDlargeBlue.png” and retype it with “NANDlargeRed.png” (see gfx assets picture below).
4. Make SOP NAND circuits. Now let’s try some harder things. Right shifting every other row of NAND gates by ½ brick will cause the NAND gates to form SOP circuits. Your instructor will show you how to use an if else statement to do this, shifting every other row of gates (the back gates) by $\Delta x = delX = blockWidth/2$.
5. Making proper form SOP circuits. The SOP circuits in Step 4 are not drawn in 2-level proper form. Replace the shifted Red NAND gates with Green OR2B2 gates (see gfx assets below).
6. Increase the number of gates. Now replace all of the large gates with small gates from the gfx assets directory and increase the number of rows of gates in the display. (If the ball launch is interfered by the lower rows of the gate bricks, lower the ball launching point on the screen.)
7. Pong Ball. Launch the balls from the middle of the pong.
8. Advancing Gates. Move rows of hungry NANDs downward toward the shooter, like an alien arcade game. Feed more rows down from the top.
9. Curved Pong. Ball deflects at non 45° at pong edges. (Use curved pong.)
10. Other ideas? Good ideas can earn extra credit!