The Evolution of the Argon Web Framework Through Its Use Creating Cultural Heritage and Community–Based Augmented Reality Applications

Gheric Speiginer[1], Blair MacIntyre[1(✉)], Jay Bolter[2], Hafez Rouzati[1], Amy Lambeth[3], Laura Levy[3], Laurie Baird[4], Maribeth Gandy[3], Matt Sanders[6], Brian Davidson[5], Maria Engberg[7], Russ Clark[5], and Elizabeth Mynatt[1,4]

[1] School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA, USA
{gheric.speiginer, blair, hafez, mynatt}@gatech.edu
[2] School of Literature, Media, and Communication, Georgia Institute of Technology, Atlanta, GA, USA
jay.bolter@lmc.gatech.edu
[3] Interactive Media Technology Center, Georgia Institute of Technology, Atlanta, GA, USA
{amy, laura, maribeth.gandy}@imtc.gatech.edu
[4] Institute for People and Technology, Georgia Institute of Technology, Atlanta, GA, USA
laurie.baird@ipat.gatech.edu, mynatt@gatech.edu
[5] Research Network Operations Center, Georgia Institute of Technology, Atlanta, GA, USA
{bdavidson, russ.clark}@gatech.edu
[6] Office of Information Technology, Georgia Institute of Technology, Atlanta, GA, USA
msanders@gatech.edu
[7] Department of Media Technology and Product Development, Malmö University, Malmö, Sweden
maria.engberg@mah.se

Abstract. The Argon project was started to explore the creation of Augmented Reality applications with web technology. We have found this approach to be particularly useful for community-based applications. The Argon web browser has gone through two versions, informed by the work of our students and collaborators on these kinds of applications. In this paper, we highlight a number of the applications we and others have created, what we learned from them, and how our experiences creating these applications informed the design of Argon2 and the requirements for the next version, Argon3.

Keywords: Augmented reality · Web-based technology · Community computing

M. Kurosu (Ed.): Human-Computer Interaction, Part III, HCII 2015, LNCS 9171, pp. 112–124, 2015.
DOI: 10.1007/978-3-319-21006-3_12

1 Introduction

The Argon project (http://argon.gatech.edu) started as an industry-funded research program in 2009, aimed at understanding how web technology can be used to create mobile augmented reality (AR) applications [1], and was inspired by our previous work on the Real-World Wide Web [2] and by Spohrer's Worldboard [3]. The cornerstone of the project is the Argon iOS application, a web browser with AR capabilities. By integrating with the web, the Argon project hopes to provide AR capabilities within a mature application ecosystem, rather than focusing on creating yet-another environment for developing "AR applications".

Argon applications are based on the Web application model: each AR-enabled web application is fetched from a web server at a known URL (just as any web page or web-application), and is implemented with client-side Javascript to control the interface and communicate with remote web services. Over the years, Argon developers have implemented application prototypes across a variety of domains, and the Argon architecture has changed in response to their experiences.

In this paper, we discuss how the Argon framework evolved from Argon1 to Argon2, and the requirements we have identified for the next version (Argon3), based on the experiences of developers working with applications in the areas of community computing and cultural heritage. These domains have been a key motivator for this project from the beginning. Our work in historic sites over the years (e.g., Historic Oakland Cemetery [4] and the Auburn Avenue area of Atlanta [5]) has shown us how cultural heritage organizations often have significant amounts of pre-existing web-based content that they would like to re-purpose for an AR experience. For them, AR is exciting when used at the right moment, but much of the content (images, videos, maps) will be most useful when presented using traditional 2D display techniques. AR is one piece of a larger information system in which content will be accessed on conventional web browsers before and after visits to their sites. For similar reasons, a web-based solution is very appealing for many organizations, not just cultural heritage sites.

The key lesson we learned through the projects discussed in this paper is the value of tightly and cleanly integrating with the web ecosystem. The major difference between Argon1 and Argon2 is the format of the content that the web server delivers to the browser. In Argon1, the content is KARML [6] (a version of KML [7], the location-based XML format for Google Earth and Maps, that we extended to include AR-relevant elements). In Argon2, we change our approach and focused on cleanly integrating our tools with standard HTML5 content (with AR capabilities presented via the argon.js Javascript library). This change has had significant benefits, as we discuss throughout the paper, and in Argon3 we will continue to refine our HTML5-based approach to better leverage the web ecosystem.

