Educating the Next Generation of Mobile Game Developers

Michael Zyda, Dhruv Thukral, and Sumeet Jakatdar
University of Southern California GamePipe Labs

Jonathan Engelsma, James Ferrans, Mat Hans, Larry Shi, Fred Kitson, and Venu Vasudevan
Motorola Labs

Mobile gaming is one of the fastest growing segments in the video game industry.[1] The $680 million acquisition of Jamdat by Electronic Arts to form EA Mobile in 2006 is indicative of how lucrative this segment is becoming to game development and publishing companies. The attraction lies in the huge, accessible market of mobile phones: it is estimated that more than two billion are in use. Most of these mobile phones have at least the computing power of earlier Pentium-based PCs. Yet, development on these phones is still far from simple, with various standards and platforms available for the plethora of phone types on the market. Even Java game developers have to publish hundreds of versions to cover the great variety of mobile devices.

Recently the push has been toward using open source Linux as the basis for a standard platform. Some mobile devices already ship with Linux, and the share of Linux devices is expected to grow. The bulk of today's devices ship with proprietary operating systems, while the rest use Linux, Symbian, Windows Mobile, AJAR, and other platforms. Platform fragmentation characterizes the industry, and Apple's iPhone, with its OSX-based platform, will not improve the situation.

Fragmentation makes it hard for mobile developers, who must support many different implementations. Working toward a common platform will enable faster and less expensive development and deployment of applications for third-party developers. Based on this broader direction is the initiative to build open source games that use the Linux platform and its rich functionality.

Through a partnership between Motorola Laboratories and GamePipe Laboratory at the University of Southern California (see Figure 1), we are exploring Linux's capabilities for mobile gaming and to provide developers with an alternative to what is predominantly a Java-based medium. Moving beyond Java lets game developers fully leverage the hardware advances and software capabilities of high-end smart phones. Doing so will help bring to mobile game development the same kind of developer's utopia we already have in PCs.

1 GamePipe Laboratory at the University of Southern California.

Initial preparation

To help achieve our goals, we divided the project into two phases: analyzing existing technology and undergoing required training.

Existing technology

To begin with, we evaluated current technology (hardware and software) to determine whether it was mature enough to provide the basic functionality we needed to build games—namely graphics, networking, and sound.

On the hardware side, we used Motorola's Linux-based E680i smart phones. The E680i has strong hardware specifications (see Figure 2 on page 92), with a huge touch screen and ample memory and processing power. Interestingly, the games preinstalled on these devices for China Mobile are nearly all Java-based, reflecting the current market.

On the software side, we chose to use whatever standard graphics, network, and audio libraries were available on the phone as part of the EzX platform. EzX, Motorola's Linux framework for mobile phones, is built on Montavista's Linux Consumer Electronics Edition running on a Linux Kernel 2.4 core. The framework comes with a range of APIs for standard applications, such as communications and multimedia. However, with the exception of Java 2 Micro Edition, the framework didn't have direct support for games. We chose to use the existing APIs for game development and, as the platform matured, add further API support to provide full flexibility to mobile game developers.

Continued on page 92

Continued from page 96

Network: GSM 900/ GSM 1800/ GSM 1900
Size dimentions: 109 × 53.8 × 20.5 mm, 105 cc
Display type: TFT touchscreen, 65K colors
Navigation: 8-way navigation key
Memory: 50 Mbytes shared memory
Card slot MMC/SD: Up to 2 Gbytes
Processor: Intel xScale 300 MHz processor
OS: Linux
GPRS: Class 10 (4+1/3+2 slots), 32-48 Kbps
WLAN: No
Bluetooth: Version 1.1

2 E680i technical specifications.

3 Symon Says uses Qt for the graphical interface.

In terms of graphics, the phones come with Trolltech's Qt. This is a C++-based GUI toolkit that developers use to build complex interfaces, such as the Linux K Desktop Environment. Although not primarily intended for game programming, Qt provides a powerful ability to build widgets, sprites, and other GUI-based elements that attracts game developers. The Qt API comprises useful graphics-related classes suitable for building 2D games. Through OpenGL, Qt also can provide an interface to 3D graphics, but this feature does not come with the E680i.

For networking capabilities, although the commercially shipped phones support numerous commonly used Bluetooth profiles, they didn't support the personal area networking profile. So, Motorola added the open source BlueZ Bluetooth stack for our use.

