FORUM NOKIA

Introduction to Mobile Game Development

Version 1.1; January 2003

Games

NOKIA

Contents

Contents

Legal Notice

Introduction to Mobile Game Development

Version 1.1; January 2003

1. Introduction

Modern mobile phones are small computers, with limited processing power by desktop standards, but power enough to run a small game. If you have a recent phone, you have more processing power in your pocket than ran the Lunar Lander.

Today's phones are also by their very nature networked computers, efficiently sending and receiving digital data. Primarily geared for voice data, they can send and receive other kinds of data as well. This inherent ability to share information offers a unique opportunity to design games wherein players interact with other players, perhaps even on the other side of the world.

In terms of processing power and capabilities, the current generation of Java™-enabled phones is close to the second generation of arcade machines, early 1990s home computers, and early handheld game machines. RAM is generally limited—typically 128 KB to 500 KB—although some smartphones, like Nokia 3650, have as much as 4 MB of memory. They also have, by comparison to PCs, limited input and display capabilities: small screens (many still black and white), keypads optimized for phone dialing rather than text entry, and limited sound handling.

What they lack in raw power they more than make up for in connectivity and sheer installed base. The vast majority of the world's adult population owns a least one mobile phone, and Nokia expects to ship between 50 million and 100 million Java-enabled, full-color devices by the end of 2003.

2. How Games Are Implemented for Mobile Phones

A number of different technologies, listed below, are used for games on mobile phones.

2.1 Embedded Games

Some games are programmed to run natively on a phone's chipset, installed on the phone at the factory, and shipped with it. Snake, available on many Nokia phones for more than four years, is the most famous example. New embedded games cannot be installed by the consumer, and they are becoming less prevalent.

2.2 SMS Games

Short Message Service (SMS) is used to deliver short text messages from one phone to another. Users typically pay about 10 cents per message. SMS games are played by sending a message to a phone number that corresponds to the game provider's server, which receives the messages, performs some processing, and returns a message to the player with the results.

SMS is not a particularly good technology for games, because it is dependent on text entry by the user, and thus is, in essence, a command-line environment. It is also expensive for a game of any depth, since a mere 10 exchanges with the server will cost a user $1 or more. Although the deployment of Multimedia Message Service (MMS) technology makes message-based games more appealing, this is still not a great gameplay environment and will not be explored here. Documents discussing SMS and EMS are available at Forum Nokia, www.forum.nokia.com, under the "Messaging" link in the right-hand column.

2.3 Browsing Games

Just about every phone shipped since 1999 includes a Wireless Application Protocol (WAP) browser. WAP is, in essence, a static browsing medium, much like a vastly simplified form of the Web, optimized for the small form factors and low bandwidth of mobile phones.

WAP games are played by going to the game provider's URL (usually through a link on the carrier's portal), downloading and viewing one or more pages, making a menu selection or entering text, submitting that data to the server, and then viewing more pages. One version of WAP (1.x) uses a unique markup language called WML and allows users to download collections of pages called decks. The new version of WAP (2.x) uses a subset of XHTML, delivers one page at a time, and allows better control over display formatting.

Either version of WAP offers a friendlier interface than SMS, and is generally less expensive for consumers who pay for airtime only, rather than by the message. But it is a static browsing medium; little or no processing can be done on the phone itself, and all gameplay must be over the network, with all processing performed by a remote server.

Phones will continue to contain WAP browsers, and developers may find WAP useful to deliver more detailed help or rules to players than can be contained in a game application, since most games are still subject to strict memory limits. However, game developers are moving to the richer capabilities of the J2ME platform. We will not explore WAP in any detail here. Documents discussing WAP and XHTML are available on Forum Nokia; click "Browsing/WAP" in the right-hand column.

2.4 J2ME and Other Interpreted Languages

Java 2 Micro Edition (J2ME) is a form of the Java language that is optimized for small devices such as mobile phones and PDAs. Nokia (and most other phone manufacturers) have made a strong commitment to Java phone deployment. Tens of millions of Java-enabled phones are already in consumers' hands.

J2ME is limited by comparison to desktop Java, but it vastly improves the ability of mobile phones to support games. It allows far better control over the interface than either SMS or WAP, allows sprite animation, and can connect over the air network to a remote server.

Because of its capabilities and the widespread and growing deployment of Java-enabled phones, it is a natural for mobile game development today, and we will examine J2ME game development in detail here and in subsequent documents.

