Towards building augmented reality web applications

Jens de Smit
SURFnet
Jens.deSmit@surfnet.nl

May 26, 2010

Abstract

Augmented reality has recently become very prominent in mobile and desktop computing. Many AR applications use some parts of web infrastructure standards, but all use some proprietary technologies as well because the suite of web standards does not offer everything that is required to build AR applications. This paper discusses what would be required to be able to build AR aplications using only web technology and how to prepare the suite of web standards for future developments in ubiquitous computing technology.

1 Recent augmented reality developments

In the last year we have seen a surge in the development of mobile and desktop augmented reality (AR) applications. These applications are built to add digital content to the world around us. Most of these currently available applications fall in one of two categories: geolocation-based AR or marker-based AR. The defining factor in these two variants of AR is how they determine what parts of the world to augment. AR based on geolocation uses positioning information such as GPS and an electronic compass to determine where the user is and where he is looking, marker-based AR uses image recognition to determine which part of an image (usually a live camera stream) to augment. The common factor of all AR is that everything in AR is about *context*. The geographical information or the marker recognition provide an amount of contextual information about what the user wants to see and where to display this information that is unparalleled by traditional user interfaces.

Beyond the gimmick of seeing and manipulating something that is not really there, there are very real applications for augmented reality in many different situations. Instructional animations overlaid over real animations can help in education but also function as a reference for experts such as mechanics or surgeons when performing critical tasks. Virtual objects can be used for games,

for advertising or as tourist information. Many companies are already using webcam-based augmented reality to let potential customers check out the latest cars or sunglasses. Mobile AR application Layar [1] is a big hit with its "local discovery" capabilities where users simply pick a subject and pan their mobile phone around to find out what is available nearby. More examples of recent AR technology include applications such as Wikitude [2], Bionic Eye [3] and "We are Autobots" [4], part of the marketing campaign for the Transformers 2 Movie. It should be clear that AR is a technology that will be important in future computer applications.

2 Relation to the web

Most of the above examples are, or could be, based on web technology. HTTP, the fundament of the web, is the protocol Layar uses for its client-server communication. The webcam-based marketing gimmicks run inside a user's web browser using Flash or some other plug-in. Augmented reality browsers Layar and Wikitude use JSON and XML respectively to transport their store and exchange *points of interest* (POIs); both are technologies that herald from the web. Yet, for all the web technology these applications rely on, all of these applications use some proprietary technology to achieve their goal: the mobile AR applications run as native applications on iPhone or Android, the desktop applications require Flash or something similar to work.

The crucial component to augmenting reality is leveraging more input than a user's clicks, taps and keyboard strokes. AR is about recognizing sights and sounds or automatically determining location and direction and thereby knowing what the surroundings are and augment them. The problem with developing web applications that do this is that there is currently no sufficient way to gather this information without resorting to non-standard tools. To demonstrate what is still required the following paragraph will dissect the Layar application as an example to show what is still missing from the currently available standards to build an augmented reality application.

2.1 Example application: Layar

Layar is one example of a group of applications called "augmented reality browsers". It is currently available as a native application for Android and iPhone 3GS phones. The goal of the application is to let users find *points of interest* in their vicinity. By definition it is an application targeted towards mobile devices and is currently available on certain popular smartphones. In essence, the application works as follows:

- user selects a subject from a list;

[1] http://www.layar.com
[2] http://www.wikitude.org/
[3] http://www.bionic-eye.com/
[4] http://www.weareautobots.com/ww/index.php

- Layar client requests information about chosen subject from Layar server, appending geographical information about the user to the request;
- Layar server retrieves POIs for chosen subject in the vicinity of user's location and sends them to the client;
- Layar client displayed the phone camera's video feed, overlaid with the POIs that it got from the Layar server. The POIs are correctly laid out using the phone's tilt sensor and electronic compass the discern the phones camera's orientation.

The first three steps of this process are easy to realise using common web standards and, in fact, two of them are. The first step, a simple user interface, is no challenge for HTML, CSS and JavaScript. The second and third step, requesting information from a server and getting a response back, can be performed over HTTP as a web service request/response action. Layar does this: the client sends a GET request with the parameters in the URL and the server responds with a JSON-formatted document describing the result of the request and containing nearby POIs for the chosen subject. The geographical information required for the request are retrieved using the device's native interface because Layar is a native application, but HTML 5's GeoLocation interface would work just as wel for this.

The fourth step is the step where Layar does a bunch of things that are currently not supported by web standards. Access to the phone camera's video signal is not supported, not using the HTML `<video>` tag or any JavaScript interface. The same goes for the tilt sensor and the compass: required information not accessible from inside a web browser. It is, however, this last step that makes Layar an augmented reality platform and not the next mobile search widget. It is also one of the things that makes Layar (and other augmented reality browsers) so popular.

Access to these extra sensors is crucial for developing innovative web-based applications that leverage the abilities of mobile devices. Camera feed, tilt sensor and compass are only a few examples, other input devices are possible as well, such as air quality sensors, altimeters or biomonitors. Not all of these inputs may be directly applicable to augmented reality, but enabling their use is paramount to innovation of mobile applications. Each sensor represents a different measurement, but what they share is a relatively high frequency of updates and an applicability specific to mobile devices: each conveys information about the the state of the phone's holder, something not commonly found in desktop systems. A generic framework for handling input from these sensors in JavaScript would be a valuable addition for working with these and new sensors.

3 Current state of affairs

The *W3C Device APIs and Policy Working Group* [5] has incorporated a *Camera API* into its charter as well as a *System Information and Events API*, although the latter seems to focus more on more traditional system information: included examples mention battery level and network status. Access to typical mobile sensors such as compass and accelerometer are not mentioned explicitly.

On May 12 2010 a draft for the HTML5 `device` element [6] has been submitted by a member of the WHATWG, which is a first suggestion for generic device access. However, this draft is still in its very early stages and may or may not converge to an interface for only audiovisual devices or for more generic device access. The generic route would open up access to a broad range of devices, but this has the risk of forcing developers to do a lot of low-level coding to access devices, which may not be desirable. On the other hand, a (stub) interface to any device could be very valuable in developing innovative web applications that leverage a slew of new input methods.

4 Proposed course of action

Considering the obvious interest of the web at large in augmented reality, the recent rise of new input sensors in mobile devices and the activity of different web-concerned working groups, now would be a good time to discuss how to incorporate new input devices into web standards in order to be able to create augmented reality web applications, preferably in such a manner that new types of devices can either be accessed without any new standards definition, or in a framework that allows for swift incorporation of new devices in existing standards.

To make truly captivating and innovative AR applications, a second development may have to be to define and implement a standard for making sense out of the input, mainly sound and image recognition. While this could be implemented by web developers once they have access to webcam and microphone data, the skill to do this may not be in the portfolio of traditional web developers. Additionally, good recognition algorithms tend to be computationally intensive: implementing them as native code in a web browser will be more efficient than in an interpreted language such as JavaScript. However, this development is still one more step away from the one outlined above and perhaps better suited to discuss once a standard for sensor access has started to solidify so it can be built upon. Once we get there, one of the most important factors will be deciding upon a standard for audio and video recognition formats.

[5] http://www.w3.org/2009/05/DeviceAPICharter
[6] http://dev.w3.org/html5/html-device/