Authoring Tools for Augmented Reality: An Analysis and Classification of Content Design Tools

Rafael Alves Roberto[1(✉)], João Paulo Lima[1,2], Roberta Cabral Mota[3], and Veronica Teichrieb[1]

[1] Voxar Labs, CIn, Universidade Federal de Pernambuco, Recife, Brazil
{rar3,vt}@cin.ufpe.br

[2] DEINFO, Universidade Federal Rural de Pernambuco, Recife, Brazil
jpsml@deinfo.ufrpe.br

[3] University of Calgary, Calgary, Canada
roberta.cabralmota@ucalgary.ca

Abstract. Augmented Reality Authoring Tools are important instruments that can help a widespread use of Augmented Reality. They can be classified as programming or content design tools in which the latter completely removes the necessity of programming skills to develop an Augmented Reality solution. Several solutions have been developed in the past years, however there are few works aiming to identify patterns and general models for such tools. This work aims to perform a trend analysis on content design tools in order to identify their functionalities regarding Augmented Reality, authoring paradigms, deployment strategies and general dataflow models. This work is aimed to assist developers willing to create authoring tools, therefore, it focuses on the last three aspects. Thus, 24 tools were analyzed and through this evaluation it were identified two authoring paradigms and two deployment strategies. Moreover, from their combination it was possible to elaborate four generic dataflow models in which every tool could be fit into.

Keywords: Augmented reality · Authoring tools · Content design tools

1 Introduction

Recently, Augmented Reality (AR) technology started to be widely used in various application domains, such as advertising, medicine, education, and others. However, the time and technical expertise needed to create AR applications is one of the reasons that has prevented widespread use. In this sense, authoring tools have become a largely used solution to boost mainstream use of AR since they facilitate the development of AR experiences [25].

AR authoring tools can be broadly organized into two different approaches: AR authoring for programmers and non-programmers [11]. In the former case, tools are typically code libraries that require programming knowledge to author the application. In this work, these approaches are called programming tools. In the latter case, abstraction is added and low level programming capability is removed or hidden. Thus, tools for non-programmers are content driven and commonly include graphical user interfaces for building applications without writing any lines of code. Here, it is addressed as content design tools.

These two generic categories can be further organized into low-level and high-level, as seen in Fig. 1. Low-level programming tools require low-level coding while high-level ones use high-level libraries. Furthermore, low-level content design tools demand scripting skills and high-level tools use visual authoring techniques. All of these authoring approaches are built upon each other. Abstraction is gradually added and low-level functionalities and concepts are removed or hidden. Also, different abstraction levels address different target audiences with different technical expertise.

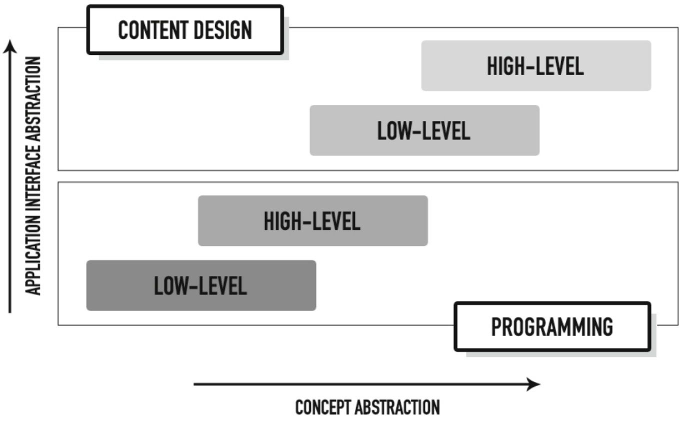

Fig. 1. Schematic view of AR authoring tools [11].

Among the approaches of AR authoring tools, it is important to note the relevance of the content design tools. They are particularly important because they leverage the widespread adoption of AR, since they highly simplify the authoring process and allow the development of applications and content by ordinary users, which do not need to have programming knowledge. Therefore, content design tools in AR have greater relevance when we take into account the potential amount of users that can use AR solutions in the future.

