Using the PyMT toolkit for HCI Research

Thomas Hansen
University of Iowa
tehansen@cs.uiowa.edu

Christopher Denter
University of Koblenz-Landau
cdenter@uni-koblenz.de

Mathieu Virbel
txprog@gmail.com

Abstract
We discuss the use of PyMT, a post-WIMP multi-touch UI toolkit, for the purposes of performing research on human computer interaction with multi-touch and similar novel input technologies. We present a short overview of PyMT as a software platform and line out several key features that are of benefit for tasks arising in HCI research. We argue that PyMT's use of a dynamic language and design goal of enabling rapid prototyping, it's multi-input support, and integration with tools for dealing with quantitative data make PyMT an ideal tool for anyone involved in creating software interfaces in a research setting.

INTRODUCTION

Multi-touch and other new input technologies are becoming widely available. Great diversity in input hardware brings many challenges in HCI research of these interfaces as well as makes it hard to program. PyMT is mean to make rapid application development and prototyping possible in this setting. In this paper we give an overview of pymt, and discuss specifically some of it's features that are useful for HCI research on multi-touch and related technologies.

Whether the goal is to build a real world application, a proof-of-concept prototype, or software to perform a lab experiment with participants ; we always need to implement software to define how it should work. The idea behind PyMT is to be a toolkit that makes this task easier when dealing with multi-touch and related new input technologies.

PYMT Overview

This section discusses the main features and ideas of PyMT. For a more detailed discussion and related work see [3]. The projects wiki and API documentation are also good sources of information about implementation details and features (available at http://pymt.txprog.net). PyMT is an open source project, and can be modified by anyone freely under the terms of the LGPL(which requires changes to the core library to be made public under the same license, but applications using the toolkit can be licensed under any terms).

Platform

PyMT and applications for it are written in Python[2], a popular dynamic programming language. Python's dynamic properties, concise syntax and compilation free workflow allow for quick prototyping. The availability of many open source libraries (modules) for a wide variety of purposes lets programmers focus on the task at hand. Like python, PyMT is cross platform and runs on Windows, OSX, Linux/Unix. Relying on OpenGL for graphical output allows PyMT to provide hardware accelerated graphics. While PyMT provides drawing abstractions, advanced user can make direct OpenGL calls to perform drawing operations for maximum flexibility.

Widgets

To program PyMT applications, programmers implement widgets. PyMT provides a variety of ready to use widgets, to allow developers to use commonly used interaction techniques and extend them for their purposes. PyMT provides multi-touch-ready widgets for many basic interactions like rotate-scale-translate, buttons, layouts, sliders, text input, virtual keyboards, etc. A scene graph can be defined in code or declarative in XML syntax. Widgets can be styled using CSS syntax, using id's or classes. A built in animation framework can be used to animate any numerical property of arbitrary widgets easily.

A custom widget implementation will usually consist of implementing methods for touch events and the draw function, which is called each frame. Programmers can implement these by subclassing or attaching event handlers. So if programmers do not want to use any of the provided widgets and the scene graph based architecture, they can implement an application starting with single widget, much like they would when using e.g. a toolkit like Glut[1].

Input Providers

At the core, PyMT provides an abstract input architecture. Support for new input devices or protocols is implemented by writing input providers. PyMT already provides many input providers for various hardware and systems. Some supported input providers are :

- TUIO[5]
- Mouse
- Windows 7 multi-touch
- Windows Pen
- OSX multi-touch pads and mice
- Linux HID multi-touch

The input providers are responsible for reading data from an input device and populating Touch objects with the in-

formation. Widgets are notified through events of new touch objects and when they are updated. Because the input information is presented as object instances, input providers can add new functionality to touch objects that reflect specific hardware capabilities. Programmers can attach data to touch objects and query them about capabilities or device origin.

At the most basic level a touch object is a 2D cursor session. Input sources can be configured in a configuration file and are named. The lines 'touch = wm_touch' and 'pen = wm_pen' on a windows 7 system for example will cause application to receive input from both pen and multitouch modalities if available. Programmers can check a touch events device property to distinguish between input sources. Configuring the input sources in this manner allows for flexibility and easier debugging. For example the source code if touch.device == 'pen' can stay unchanged in the program, and the configuration can be changed to 'pen = mouse' during debugging to simulate simple pen interactions (The mouse input provider lets users place multiple cursors using right click and drag them around independently for debugging).

Dynamic modules

PyMT supports dynamic loading of modules. Programmers can use existing modules or write their own by loading a module when an application is launched (command line parameter) or during runtime (function calls). Being able to load modules various existing PyMT applications can be extended with the same functionality.

For example an existing PyMT module provides subtle visual feedback like for example in[8]. The module simply implements a regular widgets that receives touch events and draws the feedback. When it is activated, the widget is placed at the root of the scene graph. Similar modules can provide more detailed feedback, aid in debugging, introduce additional functionality, modify running applications and/or pre-process input events.

