Building Capable, Energy-Efficient, Flexible Visualization and Sensing Clusters from Commodity Tablets

Thomas Delgado Dias  
*Loyola University Chicago*

Xian Yan  
*Loyola University Chicago*

Konstantin Läufer  
*Loyola University Chicago*, klaeufer@gmail.com

George K. Thiruvathukal  
*Loyola University Chicago*

---

**Recommended Citation**

DroidWall: An Energy-Efficient Video Wall of Android Tablet Computers

T. Delgado • X. Yan • K. Läufer • G. K. Thiruvathukal
Emerging Technologies Laboratory • Loyola University Chicago • USA
home.etl.luc.edu • laufer@cs.luc.edu

Abstract/Position Stmt.
We propose to build an inexpensive, energy-efficient 4x4 video wall from off-the-shelf 7" Android tablet computers. These tablets will form an innovative computational cluster along the three dimensions of video display, central and graphics processing units (CPU and GPU), and input sensors, thereby forming a whole that is greater and more powerful than the sum of its parts: In particular, their combined display resolution is eight times that of commercially available flat screen monitors at roughly the same cost.

Communication: Skeenzone
To enable communication among our tablets, we have built our application on top of the Skeenzone middleware for distributed mobile applications:
http://code.google.com/p/skeenzone

Programming Language
What is Scala?
Why Scala for Android development?
• statically typed
• runs on Java Virtual Machine
• emerging as a development option for building Google Android applications
• hybrid object-oriented/functional language
• concise (up to a factor of ten less code compared to Java)
• Scala is faster in some cases
Google has published a paper comparing the performance of four programming languages, C++, its own language Go, Java and Scala. Their conclusion is: Scala provides the most concise notation and best constructs for managing code complexity, and the second-best performance after C++.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Time[Sec]</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++ Opt</td>
<td>23</td>
<td>1.0x</td>
</tr>
<tr>
<td>Java 64-bit</td>
<td>134</td>
<td>8.6x</td>
</tr>
<tr>
<td>Java 32-bit</td>
<td>290</td>
<td>12.6x</td>
</tr>
<tr>
<td>Scala</td>
<td>82</td>
<td>3.7x</td>
</tr>
</tbody>
</table>

DroidWall
The tablets run the open-source Android operating system, which allows us to install arbitrary custom software. As a result, the proposed wall amounts to a testbed that supports an unprecedented degree of experimentation. Given that each tablet is equipped with a touch screen and various environmental sensors, the wall’s technical capabilities form a “trilogy” of visualization, sensing, and control.

Use Cases
Numerous applications are imaginable!
• environmental and security monitoring
• exploring a three-dimensional molecule
• teaching the color space to art students
• visualizing relationships among versions of a text
• calculating and visualizing the energy footprint of an individual or group
These initial ideas will tie in with a range of research questions in both technical and application domains.

Layout
• Comprises 16 tablets (4x4) connected through a rack which provides energy and support for those tablets.
• Intended to act like a video wall.
• Dynamically reconfigurable as needed.

Setting Tablet Orientation
How do we determine the tablets orientation? It’s up to the programmer! Using the strategy pattern, it’s possible to elegantly add different kinds of orientation logic.

public void setOrientation(int orientation)
{ switch(orientation){
  case Orientation.HORIZ:
    this.orientation = new Orientation(); break;
  case Orientation.ROT90_RIGHT:
    this.orientation = new Orientation(); break;
   case Orientation.ROT180:
    this.orientation = new Orientation(); break;
   case Orientation.ROT270_LEFT:
    this.orientation = new Orientation(); break;
}

public class RoomActivity extends Activity {
  // Java [not tested]
  private EditText usernameEditText;
  private EditText passwordEditText;
  private Button loginButton;

  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.home);
    setContentView(new View.OnTouchListener() {
      @Override
      public boolean onTouch(View v, MotionEvent event) {
        switch (event.getAction()) {
          case MotionEvent.ACTION_DOWN:
            return true;
        }
    });
    loginButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onCheckboxClick(View view) {
        // do something useful
      }
    });
    loginButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        // do something useful
      }
    });
  }