From the first solutions [12] to the most recent ones [27], it is possible to see that several content design tools have been developed. However, there are a few works aiming to provide a classification or to identify tendencies and patterns for such tools. To the best of the authors knowledge, there is only one work that proposes a taxonomy for AR authoring tools. In turn, this classification is according to the application interface and concept abstraction [11], in which the most abstract application interfaces are named content design frameworks and the least abstract are called programming frameworks.

Due to the relevance of content design tools, this paper aims at conducting a trend analysis in order to understand the current tendencies of such tools. This investigation attempts to identify the current tendencies regarding the authoring paradigms and deployment strategies of AR experiences that have been used in both commercial and academic realms. Furthermore, these strategies were combined to elaborate generic dataflow models in which all of the content design tools could fit into. Finally, based on these findings, it was introduced a taxonomy representing the different authoring and deployment trends, as well as each of the general models. This classification may guide researchers and companies to develop solutions aiming at their needs.

This work is organized as follows: Sect. 2 describes the methodology used to perform the trend analysis. Section 3 presents the results obtained from the analysis while Sect. 4 discusses them. Finally, the conclusions of this work are drawn in Sect. 5.

2 Methodology

Three steps were taken to explore the trends regarding AR authoring tools. The first one was the selection of content design tools available in the marketplace and the literature. Then, as a second step, the analysis relied on observing the dataflow of development and access to the AR content of each one of the selected tools. Finally, the third step consisted in discovering the authoring paradigms and deployment strategies used in the content design tools. Their combination was used to elaborate general dataflow models.

2.1 Selection

Initially, the keywords and expressions *("authoring tool" AND "augmented reality")* were searched in IEEE Xplore Digital Library and ACM Digital Library in order to find relevant papers concerning authoring tools in AR. That allowed for an investigation over important publications from 2001 to 2015. During this examination, only authoring tools classified as content design tools were selected for analysis.

2.2 Analysis

Following this, a deeper analysis was performed of each one of the selected authoring tools in order to understand the dataflow for development and access to the AR content. On a high level, this dataflow describes the end-to-end scenario that outlines the authoring and deployment of AR experiences, from the creation of AR semantic through authoring tools to its visualization by end-users.

2.3 Categorization

The deeper analysis performed on each of the selected content design tools made possible the observation of trends regarding authoring and distribution strategies of AR experiences that have been used. Furthermore, this observation also tried to understand (a) how the different authoring paradigms may support AR content development, and (b) how the deployment strategies seek to reach a larger number of end-users. Finally, the identification of AR authoring and deployment trends allowed the translation of the project-specific dataflow, observed in the selected content design tools, into the creation of general dataflow models. In this sense, a minimum number of combinations of trends was performed in order to elaborate generic models, in which all of the content design tools could fit into.

3 Results

The search on the scientific libraries returned 147 papers and 14 works about content design tools were selected for analysis. Moreover, 10 commercial tools that are well consolidated or relevant in the market were chosen. Thus, taking into account both academic and commercial realms, there were 24 content design tools. Thereafter, a dataflow analysis was performed in each of the selected tools.

3.1 AR Authoring Paradigms

Once individual analyses were performed in each of the previous selected content design tools, it was observed that two authoring paradigms have been used to create AR solutions: stand-alone and AR plug-in approaches.

Stand-Alone. AR authoring tools that use the stand-alone paradigm are software with all the necessary components for the development of complete AR experiences, as can be seen in Fig. 2(a). In turn, these components may include a graphical user interface, a series of importers, sensor interfaces, tracking and rendering engines, among others. In this sense, each stand-alone content design tool is a new software that allows designers to create their custom AR projects with more or less ease.

As an example, the Layar Creator [17] provides a complete set of features along the entire creation workflow, such as graphical user interface including drag and drop to ease the scenario creation.

AR Plug-in. Similar to conventional digital plug-ins, AR plug-ins are third-party software components installed on host applications in order to enable additional features, as illustrated in Fig. 2(b). In this sense, these authoring tools provide AR capabilities to software that natively does not support it, such as tracking techniques, access to physical sensors, three-dimensional rendering engine, among others.