For sound, we chose to use either Qt's internal API for sound support—the E680i's implementation of the MPEG Audio Decoder library—or Libmad—a sound-decoder library that works for MP3 files.

Training

The next phase of the project involved the training we needed to work with all the internal tools that Motorola provided to GamePipe. The EzX framework is built on layers of standard as well as EzX-specific APIs, and has an SDK not yet generally available outside Motorola. Two graduate students went to Motorola's headquarters in Schaumburg, Illinois, for training on various aspects of EzX, ranging from setting up the desktop development environment to working with the phone setup.

Next, the two students undertook a project to create a multiplayer prototype game that used Qt for the graphical interface and BlueZ for networking. The purpose was to test the framework's feasibility for game development as well as test the learning curve for such an effort, considering that this collaboration was to teach students how to build games on Linux-based phones using APIs that they had never worked on before. The result of the effort was *Symon Says*, built in a space of six weeks.

The success of *Symon Says* (see Figure 3) enabled further possibilities of integrating new APIs into EzX. Moving from 2D to 3D was the next obvious direction. For this, we used Simple Direct Media Layer, a multimedia library written in C that abstracts over various graphics platforms and is widely used for Linux-based games. We ported SDL and SDL Mixer—SDL's audio component—to EzX to enable both 2D and 3D game development.

In addition to providing multiple API support for graphics and audio, Motorola provided another option for networking in the form of WiFi Secure Digital I/O cards. This option lets programmers use the standard UNIX sockets interface rather than going through the Bluetooth adapter and BlueZ specific bindings.

Design philosophy

With significant involvement from Motorola, the University of Southern California offered Advanced Mobile Games and Devices for the first time in the Fall 2006 semester. Mitch Lasky and Zack Norman, the founders of Jamdat, gave our introductory lecture. This confluence of a relatively new and constantly maturing platform and the participation of two of the best-known and well-respected personalities in mobile gaming made the course very attractive; thus, class enrollment was beyond both our expectations and planned capacity.

In their talk, Lasky and Norman stressed that despite the recent advances in mobile devices' graphic capabilities, mobile game development still favors design innovation over flashy graphics and extra features. They emphasized that mobile devices are inherently mobile, and that developers and designers must factor into game design that the end user is not sitting at home. People play games on mobile phones on demand, wherever they are. Mobile games must be tolerant of interference from the real world, as the human world is often noisy,

crowded, and busy. Moreover, mobile gaming remains a spare-time medium, so play time on these games should take no longer than a few minutes.

When designing a game, developers must also consider mobile devices' unique interfaces. Keypad layout, the ability to press multiple key presses all at once, and finger travel all can affect gameplay. Developers must also consider screen size, battery life, and network latency.

The games

Lasky and Norman's design philosophy influenced the game design of each of our student project teams in the class. Altogether, the four-to-five member teams had to create five multiplayer mobile games by the end of the semester, when they would showcase the games at GamePipe's Industry Demo Day in December 2006. Each of the games is multiplayer, requiring one phone per player.

Sizzlin Stylus

Sizzlin Stylus (see Figure 4) is a two-player game played using stylus pens. The game begins by showing both players a randomly selected path. The players start from different ends of the path at the same time and use their pens to move their characters along the path to the other ends. If a player moves his stylus outside the path, he must move it back in the path to continue. The player who finishes the path first wins the game. Both players have 30 seconds to finish the game; if neither finishes in time, the game ends in a tie. Both players' visited paths are painted in a different color, so the players know each other's position. The game was inspired by the arcade classic called *Flamin Finger*.

4 Sizzlin Stylus.

5 BattleBoats.

BattleBoats

BattleBoats (see Figure 5) is a two-player game in which each player chooses a fleet of three ships each drawn from types that have positive and negative attributes. They take turns moving and/or attacking the opponent's ships. At the beginning of each round, each player is given several points to allocate to moving ships, laying mines, or attacking enemy ships. Any unused points roll over to the player's next turn. The game ends when one player loses all of her ships.

6 Finger Football.

Finger Football

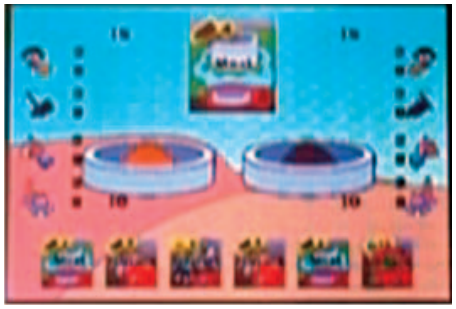
Finger Football (see Figure 6) is a multiplayer game in which players take turns and try to score a goal in a maximum of three strokes. Like the tabletop game, three coins form a triangle, and each stroke consists of attempting to hit the rearmost coin between the two front coins.