J2ME is not the only interpreted language deployed on phones, but it is an industry standard backed by many manufacturers and therefore offers a large and growing installed base. Some proprietary interpreted languages have significant regional presence, including Qualcomm's Binary Runtime Environment for Wireless (BREW) in North American and a standard called GVM supported by some Korean carriers. Games initially developed for the large J2ME installed base can be recoded in these proprietary languages if a sound business case presents itself.

2.5 C++ Applications

Mobile games can also be developed in C++, a language that compiles to native machine code. Compiled languages in general offer better control over the UI, direct access to the phone's hardware, and greater speed for the same processing power when compared to an interpreted language. Development in C++ enables rich, high-performance games.

C++ developers can target Series 60 Platform devices. Series 60 Platform is a multi-vendor standard for one-handed smartphones that supports application development in Java MIDP, C++, and browsing environments. For the current list of manufacturers licensing Series 60 Platform, see www.forum.nokia.com/series60.

3. How Mobile Game Development Differs from Conventional Game Development

Mobile game development differs from conventional game development in a number of ways.

3.1 Team Size

Conventional PC and console games typically require teams of 12 to 30 people. Because most mobile games are less extensive than their console counterparts, they are typically developed by teams of 3 to 5 people, and often by lone programmers/designers.

3.2 Budget

Conventional games have budgets in the $1.5 million to $5 million range. Most mobile games are implemented on budgets of less than $100,000. Essentially, the limited display capabilities of mobile phones, coupled with limitations on application size, make it difficult to spend the huge amounts devoted to conventional games. This is, in a way, a strong advantage.

3.3 Development Cycle

Conventional games typically take two to three years to develop. Most mobile games are developed in a few months.

In other words, with a small team, and a small budget, you can develop and deploy a professional-quality mobile game. For many developers, frustrated by the conditions of the conventional game market, it is one of mobile game development's strongest appeals.

3.4 Networked Devices

Mobile games can be unlike any games we've seen before: limited in terms of media, but networked and multiplayer. Modems for PCs have been widely deployed only for the last eight years or so; consoles are only now going online. Mobile phones are networked devices by their very nature. Even though their processing capabilities are reminiscent of older computing technologies, their network capabilities are far superior.

3.5 Open Standards

Console development requires authorization and support from console game manufacturers, who use their control to require high "platform royalties" from game publishers, and to control what sorts of games get developed for their hardware. In the world of wireless (as in PC game development), you are free to develop whatever sorts of games you like, without paying Nokia, Sun, or anyone else. Moreover, the standards underlying mobile game development are published, open, and available for review by developers.

This is not, however, true of Nokia's N-Gage, and may not be true of future special-purpose mobile game platforms, which use more conventional game industry business models.

3.6 Deployment

Conventional games are (mostly) purchased in software outlets. Mobile games are (mostly) downloaded and installed by the user. In most cases, they are downloaded over the air network; some phones allow you to download an application to a computer, then hotsynch it to the phone.

Consequently, the distribution channels for mobile games are quite different. Users typically find mobile games through a carrier's game menu, a game menu pre-installed on the phone by a manufacturer, or a wireless portal such as Club Nokia, Handango, or Vizzavi.

It is possible to make deals with these distributors directly, though in many cases, you may find it worthwhile to work with a publisher or aggregator that already has distribution relationships—such as Nokia's Tradepoint Java Broker Service. For more information, go to the Forum Nokia main page; click the "Business Opportunities" link at right, and then the "Tradepoint" link.

4. Strengths of the Medium

4.1 Huge Potential Audience

More than a billion mobile phones are in use today, and the number is growing. In every developed country except the United States, a higher proportion of the population owns a mobile phone than owns a computer. While only a small portion of those phones are Java-enabled, and an even smaller number run an OS, the numbers are increasing rapidly, and within a few years, Java phones will be the norm. Your potential audience is larger than the potential market for any other platform—Playstation and GameBoy included.

4.2 Portability

There's a reason that GameBoy has sold more units than any other game console ever manufactured: portability is prized. People like being able to play whenever and wherever they choose. A phone may not be a great game device by comparison to modern consoles or computers, but people have their phones with them almost all the time. Give them good games to play when out of the home, and they will play.

4.3 Networked

Because mobile phones are networked devices, multiplayer games are feasible even given their other limitations.

