Journal of Information Technology Education: Innovations in Practice

Volume 12, 2013

A Template-Based Short Course Concept on Android Application Development

David Akopian, Arsen Melkonyan, Santosh C. Golgani, Timothy T. Yuen, and Can Saygin
University of Texas at San Antonio, San Antonio, Texas, USA

david.akopian@utsa.edu; arsen.melkonyan@utsa.edu; bottachandana@gmail.com; timothy.yuen@utsa.edu; can.saygin@utsa.edu

Executive Summary

Smartphones are a common accessory to provide rich user experience due to superior memory, advanced software-hardware support, fast processing, and multimedia capabilities. Responding to this trend, advanced engineering systems tend to integrate mobile devices with their solutions to facilitate usability. With many young students showing interest in learning mobile application development, conventional electrical engineering undergraduate education cannot meet the needs of this workforce due to fast changes in mobile technology and limited curricula hours. Template-based learning (TBL) methods may overcome these limitations by shortening the learning cycle through fast hands-on introduction to development tools, basic programming, and application development and integration process. Students manipulate code fragments in provided templates, and compile, embed, and run applications. They also implement new applications reusing fragments from other similar templates. TBL modules can be integrated in pre-existing conventional courses to provide basic and fast exposure to the subject. This paper provides an example of a TBL template library for Android phones, which has been used in a classroom setting to collect student attitude data and assess efficiency of the TBL approach.

Keywords: Learning, Mobile Applications, Mobile Education

Introduction

Mobile phones are technologically advanced devices that provide more and more communication and other services. Numerous applications exploit superior memories and cameras, various wireless channels for voice and data, inertial and satellite positioning systems, advanced signal and graphics processors and accelerators, etc. Acquisition, processing, and visualization of various media also are common due to advanced software platforms and applications.

Educators are inspired to use cell phones to communicate learning content in new formats as Millennials (ages 18-34) are by far the most technologically advanced user group, with a 95% cell phone ownership (Caverly, Ward, & Cavarly, 2009; Zickuhr, 2012). Mobile learning, or m-learning, is defined as the acquisition of any knowledge or skill through using mobile technology (Geddes, 2004). This concept extends to

Editor: Jane Nash

ubiquitous learning (Sakamura & Koshizuka, 2005), the concept of learning "anything at anytime and anywhere," as people spend more than 50% of their time outside their office or classroom (Hayes, Joyce, & Pathak, 2004) and essential learning subject exists in our daily environment (Laine, Sedano, Joy, & Sutinen, 2010). Mobile learning also is related to e-learning, which enhances education through access to learning materials over the Internet (Rosberg, 2001). All of these concepts belong to a broader framework of technology enhanced learning (TEL) (Lytras, Gasevic, & Ordonez De Pablos, 2008), which exploits technological innovations to improve the efficiency and cost effectiveness of traditional teaching methods.

In Japan, a survey of 333 Japanese university students concerning m-learning (Thomton & Houser, 2004) revealed that 100% of them own a mobile phone, 99% send email on their phones, exchanging some 200 email messages each week. 66% email peers about classes; 44% email for studying. In contrast, only 43% email on PCs, exchanging an average of only two messages per week. Most of the subjects preferred receiving educational materials on mobile phones rather than PCs and liked using phones for teaching. Recent examples of m-learning efforts can be found in Tribal's Digital Learning Studio (www.mlearning.org) for example.

M-learning likely will enhance education by closing digital gaps as the largest growth in mobile phone ownership is predicted to come from people from low socio-economic status (Portio Research, 2012). It also essentially will affect education in developing countries, where mobile phones have a much higher penetration rate than laptop and desktop computers (Ullrich, Shen, Tong, & Tan, 2010).

Even though there are psychological, pedagogical, and technical limitations for m-learning (Shudong & Higgins, 2006) (small screens, typing inconvenience, etc.), it can be claimed that, in general, students are willing to use their mobile devices for educational purposes.

Most m-learning applications are designed for content delivery and learning facilitation. For example, mobile apps such as myHomework (https://myhomeworkapp.com/) allow students to track their course material on their smartphones. In addition to being a platform for content delivery and user experience enrichment, the mobile phone can also serve as a "technology kit," which can be used to learn computing aspects and application development itself.

With almost four billion wireless connections in 2008 worldwide, the cell phone industry generated vast services and large job markets. Undergraduate education should address this demand by properly training the workforce for the future. Unlike computer and information sciences (Tam & Chen, 2006), electrical engineering (EE) students have very limited exposure to mobile programming but must work in multidisciplinary fields where mobile devices are incorporated in control chains of various systems.

The diversity of mobile platforms, dramatic changes in industry, and limited curricula hours make it challenging to integrate mobile computing courses in conventional undergraduate EE programs. Figure 1 schematically illustrates significant market share redistributions of mobile phone platforms in evolution using data from the Gartner Smartphone Marketshare Reports (www.gartner.com). Android OS platform (http://developer.android.com/index.html) recently gained popularity as an open source platform that relies on Java language for mobile application development. Even though platform penetration dramatically changes over time and other platforms may prevail in the future, Android is attractive for education needs as it provides free development resources, which will help to scale up successful efforts for broader dissemination and student adoption. Additionally, many student-owners of Android phones can work within a platform that they are familiar with and use on an everyday basis.

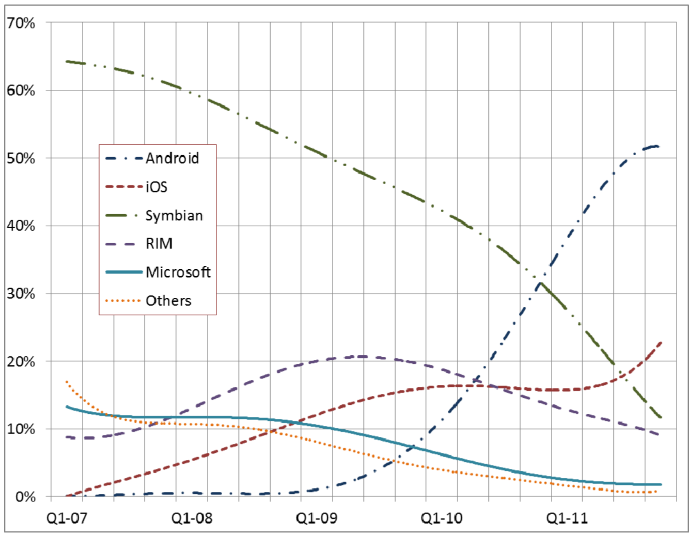


Figure 1: A schematic illustration of worldwide mobile smart-phone OS platform market share evolution, from 2007 to 2011 using data from (www.gartner.com).

