FORUM NOKIA

Series 60 Developer Platform 1.0: Usability Guidelines For J2ME™ Games

Version 1.1; April 21, 2004

Series 60, J2ME

NOKIA

Contents

List of Tables

List of Figures

Change History

October 29, 2003	Version 1.0	Final / Idean Research Ltd, www.ideanresearch.com
April 21, 2004	Version 1.1	Minor updates to terminology, Chapter 2 updated

Series 60 Developer Platform 1.0: Usability Guidelines For J2ME™ Games

Version 1.1; April 21, 2004

1 Introduction

This document provides guidelines for developing easy-to-use, fun, and challenging Java™ 2 Platform, Micro Edition (J2ME™) games for Nokia Series 60 mobile phones. Research with three different J2ME mobile phone games from three developers was conducted in order to come up with the general usability recommendations for this dimension path (Series 60 J2ME games). The results are reflected in the following guidelines and recommendations. Testing with real users was conducted with a Nokia 3650 device.

The most important guidelines are listed separately; these are the guidelines that are violated most frequently, with the most serious consequences. The guidelines that apply only to the dimension path studied (Series 60 J2ME games) are marked accordingly. They may apply to other platforms and devices as well, but have not been tested on them.

A separate game usability guideline document exists for Series 40 J2ME games. The contents of these two documents have much in common. Most significant differences between Series 40 and Series 60 guidelines are summarized in Appendix A.

1.1 Related Documents

The following related documents provide additional information:

Document name
Series 40 Developer Platform 1.0: Usability Guidelines For J2ME™ Games
Designing Single-Player Mobile Games
Developing Java™ Games for Platform Portability: Case Study: Miki's World
Guidelines for Game Developers Using Nokia Java™ MIDP Devices v1.0

2 Foreword by Tom Ojala

Tom Ojala, Director, Marketing, Forum Nokia

GAMES USABILITY ON SERIES 60 DEVELOPER PLATFORM 1.0

Usability is an integral part of the mobile application creation process. Mobile devices use size, form factors, user navigation, input logic, and styles to differentiate themselves and to appeal to different audiences, service consumption patterns, and preferences, all of which may vary greatly among different cultures.

Usability should not be the last step of the R&D process through end-user testing; rather, it should begin in the application design phase and remain a constant factor throughout the entire development process. Contexts such as multiplayer networked games pose additional challenges in delivering end-user enjoyment.

Usability means success. At best, it means increased consumer service adoption, increased consumption, and repeat usage, and may result in viral marketing benefits. This in turn translates into profitability, greater returns, and faster payback on the R&D investment.

Consumers don't care about code; what makes them use a service is satisfaction with their experience. Lack of usability, in turn, is a barrier to service adoption and growth, and under-delivery of usability can result in the user rejecting the service.

Nokia's Developer Platforms help developers address larger markets. Multiple devices comply with a common API definition per Developer Platform release specifications, which allows developers to use this common functionality as a core for application business logic and usability. Developers can then optimize functionality and usability for different elements in specific devices.

Differentiators that are not part of the common definition include device-specific user interfaces, hardware limitations, and leading technologies implemented on top of the common software. These new features enable developers to implement functionality as soon as it is ready, but utilizing it requires a modular application design approach.

Series 60 Developer Platform 1.0 offers developers a rich and consistent application development environment for Java Mobile Information Device Profile (MIDP) and C++. Series 60 Developer Platform 1.0 uses Symbian OS v6.1 and the MIDP 1.0 runtime environment. The first announced device compatible with the specification is the Nokia 3650 device.

The usability guidelines for games, described in detail in this document, are for the Series 60 Developer Platform 1.0 Java/MIDP 1.0 environment.

3 Mobile Phone Game Usability

3.1 Definition

Usability is not a one-dimensional property. It has many overlapping components; some even contradict one another. In most cases, usability is associated with the following attributes:

Satisfaction	A subjective feeling of contentment
Efficiency	A minimal amount of time wasted
Learnability	Degree of ease when starting to use the system
Errors	Number of errors the user makes and degree of seriousness
Memorability	How well the user remembers the system when returning to it

Table 1: Usability factors

Game usability needs to be differentiated from playability, which refers to a user's overall experience with a certain game. The most comprehensive definition of playability states: *The degree to which a game is fun to play, with an emphasis on the interaction style and plot-quality of the game; the quality of gameplay*. Playability is affected by the quality of the storyline, responsiveness, pace, usability, customizability, control, intensity of interaction, intricacy, and strategy, as well as the degree of realism and the quality of the graphics and sound.

The importance of the usability factors listed in Table 1 varies – for example, in a flight-booking system for expert use, efficiency and lack of errors are very important, but for an information kiosk, learnability, memorability, and satisfaction are higher priorities.

Game applications are not terribly complicated when compared with word-processing applications, for example. Mobile phone games are typically played for brief periods of time, and are played for enjoyment or challenge, which poses different usability needs. The special nature of games, especially mobile games with their small screens, creates specific needs for their user interface.

Care must be taken to ensure that the game interface and concept are pleasing to the user. The key to usability is simplicity – a complex solution is itself a problem. Efficiency is not a particularly important usability attribute for games, at least not in and of itself. In the end, of course, efficiency produces satisfaction and must not be ignored, but the user is not usually trying to leave the game as soon as possible.

Figure 1: Components of playability

The most important of all usability criteria is simple: *Know the user*. In order to design a product, a designer must know his or her audience. Guessing about the age or education level of users is a risky foundation for a business.

Even with demographic data at hand, it is not always clear what conclusions to draw. With mobile games, it is essential to know where they are played, for how long at a sitting, in what situations, how they are paid for, what the function of playing is, etc. All of these factors should be considered in the design.

3.2 Usability Requirements

Different usability attributes are usually weighted differently. With games, satisfaction is probably the most important criterion. However, measuring usability requirements can pose problems. Some attributes are somewhat subjective, such as satisfaction, but their target levels can be set and evaluated nonetheless. Table 2 lists possible metrics for defining and evaluating the usability standard.

Satisfaction	Anonymous questionnaire
Efficiency	Time to perform certain task(s)
Learnability	Time to reach certain stage(s)
Errors	Number of errors, while using a system, in certain task(s)
Memorability	Number of correct answers (a memory test)

Table 2: Usability metrics

3.2.1 How to set the requirements

Different levels of usability can be specified for each attribute evaluated. For example, let's assume satisfaction is measured on a scale from 1 to 5. Optimal performance is 5, but the target level may be set at 4, which is still difficult to achieve. The minimum acceptable level could be set at 3 and the current level assigned a value of 2.5.

Other attributes, such as the number of errors, are more straightforward to evaluate. Identifying them may, however, require experience and expertise. A fairly simple method is to evaluate an existing user interface with real users and use that as a basis for new requirements.

When no competing products exist to use as a benchmark, usability goals are very hard to establish. However, projects rarely start from scratch without some kind of prior knowledge or experience. The minimum level of usability should be set to what it was on a previous project.

3.2.2 When to set the requirements

Usability requirements should be defined when the other requirements of the game have been established. At that point, it is not usually possible to define exact numerical values for the evaluation, especially as the questionnaires have not been drafted and the final game UI has not been implemented. However, the requirements can be established in a less detailed way and the weight of each usability attribute can be assessed. Typically, the usability requirements are defined after the technical requirements have been set.

If the usability requirements are defined too late in the process, it may be too expensive to implement them, even if it turns out that the requirements are not being met. Therefore, the requirements must be laid out before programming begins, and usability should be evaluated against the requirements at different stages of the development cycle.

3.3 Cost Justification

The product development cycle as a whole must be profitable. This means that eventually usability research and implementation must pay off in increased sales, increased productivity, decreased support, etc.

3.3.1 The user's experience

If users feel stupid, incompetent, or frustrated when using an application, they probably won't touch it again unless they have to. With games, users *never* have to play – this means that a positive user experience is a necessity. The whole point of a game is to provide a fun user experience.

Companies that invest in ease-of-use enjoy increased sales and a positive image. Good usability as a result of user testing, usability research, and implementation of these guidelines will yield at least some of the following results:

- Shorter learning curve for games
- Increased likelihood that players will buy the game
- Developer focus on the game instead of user interface details
- A harmonized user interface of Series 60 and Series 40 games
- Decreased number of expensive changes in the final stages
- Increased sales through reliable company reputation
- Shorter development cycle for games and reduced number of necessary iterations

These benefits have been established through research[1], some via Web sites rather than applications. Investing in usability can be summed up as follows:

- Return on investment in usability for a typical project is between 200 percent and 800 percent.
- Average percentage of development budget required for usability is just 2.2 percent.
- Key cost metrics (such as support calls, use of help features) fall by at least 10 percent.
- Conversion ratios (sales) increase by an average of 16 percent.

