The PortARble Museum: Developing Augmented Reality For The Web Using AR.Js

Fred Leighton, University of Wisconsin-Whitewater, USA

Abstract

Using AR.js, a JavaScript library for the development of Augmented Reality (AR) content for mobile Web browsers on smart phones with Android or Apple iOS 11 operating systems, AR Web projects inspired by Duchamp's Portable Museum can be realized today with a set of simple and straightforward development tools. In this session, a demonstration of content developed with AR.js will be shown along with a step-by-step process for creating your own mobile AR Web pages with content triggered by visual markers. Participants will be able to work with examples during the session and should have HTML and CSS development experience. A laptop computer with Web connectivity and HTML editing software, along with a built-in camera, is needed A smart phone is not necessary, but will add to the understanding of the subject. Smart phones must have either recent Android or Apple iOS 11 operating systems. Attendees can also watch and learn without coding. Resources provided will allow for following up on the session content later on.

Keywords: Web development, augmented reality, mobile, handheld, smart phone

Introduction

referent, these “items” thus resist fixing the boundaries, properties, and functions of the works on which they are based.” (Filipovic, 2009)

Duchamp stated

> *Instead of painting something new, my aim was to reproduce the paintings and objects I liked and collect them in as small a space as possible. I did not know how to go about it. I first thought of a book, but I did not like the idea. Then it occurred to me that it could be a box in which all my works would be collected and mounted like in a small museum, a portable museum, so to speak. (Filipovic, 2009)*

Using AR.js, a JavaScript library for the development of augmented reality content for mobile Web browsers on smart phones with Android or Apple iOS 11 operating systems, AR Web projects inspired by the *Portable Museum* can be realized today with a set of simple and straightforward development tools. AR markers, on paper or displayed on a screen, when viewed through the smart phone camera, can trigger associated 3D model content, both still and animated. Markers can be organized in any manner for the associated mobile AR content.

AR.js allows for efficient displays of 3D content in Web pages tied spatially to AR markers. Attendees who want to learn AR.js need to have basic HTML and CSS coding experience and have some understanding of JavaScript. AR.js is a published library, with examples and documentation online. The A-Frame VR framework is available for making models; 3D models in the glTF 2.0 format can also be used for developers who would rather use existing content or convert content to this format from other 3D models types.

In this session, a step-by-step development process will be demonstrated for creating and/or accessing 3D model content, and authoring Web pages utilizing the AR.js library. The testing of Web pages containing AR content with smart phones and visual markers will also be explained.

development of html Web pages to include AR (augmented reality) content (Etienne, J., 2017c). AR.js requires WebGL (Khronos Group, 2018c), a 3D graphics API (Application Programming Interface) for the HTML5 Canvas element, and WebRTC, a set of browser APIs and protocols that allow for real-time communications of audio, video, and data in Web browsers and native apps (WebRTC, 2017). WebRTC requires encryption, and a Web server configured for https (secure hypertext transfer protocol). WebGL and WebRTC are supported in current Android mobile devices and Apple iPhones with iOS 11. AR.js can be used in conjunction with three.js (Three.js, 2017), a JavaScript library for the development of 3D Web content; and, as demonstrated in this session, AR.js can be used with the A-Frame framework (Etienne, J., 2017a), a JavaScript library for creating Web VR (virtual reality) content (A-Frame, 2015-2017b). Performance on mobile devices is best in Chrome for Android, and Safari for iOS, but other browsers can also be used, such as Firefox. A-Frame is "a powerful entity-component framework that provides a declarative, extensible, and composable structure to three.js" (A-Frame, 2015-2017a). Leveraging features in ARToolKit (ARToolkit5, 2016) and A-Frame, AR.js makes the development of AR for the Web a straightforward process that can be implemented by novice coders. It does not require the skills of a mobile app developer (Stewart, 2017), or a headset, like the Microsoft HoloLens (Microsoft, 2018). AR.js and A-Frame are open source software, not proprietary tools like ZapWorks (ZapWorks, 2018). Other open source development solutions for Web AR include argon.js (Argon.js, 2018), developed at the AEL (Augmented Environments) Lab at Georgia Tech. Argon also supports A-Frame (Aframe.argonjs.io, 2017), but requires learning to code in argon.js, a level of skill not needed when developing with AR.js and A-Frame. Working with AR.js, developers implement attributes in AR.js and utilize the A-Frame API to develop AR Web pages for use on desktop computers and mobile devices. AR markers are recognized and tracked using the camera, and the 3D content is positioned relative to this spatial information. In this session, a development process is demonstrated for working with AR.js and A-Frame. Web pages will be developed for use with AR markers, printed, and displayed on computer screens. 3D models will be created using A-Frame primitives and glTF 2.0 (GL Transmission Format) models. Online resources are provided for the all of the session content, underlying technologies, and the various implementations of AR.js.

AR markers

computer screen; a separate display from the mobile device and browser with AR-enabled content, the marker can be read by the mobile device camera by aiming it at the screen.

Figure 1: Hiro marker, example of AR pattern marker

Developing AR Web pages using AR.js and A-Frame

A-Frame is a Web framework for developing VR content. Created as an extension to three.js, A-Frame is

To begin developing an AR Web page with A-Frame and AR.js, first include the JavaScript libraries in the <head> tag of the .html page:

```
<head>

<script src="https://aframe.io/releases/0.7.0/aframe.min.js"></script>

<script src="https://jeromeetienne.github.io/AR.js/aframe/build/aframe-ar.js"></script>

</head>
```

Note that the A-Frame JavaScript is the most current version, 0.7, and that the AR.js library is the version developed for use with A-Frame.

