CARL – A Language for Building Contextual Augmented Reality Environments

Dariusz Rumiński and Krzysztof Walczak

Poznań University of Economics,
Niepodległości 10, 61-875 Poznań, Poland
{ruminski, walczak}@kti.ue.poznan.pl
http://www.kti.ue.poznan.pl

Abstract. The paper describes a novel language that enables modelling ubiquitous, multi-device, contextualized augmented reality environments. The language, called CARL – Contextual Augmented Reality Language, is highly componentized both with regards to the structure of the AR scenes as well as the presented contents. This enables dynamic composition of CARL presentations based on various data sources, depending on the context. CARL separates specification of three categories of entities constituting an AR environment – physical markers, virtual objects and physical interfaces, which makes the language more flexible and particularly well suited for building collective awareness systems based on ubiquitous AR-based information visualization.

Keywords: Augmented Reality, AR, CARL, AR Services

1 Introduction

Augmented reality (AR) technology enables superimposing rich computer generated content, such as interactive 3D animated objects and multimedia objects, in a real time on a view of a real environment. Augmented reality enables building advanced localized information visualization systems and therefore forms a solid basis for the development of collective awareness systems.

Widespread use of the AR technology has been enabled in the recent years by remarkable progress in consumer-level hardware performance, in particular, in the computational and graphical performance of mobile devices and quickly growing mobile network bandwidth. Education, entertainment, e-commerce and tourism are notable examples of application domains in which AR-based systems become increasingly used.

There are a variety of tools available for authoring AR content and applications. These tools range from general purpose computer vision and graphics libraries, requiring advanced programming skills to develop applications, to easy-to-use point-and-click packages for mobile devices, enabling creation of simple AR presentations. However, the existing tools are designed for manual authoring of AR presentations – either through programming or visual design. To enable more widespread use of AR technology for visualization of different kinds of
up-to-date data, needed in collective awareness systems, a different paradigm is required. The presentations must be created automatically based on the available data sources, through selection of data and automatic composition of AR scenes. A key element enabling selection of information for visualization is context, which incorporates aspects such as user location, time, privileges, preferences, device capabilities, environmental conditions and others. To enable meaningful contextual selection and automatic composition of non-trivial interactive AR interfaces, semantic web technologies may be used.

In this paper, we present foundations of a novel high-level programming language, called CARL – Contextual Augmented Reality Language, which enables building ubiquitous contextual AR presentations. In the presented approach, the AR content that is presented to users is created by selecting and merging content and rules originating from different AR service providers, without the necessity to explicitly switch from one to another. Moreover, in CARL, the rules for tracked objects (where the information will appear), presentation objects (what will be presented) and interfaces (how the information will be accessible) is separated, enabling flexible composition of presentations meeting real business requirements.

The reminder of this paper is organized as follows. Section 2 presents related works in the context of building AR-based visualization systems, in particular focusing on the use of AR for building collective awareness systems. Later, in Section 3, the concept of building ubiquitous contextualized AR presentations is explained. In Section 4, the CARL language is presented with examples of application. In Section 5, the current implementation of CARL is outlined. Finally, in the last section, conclusions and directions for future research are presented.

2 AR Environments and Collective Awareness Systems

A number of research works have been devoted to the problem of modelling and building AR environments combining the views of real-world objects with synthetic and multimedia content. One of the well-known toolkits that supports rapid design and implementation of AR applications is DART – Designer’s Augmented Reality Toolkit [1]. DART supports both visual programming and a scripting interface.

Many AR applications are based on the ARToolkit library, which requires advanced programming skills and technical knowledge to create AR applications [2]. GUI-based authoring applications that enable mixing virtual scenes with the real world have been presented in [3–8]. These systems provide user-friendly functionality for placing virtual objects in mixed reality scenes without coding and can be used for building AR environments, but they target only desktop computer environments and do not provide functionality to build open, interoperable AR environments.

Stiktu is a mobile AR authoring application that enables attaching (‘sticking’) simple content, such as text messages and images, to particular views of real places [9]. Stiktu is appropriate for non-technical users. It also enables creation
of a social network, where users can express and exchange their opinions with other users who have created virtual objects. The main limitation of this application is the lack of a possibility of creating complex objects, such as interactive 3D models.

Aurasma is an application that enables augmenting arbitrary real-world views using a mobile device’s camera [10]. First, a user needs to select a digital object from a library, e.g., an image, a video or a predefined 3D model. Next, the user needs to capture an image that will serve as a marker for accessing the digital content. The last step is to position the content by adjusting and rotating the digital object until it gets the most appropriate fit. After saving these properties, a so-called ‘Aura’ is available for use. The user can make his/hers ‘Aura’ public, so it becomes publicly available. Aurasma implements also social tools that provide functions that help to share ‘Auras’ with friends.

Layar, Junaio and Wikitude are other examples of AR applications that enable users to share AR content [11–13]. These platforms are good examples of collective awareness applications, where users contribute to ‘augmenting’ the real world with virtual objects. Their growing popularity indicates that there is public interest in new forms of information visualization, even if the functionality of these platforms is still highly limited.