Case studies are too numerous to be listed, but a few examples make the point:

- For developers and manufacturers, the advantages of creating usable products far outweigh the costs. The rule of thumb: Every dollar invested in ease of use returns $10 to $100.[2]
- Systems designed with usability engineering have typically reduced the time needed for training by about 25 percent. User-centered design typically cuts errors in user-system interaction from 5 percent to 1 percent.[3]
- Revenues for one Digital Equipment Corporation (DEC) product that was developed using user-centered design techniques increased 80 percent for the new version of the software, and usability was cited by customers as the second most significant improvement.[4]

For a more comprehensive list of cases, please visit the Usability by Design Web site.[5]

3.3.2 Make it right

Investing in usability may seem like an extra burden on top of everything else, but this view doesn't take the benefits into account. The sooner problems are found, the more time is saved and the more money is made. It's about 40 to 100 times more expensive to fix problems in the maintenance phase of a program than in the design phase.[6] Again, there are several case studies and research projects proving these points:

- For enterprise software and Web sites, 80 percent of the software lifecycle costs occur after the product is released, in the maintenance phase. Of that work, 80 percent is due to unmet or unseen user requirements; only 20 percent is due to bugs or reliability problems.[7]
- Without a user-centered design, the user interface of a software product typically has about 40 flaws that can slow users and lead to errors.[8]
- Around 63 percent of software projects exceed their cost estimates. The top four reasons are: 1) Frequent requests for changes from users, 2) Overlooked tasks, 3) Users' lack of understanding of their own requirements, 4) Insufficient user-analyst communication and understanding.[9]
- If it costs $10 to make a program change during development, it will probably cost $400 to make the change after the system is in the field.[10]

For a more comprehensive list, please visit the Usability by Design Web site[11]. For additional reading, see *Cost-Justifying Usability*[12] by Randolph G. Bias and Deborah J. Mayhew.

3.3.3 Usability return on investment

To sum it up, Jakob Nielsen[13], a usability pioneer and guru, states that developers should invest 10 percent of their project costs in usability for maximum payoff.[14] However, return on investment (ROI)

is more difficult to assess with usability, since costs are measured in money while usability is measured in increased use, more efficient use, or greater user satisfaction.

Converting usability improvements to dollars is easy in e-commerce, where doubled sales have an immediate value. For intranets, productivity gains are also fairly easy to convert into monetary estimates: Simply multiply the time saved by the hourly cost of employees.

Other types of design projects are harder to convert into an exact ROI. What is the value of increased customer satisfaction? Of more traffic or increased use of a Web site's target features? These estimates vary between companies, and thus the monetary value of doubled usability also varies. However, it will be substantial in most cases.

Typically, the more people use a design, the bigger the usability ROI will be, since the benefits come from the added value that ease of use brings to each user. For an application with many users the benefits are greater than for an application with a limited user base. Typically, games, especially successful ones, have a very large and variable user base. This translates into making usability a high priority, since the benefits are substantial. Moreover, good usability leads to positive first impressions among customers when developers promote their games.

3.4 Context Issues

Context issues are especially relevant to mobile game applications, since the gameplay environment can vary substantially. Typically, mobile game sessions last less than ten minutes and take place when the user is waiting for something – for the bus to arrive, for a lecture to begin, etc. Even so, there is great variability: The game can be played outside or inside, under different lighting conditions, etc. In short, context is the essence of mobility.

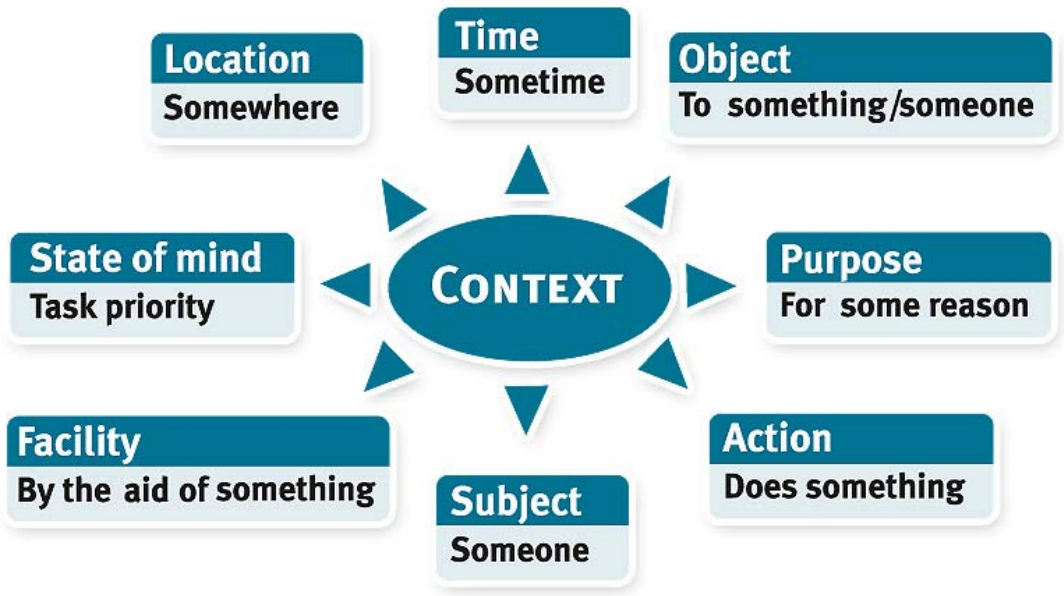


Figure 2: Components of context

Certain context features apply to more than one category. Network games are affected by almost all of the categories. Social factors and competitiveness affect the reasons why games are played and what meaning they have. All of this must have an effect on the way games are designed.

Currently, games are not context-aware. One important reason for this is that it is difficult for the Java application to determine the status of the phone. However, in the future, games will be more adaptable: Their behavior, navigation, and even rules may change if, for example, there are other people in the game, if the network is accessible, etc.

3.4.1 Purpose

With games, the most likely purpose is to have fun, perhaps with a social aspect attached to it. However, there may be several different underlying reasons: the intellectual challenge of a puzzle game, the reaction challenge of a fast action game, or the competitive challenge of playing with others.

According to a 2001 survey in the United States, 87 percent of frequent computer and video game players said the number one reason they play games is because it's fun. About 72 percent play because of the challenge, 42 percent for interaction with friends and family, and 36 percent because games provide a lot of entertainment value for the money.[15]

3.4.2 State of mind

Users may have different priorities at different times. The motivation to play a game might be stronger if there is a competitive element; if a user is playing simply to pass time, the game may be more easily interrupted. Also, the user may be doing something else simultaneously, such as watching TV.

3.4.3 Object

The object of the action is clear — it is the game being played. However, the platform where the game is played offers variation: Screen size, sound, keypad, memory, and technical issues differ from phone to phone.

3.4.4 Action

The specific action is playing the game, but games vary widely in terms of the actions they permit.

3.4.5 Facility

Some facilities must exist to play a game. Naturally, there must be a device to play on and a game to play. There must be enough power in the battery or a power supply. If the game is played outside and the weather is cold, the battery will run out much more quickly. If the game is intended to be played outdoors, this must be taken into account.

There must be a SIM card in the phone for it to work, and there must be a connection open to use the network. The light must not be so bright that it obscures the screen, and the ambient sound level can't be higher than the game sounds.

3.4.6 Subject

Who is playing the game? It is not enough to know the age, sex, or income of the player – and even these demographics are often guesses. Individual characteristics need to be taken into account. For

example, what is the user's vision like? How much red-green color blindness exists in this group? What are the culture and peer pressure? What is the attitude towards mobile games? What are the group's values? Do the users appreciate competitiveness? Are they social or withdrawn? Would they prefer a one-time payment for games, or an ongoing subscription fee? How much are they willing to pay and what genre of games do they prefer? If they are young, how do their parents feel about mobile phone games?

The user population is likely to be a partial surprise. For computer and video games, the average player age is 28 years old and almost half of those who buy games are women. They may, however, be purchasing games for their children or siblings, but even if that is the case, these women are making the decision to buy a game. Adults older than 18 years of age purchase 96 percent of all computer games.[16] This still leaves many unanswered questions about the persons playing the game, and many of these questions need not be answered. However, critical questions concerning a specific game must be asked and answered. And while the answers may not be difficult to find, often the relevant questions are.

It is almost certain that networked game play will increase, and this will probably affect play styles. Social issues are more clearly present on the network, a fact that may attract more females to games. On the other hand, strong competitive factors may attract males. It is important to remember that the user base is not a stable mass and should be monitored as to future changes and possibilities.

3.4.7 Location

Based on user interviews, the typical scenario for playing a mobile phone game is when users are waiting — for a bus, a lecture, etc. However, users may not remember or take into account all possible situations. Even if they don't play in certain places, it may not be because of the environment but, rather, because of the game. If the game features uses sounds that can't be turned off, then game play is limited to places where it is all right to make noise.

A small screen size may make it difficult to play in certain locations, or the correct keys may be hard to hit when the user is wearing gloves. A user may not be interested in playing a mobile phone game at a sports event, but if the game is about the same sport as the event, the user motivation may change. A variety of factors may affect the environment where the games are played.

The concept of the location awareness capability of phones is beginning to emerge, and it offers numerous possibilities — it means, for example, that games can be played only at certain places, or that the user must go to a certain location in order to finish a level. Integrated with network play, the whole picture of mobile game playing may be broadened and extended significantly.

3.4.8 Time

Play sessions tend to be short, no more than ten minutes. This means that the game must be easy to save or save automatically, and that the game story should probably not be long or complicated.

Time also refers to the time of day when playing takes place. Do users play mobile phone games during the day, or mostly before and after school/work?