It is relevant to note that, from the practical point of view, an AR plug-in instance will appear in the target software as a set of GUI elements, such as one or more menu items and toolbar buttons. Therefore, the whole AR authoring process occurs within the hosting environment, as the designer completely configures the desired AR experience by means of those elements along with the ones already provided by the target software.

As an example, the DART [7] system is built as a collection of extensions to the Adobe Director [2], a widely used environment for multimedia content creation, to support the development of a variety of AR applications.

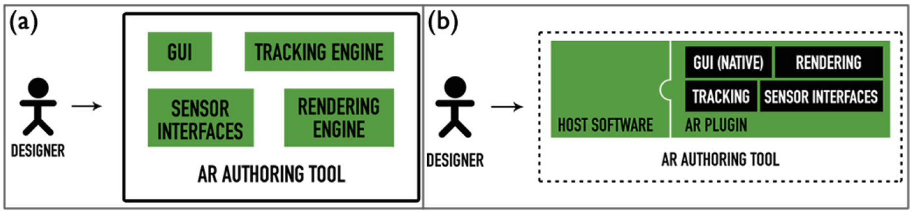

Fig. 2. (a) Stand-alone AR authoring tools enable building entire AR experiences. In order to provide AR capabilities, these tools integrate components such as sensor interfaces, tracking and rendering engines; (b) AR plug-ins provide AR functionalities for non-AR authoring environments. The designer interacts directly with the hosting software in order to create AR experiences.

3.2 AR Deployment Strategies

It was noticed that two deployment strategies have been applied to make these AR experiences available for end-users: platform-specific (PS) and platform independent (PI) methods.

Platform-Specific. In the PS approach, AR projects built using authoring tools are exported to archive files to be independently distributed. Some common archive file formats include .exe in Windows, .dmg or .app in Mac OS, .apk in Android, and .ipa for iOS operating systems. Note that these archive files are software packages used to distribute and install native application software. A native app, in turn, is considered a stand-alone program itself since it is a self-contained program that does not require any auxiliary software to be executed, as can be observed in Fig. 3(a). Native apps are usually available through application distribution platforms, such as App Store, Google Play, and Windows Phone Store. However, they must be downloaded from the platform to the end-user devices, such as iPhone, Android, Windows phones, or even laptops or desktop computers.

As an example, the Wikitude Studio [28] supports deployment options to mobile applications for iOS/Android platforms, and to executable programs for Windows/Mac OS computers.

Platform-Independent. The PI approach delivers the AR solutions as data files read and executed on a software platform (SP) running on the end-user device. Also, it is worth pointing out that, after the authoring process, the generated content requires a platform on which it must be executed. Therefore, the content cannot be considered a stand-alone program. Rather, it comprehends data files (commonly structured in XML-based formats) that are interpreted by the SP, as illustrated in Fig. 3(b).

As an example, k-MART [6] allows designers to export AR solutions as X3D-based data files. In turn, these files are later executed on a separate content browser.

Furthermore, since the content does not need to be installed in the device, a major advantage is the possibility of implementing a cloud-based deployment service. This increasingly popular variant approach uses a server infrastructure as a backbone. The remote server is responsible for content storage and retrieval as requested by the clients. The clients, in turn, are responsible for presenting the retrieved content on end-user devices. Also, a client comprehends a cloud-based SP that reaches into the cloud for contents on demand. In turn, all data files remotely accessed are here referenced as cloud-based applications.

As an example, AR companies like Layar and Wikitude developed Layar App and Wikitude World AR browsers, respectively. To the end-user, an AR browser looks very similar to a typical native app: it is downloaded from an app store, stored on the mobile device, and launched just like a native app. However, the most prominent advantage of AR browsers is that end-users need only one app for multiple contents. Once installed, they pull new cloud-based apps on demand.

