Geometry Teaching in Wireless Classroom Environments using Java and J2ME [1]

Ulrich Kortenkamp [a,b] Dirk Materlik [b]

[a] *TU Berlin, Fachbereich Mathematik, Straße des 17. Juni 136, 10623 Berlin*
[b] *FU Berlin, Institut für Informatik, Takustraße 9, 14195 Berlin*

Abstract

Interactive geometry software is established as an important tool in geometry and math education. We present our research on possible ways to use such software in wireless classroom environments. In particular, we address user interface issues on portable devices and describe how we maintain a common code base for both desktop and mobile environments, thus increasing the stability of the application. We also report on our empirical data comparing different Java virtual machines that are available for portable devices using a prototype implementation of the Interactive Geometry Software Cinderella for J2ME.

Key words: Java, J2ME, wireless classroom, geometry, Rendezvous, Cinderella, Zaurus, collaborative environments

1 Introduction

1.1 The Interactive Geometry Software Cinderella

The starting point of our investigations is the software package *Cinderella* [21], which has been developed by our group since 1996. Cinderella is a software for doing geometry on a computer. You can work with points, lines, segments, circles, conics, polygons and other objects using the mouse. In contrast to drawing software like, e.g. *Corel Draw* or *Adobe Illustrator*, the objects you

Email addresses: kortenka@math.tu-berlin.de (Ulrich Kortenkamp), materlik@inf.fu-berlin.de (Dirk Materlik).

[1] Supported by the DFG research center "Mathematics for key technologies" (FZT 86) in Berlin.

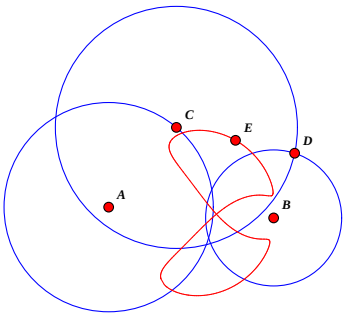

Fig. 1. A simple Locus: All circle radii are fixed. Point C moves on the circle around A. D is the intersection point of the rightmost circles. E is the midpoint between C and D. The red curve is the locus of E, i.e. all possible positions of E under movements of C.

create are not independent of each other, but may be connected by mathematical relations. The most basic example is a line *through* two points which will automatically update its position when one of the points is moved. Using relations like *orthogonal*, *parallel*, *midpoint of*, etc. it is possible to create complex constructions consisting of *free* and *dependent* elements.

When a free element is moved, the elements dependent on it are updated automatically. This guarantees that user-supplied mathematical constraints will always be fulfilled. In a way this is comparable to parametric CAD systems, but Cinderella is not intended as such, but it is rather meant as a tool to be used in math research and education.

For almost 15 years many similar packages have been developed, starting in the end of the 1980's with *Geometers' Sketchpad* [8] and *Cabri Géomètre* [15], which are still the most widely used ones. Cinderella, while lacking some of the features of Sketchpad and Cabri, sets itself apart by providing features that make it useful for advanced geometry research. Among these are multiple views that support multiple (non-euclidean) geometries, a consistent implementation of continuous movements, and self-exploring loci (see Fig. 1 for an example of loci). For a description of the mathematical foundations of Cinderella we refer to [11], and for in-depth coverage of the non-trivial algorithmic and mathematical problems we recommend [22].

An innovative feature of Cinderella is the ability to export its constructions as interactive web pages [13]. This is possible because Cinderella is written entirely in Java [14]. A Java runtime component, packaged as a *jar* file, that contains an applet version of the software, is provided for that purpose. It may be copied freely and even be placed on the web. All Cinderella owners can thus publish their work in an interactive form. The constructions can be manipulated on a web page by moving free elements in exactly the same way as in the stand-alone version.

It is even possible to export part of the user interface to the web. We use this to support *interactive construction exercises*: A teacher can take parts of a construction and hide them, mark parts of the construction as suitable intermediate results, and select the final solution elements a student should find. Using either a predefined set or a custom choice of the construction tools a student should then try to find a the solution on her own, starting from the non-hidden parts. She can request help which will reveal the intermediate results one after the other. You find a examples of interactive exercises on our web page [2].

This seems a rather rigid way of solving exercises, but the mathematical methods inside Cinderella ease the restrictions: Cinderella is able to *prove* whether a different solution is equivalent and thus also correct. The student may find her own ways of solving the exercise, even ways that the teacher did not know or think of.