This paper investigates template-based learning (TBL) for Android mobile application development to help overcome EE program constraints as described above. The idea is to develop learning modules that can be integrated in electrical engineering courses (e.g., signal processing or communications courses). Due to short cycles, students learn by exploring already available templates, which can be manipulated to a limited extent to gain initial application development experience. While the idea of using template designs in programming was exploited previously (Al-Imamy, Alizadeh, & Nour, 2006; Schank, Linn, & Clancy, 1993), this paper extends a similar approach to mobile programming. As cognitive aspects of learning in mobile programming are not addressed fully yet, we rely on our own learning experiences to hypothesize concerning efficient learning strategies. Particularly, two learning alternatives can be considered: (1) initial fast-track exposure to the subject to see the big picture, and (2) conventional routine acquisition of knowledge and experience.

The focus of the paper is on the first stage of mobile programming, which introduces students to the subject. The sample codes are available in a predesigned library of application templates that can be used or manipulated. In this approach, students can quickly develop functional apps without having to go into depth with mobile programming or spend a great deal of time coding from the ground up. The library can be distributed or placed on a server for remote development as well. Topics about signal and image processing and wireless communications can be studied in relation with the hands-on labs on mobile devices. Camera images and video, audio and voice signals, wireless connectivity standards, signals from accelerometers, and many more built-in phone features can be used in designing such labs.

Thus, the research question is the following. How efficient can template-based short learning modules be as fast introductions to challenging areas of mobile programming typically covered by a dedicated course or sequence of courses? Here, “efficiency” is understood as a transformative factor: (a) change of conventional attitudes; (b) gaining confidence; and (c) ability to successfully complete assignments in the focus area and to develop their first apps independently.

The rest of the article has been organized further in the following way. A brief description of the proposed template-based education philosophy is presented. Development tools are described in the following section. Then, examples of template mobile applications are reviewed and one of them is presented in detail. The next section describes the image processing toolbox to demonstrate how templates can be collected in libraries. A quiz application developed by the authors is briefly described as a relevant supportive self-assessment tool. The next section describes a case study using TBL in a classroom setting and provides evaluation results by analyzing survey results concerning student attitudes and assignment grades. Finally, the research effort is summarized in the conclusion.

Philosophy of the Approach

This section addresses the rationale of a template-based approach and its relation to the state-of-the-art. It is necessitated by the fact that formulating structured solutions, understanding program execution, and learning a rigid syntax and commands can be challenging for beginners of all ages. Multiple programming languages have been created and environments have been built to make programming more accessible to beginners, particularly young programmers. Several programming languages and development environments were designed for the purposes of making some of the abstract concepts more concrete, alleviating some of the cognitive load associated with problem solving and computation, and motivating students to learn programming. For instance, BlueJ, the interactive Java environment, (www.bluej.org) is an integrated Java environment specifically designed for introductory teaching about object-oriented programming through code visualization. Another example is Scratch (http://scratch.mit.edu/) from MIT Media Lab, which was designed to help young children to program through a drag-and-drop interface. After being exposed to an easier learning experience, students start programming using general-purpose languages and professional development tools.

Educational programming environments have many advantages for beginners compared to professional text-based and visual programming tools. Many of the abstract concepts have been hidden from the user as well as the advanced tools that are typically associated with industry-grade development environments. At the same time, transition to professional tools might be challenging. In some of these programming systems, students might focus on the fun part of the system more and forget about the main goal, which is programming. Similarly, design tools such as MIT App Inventor (http://appinventor.mit.edu/) facilitate application development, but do not expose students to the "real development" world. Rather, it provides an easy-to-understand interface to app development.

TBL's goal is different. It is to provide real experiences with professional tools in a short time. Exposure to the big picture has its own educational value; it will alleviate technology fears and may motivate further in-depth studies. Using a metaphor, kids can use toys to pretend to help their parent fishing, but helping the parent with small tasks during "real fishing" with real tools is a completely different experience.

Unlike conventional application development learning approaches (Lutes, 2012) the TBL process will provide students with ready-to-run programs, giving them the ability to modify those and immediately experience the programming process and see the results. The goal is to educate using a compiling and integration process, familiarize them with software development kits/IDE, and provide them "real" exposure to the professional world. For more advanced students, the templates will help to shorten the coding cycle by reusing samples from templates. A well-designed template could be used to support the learning process as well as the software development process.

Templates have been used to facilitate conceptual understanding of programming concepts (Al-Imamy et al., 2006; Schank et al., 1993; Yuen & Liu, 2011). Yuen and Liu (2011) studied how a video game template supported and guided conceptual understanding of object-oriented programming (OOP) as students used it to create their own interactive multimedia games. In using templates, Yuen and Liu found three levels of interaction that result in better conceptual understanding. First, visual feedback from the games, specifically unexpected behaviors, brought attention to flaws in the software design. When students encounter these errors, they begin another level of interaction with the template, which is to explore the template resources. This exploration can lead to the discovery of important information, discovery of relationships between resources, and model code that can later be used. As they explore, students begin to refine their code as well as their understanding of the code. At the cognitive level, the template scaffolds conceptual understanding as it provides the foundation or incomplete code base from which students can build their programs. If designed correctly, the template code can be used as models or exemplars that guide students' thinking and coding.

In the case of the course described in this paper and in Yuen and Liu (2011), templates are used as cognitive tools. Cognitive tools are computer-based tools that assist learners in the reorganization of cognitive structures to support higher order cognitive activities (Jonassen, 2006). The purpose is to engage learners in problem-solving tasks that are within their zone of proximal development; that is, the cognitive tools help learners go beyond what they can do on their own by providing necessary scaffolding (Vygotsky, 1978). Scaffolds generally include cues and prompts, models, and guidance, which are tailored to individual students. Eventually, assistance is slowly retracted, known as fading, which results in the learner being able to accomplish those tasks alone.

