To appear in Proceedings of the
ACM Special Interest Group on Computer Science Education Conference
Information Technology in Computer Science Education, Madrid, Spain, June 2008

Distributed Application Launching for High Quality Graphics in Synchronous Distance Education

J. Mark Pullen
Department of Computer Science
George Mason University
Fairfax, VA 22030
+1.703.993.1538
mpullen@netlab.gmu.edu

Jim X. Chen
Department of Computer Science
George Mason University
Fairfax, VA 22030
+1.703.993.1720
jchen@cs.gmu.edu

ABSTRACT
Previous ITiCSE papers have reported on Network EducationWare (NEW), an open source software system that supports synchronous and asynchronous distance education easily and inexpensively via the Internet. This paper reports on an innovative capability recently added to NEW that enables simultaneous execution of applications on all participating computers in both Windows and Linux systems, and explains how we have used this capability to teach a course in Computer Graphics online. Teaching graphics in this way previously was impossible because transmitting the visual output of graphics programs in real time with good quality is too demanding for normal Internet connections. This problem was solved by using the NEW application launcher to invoke Java classes, provided by the instructor, on all student computers simultaneously. The capability works with modest Internet capacity and also is captured in recorded sessions for asynchronous use. We explain how the NEW capability works and describe its use in online teaching of Computer Graphics.

Categories and Subject Descriptors
K.3.1 **[Computing Milieux]:** Computer Uses in Education – distance learning

General Terms
Design, Performance, Human Factors

Keywords
Internet distance education, accessibility, application launching

1. INTRODUCTION
A groundswell of demand for online course delivery began in the 1990s and continues to grow [13,15], because the Internet provides great accessibility to students who don't have a residence near a lecture hall for all of the courses they want to attend. Delivery technology varies but most offerings can be characterized either as *asynchronous*, where students and instructors normally communicate through a written or recorded medium (most often, webpages), or *synchronous*, where students receive classes as they are delivered, in real time. Goodwin [5], Pullen [8,9] and Snow *et al.* [14] have described the cost and time benefits of simultaneous delivery to the classroom and the network, which we call *simulteaching*.

At George Mason University (GMU) we began experimental synchronous online teaching in the mid-1990s [8] and eventually went on to offer a Systems and Networking subset of the Master of Science in Computer Science online [12], using locally integrated multimedia tools in the Network EducationWare (NEW) system described below. One of the continuing frustrations of our program has been the inability to teach subjects online that require high resolution and/or motion graphic display. This would be true even if we were willing to insist that the student have a very high capacity ("broadband") Internet connection; but we don't require that because it runs against our goal of accessibility for all students. The very worst case in this regard is teaching Computer Graphics online, since good examples of graphics programming almost inevitably demand both high resolution and considerable motion.

The general capability needed for group display of computer graphics is called *application sharing* and is implemented by exporting the multimedia output of a program as rapidly as possible over the network used for group communication. The ability to export high resolution motion graphics in this way is beyond almost all home Internet connections. The solution we have adopted in *application launching* (AppLaunch), where the application to be run and its data are propositioned on all computers and invoked simultaneously. The remainder of this paper describes how AppLaunch has been added to NEW and how it was used to teach a Computer Graphics course. Section 2 describes the current production version of NEW; section 3 describes how AppLaunch was added to NEW, and section 4 describes the course. We conclude with a summary and some observations about future use of NEW with AppLaunch.

ITiCSE'08, June 30–July 2, 2008, Madrid, Spain.

2. NETWORK EDUCATIONWARE

NEW was developed in Pullen's laboratory [11,14]. It is the third generation of synchronous online education software used in teaching Computer Science at George Mason University (GMU). From the first two, we learned a great deal about what works and what does not work. Based on those lessons, and grounded in the philosophy that synchronous Internet teaching could be much less time-consuming and expensive than the current state of practice allows, we set out to create an open-source suite of software that relies as much as possible on quality software tools created by other parties [7]. The first five years of experience had taught us that the following properties are essential for simulteaching [12]:

- Quality support for at least audiographics (voice, slides and real time annotations), using open media standards; video also is desirable, although not essential [10].
- Ability to support audiographic streaming over standard 56 kb/s modem connections to the Internet.
- Effective support to teach students simultaneously in the classroom and online.
- Audiographics recorded during class and accessible from a server, either by streaming or file download, using the same client suite as live access.
- Web-based access and software load/checkout for ease of operation.
- Availability on multiple platforms for ubiquitous use (Windows, Linux and Macintosh systems).

The NEW system software that we have assembled to meet these requirements is represented in Figure 1 and summarized in the remainder of this section. Executable and source code for all system components is openly available for academic purposes.

