

A “Technology Probe” approach to designing Location-Based Services

John Canny, Professor
University of California, Berkeley



Why “Technology Probe”?

- **We have many mainstream LBS apps:**
 - Service recommendations: restaurants, stores
 - Security: emergency, child tracking
 - Social: friend-finding, dating
 - Navigation
 - Location-based gaming
 - Commerce: fleet tracking, coupons
 - Others,...

Why “Technology Probe”?

- **But many others go “under the radar”:**
 - **On-site service augmentation:**
 - Previews in a video or music store
 - Comparison shopping
 - **Ride-sharing, carpooling**
 - **Location-assisted commerce**
 - Location-based eBay
 - Market-making for small vendors
 - Virtual garage sales
 - **New kinds of social application**
 - Driven by communities and how they associate

Why “Technology Probe”?

- **Best practices for designing new applications:**
 - Context inquiry – observe real users in context
 - Needs analysis
 - Classify their tasks
 - Structured interviews to “get inside users’ heads”
- **Doesn’t work for LBSes “in the wild”**
 - Not practical to follow users around
 - Too many contexts
 - Tasks often very short
 - Can’t do field interviews unobstrusively

What is a “Technology Probe”?

- **An approach to designing highly contextualized applications**
 - Create a rough “master application” **prototype**
 - Make it as **extensible** as possible
 - **Deploy** it with real users
 - **Coach** them on use and extensions
 - **Monitor** use, do periodic interviews
 - **Build** on user **extensions**
 - **Build** on user **suggestions**

Technology Probe for Location Services

- **Most users can't program, so can they really “create” new LB applications?**
- **We don't know (this is science after all!), but we know how to find out:**
 - Match the master application as closely as possible to users' conceptual models (for place, time etc.)
 - Make the extension language as simple as possible for the tasks.
- **We are building a system called “Glaze” to test this approach**

Technology Probe for Location Services

- **Observation: most existing LBSes can be implemented in a publish-subscribe way**
 - **Service recommendations****
 - **Security: emergency, child tracking**
 - **Social: friend-finding, dating**
 - **Navigation****
 - **Location-based gaming, geo-caching**
 - **Commerce: fleet tracking, coupons**

**** Recommendations and navigation require additional “search” features**

Glaze's conceptual model

For user comprehension, we translate publish-subscribe into a simple “verb-noun” model:

- **Read** (subscribe to) a “place”
- **Write** (publish) to a “place”
- **Find** (service discovery)
 - Needed for navigation and service ranking
 - Also to find services the user doesn't know about
- **Verb-noun composition should be accessible to most users, even non-computer savants**

Glaze's conceptual model

The last piece of our conceptual model is “place”

- **Place is a user-friendly version of an event-driven publish-subscribe location database**
 - Places always have names
 - Places may have geographic extents
 - Places may have temporal extents
 - Places may be visible only to certain people

Glaze services

- **Some services (typically P2P services) require both publish and subscribe macros**
- **Others require only one side, with the other provided by a vendor:**
 - Security service would provide a subscriber
 - Restaurants and other business would be publishers
- **Our deployment will start with pre-defined services using “read, write, find”**
- **It will allow user-defined macros for new services**

Glaze examples

- Main screen contains “read, write, find” options, and also currently-defined macros “Eat, See, Shop,...”:



