Experiences Designing Hypermedia-Driven and Self-Adaptive Web-Based AR Authoring Tools

Vlad Stirbu
Nokia Research Center
vlad.stirbu@nokia.com

Yu You
Nokia Research Center
yu.you@nokia.com

ABSTRACT

In this paper we present our experiences on developing generic and adaptive web-based content authoring tools for augmented and mixed reality applications. Our approach uses hypermedia to convey the capabilities of content servers and to load on demand only the functionality needed to interact with the corresponding content server. The mechanism allows the web application to provide an optimized user experience by adapting to the environment where it is used.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]: Artificial, augmented, and virtual realities; D.2.2 [**Design Tools and Techniques**]: User Interfaces

General Terms

Design, Experimentation

Keywords

hypermedia, user interfaces, mirror worlds, augmented reality

1. INTRODUCTION

Augmented reality and mirror world applications change the way we interact with geo-tagged content or location based services. They have shifted the interaction modality from a two-dimensional interactive map to a three dimensional space in which we can interact commercial or user generated content in-situ using a see-through display or via remote exploration.

This space was used initially to visualize geo-tagged photos or content geo-tagged using the GPS sensor of mobile devices. A later development, led by providers of augmented reality application, allowed users to create virtual content that can be positioned precisely in the physical world, such as on the facades of buildings. As web formats and protocols become the dominant technological foundation of these applications, interoperability between the service providers becomes reality.

WS-REST 2012, April 2012; Lyon, France

This paper presents our experiences designing web-based authoring tools that allows user to create content that can be used in augmented and mixed reality applications. Our approach enables application developers to create generic authoring applications that adapt the user interface to the services that the user interacts with, by simply following the links provided by the content servers.

The paper is organized as follows. Section 2 provides an overview of the application domain that motivated our work. Section 3 describes our approach for developing adaptive web-based authoring tools driven by hypermedia controls. Section 4 contains a discussion on how the approach facilitates decoupling of authoring and storage services, and how hypermedia effects the development of mobile web applications. Concluding remarks are provided in Section 5.

2. MOTIVATING SCENARIO

In this section we explore the content creation in augmented and mixed reality application domain, the scenario that motivated our work. First we provide an overview of this application domain focusing on the content used, and how web can provide the common technological foundation for an open ecosystem. Then, we focus on the problems that are faced by the developers of web-based authoring tools.

2.1 Content for augmented and mixed reality applications

Modern mobile devices equipped with positioning sensors shift the preferred visualization metaphor of geo-tagged content from two-dimensional maps to full three-dimensional spaces. Augmented and mixed reality applications allow smartphone users to see and interact with content either in-situ using a see through display (e.g. Wikitude[1] or Layar[2]), or through remote exploration (e.g. Google Earth[3] or CityScene[4]). Besides commercial content and information about points-of-interest, we witness an increased amount of geo-tagged user generated content. This content is typically accessed over open interfaces provided by specialized third party services, such as photos from Flickr[5], or micro-blogging posts from Twitter[6]. This content is retrofitted into the three dimensional world based on geographic metadata associated with the content, but the positioning and the corresponding visualization are application specific.

[1]http://www.wikitude.com/
[2]http://www.layar.com
[3]http://earth.google.com/
[4]http://betalabs.nokia.com/apps/nokia-city-scene
[5]http://www.flickr.com
[6]http://www.twitter.com

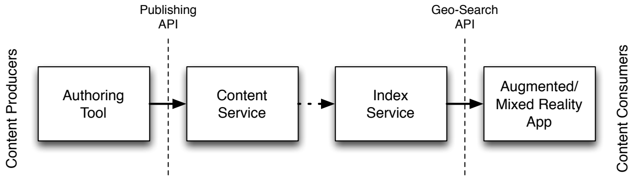

Figure 1: Content creation process for augmented and mixed reality applications

Figure 2: Web-based authoring environment

Authoring native content for augmented and mixed reality application requires precise positioning and three dimensional models based on formats understood by a wide range of applications. However, current authoring tools provided by the leading augmented reality services create content that can be used only by the applications associated with the respective service. In this context, KHARMA [5] proposes a common foundation for mobile augmented reality applications that uses KML [7] for positioning web content into the real world, and Collada [2] for describing three dimensional models. The use of common web formats allows content producers to author content that can be consumed by multiple augmented and mixed reality applications. Content producers use their preferred authoring tools to create content that is published to content storage services. The content is geo-indexed and made available via a geo-search interface, which enables applications to find relevant content using geo queries (see Figure 1).