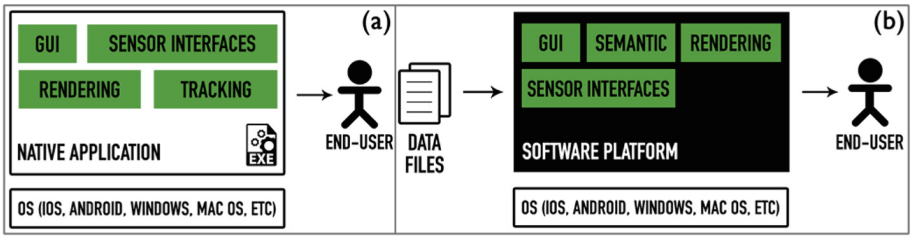

Fig. 3. (a) A native app includes all required elements to execute AR experiences, thus can be considered a stand-alone software itself. Also, the term native comprises applications compiled at runtime, such as an Android app, or precompiled executable programs; (b) Data files are interpreted by a native shell, which provides the required infrastructure to present AR experiences.

3.3 General Models

Given the authoring and deployment trends explored in the previous subsections, it was possible to elaborate four general dataflow models that represent the content design tools' dataflow analysed in this work.

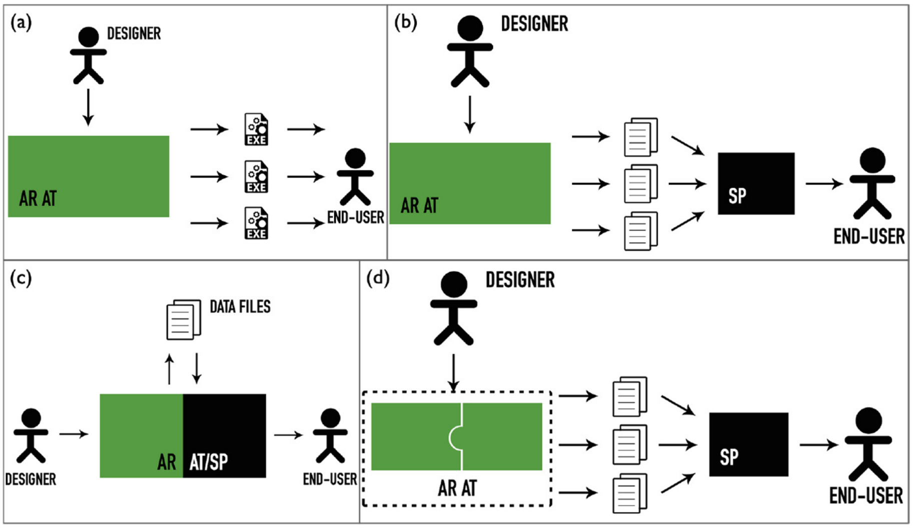

Fig. 4. (a) Model 1 combines a stand-alone authoring with a PS distribution. Therefore, each generated native application is individually installed and accessed by end-users; (b) Model 2 unites a stand-alone authoring paradigm with a PI distribution; (c) Model 3 combines a stand-alone authoring with a PI distribution. Yet, both designers and end-users utilize the same ambient to create and visualize AR solutions; (d) Model 4 merges an AR plug-in authoring with a PI distribution.

Model 1: Stand-Alone PS Model. As can be observed in Fig. 4(a), this dataflow model embodies a stand-alone authoring approach combined with a native distribution strategy. In this sense, the designer first creates AR experiences through stand-alone content design tools. Then he exports the project as PS files, which are used to deliver stand-alone, native applications for Android, iOS, Windows or other operating system.

Model 2: Stand-Alone PI Model. Similarly to the previous model, the designer first builds AR experiences using stand-alone content design tools. However, this model applies a PI strategy for reaching interoperability and maintainability. In this sense, the designer exports the authored AR solutions as data files that run on a separate SP. Note that, a content design tool can generate one or more data files which can be interpreted in a single SP and in different periods of time, as seen in Fig. 4(b). Yet, since each stand-alone content design tool is a brand new software, the data files created by distinct tools generally differ in their structures, logics, and formats.

Model 3: All-in-One Model. As illustrated in Fig. 4(c), in this model, both designers and end-users utilize the same environment to build and access AR solutions. In the sense, the designer creates and saves AR solutions as data files.