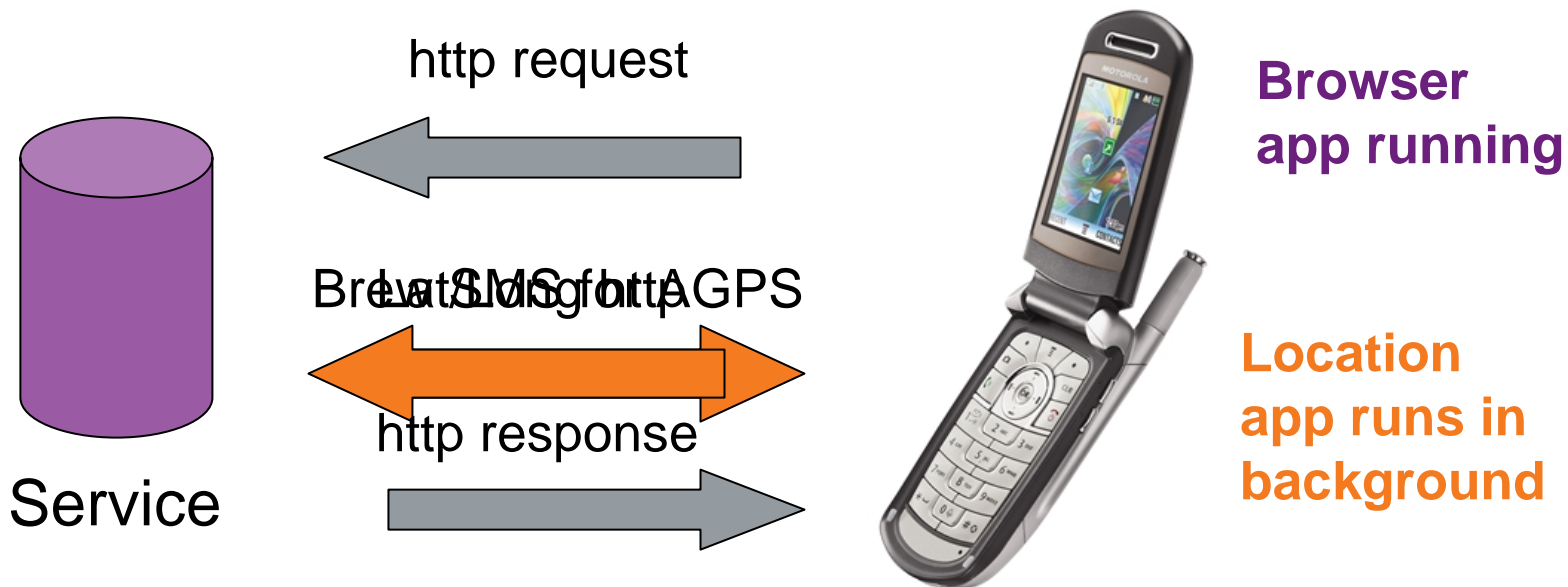
Glaze example:

- Using the “shop” macro – a kind of find



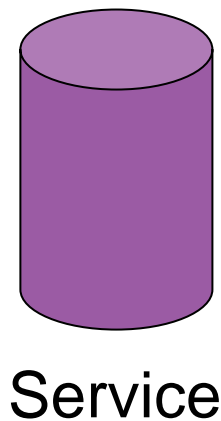
Glaze architecture

- For early design iteration we are using WAP to prototype Glaze's interface + BREW[®] for AGPS.
- BREW-directed SMS messages allow us to query the phone's location in the background.



Glaze architecture

- BREW-directed SMS also allows user tracking w/o a BREW app running on the phone –
 - Saves energy, does not conflict with other running apps