A special version for schools [20] is based on these web export features. It includes more than 130 example constructions and exercises that are especially suited for K-12 education. Free and commercial exercises are available on the web, for example at MathsNet [3].

1.2 Modern Hardware for Classroom Education

During the last years new hardware has emerged that seems to be especially suited for classroom use. Our goal is to find out whether the Cinderella system can take advantage of the new hardware without spending too much effort on porting and testing, as we can devote only few resources to this.

Here we consider two devices that represent the two extremes in size: *Electronic Whiteboards* and *Personal Digital Assistants (PDAs)*. Electronic Whiteboards are used as a mouse replacement for desktop or notebook computers: The computer screen is projected onto the whiteboard. The whiteboard detects the position of a special pen using electromagnetic technology, comparable to a large version of Wacom graphics tablets. The scenario we think of here is the traditional classroom in which the teacher stands in front of the class, explaining the subject matter. Cinderella on an Electronic Whiteboard can aid the teacher by facilitating the exact visualization of geometric constructions and proofs.

Today's PDAs, on the other hand, are shrunken versions of desktop computers, providing processing power equivalent to that of desktops a few years ago. They are powerful enough to run Cinderella. We envision them to act as "geometric pocket calculators," doing for geometry what traditional pocket calculators do for arithmetic. They act as accurate drawing paper in which the

elements are movable. This is preferable to furnishing students with desktop or laptop computers; due to their smaller size, the PDAs are much less in the way of teacher-student interaction. Additionally, they are purpose-built, therefore they cause less struggle with software installations.

With the advent of wireless networking for PDAs, further scenarios become possible. With the right software it is possible for several persons to jointly work on the same construction. A major drawback of computer-based activities – the lack of communication between students and between students and the teacher – can be avoided in this way. More specifically, we want to enable at least the following scenarios:

- The teacher presents a construction that is accessible to all students on their local machines. A student can be asked to identify parts of the construction or to complete a construction without having to present in front of the class and change the computer she is working with. [2]
- Students work in groups to solve a problem. Every student can try out her own solution locally and "publish" it for the others when she thinks they should use her partial solution to continue.
- Students work on their own under supervision of the teacher. When the teacher wants to comment on a student's work she can link to her construction and show it to the whole class.
- Students in remote locations can share their knowledge about a construction and demonstrate their findings live. This is further supported by a simple chat facility.

Again, we stress that all this can be done (and is done already) with desktop or laptop computers. However, we are sure that PDAs can be a better alternative as they do not interfere as much with human-human interaction as computers do. Also, they are cheaper than notebook computers and could be built more robust than these, which is important for classroom use.

1.3 Reasons for Using Java and J2ME

Current portable devices for geometry are based on programmable graphic calculators. In particular, there exist geometry modules (based on either Geometers' Sketchpad or Cabri Geometry II) for the TI-89/92 calculator series of Texas Instruments. More recently, the Casio Classpad 300 has been introduced, which includes built-in geometry software by Saltire Software.

[2] We understand that it is not advisable to reduce the physical exercise of todays' children. However, our experience is that teaching in computer labs leads to students that do not leave the place in front of the computer anyway and then there is no easy way to jointly present anything for the whole group.

Despite the common name of the desktop version and the mobile version, the software running on these devices was written specifically for them. This is a disadvantage, for examples when trying to share constructions between the versions: Not all features are available in both worlds, the file formats are different and have to be converted, and the behavior of the constructions depends on the platform.

Another drawback of the calculator-based devices is the lack of features that are now available on PDAs. You cannot use wireless or even wired network connections. There is only a serial port (cable- or infrared-based) that can be used for slow file transfers. The screen is monochrome; modern PDAs or even cellular phones sport color displays.

By using Java, we can port our software to PDAs, and benefit from the additional features above. Even better, we are able to share a common code base between the desktop version and the PDA version. This not only reduces the necessary resources for developing the software, but also makes it easy to ensure a common feature set and common file formats across platforms. Bugs fixed on one platform will automatically be fixed for all other platforms as well.

