Converting Computer Science Educational Applets to Mobile Devices

Dino Schweitzer and Scott Teel  
Department of Computer Science, USAF Academy, CO, USA

Abstract - Many interactive learning tools have been developed to teach students concepts in computer science. Tools such as web-based animations for distance learning, simulators and emulators, algorithm visualizations, classroom demonstrations, and virtual laboratories are all used to create hands-on active learning environments. To support our introductory course in Computer Graphics, we created over 30 educational applets each focusing on a specific concept in the course. The applets were initially embedded in web pages that explained the concepts and provided instructions and challenges for students to complete. The tools were used in the classroom to support lectures. To take advantage of mobile technology and experiment with other form factors, we converted the tools to an Android-enabled phone platform. This paper will briefly describe the tools and our experience with converting them from a web-based format to a mobile environment.

Keywords: educational software, mobile computing, active learning

1 Introduction

Active learning is a broad term applied to many forms of educational approaches that attempt to actively engage the students in the learning process [1]. Other terms, such as “cooperative learning”, “problem based learning”, “learning centered focus”, and “just in time teaching” refer to specific ways of implementing an active approach to increase student’s abilities for higher-order learning. Within Computer Science, many educators have looked at ways of employing active learning approaches such as acting out algorithms, small group problem solving, collaborative activities, games, and classroom worksheets [2].

An approach to active learning that we have successfully implemented in different courses within the computer science department at the Air Force Academy is the creation of interactive classroom visualizations (ICV’s) [3]. ICV’s are short interactive demonstrations of a concept being presented that allow students to experiment with the concept, typically in a classroom setting. The characteristics of ICV’s include:

- High interactivity
- Easily understood abstraction
- Simple
- Relevant
- Robust

We have developed and applied ICV’s in our security course, in our core computer science course, and in various other upper division courses. Most recently we created a set of ICV’s for use in our Introduction to Computer Graphics course. 30 tools were developed to provide students with hands-on interactive experience with specific topics in computer graphics. The complete list of tool topics is shown in Table 1.

### Table 1. List of computer graphic interactive tools.

<table>
<thead>
<tr>
<th>Color lookup tables</th>
<th>Preattentive perception</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color space</td>
<td>Graphic Primitives</td>
</tr>
<tr>
<td>Aliasing</td>
<td>User interaction</td>
</tr>
<tr>
<td>Barycentric cords</td>
<td>2D transformations</td>
</tr>
<tr>
<td>Fill algorithm</td>
<td>Perspective</td>
</tr>
<tr>
<td>2D Viewing</td>
<td>3D Viewing</td>
</tr>
<tr>
<td>Quadtrees</td>
<td>Fractals</td>
</tr>
<tr>
<td>Particle system</td>
<td>Ray tracing</td>
</tr>
<tr>
<td>Shading</td>
<td>Shadows</td>
</tr>
<tr>
<td>Frame rate</td>
<td>Morphing</td>
</tr>
<tr>
<td>Advanced ray tracing</td>
<td>Image formats</td>
</tr>
<tr>
<td>Seam carving</td>
<td>Convolution</td>
</tr>
<tr>
<td>Blending</td>
<td>Visualization</td>
</tr>
<tr>
<td>Toon shading</td>
<td>Face morphing</td>
</tr>
<tr>
<td>Phonemes</td>
<td>Anaglyphs</td>
</tr>
</tbody>
</table>

The tools are designed to be used in the classroom after a lecture on the topic. Each tool is embedded in a web page that describes how the tool works, provides sample activities, and gives the students a “challenge” question.
that they are to solve after experimenting with the tool. Figure 1 shows a sample screen shot from one of the tools.

The tools are each written in Processing, a simplified Java-based open source programming language and development environment (www.processing.org). Processing was chosen as the primary programming language for the course due to its simplicity to learn, its support of computer graphics and animation, and the ease of creating rapid prototypes that can be exported to an applet with the click of a button. Students wrote four different computer graphics programming projects in Processing. To assist students in learning this language, all of the interactive tools are written in Processing and source code is available to the students to download, review, and reuse in their own exercises.

2 Converting to a Mobile Platform

The computer graphic tools were successfully used to teach the course in Spring 2010. Interest was expressed amongst students and faculty on the feasibility of running the tools on a mobile device. Part of the interest was based on a lot of hype around the capabilities and flexibility offered by mobile computing. In addition, the department wanted to experiment with including mobile programming as exercises in some of the computer science courses to enhance student interest and motivation. Since the developer of the graphic tools had no experience with programming for a mobile device, it was deemed an interesting and informative exercise to convert the existing tools to a mobile platform. Although some thought was given to how to use the tools in an educational setting once converted, the focus of this paper is on the conversion experience itself.

2.1 Choice of Platform

The first decision in the conversion process was what device to target for conversion. Initially, the iPhone was thought to be the top choice, primarily because of the popularity of the device, the number of applications already available, and the amount of information online for how to develop applications for it. The iPad was also considered as it has a nice form factor with a much larger screen size and a high “coolness” factor. A downside of the iPad, due to its newness and cost, is that there are currently not many devices around. Also, since development for the iPhone and iPad requires objective C, a substantial rewrite of the applications would be required. In the course of investigations, Android mobile devices became more popular and offered the advantage of an open-source environment. With the promised introduction of an Android-based tablet in the near future, and the ability to run Java, this platform was chosen as the target for conversion.

As a strong side benefit of choosing Android, Processing recently released a Beta Version (0184) of their programming environment which directly supports Android devices. This allowed us to experiment with converting existing applications without changing the language, basic program structure, or development environment. Figure 2 shows a simple application written in Processing for the Android. The program runs in the Android emulator as shown for debugging.
purposes. When an Android device is plugged into the computer in the proper mode, the code downloads to the device directly from Processing when you run the program.