4 Device-Related Issues

Developers can do little about some of the usability issues that emerged from the tests. However, there are tips that many help them cope with situations, and some possible solutions have been suggested. Nevertheless, the problems remain and need to be addressed in future versions of the phone system software.

4.1 Hardware

4.1.1 Navi-key

Users navigated with the Navi-key fairly naturally and without much thinking. It was less obvious to them that they could also press Navi-key; some users used the left soft key to open an application when the Navi-key could have been used.

Often users accidentally pushed up when they tried to press down the Navi-key, which made users request other possible action keys. It is unlikely that additional buttons would remedy the situation, but the size and feel of the Navi-key may need slight modifications. Action games are especially vulnerable to these hardware problems since there isn't much time to adjust precise finger movements.

There isn't much that developers can do about this situation. However, games should be designed so that there are more optional action keys than just the Navi-key.

4.1.2 End key

In tests conducted with Nokia 3650, eight users out of 32 (25 percent) pressed the End key (red phone key) by accident at least once during the test session. However, the problem is not as significant as on the Series 40 phones, because the game application is only paused, not shut down. On Series 40 phones, all of the test users pressed the End key accidentally at least once during tests.

According to the Nokia Series 60 UI style guide, the phone should return to the idle state when the user presses the End key within an application. The application should not be terminated. This is logical and clear behavior. Since the game application is still open, current progress is not lost.

Many players pressed the End key in order to get out of the game. The End key seems to be a good panic exit button because pressing it doesn’t shut down the game. In other words, it causes minimal interruption.

4.1.3 Application key

Seventeen out of 32 users (53 percent) pressed the Application key (Menu key) by accident during a game. This was a very common mistake and is probably due to the placement of the Application key in 3650 mobile phone.

Of those users who exited the game by accident, almost everyone managed to get back to the game. Users returned to the game by reopening it from the Applications menu, and they were pleasantly surprised to discover that their game performance had not been deleted; instead, they could continue where they left off.

4.2 Software

4.2.1 Application menu

If there are many J2ME applications installed, the Applications menu is noticeably slow to open, at times taking almost ten seconds. Although ten seconds is not a long time, it may lead to a decrease in playing since it is somewhat frustrating to open a Java game on a Series 60 phone.

4.2.2 Controls

In tests conducted with the Nokia 3650 phone, not a single user spontaneously tried to play the game with the number keys 2 (up), 3 (left), 4 (down), and * (right). In some games these controls may be more appropriate than the Navi-key, since they allow more room for two fingers. Also, with these controls it is not possible to press the action key by accident. Other published devices that are based on Series 60 Developer Platform 1.0 have more traditional number controls (2/4/6/8). For diagonal movement in these phones, the only reasonable controls are 1/3/7/9. When optimized to the Nokia 3650 phone, controls have to adjust differently.

The information of these alternative controls may need to be more visible to users, especially in games that might benefit from them. However, since the Navi-key is much more natural for users, enhancing its usability should be the first priority. For more information about key mappings, see *Guidelines for Game Developers Using Nokia MIDP Devices*[17].

4.2.3 Location of the games

All players had problems finding the Java games, which is no surprise because this was an issue for Series 40 phones. Java games are not located under Games, but under Applications; this is misleading since players commonly have no idea how games are implemented (Java or Symbian), nor do they care. In addition, the Application list looks different from the Games menu.

4.2.4 Sounds

Even though the phone profile is set to silent, MIDI sounds played by games are still audible. Furthermore, the Java application has no way of knowing the status of the profile. It seems the only way to make the phone truly silent is to make the profile silent and turn the sounds off in the game settings. The user will certainly be surprised to hear background music when the phone is silent – especially if the user is playing where sounds are forbidden, for example, at school. This issue is not going to be changed for existing developer platform 1.0 devices. Naturally the phone should be completely silent if the profile is silent.

One solution is to provide additional sound settings in the game application. However, there are problems here as well: If the profile is silent but the game sounds are on, only MIDI sounds will play, which means that the control should probably be called “music,” as that is usually what it means to the user.

The recommended solution is to set the game sounds off as the default. If the user prefers, s/he can change them from the Game menu. When the game begins, users can be asked to supply their preferred setting.

5 Guidelines

Implementing good playability through usability is no easy task. Since games are played mostly for fun and pleasure, they have particular requirements. Business applications are used for their ability to perform certain tasks, generally as quickly as possible, but that's not necessarily the case with games. Game players want to play successful games as long as possible – or get back to them as often as possible.

Thus, games must challenge the player, but not because they're hard to use. The challenge must arise from the game itself.

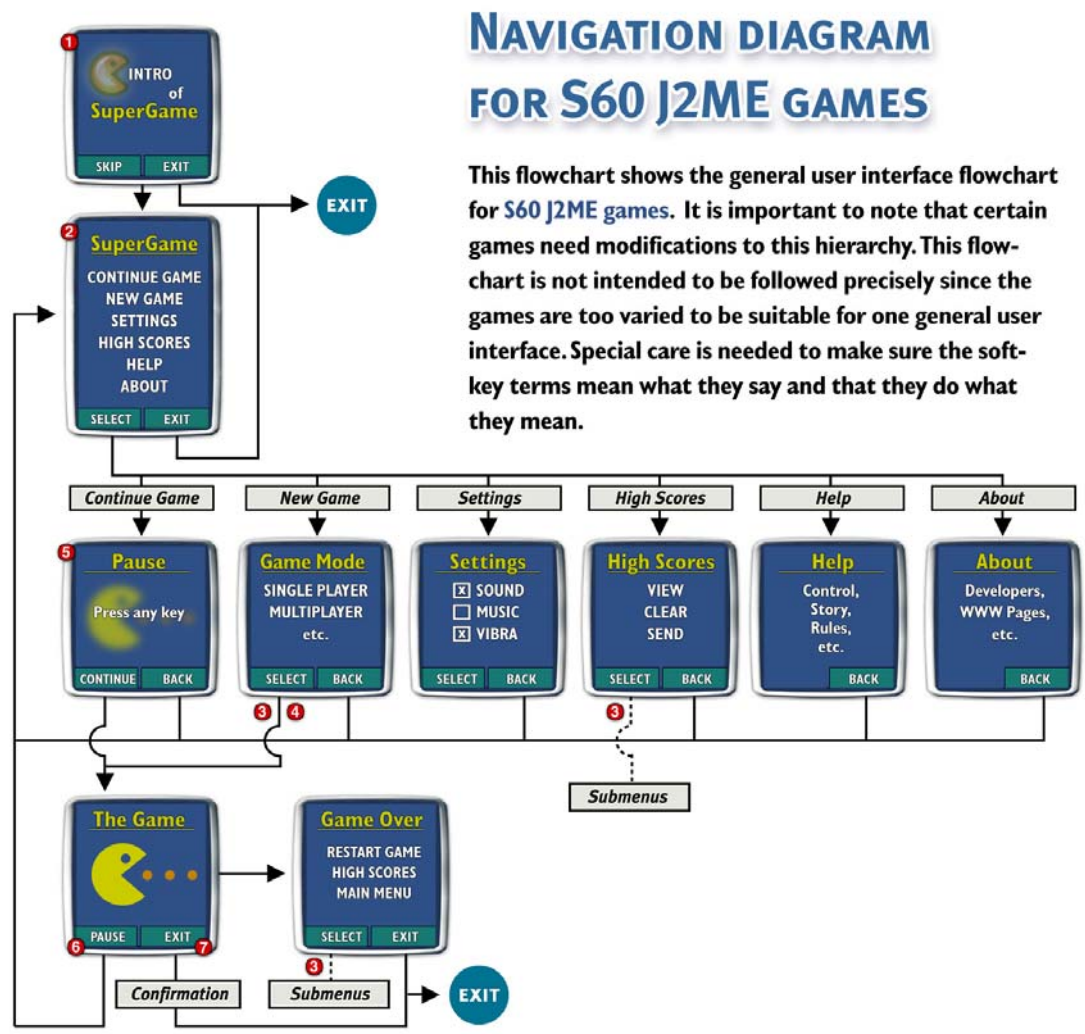
These guidelines will help developers design games that minimize time spent learning a game and maximize time spent playing. The intention is not to kill game-design creativity by forcing adherence to specific rules; on the contrary, the purpose is to allow developers to concentrate on gameplay rather than UI issues. A good user interface is not going to harm a good game, but a bad one may cripple an otherwise good game.

5.1 Navigation Diagram

Figure 3 shows a typical navigation scheme for a Series 60 J2ME game. Games vary widely, of course, and care must be taken to ensure that soft-key labels are appropriate and that screen titles and menu entries are descriptive.

Not all screens and buttons are required in every game, and each must be evaluated in context. Some games require more versatility from the user interface than shown in Figure 3. In particular, Intro, About, and Game Mode screens may not be appropriate for every game.

Note also that not every screen is depicted, for example, the screens for entering player names and the high-score list are not shown. The high-score list after the game is identical to the screen that is accessible via the Main menu, except that the Back button takes the user to the previous screen, not to the Main menu.