2 A PDA Version of Cinderella

2.1 User Interface Issues

Before we could create a version of Cinderella for whiteboards or hand-held devices we had to redesign the desktop-oriented user interface [16]. Both of these devices are *pen-driven*, which has some implications for the user interface. One, on the whiteboard the user has to move her hand a longer distance than with a mouse; this makes a toolbar-oriented interface awkward to use. On a PDA's limited screen size there is not enough room for a toolbar-oriented interface anyway. Two, with a pen, usually there is no "mouse-over"-event, only mouse button presses and mouse drags can be observed. This makes it harder to guide the user with tooltips and other hints. Three, we can rely only on a single logical mouse button, e.g., in the case of touch screens. Four, pens are different in that clicks and drags are more difficult to distinguish than with a mouse, because the pen will usually move a little between button-down and button-up.

The last problem was easily solved using a custom recognition for mouse clicks that filters the accidental movements and generates custom click events instead of the ones provided by the platform. The other points had to be addressed

on a more fundamental level.

Normally, relations between elements are introduced by choosing the corresponding mode, e.g., "Circle by Midpoint and Radius," from the toolbar. To eliminate the toolbar, two approaches are possible: Minimize the necessity of switching modes, or find a way to select a new mode that does not involve travelling long distances with the pointer and takes up no screen space. We pursued both, the first by a new mode called *Scribbling*, the second by *Cinderella Flow Menus*.

Scribbling is a gesture-based input method that translates sketches of constructions into formal construction sequences. Here, points, lines and circles are drawn as they should appear on screen. The software distinguishes them by applying certain metrics, e.g. the curvature and size of the drawn objects.

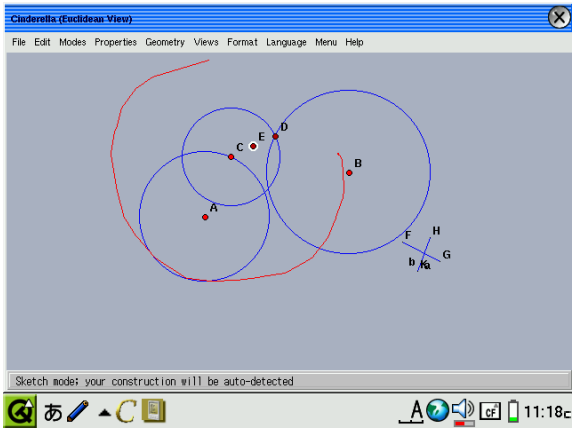

Fig. 2. Scribbling on the Zaurus C-700 (original size)

The definitions of new elements are set to default values: points are freely movable, lines will be incident to the points are incident to them in the drawing, if any. If there are no incident points, new free points will be generated. Circles are handled similarly, their current radius (as read off the drawing) will be taken as a parameter, their midpoint is either an existing point that happens to be in the center of the circle, or a new midpoint will be generated.

If a user wants to create objects that have additional properties, i.e. that are in special relation to the existing elements, she can add these either by specifying certain hints before drawing the element or by drawing certain marks after the element has been added.

Currently, the relations that can be added include:

point on, intersection When a point is drawn on top of an existing line or circle, or on the intersection of two lines, it will be a point bound to that element or intersection.

parallel lines After drawing a line ℓ_1 through points A and B, this line and another line ℓ_2 can be ticked with a short stroke, which will change ℓ_1 to a line parallel to ℓ_2 through A. Alternately, a line may be preselected; if the new line is approximately parallel to the existing one, it will be considered such.

orthogonal lines After drawing a line the intersection of this line with another line is marked with a short stroke (similar to drawing a orthogonality angle mark), which will change this line to a line orthogonal to the second line. Alternately, if a line is preselected and the new line is approximately orthogonal to it, the new one will be considered an orthogonal.

midpoint If two points A and B are preselected before a point is drawn, this new point will be the midpoint between A and B.

circle by center and another point If the drawing of a circle starts at a point A, its radius will be determined by the distance between A and its center. The center can be preselected by tapping it before drawing the circle.

It is also possible to move an element instead of drawing new elements by selecting it first. It will be highlighted and can then be moved with the pen.

Flow Menus have been proposed as a variant of menus that do not require moving the pointer far away from the current focus of interest [4]. We adapted them to *Cinderella Flow Menus*. A flow menu is a variant of a radial menu in which a menu item is selected when the pointer moves from one of the outlying areas back to the center one. Submenus are also selected when the pointer moves in the opposite direction, from the center to the outlying area. Implicitly, commands chosen from submenus become *gestures* that the user learns automatically.