2 Argon Version 1

In this section, we will give a brief overview of Argon1, describe a selection of AR applications created with it, and discuss how our experiences influenced the design of Argon2, the second version of Argon.

The first version of the Argon browser (referred to here as Argon1) has been available for iOS since February 2011. It has been used in dozens of research and class projects at Georgia Tech and elsewhere, despite modest capabilities. While Argon1 uses a standard iOS WebView for running user-created applications, we did not use HTML5 as our markup language for AR content (see [1] for architecture details). Rather, we created KARML [6] to define "channels" of AR content. With KARML, developers create AR content elements (positioned at 3D geo-locations or on simple AR markers) using the full collection of contemporary web standards (HTML5, CSS3, Javascript, etc.), using the tools, techniques, and server-side technologies they are already familiar with.

(a) (b) (c) (d)

Fig. 1. (a) The Voices of Oakland experience, showing an image of a "ghost" at a grave site on the tour. (b–d) The Lights of St. Etienne, ported to Argon2. (b) The main interface, (c) the navigational map, and (d) 3D model of the cathedral in approx. 1380 AD.

A novel capability of Argon1 (and Argon2) is that multiple channels can be displayed at once, by overlapping transparent web views. This is a first step toward an AR ecosystem where all AR content from independently-authored AR experiences is safely and simultaneously available to the user in one merged space. We will not discuss this further, as it has little impact on individual channel authors. Using multiple channels to create new kinds of user-experiences is a topic we are pursing in our future research.

2.1 Applications Built with Argon1

In this section, we highlight some projects that informed the requirements of Argon2.

The Voices of Oakland. *The Voices of Oakland* is a prototype AR tour designed to introduce visitors to the history and architecture of Oakland Cemetery, Atlanta's oldest cemetery [4]. The system was originally developed using Macromedia Director on a Windows XP laptop, but was ported to run on the iPhone3GS using Argon1. The user was able to walk among the graves and listen to the voices of historical figures interred

at Oakland (voices were provided by voice actors from scripts written for the prototype). The user was guided from grave to grave by a narrator (although Argon1 supported GPS, the prototype did not use it, but relied instead on timed directions, much like a traditional audio tour). At each grave site the user was invited choose audio segments, and could delve deeply into different categories depending on their particular interests.

The channel functioned in two modes. If the user was at the cemetery, only audio was provided. If the user was at a remote location, the channel could include panoramas of the cemetery to provide a virtual tour. The code therefore consisted of a screen-overlay (buttons that accessed the audio and panoramas), photo-overlays (panoramic images), and geo-located audio placemarks (where the audio was conceptually located for playing). The interface functionality was implemented with HTML5 and Javascript (see Fig. 1(a)). The prototype was limited to four graves and the walk between them. This prototype was created relatively quickly and demonstrated the ease with which existing content elements can be turned into an AR application using a web-based toolset.

The Lights of St. Etienne. A more ambitious tour prototype, *The Lights of Saint Etienne* was developed with the participation of students at Georgia Tech Lorraine in France. This channel served as a guided tour around the cathedral in Metz, France. The purpose was to explore a variety of different presentation modes for historical/cultural information, including interactive panoramas, still images, text, and audio. There were six main interface screens, all written in HTML5. The main interfaces were traditional 2D web content, with the AR view revealed only when appropriate.

A map indicated five sites around the cathedral the user could visit, together with the kind of information (panoramas, images, text, audio) that was available at each site. As with the Voices of Oakland, the tour did not use GPS (it was designed to be used inside the cathedral), so the user had to tell the app where she was by clicking on a map of the cathedral. An interesting feature of the tour was the use of historical panoramas at sites 1 and 2 on the map: these were pre-rendered from simple 3D models showing how the architecture of the church has changed over the period from 1207 to the Renaissance and beyond.