Table 1. Classification of each commercial (without year) and academic tool according to the general dataflow models.

Content Design Tools	**Year**	**M1**	**M2**	**M3**	**M4**
Metaio Creator [22]	-	🟩	🟩		
Metaio AR Creator Plugin [21]	-				🟩
Wikitude Studio [28]	-	🟩	🟩		
Layar Creator [17]	-		🟩		
Build AR [14]	-		🟩		
AR-media Plugin [15]	-				🟩
Playme [23]	-		🟩		
CraftAR [4]	-	🟩			
Aurasma Studio [13]	-		🟩		
DAQRI 4D Studio [8]	-		🟩		
Powerspace [12]	2002		🟩		
Authoring Wizard [29]	2003		🟩		
AMIRE [1]	2004		🟩		
DART [19]	2004				🟩
CDT1 [18]	2004			🟩	
ComposAR [26]	2008			🟩	
VREditor [20]	2009		🟩		
ARBookCreator [9]	2009			🟩	
AR Scratch [24]	2009			🟩	
k-MART [6]	2010		🟩		
CDT2 [16]	2012		🟩		
AVATAR [10]	2012				🟩
CDT3 [3]	2013		🟩		
CDT4 [27]	2014			🟩	

Eventually, these files are read and executed within the same environment in order to present the AR experience to end-users. Hence, similarly to the previous model, the all-in-one comprehends a stand-alone authoring approach combined with a PI distribution. However, the major difference resides in the fact that production and delivery services are merged within a single system.

Model 4: AR Plug-in PI Model. In this dataflow model, the designer first builds AR projects using hosting software integrated with AR plug-ins. Then, these projects are saved as data files that are later retrieved and executed on a separate application. In other words, this model includes a plug-in approach combined with a PI deployment strategy, as can be observed in Fig. 4(d).

All the content design tools that were selected and analyzed in this work are listed in Table 1. The table divides the commercial and academic tools and indicates to which of the four general dataflow models each tool belongs. It is important to keep in mind that it is not mandatory for a tool to be categorized into only one general model since a content design tool can provide different distribution approaches and, consequently, different dataflow models.

4 Discussion

The major findings permitted a discussion in regards to (a) the benefits and shortcomings of each of the AR authoring paradigms and deployment strategies and (b) the results obtained from the classification of the analyzed content design tools according to the four generic dataflow models. Each one of these discussions are approached in the following subsections.

4.1 Stand-Alone vs. AR Plug-in

Reusability. Stand-alone authoring tools generally offer a smaller set of features when they are compared to full-fledged hosting software. In the plug-in approach, the designer may create the AR experience by using not only the extra functionalities provided by the plug-in, but also the mature existing features in the target software. On the other hand, it would require strong effort and time to implement these features in a stand-alone authoring tool.

Domain specificity. Another benefit of the AR plug-in over the stand-alone approach is that each created plug-in can be specialized for a specific application domain. By using the plug-in approach, one can propose a set of functionalities tailored to one specific application domain. Thus, only the required features are implemented and not every possible one that usually a stand-alone tool must provide.

Learning curve. Given a situation in which stand-alone tools are focused on specific tasks or application areas, the learning curve and the time employed to learn how to use each new tool, with different interfaces, is longer if compared to integrated tools. Since the plug-ins are usually implemented on the same environment, each focusing on a specific task or application domain, the designer would not need to interact with a new interface.

However, the time spent to learn how to create an AR application using a plug-in can be bigger when compared with a stand-alone solution in case the user does not have experience with the host tool. It is due to the fact that they commonly have a complex interface. Thus, the learning curve for simple and focused tools is usually shorter.

Integration restraints. Bringing AR capabilities inside non-AR authoring tools is neither straightforward nor trivial. First, it must be considered the availability of an SDK for the target software. It is extremely hard to manage the integration when dealing with a closed system. Another factor is the GUI. When designing an integrated AR plug-in for a host software, it is mandatory to create GUI elements with the same look-and-feel as the target software. Moreover, the authoring metaphor must be used in a coherent way with it. In this sense, obstacles may arise during the development of functionalities that follow the traditional authoring style and, at the same time, enable AR content creation with ease.