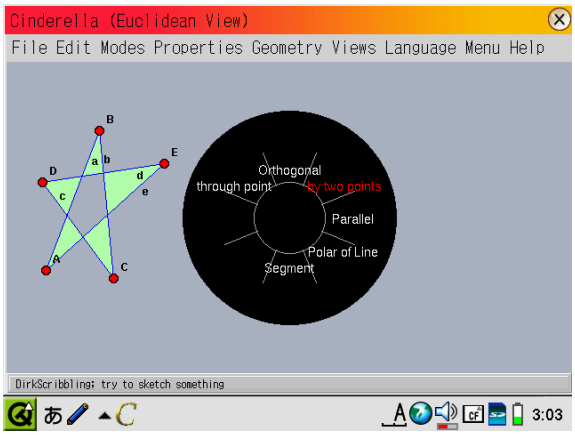

Fig. 3. A *Cinderella Flow Menu* on the Zaurus C-700 (original size)

The mode-selecting *Cinderella Flow Menu*, as shown in the picture, is activated either by a special gesture from the *Scribbling* mode or from the menu. Using flow menus on a pen-driven device feels natural after a short time and is much faster than the alternative of drop-down menus.

Further details of the design and implementation of both approaches can be found in [16].

2.2 Target Devices and Licensing

Modern PDAs are powerful enough to be used for interactive applications like Cinderella. For us, it was important to use a device that supports Java, as explained in section whyjava. Furthermore, it is important to stick to the graphics and user interface toolkit. For maximum compatibility, Cinderella uses the Java AWT version 1.1, which is included in almost every desktop Java VM, in particular in most versions of *Microsoft Internet Explorer*.

The dominant PDA operating system today is *PalmOS*. When we first ported Cinderella, PalmOS was only available in version 3, which was not able to support Java and the standard Java windowing toolkit AWT. Today, in version 5 and on modern hardware, J2ME support should be possible; however, it seems there is no VM available yet, but various efforts have been announced. All these efforts do not provide standard AWT compatibility, but conform to the MIDP profile [17] which means that the whole graphics code of Cinderella would have to be rewritten. Therefore, we currently do not consider PalmOS PDAs.

The first device we did test was a Casio Cassiopeia E-125G. This is a PocketPC using Microsoft Windows CE 3.0 as operating system. As there is no pre-installed Java VM on this device we had to look for a third-party implementation. We finally obtained a commercial implementation of Java by Insignia Solutions, the Jeode Java VM. Although we have been able to port Cinderella to this device using a developers' version of Insignia Jeode, we did not pursue this further as it is not possible to obtain single-user licenses for the Cassiopeia version of Jeode. The Jeode VM was made available as a user-installable add-on for the Windows CE 3.0 PDA series made by Compaq later, but we did not test this version yet.

When SHARP introduced the the Linux-based Zaurus SL-5000/5500, Java support was advertised as one of its main features. The Zaurus SL-5x00 and, more recently, the SL-C7x0 series ship preinstalled with the Jeode VM version 1.10.7 by Insignia Solutions. It conforms to the *PersonalJava 1.2* specification by Sun Microsystems [7]. However, PersonalJava is not a Java 2 environment, it only conforms to version 1.1 of the language specification with some additional APIs, and it will soon begin the Sun End of Life process [24].

Recently, Sun has made available an early access version of a *Java 2, Micro Edition* virtual machine for Zaurus at [25]. This is a scaled-down Java 2 environment including the personal profile [9]. The version available to us is the

early access release 1.0ea4, the version shipping on the new SL-C750/C760 is the final version and may differ in performance and bug fixes from this one.

For the Cinderella development team it would be advantageous to drop support for the older Java 1 VMs and be able to always use the new Java 2 features; however, as we decided to stay compatible to the Microsoft VM preinstalled in many Windows versions we did not do the transition yet. Therefore, running Cinderella was easily possible with both Zaurus VMs; we have prepared a packaged version that allows the user to run Cinderella with both VMs [12].

When the Zaurus was introduced an alternate environment called *Intent* was presented by TAO group [26]. This environment has its own assembly code that is interpreted on top of a special operating system. Among other things, there was a Java compiler to this proprietary code. At *CeBIT 2002*, we had the opportunity to let Cinderella run under Intent on a Zaurus SL-5500. This is a very fast way to execute Java on the Zaurus, significantly faster than under any other VM. However, deployment is much more involved and error-prone, it does not integrate well with the rest of the Zaurus, and there were several problems that seemed non-trivial to fix. We were not able to obtain a development version of Intent to study this further, and the Java parts do not seem to be available any longer.