1. This screen is optional, but recommended.
2. In Pause mode the screen title may be adjusted accordingly.
3. For simplicity, the navigation flowchart is not drawn to its full depth.
4. Player names are asked for after the game mode has been selected. Game difficulty level is also implemented in this screen.
5. This screen is needed only for action-based games. Its purpose is to allow the player to adjust to the game situation and prepare for action. The left key may be shortened if Continue does not fit. Other options include Go, OK, Game, and Proceed.
6. The left soft key may be labelled Menu or Pause, depending on game type and context. Pause is recommended for action games, Menu for turn-based games. If a player selects Exit, s/he must be asked for confirmation or the game must auto-save. The left soft key may be labelled with a symbol, such as a triangle, if soft-key text is not displayed during the game.
7. Implementing the right soft key Exit is optional.

Figure 3: Navigation diagram for Series 60 J2ME games

5.2 Top Ten Guidelines

The most significant usability guidelines to emerge from Nokia's study of J2ME games for Series 60 phones are listed below. Developers should note that the usability tests were performed using a Nokia 3650, and may not apply to devices with different keypad configurations or system software.

This list should be read and understood by all who design games for this platform. Two additional caveats are paramount and must be remembered at all times during development: *Know the user* and *do not annoy the user*. The game experience is not measured by technical features or even usability as such; it can be described as the level of immersion experienced by the player. To pinpoint the differences, new guidelines and those with significant changes have been marked with a S60 symbol.

5.2.1 Pregame

Appropriate main menu implementation

S60 → *Select an appropriate main menu implementation; A game-like interface and background for the main menu are recommended.*

→ *Use a structured navigation diagram*

→ *Minimize number of intermediate screens*

→ *Prefer a wide hierarchy to a deep one, but avoid scrolling.*

5.2.2 Game experience

Simplicity is the key

→ *Make your game a minute to learn, a lifetime to master.*

→ *Often additional rules make games more complicated but not better.*

→ *It is often complicated to make things simple.*

→ *Provide a realistic physics model. Players should not be forced to learn new things when they can use prior knowledge. Implement a realistic physics model in relevant games (racing games, for example).*

Help where help is needed

→ *Provide in-game help. Place "information items" in the game.*

→ *Display the information item text only after user action (action button or button 8), not automatically. Experienced players are annoyed by the slow-down.*

→ *Make the information item stay on-screen so that the user can return to it later.*

→ *The in-game help feature should not stop the game. Players should be able to skip the help. Help text that pauses the game is annoying to advanced players.*

Provide challenge

→ *Provide a difficulty setting if applicable.*

→ *Do not make the game more difficult by altering constants, such as the physics model attributes. The challenge should come from difficult tasks.*

→ *Create as human-like AI as possible — reasonable but not completely predictable.*

→ *The challenge level needs to be near optimum. Too difficult means frustration, too easy means boredom.*

Use natural controls

S60 → *The left soft key is OK and Accept. It should also bring up a context-sensitive menu – usually the game main menu.*

S60 → *The right soft key should act as Back, Quit, Cancel, Clear, and Exit. Within the game, it should exit the game (with confirmation and possible auto-save).*

→ *The left and right soft keys should not do the same thing. In case of conflict, the right soft key should be disabled.*

→ *Games should be designed in such a way that they do not require diagonal movement impossible with navigation key.*

→ *If diagonal movement is needed, aid the user. For example, in racing games it is probably not necessary to keep the accelerator down all the time since it is the default situation, so it could be made automatic.*

Provide pause and save game

S60 → *Provide pause game capability.*

→ *Provide save game capability (except for very short games.)*

Easy on the sounds

S60 → *Prefer a silent start-up to one with music.*

S60 → *Provide a sound menu in the game to make sure the player can deactivate or activate sounds.*

S60 → *Background music should be off by default.*

Give control to the user

→ *Users should get feedback on their actions.*

→ *User should not be forced to perform a specific procedure in a certain order.*

→ *Do not leave users in situations where they can't affect their destiny in any real way.*

→ *Let the user get prepared, particularly when starting the game or continuing from a pause, but also within the game itself.*

S60 → *The game should go into pause whenever the player puts the game application in the background.*

5.2.3 Post game

High score, a souvenir of success

→ *Provide a high score list. Provide preset results so that getting on the list feels like reward.*

→ *Do not require user to enter a name. Offer genre-specific default names and remember last entries.*

→ *As an interesting addition, developers should consider providing a method of sharing game success, for example sending an SMS or MMS with the game name, score, and text to a service provider or a friend.*

Easy restart

→ *Provide an easy way to restart the game. For example, in the Game Over screen, provide a "Replay", "Play again," or "Restart" command, taking the user back to the beginning of the game, with the same settings as before.*

5.3 Detailed Guidelines

The Game Usability Guidelines are described below in detail. Many of these are similar to the guidelines in the *Series 40 Developer Platform 1.0: Usability Guidelines For J2ME™ Games* document.

5.3.1 Pregame

5.3.1.1 Preparing for the game

Be consistent

Computer games are no longer a new item, and certain standards have evolved. These conventions need to be taken into account when designing new games.

→ *When designing a game user interface, developers should understand the user interfaces of similar games, both on phones and other platforms. However, they should not just copy the UI. In case of conflicts, developers can refer to this document.*

→ *Everything in a game must serve the same purpose. For example, if the game is a Formula 1 racing game, terms that support this picture should be used, not terms from a rally game.*

Load in reasonable time

Speed is generally the most important single usability criterion. Slow responsiveness affects the user experience severely and negatively. If the game takes too long to launch, it annoys the user, and because mobile games are played mostly while waiting, the user might not find it worth the trouble.

→ *Make the game launch in reasonable time, the faster the better. The game main screen should open in five seconds or less, and preferably in three seconds. The same goes for starting the game from the Main menu.*

→ *To reduce the perceived time to launch, make something happen on the screen. Even better, give the user something to do while the game loads in the background.*

Make it possible to skip the intro

A game introduction is useful if the game takes a while to load and it imparts hints for play. But users find it annoying when forced to sit through the same intro every time.

→ *Make it possible to skip the intro. Preferred keys are the Navi-key, number keys, and left soft key. The right soft key should cancel game loading. The End key and the Application key should work normally. Provide instructions, such as "Press 5 to skip intro."*

→ *The intro splash screens should not be displayed for an extended times: two to three seconds per screen.*

S60 Reasonable default settings

If the game has intro music, this will annoy others if the user starts it up in a location where silence is preferred, and thus the user may choose not to play. In Nokia's study, most users did not comment about game sound one way or the other, and when they did, comments were almost always negative. Developers should make sure that the sounds do not annoy the user. MIDI sounds are played even if the profile is set to silent. Please see Section 4.2.4, "Sounds," for details.

→ *Intro music in MIDI format should not be used, because they are played even if profile is set to silent. But if it is implemented for some reason, there must be a way to switch it off. However, it is difficult to provide a switch in the intro screen. The setting can be in the Settings dialog, but the default should be "off". The sound should not startle the user – they shouldn't appear where the user doesn't expect them.*

→ *In the Settings dialog, provide a way to disable the vibra functionality, if the game uses it.*

→ *Use checkboxes in the Settings dialog:* [X] Sound. *If this is not possible, use* Sound: On *format.*

Do not force the player to restart from scratch

If the game is level-based, it should not require the user to replay levels if no additional challenge is provided. The user should be able to continue from his/her previous level, but should also be able to restart from level 1.

→ *Provide a way to skip levels the player has already completed, if relevant.*

5.3.1.2 Efficiency

S60 Enable efficiency through shortcuts

Efficiency must be assessed from the user's point of view. If saving a game takes ten seconds but the user can continue playing at the same time, this is better than taking five seconds to play during which time the user must wait. The latter may be technically faster, but the user doesn't perceive it that way.

→ *If possible, time-consuming operations should be done in the background while the user does something else.*

→ *In order to make restarting easier, provide an easy shortcut at the end of the game, e.g., "Try again," "Replay," etc. If there are settings to be altered, this shortcut should skip them and use the same ones the player used in the previous game.*

→ *Provide a consistent method of going back. Use the right soft key.*

→ *The player must be involved in the game quickly. It is crucial to get the player in the game and arouse his interest. This means that there should be no complicated settings to adjust before playing, or a "quick start" option.*

Conserve the user's time

Time spent navigating and configuring the game should be kept to a minimum, and time playing to a maximum. Users should understand what they must do in the game.

→ *Whenever possible, the game must provide reasonable default values for data asked from the user. If the user has previously entered data, that value should be used as the default. If no data has been entered, provide a reasonable value, or zeroes for numerical data to indicate the data format. However, defaults should not be provided if no purpose is served. For example, there is no point in providing "First name Surname" as default values in a player name field – this does not conserve the user's time since s/he must erase the old text anyway. It does make sense to provide famous athlete names in a sports game.*

→ *The game should remember what the user has done and not force him to do it again. For example, if the game challenge comes primarily from solving puzzles on levels, the game should not require the user to play these levels every time if there is no additional challenge. The user should be able to continue on the level that s/he has achieved, but should also be able to restart from level 1 if s/he chooses.*

→ *The game must preserve all the data the user has entered. This applies to the name s/he gives, the options s/he selects before playing, and the options s/he selects during the game.*

Efficient text entry fields

Text entry screens should follow the user's expectations, asking for information that users expect, and in the order they expect it. Users should always understand what is being asked, why, and in what format data should be entered.