An unlimited number of players can participate in one game. Tournament play is in a knockout fashion, with the winner of each round advancing to the next. During one player's turn, the other players watch the strokes and coin positions. This turn-based game was designed to continue for a fixed number of rounds. The player with the most goals at the end of all the rounds wins.

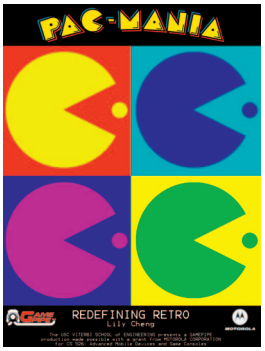
Titan

Titan (see Figure 7 on page 94) is a two-player game in which each player is building a castle, along with walls that shield it. By drawing cards from a deck, a player either builds up his castle to 100 points to win the game or destroys his opponent's castle by bringing it down to zero points.

The game also adds strategic elements in the form of card attributes, chief of which are strength, resource, God's favor, and gold. Strength is the ability to attack and destroy your opponent's castle and attributes. Resource includes such abilities as restoring your defenses and building up your castle. God's favor is the magic in the game, used for such actions as carrying out an enormous attack on your opponent's castle.



7 Titan.



8 Pac-Mania.

Pac-Mania

Pac-Mania (see Figure 8) is a two-player version of the classic Pac-Man arcade game. Each player controls a Pac-Man and races the other player to eat the most dots on the screen. Players can also eat each other for a time after eating "power pills." The player with the most points, or the one who eats up all the other Pac-Mans, wins.

Lessons learned

Last fall, GamePipe Demo Day showcased the work of students taking project classes at GamePipe. More than 35 representatives from companies such as Disney Buena Vista Games, Electronic Arts Los Angeles, THQ, Activision, Disney Feature Animation, Verizon Wireless, Crystal Dynamics, EA Mobile, and Motorola attended, and each appreciated the efforts of the students of the Advanced Mobile Games and Devices course.

Participants analyzed each game from design and technical standpoints to determine the project's maturity. At the end of the day, games from the Advanced Mobile Devices and Games course were considered sound in technology and design, and a few graduating students were also offered internships as a result.

Getting to this point, however, was an uphill battle.

Technological hurdles

The course's technological aspects were too intense for some students, most of whom had never worked on embedded systems. Every game leveraged various technologies, and each technology had its own learning curve, making progress estimation difficult.

However, once the students were past the learning curve, they warmed up to the platform quite well and completed almost 70 percent of production during the last six weeks of the semester. During that time, some teams even dabbled in adding multimodal features to their games. Multimodal user interfaces integrate voice, visual, and sometimes other user-interface modalities. Motorola ported their multimodal application framework to the E680i for our use.[2]

A couple of team projects also faced design flaws halfway through their implementation because they overlooked limitations of the medium. These teams had to rethink their game designs and modify their existing implementations to fix the design flaws without affecting their final production schedules. For instance, the alpha version of Titan was difficult to play because of information and display overload on the phone screen. There were just too many visual elements, making it hard to comprehend what was going on. The team needed to modify the game engine core to work with fewer game strategy elements, so that the team could simplify and unclutter the look and feel of the game.

Interpersonal hurdles

Team dynamics turned out to be an interesting soft issue. The students selected the initial four games by vote—out of 20 proposed game concepts. It was difficult to convince a few of the students whose games weren't chosen to work on teams, since they didn't like any of the games chosen. A couple of those students proposed forming a separate team so they could work on their own games. To avoid fragmenting the class into small groups, they finally chose to work on two selected projects and ended up making significant contributions to each. This approach reflects a philosophy adopted by GamePipe for all its classes, using such situations to acclimate students to real-world soft issues such as the ability to work in a team. The exercise also demonstrates that in industry you sometimes might be assigned projects you don't like, but you are still expected to do a good job.

By the end of the course, both the instructors and the students learned a lot about the entire process.