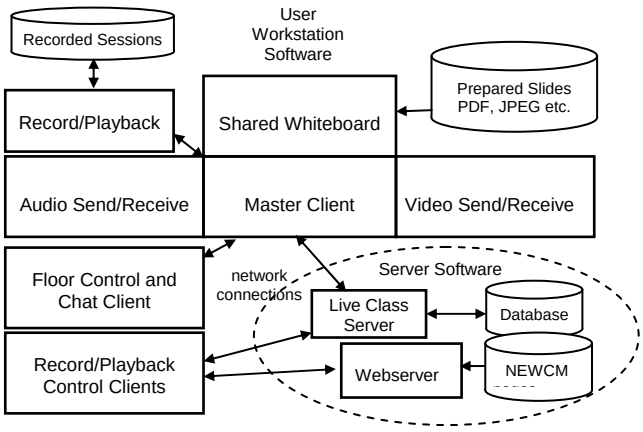

Figure 1. System organization of NEW components

2.1 Client Software

The multimedia interface software in NEW derives from a variety of sources and runs on Windows and Linux-x86 platforms, and on late-model Macintosh platforms running Apple's Parallels™ with either a Linux or Windows virtual system. The composite user interface for all tools is shown in Figure 2.

We consider the *Speak Freely Internet Audio* software to be the most important component in the NEW system, both because it is essential to the students' learning experience and also because conveying voice with good quality over the Internet at low data rates presents a big challenge. SF is capable of passing good voice quality over the Internet, using a standard sound interface, and requiring only 20 kilobits per second of network capacity. We have added a graphic interface that provides all needed user functions in one easy-to-use package.

The *Whiteboard* provides the other key element for teaching online: graphics. It will display a precomposed graphic prepared in several open formats: text, HTML, JPEG, and Adobe Portable Document Format (PDF); and it will convert LaTeX, OpenOffice, and Microsoft PowerPoint formats to PDF and JPEG automatically. It also will capture a JPEG image of any other window on the machine and export that to the other whiteboards in the group, a crude sort of application sharing that operates at about one frame every ten seconds but is not adequate either in resolution or motion rate. The precomposed graphics and captured images can be annotated during class with lines, rectangles, ellipses, handwriting, and text in any color, a very useful feature for maintaining the attention of the visual learner. The whole image is saved and can be called back during class in addition to being recorded. We prefer to use the whiteboard with a Tablet PC interface so that it becomes a surrogate chalkboard.

The optional *Video* tool is capable of multiple network formats, including standard H.323 conferencing. A typical delivery rate for NEW is two frames of 320 by 240 pixels per second, although rates up to 30 frames per second are possible. While we have found that, for teaching Information Technology, video provides a marginal benefit at relatively high cost, as reported by Pullen [10], we offer it as an option to students who have high quality Internet service.

The *Master Client* encapsulates data from the multicast applications into TCP tunnels to the Live Class server, prioritized according to the importance of each multimedia tool (audio first, whiteboard second, video last). It can support a viable class connection over a 56 kb/s modem, without video. The combination of clients and their network configuration established by the master client is controlled by a configuration file downloaded from the supporting webserver at the beginning of a NEW session. If software updates are indicated, the master client also downloads and installs them.

The NEW *Floor Control* shows the participants in the session, controls access to the virtual classroom "floor," provides for text questions to the instructor and text chat among the participants, and accepts URLs from the floor holder for browser launch on all participating client systems. It supports a "virtual hand raising" mode for lectures and an "anyone can have the floor" mode for seminars and meetings.

NEW *Record and Playback Clients* control their respective servers. They feature VCR-like button icons and an elapsed time readout. The playback control also is capable of jumping forward and backward to the next slide in the presentation. Recordings require about 5 Megabytes of disk per hour of class.