→ *Fields in forms should be grouped logically and presented in a natural sequence.*

→ *Do not force the user to enter a name at all, especially a name of certain length.*

→ *Ask for name entry only when needed, not before.*

→ *Provide the name entered last as the default so it can be accepted quickly.*

→ *The application must not ask for data that it can reliably find on its own.*

→ *Forms must be consistent with each other, the application, and the UI of the phone. The terms used, command and option names, and screen layout must be familiar to the user, based on the user's experience with the phone and the current game.*

→ *Form fields must indicate the format of the data requested. If the number of choices is limited, they should be presented in a drop-down menu or a similar structure that allows selection from a limited, predefined list.*

→ *If the user enters the data in the wrong format, the game should correct the user as much as possible. For example, if the user uses a comma as a thousands separator but the game expects a space, the string could be corrected automatically. However, this can be done only if it is certain that the entered data is inappropriately formatted. It is a good idea to inform the user about the correction.*

5.3.1.3 Navigation

S60 Appropriate menu implementation

There are three alternative ways to present the game's menu:

1) Using only the main pane of the screen for the game. This means that the phone system software shows the icons and application name at the top pane, and the bottom pane shows the soft-key labels. If this method is used, the left soft-key menu "pops up" from the lower-left corner, just like in other Series 60 applications. This is familiar to the user and conveniently leaves the game application dimmed in the background. As a downside, this method provides the smallest usable screen size available for the game itself.

If the full screen is used, the content of the main menu may be displayed in two different ways:

2) The menu may be shown in the normal Series 60 font and outlook. This is like the menu that pops up from the left, but this menu remains visible as long as the user is in the main menu screen. The menu fills the whole screen and is not transparent. As with the pop-up menus, the system software provides an Exit command in the left soft-key menu, which cannot be removed from there. Thus, the left soft key cannot be labeled "Select," even if there is only the Select command provided by the application, since in that case there would be Select and Exit. To select a command, the user must press the left soft key (preferably labeled Options) and choose Select or use the Navi-key. If there is an Exit command in the menu and the user selects it, the menu that appears is somewhat confusing. There is "Select," which selects the Exit command and therefore Exits the game; and there is "Exit," which also exits the game. By providing the Exit command in the main menu, the user is more likely to regard the pop-up menu Exit command as "Exit the menu," etc. If the developer defines a Command.EXIT command, it is mapped to the right soft key. This is done automatically and cannot be removed by the game software.

3) Finally, developers can create the entire menu system from scratch. This provides the most flexibility, but is also the easiest way to make mistakes and requires the most memory. Using this implementation will make the application different from the rest of the phone, which may be either positive or negative. These games typically take longer to load and while loading the left soft key has an Options menu with just one command, Exit, provided by the phone system software. If the loading takes too long, the user may get bored and quit the game without even reaching the main menu.

→ *Methods #1 or #3 are recommended. There are limitations in both, but good usability is possible. Method #2 poses too many restrictions that are hard to overcome.*

→ *Implementation #3 provides the best results, but requires the most from the developer. If this implementation is used, the left soft key should be simple Select in the menu structure, the right should cancel/go back, etc.*

→ *If method #1 is used, there is no separate main menu screen. There may be a background picture, but the menu itself "pops up" from the left soft key.*

→ *However, if the game uses its own menus, there must be an Exit command. The left soft key should select the command, but the right soft key should be an active Exit, too. It is not a problem to provide two Exits; it would be worse to have no visible Exit at all.*

→ *The menus should look like part of the game, instead of part of the phone. The application should be recognizable by the menu background colors, fonts, etc. However, when implementing this, developers should conserve phone memory, using the same graphics as in the game itself, or using only coloring instead of different graphics. They should implement this only if it can be done well – it should work at least as well as the higher-level UI. Also, using image collections instead of separate images will help conserve heap memory usage.*

S60 Main menu

The Main menu must provide certain familiar commands to the user, as follows. When adding additional commands, scrolling should be avoided. The name of the game should be used as the Main menu title. Note that not all of these commands are applicable. For example, if the application uses #1 type menu, there is no sense in providing Continue game when the game is active behind the menu.

→ *Continue game. Available only if a game is paused or only one game can be saved at a time (recommended). The saved game is loaded (or a paused game continued). The user is taken to the game, but the game is paused and the text "Paused. Press any key to continue." is on the screen. After the user presses a key, the game continues. If #2 type main menu is implemented, this command should be "Pause" when the game is in progress.*

→ *Save game. Available only if several games can be saved (not recommended). Saves the current game.*

→ *New game. Starts a new game, or if there are multiple game modes, provides a menu to select from them.*

→ *Saved games. Available only if several games can be saved (not recommended). Brings up a list of saved games. The right soft key takes the user back to the Main menu. The left soft key brings up a sub-menu listing Continue game, Delete, and possibly Rename. When loading, the user is taken to the game, but the game is paused and the text "Paused. Press any key to continue." is on the screen. After the user presses a key, the game continues.*

→ *Settings. If there are no more than two settings, they can be integrated in the Main menu. The format* "Sound: On" *should be used instead of* "Set sound off."

→ *High scores. Available only if there is a point in keeping track of high scores. The high-score list is sorted from the best to the worst, best scores first.*

→ *Help. Instructions for the game. Concentrate on the controls; be very brief about the story, if it is mentioned at all. The rules and purpose of the game should be covered. The help text should scroll more than one line at a time with a single key press.*

→ *About. Copyright, version, and vendor information about the game. This is optional.*

Menus and lists

Menus must be consistent with each other and clear to the user. The user must know where s/he is at all times and how to navigate around.

→ *The title of each screen should describe the contents of the screen. Do not use "Please select" or "Main menu" as the title.*

→ *Menu commands should be listed by decreasing frequency of use. However, keep items that belong together close to each other, such as Open and Save. Also, commands should appear in a logical order – if certain things are generally done before others, they should be listed first in the menu.*

→ *If there are no items in a list, do not show a blank screen; instead, show "No items" or "No high scores," etc.*

→ *No command should be listed in a menu more than once, not even under different names.*

Logical information hierarchy

When the user explores the application, s/he can get lost easily. Each screen is a new cognitive and visual challenge and provides more things to remember.

→ *See 5.1,* "Navigation Diagram," *for a navigation diagram.*

- → *Use a single Main menu accessible with the left soft key. Note that when navigating inside the menu structure of the game, the left soft key should act as an OK button, or open a sub-menu specific to the higher-level menu item selected (if appropriate). The right soft key should be used to return to the previous screen.*
- → *It is optional but recommended for the Main menu to look like a game menu. Use a different font, different background color, etc., but make the user feel like s/he is inside a game, not in the normal menu structure. However, game-looking menus should only be implemented if they do not waste memory and are proven (tested) to work, both technically and from the user's perspective.*
- → *The Main menu should be short. Users do not like to scroll; do not force them to. All controls should be visible, and a deeper menu structure used if necessary to avoid scrolling.*
- → *The number of screens should be minimal. Do not provide eye candy without any real content.*
- → *A wide hierarchy is preferred over a deep one. Use more items at each menu level rather than a few items in a deeper structure. However, make sure that the user does not need to scroll the screen in order to reach a command.*
- → *Every screen (except zoom/playing screens) should contains information about at least one of the following: status of the application, the content of the window, indications about the actions to perform, or the title of the application.*
- → *Abbreviations should not be used (except for very well-known ones, such as CD or GSM,) and information should not be squeezed.*
- → *The general hierarchy of information is from the general to the specific.*

S60 Consistent soft key functions

Naming conventions for the soft keys, as well as their functions, should be consistent.

- → *Use Options or Select as the label for left soft keys in game menus. Inside the game itself, use Menu as the label. The label should be displayed in FullCanvas mode.*
- → *Use Back, Quit, Exit, Cancel, or Clear as the label for the right soft key. Within the game itself, use Back as the label. However, implementing the right soft key in the game itself is optional. The label should be displayed in FullCanvas mode.*
- → *The Navi-key button should be used as a default action button. For menus, it is usually select. Inside the game, it might be fire, place a piece on a board/flip it/move it, etc.*

5.3.1.4 Make the user comfortable

Respect the user's privacy

Generally users do not know if the data they enter is being sent over the network, where it is stored (phone memory or SIM), etc. Accordingly, they hesitate to enter personal information, such as an e-mail address. When personal information is requested, the reason should be clear to the user.

- → *Do not ask for (personal) information that is not required, and make sure the user knows why sensitive data is being requested.*
- → *Store sensitive data in one place only, and the user should be allowed to erase his/her data.*

Do not annoy the user

Typically, games are played in locations where it is not suitable or socially acceptable to have the sound on.

- → *Do not abuse vibration functionality. The user must be able to switch the vibration feature off and the game must be playable without it. The application should not use vibration too much, and if it is used, it should mean a specific thing.*
- → *The user must be able to switch sound and vibration on and off independently of each other.*
- → *The application should be reasonably fast. The perception of speed can be increased by performing actions in the background. Even short delays decrease a user's willingness to play a game.*

5.3.2 Game experience

5.3.2.1 Controls

S60 Soft keys

Soft key names and functionality should be consistent.