This prototype showed that it was relatively easy to integrate non-trivial HTML5 and Javascript into the KARML framework. The interface consisted of numerous HTML5 DOM elements, sliced images, buttons that called Javascript code for interactive elements, and so on, just as any interactive web application would do. See Fig. 1 (b–d) for images of the Argon2 version of "The Lights of Saint Etienne", which is identical to the Argon1 version.

Campus Tour. *Campus Tour* is an augmented reality tour of Georgia Tech's campus implemented for iPad using Argon1, shown in Fig. 2(a–b). The tour gives information to users through geo-located text, pictures and videos. Our aim was to create an interactive and unique tour experience for prospective students, visitors, or anyone interested in Georgia Tech. *Campus Tour* can be viewed remotely via the use of panoramic images associated with a geospot (geo-located panorama), or using live video from the iPad's camera. The participant can select from a number of tours, see their progress, path and tour stops on a map, and delve into content that interests them.

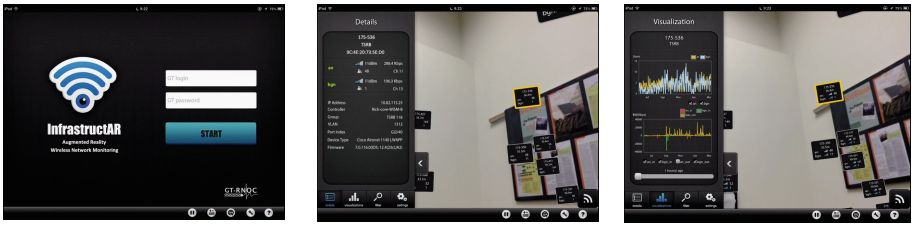
Campus Tour is a fully dynamic web application. A server-side authoring tool allowed the content and tours to be managed from a desktop computer, and the server generated the tour application on the fly for the user. *Campus Tour* is one of most complex AR applications developed with Argon1, and proved to us that this model was useful for more than simple prototypes.

InfrastructAR. *InfrastructAR* was designed to reveal the typically hidden wireless network performance and connection information to anyone in the Georgia Tech community. Unlike other tools which visualize wireless information sensed directly from a mobile device, this application was designed to fetch and display real-time data from the wireless network management system itself. This dynamic content was rendered using HTML5 tools to visualize performance and activities of nearby wireless access points, as well as aggregate views of wireless performance in remote buildings (Fig. 3).

Fig. 2. Georgia Tech Campus Tour. (a) An overview map of the currently selected tour. (b) A stop on the tour, showing a mix of 2D content on the slide-in tab, and 3D content aligned with a panoramic image of the tour stop. (c–d) The second version of the tour, created in Argon2.

InfrastructAR is interesting because it uses AR to represent signals and activity in physical space that is changing dynamically, but not directly associated with a single visual object. Rather, the visualization represents the interaction between the wireless medium and any number of clients associated with the access points in a location.

Through this prototype, we came to appreciate that community applications go beyond user generated content to include community data where both operators



(a) (b) (c)

Fig. 3. InfrastructAR, an AR application designed to help the Georgia Tech community understand how the hidden WiFi infrastructure works. (a) shows the login screen based on Georgia Tech's web-based authetication system, (b) shows the details of the access point, and (c) is a real-time visualization of the network traffic flowing through that access point.

(experts) and end users are able to demystify wireless to gain a better understanding of current state and issues. Additionally, the system demonstrated how easy it was to integrate existing web services (both authentication and real-time, location-based data sources) into an Argon1 application.

2.2 Discussion of Argon1

Argon's use of the web-programming model proved to be both powerful and flexible, allowing developers to use existing web architectures for publishing AR content, ranging from static KARML files hosted on a simple web server, to dynamically generated KARML using client-side processing and AJAX communication with remote web services. Virtually any web architecture that worked for standard web content was possible to use with KARML in Argon1.

From KARML to HTML. Despite the power and flexibility afforded by a web application model, KARML proved to be awkward for most web developers to work with. The problem was not so much with KARML itself, but rather the AR-centric application model it imposed upon developers. This became a problem in each of these applications. Each had non-trivial HTML-based GUIs (including interactive charts and graphs in *InfrastructAR*), but a relatively simple AR view. Nevertheless, the entire application structure and logic had to be embedded in a KARML document, which was unintuitive for an application where AR was not the primary mode of interaction.

