

OpenTrek: A Platform for Developing Interactive Networked Games on Mobile Devices

Johan Sanneblad and Lars Erik Holmquist

Future Applications Lab
Viktoria Institute
Box 620, SE-405 30 Göteborg, SWEDEN
www.viktoria.se/fal
{johans, leh}@viktoria.se

Abstract. Programming interactive networking applications for mobile devices is currently a laborious process, due to the lack of standardized development support. We introduce a new software platform, *OpenTrek*, primarily intended to assist the development of multiplayer networked games on Pocket PC devices. OpenTrek is similar to game development environments on stationary PCs, such as DirectX, but is fully optimized to work with mobile devices. It is a freely downloadable package with a fast learning curve, which includes support for ad hoc networking and efficient graphics. We successfully deployed OpenTrek in a course at a local university. 28 students with no previous experience in Pocket PC programming were able to create 12 different advanced multiplayer networked games in only five weeks (which included introduction to the platform). By easing the development of advanced interactive applications on Pocket PC, OpenTrek can lower the hurdle for researchers who wish to prototype and test novel user interfaces for mobile devices.

1 Introduction

In many ways, *games* are at the forefront of mobile HCI. A successful game must provide instant and intuitive interaction to novice users, while at the same time pushing the latest mobile technology to its limits. Previously, mobile gaming has primarily been the domain of specialized devices, such as Nintendo's *Gameboy*. Although most mobile phones also have a few built-in games, these have been fairly primitive. Recently, however, the capabilities of handheld computers and mobile phones have been increasing. A typical Pocket PC such as the *Toshiba e330* has a screen with 64,000 colors and a resolution of 240 x 320 pixels, a processor running at 300 megahertz, and built-in wireless networking. The *Nokia N-Gage* mobile phone is specifically developed to run games, and includes a 4096 color display with a resolution of 176 x 208 pixels, a 104 MHz ARM processor, plus Bluetooth and Java support. With devices such as these in the pockets of general consumers, we are likely to see an increase in full-color, networked interactive applications – many of which will be games.



Fig. 1. *Spaceball* is one of 12 interactive networked multi-player games created by students using the OpenTrek platform

But in our opinion, faster processors and colorful graphics is only part of what will make the future of mobile applications so interesting. What makes the next generation of mobile gaming devices different, be they Pocket PC, mobile phones or something else, is the inherent support for wireless networking. This will come in the form of the new generation of mobile phone networks, such as GPRS (2.5G) and broadband (3G) nets. It will also come in the form of wireless local-area networks such as Wi-Fi, and personal-area networks such as Bluetooth. All of these networking capabilities open up new possibilities for interactive entertainment and collaboration, which so far remain mostly untapped. The

only gaming device that is currently designed primarily with wireless networking games in mind is the *Cybiko* [4] but more are certain to follow as the technology matures.

However, the initial hurdle that needs to be overcome to develop applications for mobile devices is currently much greater than the equivalent for desktop computer. There are several game middleware platforms that support Windows PCs, such as DirectX [6], SDL [11] and ClanLib [3]. These platforms offer a standardized way of accessing advanced hardware features, such as graphics acceleration; they also provide a level of abstraction that makes the development of networking functions much easier. Although they are primarily intended for commercial development they have successfully been used in research projects, for instance to develop virtual reality applications [2] and videoconference systems [8]. But whereas for instance Microsoft's Pocket PC platform has become a widespread standard for handheld computing systems, there is yet no equivalent development support for such devices.

We have developed *OpenTrek*, a software platform for mobile devices, in particular those based on the Pocket PC standard. It includes middleware layers that aid in the development of interactive graphics and ad hoc networking, and a set of helper applications and APIs to ease development. The platform was designed primarily for prototyping multiplayer networked games, but can equally well be used for other demanding interactive applications. The design of OpenTrek was based on a game middleware platform for stationary computers, which will make it possible to transfer existing interfaces and prototypes from stationary to mobile devices.

In the following we will first briefly discuss some problems in designing software for mobile devices, after which the OpenTrek platform and its components will be described in depth. Following this we account our experience of the first practical deployment of OpenTrek during a university course, and exemplify with two games developed by students. Finally we conclude and discuss some future work.