![Figure 2](image2.png)

Figure 2. Simple application written in Processing running on the Android emulator.

### 2.2 Challenges in Converting Tools

At the beginning of this project, the authors had little experience developing applications for a mobile environment, and did not know the limitations or pitfalls to expect. The fact that Processing supports Android development greatly simplified the initial conversion. It was quickly discovered, however, that Processing 0184 is, in fact, a Beta Version, and not without bugs. We found some of the bugs were related to using Processing in a Windows environment, and had more success running the environment on Ubuntu 9.10. Surprisingly, at least to us, we were able to successfully compile and execute over 40% of the applets without any modifications to the code. User interfaces such as button presses, slider bars, keyboard input, text fonts, and dragging objects all worked more or less the same on the Android as the web-based versions. While using a finger as the mouse did not offer the same positional accuracy, it was sufficient for most of our required interactions.

One user interface that required changing was any application that used the Processing mouseButton to indicate right or left button click. Since, there is no mouse button in the mobile environment, these interactions had to be changed to a different user interface, such as using a button to set a mode. Another shortcoming documented in the literature which did not affect our applications, is regarding mouse rollovers. For example, if an application changes the color or appearance of a button when rolled over, this would require change in the mobile environment. We did not use this in any of our applications.

Another minor inconvenience was the Java Point class was not available. It was easy to fix by adding our own class with necessary functionalities. A more challenging problem was that several of the applets demonstrate real-time 3D interactions such as transforming 3D objects into different viewing positions. In Processing, the P3D render mode allows for fast drawing to facilitate real-time interaction. The P3D mode is not available for Android applications and resulted in slower rendering. Since any time delay in user interaction can be a distraction, some of the 3D interaction had to be curtailed to make the tool more useable.

Another design challenge was dealing with the smaller available screen real estate. Although most of the educational applets were relatively small, the default 320x480 portrait mode for the emulator was a tight fit for some of them, and one larger demonstration had to be significantly resized. We found we could force the Android screen to open the application and remain in landscape orientation (no matter what the phone orientation happened to be), effectively giving us a 480x320 screen. The test phone platform, an HTC Incredible, has a higher hardware resolution of 480x800, and newer platforms will have increasingly larger sizes which will make the current screen resolution limitations a non-issue for these applets. However, even with the higher resolution, it was necessary to split the accompanying explanatory text
shown in Figure 1 from the application. This is not ideal, as you would prefer the student to be able to look at the tool on the screen while reading the directions. We made use of the Android menu’ing capability to make the instructions and exploration text available on demand when the hardware menu button was selected.

Accessing the Google Android API calls from within Processing, such as the screen orientation and menu’ing capability, was straightforward. You simply import the appropriate android.* package(s) into your Processing code, and then make the call to the desired package method(s). For example, to lock the screen orientation to landscape mode, you use the "setRequestedOrientation()" method from the android.app.Activity class as shown:

```java
import android.app.Activity;
import android.content.pm.ActivityInfo;

void setup(){
    // any initialization code
}

void draw(){
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
    text("Landscape mode",150,20);
}
```

Similarly, to add a menu when the hardware menu button is pressed, use the android.view.Menu package as shown:

```java
import android.view.Menu;
import android.view.MenuItem;

Time myTime;
public static final int SHOWTIMEZONE = 2;

void setup(){
    myTime = new Time();
    myTime.setToNow();
}

void draw(){
    text("Press the Menu button to show your choices.",5,20);
}

public boolean onCreateOptionsMenu(Menu menu){
    menu.add(0,SHOWTIME,0,"Show Time");
    menu.add(0,SHOWTIMEZONE,0,"Show Timezone");
    return true;
}

public boolean onOptionsItemSelected(MenuItem item){
    switch (item.getItemId()){
        case SHOWTIME:
            text("24-hour clock Time is: ",5,150);
            text(myTime.format("%H%S"));
            return true;
        case SHOWTIMEZONE:
            text("Current Timezone is: ",5,200);
            text(myTime.getCurrentTimezone());
            return true;
        }
        return false;
    }
```

With relatively minor changes, and adding some API calls, we were successful in converting all of the graphic applications to the Android.

3 Future Plans

While the overall process of converting existing tools to a mobile environment was successful, it was primarily an experiment to determine what the issues were with converting educational applications developed for a full-screen web browsing environment. A more meaningful exercise is to modify or develop tools and exercises that take advantage of the mobile environment. Educators are starting to look at how to integrate mobile devices into traditional CS courses using approaches such as providing toolkits that allow students to easily complete programming projects on mobile platforms [4]. Other researchers are trying to identify what attributes of mobile computing can be taken advantage of to enhance the educational process [5]. While it is well understood that students are motivated by using the latest technology, are there unique aspects provided by the new form factor and capabilities that extend more traditional approaches?

At our institution, we have made a commitment to finding ways of integrating mobile technology into our CS curriculum, both as a development platform for interesting student projects, and as a technology for providing unique teaching capability. For example, a mobile tablet that integrates teaching tools such as those listed above along with the supporting text, or finding ways to utilize location-based information along with teaching concepts. Are there things that can be done with the accelerometer to reinforce concepts, beyond being a “gee whiz” interface device? What can be done with the GPS or camera capabilities of mobile devices? These are questions that our faculty are beginning to investigate. We are setting up a mobile laboratory with different technologies and development environments to explore some of these issues. The results from this conversion exercise are encouraging that students can use a simplified environment, such as Processing, to develop and load applications and access the API calls to create interesting and motivational projects.
4 References


