Annotate That: Preparing Event Posters for Augmentation

Jens Grubert[+] Robert Gründler[‡] Lyndon Nixon[†] Gerhard Reitmayr[+]

ABSTRACT

Authoring Augmented Reality scenes typically involves arranging concrete multimedia assets (e.g., 3D models, images, videos) in a scene's coordinate system. When creating or collecting assets might be too time consuming, or assets are simply non-existent at the moment of annotation the authoring process fails. To overcome this problem, we employ a workflow for connecting semantic concepts, rather than concrete assets, to an augmented scene. This allows deferring the retrieval of resources to the moment when the user interacts with the object. Additionally, the context of interaction can be employed to filter appropriate content.

KEYWORDS: augmented reality, authoring, thing of interest, semantic web.

INDEX TERMS: K.6.1 [Management of Computing and Information Systems]: Project and People Management—Life Cycle; K.7.m [The Computing Profession]: Miscellaneous—Ethics

1 INTRODUCTION

Content generation is one major cost factor during the creation of Augmented Reality experiences. Typical approaches in academia and industry rely on the careful and time intensive creation of suitable assets for a specific user experience (e.g., for a car maintenance scenario) or provide content creators with templates for connecting assets to locations or images as it is common with today's mobile Augmented Reality browsers. While semantic web technologies are already used for dynamic retrieval of relevant content for in browser information access, their use in the authoring process of Augmented Reality experiences has not been widely explored.

Within this paper we propose a workflow and data model for annotating AR scenes with semantic concepts rather than with concrete assets. This postpones the retrieval of assets to the moment when users interact with an augmented scene and allows the inclusion of context factors other than location to filter relevant information. We introduce our authoring workflow and employed data model within the context of a concrete use case, i.e. the augmentation of street posters about music events but believe that our approach will be transferable to other applications as well.

2 RELATED WORK

A common approach in content creation for AR assumes a tight interplay between content creators (e.g., graphic designers, 3D modelers) on the one side and programmers of AR scenes on the other. While this approach is suitable for the generation of highly specialized AR scenes it fails, when appropriate content creators or suitable content is not available, as often the case in real world applications due to time or budget constraints.

For mobile Augmented Reality Browsers vendors like Metaio, Wikitude or Layar tackle the content creation problem by providing APIs for developers to connect multimedia assets to geolocations (through Point of Interests - POIs) or images. Those POIs can then be combined into "channels", "worlds", or "layers". Additionally, predefined behaviors like opening a link or triggering an animation can be defined.

Recent approaches extend the former proprietary approaches to include standard web technologies like CSS or JavaScript for defining custom appearances and functions of Augmented Objects [1][2][10].

Instead of relying on developers to create content, Langlotz et al. propose to leverage social-networking communities for user generated content creation and sharing [10]. This requires even simpler tools for creating content by non-experts.

However, all these approaches require the explicit connection of assets before the AR experience is delivered. We see the need for additionally supporting situations in which content is not known beforehand, e.g. when the content to be presented is dependent on the time of consumption. While there are already approaches for context-aware AR (e.g., [3][9]), they also require the a priori specification of employed assets.

3 ANNOTATING EVENT POSTERS

Within the project SmartReality[1] we try to enhance mobile AR applications with semantic web technolgoies so that they can serve as efficient interfaces to the ever increasing set of publicly available data and services that can be meaningful in the user's current context. An initial testbed for realizing this vision is the augmentation of street posters promoting music events. Play.fm[2] is a web-based on-demand music streaming service with a music community of currently 150.000 users per month. It offers music streams from past DJ events and serves as platform for event promotions. The website provides access to several thousand past events and several dozen that are promoted at any given time. The uploaded events are typically equipped with images of posters promoting the event.

We believe that the augmentation of those posters presents several challenges. Interviews with professional graphic designers revealed that music event posters are often created in a short period of time (~3h) and under tight budget constraints, especially for local events. Additionally they present only one of several communication channels (others being flyers, websites, social network advertising). The additional AR authoring process should

[+]: Institute for Computer Graphics and Vision, Graz University of Technology. {lastname}@icg.tugraz.at
[‡]: Play.fm GmbH. robert@play.fm
[†] STI International. lyndon.nixon@sti2.org

[1] http://www.smartreality.at
[2] http://www.play.fm

therefore not add much time to the overall promotion process and also be independent of the original graphics designer, as in many cases they might just be unavailable. Furthermore, collecting assets that should augment the poster might be too time consuming as well or they also might be simply unavailable at the time of the annotation.

