Educating for Mobile Computing: Addressing the New Challenges

Barry Burd
bburd@drew.edu
Drew University
Madison, NJ, USA

João Paulo Barros
joao.barros@ipbeja.pt
Instituto Politécnico de Beja
Beja, Portugal

Chris Johnson
johnch@uwec.edu
University of Wisconsin, Eau Claire
Eau Claire, WI, USA

Stan Kurkovsky
kurkovskysta@ccsu.edu
Central Connecticut State University
New Britain, CT, USA

Arnold Rosenbloom
arnold@cs.toronto.edu
University of Toronto at Mississauga
Mississauga, ON, Canada

Nikolai Tillman
nikolait@microsoft.com
Microsoft Research
Redmond, WA, USA

ABSTRACT

Computers that once filled rooms now fit in our pockets, and unlike their predecessors, mobile computers abound. The mobile industry is surging, with more smartphones being sold to consumers than PCs [17]. But does the rise of mobility impact computer science education? We claim that computer science educators must seriously consider mobility as they examine their curriculum. In this working group report, we offer a brief defense of why mobile computing belongs in our courses, summarize our survey of several hundred courses which already incorporate it, and discuss how educators might adopt it in their own courses. We hope that this work will help computer science educators make informed decisions about incorporating mobile computing into their courses and provide examples of such integration on different levels, ranging from individual projects or lecture topics to mobile computing as a learning context for an entire course.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer science education

Keywords

mobile computing, mobile curriculum

1. INTRODUCTION

We may be tempted to regard mobile computing as just another stage in the evolution of hardware, with the foundational core of computer science unchanged. In some ways, the rise of mobility can be likened to the microcomputer revolution, which put affordable and general personal computers in businesses and homes alike. Certainly, mobile devices and personal computers have a similar democratizing effect: computers are generally available to all ages, races, and income classes. What sets mobile computing apart, however, is the extent of its reach.

Consider mobility's impact on our incoming students. In 2010, personal computers were in just 80% of homes in the US [50]. In contrast, the number of mobile phones registered in the US in 2011 was 104.6% of the country's population. In 2010, 96% of 18- to 29-year-olds owned cell phones [43]. Over 50% of cell phones are smartphones, devices that support advanced user interaction and general purpose computation [20].

Powerful and fully-programmable computers are in the hands of most of our future students. Furthermore, these computers play a significant role in their lives. Mobile devices may be computers, but we view them as much more: they are our connection to family and friends, they store our experiences, and they deliver to us entertainment and information wherever we need it. It's no wonder that one in five individuals claim they would rather spend a week shoeless than a week without their mobile phone [44]. Given the ubiquity of these devices and our social attachment to them, computer science educators have an unprecedented opportunity to leverage student interest in computing.

Amongst groups long underrepresented in computer science, mobility is having an especially marked effect. At the end of 2011, the same year the global population reached seven billion people, there were six billion connected mobile devices, most of them in developing nations. 75% of Hispanics and blacks in the US own mobile devices [35]. The number of women owning smart phones is at parity with the number of men [20]. Mobility is proving to be an egalitarian force, and its thoughtful incorporation into our curriculum may restore diversity in our enrollments.

Mobility's potential in recruitment is strong. But we, as educators, may fear buying into the sensationalism surrounding mobile computing. We may even categorize it as many categorize game development: we'll teach it to attract students, but we do not believe it will have a long-term impact on our graduates' future careers. However, technology companies are aggressively bolstering their mobile services,

ITiCSE-WGR'12, July 3-5, 2012, Haifa, Israel.

with businesses generating 466,000 new jobs in mobile application development since 2007 [38]. In just five years, the mobile technology industry now employs four times as many people as the game industry [10]. Mobility has infiltrated our career fairs and is a major focus of recruiters [28].

As educators, we cannot discount the impact that mobility is having on our students before and after our time with them. No matter the quality of education we may offer, we risk being labeled irrelevant by both prospective students and industry if we do not consider this new form of computing in our curriculum. We're better off heralding it. Its success only reinforces our claim that computation plays an intimate role in our way of life.

In this report we offer an analysis of the mobile computing landscape, addressing computer science educators at all levels. In section 2, we define the distinguishing features of mobile computing that set it apart from traditional computing. In section 3, we describe how educators are adopting mobile computing into their curricula, based on our survey of several hundred existing courses from across the globe. Finally, in section 4, we offer educators who are looking at teaching mobile computing an overview of several viable course structures and mobile computing environments.

2. ESSENCE OF MOBILE COMPUTING

The numbers above suggest that mobile computing is important to consider, but what does an educator teach in a course or unit on mobility? Are the changes we must make to include mobility in our curricula only superficial? To answer these questions, we must first pin down what we mean when we speak of mobile computing.

While it is true that mobility builds on the foundational elements of computer science that we already teach, mobility also draws together the far reaches of our discipline, combining elements of human-computer interaction (HCI), systems, algorithms, and software engineering under one common theme. Such a diversity of topics is typically addressed only in project-based capstone courses.

Based on our survey of several hundred existing courses on mobile computing, we describe what we see as the salient features of mobility that should shape our courses:

Mobility and pervasiveness.

Users carry mobile devices everywhere: at bus stops, waiting in lines, while getting their hair cut, on outdoor adventures, in meetings, and so on. Devices tend to be owned by individuals and software closely integrates with the dynamics of the owner's changing location and activities, often exposing events in the owner's life to social networks. Personal area networks support social gaming, secure financial transactions, and interaction with smart appliances. Cloud-backed data and web services make computing portable.

User-interface and event-driven programming.

Commodity mobile devices are designed more for interaction with the user and less for general purpose computation. Accordingly, compared to software typically developed in traditional computer science courses, mobile apps rely very heavily on graphical user interfaces (GUIs). Most modern GUI libraries are event-driven; the operating system asynchronously awaits user input and triggers the developer's event-handling code. Developers must approach apps with design in mind and accept the external locus of control.

Interruptions.

A user playing a game gets an incoming call. A driver navigating a route passes through a long tunnel in which cell connections are blocked. Modern mobile operating systems more heavily orchestrate an app's runtime lifecycle than do their desktop counterparts. Mobile developers must consider service interruptions and the fractured usage of their apps. To account for interruptions, they must retain data until focus and service is restored.

Sensor-based input.

Desktop computing relies primarily on a keyboard and mouse for input, while mobile devices typically integrate many more kinds of input hardware: touch, multitouch, accelerometers, GPS, gyroscopes, magnetometers, proximity sensors, front- and back-facing cameras, IR sensors, microphones, and so on. This wide variety of sensors allows apps to better incorporate the physical world. Interface designers must carefully consider how to support the diverse set of input devices used to interact with their apps.