2 The Need for a Development Platform for Mobile Devices

Handheld computers and mobile phones provide many new challenges for application developers and interface designers. A handheld computer is not simply a scaled-down desktop PC. Input methods differ radically, and interaction techniques that work well with a mouse and a keyboard may very well be completely inappropriate for the capabilities provided on a handheld. The usage situation differs too: whereas developers can count on desktop users to be in a situation where they can give the application more or less their undivided attention, mobile users may be walking, riding the bus, or engaging in social interaction while using an application. Thus it is extremely important to develop and test mobile HCI techniques on *actual mobile devices*, rather than running them in a simulated environment on a desktop workstation, or evaluating them in limited lab settings. This puts a high demand on researchers to make their applications run as efficiently as possible on mobile devices, to be able to correctly evaluate new applications and interaction concepts.

Whereas standardization has made it possible to run the same graphics-intensive code with virtually no performance loss across desktop computers from a wide range of manufacturers, this is not yet the case for handheld computer. Although Microsoft's Pocket PC standard is widely accepted, hardware details vary extensively between manufacturers. In particular, the way in which graphics is stored in memory can be quite different – a Casio Pocket PC does not use the same internal representation as for instance an IPAQ. If a developer wants to access the display on a Pocket PC without using middleware it is necessary to write directly to a specific memory area. But since displays are represented differently in memory on different devices, this makes it necessary to write several versions of the code to update the display efficiently on different devices.

On stationary computers, applications typically send messages over the network using insecure UDP packets. To move such an application to a Pocket PC device without using middleware would in practice require a complete re-design of its network communications. This is because of the limited network resources on mobile devices, where the incoming network buffer by default is only capable of storing a single large network packet at a time. Unlike stationary computers, the size of the incoming packet buffer cannot be changed. A networked application targeting Pocket PCs must therefore include at least a multithreaded incoming queue system, to avoid incoming network sockets from being recycled by the network interface. Other networking features such as service discovery, guaranteed transfer of messages, message priorities and automatic multicast are standard on game middleware platforms for stationary computers but are not available for mobile devices. To provide the same functionality for a mobile device, the developer would not only have to design protocols to support these features, but must also design a software architecture to integrate the networking features with the rest of the application.

3 OpenTrek

In our previous work we created several networked applications for handheld computers, where issues like those mentioned above had to be considered in every new project (such as ProxyLady [5] and Fieldwise [7]). What eventually became OpenTrek began as several thin class libraries created for these applications, targeting graphics, network connections and network communications. Using these class libraries required extensive knowledge, and preliminary feedback from students who used them was that the learning threshold was considered too high for use in most time-limited academic settings (such as courses and thesis work). Thus, work begun on an integrated solution, a platform based on the class libraries we had created but with a much shorter learning time.

OpenTrek is the common name for our collection of helper applications, class libraries, and run-time modules to aid in the creation of networked games for handheld computers. The platform is “open” in that it is modularized, enabling developers to add, change or remove features from the platform while it is running. The name OpenTrek originated in the project where the platform was first used, in which students used their handheld computers to “Trek” different physical areas using ad hoc network connections.

Choosing a hardware platform for OpenTrek, our requirement was that the handheld computer should support Wireless LAN for network communication. When work begun on OpenTrek, the only handheld computer platform capable of WLAN communication was the Pocket PC platform by Microsoft. Choosing the Pocket PC platform also meant that a version of OpenTrek for stationary computers could be developed in parallel using the same code base. The stationary version supports high resolution and hardware accelerated graphics and can be used for development and demonstrational purposes.

We will now describe the OpenTrek platform and motivate its design, illustrate its components and highlight how OpenTrek supports the creation of networked games and other applications for mobile devices.

3.1 Platform

The most widespread platform for creating networked games on stationary computers is the Microsoft DirectX platform [6]. The DirectX platform is a collection of thin class libraries, targeting graphics, sound, input, and network communication. The DirectX platform is similar to the class libraries we created in our previous projects, but requires extensive knowledge to use. When developing the OpenTrek platform, it was designed to be as similar to the graphics and network class libraries of DirectX as possible, so that students familiar with OpenTrek also would feel proficient with the DirectX platform, and vice versa. Helper applications and “wrapper classes” that simplify the use of the class libraries in OpenTrek make it easier to use and improves the learning time for new users.