In dealing with these challenges it is useful to notice that event posters typically present the same type of information (i.e. the event name, venue, date, artist information) in a different layout each time. Furthermore, it might be even undesirable to connect concrete assets to a poster as their usefulness can be highly context-dependent. For example one could display images of the band playing at other locations before the event date and impression of the actual event afterwards, the latter being unavailable at the time of annotation.

Currently we are developing an annotation tool that does not require providing actual assets at the moment of authoring and introduce a data model supporting the retrieval of those assets at runtime. The authoring solution presented below is an initial prototype with the goal to ease the annotation of posters. It has to be evaluated with real-world users in the future.

3.1 Authoring Workflow

An overview of the poster annotation process is shown Figure 1 and is carried out in a web interface shown in Figure 2. The user starts selecting an existing event in the Play.fm event database through an auto-completion search. If a poster image is connected to the event it is loaded into the web interface for authoring. Here, the user selects regions for triggering the appearance of content as well as regions where to actually display the content. The regions are specified by dragging bounding boxes over the image (see Figure 2). Both regions can be (partially or completely) overlapping or disjoint. Instead of connecting a concrete asset to the selected regions users rather select a Linked Data URI representing a concept (such as date, venue, artist) from an existing conceptual scheme. The user however, has the option to link to any URI representing a concept that is part of any conceptual scheme (such as the DBPedia [1], which gives an URI to every concept which has a Wikipedia page). The URIs can be connected to individual regions as well as to the poster as a whole. This makes it possible to retrieve information that are not specific to an individual graphical element on the poster (e.g., information about the event as a whole).

For the use case of event posters we provide a specific look-up of conceptual instances defined in the play.fm Linked Data dataset. When editing is finished, the annotation tool generates a Thing of Interest (TOI) data model (described next) for the poster and stores it in a TOI repository (see Figure 1). Additionally, the image of the event poster is uploaded to a third party image recognition service [4], which is used to identify the poster at runtime.

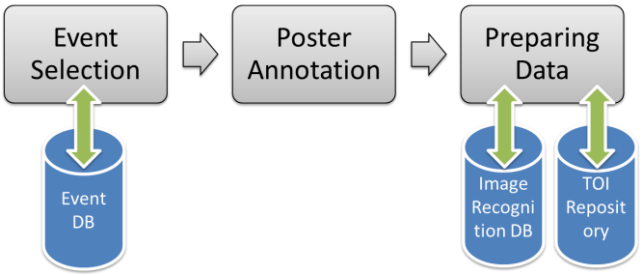

Figure 1: Annotation Workflow with involved data sources.

3.2 Thing Of Interest Data Model

The basis for linking dynamic content to an augmented scene is the TOI data model (see Figure 3), which can be seen as an extension of a Point of Interest in mobile Augmented Reality browsers [8]. A TOI itself is an entity representing something in the real world which is intended for triggering some informational display in an AR browser when a user is in the vicinity of the thing. The TOI data model is intended to capture the core information about such objects and links, via its description and metadata about what it represents, to web contents and services. The actual mapping from concepts to concrete content is carried out by a server processing the TOI and is not dealt with in the scope of this paper. Important parts of the model as well as their application in the annotation of the poster are described next.

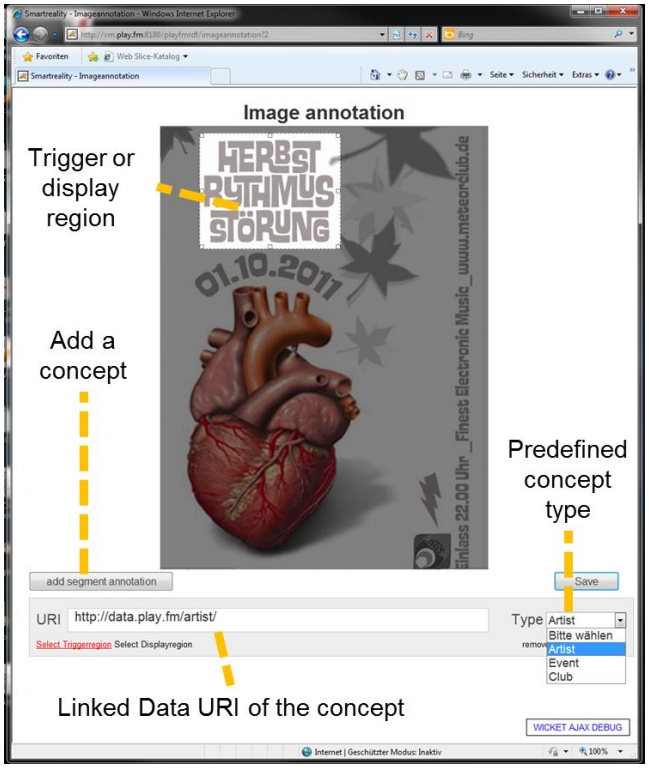

Figure 2: The annotation web interface.