As the learning modules are offered in lab formats, including step-by-step procedures, experience with tools will be educative and not be stressful. The beginners will learn progressively, from basic to more advanced modules. Templates use graphics and animation for more appealing experience.

Learning using templates abstracts many of the development steps by pre-populating design and code modules. They automate the packaging and deployment processes to make the entire software development lifecycle less intimidating and more obvious for students. These simplifications allow students and their instructors to focus their attention on the broader picture before concentrating on coding details. The differences between template-based programming and regular programming are illustrated in Figure 2. There are seven steps in the creation of an Android application. These steps include designing, coding, compiling, pre-verification, packaging, testing, and deployment. The steps are illustrated in Figure 2 in blue. The color red shows a shortened cycle of TBL, where designing and coding stages are simplified to code manipulation, and the testing stage is a primitive visual validation of results. The other four stages (in Eclipse IDE box) are the same in both approaches.

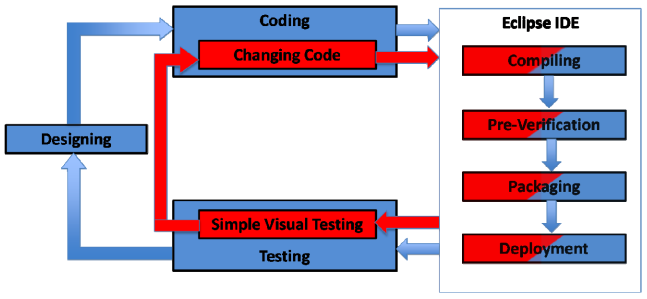


Figure 2: Template-based system vs. conventional programming process.

Development Tools

The templates are implemented in the Java programming language ("JAVA tutorial," 2012) using the Android Software Development Kit (SDK) (http://developer.android.com/index.html) in the Eclipse Integrated Development Environment (IDE) (www.eclipse.org). Eclipse is a convenient development environment, which provides an extensible plug-in system. Executable 'apk' files are deployed onto Android compatible mobile devices. Students who are familiar with Java programming can easily work with Android SDK. The Android Developer Tool (ADT) is an Eclipse plug-in, which provides different versions of emulators replicating Android phones. Sample image processing templates are available in a JJIL open source image processing library, developed in Java (http://code.google.com/p/jjil).

Template Apps Used

The following section describes the template applications used for the projects. It includes a conventional “Hello World” application, which is used to learn the development environment and perform basic modifications. Then, progressively, other template apps are described along with related learning aspects. The application “Simple Animation” is chosen for detailed explanation while others are just summarized.

Learning Modules

“Hello World” Application (Figure 3a): The “Hello World” program is typically the first program that students learn when introduced to a programming language. This application is designed to introduce the first timer to fundamentals of syntax and building a user interface, which in this case displays “Hello World” on the screen. The objectives for this unit were to have students be able to do the following:

- launch the Eclipse IDE and Android SDK on a computer,
- become familiar with the Java language,
- understand how different objects in Java interact,
- change the message text in the template code,
- test applications in Android SDK emulator, and
- install and run applications on the mobile devices.

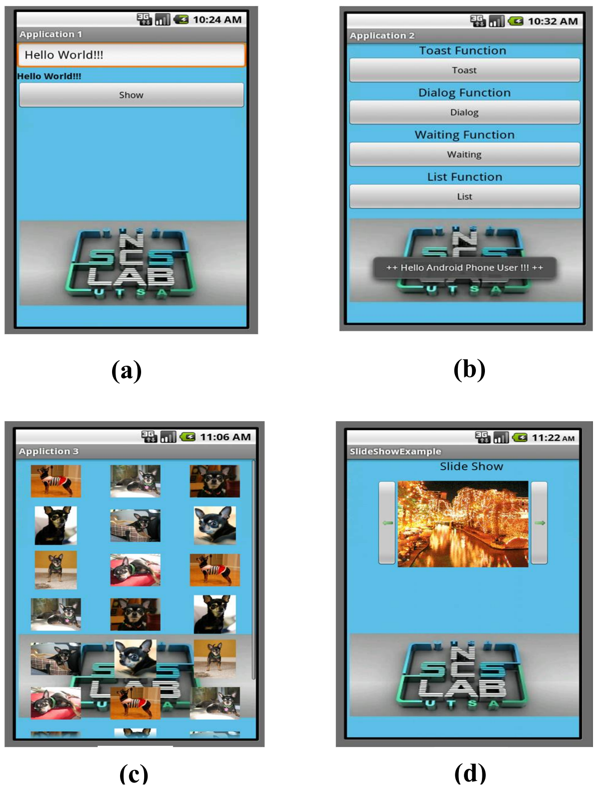


Figure 3: (a) ‘Hello World’ Application, (b) ‘On-Click Example ‘ Application, (c) ‘Image Gallery’ Application, (d) ‘Slideshow’ Application

“OnClick Example” Application (Figure 3b): The “OnClick Example” application demonstrates the addition of buttons and handling events that activate upon clicking. It uses texts and other visual effects to manifest the changes that happen on the screen in response to user interaction. From studying this example, students are expected to learn:

- the significance of methods used in Java,
- how to create new objects in Java,
- how methods work,
- handling events in Java, and
- to customize and assign events or tasks to buttons.

“Image Gallery” Application (Figure 3c): This example helps build an image gallery application with the scrolling effect applied when scrolling up and down from one set of images to another. This application introduces the following topics:

- the digital representation of an image,
- creating an array and managing its content,

- resizing digital pictures, and
- giving functionality to the up-down scroll panel.

"Slideshow" Application (Figure 3d): In this example, students use the template to build an image slideshow application by using images from both project resources and the Internet. The application contains buttons that handle all user actions to move from one image to another or back. This application introduces the following:

- working with URL resources,
- giving functionality to the right-left scroll buttons, and
- manipulating in-project/out-project images.

"Simple Animation" Application

Objectives: This application introduces a basic animation application and building a user interface, which in this case displays a black screen with the touch pad functionality. Students are expected to learn the following from this unit:

- digital color concepts,
- visualizing geometric shapes and their representation in program logic,
- understanding and manipulating the position of elements on the screen using co-ordinates along "x" and "y" axes, and
- animating static geometric shapes to create patterns.