2.3 Operating Systems on the Target Devices

The Casio PDA is a Windows CE device; the SHARP uses Linux as its operating system. This makes a signficant difference in the ease of porting. With portable devices the usual edit → compile → debug cycle is extended to edit → compile → package → deploy → debug. The turnaround time from coding to testing is dramatically longer. Also, the small screen makes it impossible to have debugging output on the device.

We ported to Windows CE using the deployment environment provided with the Jeode VM. It was not possible to integrate this with our build environment (based on *GNU make* at that time, now based on *Ant* [5]), although we always took care not to be dependent on platform-specific features[3]. Therefore, we were not able to "release often" [1,28], as the turnaround time between code changes and testing was half a day. Also, we could only install the Java VM using the developer tools from Insignia; it was impossible to create a user-installable version of Cinderella.

[3] Since 1996 we have used Sun Solaris, Linux, Windows, and now Mac OS X as primary development platform without any trouble moving from one to the other.

In contrast, porting Java software to the Zaurus is an easy task. We can connect to the device over the network and, from the command line, observe exactly what is happening. Since all of the standard UNIX tools are available, software can be deployed very easily, configuration files can still be edited after deployment, and debug logs can be obtained in the same way as on a desktop. If it was not for the lack of speed we could even do the whole build process on the Zaurus.

Since the Zaurus package format is well-documented, building an installable package of Cinderella as part of our normal build process is fairly straightforward. The normal Zaurus package management tools can install this package, making it easy for end users to use Cinderella.

3 Technical Issues

3.1 Development Environment

As a consequence of the multiple target platforms of Cinderella, which includes Java 2 SE based environments (Windows, Mac OS X, Linux/Unix), Java 1.1 virtual machines (e.g. in standard installations of Internet Explorer on Windows or on Mac OS 8/9), and the J2ME on the Zaurus, we cannot use a specialized environment for any of these. Instead, we use a general purpose Java IDE, in our case IntelliJ IDEA [10]. Version control is handled by CVS.

A big advantage of IDEA compared to other IDEs is, besides its superior support of refactorization, its nonintrusive operation. Other IDEs we evaluated, e.g. Eclipse, cannot handle source changes caused by other programs as well as IDEA does. This is important because of several reasons, for example we are using the ANTLR [18] parser generator to create additional sources.

The build process is governed by *Ant* [5], an open source Java build tool that has become industry standard. We can control the build both from within the IDE and from the command-line.

For all other details, we refer to [14].

3.2 Separation of Platform-specific Code and Code Validation

As we want to share as much code as possible between the different targets, we had to devise a way to separate platform specific code that would cause other platforms to crash or malfunction in another way.

A prime example is the use of Java 2 specific code: Any occurrence of classes that are unavailable in versions prior to Java 1.2 will cause the Microsoft just-in-time-compiler (which is only able to handle Java 1.1.x and is turned on by default on Windows machines) to crash, even if the class is neither instantiated nor referenced in any other way. We circumvent this by moving all usages of "forbidden" classes into another helper class, which will only be loaded when we are sure to be on a Java 2 platform.

Another issue are platform specific workarounds or UI adaptations. For example, on Mac OS X we are changing the overall look of the application to suit the rest of the OS. We have to do this manually, as we are not using the Swing Toolkit but the standard AWT. We also register the application for other events, like the application execution notification events of Mac OS X. These changes violate the rules for 100% pure Java, but they greatly enhance the user experience.

We do not use tools for automatic code validation. There are official tools for checking the J2ME conformance or the Java purity of code, but we are not able to use them. As described above, we deliberately breach the rules at some points. Many code constructs are not allowed or deprecated, but we ignore the warnings and work around errors at compile time, while we make sure to take care of them at run time. A code fragment that has been deprecated for the Java 2 platform might be the only solution for the Java 1.1.x virtual machine.

3.3 Benchmarks

To make it possible at all to use Cinderella on a PDA, the target platform has to be both fast enough and memory efficient. When using Java, this depends more on the VM implementation than on the processor speed or main memory of the device. In this section we describe our experiences with the two available virtual machines for the SHARP Zaurus.

We started our investigations directly with the whole application and not with small platform benchmarks, as it was easy enough to try out immediately whether the application will be usable at all. At first sight, the Sun Java 2 virtual machine, besides its increased functionality, seemed much more responsive and somewhat faster in its calculations. With this VM and the SL-C700, Cinderella *feels* fast enough to use it for real projects, while the Jeode VM is only suitable for a demonstration prototype.