3.2.1 TOI

The thing itself is an abstract notion, which exists purely in the context of usage in an AR application where a visual or geolocated object (described in the annotation) is used to trigger some informational display in an AR browser when a user is in the vicinity of the thing.

3.2.2 Annotation

As the TOI is an abstract notion it requires a structured description used by a computer system to link it to other information. This is achieved through the annotation class which links the TOI to a specific location, representation, and to what the TOI actually represents – i.e. the semantic concepts which reflect a human's interpretation of the meaning of the TOI.

3.2.3 Segment_Annotation

The Segment_Annotation can be used to link parts of an object to representations and to what they represent just as the annotation does for the whole object. For example, different parts of the poster refer to different concepts like, venue, artist or date. In the authoring process the Segment_Annotation replaces the connection of a region in a scene's coordinates system to a concrete asset by a connection between a region (representation) and a concept (represents).

3.2.4 Representation

The Representation of the TOI is the information used to 'ground' the TOI to a physical entity. For example for a street poster as a whole the representation could be an image file of the poster. For segments of a poster the representation encompasses the position and extend on the poster where to trigger (location, dimension) and where to display (displayLocation, displayDimension) the information.

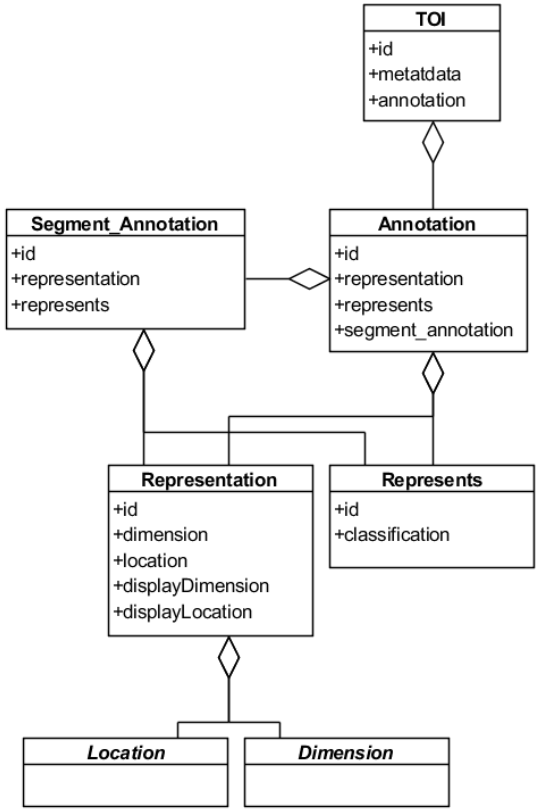

Figure 3: Important classes (and their relationship) used for modeling a Thing Of Interest.

3.2.5 Location and Dimension

The Location class is an abstract base class for specific means to identify a thing's location. For the location of a TOI we currently support the WGS1984 coordinate system and tracking data for a proprietary 6 DOF tracker (represented by the image of the poster itself). The locations of segment annotations of a planar object like a poster image are expressed in a coordinate system ranging from [0…1] in x-direction and [0…aspectRatio] in y-direction with the point of origin being in the upper left corner of the object. The dimension refers to the 2D or 3D spatial extent of the annotated object or it's parts.

3.2.6 Represents

This describes concepts which are being represented to a human by the object used by the TOI. The classification member is an URI pointing to a class from an existing ontology (e.g. MusicArtist from the Music Ontology [6]), and id is an URI representing an instance of that class in some conceptual schema (e.g. http://data.play.fm/artist/Makossa). We do not attempt here to address whether the object itself directly represents such concepts to a human or if the object is being used to represent some concept to the human in the context of being a TOI in an AR application (both interpretations are possible). So for example, while a street poster would represent a particular music event, a sticker on a wall could just as easily be used to represent some arbitrary content in the app if desired. This is also why the representation and what it represents are only connected via the annotation of the TOI; it may be outside of the TOI context they are not related at all. As part of a Segment_Annotation the Represents points to concepts associated to the annotated regions, such as artist, venue, or date. These concepts are the starting points for retrieval of actual assets and services that are finally presented to the user.