4.2 Platform-Specific vs. Platform-Independent

Portability. Since native apps are built using the device's native programming language, they only run on their designated platform. This means that the same app cannot be re-used between platforms. Thus, deploying a native app to Android, iOS and Windows Phone would require creating three different applications to run on each platform. Contrastingly, one major promise about platform independence is that designers only have to write the application once and then it will be able to run anywhere, without having to be recreated by the designer for each separate platform.
Maintenance cost. Maintaining native applications is also expensive for the designer. As a native app is built for a particular device and its operating system, whenever new OS versions are rolled out, native apps may require considerable updates to work on these newer versions. Moreover, data files run independent on a PS shell that operates as an abstraction layer that encapsulates the underlying hardware and software updates. Hence, the designer does not have to worry about updating and resubmitting apps.
Offline functionality and speed. An advantage for native apps is the off-line functionality and speed. Since the application remains installed on the device from the original download, depending on the nature of the app, no internet connection may be required. Another area where native apps have a clear advantage is speed. These apps, by definition, run at native speed. PI apps run on top of additional layers, which consume computing resources and can therefore decrease the execution speed.

4.3 General Models

As can be seen in Table 1 the stand-alone PI model is the most used approach in both commercial and academic realms. It can also be noticed that the stand-alone PS model is the less used approach. In turn, these results might indicate that there is a strong trend towards PI strategies, which provide two key determinants for delivering widely deployable AR experiences.

First, it permits cross-platform usage of the created AR experiences and the more platforms are covered, the more people are reached. In this sense, leading AR companies have commercial content design tools that suit into the stand-alone PI model. In these scenarios, AR solutions are executed on their respective AR browsers. In turn, these browsers are cross-platform applications that run across different operating systems.

Second, the PI strategy also allows content aggregation for leveraging distribution and discovering of AR solutions. In this sense, it can provide an unified AR platform to access multiple content, thus avoiding the cumbersome task of downloading and installing each one of the AR solutions.

Table 1 shows that seven out of the eight generic dataflow models target data files for specifying the AR solutions. Therefore, based on these results, it is possible to observe that there is a consensus in academia and industry for adopting descriptive data formats, which includes JSON and, most often, XML-based formats such as ARML, KARML, and XML.

5 Conclusion

AR authoring tools can provide several levels of abstraction, thus targeting audiences within a range of different technical expertise. Particularly, those categorized as content design tools allow non-technologists to explore the AR creation medium and, therefore, these tools are an essential component for helping AR to gain popularity in different application domains. Due to their relevance, several content design tools have been developed recently. However, no work was found that presents an analysis and classification of those tools.

In this sense, this work analyzed 24 commercial and academic content design tools in order to identify tendencies of such tools. The investigation revealed that there are two authoring paradigms and two distribution strategies that have been widely used for such tools. Furthermore, these authoring and deployment trends were combined to elaborate four generic dataflow models.

Furthermore, this paper discussed the authors' findings on the authoring and deployments tendencies by comparing one trend against its alternatives in order to discuss their advantages and limitations. Thereafter, it discussed the results obtained from the classification of the content design tools according to the four general dataflow models.

The authors believe that the proposed taxonomy along with the discussion regarding the major findings can help users to find the best tools for them or guide researchers and companies to develop solutions aiming their needs.