To further justify and in order to explain this rating we conducted some benchmarks. In Table 1 we list the results of three basic tests: *Startup* gives the times from the moment the user launches the application to the moment Cinderella is responsive to user input, on a freshly booted machine. *Load simple locus*

	Jeode	Sun VM
Startup	30.8 sec	23.8 sec
Load simple locus	9.0 sec	5.5 sec
Load complicated locus	12.4	7.7 sec

Table 1
Application Benchmarks: All tests were performed on a SHARP CL-700. The times are the averages of 3 tries. The times were taken by hand.

	Jeode	Sun VM (first)	Sun VM (subs.)	Laptop
int	0.6 sec.	8.9 sec.	1.0 sec.	1.2 sec.
float	6.6 sec.	20.1 sec.	4.7 sec.	2.7 sec.
double	7.0 sec	24.6 sec	13.0 sec	2.8 sec.

Table 2
Basic calculations with different number types. Each test is run 10 000 000 times. The column *first* shows the result when the code is executed in the main method of the benchmark only once. The column *subs.* shows the timing for the same code encapsulated in a method, which is called several times. This timing applies to subsequent calls only. All timings were done automatically using the internal clock of the Zaurus.

shows the times it takes to load a construction containing a simple mechanical three bar linkage and calculate a locus in it, as shown in Fig. 1. *Load complicated locus* gives the times it takes to load a more complicated construction in which the linkage is extended by another bar. The resulting locus is more costly to calculate.

In all these benchmarks the Sun VM is a clear winner. We then concentrated our investigation on the instruction level. Cinderella, as a mathematical application, does a lot of basic calculations in integer and double arithmetic, so we started by measuring the performance of basic arithmetical operations.

Table 2 shows the results of a very elementary number-crunching benchmark; a single variable was manipulated 10 million times, every time adding, multiplying by, subtracting and dividing by a constant. As expected, floating-point arithmetic is much slower because it has to be done in software. On a G4-based laptop with a floating point unit this performance hit is not as hard.

What was unexpected for us is the poor performance of the Sun VM on its first encounter of the code. By encapsulating the code into a method and calling it several times we could improve the performance significantly as this triggers the optimizing just-in-time compiler. Still, the Sun VM is slower in this benchmark than the Jeode counterpart, which is in contrast to our perception with the Cinderella application.

	Jeode	Sun VM (first)	Sun VM (subs.)	Laptop
int	1.2 sec.	26.1 sec.	1.0 sec.	1.2 sec.
float	10.0 sec.	27.1 sec.	8.6 sec.	3.3 sec.
double	10.7 sec	33.6 sec	13.1 sec	3.4 sec.

Table 3
Basic calculations with method calls

We then designed another benchmark that explains this behavior. We used the same code and wrapped methods around the in-loop calculations, effectively adding 10 million additional method calls. From Tab. 3 we conclude that there is almost no penalty for the additional calls on the Sun VM, whereas the Jeode VM slows down significantly. The performance hit is the same for the float and the double test, and it is less for the integer test.

We conclude that the Sun VM is able to inline method calls and optimize the compiled code for faster execution. On the other hand, the Jeode VM has to carry out the full method calls, and the speed is dependent on the size of the arguments and return values.

The lesson we learned here is that an application like Cinderella that is both highly object-oriented and doing a lot of calculations benefits more from optimizing the OO-specific operations than from faster calculation. The Sun VM and its JIT is very good at optimizing object creation and inlining methods, two bottlenecks in object oriented programming.

3.4 Utilizing Class Extraction for a Common Code Base

Not all functionality of a desktop version of Cinderella has to be available on a PDA. For example, the built-in exercise editor, multiple viewport support or printing do not make sense on the small devices and can be left out. This saves valuable memory resources on the device.

A similar observation holds for the web page runtime that is used for exported constructions. The first version of Cinderella used a Java class extractor, *Jax* of IBM alphaworks, to create both the standalone application and the web page runtime from the same Java class files. This cut the jar file size by approx. 40% for the application and more than 60% for the runtime [27].

