A Cross-Platform Framework for Educational Application Development on Phones, Tablets, and Tablet PCs

Wade Fagen Sam Kamin
Department of Computer Science
University of Illinois
Urbana, IL 61801 USA
{wfagen2, kamin}@illinois.edu

ABSTRACT

We describe a major extension of the SLICE framework, called SLICE 2.0. This new framework allows developers to program an application once and deploy the app natively on all major PC platforms (Windows, Mac, and Linux), tablet PCs, and Android based cell phones and tablets. The framework specifically provides functionality to create pen- and touch-enabled applications focused on active and collaborative learning. We introduce two classes of applications that are already in use in classrooms using the SLICE 2.0 framework and discuss several hardware devices running the SLICE 2.0 framework.

KEYWORDS

Software Framework, Active Learning Tool, Collaborative Learning Tool

1. INTRODUCTION

In recent years, a number of researchers have studied the effectiveness of active and collaborative learning technology in classrooms [1, 2]. These technologies often make use of laptop computers, tablet PCs, iOS or Android-based cell phones and tablets, and electronic voting systems (EVS) [3]. Many higher-education institutions have adopted various forms of EVSs, including i>Clickers, eInstruction clickers, and others [4]. Case studies and research alike have shown EVSs to be effective, though often note limited functionality as a drawback to complex active-learning exercises [3].

As a contrast to EVSs, researchers have developed and studied active learning applications on pen-enabled (tablet PCs) and touch-enabled (cell phones and tablets) devices [3]. Nearly all research we have found uses a single, custom-developed application for the task at hand running on a homogeneous platform.

Realizing that the modern classroom is filled with cell phones, tablets, laptop computers, and other intelligent devices, we have developed and successfully deployed a cross-platform, cross-device framework for developing applications primarily focused on educational use. The original framework, SLICE, or Students Learn In Collaborative Environments, was a software framework originally developed in 2006 and has been continually used in classrooms since then [5]. The applications originally developed using the SLICE 1.0 framework allowed the quick development and deployment of educational applications on Windows-based Tablet PC devices.

In this paper, we introduce SLICE 2.0, a significant milestone in the development of the SLICE framework. SLICE 2.0 builds on the SLICE framework by preserving nearly the entire existing API that developers have used to develop SLICE applications while allowing applications developed on SLICE to be

deployed on heterogeneous hardware. In this paper, we will discuss some of the hardware devices on which SLICE 2.0 has been deployed and briefly introduce several of the successful applications that have leveraged the framework.

Supported Platforms
Windows XP, Vista, or 7 Windows Server 2000, 2003, or 2008 Mac OS X 10.6+ Linux supporting Java 1.5+ Android 2.2+
User Scripts
JavaScript
User Interface Design
XML

Table 1: Overview of Slice requirement

2. FRAMEWORK

The SLICE framework is based on the Model-View-Controller [7] design pattern, pushed to an extreme: The model is an XML tree that includes interface element (buttons, panels, ink strokes). The controller – that is, the code that is invoked when events like "ink stroke entered" or "button clicked" or "message received from another computer" occurs – does nothing but make changes in the model. Unlike most versions of MVC, this code does not have to indicate whether those changes in the model must trigger changes in the "view;" that is handled automatically. Thus, each event triggers the execution of a small piece of code that changes a model whose structure is quite simple. These small pieces of code are written in JavaScript; using a scripting language provides well-known advantages of convenience and short development times (for small scripts).

This structure allows for a high level of portability by separating application-specific functionality (the scripts that modify the model) from platform-specific functionality (capturing events and rendering the various components of the model). The latter is contained in the "core"

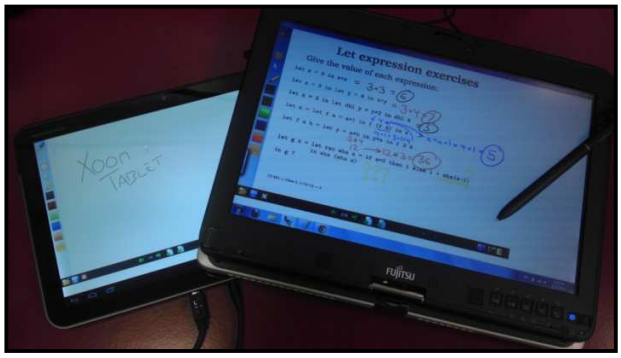

Figure 1: Two student instances of the SLICE Lecturer Application on heterogeneous platforms. Left: Android tablet; Right: Windows 7 Tablet PC.

code written once for each platform, while the former is the per-application code that runs on all platforms.