5. Limitations of the Medium

5.1 Small Screen Size

You're dealing with a small form factor. While screen resolution (dots-per-inch) continues to improve and color screens are becoming the norm, screen sizes are likely to remain small because people don't like clunky phones.

A related issue: form factors are variable. Series 60 Platform devices offer a different form factor than Series 40 devices such as Nokia 5100. While Nokia has standardized its form factors to avoid fragmenting the market for developers, developers must still optimize their games for different phones—you want to use all the space available on a particular phone. Nokia supports a limited range of screen sizes, however, and the Nokia UI API for J2ME makes the problem easier to address.

5.2 Limited Color and Sound Support

Most phones in consumers' hands are still black and white, although most Java-enabled phones on the market today support color. Twelve-bit color is fairly common among such phones.

Even though phones are inherently sound devices, applications have a limited ability to play sounds. The J2ME specification doesn't require hardware manufacturers to support sound at all, although even primitive Java phones allow the use of some sounds, and MIDI support is increasingly standard. Generally, only one voice and one channel are possible.

5.3 Limited Application Size

Most Java-enabled phones have a limited amount of memory available for MIDlets (Sun's word for J2ME applications). Additionally, there's always a limitation on the size permitted for a MIDlet. The actual limit depends on the handset and (sometimes) the carrier's policies.

Designing a compelling game to such tight limits is a challenge, but remember that the first home computers had 64 KB or less, and some people still rave about their games. Limits are much less tight for smartphones—some, like Nokia 3650, can even run multi-megabyte applications.

5.4 High Latency

Latency—the amount of time it takes between the moment a machine makes a request and the moment it receives a response—is measured in microseconds on stand-alone machines; in milliseconds over the wired Internet; and in seconds over the air network.

Latency is a continuing problem for games on the Internet, and developers expend considerable effort dealing with the issues it raises. Air network latency is an order of magnitude worse, and makes it effectively impossible to develop fast-action multiplayer mobile games. Turn-based multiplayer games are quite feasible, however, and we'll discuss other ways of dealing with the problem later.

While the carriers are working to expand the amount of bandwidth available to mobile phones, they have not made latency reduction a priority, as it is less important for virtually every other kind of application. Hence this is not a problem that will go away.

There is one exception to this rule: phones that include Bluetooth or other wireless LAN technologies can communicate with other nearby Bluetooth devices at Internet latencies (200-400 milliseconds, typically). Thus, with smartphones like Nokia 3650, you can program fast-action multiplayer games played with other users nearby. Note: At present, Bluetooth is usable on Nokia phones only by Symbian OS appliations, not by MIDlets.

5.5 Interruptibility Is Crucial

When a user receives a voice call, that call will interrupt a game in progress. The application must be able to pause and recover without crashing, causing play problems (e.g., the monsters keep moving and killing the player even while s/he's chatting on the phone, returning to a lost game), or causing memory leaks. This requires some care when coding an application, but Nokia provides documentation to help both J2ME and Symbian C++ developers understand and address the issues.

5.6 Evolving Technologies

The technologies used to develop mobile games were not designed with games in mind, and there are often specific and awkward limitations as a result. As one example, the J2ME specification does not require support for transparency, which makes sprites look ugly over anything other than a blank background.

Luckily, most device manufacturers supplement Sun's requirements for J2ME with additional features—and most Java-enabled phones (including Nokia's) do support transparency. But the upshot is that to take full advantage of J2ME's capabilities, you need to support handset-specific APIs, and often need multiple versions of the same game for the best effect. The recently-released MIDP 2.0 specification addresses some of these issues, but as of this writing no MIDP 2.0-compliant phones are available.

6. Design to the Strengths, Avoid the Weaknesses

Games are amazingly plastic; they've been implemented for every technology from the Neolithic to today's high tech. Whenever you develop for a technology you haven't used before, you need to look hard at both its capabilities and its limitations, and work to use its capabilities to the utmost, while avoiding or working around its limits.

What conclusions can we draw from our discussion of mobile games' strengths and limits?

6.1 Short Play Times

Sooner or later, they'll want to make or receive a call. And they won't want to run the battery down playing your game. Ideally, each play session should be five minutes or less. That doesn't mean a complete game has to end in five minutes—instead, you can allow players to interrupt, save, and resume games. This is a spare-moment platform. Someone with a couple of hours to kill will find a better game device.

