

Magic Wand: A Framework for Developing Remote Controlled Web Applications

Vibhor Nanavati

Carnegie Mellon University, Silicon Valley
vibhor.nanavati@sv.cmu.edu

Abstract. This poster describes the MagicWand framework for developing web applications that can be controlled remotely by smart phone running on android OS.

Keywords: Human Computer Interfaces, Android, Smart Controllers, Mobile Applications, Web Applications.

1 Introduction

With the massive adoption of smart phones since 2007, the expectations of user interfaces have dramatically increased. Users are now comfortable with two-finger gestures such as pinch scaling or gravity based mobile applications. The landscape of desktop applications and web applications in particular, has largely remained unchanged. In particular, users have seen a dramatic change in the kinds of web applications they can interact with in the last decade. But the modes of interaction with these applications are still confined to conventional interfaces. The poster illustrates a revolutionary idea that can take the human computer interaction to a next level.

2 Background

The idea of using smart phone as a conventional mouse and/or keyboard has been shown in past. AirMouse [7] on iPhone allows users to control the mouse cursor on desktop over local Wi-Fi network using the touch screen. Remote-Droid [10], an open source framework allows similar user experience on Android devices. If we look beyond the conventional interfaces, Nintendo Wii [8] consoles provide a motion controlled gaming experience. Onomy Tilty Table [9] demonstrates a version of Google Earth [2] that can be navigated using a large tilting and rotating table. Google Earth for smart phones[3] itself presents a unique experience for user where one can change the viewing camera angle by simply rotating the phone around any of the 3 axes.

3 Design Goals and Constraints

An ideal solution for a smart controller of web applications shall address following goals:

- Fast response time on motion controlled gestures.
- Install once, use anytime and anywhere.
- Make multiple interface devices redundant.
- Allow web developers to define the interactions.

Due to the given environment, the following roadblocks have to be overcome:

- Only channel of communication in browser is http.¹
- No Bluetooth API implemented by any major browser vendor.
- Browser does not allow listening on a socket.
- Browser connections to web application and smart phone are subject to cross-site scripting restrictions.

4 Solution

4.1 Framework

MagicWand framework comes bundled with a mobile application for Android [1] powered devices that provides the end user controller behavior, and a JavaScript library that can be included in the target web application. The JavaScript library hides the MagicWand protocol and provides convenient routines for a web application to connect/disconnect with an Android device and send messages to it. Messages are passed over HTTP in JSON format [6]. The application in browser can send configuration messages to control the UI and layout of MagicWand app in mobile. The mobile app sends the sensor data periodically. The period is determined by the sampling frequency which is also configurable. Apart from sending configuration requests, browser can also send a text-to-speech request or request the device to provide a haptic (vibration) feedback. The framework requires both the Android device and the desktop which runs the web application to be on the same local Wi-Fi network.

4.2 Architecture

In order to circumvent the limitations of working within the browsers, following design choices were made:

- Android application opens a listening TCP socket on start up. Web application that includes the MagicWand JavaScript makes a request to connect.

¹ With HTML5, browsers can also open websocket connections to websocket-enabled servers.

- The sensor data transfer takes place on a dedicated channel. Any configuration requests, haptic feedback or text-to-speech requests are made on a secondary channel.
- The communication with the phone takes place in a browser Iframe. A web application on different domain runs in the parent frame. The data between 2 frames is exchanged using HTML5 messages [5].
- In order to keep the response time low, the android application keeps the connection with application persistent², instead of browser making a new HTTP request for every sensor poll.

5 Results

To demonstrate this framework, I have created tiltymap.herokuapp.com. It uses Google Maps API [4] to support various mapping functionality. This sample application demonstrates how the MagicWand JavaScript is used in conjunction with the magic wand Android app. User of this application can use the phone to control the map displaying on a larger screen such as desktop. Some of the key interactions include gravity based map movement; voice-command enabled searching of locations and pinch scaling of the map.

6 Future Work

MagicWand framework can be extended to applications beyond browsers. A Java based http client library is under development and shall expose an API similar to its JavaScript equivalent. Another interesting application of the framework that will involve work in future is allowing applications to be controlled by more than one Android device.

NOTE: The project is in process of being open-sourced. The project page is hosted at <http://code.google.com/p/magicwand/>

Acknowledgements. My sincere thanks to: Ted Selker, faculty at CMU Silicon Valley for inspiring a project of this impact, Martin Griss, Director CMU Silicon Valley for proposing several possible applications of the framework and Neha Mohan, my wife for constantly contributing with valuable user feedback.

References

- [1] Android, <http://developer.android.com/index.html>
- [2] Google earth, <http://earth.google.com/>
- [3] Google earth for mobile, <http://www.google.com/mobile/earth/>

² Using HTTP response header transfer-encoding: chunked, a server can keep a long-lived connection.

- [4] Google Maps API, <http://code.google.com/apis/maps/documentation/javascript/>
- [5] Html5 inter-frame communication, <https://developer.mozilla.org/en/DOM/window.postMessage>
- [6] JSON, <http://en.wikipedia.org/wiki/JSON>
- [7] Mobile AirMouse, <http://www.mobileairmouse.com>
- [8] Nintendo Wii, <http://www.nintendo.com/wii/console>
- [9] Onomy Tilty Table, <http://www.onomy.com/video/Tilty.mov>
- [10] Remote-Droid, <http://www.remotedroid.net>