Also, KARML only worked in Argon1, which was problematic for two reasons. First, it made debugging extremely difficult because Argon1 content could only be viewed on a mobile device, preventing users from leveraging the increasingly sophisticated development and debugging tools available on desktop systems. Second, to build an application which worked in conventional browsers as well as Argon would have required two different application structures, one based on HTML5, and the other based on KARML, even if both versions were largely the same (except for AR features). For these reasons, we *switched to an HTML5-based approach in Argon2*.

Panoramas. Argon1 supported a panoramic-image-backgrounds feature, which was initially intended for rapid prototyping: a programmer could use panoramic images instead of live video while working on various content elements for their AR application without having to go to the site.

But panoramas soon proved valuable to experience designers, as shown in the *Lights of St. Etienne* (and the other tours), especially in light of the poor quality of the GPS and orientation sensors on mobile devices. Putting interactive panoramas in Argon made it possible to show the user the contrast between the environment she sees around her (e.g. the Cathedral as it is now) and some other condition (historical differences), without having to create fully interactive 3D models. Interactive panoramas used on a mobile device may constitute a kind of bridge between AR and VR, which is a very interesting design space that can incorporate the strengths of both these technologies.

The designers' use of panoramas drove us to *expand their capabilities* in Argon2, and to *extend this concept to other kinds of image-backgrounds* like Google Street-view.

Cross-Platform. The *Campus Tour* in particular highlighted the need for Argon content to be viewable on multiple platforms, because we wanted to embed the tour on a website to support remote visitors to the campus. Other students working with Argon expressed interest in head-worn displays, other operating systems such as Android, and in being able to run something like Argon on the desktop for debugging and authoring reasons. For these reasons, we *re-architected Argon2 to simplify porting to different platforms*.

2.3 Requirements for Argon2

The main requirements for Argon2, based on our Argon1 experience, were:

- Switch from KARML to more conventional HTML5-based web applications
- Add full 3D graphics support to allow richer content (using WebGL)
- Support natural feature tracking (using Qualcomm's Vuforia SDK)
- Add multi-device support (desktop, HMD, etc.)

3 Argon 2

Argon2 was completely rewritten with a new architecture to facilitate ease of porting to new platforms, and experimentation with different browser interfaces. The main change was a switch from one iOS WebView per channel to a single WebView with an embedded HTML5 iFrame per channel, and with the entire user-interface implemented with HTML5 in the WebView. While not a user-visible change, this made it easier to support our other requirements above.

Argon2 abandoned KARML, and instead exposed AR functionality through a Javascript library (argon.js) loaded from the HTML page. This approach allows AR web applications to be created in exactly the same way as traditional web applications, limiting the confusion created by an AR-centric application structure. This approach

was also intended to allow web applications using argon.js to run on popular mobile and desktop browsers (with a more limited feature set), but unfortunately support for other browsers was never completed with Argon2.

Argon2 is built on top of Qualcomm's Vuforia computer vision SDK. Considerable effort was required to architect it in such a way that the constant stream of data into the iOS WebView does not overwhelm the system. By carefully managing data flow and batching all DOM operations for each update (to prevent extra layout and rendering operations), Argon2 web applications can run on iPhone4 or later, and all iPads with cameras (iPad2 or later), while doing both vision-based AR and geo-located AR.

All content is represented in the same 3D scene graph, unifying the content for the programmer: content following a tracked image target is attached to the camera, which is in turn located in the world via GPS and the device's built-in orientation sensors. Argon2 supported both WebGL and HTML5/CSS3 3D content. Audio is supported via Web Audio. WebGL and CSS3 content are merged in a three.js [8] scene graph. In the end, a web programmer is given a simple three.js scene graph view of the AR world, with nodes for tracked computer vision targets and geo-located locations in the world. All 2D content, interaction and web programming is done just as any mobile web app.

3.1 Applications Built with Argon2

As in the above section, here we present some applications created with Argon2 and discuss what we have learned from them about using web-technologies to create community-based AR experiences. At the end, we will outline the requirements we have for the next version of Argon (Argon3, currently under development).