- → *Use Options or Select as the label for the left soft key, according to its function.*
- → *Use Back, Quit, Exit, Cancel, or Clear as the label for the right soft key, according to its function.*
- → *Use labels in FullCanvas mode, too.*
- → *Inside the game itself, Menu should be used as the left soft-key label; Back for the right. Implementing the right soft key is optional. The "Menu" text may be replaced with a symbol, such as a triangle.*

S60 Controls

Providing clear and readily understandable controls is vital to playability. There is little reason to allow the user to modify controls, as s/he is unlikely to be a UI specialist or take advantage of such a feature.

- → *The left soft key should act as OK and Accept. It should also bring up a context-sensitive menu – usually the game main menu, but submenus might have different options.*
- → *The right soft key should act as Back, Quit, Cancel, Clear, and Exit. Inside the menu structure, the right soft key should go back (one level up). If the user presses the right soft key inside the game itself, it should make the game quit (with confirmation) or take the user into the main menu and pause the game.*
- → *If a game is in progress, the right soft key should quit the game and go to the Main menu, but only with confirmation. In the confirmation dialog, the left soft key should be OK ("Go ahead, quit") and the right soft key Cancel ("No, don't quit"). It is optional to implement the right soft key within the game.*
- → *The left and right soft keys should not do the same thing. In case of conflict, the right soft key should be disabled.*
- → *In general, the Send key can be used as a shortcut for OK.*
- → *The End key should take the user out of the game, but not close the application.*
- → *Navi-key should be used for movement; number keys should be enabled, too.*
- → *The Navi-key should be the main action button, as along with button 5.*
- → *The controls in the game must be applicable to the entire game. Users generally do not remember if there are different buttons for different game modes and different parts of the game.*
- → *If more complicated controls are needed, they must be used consistently. For example, if two different jumps are needed, the same key (2) can be used; the length of the jump is determined by how long the key is pressed.*
- → *Typically users are not using the number keys for movement, but they should be enabled and not used for anything else. Using standard methods makes it likely that the game will work on future devices, too.*

Simultaneous button presses

Series 60 phones do not detect two buttons pressed simultaneously — the button pressed later does nothing. If the first button is released, the button pressed later takes effect. Players often do not understand or expect this behavior.

- → *Do not design games that encourage the user to press two buttons at the same time. For example, in a racing game, left could be designated to mean "accelerate and turn left".*

S60 Key functions

Users use the Navi-key fairly well and naturally, so all games must be controllable with it. However, due to the size and shape of the key, users occasionally push up when they try to press the key down. For this reason, other controls must be possible, too.

- → *Enable Navi-key for controlling the game.*
- → *Enable numeric keys for controls, even though they currently they are seldom used.*
- → *These keys need not literally move the character. For example, if the character can throw something upward, the 2 key can be used for this feature, especially if the 5 key is used for something else.*
- → *It is not required to be able to do everything inside a game at any time. Often it provides additional challenge to simplify the controls. For example, Instead of allowing use of every item with specific key, there could be a key to switch the active item and then the action key could be used to use each item.*

S60 Diagonal movement

Currently, true diagonal movement is not possible on Series 60 phones. This poses problems in games that could benefit from this feature, especially since users are typically unaware of this limitation. Informing users is unlikely to help, since not all of them read the help information; even if they do, diagonal movement is such a natural thing, they might try it anyway.

- → *Games should be designed in such a way that they do not require diagonal movement.*
- → *If diagonal movement is needed, developers should help the user. For example, in racing games it is probably not necessary to keep the accelerator down all the time since it is the default situation, so it could be automatic. Holding down "left" would mean "left and forward." The car must not slow down in corners because the user cannot accelerate due to technical limitations.*
- → *For level jumping games, a jump could be diagonal if the character faces either way, left or right. Also, if there is a ledge in that direction, the character could jump on top of it, even though the player jumped straight up.*

5.3.2.2 Functionality

S60 Pause and save game

Since mobile games are played mostly while the user is waiting, they are often interrupted. This means that there must be a pause and save game features to make it possible to continue an interrupted game.

- → *Enable saved games. The current game should be saved automatically if the user exits before the game is over, as in Snake. Other possibilities are: Save at a certain location, save at the end of a level, and save multiple games at a time. These are not recommended because they make the user interface much more complicated and take up more space.*
- → *If only one game can be saved at a time (recommended), the first command of the Main menu should be "Continue game" when a game is saved. If multiple games can be saved at a time (not recommended), the command is "Saved games," which takes the user to the list of currently saved games. If an automatic save is performed, that game should be the first one in the list.*
- → *If the player exits the game with the End key, the application should not close, but stay open in the background.*
- → *If possible, the game should go into pause mode for any of the following: an incoming call, a received vCard, a clock alarm, a timer alarm, a calendar alarm, an incoming message, an incoming infrared and/or Bluetooth transmission, the player switches to another application, or when changing batteries.*
- → *When in pause mode, the game should go into its Main menu where the first command is "Continue game." Thus, the pause mode is identical to left-soft-key use while playing.*
- → *Application should not change focus from screen to another within hideNotify() method. (e.g. Display.setCurrent(Displayable d)) In Series 60, this causes focus to be regained by the game*

application. From users point of view, it looks like pressing End key or Application key leads to menu, which is not desirable effect.

S60 The End key

According to the *Nokia Series 60 UI Style Guide*, the phone should return to the idle state when the user presses the End key within an application. The application should not be terminated. This is logical and clear behavior. Since the game application is still open, the current progress is not lost.

→ *If the user presses the End key in the middle of a game, take the user out of the game, pause it and move the game application to the background.*

One key for one function

If there are several keys for one function, users may be uncertain whether or not the functionality is indeed the same. If there are several functions for a single key, the user may become confused, since the same key does different things at different times or when operated differently.

→ *There should be only one function associated with each key and one key for each function. However, if the functionalities perceived by the user are close to each other, the same key can be used, e.g., the right soft key can mean Cancel or Back, depending on the context.*

5.3.2.3 Game world match with real world

No invisible barriers

If there are sections of the game area or level that the player should not access, s/he must be prevented from going there. Areas that are inaccessible must look different from areas that are normally accessible.

→ *There should be no invisible walls or invisible barriers. If something is inaccessible, it should not appear to be accessible, otherwise users will get frustrated.*

Prepressed keys

If a game has animations or loading screens during which the player can't do anything, the keypad should be disabled while these are happening. If the user presses a key and the game remembers it, that key takes effect when the game starts and most likely that game session is ruined.

→ *Disable prepressed keys.*

Bonus and special levels and features

Special levels are wonderful features that bring more variety to play. They also increase the replay ability of the game. However, if the game rules for these levels are different, players must understand this.

→ *Bonus and special levels must look distinctively different from normal levels.*

→ *If player actions can determine when a special level is presented (for example, by finding a treasure or gaining a certain score), the special level must begin right when the goal is achieved, not after the current level. There must be a clear connection between what the user does and what happens as a result.*

→ *If a bonus level starts after the current level, it should be especially clear that the level is different; this can be accomplished by using in-game help text, a different background, different sounds, etc. It is better, however, to open bonus levels immediately after the user finds the way to that level.*

Getting killed

If it is possible to die in a game, and particularly if the death comes as a surprise to the user, s/he must have a short period of immunity, where s/he cannot die again and has time to recover and get his/her bearings.

→ *The player should be invulnerable for a moment after dying. This can be accomplished by taking the user back to a safe place, shielding the user to prevent a hit, or taking the user back to the beginning of the level.*

Provide feedback

The player needs to know how s/he is doing and what s/he has accomplished. Part of game fun is derived from knowing how well the user has performed and what s/he needs to improve.

→ *Provide clear feedback on essential elements in the game: when a level is completed, when a bonus level is reached, when the player succeeds (scores a goal or kills an enemy), etc.*

→ *There must be some visual indicator for everything essential that happens in the game, e.g., the user should be informed if a level ends when s/he gains a new skill, etc.*

→ *Players must be able to identify their status in the game at all times.*

No prior knowledge required

The game must be playable without the user having prior knowledge of how it works. In theory, this means that a game could be solved and a high score reached in the first play session; in practice, this shouldn't be the case or the game will lack challenge. Game success must not depend on knowledge of things about to happen in a game, such as where a treasure is located. Of course there can be secrets and surprises within a game, but developers should not annoy users by killing them when there is nothing they can do about it. For example, users should not be forced to take a 50/50 chance in which the wrong selection kills them, or know what will happen next in order to make it possible to clear that area of a game.

→ *Game challenge must not come from knowledge that a player must possess prior to playing.*

Realistic physics model

The rules of everyday physics are understood intuitively. Although people often make mistakes in assessing the trajectories of objects, they are fairly good at estimating whether or not the movement of an object is believable. This implicit knowledge should be exploited, not violated. Applying a realistic physics model to a game makes it feel more familiar and predictable.

→ *Do not force the player to learn new things when s/he can use prior knowledge. A realistic physics model should be implemented in relevant games (racing games, for example).*

S60 Status

The user must always understand his/her current status. In particular, critical information concerning such topics as game character's health, ammunition, or money, must be conveyed to the user clearly and without risk of misinterpretation.