To an application programmer, SLICE provides a rich set of callbacks to run the JavaScript code the programmer designed. Additionally, SLICE provides over eighty functions available to application developers to interact with the "model" and "view". For example, the XML necessary to show a **`Button`**, and register a callback when the **`Button`** is clicked by the user, is simply the following:

```
<Button OnClick="ButtonClicked" />
```

The JavaScript function that is invoked when a button is clicked by the user could then be:

```
/* ButtonClicked(): Invoked by SLICE
   when the user clicks the button
   on the screen. */
Function ButtonClicked() {
    Sender["Text"] = "Clicked";
}
```

In the example above, the text displayed on the button that was clicked will be modified to the string "Clicked". Detailed documentation on every callback, API call, and variable is provided on the SLICE website [6].

3. APPLICATIONS

Since 2006, over a dozen known applications were developed using the SLICE framework. In the past year, we have deployed two notable classes of applications to meet specific educational objectives based on what we have learned. This paper will introduce these applications and how they are in use today.

These applications are samples of what can be developed and focus on specific learning objectives and needs of a class. We encourage the reader to use these pre-developed applications themselves or develop their own application using the SLICE framework for cross-platform deployment.

3.1. Lecturer Application

The "Lecturer Application" is a generic phrase describing the most broadly used SLICE application since 2006. As part of this application, small changes are made nearly every semester to meet instructor feedback and new educational goals. The original motivation came from instructors noting the advantages of having pen input on projected slides, as compared with using a whiteboard, transparencies, or PowerPoint presentations. With that in mind, the original Lecturer Application provided a way for PowerPoint or PDF slides to be loaded into a SLICE application on a pen-enabled Windows tablet. The instructor would connect his or her tablet up to the projection system and could write, draw, or otherwise annotate their slides with their tablet PC [7, 8].

In classrooms with a house computer, the Lecturer Application can be run in "networked mode." The display on the lecturer's tablet is wirelessly cloned on the room display, allowing the lecturer to move about the classroom, tablet PC in-hand, and even allow students to use the tablet to write answers to

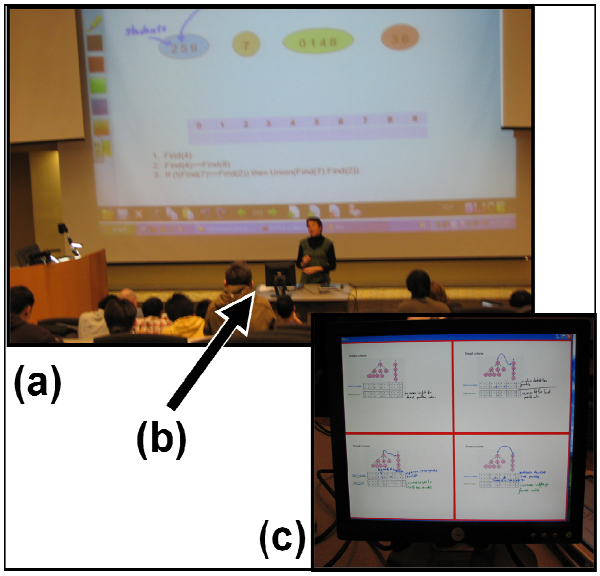

Figure 2: An iteration of the Lecturer Application in use in a classroom using a separate dashboard display. (a): An on-going lecture in a CS2-style course. (b): The dashboard display monitor. (c): The dashboard display visible to the lecturer.

questions. This application proved to be a useful tool in creating an active learning environment and a more collaborative classroom.

With SLICE 2.0 in mind, this application was extended several more times focusing on increasing active learning. We are currently studying three specific variations on this application. All three variations focus on large classrooms where some students have a SLICE-enabled device and actively participate in the lecture by taking notes or answering problems proposed by the lecturer.

The first application allows the lecturer to *constantly* monitor a subset of students. It employs an additional display, placed in a location where the lecturer, but not the students, can see it. That display shows a dashboard of all of the current students' notes. Figure 2 shows a dashboard with four students' work displayed. The instructor can gauge student progress on a question, answer questions posed by the student through the tablet, or gauge understanding of a topic by the students' notes. This application can be helpful in an active learning environment

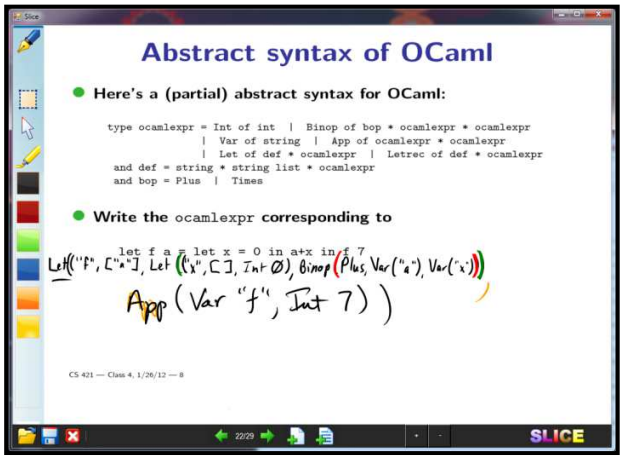

Figure 3: A student view of a Lecture Application where a student's work has been appended by the instructor.