Code Walk Through: There are two critical methods in the *SimpleAnimation.java* class. In the *DrawView* method the figure shape and color is set by *paint.setStyle* and *paint.setColor* methods. In the *OnDraw* method, the background color is set to white using the *canvas.drewColo*r method. The second method also defines the drawing figure shape and size. The shape can be changed from a circle to a rectangle or other shape. Some code fragments from the *"Simple Animation"* application are shown in Sample Code 1.

Sample Code 1: Code Fragments from 'Simple Animation' Application

```
public DrawView(Context context) {
      super(context);
      setFocusable(true);
      setFocusableInTouchMode(true);
      this.setOnTouchListener(this);
      paint.setStyle(Style.FILL);
            // Change the 'Style.FILL' to 'Style.STROKE' to change
            the figure style.
      paint.setColor(Color.rgb(255, 0, 0));
            // Change the '255, 0, 0' to any integer numbers within
            the range 0-255 to change the figure color.
      paint.setAntiAlias(true);
  }
  @Override
  public void onDraw(Canvas canvas) {
            canvas.drawColor(Color.WHITE);
             // Change the background color "Color.WHITE" to
            "Color.GRAY" or any other color from the given list.
            int size=6;
            // Change the '6' to any integer number to change the
            figure size.
            for (Point point : points) {
            // Change the figure shape by uncommenting one of the
            lines above. Note: Once you uncomment any line with
            "canvas" comment the active line.

            // canvas.drawPoint(point.x, point.y, paint);

            canvas.drawCircle(point.x, point.y, size, paint);

            //canvas.drawRect(point.x, point.y, point.x+size,
            point.y+size, paint);
      }
  }
```

Code Manipulations: The code presented as a template in the Eclipse IDE contains highlighted sections of code that can be changed in succession to observe and grasp the overall structure of the mathematical logic and the program syntax that work to create shapes, colors, and effects that change position and direction based on user interaction (See Table 1).

Testing: Before deploying the application, it must be tested using a base common emulator device that mimics the functionality of an actual device on a user's computer. This emulator is part of the Android SDK and provides functionalities that are sure to be present in the majority of devices running the OS and/or platform for which the application is targeted.

Table 1: Code Manipulations Applied to 'Simple Animation' Application

Manipulations	Code
Change the animation figure style from 'FILL' to any other style	paint.setStyle(Style.FILL);
Change the figure color.(Change'255','0', and '0' in prentices with the new integer numbers within the range '0-255')	paint.setColor(Color.rgb(255, 0, 0));
Change the screen background color from 'WHITE' to any other color.	canvas.drawColor(Color.WHITE);
Change the figure size from '6' to any other size.	int size=0;

Change the animation figure shape to the new one such as rectangle, line, circle, or triangle shape. (Comment and uncomment the lines)	//canvas.drawPoint(point.x, point.y, paint);
	canvas.drawCircle(point.x, point.y, size, paint);

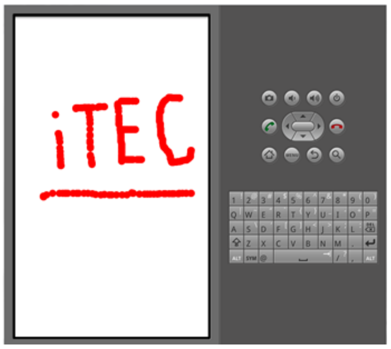
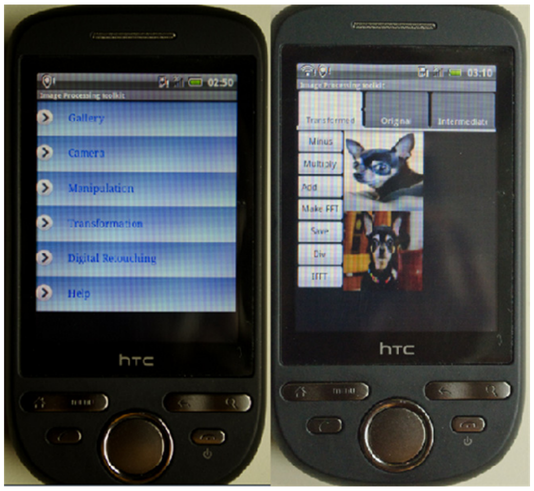


Figure 4: 'Simple Animation' Application shown in emulator.

After completing the desired customization as shown in Table 1 in the application template, it is ready to run on the emulator. We have the option to change the emulator device by selecting a new device from the project properties option. This enables us to observe the application's compliance with basic as well as advanced handsets. Figure 4 shows us the result of running the "Simple Animation" on a standard Android SDK capable emulator.

Deployment: There are two ways to deploy the executables on a mobile device. The first is via an ad-hoc connection between user computer and handset. This can be done via either a USB cable or a Bluetooth wireless connection, based on user device capabilities. Most Android devices allow a user to install applications via these connections. The second way is via the Internet by publishing the developed application in Web stores. This project, we accessed via the USB cable deployment option.



Menu

Gallery
• Select image from gallery

Camera
• Launch camera
• Select image from SD card

Manipulations
• Crop image
• Skew image
• Transpose
• Rotate
• Scaling
• Add images
• Subtract images
• Clone image to image2

Transformations
• FFT
• FFT domain processing
• DCT

Digital Retouching
• Adjust brightness
• Invert image
• Histogram
• Salt and pepper noise

Gaussian noise
• Average filter
• Median filter
• Min-Max filter
• Laplace filter
• Sobel filter
• Gaussian smoothing
• Hough transform

Figure 5: (Left) Menu structure of used library on Android phone; (Right). Sample demonstration of FFT domain processing with two images

Image Processing Template Apps

The learning flow starts with a simplified "design" (template manipulations) followed by "coding," "pre-verification," "deployment" and "testing." The toolkit abstracts many of these steps to make it easier for students to handle the process.