→ *Determine the most important information for players and display it clearly on the screen, making sure there is not too much information – one or two status indicators are plenty for most games. If this is not sufficient, the design may need to be refined to allow a simpler interface.*

→ *When the user opens the menu from which a game application is opened, a Details command is listed that typically shows the vendor and the name of the game. More information can be provided here, such as a very brief description and a Web page.*

S60 Give control to the user

The user needs to feel s/he is in control of the situation. The application must adapt to the user's way of doing things, not vice versa.

→ *Give the user feedback on his/her actions. Click sounds and effects should begin within 50 ms of the action. If the action takes between 0.5 to 2 seconds, provide an indication that something is happening. If the expected pause is longer than 2 seconds, the user needs to know how much longer s/he must wait and, if possible, s/he should be able to switch to something else while waiting, or cancel the current action.*

→ *The user should not be forced to perform a specific procedure in a certain order. This applies to the settings as well as to the game itself. Well-designed games allow the player to reach his/her goal in many different ways.*

→ *The user should not be left in a situation where s/he cannot affect his/her destiny in any real way. It is frustrating to fail and not be able to do anything about it.*

→ *Let the user get prepared, not in such a way that it ruins the game or provides no challenge, but so that the user can survive. In particular, this applies to game startup and continuing from a pause, but also within the game itself.*

S60 Challenge

In addition to being easy to use, games must be challenging enough for advanced users, but easy enough for beginners to stay motivated.

→ *Provide a difficulty setting if applicable.*

→ *Do not make the game more difficult by altering constants, such as the physics model attributes. The challenge should come from more challenging tasks. By altering the constants, developers force players to learn a new set of rules, and once they have learned them, the game is likely to be just as easy or hard as before; all that has been gained is an extended learning curve.*

→ *The AI of the games must meet several challenges. First, it must be reasonable and act pretty much as a human would. It must be challenging for experienced players and there should not be one single strategy that always wins.*

→ *The challenge level needs to be near optimum. Too difficult means frustration, too easy means boredom. Since players have different abilities, the game should either adjust difficulty automatically or provide a setting to do so.*

5.3.2.4 Sounds

S60 Profiles

MIDI sounds played by games are audible, even the phone profile has been set to silent. It seems that the only way to make the phone truly silent is to make the profile silent and turn the sounds off in the game settings. This is a known issue and makes adjusting sound a bit complicated. The user is certain to be surprised to hear background music when the phone is silent – especially if the game is being played in a public place, like in a church or school. The Java application has no way of knowing the status of the profile.

→ *Provide a sound menu in the game to make sure the player can deactivate sounds.*

→ *If game has background music, it should be off by default.*

Use unique sounds

The sounds of a game must be different from the sounds of the phone, as well as distinctive from each other. The user must be able to differentiate among the sounds, even without seeing the screen.

→ *The sounds should be unique – they must not be confused with each other.*

→ *The sounds must not sound like ring tones, SMS tones, etc.*

Do not rely solely on sounds

Very often mobile games are played in situations where sounds must be turned off. The game cannot assume that the user has the sound on.

→ *Make the game playable and understandable without sounds.*

Relay information

The auditory channel is an excellent way to relay information about things that are outside the user's attention. The user should be aware of what goes on in the game, which translates into using sound as a way of communicating with the user.

→ *Sounds should convey information, and can be used as an additional channel to relay information to the player.*

No sound abuse

Most players think that sound in mobile games is nice but not necessary. In our study, users rarely commented on a game's sound, and when they did, the comments were almost always negative. It is important not to annoy users with too many and loud sounds.

→ *Keep the sound volume close to the volume of the phone's regular sound — not too quiet, but not much louder, either.*

→ *Sounds should be consistent with what happens in the game – sad events should produce sad sounds.*

→ *Sounds must not be too loud or too high-pitched, and users must be able to turn the sounds off easily. The game must be playable without sound and there must be a visual signal for each event that the sounds symbolize.*

→ *Do not enable startup music since the game may be played in a place where sound is inappropriate.*

5.3.2.5 Language

Users' language and terms

The user must understand what the application is trying to say. Use natural language, not techno-babble. When feasible, it should be localized to the user's native tongue.

→ *What is the end user's native tongue? The game should use that language, ideally via the phone's language settings. If there are separate versions with different language string, it is more difficult to keep updated and more complicated for the player. If the .jar size permits it, developers should include all the relevant localization strings in the original package.*

→ *Users should understand the terms used.*

→ *Acronyms and difficult terminology should be avoided.*

→ *Text should not be truncated.*

→ *Use consistent terminology, and use the same names throughout the application. Be consistent with the phone UI, too – and make the terminology consistent with the phone UI.*

No foul language or offensive terms

Some users are offended by foul language and its meaning may not be fully conveyed to the user.

→ *Swearing or foul language should not be used.*

→ *Use neutral, not offensive words.*

→ *Users should be treated well, and with respect. They should not be blamed for errors; rather, users should be helped to recover from them.*

Minimize the amount of text

A mobile phone screen is small and it is frustrating to read long passages of text on it.

→ *The amount of text in games should be minimal.*

Only the first letter is capitalized

Capitalized text is more difficult to read and understand.

→ *Text should not be in ALL CAPS.*

→ *Text must be grammatically correct*

Images do not substitute for essential text

Images are typically more difficult to perceive than single words, and are more likely to be misinterpreted.

→ *Images should not substitute for essential text.*

5.3.2.6 Graphics

Harmonious colors

Although a color screen can be very informative, it should not be overused. Colors that have special meaning, such as a color highlight for a selected item, must be easily recognizable.

→ *A consistent coloring scheme within the game and the navigational structure should be used. Things that look similar should behave similarly.*

Match functionality with appearance

Graphics need to be distinguishable from each other. If the player is not looking directly at an animated object, the animation needs to be more visible to ensure that the player sees it.

→ *Different items must appear clearly different. Items should be truly distinctive and animations sufficiently unique. In particular, different enemy characters must be unique.*

→ *The objects' and characters' appearance must match their functionality/activity. It should be possible to make a reasonable guess about a character's purpose without seeing it move.*

Use each screen fully

Ideally, developers should use the full screen, without leaving blank areas. If blank areas exist, images can be centered in the screen.

→ *Balance graphics and blank areas evenly on the screen*

The background must not look bland or too busy

The background image of the game must not be confused with the game objects and characters. The background is not there just to be pretty – advanced players can use it to time or align actions and events.

→ *The background should be different from the foreground; Make its objects less detailed and less colorful. It should look like "background."*

Backlight

If the backlight goes off during a game, it is difficult for a player to see what to do. If the user presses a key to get the light back, the application will probably react to it just like any other key press, and it might affect the game. Also, many phones turn the light off after 15 seconds of idle time. That's a pretty short time for some games.

→ *There is no way for a MIDP 1.0 application to know the status of the backlight, but it is possible to turn it on with DeviceControl.setLights. The game should turn the light on every 14 seconds, so that it never goes off during game play. Approximately a minute should go by before the phone turns the light off. If the application knows the light status, developers should have the game pause when the backlight turns off.*

5.3.2.7 Help

Use pictures

Help text typically describes the game controls and possibly the story. An illustration depicting the effects of different keys is a better way to explain game play than plain text; plain text requires more effort from the user to understand.

→ *Use a picture to explain controls, if possible. This works particularly well for movement controls. However, the pictures should not show the keypad and its buttons, but a character from the game and arrows depicting its movement.*

Concentrate on the controls

Even though many players claim they read help text, most do not — until they encounter a problem. Even when they do read the text, they typically remember only what they were looking for – most often, the controls.

→ *Help text should be brief. Provide help on the controls first, as this is the first thing players look for. The game story can be skipped altogether or mentioned very briefly. If it is mentioned, it should be after the controls have been described. The game idea must be described in four sentences or less.*

→ *The number keys should be the primary controls: 4=left, 6=right, etc. However, the arrow keys should function, too.*

→ *Since not all phones support simultaneous key presses, be sure to mention diagonal movement, if it is available (1, 3, 7, and 9).*

→ *The total number of controls should be relatively low.*

S60 Provide in-game help

Users do not typically remember help text. It is much better to provide information to users when they need it. If it is relevant to their situation, it is more likely they'll remember it.

→ *Provide in-game help to the user. Place "information items" in the game. When the player walks over these items, text appears. If the text is very short, it can be displayed in full, such as "Thieves, look out." For longer text, display a text that tells the user how to access the text, e.g., "Press 8 to read note." This way, beginners can access help, but it won't bother experienced players.*

→ *There might not need to be a special visible item for help, although it is recommended. The short text can appear automatically when the player encounters the first enemy of a certain kind, for example. It is important to keep in mind that the help text must not interfere with the play of experienced players.*

→ *Text should also appear when players encounter new items or need new controls that they haven't used before. Examples of in-game help include "Press 3 to jump," "Press 8 to dive," "Take the treasure."*

→ *The in-game help system can also impart some of the game's story by, for example, telling the player what must be done next. However, if this is necessary, it is possible that the story is obscure and not immediately understandable by the player. It may be preferable to simplify the game story.*

→ *In-game help can also be provided for movement, especially if it is not evident that the player can move. The first time the character is visible, show arrows around him and corresponding controls beside the arrows.*

→ *The in-game help feature should not stop the game. The player should be able to skip the help should s/he so choose. Help text that pauses the game is annoying to advanced players.*

→ *Controls should appear near the object they are controlling. For example, arrows with 4 and 6 could appear beside the player's character, not at the top or the bottom of the screen.*

→ *It may be necessary to have a special training level where the functions of the game are learned through play. If so, help text should not appear in the real game, just in the training mode.*

5.3.3 Post game

High-score list

When a player reaches the high-score list, s/he should be notified. However, not all players care about this. Therefore, the player should be asked what to do.