A number of Argon1 applications were rewritten for Argon2. *The Lights of St. Etienne*, discussed above, was quite easy to port because all of the interface elements in HTML and Javascript could be directly reused, and the resulting interface was almost indistinguishable from the original Argon1 version. Debugging the port was made even easier because the Argon2 app (now just a single WebView) can be remotely debugged via a desktop computer, just as if it were a web page running in the Safari browser.

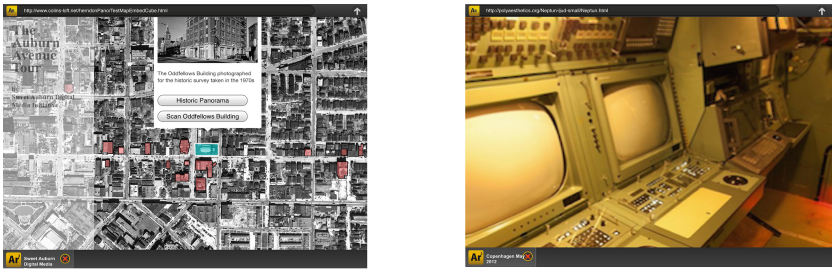
The second version of the *Campus Tour* ported the experience to Argon2 and improved the general user interface with an enhanced main menu allowing the user to view the route a tour takes, and see the stops along the tour visualized on a customized map within the channel instead of the native map which Argon1 provided (which showed all points of interest Argon1 knew about). From this map the user can select placemarks to "jump" to a geospot, switching into the AR experience at that location.

The biggest improvement to the *Campus Tour* was to support authoring of personalized AR experiences; students and faculty could create their own tours of the campus, showcasing Georgia Tech from their individual point of view via an associated web based authoring tool (the campus tour editor). This includes adding new points of interest or panoramic images to existing tours, or creating new tours, panoramic images, as well as points of interest with audio, video, or text content. Unfortunately, the tools attracted relatively few authors, highlighting to us the need for creating much

more sophisticated authoring tools. The limited acceptance also highlighted the limitations inherent in doing AR on mobile devices. The poor quality of outdoor localization, and the limited opportunities for outdoor computer vision, made it hard for authors to create the kinds of experiences they envisioned.

Auburn Avenue. Using Argon2, we began to develop a tour of Auburn Avenue, the historical center of African-American culture in Atlanta in the twentieth century. The interface is similar to the Metz Cathedral example, although in this case the user is outside walking down an avenue. A scrolling timeline is used to indicate buildings of interest on an aerial map: these buildings are highlighted and clickable (see Fig. 4(a)). Clicking will trigger various kinds of information and interaction, including audio and historical panoramas. Vuforia-based image tracking is used to highlight details and presented information about the facade of one historical building.

The Auburn Avenue prototype is significantly more elaborate than the historic tours developed in Argon1, owing largely to the ease with which content can be authored and tested using desktop web browsers, and then integrated into the actual tour.



(a) (b)

Fig. 4. (a) The Auburn Avenue map, and (b) a panorama in Neptun

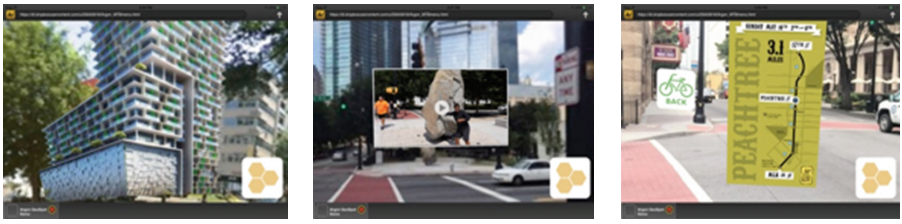
Neptun. *Neptun* is based on a historic submarine that belonged to the Swedish navy, which is on exhibit in the Marine Museum in Karlskrona, Sweden [9]. The application was designed as a special companion piece to the opening of the exhibit. It is possible to go into the submarine itself; however, the confined area inside makes the visit difficult or impossible for visitors with claustrophobia or disabilities. *Neptun* offered the opportunity for a virtual visit by offering panoramic views of the upper and lower deck of the boat, along with dramatic audio narrative. On the opening day of the exhibition, iPads were available for those who wished to try this experience. By implementing the panoramas directly in three.js, the experience could be taken in a conventional web browser; the Argon2 version allowed the user to examine the whole 360 degrees of the panorama by rotating the iPad screen around herself.