This section demonstrates how template apps can be clustered in libraries to address specific application areas. The EE students deal with signal and image processing algorithms through related templates and hands-on exercises to study the effects of the algorithms on images captured by a handset camera or downloaded from the Internet. The image processing algorithms are collected in a template library and partitioned as 1) manipulations; 2) transformations; and 3) digital retouching (see Figure 5). A detailed description of this module has been reported by the authors in Golagani, Esfahanian, Akopian, and Saygin (2012) and is reviewed here. The "manipulations" category includes simple mathematical operations such as add/subtract two images, scale, crop, clone, transpose, skew, and rotate an image.

The "transformations" contains algorithms that convert images in the spatial domain to the frequency domain and inversely from the frequency to the spatial domain (discrete cosine transform and fast Fourier transform). "Digital retouching" consists of a set of image evaluation operations such as histograms, noise removal, and edge detection filters. Some sample real time images are stored in the gallery; otherwise, the user can take pictures using the built-in camera. The images on the handset screen are placed under different tab controllers, namely "transformed," "original," and "intermediate," to compare processed and original images.

Sample applications are "median," "averaging/mean," and filters. The filters are a sliding window functions (i.e., each pixel of the image is replaced by a value), which is computed using a window of neighborhood samples.

Sample code 2a and 2b depict operation of the median and mean filters in the window. The median filter effectively removes impulsive salt-and-pepper noises while mean filters are useful for filtering Gaussian noises. Figure 6 illustrates denoising examples by these filters on handset screens. By default, a 3x3 kernel is defined, which can be changed by students. In case a filter performs a linear operation, (i.e., samples in the window multiply to kernel weights), then various kernel selections may result in low-pass, high-pass, and edge detection filters. Examples of template manipulations for implementing edge detection filters using the "averaging" display are shown in Table 2. Figure 7 illustrates edge detection using the Laplace filter, and Figure 8 demonstrates two-dimensional convolution in the frequency domain.

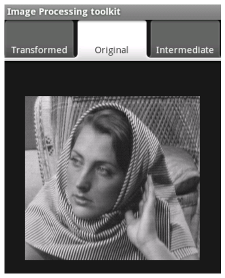
Sample Code 2: (a) Median Filter Averaging (Mean) Filter, (b) Mean Filter

(a)

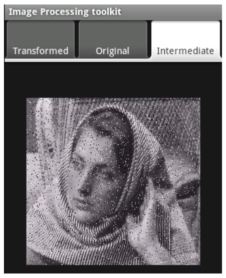
```
int  center =4; //center  index, index starts from'0'
int[] window  = new  int[2*center+1];
//vector to hold pixels from 3x3 neighborhood window
for(int y=1;y<=height-2;y++){
 for (int x=1 ;x<=width-2;x++){
 // fill the window with pixel values
     int k=0;
     for (int j= -1;j< =1; j++){
       for (int i= -1;i< =1; i++){
         window[k]= inputimg.getPixel(x+i , y+j);
         k++;
       }
}
  Arrays.sort(window); // sort in ranking order   modi-
fiedpixels[x][y]=window[center];
 //replace pixel with median
  }
 }
```

(b)

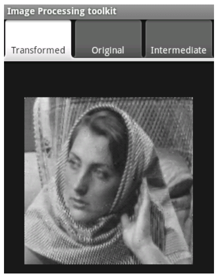
```
int  kernelsize = 3 ;  int  total_kernelelements= 9;
int  window[][] = new int[kernelsize][kernelsize];
int[][] kernel={{1,1,1},{1,1,1},{1,1,1}};
for (int  y=1;y < height-2;y++) { //slide in 'y'dimension
   for (int  x=1;x < width-2;x++){ //slide in 'x'dimension
  //get input pixel values to kernel
int  r=0,g=0,b=0;
for (int  i=0;i<kernelsize;i++){
   for (int  j=0;j<kernelsize;j++){
//get R,G,B values from input image and add all elements
      r= (r+ kernel [i][j]*RgbVal.getR(window[i][j]));
      g= (g+ kernel [i][j]*RgbVal.getG(window[i][j]));
      b= (b+ kernel [i][j]*RgbVal.getB(window[i][j]));
  } }
// compute average
 r= r/total_kernelelements;  b=b/total_kernelelements;
 g=g/total_kernelelements;
 int  mean  = RgbVal.toRgb((byte) r,(byte) g, (byte) b);
```



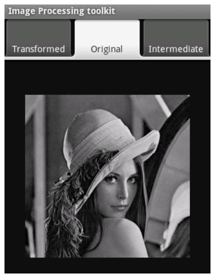
(a)



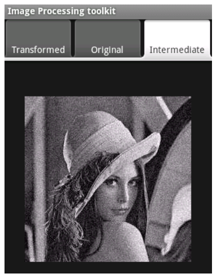
(b)



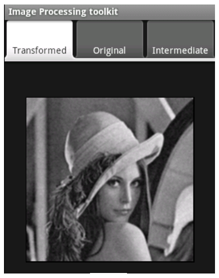
(c)



(d)



(e)

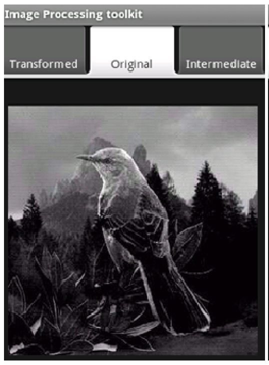


(f)

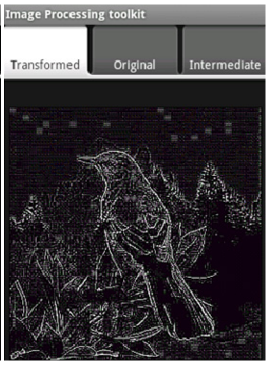
Figure 6: (a) Original 'Barbara' image; (b) 'Barbara' image corrupted by salt & pepper noise; (c) Corrupted "Barbara" image restored by a median filter; (d) Original 'Lena' image; (e) 'Lena' image corrupted by Gaussian noise; (f) Corrupted "Lena" image restored by average or mean filter

Table 2: Code Manipulations Applied to Edge Detection Filters

Manipulations	Code
Apply Sobel1 kernel. This makes edge detection in vertical direction and smoothing in horizontal direction	**int[][]** kernel = {{1,0,-1},{2,0,-2},{1,0,-1}};
Apply Sobel2 kernel. This makes edge detection in horizontal direction and smoothing in vertical direction.	**int[][]** kernel = {{1,2,1},{0,0,0},{-1, -2,-1}};
Apply 3x3 first Laplace kernel.	**int[][]** kernel = {{1,1,1},{1,-8,1},{1, 1,1}};
Apply 3x3 second Laplace kernel	**int[][]** kernel = {{0,-1,0},{-1,4,-1},{0,-1,0}};