(see Figure 2), but it is mainly intended for a more traditional classroom; the primary feedback to the lecturer is the notes that students are taking during the lecture.

Another application focuses on providing the instructor feedback *immediately after* the class, which can then be used to guide subsequent classes. Instead of a student's work being shown to the instructor, the student uses a special interface to leave questions on the slides. At the end of lecture, the instructor can review those questions.

The third application addresses classes that employ active learning, where a considerable amount of class time is devoted to student problem-solving. This is similar to the first application (the "dashboard"), but the dashboard resides on the lecturer's own tablet, and student work can be seen only by changing modes. In "monitoring mode," the lecturer can scroll through the students who are using tablets and see their progress on the current in-class exercise. The instructor may then write directly on a student's slide privately (in effect, sending a message to that student), but the more important feature is that he can show a student's slide on the classroom display. This use of a peer's answer publicly allows for the entire class to collaborate on an answer with the instructor being able to fill in errors or incomplete portions of the student's answer. Figure 3 shows a student machine's display where the instructor added to the answer the student originally wrote. More details on this application are given in [9].

With these applications, we are exploring the possibilities for enhancing instruction, especially in large college classrooms, by using pen-enabled computers. A major barrier to deployment is the cost of these devices. SLICE 2.0 addresses this issue by allowing these apps to run on less expensive platforms (see section 4). As they are used in the classroom, we will report significant successes or failures of these applications in future work.

3.2. Code Review Application

During the past several years, an increasing number of undergraduate programs have begun to include a course designed to build a student's individual programming skills. Students in the Computer Science department at the University of Illinois are required to take such a course, called CS 242: Programming Studio, detailed in a 2007 publication [10]. The key feature of this course is the two-hour discussion section each week, where students present their programming assignment for the week. Each student presentation lasts 20-25 minutes and focuses on the program's design and source code.

To facilitate conversations on source code, each discussion section meets in a lab equipped with a large TV display. Students traditionally would connect their own laptop to the display and discuss their design and source code by pointing to the display or using their mouse to highlight specific code regions.

As part of an effort to increase peer collaboration among the six students participating in the discussion section and the discussion moderator, a SLICE application was developed to aid in code review. The setup of this application consists of each student having

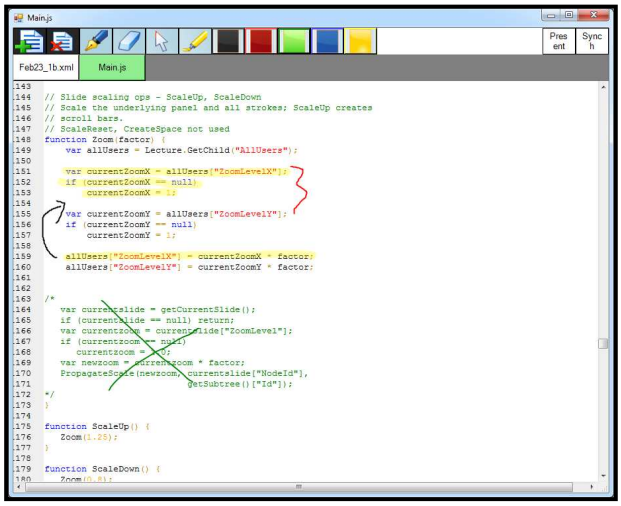

Figure 4: The Code Review application with several users highlighting and annotating the source code.

a SLICE-enabled device running the Code Review application and a server Code Review application running on the machine directly connected to the TV display. When a student is the presenter, his or her source code is displayed both on the TV display and every other student's device.

During the discussion of the code, the presenter, any student, or the moderator may write or highlight on any of the source code being presented by the presenter. These annotations are seen by every user and displayed on the monitor, making this application a sort of "shared whiteboard" with the ability to load source code (with syntax highlighting) as the ``background image'' on the whiteboard. We have preformed several detailed studies on the introduction of this application into CS 242. Early results suggest that it has made the code review discussions more efficient and more engaging. As we continue to analyze the results, we will report on the full results of these studies.

4. HARDWARE PLATFORMS

One of the primary advantages that SLICE provides that is unique in the space of educational software frameworks is the ability to code an application a single time and have it run on multiple platforms. While not all applications may be optimal for every platform (for example, it is difficult to write full sentences on cell-phone screens), the ability to give students access to the technology at no added cost to them or the university may prove key for wider adoption of sophisticated active and collaborative learning tools.