Microsoft has recently ported part of their DirectX platform (DirectPlay for network communication) to handheld computers. Since it is similar to the version for

stationary computers, it still requires a significant learning time and experience to use in actual projects. This makes it unsuitable for student work in academic settings.

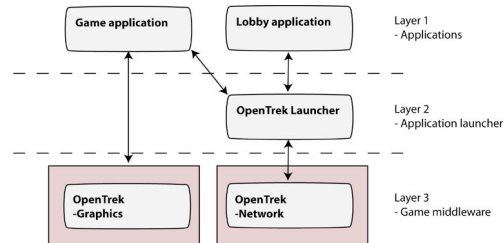


Fig. 2. The OpenTrek platform

The OpenTrek platform is shown in Figure 2 above. Hardware access is controlled with a graphics and networking middleware, and applications created using the platform can take advantage of several thin class libraries and “wrapper classes”. An OpenTrek application can itself not be started separately, but has to be initiated by a helper application called the *OpenTrek Launcher*. The OpenTrek Launcher can start several applications in parallel on the same handheld device, all of which may use the same graphics and networking middleware. The OpenTrek Launcher monitors the performance of each application so that the applications can cooperate using the same limited amount of graphics and networking hardware. The OpenTrek launcher has by itself no user interface. The user interface is started from the OpenTrek Launcher as an application called a *Lobby*, enabling the end user to customize its use and function. A typical function of a Lobby is to show what users have their devices switched on and what their activities are, as seen in Figure 3.

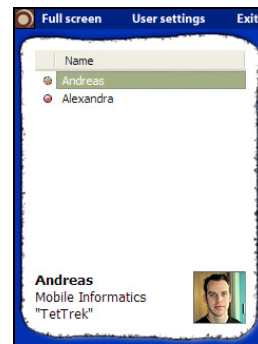


Fig. 3. A typical Lobby application, showing online users and their activities

3.2 Using OpenTrek

There are already many software platforms assisting with the development of networked games (such as DirectX [6], SDL [11] and ClanLib [3]). OpenTrek differs from these in that it provides a combination of high abstraction “wrapper classes” and helper applications that are specifically designed for networked games development for handheld devices. Both the wrapper classes and helper applications simplify the development process and shorten learning time. We will now describe the development process for a minimal whiteboard application to highlight how OpenTrek can be used in practice.

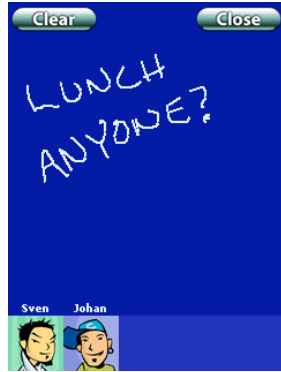


Fig. 4. The BlueBoard application

The BlueBoard Application. BlueBoard (as seen in Figure 4) is a simple whiteboard application that captures stylus strokes on a handheld computer and sends them to other devices on the same network. People should be able to initiate, join and leave whiteboard sessions at any time. Multiple whiteboard sessions should be able to run on the same network. BlueBoard should support ad hoc networking, so that it can be used in places where no Wireless LAN infrastructure is available.

new application shell. The wrapper class is then extended to handle coordinates received from the stylus input, and to pass them on to the OpenTrek Launcher as messages to be sent over the network. Finally the wrapper class is extended to handle coordinates received from other users over the network and draw them on the display.

The BlueBoard Implementation. There are three steps involved in creating BlueBoard. First an application wrapper class is used to create a

Many of the features in BlueBoard are provided by the OpenTrek helper applications. The OpenTrek Lobby application provides a way for people to initiate, join and leave sessions such as whiteboard and game sessions on the current network. The networking middleware of OpenTrek includes support for ad hoc networks. The graphics middleware provides a toolkit to draw double-buffered brush strokes on the display.

The application wrapper class in OpenTrek encapsulates device dependent logic such as messaging events, stylus input and button presses. It also contains logic for linking the application and the OpenTrek Launcher together. This link enables the OpenTrek Launcher to start and close applications as needed. One such example is when a person uses a Lobby to start a new BlueBoard session (as seen in Figure 5): the OpenTrek Launcher automatically starts BlueBoard on all devices simultaneously.