Finite resources.

Mobile devices tend to have reduced memory, storage capacity, and execution speed compared to desktop computers. Battery life is limited. The benefits of consuming resources conservatively, employing low-complexity algorithms, keeping background services lightweight, and limiting use of sensors and intensive graphics are much more apparent on mobile devices than in desktop computing.

Proximity to users.

Developers have straightforward access to a massive and diverse user base through vendor and third-party application distribution channels. Users rely on mobile devices to record sensitive personal information, including email and text messages, photographs, location, calendars, and contacts. It is easy for a developer to release software that accesses and augments a user's digital identity, therefore, the developer must consider significant ethical issues, like how privacy is managed and how the integrity of personal data is preserved. The lowered distribution barrier also allows users to easily communicate feedback to the developer.

Mobility is a natural arena for the coherent study of computer science concepts, including traditional computer science concepts and newly-emerging issues.

3. SURVEY OF MOBILE COURSES

To help us understand the impact of mobile computing on computer science education, we surveyed nearly 200 courses that covered some aspect of mobile computing. The courses were identified through mailing list solicitations, literature reviews, and web searches. The interested reader can find the complete list of our resources at [14]. Though we accessed only courses with a publicly available syllabus or description, we found that mobile computing has been incorporated into computer science programs in various ways, from a course strictly devoted to mobile computing, to a unit in a course, to a context for motivating and engaging students.

Below we discuss some of the typical ways mobile computing is incorporated into computer science learning environments.

3.1 As a New Standalone Course

Mobile computing can appear as a course in its own right, for example, New York University's v22.0480, iPhone Programming [46], and Boston University Metropolitation College's MET683, Mobile Application Development [19]. Such courses typically cover the design, development, testing, deployment, and maintenance of mobile applications. In a course for computer science majors (or majors in a related field) students learn ways in which mobile application development differs from more traditional desktop development. In a hands-on course, students get practice using one or more of the existing mobile platforms.

Current mobile application development courses typically cover some combination of event-driven programming, XML-based layouts, model-view-controller processing, touch sensing, user experience issues specific to mobile devices, resource usage (constrained resources include screen size, memory, processing power, network connectivity), performance issues, sensor programming, location-based services, 2-D and 3-D graphics, animation and media, security, persisting data, testing and deploying a mobile app, and platform fragmentation (including the support of differing hardware profiles).

Below we further detail the topics that tend to appear in upper-division, single-term courses that focus on mobile application development.

User Interfaces.

Many of the mobile courses surveyed have a section devoted to user interface design. Most courses include an introduction to human-computer interaction and quickly move on to the principles of user interface layout and enabling interfaces to adapt to the screen resolution of the particular device they are run on. Other topics addressed include a discussion of interfaces that adapt to screen orientation and differing hardware profiles. The iOS User Interface Guidelines [7] and the Android User Interface Guidelines [30] serve as useful resources for students to investigate.

Part of the actual implementation of a custom user interface typically involves writing event driven code connecting user interface events with custom developer code, so the event driven programming model is typically discussed at this point. Depending on the platform used in the course, a discussion of the model-view-controller design pattern may be relevant as well.

All of the major platforms have 2-D and 3-D graphics libraries supporting custom drawing. Many courses complement this with a discussion of color representation, pixels, graphics contexts, and drawing.

Device I/O.

A key aspect of mobile development is the deep interaction users have with their devices through a variety of sensors and communication features. Many of the platforms use events to signal input from touch, orientation, accelerometer, compass, GPS, audio, and camera sensors, so most courses cover callbacks and event-driven interaction in considerable detail. Some courses include lab exercises or assignments on output through text messages (SMS), phone calls, or Bluetooth devices.

Though all major platforms allow developers to emulate real devices, students who do not have access to physical devices may not be able to investigate certain hardware features, like multitouch, GPS, and telephony. To work around this problem, some instructors maintain a pool of devices for students to use.

Networking.

Mobile devices often are connected in some way to the Internet, and mobile software regularly takes advantage of this constant network connectivity. Networked applications interact with information stored on the web and exchange data with other computers on the Internet. For example, mobile games store and retrieve high scores from a central server. The server may also be used to facilitate multi-player games, synchronizing game state for players across the globe. An application might include an interactive map or continually share location information with other users running the same application. Courses capitalize on these powerful scenarios by including a discussion on networking (client-server architectures, protocols, and state machines). In addition to general networking concepts, some instructors describe cloud-based services and the graceful handling of breaks in connectivity through caching.

Interruptions/lifecycle.

Mobile applications are typically executed, paused and later resumed, as a result, the lifecycle of a mobile application is more involved than that of a typical desktop application. Topics in this section include application lifecycle states, event notification, saving application state and resuming execution.

Fragmentation.

At present, the iOS mobile platform represents a collection of more than six different devices and three different OS versions currently in wide use. Similarly, the Android platform is run on numerous phones and tablets, with numerous versions of the operating system in current use [40]. No matter the platform, a mobile application developer will have to deal with their application running on different devices with different device capabilities under different operating system versions. To prepare students, instructors may introduce asset management systems and methods of abstracting a handful of version-dependent implementations of an interface.

Resource-constrained Computation.

Compared to their desktop counterparts, mobile applications run under a relatively constrained environment. Mobile applications run with relatively limited memory, processor speed and storage. Users of mobile applications also have significantly different expectations for the responsiveness of their applications and the impact the execution of an application will have on the device as a whole. Users expect mobile applications to respond immediately (or at least consistently) to all user interactions. They expect that a mobile application will not impact the execution of other applications. They expect that applications will not crash and that applications will not unnecessarily use the limited space available on the device. Instructors may find it useful to revisit discussions of time and space complexity from

earlier courses. Lessons on optimizing may include an introduction to profiling.

Security.

Mobile applications run in an environment containing personal information about their users, whether it is contact and calendar information, or location and call history data. While this opens up a world of possible interesting applications, it also puts users at risk. This section of the course discusses the principle of least privilege as well as the sandboxing and permissions model supplied by the various platforms. Additional reading includes articles on recent mobile malware [9] as well as iOS and Android security guides [8, 31].

Deployment.

One unit in a typical course covers mobile app stores. Topics include packaging, signing, publishing, advertising, and potentially profiting from a mobile application.

3.2 As a Context in Other Courses

The importance of context in computer science education has long been recognized [26, 29]. Using a context allows educators to focus on a subject that is important and familiar to students while learning new concepts; it provides a motivational tool and offers a wider playing field for experimentation and engaging students in the educational process. Using mobile computing as a motivational learning context has a strong potential to improve student satisfaction [37] and success in introductory computer science courses and increase student motivation to stay in the major. In fact, data from an IBM study [21] suggests that current students might be much more interested in using their mobile devices than PCs, so the role of such a context becomes even more important.