2.2 Web-based authoring

From a web perspective, the central element of the web authoring of content for augmented and mixed reality applications is the *virtual artifact*. The virtual artifact is a web resource that binds together web content to a physical location and optionally a three dimensional model. Being a first class citizen of the web, the artifact is identified by a URI, provides an interface that is accessible using the fixed set of verbs (e.g. the `GET`, `POST`, `PUT` and `DELETE` methods of HTTP protocol), and conveys it state using a KML representation that contains links to a Collada model, and to web content made available by other services. The artifacts are hosted by a artifact storage service that provides the web interface that allows user-agents to access relevant content around specific locations. The interaction between the service and the clients is done mainly through geo and reverse-geo queries. An artifact storage service can be custom build or can be provided by generic augmented and mixed reality backend service [6]. In this environment, the artifacts are created with web-based tools that are provided by an artifact authoring service. The authoring tools are typically optimized to the specific needs of the content that is created.

For example, content that is intended to be visualized at a physical location that does not change in time can be associated with a subclass of artifacts that we can call *fixed artifact*. For such an artifact, an authoring tool is concerned more with the placement operations at a particular location. Alternatively, a *mobile artifact* moves around a specific path, therefore the authoring tools will emphasize the route. So far the artifacts can be seen as just fixed or mobile, but as the applications become more sophisticated, so the the features and capabilities of the artifacts.

Developing web-based authoring applications for such an open environment poses a number of challenges. First, the authoring applications may have to interact with content services offered by multiple service providers, see Figure 2. These services can evolve over time, and some may store and index a larger set of artifacts than others (e.g. the new artifact type *X*). Content authoring tools have to be able to handle gracefully content servers with *unknown* capabilities, as well as content server that handle a smaller set of artifacts.

3. OUR APPROACH

In this section we describe our approach that uses hypermedia controls and RESTful principles [3] to drive the interactions between the web authoring tool, and the artifact storage and the artifact authoring services. We present first the format that conveys the capabilities of a artifact storage server, then we describe the overall interaction between these components from network and user perspectives.

3.1 The artifact collection document

The artifact collection is a JSON-based document format that describes lists of related items, such as virtual artifacts belonging to a user. The format follows the `Collection+JSON` type [1], and besides the list of user's artifacts, contains additional properties that enables a user-agent to interact with the artifact storage resource. For example, the `queries` property provides one or more forms that enables the user-agent to perform various queries, and the `href` property, which represent the URI of the collection, that allows the user-agent to create new items in the collection using the `POST` method of the HTTP protocol, if the server supports this feature.

```
{
 "artifacts": {
  "version": "1.0",
  "href": URI,
  "items": [...],
  "queries": [...],
  "templates": [
   {
    "href": URI,
    "rel": STRING,
    "type": STRING,
    "name": STRING
   },
   ...
  ]
 }
}
```

The server informs the user-agents of what artifact types it can handle using the `templates` array property. Each accepted artifact type is indicated by an object having the following mandatory properties: `href`, `rel`, `type`, and `name`. The `href` contains the URI from where the artifact template can be retrieved. The `rel` property conveys the artifact type that can be created by the user-agent when retrieving the template. The `type` indicates the

Figure 3: Self-adaptive user interface: network and user interaction perspective

media-type of the instances of the artifact type that are accepted by the server. The `name` is a user readable description of the artifact type that can be used at the user interface level. Having this information, a user-agent can compose artifacts of types that will be accepted by the server. The same information can be delivered to user-agent that prefer an XML encoding, such as the ATOM feed.

3.2 Artifact authoring detection

The artifact authoring detection enables the application running in the browser to query the artifact authoring service to find if the service provides a user interface that is able to compose artifacts of types accepted by an artifact storage service. The functionality is implemented as a Javascript library that follows loosely the Google Loader API [4], but instead of being a generic library loading module, it is optimized to dynamically load smaller libraries specific to one application domain. Using the library, the user-agents can *check* if the authoring server supports an artifact type. The input parameters are the `rel` and `type` values provided by the artifact storage service in the collection document:

```
artifacts.check(rel, type)
```

Once the user agent knows if the authoring server supports creation of specific artifact types, it can *load* the libraries that provide the needed functionality:

```
artifacts.load(moduleName)
```

3.3 Self-adapting user interfaces

The information included in artifact collection document and the ability to verify if the authoring service that provided the application supports authoring of specific artifact types, allows us to develop authoring tools that have the ability to self-adapt the user interface to the usage context.