6.2 Play on Their Schedule, Not Yours

Let people pick up and play whenever they want. Don't force them to wait (if you can avoid it), and don't require them to be in the game at any particular time.

6.3 Avoid Latency Issues

That's easy for a soloplay game. Multiplayer games are still doable, but you need a solution for the problem of latency. (We'll discuss that later.)

6.4 Use the Network

...not necessarily for every game, but the feeling of competing with others, even if only for a leaderboard position, is a big draw for many players. Remember that the phone is essentially a social device, and adding some kind of social component to your game will benefit it.

6.5 Keep the Game as Small as Possible

Remember that people still rave about the great games of the 80s. In some ways, technical limits force you to pay more attention to basic gameplay. And remember also that some of the poetry in the English language is in sonnet form—a highly restrictive medium. Constraints can breed creativity.

As we delve more into development, we'll discuss some techniques.

6.6 Plan to Support Multiple Handsets

At a minimum, expect to support different screen sizes, which for Nokia devices is easy. Develop one version for Series 40 and another for Series 60. In many cases, you will also want to take advantage of phone-specific features and APIs, such as Nokia's UI and SMS APIs, while doing more limited versions for phones that don't have the same feature set.

6.7 Plan for Ease of Localization

Mobile phones are in use worldwide, and English will not work for many markets. Plan your development to make localization into another language as easy as possible.

7. Dealing with Latency

How can mobile games work around the high latency of the air network?

7.1 Soloplay Games

Single-player games don't need to hit the network, except perhaps to upload a high score to a leaderboard, or allow the player to view the leaderboard (and not even then, if you don't implement such a feature). That kind of communication is not critical to gameplay, and a delay of a few seconds will not be objectionable to the user. Essentially, it's not an issue for most single-player games.

7.2 "Multiplayer" Soloplay Games

In a multiplayer soloplay game, players feel as if they are playing in a multiplayer game, but actually, each is facing the same single-player challenge, with scores compared at the end of the game or round. When a player joins a game, s/he is told the IDs of the other players. S/he then plays a single soloplay game. The server either sends a gamestate file containing identical information to each player, or else it sends a code from which the client software constructs a starting gamestate. Each player plays the game, trying to achieve the highest score. When a player is done, his/her client submits the player's score to the server. When all players are done (or after a timeout), the server tells each player who had the best score and what scores each player achieved.

Games of this style are quite successful on the Internet; Slingo, for many years AOL's most popular game, is a good example.

Because information only needs to be exchanged with the server at the beginning and end of each game or round, latency only becomes an issue at those times.

7.3 Turn-Based Games

In a turn-based game, players enter their moves, and then expect a little delay before they receive the results. A delay of a few seconds is tolerable.

There are two types of turn-based games:

7.3.1 Round-robin games

In a round-robin game, each player takes his/her turn in order. Players wait until it is their turn to act. Class games like Chess, Poker, and Bridge are examples.

The drawback to such games is that players do nothing (except perhaps watch other people's card play) until their turn comes around again. This can be moderately dull, although classic games do receive a great deal of play on the Internet.

As a result, it is a good idea to limit the number of players in a round-robin game, so the delay does not become too extreme. Two to four players is manageable.

7.3.2 Simultaneous movement games

In a simultaneous movement game, each player plans his/her moves independently from the others. When a player is ready, s/he sends his/her orders to the server. The server waits until it receives orders from all players, then resolves the turn, and dispatches the results to all of the players.

Simultaneous movement games are rare in electronic gaming, although the XCom series is an example of how this can be done well. Another example is Laser Squad Nemesis (see www.lasersquadnemesis.com), which was developed by the original developers of the XCom series. Simultaneous movement is more common in boardgames (e.g., Diplomacy).

7.4 "Act Whenever" Games

In an "act whenever" game, the game persists for a long period of time (days, weeks, months, possibly forever). Players can sign into the game at any time, and perform actions in the game. In some games, they might only be able to interact with (attack or perform other actions against or in concert with) other players who are also signed into the game at the same time; in other games, they might be able to interact with the positions of other players in the game even if they are offline.

In order to prevent obsessive players from gaining a large advantage over casual players, it is useful to limit the number of actions a player can take during a period of time. It encourages players to sign into the game for short, frequent sessions, but not too many. This reduces the load on your server as well.