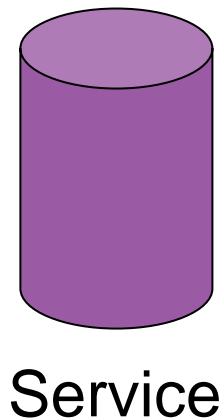


Any app

Location app runs in background

Glaze architecture

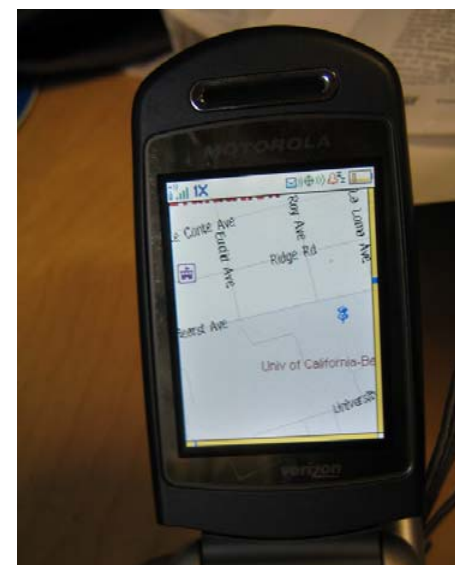
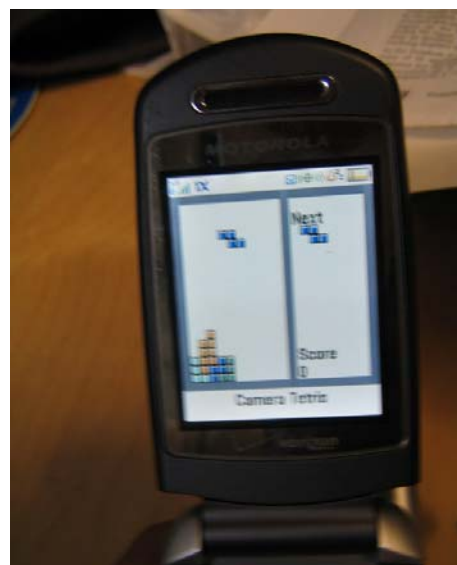
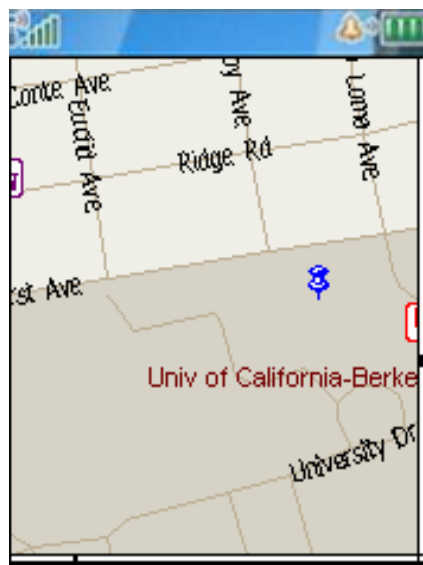
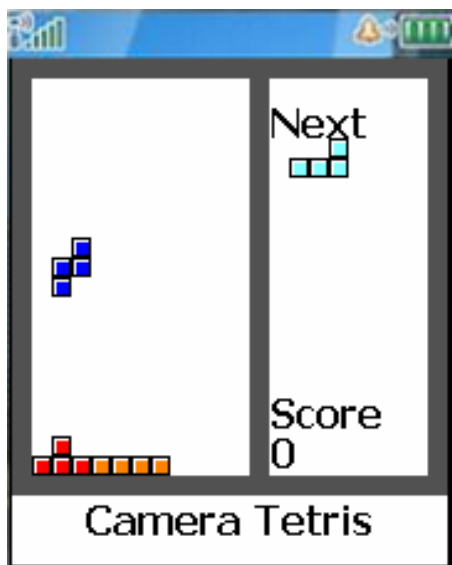
- BREW-directed SMS also allows user tracking w/o a BREW app running on the phone –
 - Saves energy, does not conflict with other running apps