Future work

In the Fall 2006 semester, two projects tried multimodal interfaces in small ways. For example, one project used a multimodal interface to activate cheats in the game to make gameplay more interesting. The next iteration of Advanced Mobile Games and Devices will be offered in Spring 2007 and have a stronger focus on using multimodality as a key area in mobile gaming. Games will use speech recognition as a part of gameplay, and teams will design games around this multimodal enabler.

As we mentioned, Motorola developed and deployed a multimodal application framework on the E680i for this purpose.[2] As Figure 9 shows, the game server and game client software on the devices work in conjunction with a multimodal voice server deployed on the same local area network. Motorola's client framework allows access to multimodal enablers from within a Qt-based game. So, when a user gives a voice command to the game, it makes a request to the voice server (in our case over the WiFi connection).

This multimodal server is actually a slightly modified commercial-grade VoiceXML server. VoiceXML is the World Wide Web Consortium's standard markup for specifying interactive voice dialogues between a human and a computer.[3] It lets developers create and deploy voice applications in an analogous way to HTML for visual applications. Just as a visual Web browser interprets HTML documents, a voice browser interprets VoiceXML. Billions of calls are made each year using this technology. This VoiceXML example says "Hello world!" in synthesized speech:

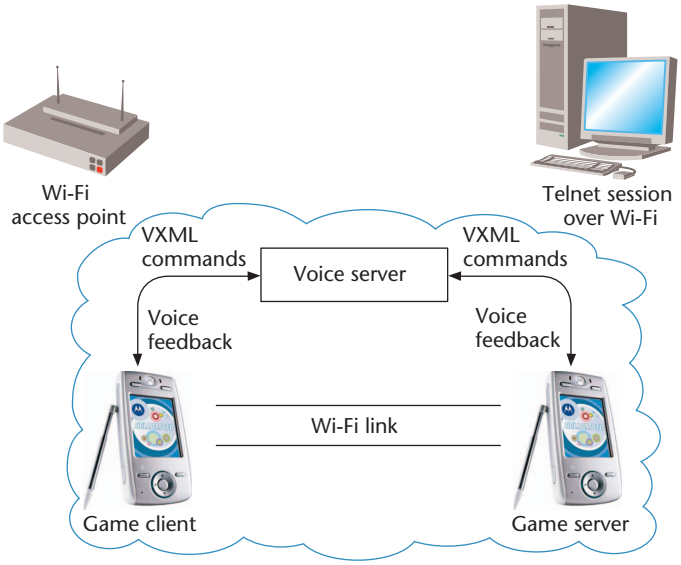
```
<?xml version="1.0"?>
<vxml version="2.0" xmlns=
 "http://www.w3.org/2001/vxml">
 <form>
 <block>
 <prompt> Hello world! </prompt>
 </block>
 </form>
</vxml>
```

Conclusion

One topic that came out of this experiment was the controlled multiplayer environment in which the games were built. The world space in all the games was highly localized, limited by the Bluetooth and WiFi ranges to either a room or a building. We did not test any games over the wide-area carrier data networks. A better test for the games' usability would be playing them on such a network, wherein latency and data consistency issues could drastically affect gameplay.

We were encouraged by the many possible directions to take the Advanced Mobile Devices and Games course in the future. With access to camera APIs, semacode image recognition, and rich context aware information that today's mobile devices can provide, developers can radically change gameplay. The expectation is to produce results of the same or better quality than what has been produced, and therein lies the challenge.

Despite various challenges, this collaborative effort to produce games on an open source mobile platform provides a feasible and flexible medium on which future developers can build state-of-the-art games using cutting-edge technologies. ■



9 Motorola multimodal application framework.

Contact author Michael Zyda at zyda@usc.edu.

Contact department editor Miguel Encarnação at me@imedia-labs.com.

References

1. J.R. Engelsma et al., "Ubiquitous Mobile Gaming," *Proc. System Support for Ubiquitous Computing Workshop* (UbiSys), 2006; http://www.magic.ubc.ca/ubisys/positions/engelsma.pdf.
2. D. Pearce et al., "An Architecture for Seamless Access to Distributed Multimodal Services," *Proc. 9th European Conf. Speech Communication and Technology*, Int'l Speech Communication Assoc., 2005, pp. 2845-2848.
3. S. McGlashan et al., "Voice Extensible Markup Language (VoiceXML) Version 2.0," World Wide Web Consortium (W3C) recommendation, Mar. 2004, http://www.w3.org/TR/voicexml20/.