In the section, we discuss separately mobile computing used as a context in lower- and upper-division courses.

3.2.1 In Existing Lower-division Courses

Mobile devices have been integrated into computer science curriculum in different ways. They have been used as a motivating tool, increasing student interest in computer science and other STEM disciplines [39]. They have been used to teach computer literacy [36] and they have been used in introductory programming [37] courses.

Many CS programs experience high enrollments in their first introductory course, but witness a precipitous drop in the number of students who stay in the program and continue on with subsequent courses [12]. While some students may be mismatched with the program, many others decide not to continue with studies in computer science because they feel that the course material lacks connection with the real world [13, 11]. Some students feel that the problems being solved in introductory courses are too contrived and lack any relevance to real life; that their future work will involve nothing but coding; and that there will be little room for creativity and personal growth in their future computer science careers. Additionally, a CS1 course is often the only CS course that students from other majors are required to take as part of their graduation requirements. Mobile computing contexts can present the richness of CS as a creative, relevant academic discipline to students early in their studies, and so, could help decrease the attrition rates for CS programs. It could also attract non CS students to the discipline, possibly encouraging them to consider a second major or a minor in CS.

We present two case studies which describe educators experience using mobile computing in first year/introductory computer science courses.

Example Course 1.

A course described by Thiemann and Becker [51] introduces first-year computer science students to mobile computing. The course begins with teaching students the fundamentals of object-oriented programming along with the basics of test-driven development. Once the students are proficient with designing their own basic classes and using the Eclipse IDE for designing GUIs, they are offered project templates that focus on developing user interfaces for a number of simple Android applications. This gentle introduction to the Android platform is followed by slightly more complex projects that shift the focus from mobile interfaces to writing Java code and running it on an emulator, as well on the actual Android devices. Learning activities further emphasize code abstraction, as well as design elements, which is supported by a simple framework for building interactive games for Android. These activities help students learn about callback methods, as well as access the device sensors, such as accelerometer. Towards the end of the course students learn about additional Android APIs, although the paper does not provide specific details.

Example Course 2.

Kurkovsky and Defoe [24] describe a set of eight learning modules designed for introductory computer science courses using Java. These modules use mobile game development as a learning context. Each learning module serves as an opportunity to introduce students to an advanced CS topic, such as algorithms, artificial intelligence, computer networking, computer security, computer systems, database management, human-computer interaction, or software engineering, as well as to practice a fundamental topic, such as arrays or inheritance. By demonstrating these and other non-programming and diverse aspects of the discipline to the students early, this approach may help dissolve a widely popular misconception that "CS is all about coding."

Each learning module consists of a small laboratory project suitable for CS1 and beyond. Each project focuses on a single mobile game, introduces one advanced topic, and reinforces one core concept. In addition to the project, deliverables for each module include a set of instructional and supplementary materials. These include an instructor's manual that provides a sample syllabus to go with each project and guidelines for the adoption and adaptation of the curricular material. Instructional materials consist of lecture notes, illustrations, references, and sample solutions, as well as supporting materials such as project source code and documentation.

Evaluation results indicate that students in the test groups were more engaged with the course in which the mobile game development modules were presented and their motivation did not decline as much as the control groups.

3.2.2 In Existing Upper-division Courses

Below, we discuss upper-division courses which address mobile computing issues. Examples include the appearance

of mobile computing in courses on human computer interaction, game programming, and networking.

Human Computer Interaction.

The ACM Computing Curriculum recognizes the role of mobile computing in today's world as well as in the computer science discipline. According to section 4.2 of the ACM Computer Science Curriculum 2008 [25], building interfaces for mobile systems is among many areas of practical capabilities in which prospective graduates should have experience.

Mobile devices, by their very nature, offer a more interactive experience. They are worn, held, touched, moved and viewed by users in many ways, for many purposes, and in many contexts. They offer a wonderful platform, motivating students of the importance of the human-computer interaction concern. In fact, most students quickly acknowledge the fact that a good user interface experience is of the utmost importance for mobile applications.

The 2010 offering of CS160, Introduction to Human Computer Interaction [48] at the University of California, Berkeley, had a special focus on mobile applications. The course focus is on user-centered design. Students were expected to prototype, evaluate, and design a user interface particularly targeting a mobile device. Topics covered by the course included general HCI principles like the design cycle, evaluating the effectiveness of a design, affordances, conceptual models, and UI design patterns. Additionally, the standard HCI curriculum was supplemented with mobile topics such as Android UI design patterns [27] and an investigation of touch as a means for human, computer interaction [16] (multi-finger vs multi-hand vs multi-user, stylus, direction of approach etc.).

Games and Mobile Computing.

Current research indicates that mobile devices and mobile games are a valuable teaching and learning tool [18] and as a powerful tool used in augmented learning [32]. Mobile game development also gives students the benefit of instant gratification. Students can build graphical games on their own mobile devices and play the games with their friends.

Casual games [33, 34] are the most popular kind of mobile games, mostly due to the patterns of mobile phone usage. Casual games are characterized by extremely simple gameplay (e.g., puzzles or card games) and generally appeal to "casual consumers" who do not regard themselves as "gamers." Casual games are typically played in short bursts: during breaks, while waiting in line, on public transportation, etc. Their rules are simple, and, unlike many console games, they do not require a long- term time commitment or any special skills. Most importantly, current studies indicate that mobile gamer demographics are split fairly equally between males and females [33, 10].

One of the substantial barriers to adopting game development into the curriculum is the inherent complexity of developing a playable and well-designed game [15]. Due to the relative simplicity of many mobile games, a playable game can be developed within the timeframe of one semester by many CS students with reasonable programming skills.

DePaul University's GAM 386, Game Development for Mobile Devices [45], has as its prerequisite a sequence of Data Structures courses (in Java or in C++). The course covers multiplayer game programming with a wireless internet connection. Wireless and socket communication between devices are discussed in detail.

Networking and Mobile Computing.

Mobile computing makes its appearance as a main topic in some Networking courses. Such courses discuss the OSI Model as seen through the lens of mobile networking. To put these courses in perspective, the mobile development courses discussed in section 3.1 focus mainly on the OSI application layer and above as well as deal with orthogonal issues such as interacting with mobile device sensors, user interface design and app store ecosystems. Mobile networking courses focus almost exclusively on the seven layers of the OSI Model, ignoring the orthogonal mobile development topics.

Below we briefly describe two example mobile networking courses.

The University of California, Berkeley, offers CS294-7, Wireless Communications and Mobile Computing [49], a graduate course that covers topics such as radio spectrum, cellular networks, voice communications, mobile data, and, satellite networks. Course lectures focus on the physical, data link, network and transport layers of the OSI Model. Readings are taken from the research literature and projects include topics such as Mobile IP Security and Service Beaconing in Cellular Wireless Networks.