→ *If the score is good enough to reach the high-score list, the game should say so, for example, "Congratulations! You've reached 2nd place in the high-score list." The left soft key should take the user to a screen where s/he can enter his/her name. If the player chooses not to do this, the right soft key Cancel should take him/her to the Main menu.*

→ *No more than ten high scores should be listed.*

→ *Provide some predetermined values for the high-score list. This way the user receives feedback on how good s/he really is — it adds to the reward value of the high-score list. Entering a name for bad score feels like punishment, not like reward.*

→ *Do not require the user to enter a name.*

- → *A name of certain length should not be required.*
- → *The name entered last should be the default value for the name entry.*
- → *As an interesting addition, developers should consider providing a method of sharing game success. The application could provide a way of sending an SMS or MMS with the game name, score, and some text to a service provider or friend*
- → *Use of the T9 keypad for name entry should be avoided. Users do not expect it, and it doesn't work well for names. However, it is not possible to do otherwise when using TextFields.*
- → *Standard tools should be used for text entry, as they are already known to users.*

S60 Easy restart

If the user enjoys the game, s/he will probably want to play again as soon as possible. If the game has settings, such as player names or game modes, it is frustrating to set them again, even if the application helps the user.

- → *Provide an easy way to restart the game. For example, in the Game Over screen, provide a "Replay," "Play again," or "Restart" command, taking the user back to the beginning of the game, with the same settings as before.*

Appendix A Differences Between Series 40 and Series 60 Guidelines

Developers familiar with *Series 40 Developer Platform 1.0: Usability Guidelines For J2ME™ Games* document will recognize many of the issues explored in this document. To avoid repeating these issues, most significant differences between the guidelines are described below. Note that Series 40 document also includes implementation model for usability guidelines.

A. 1 Issues Present in Series 40 Guidelines but Missing from Series 60 Guidelines

The issues listed below have either been removed or their significance has decreased dramatically.

A.1.1 End key

The End key on Series 40 phones presented a common problem. Pressing the key during a game closes the application, and all users pressed it at least once during tests. The recommendation was to auto-save the game.

If the user presses the End key on a Series 60 phone, the application is not closed, but only switched to the background. Also, only 25 percent of users pressed the End key by accident.

→ *Recommendation has been changed from auto-saving the game to going into pause mode.*

A.1.2 Provide information to the user

In Series 40 phones there is a command to show information about the selected application to the user. Series 60 phones have a different menu structure and this option is not so openly present.

→ *The recommendation to include game information in the Details command has become less important.*

A.2 Issues Present in Series 60 Guidelines but Missing from Series 40 Guidelines

The following issues have been added to the Series 60 guidelines. This may be because of technical reasons (changes in the platform) or keypad modifications.

A.2.1 Sounds

Midi sounds are played by games even if the phone profile is set to silent. This is confusing and annoying to the user, and may even prevent playing of games.

→ *A game sound menu is now required (if there are sounds in the game).*

→ *Intro music is even more strongly discouraged than before.*

A.2.2 Application key

About 50 percent of users pressed the Application key by accident during tests. This key is not present in Series 40 phones, so it is not an issue on Series 40 phones.

→ *The game should go into pause mode if the player presses the Application key during a game.*

A.2.3 Appropriate main menu implementation

There are three different methods for implementing the game main menu in Series 60 phones. This was very different under Series 40 phones. See Section 5.3.1.3, “Navigation,” for an in-depth discussion of this issue.

→ *Included description of different implementation methods of the game menu and recommendation on which method to use.*

A.2.5 Artificial intelligence

As phone performance increases, games can become more complicated and have more sophisticated artificial intelligence (AI). Some guidelines on implementing an AI has been added.

→ *If relevant, the AI should adjust to the player's skill. Also, it should be made as human-like as possible; predictable, but still with surprising elements.*

A.2.6 Navi-key

The five-way Navi-key is not present in Series 40 phones so this issue doesn't apply to them. When trying to press down the Navi-key, many players accidentally pushed the key up.

→ *The developer should take this into account as much as possible; for example, not designing levels where "up" in place of "action" would most likely kill the player.*

A.2.7 Diagonal jumping

If users wanted to jump diagonally, they typically tried to press two buttons at a time. Users try to press diagonally on the Navi-key, which won't work.

→ *Some automatic handling of diagonal movement is recommended. Developers should aid the user in diagonal jumps (for example, by guessing where s/he wants to go) and design games so that diagonal movement is not necessary.*

A.3 Issues Present in Both but Updated from Series 40 to Series 60

Finally, it should be noted that these issues have been significantly modified since the Series 40 guidelines. There are smaller modifications in almost all guidelines, but their basic ideas have not been changed and therefore they are not listed here.

A.3.1 Top ten guidelines

The platforms are different and their usability issues vary in significance. As a result, the top ten usability guideline list has been modified to match Series 60 phones.

A.3.2 Controls

The controls section has been modified significantly. The Nokia 3650 keyboard layout is very different from that of other phones. There are new keys (Application key and Navi-key), and the layout of the numeric keypad is different.

A.3.3 In-game help

Earlier it was recommended that in-game help should be provided to the player. This is still valid, but further research shows that this guideline needs to be modified.

→ *In-game help should be provided as "information items," but the game should not be stopped. The help text should be shown only if the user requests it by pressing a key. This key (preferably 8) can and should be shown on the screen: "Press 8 to read note."*

A.3.4 Shortcuts and efficiency

The Shortcuts and Efficiency guideline has been modified.

→ *The "Restart" command should be included at the end of the game to enable an easy way to play again.*

→ *Players should be involved quickly; they shouldn't be forced to adjust several variables just to play a quick game and to get the feel of it.*

A.3.5 Status

This item has been moved to a more logical place, from Section 5.3.1.3, “Navigation,” to Section 5.3.2.3, “Game world match with real world.”

A.3.6 Give control to the user

This item has been modified. A significant addition is included; user control also applies to the game itself, not just the navigation or the story of the game.

→ *Developers should let the player prepare himself for the future.*

→ *They should not create a situation where the player cannot affect his/her destiny in any real way.*

A.3.7 Challenge

Although this issue is relevant for Series 40 games as well, some significant changes have been made.

→ *The challenge level should be kept near the optimum. If it's too difficult, it leads to frustration; too easy, it leads to boredom. The game might automatically adjust the difficulty level to match the player's skill.*

Appendix B Scope and Accuracy

B.1 Nature of the Guidelines

The guidelines were developed based on user testing with the Nokia 3650 and knowledge of human memory and perception information processing. While there are valid reasons for violating the guidelines, doing so may pose special risks.

These guidelines offer a way of creating usable mobile game, but they are not the only right way. They are recommendations, not rules, and offer a good starting point. Most likely the usability of the game will be good if these guidelines are understood and the ideas behind them grasped. However, there are times where it is reasonable and perhaps even recommended that they be violated. When violated, there should be a well-defined reason behind that decision, and the game must be tested again and again. Even then, careful consideration should go into violating a well-established, consistent user interface.

B.2 Cultural Issues

User interfaces have some very culture-dependent issues, such as language. These guidelines were developed in Finland using English as the phone language. There are cultural differences as well as requirements posed by operators.

Language and terms cannot be translated word for word, but local language and phone conventions need to be taken into account. It is important to preserve consistency – to use the same terms that are used elsewhere in the phone UI. If that is not possible, the terms selected should be tested with real users.

Differences in operator requirements might include an operator requiring that the right soft key brings up the menu; in that case, the guidelines would need to be adjusted accordingly. Testing the modified guidelines is not only recommended — it is essential to make sure that they provide improved usability.

Appendix C References

[1] http://www.usability.uk.com/

[2] http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/23

[3] Landauer, T. K., *The Trouble with Computers* (1995)

[4] Wixon, D. and Jones, S., *Usability for Fun and Profit: A Case Study of the Design of DEC Rally Version 2* (1995)

[5] http://www.usability.uk.com/

[6] Boehm, B. W., *Software Engineering Economics* (1981)

[7] Karat, C., *Usability Engineering in Dollars and Cents* (1993)

[8] Landauer, T. K., *The Trouble with Computers* (1995)

[9] Lederer, A. L. and Prasad, J., *Nine Management Guidelines for Better Cost Estimating* (1992)

[10] Pressman, R. S., *Software Engineering: A Practitioner's Approach* (1992)

[11] http://www.usability.uk.com/

[12] http://www.amazon.com/exec/obidos/tg/detail/-/0120958104/102-5825967-9525708?vi=glance

[13] http://www.useit.com/

[14] http://www.useit.com/alertbox/20030107.html

[15] http://www.idsa.com/ffbox3.html

[16] http://www.idsa.com/Top10flyerfinal.pdf

[17] Guidelines for Game Developers Using Nokia MIDP Devices, http://www.forum.nokia.com/documents