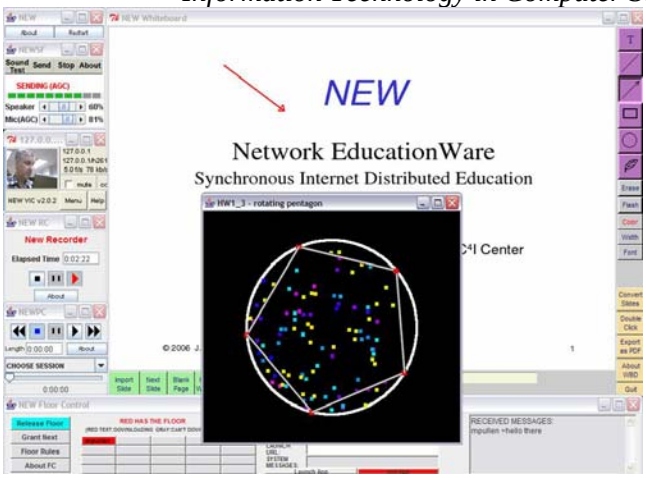

Figure 2. NEW user interface with launched Java application

2.2 Server Software

The *Live Class Server* is the core of the NEW system. It implements group communication over the general Internet among a group of participating workstations by accepting a data stream combining transmissions from multicast conferencing tools on the floorholder's workstation and sending copies to all participants' workstations. It provides access control using either the system database or an external authentication service. As delivered from our website it will support 20 users on a 1 GHz Linux system; however, it can be configured for up to 50 users on a 3 GHz system.

The *Record* and *Playback* servers are used to create and play streaming recordings that capture the information sent by NEW clients over the Internet from the instructor's workstation. Playback is accessed through the same software suite and also can be performed offline by downloading the recording files. Each segment of an online playback can be Web-linked as a URL.

Another key aspect of the server is the NEW Course Management Webpages, which provide for effective management of the mass of detail involved in supporting multiple courses. Various pages provide teaching and learning functions, support and course management functions, authentication, and system administration functions. A single portal page provides access to all of these facilities, as described in [11]. Our webpages are hosted on an Apache Webserver that supports the PHP language needed for our webpages and MySQL or any database system supporting the standard Structured Query Language (SQL). Web-based support provides ubiquity and portability. It also makes possible data access over the network that we use to implement the chat room feature.

3. APPLICATION LAUNCHING IN NEW

The key to adding AppLaunch in NEW was the recognition that data files too large to be streamed in real time can nevertheless be streamed in background, using unneeded network capacity. In the extreme case the onus for doing this can be placed on the student, by requiring that the files be downloaded before class. However, NEW has a more elegant solution.

Figure 3 shows the data flow for NEW AppLaunch. The instructor creates a file *APPLOCS.txt* defining the applications to be used and where they are to be found within the directory structure of all platforms to be used (i.e. Windows and Linux). We have used AppLaunch almost entirely for Java, however any application that is installed at a known location could be used.

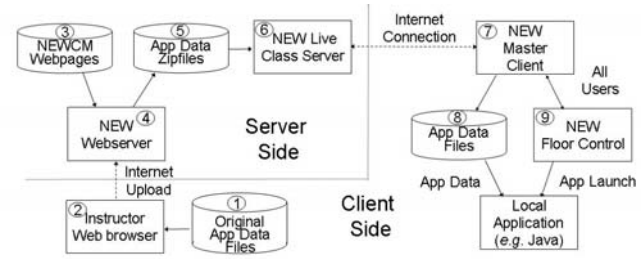

Figure 3. Information flow for NEW AppLaunch

3.1 Server Side Operations for AppLaunch

The instructor uses the NEWCM webpages to upload the application data (AppData) files to the server. The files are uploaded to a single directory per course. The directory also contains a default *APPLOCS.txt* for Java and JPEG, which supports the simplest case. As each file is added to the directory, the zipfile representing that directory is updated. This results in a set of zipfiles, one per course, that are available to the NEW Live Class Server. When a student connects for a course that is configured for AppLaunch, the server sends the AppLaunch zipfile name, size, and last modified date to the NEW Master Client. If the client doesn't have that version, it requests download of the current zipfile for the course. The server streams the zipfile to the client in background, *i.e.* without interfering with the synchronous data streams for audio, whiteboard, and video. The server also sends download status to the NEW floor control.

The NEW *Record* server runs on the instructor's computer. It was modified for AppLaunch to add the current AppData zipfile to the end of each recording. The NEW *Play* server can either run at the server site or on the user's computer. It was modified for AppLaunch to copy the AppData zipfile out of the recording into the local NEW directory. In this way the recording is played with the data that was in the AppData directory when the recording was made.

3.2 Client Side Operations for AppLaunch

The NEW *Master Client* requests user permission to save the download zipfile. On approval, it stores the zipfile in the NEW directory and date-stamps it with its date or origination so it can be identified later, to avoid unnecessary downloads, and then unzips it into to client's duplicate of the course directory. Users who have a download in progress are indicated on the floor control user panel so the instructor is aware that AppLaunch will not work for them. When the floorholder (normally the instructor) chooses to launch an application, it is only necessary to click on the *Launch App* button and select from among the data files in the download in order to launch the application associated with that data. When finished running that application, the *Kill App* button kills it on all participants' screens. Since good quality applications run at the same rate, independent of processor clock rate, the application will run on all screens at almost exactly the same time. In the course described below, the data files we used were Java class files; while programmers may consider these to be executable code, they are data for the Java Virtual Machine.