The BlueBoard application starts out as a subclass to the application wrapper class.

Extending the application wrapper class, code has to be added so that coordinates received from stylus input are encapsulated as coordinate objects inside network message objects. The network message object has fields such as ID, destination user, delivery mode, and timeout. Leaving the destination user field empty sends the message to all current users in the session. Code has to be added to pass the coordinate network messages to the OpenTrek Launcher, which places them in an outgoing network queue for immediate delivery. The code necessary to capture stylus coordinates and send them to all other users in the same session is listed below.

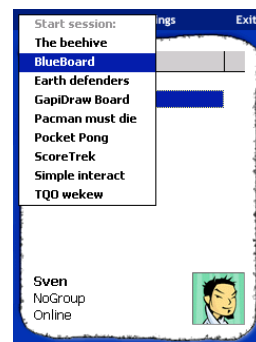


Fig. 5. Using a Lobby to start the BlueBoard application

Coordinate objects and network message objects are all a part of the OpenTrek class library, which is Open Source.

The program code to send stylus coordinates to all other users in the same session

```
HRESULT CMyApplication::OnStylusMove(POINT p)
{
    CDataPoint point(p); // Create a coordinate object
    CDataMessage msg(MSGID_COORD); // Create message object
    msg.SetData(point); // Store the coordinate
    m_pLauncher->SendMessage(msg); // Send the message
    return S_OK;
}
```

Again extending the wrapper class, the BlueBoard application is notified when messages arrive from other users (via the OpenTrek Launcher), and extract the coordinate objects contained within them. Using a list of previous coordinates received from each user, a line is drawn on the display. The code required to store the incoming coordinates received over the network is shown below.

The program code to receive coordinates from other devices in the same session and store them in a list

```
void CMyApplication::OnMessageReceived(const
CDataMessage& message)
{
    switch (message.GetID()) // The ID is set when the
    { // network message object is
    case MSGID_COORD: // created
    {
        CDataPoint point; // Create empty coordinate
        message.GetData(point); // Retrieve values
        m_incomingcoordlist.AddTail(point); // Store in list
    }
    break;
    }
}
```

The application wrapper class contains a display thread that is called several times each second, in which the lines are drawn to the display.

3.3 OpenTrek Helper Applications

In the following we will describe the two helper applications that are part of the OpenTrek platform: *OpenTrek Launcher* and *OpenTrek Lobby*.

OpenTrek Launcher. The OpenTrek Launcher is used to start applications built on the OpenTrek platform. Internally, the OpenTrek Launcher is the runtime engine that

connects OpenTrek applications with the graphics and network middleware. The OpenTrek Launcher operates as a message router, to which modules can be added dynamically while it is running. Examples of modules that can be added are Lobby applications that provide a user interface, and network modules that provide support for application sessions running on the local network.

All network communication for OpenTrek applications pass through the Launcher. The rationale for this is to allow seamless communication across multiple network interfaces (such as Bluetooth and Wireless LAN), and also to provide transparent session initiation and session management for ad hoc networks (the OpenTrek Launcher is responsible for creating a session on the network when needed). The layered approach makes it possible to send network messages to a specific application instead of a device, enabling multiple applications to run in parallel on the same device and network interface.

Another reason for using the launcher is that software running on a typical handheld computer never closes. When a Pocket PC device is “shut off”, it is actually frozen in its current state, and applications continue running as normal when the device is switched on. The feature where modules such as applications can be linked to the OpenTrek Launcher while it is running was required, since the user otherwise would sometimes be forced to reset the entire state of the device, shutting down all programs.

OpenTrek Lobby. An OpenTrek Lobby is an application whose primary purpose is to enable users to initiate, join and leave application sessions on the local network. A Lobby application contains the interface that is presented to the user when the OpenTrek Launcher is started, showing what people have their devices switched on and what they are doing. The user can switch between different Lobby applications, and it is also possible for a Lobby application to start other Lobby applications locally.

Lobbies are used in most networked computer games, and are typically located on a remote computer and accessed over the Internet (e.g. DirectPlay [6]). In OpenTrek, each network interface module is responsible for automatically joining and managing a “lobby” for the user. If the OpenTrek Launcher is run on an ad hoc network, all devices physically located nearby automatically join the lobby. If the OpenTrek Launcher is run using a module for mobile phones, lobbies are stored on specific devices running as servers (similar to “chat rooms”).