3.3 Client Side Content Retrieval

Once the poster is annotated and the image is indexed by the external image recognition service, a user is able to point its smartphone to a printed version of that poster. The overall workflow of content retrieval during runtime is presented in Figure 4. The users first point their smartphone to the poster. As soon as the poster is recognized, its ID is used to lookup the TOI in a repository by the SmartReality server. The server sends back JSON encoded initialization data to the client. This data encompasses information about trigger, display regions, URIs of initial icons to display (as the actual type and content is still unknown at this point), and a URI to the tracking data (in our case an image of the poster itself). After this initialization placeholder assets can be overlaid in 3D over the regions that will subsequently show the actual media. As soon as the SmartReality server has fetched content via Linked Data resources or web services they are sequentially transferred in a content bundle to the client. The content bundle encompasses URIs to the actual data, links to the semantic information such as importance or how content items interrelate with each other, and information on functions to execute.

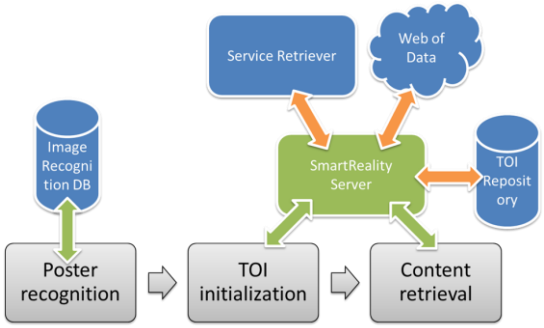

Figure 4: Data retrieval workflow during an AR session

This data then is used on the client to fetch the assets and to make layout decisions how the content should be rendered (e.g., to determine the order in an image collection). The process of content and web service retrieval through the SmartReality server as well as the actual rendering and interaction are still under development and will not be presented in detail here. For data collection it is however worthwhile to note, that we use the concept's classification (e.g., http://data.play.fm/artist) to narrow the amount of Linked Data that has to be crawled. While information collection can be done generically for any Linked Data concept by iterating over all linked URIs to a certain depth or triple limit, such as is implemented in the Linked Data crawler LDSpider [5] we define simple rules (using RDF/N3) which are parsed in the SmartReality platform. This way we can narrow down the amount of information that has to be crawled. However, this comes at the additional cost of specifying rules for different concept classes.

4 Discussion and Conclusion

While authoring AR scenes with a priori available assets gives the most control over the final appearance of an AR experience there exist use cases in which those assets are not available at the moment of creation or the process might become too time consuming if assets have to be created and connected. We presented an authoring workflow and initial data model for augmenting music event posters in an existing database. We employ Linked Data URIs pointing to entities in ontologies representing concepts – rather than final content. While this approach allows for a flexible retrieval of content at runtime based on the context of the user it also presents several challenges.

For example, the latency between initialization of the AR experience and the presentation of the actual content will likely be higher as with current AR browsers, as the content has to be additionally fetched and filtered by a server before being delivered to the client. While the content itself can be dynamic we still need to employ widgets for presenting it on the client (e.g. video player, slideshow, text view) and need to encode the functional behavior. Those widgets however still have to be created beforehand. Currently, we use proprietary widgets, but would like to support an easier creation of custom widgets, potentially based on web technologies like CSS and JavaScript, similar to existing approaches [2][10]. Finally, the issues of context-dependent content and service retrieval and filtering is a complex research issue on its own which we will continue to explore in the future.

Acknowledgements

This work is made possible by the Austrian National Research Funding Agency FFG in the SmartReality project.

References

[1] DBPedia. wiki.dbpedia.org. http://dbpedia.org. 2011.
[2] Hill, A., Macintyre, B., Gandy, M., Davidson, B., and Rouzati, H. 2010. Kharma: An open kml/html architecture for mobile augmented reality applications. In Proc. ISMAR 2010, 233 –234.
[3] Julier, S., Baillot, Y., Brown, D., and Lanzagorta, M.. Information filtering for mobile augmented reality. In Proc. ISAR 2000, 3-11.
[4] Kooaba. kooaba: Image Recognition and Visual Search http://www.kooaba.com. 2011.
[5] LDSpider - Java implementation for a linked data web crawler http://code.google.com/p/ldspider/. 2011.
[6] Raimond, Y., and Giasson, F.. Music Ontology Specification. http://musicontology.com/. 2010.
[7] Schmalstieg, D., Langlotz, T., and Billinghurst, M.. Augmented Reality 2.0. In *Virtual Realities*, Springer 2011, 13-37.
[8] SmartReality. Things of Interest | Smartreality RDF. http://www.smartreality.at/rdf/toi. 2011.
[9] Toro, C., Sann, C., Vaquero, J., Posada, J., and Szczerbicki, E.. Knowledge based industrial maintenance using portable devices and augmented reality. In LNCS, vol. 4692. Springer 2007, 295–302.
[10] Wikitude GmbH. ARchitect Features: http://www.wikitude.com/developer/overview/architect/architect-features. 2011.