For the J2ME version of Cinderella we use the same approach. We switched from Jax to *DashO-Pro* [19], which is a commercial class extractor and obfuscator. Jax is no longer supported, and is no longer usable for the changed class formats of Java 1.4. Our experiences correspond to the experiences with the web runtime, and we recommend using products like Jax or DashO-Pro

for J2ME applications.

3.5 Inter-device Communication

Currently, there are two major wireless technologies to link a PDA to a desktop computer: 802.11b Wireless LAN and Bluetooth. Of these, 802.11b has the advantage of providing a full TCP/IP protocol stack by default and having a longer range. To the application, the PDA appears to be a normal networked computer. Both variants are available as PDA-sized add-on cards. We used the 802.11b approach for our experiments. However, supporting Bluetooth should work as well.

There are two basic ways to allow collaboration on a construction: One, have a special way to create a remote view on a construction and two, have complete and independent Cinderella instances on every device. Since our scenarios are based on wireless network links, which are high-latency and of varying reliability, the second possibility is better suited to our purposes. This is further supported by experiences with VNC, a tool to allow remote access to graphical desktops. It is available for PDAs, however, it is so slow that it is only barely usable.

There are different levels on which communication could happen. We decided to do so at the construction level, because not too much data needs to be sent and it seems the most natural choice.

Synchronizing different constructions presents interesting consistency problems; we resolved the issue by nominating the publisher of a construction the master. When a remote Cinderella wants to change the shared construction, it has to ask the master copy of the construction to do so. In our current version, we have only implemented very simple synchronization. It can happen that linked instances get out of sync; for these cases, we provide a manual sync operation. This is not desirable; however, this prototype works well enough to indicate remote collaboration is a useful feature for Cinderella.

Having independent instances implies that we can calculate and display what will happen as a result of the user's actions before the master copy is updated.

Recently, Apple introduced *Rendezvous*, a technology for networked devices in the same subnet to find each other. There is a Java 2 implementation available [6]. We were able to adapt it for Cinderella. This makes it very easy to connect to a published construction in the same network, without having to know IP or port numbers. The user can just select the desired construction from a list.

4 Conclusions and Further Work

Based on the tests we did with the Sun VM on the SL-C700, we can confidently say that on current hardware and with current JVMs, demanding Java programs run fast enough for real-world usage. Handheld devices will continue to become more powerful, therefore one can plan ambitious wireless classroom scenarios that are still a bit cumbersome on today's hardware. We urge didacts to think about reasonable uses of this new technology in advance.

From our experiences, we can also conclude that the Java programming language is well-suited for wireless classroom scenarios involving different devices and configurations. Since the binaries are platform-independent, the whole building and packaging process can happen on a powerful machine. The same source can be used to derive desktop as well as whiteboard and PDA versions of the software, with only small, if any, changes necessary in the code to accommodate the technical differences of the platforms. Furthermore, using TCP/IP is trivial in Java, facilitating network use and collaboration.

For the quality of a VM our benchmarks have shown that it is most important to do good optimization of typical code structures that are used in object oriented programming.

The user interface, however, does require attention and modifications in the code. A user interface targeted at desktop computers does not work well on pen-driven devices, especially if the screen size is very limited. The alternate user interface elements we implemented, *Scribbling* and *Cinderella Flow Menus*, together lead to a much improved user experience of Cinderella on the PDA.

Exclusive-access synchronization for multiple Cinderella instances on networked machines leads to interesting new problems. Displaying changes before they are committed to the master is an essential optimization to make the program appear responsive. However, this can lead to rollbacks when a remote Cinderella instance displays a change and only afterwards notices it cannot update the master instance. One solution could be to allow only one person at a time to make changes – similar to "having the chalk" in the traditional classroom.

Our final goal is the development of an integrated mathematics PDA. The three types of mathematical software used most in schools are spreadsheets, computer algebra software, and interactive geometry software. These three are now available in beta versions for PDAs. The crucial step here will be to link these applications together in a way that the user is able to switch seamlessly between them and transparently exchange their data.

5 Acknowledgments

We would like to thank Dan Stevens of Sun Microsystems for giving permission to publish the benchmark results of the early access implementation of the Personal Profile for Java for Zaurus. We would like to thank SHARP Europe, in particular Saskia v. Boxberg, for providing SHARP Zaurus SL-5x00 and SL-C700 test machines. Furthermore we thank Kurt Klaner of Casio for providing a Casio Cassiopeia E-125G for testing.

References

[1] Kent Beck. Extreme Programming Explained: Embrace Change, Addison-Wesley, 1999.