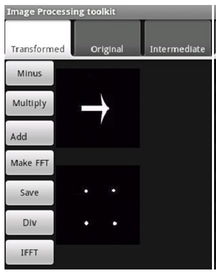


(a)

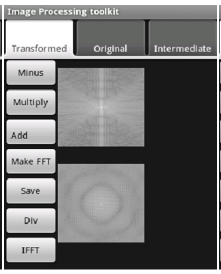


(b)

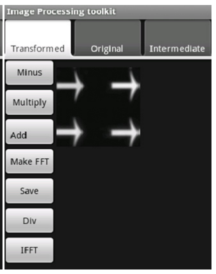
Figure 7: (a) Original image; (b) Laplace filtered image for edge



(a)



(b)



(c)

Figure 8: (a) Original two images; (b) FFT applied; (c) convolution applied and its IFFT

Quiz Application

One can also develop apps that support learning assessment. An example is implemented to support student self-assessment. Once students complete their projects with templates, they can assess their knowledge by launching the quiz application. This page launches with five random questions queried from the SQLite (http://www.vogella.com/articles/AndroidSQLite/article.html) database. Since Android provides full support for the SQLite database, the database created will be accessible by name to any class within the application. SQLite is readily available on every Android device. Once SQL statements for creating and updating the database are defined, then the database is automatically managed by the Android platform. Each question appears on a single page (Figure 9). Click "next" to go to the next question. Finally, the "submit" button appears, which gives a pop up message showing the score of the attempted quiz. At this stage, the quiz application is used for self-assessment only.

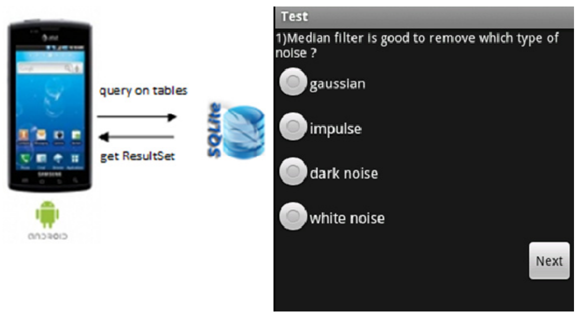


Figure 9: Sample screen of Quiz in this toolkit

Learning Module Assessment

A short course workshop module (Table 3) has been integrated in the wireless communications course, which is offered by the Electrical and Computer Engineering Department at the University of Texas at San Antonio (UTSA) during Spring 2012. About thirty-five students participated in the workshop for a total of 8 days for 1 hour 15 minutes per day. The effort is part of a broader educational initiative of the Interactive Technology Experience Center (iTEC) at the UTSA on implementing technology enhanced learning for undergraduates and outreach (http://itec.utsa.edu/).

The efficiency of the template-based learning is assessed through the follow-

Table 3: Short Course Program

Introduction to the Android Application Development (Class duration: 1h15min)	
Class 1	**Introduction.** History. Operation Systems (OS). Mobile phone development market
Class 2	**Integrated Development Environments (IDE).** NetBeans IDE and Eclipse IDE for Java based application development The students learned: ▪ Find and install the IDEs ▪ Create first project in Eclipse IDE
Class 3	**Introduction to Java.** Basics of Java programming language and examples
Class 4	**Android Platform.** Introduction to Eclipse IDE/Android SDK development environment and to the basics of android application development. The students learned: ▪ Find and install the Android SDK. ▪ Integrate Android SDK with Eclipse IDE. ▪ Create first android emulator. ▪ Create first android project. ▪ "Hello World!!!" - first Android application. ▪ Run and test the application on the emulator.
Class 5	**Template-Based Applications.** Learn by manipulating templates. The students learned: ▪ Application 1. ('Hello World' application) ▪ Application 2 ('OnClick Example' application) ▪ Application 3 ('Image Gallery' application) ▪ Application 4 ('Slideshow' application) ▪ Application 5 ('Simple Animation' application) ▪ How manipulate application templates ▪ How install and test applications on handsets
Class 6	**Android GUI Development.** Basics of the android GUI development. The students learned: ▪ The basics of Android GUI development. Integrate different GUI components (such as buttons and checkboxes) to the application. *("Temperature Converter" application was used as an example)* ▪ Install the device drivers and connect the mobile device to the Eclipse IDE/Android SDK
Class 7	**Advanced Android Application Development.** Examples of Image processing applications: median and averaging filters, edge detection, convolution in frequency domain.
Class 8	**Project Presentation:** In this class, Instructor formulated project assignments and requirements. Directions and hints are provided to students. The workshop was concluded with the Question/Answer session.

ing criteria: (a) change of conventional attitudes; (b) gaining confidence; and (c) ability to successfully complete assignments in the focus area and develop their first apps independently. The preliminary results of this assessment were originally discussed in Golagani et al. (2012), but more detail and discussion regarding the entire learning module are presented here.

To assess course effects on student attitudes, both pre- and post-workshop surveys are conducted.

Pre-workshop survey questions and data are presented in Table 4. Similar to other studies (Lang et al., 2007), the responses are scaled using answer options: the scale goes from 5-excellent to 1-poor for the first question, the scale goes from 5-high to 1-low for the second question, the scale goes from 5-hard to 1-easy for the third question, and the scale goes from 5-yes to 1-no for the last question.

Most of the students reported low or average programming experience with only 5% highly assessing their skills (mean = 2.39). Initial perception of students on the difficulty of mobile programming is average with mean = 2.71. Most of the students think that careers in mobile programming are rewarding (mean = 3.9), about 60% positive.