Location app runs in background

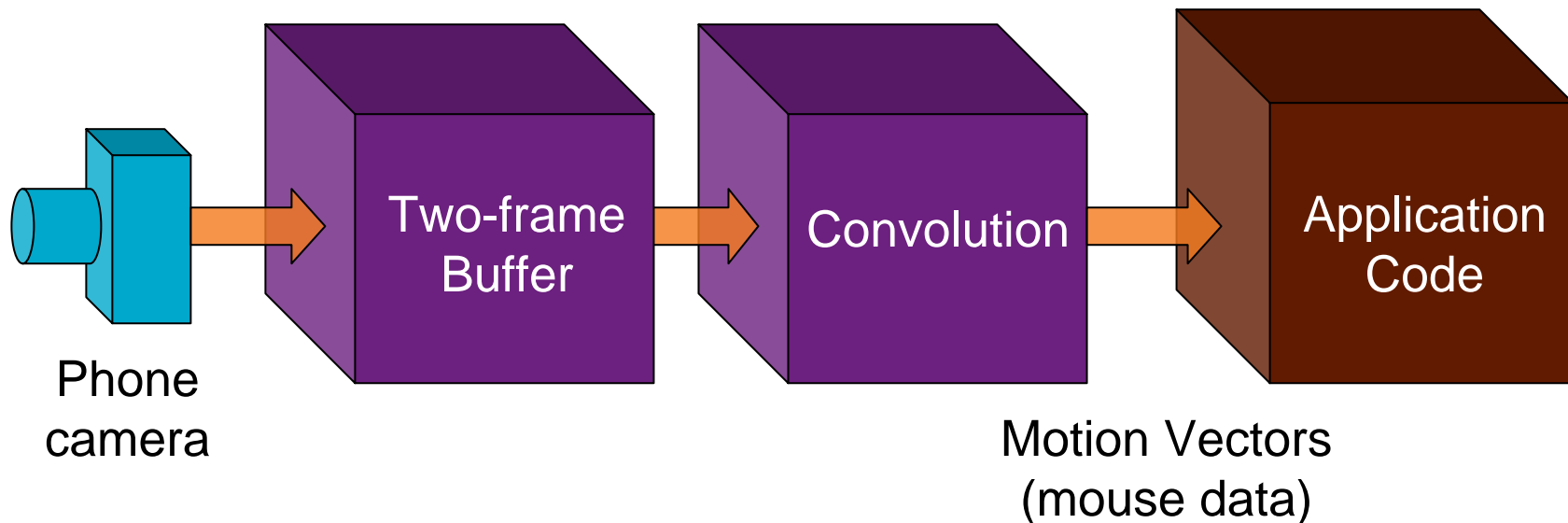
Map Services

- We've developed some new technology to support map functionality on BREW phones
 - The first piece is "TinyMotion" a software-only continuous mouse for BREW camera-phones.



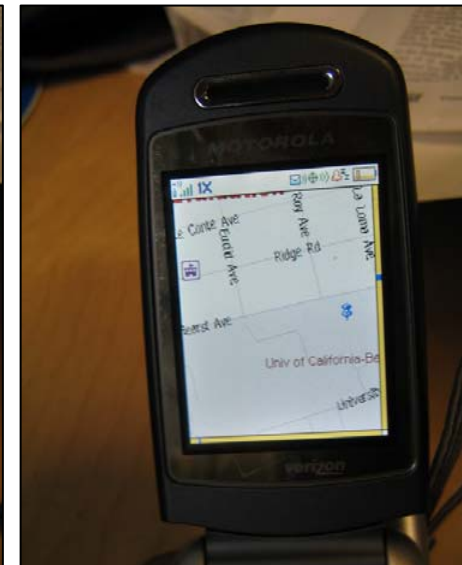
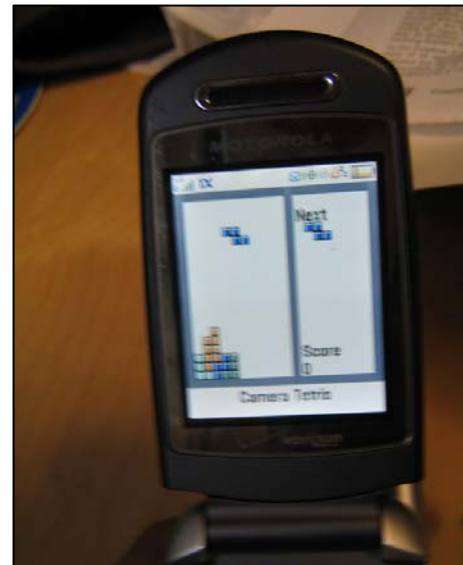
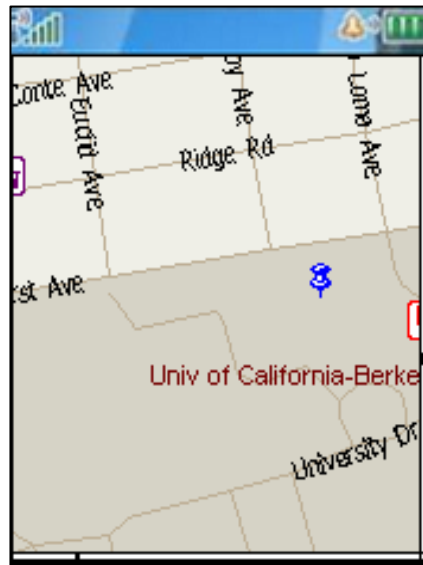
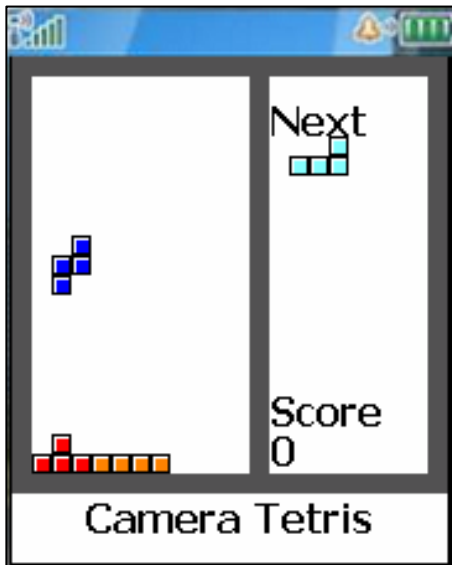
TinyMotion