Washington University offers CSE574S, Advanced Topics in Computer Networking [47], which covers all 7 levels of the OSI Model as they apply to mobile computing. A small sampling of topics covered by this course include the wireless physical layer, WiMAX, 3G and 4G networks, mobile IP, and security.

3.3 In Outreach

As discussed in section 1, mobile computing has obvious strengths as a motivational factor to attract students to computer science. Hence, it is a popular technology for outreach programs. While outreach isn't a course or part of computer science curriculum, in our survey we found many instances of educators putting on camps and workshops using mobile devices [51, 42, 1].

MIT App Inventor [42, 4] is especially interesting as it offers a visual programming environment for Android-based development. The users can build programs by composing pre-defined blocks similar to those found in the Scratch programming environment [6]. App Inventor is included in the comparison of platforms in section 4.

4. SUGGESTIONS FOR ADOPTION

Mobile computing is a rapidly changing field with several popular and mostly incompatible platforms. Educators interested in adopting mobile computing into their courses may wonder how to navigate this complexity, so we offer here a breakdown of the current platform choices and course structures. This information is subject to change as new technologies and vendors emerge.

4.1 The issue of platform dependence

With respect to platform, there are three possibilities an educator may use to structure a course: teach one platform, teach multiple platforms, or, teach principles without regard to any particular platform. We consider each of these possibilities in turn.

Note that the word "platform" can have a few different meanings. "Platform" may refer to a particular hardware category (for example, iPhone versus Android phones). "Platform" may also refer to a mobile operating system (for example, iOS versus Android). "Platform" may also refer to the programming language or development environment used to create mobile software (for example, HTML5 versus Java). In many contexts, the distinction isn't critical. For an instructor, a "platform" is one of the alternatives for the hardware or software to use in a course. In this report, when it's useful to make the distinction, we refer to the "client platform" (which includes the hardware and the OS) and the "development platform" (which includes the programming languages, the IDEs, and other software development tools).

4.1.1 Teach one platform

Advantages.

Students get concrete experience with a real system. The concreteness provides a reliable way to test the students' understanding. (That is, you test by having students write code that works, or doesn't work.) Experience with a real system gives students a clear idea of the course's goals and helps students believe that the things they're learning are immediately applicable. Students can learn details about the entire app-creation lifecycle, from the formulation of an idea to publishing the app on a particular market.

Disadvantages.

Platforms change quickly. The tools that you teach one semester are likely to change (in either minor or major ways) by the following semester. Changes in these new platforms are sometimes not backward compatible.

Students might not own the required hardware. (For example, it might not be the case that each registered student has an Android phone or an Android tablet. For a course whose platform is iOS, students might have iPhones but not have an Apple computer, which is required for iPhone development.) Even if each student has an appropriate mobile device, students might have different makes and models of devices. Of course, students can use emulators and simulators to test their code, but the emulator/simulator experience isn't as rich as testing on an actual mobile device.

If some registered students don't have hardware that's appropriate to a platform-specific course, the institution can supply devices for the students' use. Needless to say, this involves some expense and some bureaucratic/administrative overhead. Fortunately, some companies are willing to lend or donate equipment to help defray costs. For example, RIM has an academic program in which the company lends devices for student use [41]. Microsoft also lends Windows phones for academic use [23]. In addition, some phone vendors donate devices for use by students.

4.1.2 Teach Multiple Platforms

Advantages.

This approach is parallel to the age-old idea of teaching many programming language paradigms in a computer science curriculum, and has advantages/disadvantages which are similar to those in the multi-paradigm approach.

In a multi-platform course, students learn (perhaps in an anecdotal fashion) which features are particular to a platform, and which features are common to several platforms (and thus are more fundamental). Students are in a better position to critique a specific platform's features (so they can begin to think about the design of mobile app platforms). Students see that the decisions made by platform designers aren't cast in stone. (Students see this "not-cast-in-stone" concept in other computer science courses, but the idea is always worth repeating.)

Disadvantages.

There's one big difference between the multi-paradigm tradition in computer science and a multi-platform mobile development approach: An institution typically implements the multi-paradigm approach by teaching different paradigms in different courses. In only one course devoted to development in several platforms, time is more scarce so depth-of-coverage for each platform is bound to be sacrificed.

4.1.3 Teach Principles Only

One might suggest that mobile computing be taught without reference to any platform.

Advantages.

The instructor doesn't have to worry about specialized hardware. The course material is not likely to become obsolete very quickly (although the broad concepts of mobile computing are still in some state of flux). Also, a course of this kind is market-agnostic.

Disadvantages.

At this point in the development of mobile computing, it's difficult to separate the principles from the realities of the hardware. Students with some experience developing software for desktop computers can benefit from a purely theoretical, hands-off course involving traditional computing principles. But no first-hand experience of the challenges of developing for mobile devices, a course based purely on principles isn't likely to resonate with students. In any course (mobile or not), concrete examples help students remember concepts. Without an underlying platform, if a student misinterprets a statement about a general concept, this misunderstanding isn't highlighted by a compiler or an app's incorrect behaviour.

A principles-only course eliminates the hazards of teaching to one or more platforms, but our survey of existing courses led us to no principles-only courses.

4.2 Notes on Existing Platforms

We now consider each platform in turn, comparing them by language, development environment, software and hardware requirements as well as application distribution channels and cost. We supplement this with a discussion of the advantages and disadvantages of each platform, with a special emphasis on using the platform for educational purposes. The reader should be cautioned that this discussion, while valid at the time of this writing, may become dated as the state of devices and platforms evolves. Having the discussion is, nevertheless, useful. A summarizing table of this discussion can be found in appendix A.

Note that in some cases, a client platform and development platform are intimately connected. For example, Objective-C is the primary language for developing iPhone

apps, App Inventor projects compile specifically to Android runtime packages, and TouchDevelop (a development platform) runs exclusively on Windows Phone (the client platform). In these cases, we use the term "platform" liberally, referring to both the client and development platforms at once.

Android.

Android development's *lingua franca* is Java, a primary language of instruction for many computer science departments. So many students don't have to learn a new programming language in order to begin mobile app development for Android. (Note: Android's Java is different from Oracle Java. Android uses Apache Harmony, and compiles source code for use on its own Dalvik virtual machine. But for most students who are starting out in mobile application development, the differences between Oracle Java and Android Java aren't noticeable.) Also, the heaviest tooling for Android development uses Eclipse plug-ins, and Eclipse is a commonly used IDE in educational institutions.