Figure 2 shows a Java graphic running under the NEW AppLaunch capability.

3.3 Security

There is a security issue involved with application launching. The students who participate are in essence giving the instructor permission to run any data that is loaded through the NEW system on software that exists on their computers. This includes Java classes, which are very nearly as powerful as native executable code. However, the same students, when using NEW, already are trusting the NEW software to behave properly on their computers. Thus the additional level of trust required is for the student to have confidence the instructor will not load pernicious data such as misbehaving Java classes or JPEG files crafted with a "virus" to attack the student's computer. This risk is limited to the instructor and NEW system staff, since only they are able to upload data files. Students, when given floorholder status, are able to launch applications from the AppData directory, but they can't change its contents except on their own computers.

3.4 Limitation

The AppLaunch approach does have one limitation: it is not possible for the floorholder to enter data at run time into the instances of the program that run on distant Internet computers. The two alternatives for this are to either have the online student enter the data or to put a file in the AppData directory that is read by the launched application. For example, a Java program could read a text or other data file in this way.

4. TEACHING COMPUTER GRAPHICS USING APPLICATION LAUNCHING

This section reports on a pilot course taught with NEW AppLaunch capability, CS 652 Computer Graphics Fall 2007.

4.1 Course Organization

Chen has taught Computer Graphics every semester for over 12 years. His course covers graphics principles and programming in 15 three-hour classes. Topics in the course include graphics hardware, antialiasing, transformations, viewing, illumination, blending, texture mapping, color models, curves, surfaces, scene graph structure, and virtual environments. Over the years, he has adopted several different textbooks [2,3,4,6,16]. Since 2003, the curriculum in the GMU Computer Science Department has adopted Java as the platform for Computer Science major instruction. Chen therefore has adopted use of JOGL (https://jogl.dev.java.net/) as a teaching graphics programming library and Java platform. One of the instruction methods is using interactive graphics examples to explain concepts in 3D graphics itself. Chen has coauthored a new Computer Graphics textbook [3] that includes a complete set of JOGL/Java 3D animation examples embedded in the lecture notes.

4.2 Structure of the Java Class Library

The Java class library includes JOGL examples is shown in Figure 4. The first example J1_0_Point that draws a point is extended from the Frame class of the abstract window toolkit in the original Java library. All the other classes are extended from previous classes, inheriting their existing methods. Student homework can be extended from the sample programs as well. This helps students focus on the required problem by omitting other surrounding functions and setup. Homework sample answers are provided after students submit their answers.

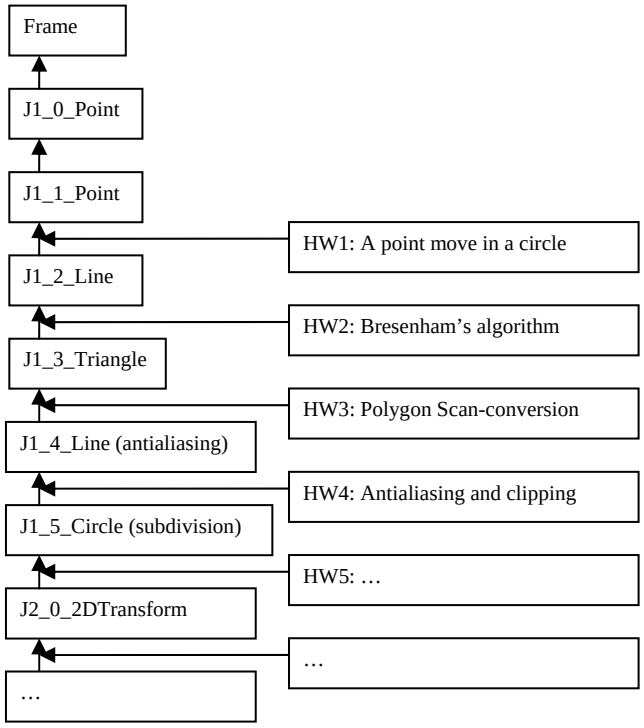

Figure 4. Structure of the Java class library

4.3 Teaching Experience with AppLaunch

Development of NEW AppLaunch and its application to CS 652 was an experimental effort between Pullen's laboratory (the developers) and Chen (the professor). Chen's initial use of NEW resulted in frustration because previously 3D animation could only be achieved through importing a window to the whiteboard by converting frames of animation into images and sending the converted images across the network to distant students at a rate of one frame every few seconds. While this might have been very satisfactory for static output, it failed to capture the effect of animation. In effect, animation was not possible when only intermittent frames are displayed. The authors solved this problem together, resulting in a prototype of NEW AppLaunch to support CS 652 in a matter weeks.