[2] Cinderella web site, `http://www.cinderella.de`.

[3] B. Dye. Mathsnet. `http://www.mathsnet.net/`.

[4] François Guimbretière, Terry Winograd: FlowMenu: Combining Command, Text and Data Entry. Stanford CS Technical Report CS-TR-2000-01. May 2000.

[5] Erik Hatcher and Steve Loughran. Java Development with Ant. Manning Publications, 2002. See also `http://ant.apache.org`.

[6] A.v. Hoff. A Java implementation of Rendezvous. `http://www.strangeberry.com/java_rendevous.htm`.

[7] Insignia Solutions. Jeode VM Information. `http://www.insignia.com/content/products/jvmProducts.shtml`.

[8] N. Jackiw.The Geometer's Sketchpad. Key Curriculum Press, Berkeley, 1991–1995.

[9] Java 2 Micro Edition web site. `http://java.sun.com/j2me/`

[10] Jetbrains, Inc. IntelliJ IDEA, version 3.0, see `http://www.intellij.com`.

[11] U. Kortenkamp. Foundations of Dynamic Geometry. Dissertation, ETH Zürich 1999. Available at `http://www.kortenkamps.net/papers/diss.pdf`.

[12] U. Kortenkamp and D. Materlik and J. Richter-Gebert. A beta package of Cinderella for the SHARP Zaurus series. `http://www.cinderella.de/en/info/zaurus.html`.

[13] U. Kortenkamp and J. Richter-Gebert. Geometry and education in the internet age. In *Proceedings of the ED-MEDIA & ED-TELECOM 1998 World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 790–799, Freiburg, March 1998. Association for the Advancement of Computing in Education.

[14] U. Kortenkamp and J. Richter-Gebert. Cinderella. In *Erfahrungen mit Java*, chapter 16, pages 381–401. dpunkt.Verlag, Heidelberg, 1999.

[15] J.-M. Laborde and F. Bellemain. Cabri-Geometry II. Texas Instruments, 1993–1998. Copyright Texas Instruments and Université Joseph Fourier, CNRS.

[16] D. Materlik: Using Sketch Recognition to Enhance the Human-Computer Interface of Geometry Software. Diploma thesis at the Freie Universität Berlin, 2003. Also available at `http://page.mi.fu-berlin.de/~materlik/DirkMaterlikThesis.pdf`.

[17] Motorola, Inc. and Sun Microsystems. Mobile Information Device Profile, v2.0 (JSR-118). `http://java.sun.com/products/midp/`.

[18] Parr, T. J. and Quong, R. W. ANTLR: a predicated-LL(k) parser generator. Software-Practice and Experience. Chichester: John Wiley & Sons, Ltd.. 25, No.7, 789-810, 1995. See also `http://www.antlr.org`.

[19] PreEmptive Solutions, Dasho-Pro, `http://www.preemptive.com/`.

[20] J. Richter-Gebert and U. Kortenkamp. Die interaktive Geometriesoftware Cinderella. HEUREKA-Klett Softwareverlag, Stuttgart, 1999.

[21] J. Richter-Gebert and U. Kortenkamp. The Interactive Geometry Software Cinderella. Springer-Verlag, Berlin Heidelberg New York, 1999.

[22] Complexity Issues in Dynamic Geometry. J. Richter-Gebert and U. Kortenkamp In *Foundations of Computational Mathematics (Proceedings of the Smale Fest 2000)*, World Scientific, 2002.

[23] Rojas, Raúl; Knipping, Lars; Raffel, Ulrich; Friedland, Gerald: Elektronische Kreide: Eine Java-Multimedia-Tafel für den Präsenz- und Fernunterricht. Inform., Forsch. Entwickl. 16, No.3, 159-168 (2001).

[24] Sun Microsystems. PersonalJava specification (end of life announcement). `http://java.sun.com/products/personaljava/`.

[25] Sun Microsystems. Personal Profile for Zaurus early access version. `http://developer.java.sun.com/developer/earlyAccess/pp4zaurus/`.

[26] TAO group web site. `http://tao-group.com/`.

[27] F. Tip, C. Laffra, P. F. Sweeney, and D. Streeter. Pratical experience with an application extractor for java. In *Proceedings of the 14th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA '99)*, Denver, Colorado, November 1999. ACM.

[28] D. Wells, `http://www.extremeprogramming.org/rules/releaseoften.html`.