Webizing Mobile AR Contents

Sangchul Ahn∗
University of Science and Technology
Heedong Ko
Korea Institute of Science and Technology
Steven Feiner
Columbia University

ABSTRACT
This paper presents a content structure to build mobile AR applications in HTML5 to achieve a clean separation of mobile AR contents from their application logic to scale like the web. By extending POI’s (Point of Interest) to objects and places with Uniform Resource Identifier (URI), we could build objects of interest for mobile AR application as DOM (Document Object Model) elements and control their behavior and user interactions through DOM events. Using our content structure, a mobile AR application can be developed as normal HTML documents seamlessly under current web eco-system.

Index Terms: K.5.1 [Information Interface and Presentation (e.g., HCI)]: Multimedia Information Systems—Artificial, augmented, and virtual realities; K.5.4 [Information Interface and Presentation (e.g., HCI)]: Hypertext/Hypermedia—Architectures

1 INTRODUCTION

Most mobile AR applications have been developed in two extreme cases: one at a small scale indoors on a table top or in a room environment with a handful of objects in place, and the other at a very large scale outdoors with mostly location and orientation information with little or no object recognition and tracking. Recently, indoor AR applications are getting off of the desktop and becoming mobile to the next room, across the hallway, and to other buildings or across town. These mobile AR applications must be aware of their own location context in order to bring the objects of interest to bear dynamically for augmenting.[2]

KARML[4] extended KML, location-based contents standard by OGC, to include the AR XML elements. KML is very popular through Google Maps and Google Earth applications but it is not based on HTML so the interoperability between KARML and existing web contents is limited. Our approach is to make use of existing HTML standard but extend the use of URI to identity and reference physical objects and places. Any HTML DOM elements can be linked to these physical URI’s and DOM event mechanism can be handled by javascript layer for behaviour and user interaction. X3DOM[1] also attempts to integrate X3D scene elements as HTML DOM elements and events seamlessly like us. However, it does not have the notion of physical URI’s as objects and places of interest for the in-situ AR and no notion of location context. In our approach, location-based AR contents can be intermixed within regular HTML documents by augmenting any HTML DOM element with its location context through extension of CSS3.

There are some commercial HTML5-based AR platforms like Junaio and ARCHitect. Junaio provides AREL as a javascript AR API for location-based AR and GLUE as its XML extension for image recognition-based AR. ARCHitect is another approach of javascript SDK for AR by Mobilivy. Both provide javascript library for building AR application with location and image recognition based tracking. Their approach introduces AR functionalities procedurally through javascript objects. This procedural approach makes AR contents difficult to intermix and interoperate with standard HTML web contents and their interaction through DOM event handling mechanism. Our AR content structure is based on HTML and selection/binding mechanism of CSS with their associated javascript event handling of DOM events.

2 MOBILE AR CONTENT STRUCTURE IN HTML

Webizing mobile AR content refers to the process of combining virtual and real world objects through three components: use of URI for dealing with physical objects, CSS Place Media module, and the DOM event extensions for situated interaction.

2.1 Point of Interest

In our model, POI refers to physical object and it is viewed as a physical resource, an extension of web resource. It is assumed that any POI has its URI and accessible through HTTP. However identifying and tracking physical objects require the feature samples suited for the recognition method. To address this, we use the REST (REpresentational State Transfer) to provide the various types of feature samples as a representation of its URI via HTTP content negotiation. And the browser can choose one of available tracking methods and request the feature sample data to URI of target POI. As a result, the AR content does not need to include all feature samples for tracking but just refer to URIs of target POIs.

Place is an extension of POI that can contain other POIs their locations. In addition to the default properties of POI, they have several environmental properties: boundary, Local Coordinate Reference System (LCRS), and local POI map. Boundary is the geometry separating the inside/outside of the place. The subspace of place is defined a LCRS description represented to a simplified schema of Geography Markup Language (GML) CRS package model.[6] A local POI map contains the locations of POIs within the place boundary. By this extension, the space partitioning tree of the physical world can be constructed recursively. The scope is dynamically determined by the user location. The place is regarded as a POI when the user is outside. However it forms a spatial environment when the user enters it (e.g., building, room, and earth).