Several studies have been performed in the domain of declarative languages for building AR environments. For example, ComposAR-Mobile is a cross-platform content authoring tool for mobile phones and PCs [14]. It allows users with programming skills to create simple mobile AR applications on a PC. It generates an XML file, which describes the AR scene. InTml is a domain-specific language based on XML that describes VR and AR applications [15]. It supports Mixed Reality development for multiple hardware setups, in order to facilitate rapid prototyping or application migration. MPML–VR enables describing multimodal presentations in 3D virtual spaces [16]. MR–ISL is a high-level language used for defining interaction scenarios [17]. The language enables creation of interaction scenarios describing possible user interactions and system reactions. VR–BML is a high-level XML-based language developed as a part of the BehVR approach [18]. Using VR–BML, a content-designer can declare what happens when a VR object is initialized, what actions can be performed by the object and what are responses of the object at specific points in time or as a result of user actions or interactions with other objects. AREL is a JavaScript binding of the metaio SDK’s API in combination with a static XML content definition. AREL allows scripting interactive AR experiences based on common web technologies—HTML5, XML and JavaScript [19].

A common motivation for developing novel platforms and declarative AR languages is to support developers by simplifying the process of programming concrete AR applications. However, the existing languages are not intended for building dynamic information visualization systems needed for creating collective awareness systems. In such systems, different types of data must be retrieved in the real time from distributed sources and automatically composed into meaningful interactive AR scenes.
3 Contextual Augmented Reality Environments

Existing AR platforms support two main forms of augmentation. In the first form of augmentation—directional augmentation—information is associated with points of interest in physical locations in the real world and is presented on mobile devices as directions how to navigate to these locations (e.g., hotels or restaurants in Wikitude and Junaio). Coordinates of the mobile device are monitored through GPS or other sensors. This form of augmentation is usually functionally and graphically very simple—it rarely goes beyond displaying simple textual descriptions and the content is not geometrically registered in the real environment.

The second form of augmentation—natural augmentation—is much more powerful. Position and orientation of content to show is identified directly through a device’s camera by tracking known markers (fiducials) in the captured images (markers can take the form of natural images). This kind of augmentation can be based on complex interactive 3D content, which can additionally incorporate other multimedia content, such as images, sounds and videos. The presented objects are geometrically integrated with particular places in the real environment extending it with presentation of different types of data. However, a common problem with the development of systems based on natural augmentation is their limited scope with regards to the number of tracked patterns (markers) and the amount of the presented content. These types of systems can be successfully used as interactive extensions to printed advertisements or animated content enriching books, but their wider application for creating practical information visualization and collective awareness systems is largely limited.

As a solution to this problem, we propose the notion of CARE—Contextual Augmented Reality Environments. CARE is a new approach to building augmented reality applications, which combines the advantages of both directional and natural augmentation systems. In CARE, the augmented reality scenes that are presented to users are based on natural augmentation, but they are dynamically composed in the real time based on a variety of data sources and the current context, which includes such elements as location, time, user privileges and preferences, device capabilities and environmental conditions. Therefore, the same application can be used to display elements coming from different content- and service-providers. Contextualized approach to AR application development is a necessity to enable access to different data sources, guarantee scalability and provide for seamless operation.

To enable implementation of CARE environments, a language is required that would enable passing augmentation information from distributed service providers to the AR browser. Such a language, must be designed to support dynamic composition of complex interactive AR scenes, and must enable passing information about different entities—trackable objects, augmentation content and interfaces. In practical applications, these aspects may be separated. For example, trackable objects may come from municipal services, augmentation content from service providers, while interfaces from application developers. Only in such heterogeneous environment, large usable AR environments may be realized.
In this paper, we present a language, called CARL—Contextual Augmented Reality Language, that meets these criteria. The language is presented in details in the next section.

4 The CARL Language

CARL enables specification of three types of entities that form an AR presentation—Trackables, Objects and Interfaces. The independence of trackables, objects and interfaces in connection with the dynamisms of each of these elements requires the use of loose coupling between descriptions of these entities. The proposed solution to this problem is the use of semantic modelling and semantic linking, which is however out of the scope of this paper. Below, the main elements of the CARL language are presented.

4.1 Trackables

The Trackables element, shown in listing 1.1, groups Trackable objects that are responsible for identifying all real-world objects that an application can detect and track in 3D space. Every Trackable element has its unique uri, which identifies its binary resource data used in the tracking process. The Trackable element has three child elements: Begin, Active and End.