With AppLaunch, Chen was able to teach by launching Java applications with simple mouse clicks as standalone applications, while the distant students all viewed the same animation. This allowed the powerful approach of teaching by example, *i.e.* explaining concepts using real-time animation, a technique Chen had developed to deal with teaching the complexity of Computer Graphics. As a result, NEW has another advocate who is interested in exploring distance education. Chen is able to combine teaching by example with the whiteboard's ability to retrace discussed annotations. This underscores our philosophy of distance education: the lecture is not just for the audience in the classroom, but also for those many distance receivers at home or at work, who are able to participate in class virtually, receiving animation programs that are delivered transparently and invoked

when the instructor is ready to use them to support the teaching process.

5. CONCLUSIONS

We have reported on a powerful approach to synchronous online education: application launching. By adding this capability to the existing NEW system, we were able to teach a Computer Graphics course online with excellent illustrations of programming techniques that ran in real time on all network participants' computers under control of the instructor. We also reported on the pilot course taught in this way, which worked very much to our satisfaction and for the first time allowed distance education students to participate in a Computer Graphics course.

The experience reported here has increased our confidence that we can continue to expand access to our teaching by employing innovations in online teaching technology. Thus the student who formerly could not afford the time or travel to attend classes will no longer face these limitations. This will allow us to "do well by doing good," in the sense that we will expand our teaching horizons even as we meet the needs for more students, a worthwhile achievement in these days of declining Computer Science enrollments.

6. REFERENCES

[1] Angel, E. *Interactive Computer Graphics: A Top-Down Approach with OpenGL*, Addison Wesley, 2003.

[2] Chen, J. *Guide to Graphics Software Tools*, Springer Verlag, 2002.

[3] Chen, J. and E.Wegman, *Foundation of 3D Graphics Programming Using JOGL and Java3D*, Springer Verlag, 2006.

[4] Foley, J., A van Dam, S. Feiner and J. Hughes, *Computer Graphics: Principles and Practice*, Second Edition in C, Addison-Wesley, 1995.

[5] Goodwin, C. and Bowman, M., Is the bottom line of online out of line? Calculating the total cost of online courses in Technology Curricula, *Proceedings ASEE Annual Conference 2004* (Salt Lake City, UT, June 2004)

[6] Hearn, D. and M. Baker, *Computer Graphics*, C version, 2nd edition, Printice-Hall, 1996.

[7] Macedonia, M. and Brutzman, D., Mbone Provides Audio and Video Across the Internet, *IEEE Computer 27*, 4 (Apr. 1994), 30-36

[8] Pullen, J., Synchronous Distance Education and the Internet, *Proceedings Internet Society Annual Conference 1998* (Geneva, Switzerland, July 1998), published online at http://www.isoc.org/inet98/ proceedings/ 4b/4b_1.htm

[9] Pullen, J., The Internet lecture: converging teaching and technology, *ACM Special Interest Group on Computer Science Education (SIGCSE) Bulletin* Vol 32 No 3 (Sep. 2000), 101-104

[10] Pullen, J., Applicability Of Internet video In distance education For engineering, *Proceedings IEEE Frontiers in Education 2001* (Reno, NV, October 2001), T2F-14–T2F-19, online at http://fie.engrng.pitt.edu/ fie2001/papers/1242.pdf

[11] Pullen, J. and McAndrews, P., A Web portal for open-Source synchronous distance education, *IASTED Journal on Advanced Technology for Learning 2*,1 (Jan 2005), International Association of Science and technology for Development, Calgary, AB

[12] Pullen, J., Scaling up a distance education program in computer science, *ACM Special Interest Group on Computer Science Education (SIGCSE) Bulletin 38*, 3 (Sep 2006) 33-37

[13] Sloan Consortium, Sizing the Opportunity: The Quality and Extent of Online Education in the United States, 2002 and 2003, online at http://www.sloan-c.org/resources/ overview.asp

[14] Snow, C., Pullen, J. and McAndrews, P. (2005), An Open-Source Web-Based System for Synchronous Distance Education, *IEEE Transactions on Education 48*, 4 (Nov. 2005), 705-712

[15] Wilson, J., After the Fall: Lessons of an Indulgent Era, plenary presentation, Distance Education 2003, University of Wisconsin, unpublished, available online at http://www.jackmwilson.com/ArticlesTalks/eLearning-Wisconsin2003.ppt

[16] Woo, M., J. Neider, and T. Davis, *OpenGL Programming Guide Version 2.1*, Addison Wesley, 2007