HCI RESEARCH USING PYMT

PyMT was originally conceived to provide a platform for post-wimp multi-touch research. Various properties and features of pymt offer concrete benefits to research practices in multi-touch computing.

Cross platform/input research

As commercial multi-touch solutions are becoming more widely available, many of them offer SDK's to allow programmers to develop software for their products[7, 6]. The widespread adoption of vertical integration in business models however poses certain challenges in research trying to identify overreaching concepts.

Having to implement several versions of a project so that it can run on different hardware poses at the very least practical problems. While write-once-run-anywhere is not necessary for many real world applications (and arguably can impose bad design choices). Having the possibility to run exactly the same software on different hardware is needed e.g. for research comparing different input hardware. Based on the complexity of the interaction techniques, PyMT applications for example work on many multi-touch displays or suing a mouse without any modification or special code.

Collaboration efforts between different laboratories running similar hardware is also made easier by eliminating the need for common infrastructure on the operating system level. By using a cross platform toolkit such as PyMT with support for diverse input hardware used in research labs, the research community can create an environment in which sharing software used to perform experiments and collect data is easier. Placing the shared infrastructure at the toolkit level would make repeating experiments and confirming results independently more plausible. Although it is unclear if lack of common infrastructure is really to blame for the apparent lack of collaboration of this nature in HCI research.

Experiments & Data Collection

The scientific study of HCI and multi-touch user interfaces relies on being able to perform controlled experiments and collect data which can be interpreted. PyMT offers functionality to record input data, serialize it and replay it. This can be very useful in the development process for debugging purposes. More importantly it offers a valuable tool in collecting quantitative data. By recording input events, researchers can go back and analyse aspects of the experiment that were not explicitly built into the experiments data collection ahead of time. The use of dynamic modules allows recording or playback of input events without the actual application even being aware of it.

This is also another area where the PyMT's use of the Python programming language can be taken advantage off. First of all, Python offers built in serialization of virtually any data structure, making it trivial for programmers to store and load any objects or data created during execution. A variety of python modules, such as Scientific Python[4]can be used to analyse the data e.g. statistically. Visualizing collected datasets using OpenGL from within PyMT is also a natural use of collected data. Pythons scripting proprieties and many modules have made it a broadly used tool in many other scientific disciplines such Astronomy, Chemistry and various Biological disciplines.

As already mentioned being able to run the same software in various settings can be of practical benefit. We argue that if possible, it is in fact important to use the same software when comparing different input devices or performing the same experiment in different settings. Software bugs are an unavoidable part of software development, and often times they can go undetected. Unintentional, even if small, differences in how data is collected or a sys-

tem responds can have significant effects on results. Running the exact same program can avoid these pitfalls.

Rapid Prototyping of Interfaces and Interactions

Many design processes involve iterative design or early prototyping. Python's dynamic nature, and compilation free workflow allows for quick prototyping. Many mature GUI toolkits also allow for quickly putting together interfaces using pre-defined widgets. Often times, they provide additional tools like visual interface designers, something PyMT does not yet provide.

PyMT however aims to provide rapid prototyping especially for custom interfaces that go beyond common interface conventions. While it provides reusable widgets for many tasks that often times come up in interface design, PyMT makes the process of defining new widgets and interactions as accessible as possible. Instead of focusing on providing a feature complete set of pre-defined widgets, PyMT focuses on letting programmers write code to handle input events and create graphical output. We think that making the touch event and draw handling accessible at a higher level rather than relying on pre-defined widgets and UI guidelines allows programmers to more quickly create custom interfaces.

Focusing on input processing and graphical output may require a deeper understanding of computer graphics and input models and more programming experience than using mostly common components. But we think it makes more sense in the given context of multi-touch research, because these novel interfaces have not been studied sufficiently to establish comprehensive interface guidelines or paradigms (or whether that would even be desirable).

BIBLIOGRAPHIE

1. Glut - the opengl utility toolkit. http://www.opengl.org/resources/libraries/glut/.
2. Foundation, P. Python. http://python.org/.
3. Hansen, T.E., H. J. V. M. P. S., and Serra, T. Pymt: A post-wimp multi-touch user interface toolkit. In *Proceedings of Tabletops and Interactive Surfaces*, 2009.
4. Jones, E., Oliphant, T., Peterson, P., et al. SciPy: Open source scientific tools for Python. http://www.scipy.org/, 2001.
5. Kaltenbrunner, M., Bovermann, T., Bencina, R., and Costanza, E. Tuio: A protocol for table-top tangible user interfaces. 2005.
6. Microsoft. Microsoft surface. http://www.microsoft.com/surface/.
7. Technologies, S. Smarttable. http://smarttech.com/.
8. Wigdor, D., Williams, S., Cronin, M., Levy, R., White, K., Mazeev, M., and Benko, H. Ripples: utilizing per-contact visualizations to improve user interaction with touch displays. In *Proc. UIST*, pages 3–12, New York, NY, USA, 2009. ACM.