3.4 OpenTrek Middleware

The OpenTrek platform comprises two middleware components: networking and graphics.

OpenTrek Network. The OpenTrek Launcher has an Application Programming Interface (API) to which it is possible to plug in network modules. The API defines commands and requests sent by the OpenTrek Launcher to the network module, and callback operations that the network module can use to notify or request information from the Launcher. Examples of commands sent from the OpenTrek Launcher to the

network module include: *go online*; *get a list of online users*; *get a list of running sessions*; *create session*; *join session*; and *send message*.

The interface for the network modules was designed so that it would suit most types of network hardware, from infrared connections, to server-based connections over mobile phone connections, to ad hoc wireless network connections over WLAN. To test the interface specification in practice we implemented network modules for infrared, mobile phone and WLAN networks. The features that should be implemented by the networking module were defined by compiling a feature list from the Microsoft DirectPlay documentation [6]. Some of these features are: reliable and unreliable delivery of messages, sequential and non-sequential delivery of messages, automatic multicast delivery if applicable, automatic message fragmentation and reassembly, congestion control, and message timeouts using multiple outgoing queues.

Peer-to-peer. OpenTrek network modules are strictly peer-to-peer, meaning that there is no central host controlling the sessions on the network. Each network module is responsible for sending out online status information to all other devices, and to collect information on other devices and report back to the OpenTrek Launcher. The network module in OpenTrek that supports mobile phone connections sends and retrieves online status information from a central device that has been configured as server, and can be placed anywhere on the network. The approach where no specific device is used as a host enables ad hoc networking, where people can join and leave network sessions immediately as the device is switched on and off.

Server Mode. The OpenTrek Launcher can be configured to run as a “server”. This does not alter the functionality of the launcher, but it may change the functionality of the modules plugged into it. If for example a networking module designed for mobile network connections is running in server mode, it will change mode from send and retrieve information about online status to collect and distribute. Examples of OpenTrek applications using server mode are file servers, where the application changes from downloading files to distributing files to users in the proximity. A Lobby application running in server mode might be used to start an advertisement application on a user’s device as soon as the person comes within the proximity of the server node. The server mode of the launcher is typically used on devices with no user input, such as service-providing nodes in proximity-based and context aware service networks.

Session Initiation. The network modules in OpenTrek do not assign a host for each session. However, in some situations applications created on the OpenTrek platform need to assign one of the devices in the session as a controller, deciding who should be able to join the session and when. This controlling device could also be used to generate data that should be available before the session is initiated (such as map data for a game).

The OpenTrek Launcher initiates sessions on the network by starting the application on the first two devices synchronously, where the rest are allowed to join asynchronously. Applications can check during startup how many users are already participating in the session, meaning that one device will have only one user in the

session (the local user) and another device will have two users. The motivation for starting the first two devices synchronously is so that one device can be assigned as a backup host, in case the primary host should choose to leave the session. The Session Initiation Protocol used by the OpenTrek Launcher can be seen in Figure 6.