Android is a (mostly) open ecosystem, so students have quite a bit of flexibility. Android devices come from many different vendors, so the devices are competitively priced. For a small, one-time payment, students can join the Android market, now called Google Play [2] and publish their apps without going through a referee process. More advanced students can study the internals of Android with variants such as CyanogenMod.

iOS.

Development for iOS can be both simpler and more restrictive. A developer targeting the iOS platform need only consider a few different devices (Apple iPhone, iPod and IPad), and OS versions. In comparison, an Android developer may need to consider their applications performance on potentially hundreds of different devices and OS versions. iOS development usually means using the Objective-C programming language under Xcode, the Apple IDE that only runs on Apple hardware. This dependency on Apple hardware is probably the main drawback when choosing iOS as a platform for teaching mobile development. On the plus side, once given the hardware, the software is free and educational institutions can apply for a licence allowing their faculty and students to upload software to iOS devices.

Another possible disadvantage is the use of Objective-C, in the sense that, for most students, it will be a new language. Yet, this aspect is often overemphasized: in fact, Objective-C shares a large part of its object-oriented nature with Java, C# and C++. It also has the same main concepts that students already know from those languages. One major difference is the absence of garbage collection on the iOS platform. It is possible to use automatic reference counting (ARC), which allows for a much simpler transition from languages with garbage collection, as Java and C#. One could argue that making students worry and learn about memory allocation is an important part of their studies. It is interesting to note that in July 2012, Objective-C became the third language in the TIOBE Programming Community Index, in front of C++ and C#, and only surpassed by C and Java. In the same index, on July 2007, Objective-C ranked 46th. This extraordinary rise in popularity is clearly due to iOS development. The graph on the TIOBE site dramatically illustrates the rise of Objective-C. Hence, development for iOS can be seen as a motivator to teach and learn an additional programming language. It also offers a meaningful context to present and discuss memory management using a modern operating system and programming environment.

Windows Phone and Windows Runtime.

Programming for Windows Phone 7 can be done in any of the most popular .NET languages (C#, Visual Basic, F#). The new Windows Runtime (WinRT) programming model for Metro-style apps is available to a variety of supported languages and runtime environments, including C++, .NET languages, and HTML5. Visual Studio 2010 (for Windows Phone 7 development on Windows Vista or better) and 2012 (for Windows Runtime development on Windows 8) are available for free as Express editions. For a small annual fee, apps can be published in the Windows Phone Marketplace and the Windows Store. With Microsoft Dreamspark [22] project, students and educators can get free access to the Professional editions of the tools and they can get the annual fee waived to publish Windows Phone 7 apps. Apps created for Windows Phone and the Windows Runtime run on a variety of devices produced by many vendors.

BlackBerry.

The BlackBerry OS offers flexibility because it supports several development platforms including C/C++ (the native platform), HTML5, Adobe AIR ActionScript, BlackBerry Java, and the Android Java SDK. Apps written in any of these development platforms can be packaged for use on a BlackBerry device. In addition, through Research in Motion's BlackBerry Academic Program [41], instructors can get loaner devices for use by students in mobile app development courses.

HTML5, JavaScript, and CSS.

Mobile devices are now commonly equipped with powerful, HTML5-compliant web browsers making the web browser an ideal platform for mobile development. (For brevity, we frequently use the term "HTML5" to mean the HTML5, JavaScript, and CSS combination.) Modern HTML5-capable browsers present a JavaScript API capable of communicating sensor information, storing state locally, playing audio and video, rendering 2D and 3D graphics, communicating with servers, and handling touch, accelerometer, GPS and compass events. They are also capable of caching applications client side, making them available to a user even when offline. At present, though browsers typically do not have access to all device features (such as contacts and camera), they do give JavaScript applications access to a subset sufficient for the development of compelling mobile applications. Additionally, this platform, if sufficiently standardized, leads to the possibility of near universal application deployment. Mozilla's FirefoxOS project, formerly Boot to Gecko, represents the extreme of this approach, making the browser itself the platform for all device applications. On a FirefoxOS phone, the calendar, mail and dialer applications are all written using HTML5.

Below, we consider the advantages and disadvantages of HTML5 as a platform for a mobile development course. While mobile development using other platforms may require significant instructor and student preparation, HTML5 has a readily available development, deployment and runtime environment. One can build mobile applications using

any text editor and run them on any HTML5 compliant browser, on the desktop, on a tablet or on a mobile phone. This platform allows students to immediately deploy and update their projects on any web server, and so, be immediately consumed by family and friends, without the cooperation and constraints inherent in app store ecosystems. Additionally, accessing hardware on the mobile device is achieved through a JavaScript API. For example, touch, accelerometer and GPS are provided to applications via rich objects delivered to callback functions. This familiar model is used by Javascript developers writing everything from interactive web forms to AJAX applications. Finally, by leveraging student familiarity with JavaScript and HTML, Mobile development with HTML5 can easily be added to an existing web development course, instead of being added as an independent course.

The disadvantages of HTML5 as a platform include fragmentation, that is, the degree to which browsers support the current HTML5 standard. Additionally, as stated above, the HTML5 standard is currently missing support for many standard device features, for example, access to contacts and calendar and access to multimedia sensors. HTML5 does promise to include support for more device features in the future. In general, a user's experience depends on both the devices capabilities as well as the degree of HTML5 support. A few websites [5, 3] list current devices, operating systems and browsers and the degree of HTML5 support they provide. Some current, well supported platforms include the iPad 2, iPod 4, iPhone 3GS with recent iOS (5.1) and Safari. Samsung Galaxy Tab 10.1, Samsung, Galaxy Nexus running ICS and Firefox Mobile 10 or Chrome.

Apache Cordova (PhoneGap).

With the aforementioned HTML5, JavaScript, and CSS approach, a developer creates a web page that's optimized for use on a mobile device. The user runs a web app in a browser. But with Cordova, the developer's HTML5, JavaScript and CSS code is cross-compiled into platform-specific code (for example, into Android Java or iOS Objective-C). As a result, students can write one app, deploy the app natively to more than one device, and observe the differences in the native behavior on each of the devices.

MIT App Inventor.

App Inventor [4] is a visual blocks-oriented development platform for Android. App Inventor uses the jigsaw-puzzle-piece programming model in Scratch [6]. App Inventor's heritage is strong on teaching. In fact, App Inventor is similar to Logo and Alice in that students learning App Inventor focus on logic without having to deal with syntax.

Apps developed using App Inventor can be published on Google Play, but some features available to Android Java developers are not available in a project that's created using App Inventor [52]. For example, App Inventor Apps have limited total size, limited access to the device, limited access to the web, and, code is strongly tied to specific components, preventing the creation of abstract code.

TouchDevelop.