This prototype showed the value of being able to move content easily between conventional desktop browsers and the Argon2 application, allowing the content to be consumed in many ways.

Midtown Buzz. Midtown Buzz (MB) (midtownbuzz.org) is a partnership between Georgia Tech and the Atlanta Midtown Alliance (MA) (www.midtownatl.com), focused on engaging urban communities through mobile innovation [10]. Since 2013, we have been collaborating with the Midtown Atlanta community with the goal of transforming the area into an innovation district, while creating a living laboratory for civic computing research. During the two years of this project we have engaged in a participatory design process with stakeholders ranging from MA staff, local start-up companies, student developer teams, and community thought leaders.

Our work culminated in the creation of some technology artifacts, including Midtown Buzz Mobile (MBM) as a gateway to a variety of applications. MBM is implemented using Argon2 and provides connections to other Georgia Tech applications that are associated with MB (e.g. Cycle Atlanta and One Bus Away) as well as to various external web resources.

MBM also includes a mixed reality storytelling experience that focuses on *personal expressive content*, aimed at creating rich community resources that are not motivated by efficiency and productivity, and are instead based on a philosophy inspired by urban computing projects [11–13]. The web-based model of Argon2 made it easy to embed mixed reality experiences inside of the application web content.



(a) (b) (c)

Fig. 5. The Midtown Buzz Mobile Mixed Reality Storytelling Experience. (a) An architectural rendering of a future building, (b) a video of a community author interacting with local sculpture, and (c) a map of a cycling event on Peachtree Street in Atlanta.

In MBM, key members of the community tell their stories of the Midtown neighborhood. Our prototype allows users to explore what areas will look like once new construction is complete, to get an insider's view of cycling around Midtown, all while getting a taste of the history of key locations (see Fig. 5). The site incorporates web elements in a mixed reality presentation where users can view videos, articles, photos, and other media presented over a 360-degree panorama of Midtown locations. If the user is at the locations featured, she can turn off the provided panorama image and have a true augmented reality experience overlaid on the live video feed from their device.

3.2 Discussion of Argon2

Argon2 targets experienced web developers by encapsulating AR functionality in the argon.js library, simplifying integration of third-party web services and libraries. Many of our developers used the tools they were familiar with, such as jQuery. The popular three.js rendering library was bundled with argon.js inside of the Argon2 application. (We chose three.js because it is a relatively popular library for 3d web content, with many tutorials, examples, and resources).

Tight-coupling with three.js. Our original intention for bundling three.js with argon.js was to simplify development and bypass slow download times on mobile devices, as well as avoid library incompatibilities as argon.js and three.js evolved. In practice, however, the argon.js API became tightly coupled to specific versions of three.js, which changes rapidly, meaning that developers had to be careful in making sure their applications were based on the version of three.js we included in argon.js (and did not necessarily correspond to the latest three.js documentation and examples). Due to the problems that arose because of this tight coupling and bundling, in Argon3 we plan to *decouple argon.js from any particular rendering library*, expose bindings to specific rendering libraries (such as three.js) as *separate rendering plug-ins for argon.js*, and *no longer bundle argon.js* with the Argon3 browser. The argon.js toolkit should enable web developers to leverage Argon's capabilities, and to also leverage the capabilities of other platforms and rendering libraries.

Debugging. Argon2 moved most of the core logic from native code to javascript, allowing channels running in Argon2 to be debugged via desktop Safari's remote debugging tools. Furthermore, by modularizing a channel's code such that components which depended on argon.js are only executed in the Argon browser, most of the application code could be developed and tested on the desktop browser leveraging its full debugging support. This strategy not only facilitated debugging, but also contributed to a multi-tiered design strategy: experiences could be designed to work on various devices (laptop as well as mobile) with various features, making it possible to conceive of and implement "transmedia" apps. Each of the applications discussed above (Auburn, Neptun, Midtown Buzz) leverage this strategy.