- **Tiny motion captures images continuously**
 - Compares consecutive images to get camera motion
 - Source code available on web (google “Tinymotion”)
 - In C for BREW 2.1, w/ 5x assembly acceleration (ARM9E)



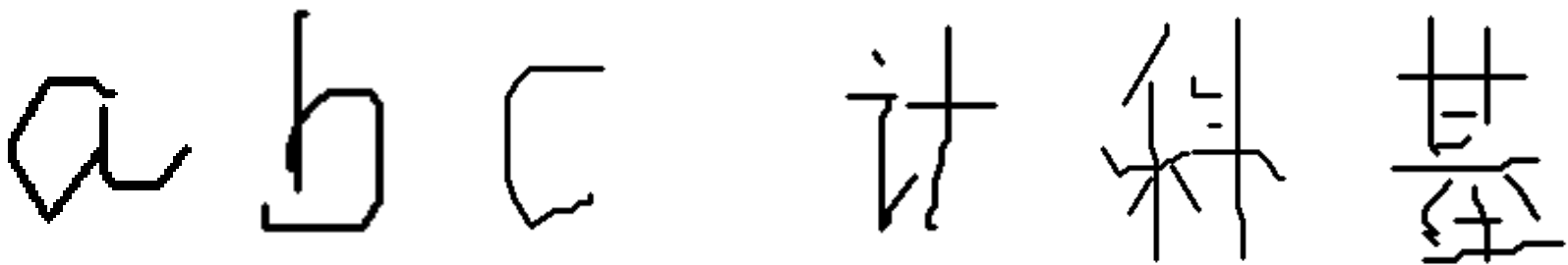
TinyMotion Performance

- Update rate about 12 fps – OK for most pointing
- Tested on apps for gaming, menu selection, map pointing,..



TinyMotion Applications

- Currently testing gesture recognition using TinyMotion – a new input mode for phones



Where TinyMotion Works



Glaze in Production: The LAP Script Engine

- **Implemented on the cell phone in BREW**
- **Supports most high level language facilities**
- **Provides built-in Location-Aware Primitives (LAPs)**
- **More descriptive than JavaScript, more efficient than J2ME**
- **End user can create LBS applications in LAP scripts by menu selection operations**
- **Based on a customized version of the Lua Interpreter**

Summary

- **Technology probes: design in the wild**
- **Glaze: a probe for LBSes in context**
- **Conceptual model:**
 - Write → publish
 - Read → subscribe
 - Find → discover
- **Tinymotion: mouse input for camera-phones**