TouchDevelop [23] is a programming environment that does not require a separate development machine such as Mac or PC, but it runs directly on mobile devices (currently, Windows Phone). By merging the development and execution environments, many complexities in the traditional mobile app development model — which is basically cross-platform application development — are bypassed. While TouchDevelop's programming language is similar to other traditional procedural imperative languages, it uses a semi-structured code editor that has been specialized for touch-screens as the main input device. Programs can use many sensors and data present on mobile devices. The resulting simplicity of the mobile development environment make TouchDevelop suitable for introductory computer science courses. Apps development using TouchDevelop can be shared via the TouchDevelop script bazaar, and also published to the Windows Phone Marketplace using the regular publishing procedure.

4.3 Assignment Ideas

As part of our research, we sent out a call for instructors to share their interesting and nifty mobile assignments. Below we provide a summary of these as well as some that were found in other mobile courses we reviewed. When appropriate, we include the assignment's platform details. Links, where available, are provided at [14]. The reader may like to use the ideas presented in the assignment as is or move them to an alternate platform.

Pong.

Create a game in which the user drags a paddle to meet with an oncoming object.

This assignment involves understanding the game loop, drawing and rendering the scene so that it updates gracefully in real time, and responding to touch events when the user moves the paddle. Rudimentary collision detection is necessary to determine when the ball hits the paddle. Also, sound generation can enhance the feel of the game.

Auto Messaging.

Report sensor events via multiple channels. Part one of the assignment has students develop code which sends SMS messages to recipients, reporting the state of one (or more) of the device's sensors. Later, the project incorporates a text-to-speech engine to make phone calls reporting the state of the sensors.

This assignment involves using the platform's API to gather sensor data. The student also initiates outgoing communication (voice or text).

Programmatic Social Networking.

Present a mobile interface allowing a user to create, read, and update information using the Twitter API or the Facebook API.

This assignment makes significant use of UI design principles. The student is also immersed in the use of public APIs.

Maze.

Create a game in which the player navigates from the inside to the outside of a maze. (Create a new maze each time the user plays the game.)

This project requires touch-sensing and real-time drawing/rendering. While it's not strictly a mobile topic, creating a maze can be an interesting graph/recursion exercise. Students can explore other input modalities such as

device orientation (tilting) to navigate the maze, and the accelerometer to jump over obstacles in the maze.

E-Reader.

Create an e-reader with attention to bookmarking and searching. Later, add the ability to rate and share books.

This assignment emphasizes usability issues to make the reading experience as seamless as possible. The rating and sharing aspects of this assignment involve social networking and the use of public APIs. Alternatively, this assignment can appear in a web development course, with rating and sharing implemented as a custom web service.

Find the Duck.

Students create an application with multiple "users" and multiple "ducks," the targets to find. Users then physically race to the physical location of the ducks with their mobile devices. Each device displays the user's position on a Google MapView under Android and indicates the user's distance to a nearby target by displaying a "hot," "warm," or "cold" message. The application determines if the user is located at the target, and if this user is the first to arrive at the target. An additional requirement is to have the application determine if at least 2 other users (3 unique users) have checked into a particular area recently and, if so, return a Google StreetView image as a hint.

This assignment exposes students to location services as well as public API, web services and mapping. It also raises issues of concurrency and mobile communication, depending on the depth to which students wish to explore the application. Finally, this assignment raises questions about the ethics of user location tracking.

Geolocation via Access Points.

This application uses the beacon signals transmitted by neighbouring wireless access points to implement geolocation. First, the application measures the signal strength of the mobile device to nearby wireless access points. Next, the application submits these to a geolocation web services which determines the approximate physical location of a mobile device. Finally, the mobile application uses the result to request a photographic image of the scene that should be observable from that location.

In addition to the issues in the "Find the Duck" assignment, this assignment involves the use of access points and the handling of images from a web service.

WalkAbout.

Allow users to record their GPS location information as they travel. While the application records the user's GPS data, it displays it back to the user in the form of a path drawn on top of a Google Map. While recording data, the user can launch a Camera activity that will capture and store pictures on an SD-Card. When finished recording, the application gives the user the option of storing the current GPS data as a private application file to be loaded and displayed at a later time.

In addition to other geolocation issues, this assignment involves using the device's camera and writing to the device's local storage.

The "Me" App.

Display a virtual identity card, showing the owner's photo and vital statistics. Customize the identity card's layout depending on the device's orientation: a horizontal side-by-side flow, or a vertical top-to-bottom flow. Load the photo according to device pixel density. Label the data according to the phone's locale. In a follow-up assignment, add editing capabilities.

This assignment focuses on the UI (in particular, responding to changes in the device's orientation with changes in the UI). The student also gets practice supporting more than one locale.

Parking Space Finder.

Display a webcam-provided view of a parking lot, with current empty spaces highlighted. The webcam server actively scans for open spaces and communicates a marked-up image to mobile clients.

This assignment highlights computer vision and its related heuristics in order to determine which spots are open, and which are already taken.

AppRater.

Develop an application that suggests other apps for users to download and try. The purpose of the application is to share fun and interesting apps with other users. The users can then rate the downloaded apps.

Among other things, this assignment involves heuristics to work with people's' preferences. The student must decide on a method for classifying apps, and on methods for determining which apps are similar to other apps. As written, this assignment also focuses on the Android API, including posting to the notification bar, the idea of an Android Service and using Intents.

Top Places.

This assignment has students build an iPhone application which uses the Flickr API to present users with a list of popular Flickr photo spots. Users can then click on any of these popular spots and be shown photos taken at that location.

This assignment involves the use of a web service. In addition, students must think carefully about the user experience in browsing and marking places and photos. At the API level, this assignment focuses on MVC and the use of components, for example, the iOS UITableView and UITabBarController.

5. CONCLUSION

The widespread use of mobile devices by consumers is changing the face of computing and of computer science. Mobile computing offers both challenges and opportunities for computer science educators. The challenges include the ongoing need to keep up with changes in the field. Curricula must be revised, courses must be created, and educators must prepare to teach new material.

Another challenge includes the problems posed by the need for new hardware. The last wave of academic hardware enhancement was in the mid-1980s when personal computers hit the scene. Mobile devices are much cheaper now than personal computer were in the mid-1980s. But, unlike the

mid-1980s, the world's economies are weak and academic budgets are very tight.

Adding to these challenges is the fact that innovations in mobile computing have not reached a plateau. Today's investment in technology and training can quickly become obsolete.

But the opportunities outweigh the challenges. Mobile devices bring computing to the masses in a way that even personal computers cannot. And mobility sparks new interest in the economy's tech sector. Public awareness of their choices in consumer electronics purchases is at an all-time high. Opportunities for computer science educators include new computer science concepts, new ways to teach long-standing computer science concepts, and new ways to engage students.