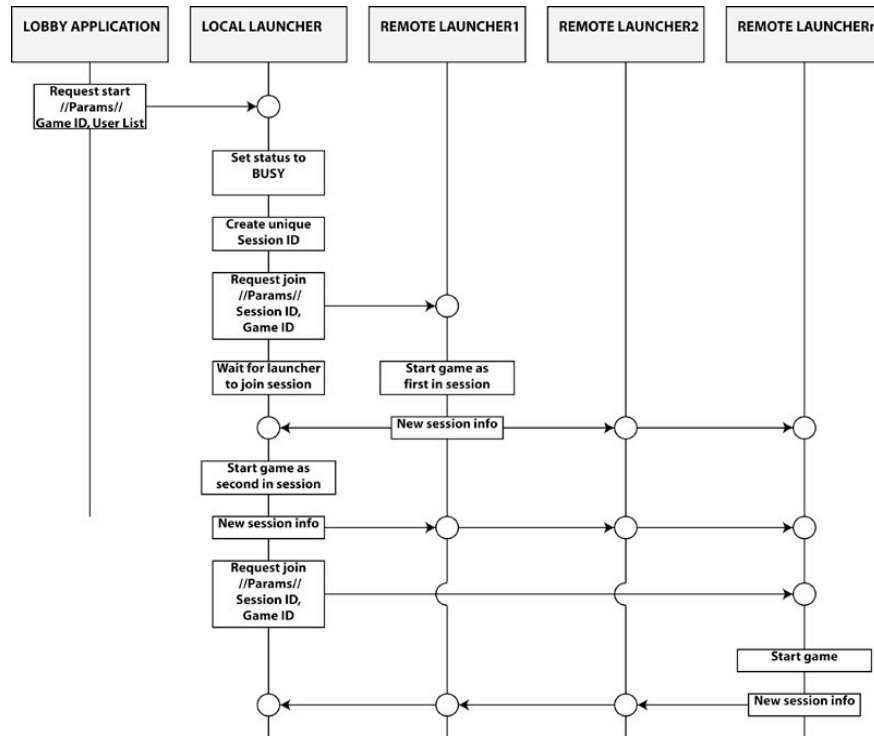


Fig. 6. The OpenTrek Session Initiation Protocol

Network Protocol. Networking modules in OpenTrek use the built-in IP stack for communication. Messages are sent as UDP packets, enabling multicast when used on local networks. Multicast improves performance scaling when the number of users joining a session increases. Networked games often send time-critical redundant data repeatedly, using UDP guarantees that the network hardware will not try to resend packages lost due to high packet-loss wireless networks.

One of the features in the network modules is the ability to send guaranteed messages. Guaranteed messages are sent using the Trivial File Transfer Protocol [13], TFTP. Messages are sent as small packets that are each acknowledged by the receiver. OpenTrek implements TFTP extensions such as the TFTP Blocksize option [12], to enable a variable packet size based on measured packet loss.

Information about online status is broadcast continuously in the current network Lobby as a small UDP packet. The online status information comprises information such as user id, device id, what sessions the user participates in, and the ID of every application participating in each session.

OpenTrek Graphics. The graphics middleware in OpenTrek is a toolkit we have created for fast graphics on handheld computers, called *GapiDraw*. *GapiDraw* is not accessed through the OpenTrek Launcher, but is called directly by the OpenTrek applications using a class library. The OpenTrek Launcher is however responsible for starting the middleware and sharing it to the applications. The OpenTrek Launcher also determines when a specific application can draw to the display, in case several applications are started in parallel.

The interface to using *GapiDraw* is nearly identical to using *DirectDraw* in the Microsoft *DirectX* platform. The motivation for creating *GapiDraw* came from the lack of available platforms to create fast graphics on handheld devices. Commonly available platforms for stationary computers such as *DirectX* [6], *SDL* [11] and *ClanLib* [3] are not optimized for and will not work on handheld computers. Handheld computer platforms such as *Overloaded* [9] and *PocketFrog* [10] have all been abandoned by their authors and are no longer being supported or updated. As of current, the only graphics toolkit available for Pocket PCs still being maintained, updated and supported is our *GapiDraw* toolkit.

GapiDraw has been separately released on the Internet and input from game development companies have resulted in a feature set comprising most 2D operations necessary for games development. Some of these features are alpha blends, rotation, zoom, bitmapped fonts, collision masks and sprite intersections. An example of a *GapiDraw* game, *Pocket Everquest*, is shown in Figure 7.



Fig. 7. *Pocket EverQuest* by Sony Online Entertainment was created using our graphics middleware *GapiDraw*

4 Implementation

The OpenTrek platform is implemented in C++ and runs on Pocket PCs and stationary computers. For a Pocket PC device to use the OpenTrek platform it must be equipped with a network interface, such as Wireless LAN or GPRS. An internal test version of OpenTrek for the Symbian operating system was created to verify that the class libraries and APIs would work on devices other than the Pocket PC.

Applications created for the OpenTrek platform will automatically run on both Pocket PCs and stationary PCs, simply by recompiling them for the correct system. When OpenTrek runs on a stationary computer, the built-in video card will accelerate all graphics operations. Video hardware acceleration enables high-resolution stationary applications that communicate with their Pocket PC counterparts.

The modular design of the platform has made it possible for third party developers to expand the platform with additional features. Extensions currently available are 3D graphics, streaming video and sound support. All of these extensions build on our middleware and extends it with new functionality.