**Listing 1.1. Specification of trackables in CARL**

```xml
<Trackables>
  <Trackable uri="http://carl.ar/tr1.trackable">
    <Begin>
      <ObjectBegin id="1"/>
      <ObjectBegin id="2"/>
      <InterfaceMessage value=""/>
    </Begin>
    <Active>
      <DistanceChanged change="0.1" distanceMin="2" distanceMax="5" angleMin="30" angleMax="90" heightMin="0" heightMax="10">
        <ObjectAction id="1" action="show"/>
        <ObjectAction id="1" action="playAnimation"/>
      </DistanceChanged>
    </Active>
    <End>
      <ObjectEnd id="1"/>
      <ObjectEnd id="2"/>
    </End>
  </Trackable>
  <Trackable uri="http://carl.ar/tr1.trackable"/>
</Trackables>
```

In the runtime, when an application detects a trackable object, the Begin section is executed. In the presented example, it initializes Objects using ObjectBegin and displays a message using InterfaceMessage. In turn, when an application tracks a trackable object, it can trigger actions depending on specific conditions declared within the Active element, such as device position change relative to the tracked marker. For instance, when a user watches some 3D content and he or she changes his/her view (relative device position), an application can recognize this fact and can trigger object actions, such as presentation of additional content or animation.

The End tag contains specification of actions, which should be performed when an application loses track of the trackable object.
4.2 Objects

The second element of AR environment specification in CARL are *Objects*, containing multimedia content to be shown to end-users while augmenting real-world objects (listing 1.2). Each *Object* has an *id*, which is used while calling object actions. The *id* can have a form of a fixed literal or semantic expression to enable loose coupling with other elements of the dynamically created augmented reality scenes.

Objects support states. The initial object state is specified by the `state` attribute. The state can be changed by the use of the `ObjectState` command in any of the actions. The `Resources` element contains information about particular `Components` and `Locations` of component resources. Several locations can provide alternative URIs for components meeting particular criteria, e.g., regarding the model complexity or language.

The `Actions` element specifies actions that can be called on this object. An action is declared within the `Action` element, which has a mandatory attribute `name` and an optional attribute `state`. If the `state` is specified, the action (if called) will be executed only when the object is in the given state.

Within the `Action` element, commands based on the VR-BML [18] language are used. These commands can be used to load, manipulate and animate object components, change object state, execute interface operations and control other objects.

Listing 1.2. Specification of presentation objects in CARL

```xml
<Objects>
  <Object id="1" initialState="hidden">
    <Resources>
      <Component id="c1">
        <Location details="low" uri="http://opus.ar/o/badge.l.res"/>
        <Location details="medium" uri="http://opus.ar/o/badge.m.res"/>
        <Location details="high" uri="http://opus.ar/o/po/badge.h.res"/>
      </Component>
    </Resources>
  </Object>
</Objects>
```

4.3 Interfaces

The third element of the CARL specification are *Interfaces* (listing 1.3). The *Interfaces* element contains specification of interactions on an end-user device. Listing 1.3 shows examples of touch, pinch and pan gestures that can trigger actions. The interface can recognize interactions with objects and interactions with trackables. Interaction with objects enables activation and manipulation of objects. Interaction with trackables can be used, e.g., to display objects associated with the tracked marker (e.g., information about a known product).

Listing 1.3. Specification of interactions in CARL

```xml
<Interfaces>
  <Action name="show" state="hidden">
    <SetPosition component="c1" value="0, 10, 0"/>
    <Activate component="c1" active="true"/>
    <ObjectState value="shown"/>
  </Action>
</Interfaces>
```
5 Implementation

CARL has been implemented within the Mobile Augmented Reality Authoring Tool (MARAT) for virtual museum exhibitions [20]. Currently the application enables only playback of the CARL presentations, but the intention is to implement also content authoring directly on a mobile device within MARAT.

The following figures show examples of CARL usage in the MARAT application. Figure 1 presents an example when an end-user moves his/her finger on a device screen. A Pan event that occurs triggers an action responsible for object rotation.

![Figure 1](image1.png)

**Fig. 1.** Triggering the ObjectRotate action on Pan event

Figure 2 shows examples of using a Pinch gesture, which triggers an action responsible for object scaling (ObjectScale).

6 Conclusions

In this paper, the overall concept of Contextual Augmented Reality Environments and fundamentals of the CARL language, designed to support creation of such environments, have been presented. CARL enables modelling ubiquitous,
multi-device, contextualized AR environments. The language is highly componentized with regards to the structure of the AR scenes as well as the presented content, enabling dynamic composition of complex AR environments. The CARL language can support building ubiquitous AR-based data visualization and collective awareness systems.

Future research will incorporate several facets. First, the method of semantic coupling of elements of dynamic AR environments will be elaborated. Second, methods of combining CARL specifications from independent sources will be investigated, with particular focus on security. Finally, methods of automatic creation of CARL specifications through WYSIWYG actions on mobile devices, to enable in-situ authoring of AR environments, will be designed.

Acknowledgements

This research work has been supported by the Polish National Science Centre Grant No. DEC-2012/07/B/ST6/01523.

References

2. ARToolKit, http://www.hitl.washington.edu/artoolkit
19. AREL - Augmented Reality Experience Language http://dev.metaio.com/arel/overview/