2.2 Augmenting HTML Elements with CSS

Elements in a HTML document have been rendered to a 2D plain page and their layout is defined by Cascading Style Sheets (CSS). HTML elements describing POIs and their augmented model must be rendered in 3D environment, not on a Page. For this reason, we have developed a place-based augmentation model to situate the virtual objects to physical world in human scale. The AR contents are situated to a Place (e.g., building or room) providing its own CRS and POI map. To manage 3D augmentation model, we extended CSS media type to Place media[3]. Place media module selects the HTML element to augment to the target POI and its 3D position and orientations.

A CSS rule has two main parts: a selector, and one or more declarations consisting of property-value pairs. A selector determines the valid elements to apply the CSS declarations. -ar-target property is used for specifying the target POI or place. With this property, the selected HTML element #about_tower is transformed relative to the target POI. -ar-transform applies 3D transform to the elements using three methods: translate(), rotate(),

∗e-mail: prime@ust.ac.kr
and scale(). This property has come from the CSS 3D transform but differs in the dynamic determination of origin with -ar-target.

```html
<style type="text/css" media="place">
    #about_tower {
        -ar-target: poi("http://dpoi.org/seoul_tower");
        -ar-transform: translate(10,23,15);
        -ar-transform: rotate(15,41,26,230deg);
        -ar-transform: scale(2,2,2);
    }
</style>
</video>

2.3 DOM Event Extensions for Situated Interaction

The DOM level 2 Event specification defines the primitive interface to generate, handle, and register event listeners.[5] By default, five UI events are provided: FocusEvent, MouseEvent, KeyboardEvent, WheelEvent, and CompositionEvent. Recently TouchEvent has been introduced to support mobile devices. To address the requirements for situated interaction, we have extended UI events: PlaceEvent, PoiEvent, and SightEvent.

PlaceEvent occurred when the place is changed by the movement of user. During a situated interaction, the environment context is changed when the user moves from one place to another place. It is determined by continuous containment checking of the boundary of nearby places and the user’s position. A PlaceEvent contains a place identifier, timestamp, and the local CRS description. It has three event types: placein, placeout, and placechanged.

PoiEvent is generated when the state of POI is changed. When the user enters a certain place, POIs in the place are also situated. The initial state comes from the place URI using the restful service invocation and is updated by trackers. A PoiEvent includes a POI identifier, timestamp, and its position. It has three event types: poinin, poiout, and poimove.

SightEvent is created when the target POIs enter or leave the view frustum. It is motivated from MouseEvent, one of most frequently used UI events. Like mousein, mouseout, and mouseover events are fired according to the spatial relationship between the pointer and the visible HTML elements, SightEvent is created when the virtual and physical objects enter or leave the view frustum. A SightEvent contains a URI, its position, and timestamp. It also has three event types: targetin, targetout, and targetmove.

3 Implementation

To validate the proposed content model, we have developed a prototype of mobile AR web browser for iOS and Android. Here we explain the distinguished components as contrasted with typical page-based web browsers. It consists of four major components: target manager, positioning subsystems, place manager, and situated layout engine. Target manager picks the target POIs from stylesheets and gets the feature samples through REST access to their URIs. The types of feature samples are extensible for the supported tracking methods. Currently RFID and image recognition features are supported. Positioning subsystem determines position and orientation of the user and the movable POIs. Its major components are classified into Mapper and Tracker. Mappers correlate the different location representations from diverse local positioning systems into LCRS of the situated place. Tracker includes WifiTracker, MotionTracker, and ImageTracker. WifiTracker deals with the WiFi fingerprint-based positioning engine. MotionTracker brings the value of gyro and compass of mobile device. ImageTracker is the image recognition-based tracker. We have two separated Image-Tracker implementations, KistNifImageTracker and VuforiaImageTracker based on Qualcomm Vuforia. Place manager serves the descriptions of place environment and the local POI map. Situated renderer manages the unified scene graph to combine physical objects, virtual elements, and user viewpoint into the situated place.

To address the device-specific requirements Adobe PhoneGap is used as the hybrid mobile application platform that provides JavaScript APIs to access the device capabilities and the plug-in architecture to extend additional native functions.

4 Conclusion

‘Webizing’ mobile AR contents cannot be achieved partially by embedding HTML and XML tag sets, but rather by fundamentally adapting AR content model according to web principles of separation of content from its structure and behavior. We have presented a HTML-based mobile AR content model without introducing any new tag sets and their structure and behavior can be captured within CSS and javascript framework of HTML5 so that the current page-based web contents can be leveraged for mobile AR contents using web eco-system fully.

Acknowledgements

This research is supported by Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program 2012.

References