OpenTrek has been tested to run on stationary operating systems such as Windows 95/98/ME/2000/XP and most 12-bit and 16-bit color Pocket PCs.

5 Experience: OpenTrek in Education

OpenTrek was developed to allow rapid prototyping of interactive networked applications on handheld computers, primarily games. An important requirement of the platform was that it should be so easy to use that even developers with little or no experience could quickly get up to speed and develop new applications. In particular, we intended OpenTrek to be used in an educational setting.

The first practical use of the OpenTrek platform was in a course at the local university, attended by 28 students. The students were at the graduate level, with approximately three years of studies in computer science or equivalent. Each student was equipped with all the necessary development tools for targeting both stationary computers and handheld devices, including a personal laptop and a WLAN-equipped Pocket PC (Compaq IPAQ H3630). The goal with the course was to develop a game from scratch in five weeks.

All students had at least two weeks of C++ programming experience, but none had previous experience of game development. The first week of the course we introduced the students to the OpenTrek platform. We also taught the students some fundamental principles in game development, and supplied them with various design patterns suitable for real time networked games development. After the initial week, the students were divided into 12 groups, ranging from 1-3 people. The groups were then allowed an additional four weeks to develop their games.

The major requirement to pass on the course was to produce a fully working multiplayer networked game within the time given for the course. The game should be responsive in real-time, fully interactive and preferably multithreaded. To pass with distinction, the game should use advanced visually responsive graphics. Ideally, the game should also define some new and innovative uses of mobile ad hoc networks for gaming.

After the four weeks were up, each group was required to present not only their game concept and design, but also demonstrate their game live using two or more actual handheld devices. At this time, all students sent us the latest version of their source code so we could perform a thorough code review to search for problems and provide feedback for improvements.

5.1 The Games

All of the student groups managed to develop and demonstrate a fully working game during the five-week course. The twelve games ranged from simple card games to advanced real time action games. Eight of the game used what we would term

visually responsive graphics, i.e. they were full-fledged arcade-style action games. Two of the games supported more than two players.

Several groups introduced innovative ways to use networking to support new types of games. For instance, one group created a form of meta-game where players would move between different “islands” where they could buy and sell various items, that could then be used in other games. Each island was in fact a Pocket PC, and the size of the island was defined by the range of the devices’ wireless network range. In other words, to be able to trade with an island, the player would have to be within physical range of it, or rather the device which represented it. This is a way of promoting players to move around and socialize, much in the way of context-aware social games like *Pirates!* [1].

In the following we will highlight two of the more advanced games. More information about all games developed during the course can be found at:

<http://www.cafetrek.com/>

TrekFighter. *TrekFighter* (Figure 8) is an updated version of the classic multiplayer game *Snake*, often found in mobile phones (e.g. Nokia 6110). In the game each player controls a small figure on the display – the “snake”. The player must take charge of his or her snake so that it does not run into the wall or into another player – or its own tail! During the course of the game, the length of the snake increases, making it more and more difficult to maneuver. The winner of the game is the player who is the last to run into a wall or the tail of the other persons snake.

The students’ game is a significant advance on the traditional version. It supports up to four players over the network; when the game starts, the players’ snakes are distributed over the corners of the playing area. The students added many features such as “pick up bonuses” that give the ships a speed-up or slow them down.

The application even supports game maps that are larger than the actual display area; the graphics will scroll smoothly to focus on the player’s own snake. An external artist was called in to design the game’s graphics.

The network communication was implemented using unreliable peer-to-peer model. The first device to start a session is automatically assigned server status, whereas the second device is assigned as a backup server. All devices in a game session continuously send their ship coordinates and ship rotations to the server device. This device in turn continuously broadcasts all ship coordinates using automatic multicast to all the other devices in the current session. This model does lead to some problems, since if the server device leaves the game, all clients will be forced to leave the game as well.



Fig. 8. *TrekFighter*, a variation of the classic game *Snake*, supporting up to four players playing over a wireless network

TetTrek. *TetTrek* (Figure 9) is based on the classic game *Tetris*, where the player controls falling bricks. If the bricks can form a solid horizontal line, that line is removed from the screen. The goal is to keep playing for as long time as possible, and points are awarded as lines are removed. This multi-player version of Tetris allows two opponents to play over the wireless network. Each player can see both his or her own map and that of the opponent.