Table 4: Students' Pre- Workshop Feedbacks

Questions	Mean	SD	Percentage of 5 and 4 selections
How do you grade your programming skills? {Excellent 5 – 1 Bad}	2.39	0.9	5%
How do you grade your interest in Mobile Application Development? {High 5 – 1 Low}	4.13	1.1	83%
At this point, how do you perceive the difficulty of mobile phone programming? {Hard 5 – 1 Easy}	2.71	1.05	22%
Do you think that the mobile programming is a rewarding career but you have doubts on possible difficulties when pursuing it? {Yes 5 – 1 No}	3.9	1.06	60%
How do you grade your interest in having more Android Application Development experience? {High 5 – 1 Low}	4.27	0.98	85%
Were homework 1 applications useful as initial learning experience? {Yes 5 – 1 No}	4.21	0.72	83%
Did homework 2 ("Temperature Convertor" Application) help to advance your Android exposure? {Yes 5 – 1 No}	4.48	0.72	94%
Modifying known applications as in the last project (image processing) might help to overcome programming skill limitations for beginners. {Agree 5 – 1 Don't agree}	3.79	1.16	62%
Were Android Application Development tutorials helpful? {Yes 5 – 1 No}	4.49	0.81	92%
Were PowerPoint presentations helpful? {Yes 5 – 1 No}	4.31	0.91	89%

Table 5 shows post-workshop survey results. The majority of students, approximately 76% (Mean = 3.93), changed their opinion on the difficulty of mobile programming and think that they can develop Android applications if needed (69%, mean = 3.89). Ninety percent of students (mean = 4.41) understood the application development process. Eighty-two percent (mean = 4.03) would consider the usage of mobile phones in their senior design projects. End of semester monitoring confirmed that 12 students eventually did so. Overall, students were satisfied, as their expectations were met (87%, mean = 4.34). This demonstrates the efficiency of the approach in terms of "change of attitudes" and "gaining confidence."

There were three homework assignments on application development: (1) replicate the design of a set of template applications; (2) design a "currency conversion" application using template "temperature conversion" application; and (3) investigate the "image processing apps" toolkit and implement filters with various kernels. All homework assignments were assessed positively by students (83%, 94%, and 62% for assignments 1, 2, and 3, respectively), which inferred that students had good experiences with the template-based assignments.

Table 5: Students' Post- Workshop Feedbacks

Questions	Mean	SD	Percentage of 5 and 4 selections
Has this workshop changed your perception on the difficulty of mobile phone programming? {Yes 5 – 1 No}	3.93	1.31	76%
Do you think that you would be able to develop Android applications if needed? {Yes 5 – 1 No}	3.89	1.21	69%
Has this workshop clarified the phone application development and integration process? {Yes 5 – 1 No}	4.41	0.89	90%
Might this workshop influence your opinion on a possible use of Android Application in your senior design project? {Likely 5 – 1 Unlikely}	4.03	1.13	82%
Are your expectations from the workshop met? {Yes 5 – 1 No}	4.34	0.79	87%
How do you grade your interest in having more Android Application Development experience? {High 5 – 1 Low}	4.27	0.98	85%
Were homework 1 applications useful as initial learning experience? {Yes 5 – 1 No}	4.21	0.72	83%
Did homework 2 ("Temperature Convertor" Application) help to advance your Android exposure? {Yes 5 – 1 No}	4.48	0.72	94%
Modifying known applications as in the last project (image processing) might help to overcome programming skill limitations for beginners. {Agree 5 – 1 Don't agree}	3.79	1.16	62%
Were Android Application Development tutorials helpful? {Yes 5 – 1 No}	4.49	0.81	92%
Were PowerPoint presentations helpful? {Yes 5 – 1 No}	4.31	0.91	89%

A general assessment survey is summarized in Table 6. Students found the mobile application development workshop to be excellent (92%, mean = 4.34), interesting (95%, mean = 4.48), relatively easy (67%, mean = 3.72), useful (93%, mean = 4.41), valuable (91%, mean = 4.31), motivational (63%, mean = 3.34), and balanced in effort (54%, mean = 3.27). Presentations and tutorials were helpful for 89% and 92 % of students, respectively. Eighty-five percent (mean = 4.27) of students were motivated to learn more about mobile application development.

Table 6: Students' Feedbacks about the Workshop

How do you grade Mobile Application Development workshop in general?					Mean	SD	Percentage of 5 and 4 selections
Excellent	5	-	1	Bad	4.34	0.92	92%
Interesting	5	-	1	Boring	4.48	0.89	95%
Easy	5	-	1	Hard	3.72	0.91	67%
Useful	5	-	1	Useless	4.41	0.85	93%
Valuable	5	-	1	Worthless	4.31	0.91	91%
Motivational	5	-	1	Dry	3.34	1.15	63%
Effortless	5	-	1	Labor-intensive	3.27	0.82	54%

Lutes (2012) reported many challenges when teaching mobile programming even for dedicated semester-long course settings. For the short course described in this paper the students learned essential concepts of mobile programming as evidenced by the results. Ninety-four percent of students had grades B and higher for assignment 1. Eighty-three percent of students had grades B and higher for designing their first independently developed "currency converter" application in assignment 2. This addresses the third efficiency criterion "ability to develop apps independently." Similarly, 83% of students had grades C and higher for completing the very challenging "image processing app" homework assignment 3. All three homework grades demonstrated efficiency in terms of the three criteria defined above. In other words, template-based education appears to be a very efficient learning approach for mobile application programming.

Conclusion

The paper summarized the study on effectiveness of template-based undergraduate learning for mobile application development in electrical engineering departments, which may not provide conventional course tracks for comprehensive learning on the subject due to curricula hours' constraints. Short modules are integrated in a conventional course and basic exposure to mobile computing was provided without specific prerequisites. Survey results demonstrate high efficiency of the approach. It motivates students to elaborate similar topics further and consider Senior Design topics based on mobile apps. Moreover, templates served as hands-on exercise for students to work on topics covered in their conventional signal and image processing courses by capturing and manipulating images from phone cameras.

Acknowledgment

This work was partly supported by iTEC-UTSA Learning Center and NSF grants 0942852 and 0932339.

References

Al-Imamy, S., Alizadeh, J., & Nour, M. N. (2006). On the development of a programming teaching tool: the effect of teaching by templates on the learning process. *Journal of Information Technology Education, 5*, 271-283. Retrieved January 20, 2013, from http://www.jite.org/documents/Vol5/v5p271-283Al-Imamy115.pdf

Caverly,C. D., Ward, R. A., & Cavarly, A. (2009). Techtalk: Mobile learning and access. *Journal of Developmental Education, 33*(1), 32-39.