Massively multiplayer games like EverQuest are, in a sense, "act whenever" games; the world persists even when your character is offline. Dataclash (www.dataclash.com), a WAP game from NGame, is another example: you are a hacker, and you build your defenses when signed in, and can attack the defenses of other players, also. But defense against attacks happens even when you are offline.

7.5 Slow Update Games

A slow update game operates continuously and persistently on the server. A player can sign in at any time to see how s/he is doing in the game, and change the default behavior for the units/cities/characters/whatever s/he controls in the game. They continue performing their default behavior even when the player is offline. The player does not have to manage his/her position in the game continuously, although players who check in and modify their orders frequently have some advantage, because they will respond more rapidly to changing conditions.

An example of a slow update game is the old PC game Empire (based on an earlier academic game played on Unix machines). A player sets each city to producing a kind of unit, and sets a destination for those units, which move to the destination after production. Builds and destinations can be changed at any time, but will continue until changed.

7.6 Build Latency into the Fantasy Behind the Game

Another approach is simply to accept a high level of latency, and attempt to justify its existence through the game rationale itself. For example, most starship combat games feel like World War II air combat, with ships zipping around and shooting at each other. Instead, it might be feasible to do a game that feels more like World War I naval combat, with starships maneuvering slowly, firing volley after volley at each other, with missiles moving slowly across space toward their destination. You could conceivably hide a few seconds of latency with such an approach.

An example here is Quake, which hides the 200 to 400 millisecond latency typical of the wired Internet by ensuring that, in most cases, it takes several hundred milliseconds for bullets to reach their target—thus the firing player's machine knows whether or not the target was hit before it must display the appropriate animation.

7.6.1 ...Be creative

There are doubtless other solutions for multi-second latency. It's a problem worth addressing.

Games for mobile phones must cope with the fact that you have a much smaller screen to play with. Some ways of addressing the problem are explored below.

8. Dealing with the Form

8.1 One Small Window on the World

Avoid designs that require a player to look at many different locations in a large space in a short period of time, or to switch rapidly from a view at one location to another view at a distant location. It's very difficult to do this without a large screen and a pointing device. Examples of games that would not work on a mobile phone include Civilization and StarCraft (or any real-time strategy game).

It's ideal if you can create a game whose entire world can be displayed on a small screen—Chess, for instance. But another approach is to make the screen a window onto a larger world, and allow the player to scroll around the world. This works best if the player controls only a single actor in the world—a side-scrolling arcade-style game, or a role-playing game, for instance. You only need to display the portion of the world directly encompassing the player's actor.

8.2 Games that Avoid a Layout

Another approach is to avoid games that require a physical depiction of space. Rock-Paper-Scissors is an example; virtual pet games are another. These games can be carried in text, with some cute animations and sounds.

8.3 Zoom Levels

Yet another approach is to have a large world, but to allow the player to "zoom in" and "zoom out." With this approach, s/he can see the entire world but with very little useful detail, then zoom into a specific area of interest. It is still awkward to make large jumps across the world, so the game should be designed so that a player needs to examine only a few areas of interest in a short period of time.

9. Designing to the Business Model

Designers of conventional games generally deal with fairly stable platforms and well-established business models. In wireless gaming, technology is changing rapidly, and business models are still evolving. Some of the preconceived ideas about games, based on these older models, don't apply to wireless, or at least not to the same degree:

"You need at least 12 hours of gameplay." That may be true for $50 PC or console titles. It isn't true for mobile games, which are cheaper to play.

"Keep them online as long as possible." That may be true for online games that drive ad impressions; unless your game is ad-supported, it's not true for wireless. It was also true for online games back when people paid by the hour, and MMGs still reflect that legacy. It isn't true for mobile games, unless you're getting a share of airtime revenue, and maybe not even then (you don't want people to be shocked by excessive phone bills for gameplay).

9.1 Application Sale

At present, the usual model for J2ME games is a one-time application sale: the user pays a fee, downloads and installs the game, and plays to his/her heart's content. You probably want a demo version (first level, perhaps) for free.

This encourages you to design soloplay games; multiplayer games require you to provide after-sale support via the game servers, imposing continuing hardware and bandwidth costs.

Note also that it's feasible to do a game as several applications—each MIDlet containing, say, five levels of gameplay. This is one way to get around the size limits for MIDlets. It's also a way to earn additional revenues from your customers, by selling them the next "level pack."