Figure 3 depicts a timeline of the interactions between the web authoring tool and an artifact authoring and artifact storage services from a network perspective, and the effects of each interaction at user interface level. First the web browser loads the application from the artifact authoring service and renders the user interface. The application contacts the artifact storage service where the user has content. The content storage service returns a artifact collection document that contains the user's artifacts together with an array of templates. In our scenario, the artifact storage service is able to handle fixed and mobile artifacts:

```
"templates": [
 {
  "href": "http://example.com/fixed-artifact",
  "rel": "artifact fixed",
  "type": "application/vnd.google-earth.kml+xml",
  "name": "Fixed artifact"
 },
 {
  "href": "http://example.com/mobile-artifact",
  "rel": "artifact mobile",
  "type": "application/vnd.google-earth.kml+xml",
  "name": "Mobile artifact"
 }
]
```

Having this information, the application running in the browser queries the authoring service for finding out if the service is able to create fixed or mobile artifacts. The authoring service response is positive for both artifact types. When the user presses the *New* button, the application displays a dialog that allows the user to select what type or artifact to create. The `name` property included in the artifact collection document is used for button captions. As the user makes the selection, the application loads the corresponding javascript functionality from the authoring server and creates a new view that is optimized for authoring the respective artifact type.

4. DISCUSSION

In this section we discuss the benefits of using hypermedia for creating applications that have the capacity to adapt to the particular characteristics of environment in which they are used. First we discuss how the information embedded in the representations returned by content servers allows decoupling of authoring and storage services. Then, we discuss how mobile web applications can leverage this information to optimize their performance on the mobile devices.

4.1 Service decoupling

Our tools are developed mainly for augmented and mixed reality application domain. In this context, we are mainly concerned with processing collections of artifacts. The selection of artifact types in our applications is limited to fixed and mobile artifacts, while their representations are typically KML documents with embedded links to Collada models.

Even if our operating environment is relatively homogenous, the information contained in the artifact collection document and the artifact authoring detection enables the decoupling of authoring and storage services. Each service can evolve independently by adding new artifact types or developing new tools for artifact authoring, without harming the evolution of the other. A user-agent could interact with multiple authoring or storage services and still provide the best user experience for the particular situation.

The basic mechanism of informing the nature of the items that they can create in a collection, is valuable to the user-agents in any scenario that requires working with lists or collections. Without prior knowledge, a user agent can create new items in a collection by following the links and filing the provided template.

4.2 Generic and adaptive web applications

Mobile web applications run in a constrained environment. The mobile devices offer relatively limited processing power and memory, and the multi-tasking capabilities provided by the operating system are more restrictive than the desktop counterparts. Therefore the web artifact authoring tools have to reduce as much as possible the amount of memory used. Using the information provided in the artifact collection document and the ability to probe the template support on the authoring server, allows our web authoring tools to load only the javascript libraries and the HTML content that are strictly needed for the task performed by the user. The process allows the application developer to create a generic application that is able to adapt at runtime to the particularities of the environment.

5. CONCLUSIONS

In this paper we described our experiences developing hypermedia-driven authoring applications for augmented and mixed reality domain. Embedding the capabilities of the content servers in their representations, allows the web-based applications to determine the content that can be created and stored on the respective services. The same information can be used to optimize the resources consumed by the the web applications, by loading only the needed functionality required to perform a specific task. Although we used these techniques in the context of augmented and mixed reality application domain, they can be successfully used in other application domains where users are expected to generate content.

Acknowledgements

We would like to thank TEKES for funding this research. We also want to acknowledge the Live Mixed Reality team in Nokia Research Center in Tampere.

6. REFERENCES

[1] M. Amundsen. *Hypermedia APIs with HTML5 & Node*. O'Reilly Media, 2011.

[2] M. Barnes and E. L. Finch. *COLLADA - 3D Asset Exchange Schema, Release 1.5.0*. Khronos Group, 2008.

[3] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of Califormia, Irvine, USA, 2000.

[4] Google. Google Loader Developer's Guide. https://developers.google.com/loader/.

[5] A. Hill, B. MacIntyre, M. Gandy, B. Davidson, and H. Rouzati. Kharma: An open kml/html architecture for mobile augmented reality applications. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*, pages 233 –234, oct. 2010.

[6] P. Selonen, P. Belimpasakis, Y. You, T. Pylvänäinen, and S. Uusitalo. Mixed reality web service platform. *Multimedia Systems*, pages 1–16, nov. 2011.

[7] T. Wilson. *OGC Keyhole Markup Language, 2.2.0*. Open GIS Consortium, 2008.