The students added some additional features to spice up the game. In particular, “bombs” can be earned by successfully clearing rows of bricks in a certain pattern. These can then be set off on the opponents map to cause various types of mischief, such as creating walls, adding bricks, etc. All graphics for the game was developed by the students themselves. The network communication in *TetTrek* is based on a model with guaranteed message transfers. The local game area is continuously sent to the other device in the current session. Other commands, such as bomb transfers, are also sent in a secure way.



Fig. 9. *TetTrek* is a two-player networking version of Tetris

6 Conclusions and Future Work

We have presented OpenTrek, a software platform that allows developers with little or no previous experience of software development for handheld computers to quickly create advanced interactive networked applications. Based on the results from the use of OpenTrek in an educational setting, we argue that the platform provides an efficient way to quickly develop new games as well as other interactive applications. By giving developers and researchers the ability to make efficient interactive prototypes that can be run on a variety of Pocket PC devices, OpenTrek makes development and testing of mobile interfaces much easier. This means that it can be an important contribution towards enhancing the usability of mobile applications.

Future work involves adapting the OpenTrek platform to work on mobile phones. Mobile phones introduce use situations that differ from handheld computers, situations that can be explored with the help of new OpenTrek applications. Work is also planned to create interfaces to programming languages other than C++. For example using a managed wrapper for the Microsoft .NET platform, OpenTrek applications can be created with programming languages such as Visual Basic, C# and Java.

7 Acknowledgements

This research was funded by the Swedish Research Institute for Information Technology (SITI), and the Mobile Services project financed by the Foundation for Strategic Research (SSF). Special thanks go to all our students who worked very hard to create some amazing games!

8 Download

The OpenTrek platform can be downloaded freely from the web using the following URL:

<http://www.opentrek.com/>

GrapIDraw, the graphics component of OpenTrek, can be downloaded from:

<http://www.gapidraw.com/>

References

1. Björk, S., Falk, J., Hansson, R., Ljungstrand, P.: Pirates! - Using the Physical World as a Game Board, Proc. Interact 2001, IFIP TC.13 Conference on Human-Computer Interaction, July 9-13, Tokyo, Japan.
2. Chiang, C.C., Huang, A., Wang, T.S., Huang, M., Chen, Y.Y., Hsieh, J.W., Chen, J.W., Cheng, T.: PanoVR SDK—a software development kit for integrating photo-realistic panoramic images and 3-D graphical objects into virtual worlds,” In Proceedings of the ACM symposium on Virtual reality software and technology, 1997, Lausanne, Switzerland, pp. 147–154.
3. ClanLib, <http://www.clanlib.org/> Last visited Feb. 18, 2003
4. Cybiko, <http://www.cybiko.com/> Last visited Feb. 18, 2003
5. Dahlberg, P., Ljungberg, F., Sanneblad, J.: Proxy Lady: Mobile Support for Opportunistic Interaction, Scandinavian Journal of Information Systems volume 14, 2002, Gothenburg, Sweden.
6. Fagrell, H., Forsberg, K., Sanneblad, J.: FieldWise: A Mobile Knowledge Management Architecture, In Proceedings of the 2000 ACM conference on Computer supported cooperative work, 2000, Philadelphia, Pennsylvania, United States, pp 211-220.
7. DirectX, <http://www.microsoft.com/directx/> Last visited Feb. 18, 2003
8. Lin, C.S.: On design of network scheduling in parallel video-on-demand systems, In Proceedings of the seventh ACM international conference on Multimedia, 1999, Orlando, Florida, United States, pp 211-212.
9. Overloaded, <http://overloaded.pocketmatrix.com/> Last visited Feb. 18, 2003
10. PocketFrog, <http://pocketfrog.droneship.com/> Last visited Feb. 18, 2003
11. The Simple DirectMedia Layer (SDL), <http://www.libsdl.org> Last visited Feb. 18, 2003
12. The TFTP Blocksize Option, <http://www.faqs.org/rfcs/rfc2348.html> Last visited Feb. 18, 2003
13. The TFTP Protocol, <http://www.faqs.org/rfcs/std/std33.html> Last visited Feb. 18, 2003