References

1. Abawi, D.F., Dörner, R., Grimm, P.: A component-based authoring environment for creating multimedia-rich mixed reality. In: EGMM 2004 (2004)
2. Adobe Systems Incorporated: Adobe Director, January 2015. http://goo.gl/QnkQni
3. Barbadillo, J., Sánchez, J.R.: A web3d authoring tool for augmented reality mobile applications. In: Web3D 2013 (2013)
4. Catchroom: CraftAR: Augmented Reality and Image Recognition toolbox, January 2015. http://goo.gl/ThRVv7
5. Cho, H., Gray, J., Sun, Y.: Quality-aware academic research tool development. In: APSEC 2012, vol. 2, pp. 66–72, December 2012
6. Choi, J., Kim, Y., Lee, M., Kim, G., Nam, Y., Kwon, Y.: k-MART: Authoring tool for mixed reality contents. In: ISMAR 2010, pp. 219–220, October 2010
7. Coleman, M.G.: Creating augmented reality authoring tools informed by designer workflow and goals. Ph.D. thesis, Georgia Institute of Technology (2012)
8. DAQRI: 4D Studio DAQRI, January 2015. http://goo.gl/y8kLvr
9. Do, T.V., Lee, J.W.: Creating 3d e-books with ARBookCreator. In: ACE 2009 (2009)
10. Fei, G., Li, X., Fei, R.: AVATAR: Autonomous visual authoring of tangible augmented reality. In: VRCAI 2012 (2012)
11. Hampshire, A., Seichter, H., Grasset, R., Billinghurst, M.: Augmented reality authoring: Generic context from programmer to designer. In: OZCHI 2006 (2006)
12. Haringer, M., Regenbrecht, H.: A pragmatic approach to augmented reality authoring. In: ISMAR 2002, pp. 237–245 (2002)

13. Hewlett-Packard Development Company: Aurasma Studio. https://goo.gl/Ix8vgr
14. HIT Lab NZ: BuildAR Pro, January 2015. http://goo.gl/1LtXvt
15. Inglobe Technologies S.r.l.: AR-media - AR-media Products, January 2015. http://goo.gl/TbLeM8
16. Langlotz, T., Mooslechner, S., Zollmann, S., Degendorfer, C., Reitmayr, G., Schmalstieg, D.: Sketching up the world: In situ authoring for mobile augmented reality. Personal Ubiquitous Comput. **16**(6), 623–630 (2012). http://dx.doi.org/10.1007/s00779-011-0430-0
17. Layar: Layar creator, January 2015. http://goo.gl/rQ0PlP
18. Lee, G., Nelles, C., Billinghurst, M., Kim, J.: Immersive authoring of tangible augmented reality applications. In: ISMAR 2004, pp. 172–181 (2004)
19. MacIntyre, B., Gandy, M., Dow, S., Bolter, J.D.: DART: a toolkit for rapid design exploration of augmented reality experiences. In: UIST 2004 (2004)
20. Mavrogeorgi, N., Koutsoutos, S., Yannopoulos, A., Varvarigou, T., Kambourakis, G.: Cultural heritage experience with virtual reality according to user preferences. In: CENTRIC 2009, pp. 13–18, September 2009
21. Metaio GmbH: metaio — AR Creator Plugin — Products, January 2015. http://goo.gl/Z5E9ym
22. Metaio GmbH: metaio — Creator Overview, January 2015. http://goo.gl/6xwgyF
23. Playme AR: Playme AR Creator Features — Playme AR - Simple Augmented Reality Software, January 2015. http://goo.gl/nGAf9t
24. Radu, I., MacIntyre, B.: Augmented-reality scratch: A children's authoring environment for augmented-reality experiences. In: IDC 2009 (2009)
25. Ramireza, H., Mendivila, E.G., Floresa, P.R., Gonzalez, M.C.: Authoring software for augmented reality applications for the use of maintenance and training process. In: International Conference on Virtual and Augmented Reality in Education. pp. 189–193, October 2013
26. Seichter, H., Looser, J., Billinghurst, M.: ComposAR: An intuitive tool for authoring ar applications. In: ISMAR 2008, pp. 177–178, September 2008
27. Shim, J., Kong, M., Yang, Y., Seo, J., Han, T.D.: Interactive features based augmented reality authoring tool. In: 2014 IEEE International Conference on Consumer Electronics (ICCE), pp. 47–50, January 2014
28. Wikitude GmbH: Wikitude Studio - the world's easiest augmented reality creation tool, January 2015. http://goo.gl/Lr0uOK
29. Zauner, J., Haller, M., Brandl, A., Hartman, W.: Authoring of a mixed reality assembly instructor for hierarchical structures. In: ISMAR 2003, pp. 237–246, October 2003