We have suggested a workable definition of mobile computing, discussed some of the ways that mobile computing appears in current courses, outlined the topics covered in a mobile application development course, discussed the motivating influence the mobile platform has on our undergraduate students, provided guidance on platform choices for educators, and provided a few nifty assignments. We have provided a frame of reference for future discussions of mobile computing in computer science education as well as guidance for educators interested in integrating mobile computing in their courses.

Acknowledgements

We thank Paul Hegarty, David Janzen, James Reed, Jonathan Engelsma, William Mongan, and Chris MacDonald for the assignment ideas in section 4.3.

6. REFERENCES

[1] Alabama Summer Computer Camps. http://outreach.cs.ua.edu/camps. [Online; accessed 1-August-2012].

[2] Google play store. http://play.google.com. [Online; accessed 1-August-2012].

[3] The HTML5 test. http://html5test.com. [Online; accessed 1-August-2012].

[4] MIT App Inventor. http://www.appinventor.mit.edu. [Online; accessed 1-August-2012].

[5] Mobile HTML 5. http://mobilehtml5.org. [Online; accessed 1-August-2012].

[6] Scratch. http://www.scratch.og. [Online; accessed 1-August-2012].

[7] Apple, Inc. iOS human interface guidelines. http://developer.apple.com/library/ios/#DOCUMENTATION/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html. [Online; accessed 21-July-2012].

[8] Apple, Inc. iOS security. http://images.apple.com/ipad/business/docs/iOS_Security_May12.pdf. [Online; accessed 31-July-2012].

[9] Lucian Armasu. Android less secure? ha! iOS 5 seems to be full of security flaws. http://www.androidauthority.com/android-less-secure-ha-ios-5-seems-to-be-full-of-security-flaws-31326. [Online; accessed 31-July-2012].

[10] The Entertainment Software Association. Essential facts about the computer and video game industry. http://www.theesa.com/facts/pdfs/ESA_EF_2008.pdf. [Online; accessed 21-July-2012].

[11] Theresa Beaubouef and John Mason. Why the high attrition rate for computer science students: some thoughts and observations. *SIGCSE Bull.*, 37(2):103–106, June 2005.

[12] Jens Bennedsen and Michael E. Caspersen. Failure rates in introductory programming. *SIGCSE Bull.*, 39(2):32–36, June 2007.

[13] Maureen Biggers, Anne Brauer, and Tuba Yilmaz. Student perceptions of computer science: a retention study comparing graduating seniors with cs leavers. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, SIGCSE '08, pages 402–406, New York, NY, USA, 2008. ACM.

[14] Barry Burd, João Paulo Barros, Chris Johnson, Stan Kurkovsky, Arnold Rosenbloom, and Nikolai Tillman. Mobile course survey. http://www.edu4mobile.org. [Online; accessed 21-July-2012].

[15] Bary Burd, John Goulden, Brian Ladd, Michael Rogers, and Kris Stewart. Computer games in the classroom, or, how to get perfect attendance, even at 8 am. *SIGCSE Bull.*, 39(1):496–496, March 2007.

[16] Bill Buxton. Multi-touch systems that i have known and loved. http://www.billbuxton.com/multitouchOverview.html. [Online; accessed 1-August-2012].

[17] Canalys. Smart phones overtake client PCs in 2011. http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011, 2012. [Online; accessed 21-July-2012].

[18] Ching-Chiu Chao. An investigation of learning style differences and attitudes toward digital game-based learning among mobile users. In *Proceedings of the Fourth IEEE International Workshop on Wireless, Mobile and Ubiquitous Technology in Education*, WMTE '06, pages 29–31, Washington, DC, USA, 2006. IEEE Computer Society.

[19] Boston University Metropolitan College. MET CS683: Mobile application development. http://www.bu.edu/csmet/cs683/. [Online; accessed 1-August-2012].

[20] The Nielsen Company. America's new mobile majority: a look at smartphone owners in the u.s. http://blog.nielsen.com/nielsenwire/?p=31688, May 2012. [Online; accessed 21-July-2012].

[21] International Business Machines Corp. Ibm study finds consumers prefer a mobile device over the pc. http://www-03.ibm.com/press/us/en/pressrelease/25737.wss. [Online; accessed 1-August-2012].

[22] Microsoft Corp. Dreamspark. http://www.dreamspark.com. [Online; accessed 1-August-2012].

[23] Microsoft Corp. Touchdevelop. http://www.touchdevelop.com. [Online; accessed 1-August-2012].

[24] Delvin Defoe, Stan Kurkovsky, and Emily Graetz. Mobile game development projects for introductory cs courses: tutorial presentation. *J. Comput. Sci. Coll.*, 26(4):93–94, April 2011.

[25] ACM Interim Review Task Force. Computer science curriculum 2008: An interim revision of cs 2001. http://www.acm.org/education/curricula. [Online; accessed 1-August-2012].

[26] Andrea Forte and Mark Guzdial. Computers for communication, not calculation: Media as a motivation and context for learning. In *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 4 - Volume 4*, HICSS '04, pages 40096.1–, Washington, DC, USA, 2004. IEEE Computer Society.

[27] Google Inc. Android UI design patterns. http://www.google.com/events/io/2010/sessions/android-ui-design-patterns.html. [Online; accessed 1-August-2012].

[28] Mark Green and Michele Perras. Mobile computing research and education: bridging the gap between academia and industry. In *Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '10, pages 407–408, Riverton, NJ, USA, 2010. IBM Corp.

[29] Mark Guzdial. Education: Teaching computing to everyone. *Commun. ACM*, 52(5):31–33, May 2009.

[30] Google Inc. and Open Handset Alliance. Android design. http://developer.android.com/design/index.html. [Online; accessed 21-July-2012].

[31] Google Inc. and Open Handset Alliance. Android security overview. http://source.android.com/tech/security/index.html. [Online; accessed 31-July-2012].

[32] Eric Klopfer. *Augmented Learning: Research and Design of Mobile Educational Games*. The MIT Press, 2008.

[33] Elina M. I. Koivisto. Mobile games 2010. In *Proceedings of the 2006 international conference on Game research and development*, CyberGames '06, pages 1–2, Murdoch University, Australia, Australia, 2006. Murdoch University.

[34] Stan Kurkovsky. Making the case for mobile game development. In *Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education*, ITiCSE '09, pages 401–401, New York, NY, USA, 2009. ACM.

[35] Gretchen Livingston. Latinos and digital technology, 2010. Pew Hispanic Center, February 2011.