Multi-device Support. Although it was a significant goal for Argon2, we never implemented support for alternative hardware form-factors. Argon2 and argon.js were too tightly coupled to the phone/tablet form factor used by the application, making it difficult to support other platforms. In Argon3, we will *create AR abstractions and an AR framework* that allows application developers to create applications that can adapt to various hardware and software configurations without requiring the developer to create multiple versions of their application. Such a framework will also need to allow AR to be used in different ways within an application, instead of requiring applications to be "AR-first", as Argon2 does.

3.3 Requirements for Argon3

Our high level requirements for Argon3, derived from our experiences above, include:

- Decouple argon.js from specific rendering libraries
- Enhanced debugging and authoring support for AR
- Support AR/non-AR as modes of interaction
- Develop semantically meaningful abstractions for AR applications
- Multi-device support (desktop, HMD, etc.) (not implemented in Argon2)

4 Discussion and Conclusions

We are currently designing and implementing the third version of Argon, based on the lessons learned working with designers and developers who used Argon1 and Argon2 to create a wide variety of applications. We have been particularly inspired by the way Argon has allowed designers to integrate AR into community-based applications, moving both formal and informal content out into the world around them. A key insight from working with these communities is that while AR is exciting at first, it rarely ends up being the focus of the resulting applications; our job moving forward, then, is to make it as easy as possible to integrate AR content into web-based applications, and to give programmers and designers the ability to use AR in their applications in whatever way they see fit.

Acknowledgements. We would like to thank AT&T, Alcatel-Lucent and Qualcomm for their support of the Argon project, and all the students, researchers and sponsors who used (and supported those using) Argon. We have benefited from ongoing support from the GVU Center at Georgia Tech, and from the support of an NSF Graduate Research Fellowship for the first author.

References

1. MacIntyre, B., Hill, A., Rouzati, H., Gandy, M., Davidson, B.: The Argon AR Web Browser and standards-based AR application environment. In: 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 65–74. IEEE (2011)
2. Kooper, R., MacIntyre, B.: Browsing the real-world wide web: maintaining awareness of virtual information in an AR information space. Int. J. Hum. Comput. Interact. **16**(3), 425–446 (2003)
3. Spohrer, J.C.: Information in places. IBM Syst. J. **38**(4), 602–628 (1999)
4. Dow, S., Lee, J., Oezbek, C., MacIntyre, B., Bolter, J.D., Gandy, M: Exploring spatial narratives and mixed reality experiences in Oakland Cemetery. In: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2005), pp. 51–60. ACM, New York (2005)
5. MacIntyre, B., Bolter, J.D., Moreno, E., Hannigan, B.: Augmented reality as a new media experience. In: IEEE/ACM International Symposium on Augmented Reality (ISAR), pp. 197–206. IEEE (2001)
6. KARML Reference. https://research.cc.gatech.edu/kharma/content/karml-reference

7. KML version 2.2. http://www.opengeospatial.org/standards/kml
8. three.js, a Javascript 3D library. https://github.com/mrdoob/three.js/
9. Marine Museum in Karlskrona, Sweden. http://www.marinmuseum.se/en/
10. Gandy, M., Baird, L.D., Levy, L.M., Lambeth, A., Mynatt, E., Clark, R., Sanders, M.: Midtown buzz: bridging the gap between concepts and impact in a civic computing initiative. In: Proceedings of Human Computer Interaction International, Los Angeles, CA, 2–7 August 2015
11. Korn, M., Back, J.: Talking it further: from feelings and memories to civic discussions in and about places. In: Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design, New York, NY, USA, pp. 189–198 (2012)
12. Foth, M.: From Social Butterfly to Engaged Citizen: Urban Informatics, Social Media, Ubiquitous Computing, and Mobile Technology to Support Citizen Engagement. MIT Press, Cambridge (2011)
13. DiSalvo, C., Maki, J., Martin, N.: Mapmover: a case study of design-oriented research into collective expression and constructed publics. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, NY, USA, pp. 1249–1252 (2007)