Geddes, S. J. (2004). Mobile learning in the 21st century: Benefits for learners. *The Knowledge Tree e-journal, Edition 6*, Peer Reviewed Articles 3.

Golagani, S. C., Esfahanian, M., Akopian, D., & Saygin, C. (2012). *Template-based image processing toolkit for Android phones*. 119th ASEE Annual Conf. & Exposition, San Antonio, TX, AC2012-3546.

Hayes, P., Joyce, D., & Pathak, P. (2004). Ubiquitous learning – An application of mobile technology in education. Cantonini & McLaughlin (Eds.), *Proceedings of ED-MEDIA*. Lugano, Switzerland.

Jonassen, D. H. (2006). *Modeling with technology: Mindtools for conceptual change.* Upper Saddle River, NJ: Merrill Prentice Hall.

Laine, T. H., Sedano, C. A. I., Joy, M., & Sutinen, E. (2010). Critical factors for technology integration in game-based pervasive learning spaces. *IEEE Trans. on Learning Technologies, 3*(4), 294-306.

Lang, D., Mengelkamp, C., Jäger, R. S., Geoffroy, D., Billaud, M., & Zimmer, T. (2007). Pedagogical evaluation of remote laboratories in eMerge project. *European Journal of Engineering Education, 32*(1), 57-72.

Lutes, K. (2012). Cross-platform mobile app software development in the curriculum. *Journal of Issues in Informing Science and Information Technology (IISIT), 9*, 115-124. Retrieved January 20, 2013, from http://iisit.org/Vol9/IISITv9p115-124Lutes120.pdf

Lytras, M. D., Gasevic, D., & Ordonez De Pablos, P. (2008). *Technology enhanced learning*. IGI Publishing.

Portio Research. (2012). *The next billion: Strategies for driving growth and making profits in low ARPU mobile markets*. Retrieved January 20, 2013, from http://www.portioresearch.com/en/reports/archive

Rosberg, M. (2001). *E-learning: Strategies for delivering knowledge in the digital age*. McGraw-Hill.

Sakamura, K., & Koshizuka, N. (2005). *Ubiquitous computing technologies for ubiquitous learning*. IEEE Int. Workshop on Wireless and Mobile Technologies in Education, pp. 11-20.

Schank, P. K., Linn, M. C., & Clancy, M. J. (1993). Supporting Pascal programming with an on-line template library and case studies. *International Journal on Man-Machine Studies, 38*(6), 1031-1048.

Shudong, W., & Higgins, M. (2006). Limitations of mobile phone learning. *The JALT CALL Journal, 2*(1), 3-14.

Tam, J. M., & Chen, K. (2006). Mobile technology as a learning object and exploration tool in an IS curriculum: An innovative instruction of wireless network security. *IEEE Trans. on Education, 49*(2), 193-198.

Thomton, P., & Houser, C. (2004). *Using mobile phones in education.* 2nd IEEE International Workshop on Wireless and Mobile Technologies in Education (WMTE'04), pp. 3-10.

Ullrich, C., Shen, R., Tong, R., & Tan, X. (2010). A mobile live video learning system for large-scale learning – System design and evaluation. *IEEE Trans. on Learning Technologies, 3*(1), 6-11.

Vygotsky, L. (1978). Interaction between learning and development. In M. Gauvain & M. Cole (Eds.), *Readings on the development of children.* New York: W. H. Freeman and Company.

Yuen, T., & Liu, M. (2011). A cognitive model of how interactive multimedia authoring facilitates conceptual understanding of object-oriented programming in novices. *Journal of Interactive Learning Research, 22*, 329-356.

Zickuhr, K. (2012). *Generations and their gadgets*. Pew Research Center. Retrieved January 20, 2013, from http://pewinternet.org/Reports/2011/Generations-and-gadgets

Biographies

Dr. David Akopian is an Associate Professor at the University of Texas at San Antonio (UTSA). Prior to joining UTSA he was a Specialist with Nokia Corporation from 1999 to 2003. From 1993 to 1999 he was a member of teaching and research staff of Tampere University of Technology, Finland, where he also received the Ph.D. degree in electrical engineering. Dr. Akopian's current research interests include communication and navigation systems, and communication technologies for education and healthcare applications. He authored and co-authored more than 30 patents and 120 publications. He is a Co-Chair of SPIE Multimedia on Mobile Devices conference.

Arsen Melkonyan is a PhD student at the University of Texas at San Antonio. He received the B.Sc. degree in Electrical Engineering from the State Technical University of Armenia, Yerevan, Armenia, in 2003 and M.Sc. degree in Electrical Engineering from the University of Texas at San Antonio in 2008. His current research interests include Impact localization algorithms for Structural Health monitoring, WLAN Indoor positioning algorithms, and the design aspects of remote hardware-based educational system.

Santosh C. Golagani is received her Master's degree from University of Texas at San Antonio (UTSA) in computer engineering department. She worked in HCL Technologies as a Member Technical Staff from 2009-2010.Prior to working for HCL Technologies, she worked in Cognizant Technologies Solutions as a Programmer Analyst from 2006-2008. She pursued her Bachelor's degree in electronics and communication engineering from Jawaharlal Nehru Technological University in 2006.

Dr. Timothy T. Yuen is an Assistant Professor of Instructional Technology in the College of Education and Human Development at the University of Texas, San Antonio. His research examines tools that mediate conceptual understanding in computer science and engineering and motivate students to pursue careers in STEM fields. He is a member of IEEE, ACM, AERA, and ISTE and an affiliated faculty member of the Interactive Technology and Experience Center (iTEC) at UTSA.

Dr. Can Saygin is a professor of mechanical engineering and director of the Interactive Technology Experience Center (iTEC) at the University of Texas at San Antonio. His research interests include manufacturing systems, modeling and analysis of automated manufacturing systems, and production planning and control. He has published over 100 peer-reviewed journal and conference papers. He is the recipient of the College of Engineering 2009 Excellence in Teaching Award, the President’s 2011 Distinguished Achievement Award for Teaching Excellence, and the University of Texas System Regents’ Outstanding Teaching Award in 2012. He is a member of SME, IIE, ASME, and ASEE.