9.2 One-Month License

Some operators allow you to sell a limited-duration license—the user has to pay again if s/he wants to keep on playing after a month or a few months. This makes multiplayer games more feasible; you can think of a monthly license as equivalent to a monthly subscription. Most soloplay games have finite use; after the user has played through all the levels, s/he probably doesn't want to play again. Most multiplayer games can be played repeatedly, because you never know what your opponents will do. Thus, multiplayer games are likelier to support repeated license fees.

If this is your model, however, avoid games that take several days or weeks to play, and in which one player dropping out makes the game less fun for the other players. Inevitably, some players will choose not to renew at the end of the month, even if they are in the middle of a game.

9.3 Share of Data Traffic or Airtime Revenue

Some Asian carriers offer a share of data traffic revenue (users are charged per 256 byte block) to game providers. Most European and North American carriers share a portion of airtime revenue to WAP game providers (though not necessarily to J2ME or BREW game providers); as networks move to GPRS or other fully packetized air standards, they are likely to move to the Asian data traffic model.

This model makes multiplayer games more desirable, since network traffic generates additional revenue. The danger is that users may not realize how much they've spent until they receive their bill at the end of the month, thus it may not be optimal to encourage too much usage.

At this point, you are no longer in a product business, however—you are in a service business. Your revenues depend not on landing a one-time sale, but on keeping people playing as long as possible. Customer support (and stable servers) becomes of paramount importance.

10. Be Excited!

In the conventional games industry, spiraling budgets make publishers increasingly conservative about what they'll fund. In wireless gaming, budgets are low—and nobody has really figured out yet what types of games are going to do best in this environment. Thus, innovation is far easier. If, like many developers, you're frustrated with the restrictions of conventional game development, mobile games are an arena where you still have an opportunity to innovate and show what you can really do.

Development cycles are short—typically a few months, instead of a few years. Games are hit-driven, meaning most fail, but it's better to fail after three months of effort than three years. And in three years, it might be possible to develop 12 games, giving you much better odds of generating a major hit.

We're still at the early stages of a new gaming medium, and you may be the Miyamoto or Sid Meier of this new medium, the person who establishes a hugely successful style of games. Remember, Richard Garriott programmed Ultima in three weeks. Nowadays, it's impossible to succeed with a game like that in the conventional game industry, but it's more than possible in this new game arena.

Someday soon, someone will blow us all away with a mobile game that really takes advantage of the technology, that demonstrates how to do a connected game for limited devices that's more compelling, in its own way, than anything on a PC or Playstation or Xbox.

Now there's a worthy challenge.

Build Test Sell

Developing and marketing mobile applications with Nokia

Go to Forum.Nokia.com

1

Forum.Nokia.com provides the tools and resources you need for content and application development as well as the channels for sales to operators, enterprises, and consumers.

Forum.Nokia.com

Download tools and emulators

2

Forum.Nokia.com/tools has links to tools from Nokia and other industry leaders including Borland, Adobe, AppForge, Macromedia, Metrowerks, and Sun.

Forum.Nokia.com/tools

Get documents and specifications

3

The documents area contains useful white papers, FAQs, tutorials, and APIs for Symbian OS and Series 60 Platform, J2ME™, messaging (including MMS), and other technologies. Forum.Nokia.com/devices lists detailed technical specifications for Nokia devices.

Forum.Nokia.com/documents
Forum.Nokia.com/devices

Test your application and get support

4

Forum Nokia offers free and fee-based support that provides you with direct access to Nokia engineers and equipment and connects you with other developers around the world. The Nokia OK testing program enables your application to enjoy premium placement in Nokia's sales channels.

Forum.Nokia.com/support
NKN.Forum.Nokia.com
Forum.Nokia.com/ok

Market through Nokia channels

5

Go to Forum.Nokia.com/business to learn about all of the marketing channels open to you, including Nokia Tradepoint, an online B2B marketplace.

Forum.Nokia.com/business

Reach buyers around the globe

6

Place your applications in Nokia Tradepoint and they're available to dozens of buying organizations around the world, ranging from leading global operators and enterprises to regional operators and XSPs. Your company and applications will also be considered for the regional Nokia Software Markets as well as other global and regional opportunities, including personal introductions to operators, on-device and in-box placement, and participation in invitation-only events around the world.

Forum.Nokia.com/business