As part of the development of SLICE, we have tested the framework on a variety of devices, including:

- **Fujitsu Lifebook T730 Tablet PC**: With a Wacom active digitizer on a 12.1" display, this tablet PC is the easiest to write or annotate on and is the standard we use for comparison to other devices.
- **Motorola Xoom Tablet**: At 10.1", this Android tablet is nearly the size of many tablet PCs. At a fraction of the cost of a tablet PC, SLICE applications run as smoothly on the Xoom as the Fujitsu tablets. However, with only capacitive-touch input, some users feel drawing and annotating slides is natural but any form of writing proves difficult.
- **HTC Flyer:** Featuring a 7" display and an active digitizer, this tablet has provided the benefits of the pen-input at a substantially cheaper price than tablet PCs. While the Fujitsu tablet features a slightly smoother and more accurate active digitizer, the ability to write on the screen feels much more natural than writing on the Xoom tablet. Unlike the Fujitsu, this tablet also allows for touch-gestures, making tasks like zooming much more natural.
- **HTC EVO 3D:** With a duel-core 1.2 GHz processor, this cell phone is nearly as powerful as many of the tablets. However, the small (4.3") capacitive screen makes simple touch gestures intuitive but writing nearly impossible.

Additionally, we are particularly interested in continuing to follow the development of

Android tablets with active digitizers (devices similar to the HTC Flyer discussed above). At as little as a third the cost of a traditional tablet PC and with much of the same functionality, these devices may be the future hardware used in technology-enabled classrooms.

5. CONCLUSION

We have described the SLICE 2.0 software framework for developing cross-platform applications. The framework allows for a programmer to program an application once and deploy the application natively on a Windows, Linux, or Mac computer, a tablet PC, and an Android-powered cell phone or tablet. We have successfully deployed four applications in classrooms at The University of Illinois and SLICE is used on a daily basis in several undergraduate courses in the Computer Science department.

Building on the experience of over a dozen existing SLICE 1.0 applications, we preserved the most useful features while extending SLICE to run on a heterogeneous set of devices. Specifically, we described the classic Model-View-Controller (MVC) design pattern that is central to SLICE application development. Application developers need only to layout the desired graphical user interface components using XML and program application logic in JavaScript.

We encourage the reader to try out any of the applications described in this paper or to develop their own SLICE application. The applications described in this paper, the framework itself, and complete documentation can be found on the SLICE website at **`slice.cs.illinois.edu`**.

REFERENCES

[1] M. Nakakuni, M. Okumura and S. Fujimura, "Construction of a Collaborative Learning Environment through Sharing of a Single Desktop Screen," in *International Conference on Frontiers in Education: Computer Science and Computer Engineering*, 2011.

[2] C. H. Crouch and E. Mazur, "Peer instruction: Ten years of experience and results," *American Journal of Physics*, vol. 69, no. 9, pp. 970-977, 2001.

[3] K. Nahrstedt, L. Angrave, M. Caccamo and R. Campbell, "Mobile Learning Communities – Are We There Yet?," Information Trust Institute, University of Illinois at Urbana-Champaign, 2010.

[4] G. E. Kennedy and Q. I. Cutts, "Construction of a Collaborative Learning Environment through Sharing of a Single Desktop Screen," *Journal of Computer Assisted Learning*, vol. 21, no. 4, pp. 260-268, 2005.

[5] S. Kamin, M. Hines, C. Peiper and B. Capitanu, "A System for Developing Tablet PC Applications for Education," in *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, 2008.

[6] SLICE Research Group, "SLICE: Students Learn In Collaborative Environments," [Online]. Available: http://slice.cs.illinois.edu/.

[7] R. Anderson, "Beyond PowerPoint: Building a New Classroom Presenter," *Syllabus Magazine*, June 2004.

[8] M. Wilkerson, W. G. Griswold and B. Simon, "Ubiquitous Presenter: Increasing Student Access and Control in a Digital Lecturing Environment," in *Proceedings of the 36th SIGCSE technical symposium on Computer science education (SIGCSE 2005)*, 2005.

[9] S. Kamin and W. Fagen, "Supporting active learning in large classrooms using pen-enabled computers," in *Proceedings of the 2012 International Conference on Frontiers in Education: Computer Science & Computer Engineering (FECS 2012)*, Las Vegas, Nevada, 2012.

[10] M. Woodley and S. N. Kamin, "Programming Studio: A Course for Improving Programming Skills in Undergraduates," in *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, 2007.

[11] E. Spertus, M. L. Chang, P. Gestwicki and D. Wolber, "Novel Approaches to CS 0 with App Inventor for Android," in *Proceedings of the 41st ACM technical symposium on Computer science education* , 2010.

[12] G. E. Krasner and S. T. Pope, "A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System," in *System*, 1988.