[36] Qusay H. Mahmoud and Allan Dyer. Integrating blackberry wireless devices into computer programming and literacy courses. In *Proceedings of the 45th annual southeast regional conference*, ACM-SE 45, pages 495–500, New York, NY, USA, 2007. ACM.

[37] Qusay H. Mahmoud and Allan Dyer. Mobile devices in an introductory programming course. *Computer*, 41(6):108–107, June 2008.

[38] Michael Mandel. The app economy. TechNet, February 2012.

[39] David Metcalf, Marcelo Milrad, Dennis Cheek, Sara Raasch, and Angela Hamilton. My sports pulse: Increasing student interest in stem disciplines through sports themes, games and mobile technologies. In *Proceedings of the Fifth IEEE International Conference on Wireless, Mobile, and Ubiquitous Technology in Education*, WMUTE '08, pages 23–30, Washington, DC, USA, 2008. IEEE Computer Society.

[40] OpenSignalMaps. Android fragmentation visualized. http://opensignalmaps.com/reports/fragmentation.php. [Online; accessed 21-July-2012].

[41] Ltd. Research in Motion. Blackberry academic program. http://us.blackberry.com/company/blackberry-academic-program.html. [Online; accessed 1-August-2012].

[42] Krishnendu Roy. App inventor for android: report from a summer camp. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, SIGCSE '12, pages 283–288, New York, NY, USA, 2012. ACM.

[43] Aaron Smith. Americans and their gadgets. *Pew Internet and American Life Project*, October 2010.

[44] Telenav. Survey finds one-third of americans more willing to give up sex than their mobile phones. http://www.telenav.com/about/pr-summer-travel/report-20110803.html, August 2011. [Online; accessed 21-July-2012].

[45] DePaul University. GAM386: Game development for mobile devices. http://www.cdm.depaul.edu/academics/pages/courseinfo.aspx?crseid=008285. [Online; accessed 1-August-2012].

[46] New York University. V22.0480: iphone programming. http://www.cs.nyu.edu/courses/spring09/V22.0480-004/. [Online; accessed 1-August-2012].

[47] Washington University. CSE574S: Advanced topics in computer networking. http://www.cse.wustl.edu/~jain/cse574-06/j_1int.htm. [Online; accessed 1-August-2012].

[48] Berkeley University of California. CS160: User interface design. http://bid.berkeley.edu/cs160-fall10/index.php. [Online; accessed 1-August-2012].

[49] Berkeley University of California. CS294-7: Special topics: Wireless communications and mobile computing. http://bnrg.eecs.berkeley.edu/~randy/Courses/CS294.S96/CS294-7.S96.html. [Online; accessed 1-August-2012].

[50] U.S. Census Bureau. *Statistical Abstract of the United States, 2008*. U.S. Census Bureau, Washington, DC, 127 edition, 2007.

[51] Jules White, Jeff Gray, and Adam Porter. Smartphones in the curriculum workshop (smack 2011). In *Proceedings of the 2011 24th IEEE-CS Conference on Software Engineering Education and Training*, CSEET '11, pages 520–522, Washington, DC, USA, 2011. IEEE Computer Society.

[52] David Wolber. App inventor capabilities and limitations. http://www.appinventor.org/capabilities-limitations. [Online; accessed 9-October-2012].

APPENDIX

A. PLATFORMS

Platform	Language requirements	Software requirements	Hardware requirements	App distribution methods	Development cost
Apple iOS	Objective-C	Required: Mac OS X with XCode and the iOS SDK (which comes with iPhone/iPad simulators for testing).	Required: Apple Macintosh Computer. Recommended: iPhone device(s) and/or iPad devices.	Distribution to a device is through the App Store (which requires approval by Apple's App Review Process).	Free to develop on a simulator. $99 annual fee for an individual to publish or send to any device. Free iOS Developer University Program for educators to push to devices.
Android	Primarily based on Java. C/C++ is possible for native development. Several scripting languages through the Scripting Layer for Android.	Required: Android SDK running on either Windows, Mac, or Linux. Recommended: Eclipse plus Android Development Tools (ADT) plugin plus emulators (Android Virtual Devices) for testing.	Required: Apple Mac, or PC with Windows or Linux. Recommended: Android-based mobile device(s).	Google Play, Amazon AppStore, third party markets, and self-hosted publication.	Free for development and self-publication. One-time $25 fee to publish on Google Play store. $99 annual fee to publish on Amazon AppStore (currently being waived).
Windows Phone and Windows RT	Any .NET language for Windows RT, HTML5 and JavaScript, C++.	Required: Either Windows Phone SDK (which comes with Windows Phone Emulator for testing), running on Windows Vista or better or Windows 8 SDK on Windows 8 or better. Recommended: Visual Studio 2010 or 2012 Express or better.	Required: PC with Windows Recommended: Windows mobile device(s).	Windows Phone Marketplace and Windows Store (which require approval by Microsoft's app certification process).	Free for development on emulator. $99 annual fee for an individual to publish or send to devices. Free publishing for students.
BlackBerry	C/C++ (the native platform), HTML5, Adobe AIR ActionScript, BlackBerry Java, or Android.	Required: Development software for either C/C++, HTML5, Adobe AIR, ActionScript, or Java running on either Windows, Mac, or Linux. Recommended: BlackBerry emulator.	Required: Apple Mac, or PC with Windows or Linux. Recommended: BlackBerry mobile device(s).	BlackBerry App World, third party markets, and self-hosted publication.	Free.

Mobile Web	HTML5, JavaScript, CSS.	Required: Windows, Mac, or Linux. Recommended: An HTML/JavaScript/CSS editor, such as Eclipse with JSDT, Visual Studio 2012, or Aptana Studio).	Required: Any device running HTML5 compliant browsers such as recent Chrome on Android and Safari on iOS.	Web apps may be published immediately via any website.	Free.
Apache Cordova (also known as PhoneGap)	HTML5, JavaScript, CSS. Note: Each Cordova app is wrapped specifically for Android, BlackBerry, iOS, Windows Phone, or some other environment.	Requirements are the same as those of the target environment (e.g., Android, iOS, Windows).	Requirements are the same as those of the target environment (e.g., Android, iOS, Windows).	Each app can be distributed via regular channels of the app's respective platform.	Same as target platform.
MIT App Inventor	Visual (drag-and-drop) development.	Web browser and Google account.	Recommended: Android-based mobile device(s).	Google Play, third party markets, and self-hosted publication.	Free for development and self-publication. One-time $25 fee to publish on Google Play store.
TouchDevelop	TouchDevelop language.	Required: TouchDevelop app.	Windows Phone.	TouchDevelop script bazaar (free). Optional: Windows Phone Marketplace (which requires approval by Microsoft's app certification process).	Free for development on an individual phone. $99 annual fee for an individual to publish or send to other devices. Free publishing for students.