Following the basics for building a scene in A-Frame (A-Frame, 2015-2017e), https://aframe.io/docs/0.7.0/guides/building-a-basic-scene.html, start by adding the <a-scene> tag to the html code. This allows for the setup of the scene for various devices. Modify this tag with AR.js for A-Frame (Etienne, J., 2017a), https://github.com/jeromeetienne/AR.js/tree/master/aframe, by declaring the sourcceType attribute:

```
<a-scene embedded arjs='sourceType: webcam;'>
```

A-Frame primitive models

Inside the <a-scene> add entities (objects). These can include a box, cylinder, plane, or sphere. To include a box, declare the primitive as follows:

```
<a-box position="0 0.5 -0.5" color="blue" scale="1 0.5 1" rotation= "0 45 0"
material='opacity: 0.9;'>

</a-box>
```

In this example the position, color and material components are also set. The position is set on the x,y,

2015-2017d), https://aframe.io/docs/0.7.0/primitives/a-box.html.

From AR.js, set the AR marker, by adding the <a-marker-camera> entity. In this example, the preset parameter is set to the "hiro" marker:

```
<a-marker-camera preset='hiro'>
</a-marker-camera>
```

Using the <a-marker-camera> entity causes the camera, not the object, to be transformed as perspective changes.

The completed Web page example html code is:

```
<!doctype html>

<html lang="en">

<head>

<script src="https://aframe.io/releases/0.7.0/aframe.min.js"></script>

<script src="https://jeromeetienne.github.io/AR.js/aframe/build/aframe-ar.js"></script>

</head>

<body style='margin : 0px; overflow: hidden;'>

<a-scene embedded arjs='sourceType: webcam;'>

<a-box position="0 0.5 -0.5" color="blue" scale="1 0.5 1" rotation= "0 45 0"

material='opacity: 0.9;'></a-box>
```

Figure 2: printed pattern AR marker, box primitive, Google Pixel XL, Chrome mobile browser with example code

A-Frame primitives with textures

To add a texture to a primitive model in A-Frame, add an <a-asset> entity and create an img tag with id and source parameters, and add a source parameter to the primitive entity pointing to the ID, for example:

```
<a-scene embedded arjs='sourceType: webcam;'>

<a-assets>
```

0 45 0" material="opacity: 0.9;"></a-box>

<a-marker-camera preset='hiro'></a-marker-camera>

</a-scene>

Figure 3: printed pattern AR marker, box primitive with texture map, Google Pixel XL, Chrome mobile browser with example code

Animation with A-Frame

Animation is introduced by accessing the built-in animation system in A-Frame, referenced at (A-Frame, 2015-2017c) https://aframe.io/docs/0.7.0/core/animations.html. The <a-animation> element is added as a child of the model entity, for example:

```
<a-box src="#boxTexture" position="0 0.5 0" color="blue" material='opacity: 0.8;'>

<a-animation attribute="position" to="5 0 0" direction="alternate" dur="3000"

repeat="indefinite"></a-animation>

</a-box>
```

In this example, the position is animated from its original position to the "to" xyz parameter setting. The duration is measured in milliseconds and the motion is set to an indefinite loop.

Including glTF models

A-Frame allows for the inclusion of glTF 2.0 format models. Models can be downloaded from online sources (Sketchfab, 2017) (Khronos Group, 2018b), or created in 3D modeling software, for example,

<a-assets>

<a-asset-item id="lantern" src="lantern/Lantern.gltf"></a-asset-item>

</a-assets>

<a-gltf-model src="#lantern" position=" 0 0 0" scale="0.2 0.2 0.2"></a-gltf-model>

<a-marker-camera preset='hiro'></a-marker-camera>

</a-scene>

In the example, the model is located on the Web server in the same location as the html file that it is linked to. The <a-asset> entity is used, as in the primitive with texture map example, but, in this case, it is referencing the model directory which includes the model and all texture maps. The <a-gltf-model> entity is used to place the model in the scene.

Figure 4: printed pattern AR marker, glTF 2.0 model with textures, Google Pixel XL, Chrome mobile browser with example code.

Resources

AR.js

https://github.com/jeromeetienne/ar.js

A-Frame

ARToolKit

https://github.com/artoolkit/artoolkit5

https://archive.artoolkit.org

AR Markers

Types of Markers

https://aframe.io/blog/arjs/#different-type-of-markers-pattern-and-barcode

Multi-Markers

https://medium.com/arjs/area-learning-with-multi-markers-in-ar-js-1ff03a2f9fbe

Multiple Distinct Markers

https://aframe.io/blog/arjs/#how-to-handle-multiple-distinct-markers

Creating Custom Markers

https://aframe.io/blog/arjs/#customize-your-marker

https://artoolkit.org/documentation/doku.php?id=3_Marker_Training:marker_training<

Three.js

https://threejs.org

WebGL

https://www.khronos.org/webgl

Sources for Models

https://sketchfab.com

https://github.com/KhronosGroup/glTF-Sample-Models/tree/master/2.0

Converter for Blender

https://github.com/KhronosGroup/glTF-Blender-Exporter

References

A-Frame. (2015-2017a). "Introduction to A-Frame." Consulted on September 18, 2017. Available https://aframe.io/docs/0.7.0/introduction

A-Frame. (2015-2017b). Sources of information on A-Frame software. Consulted on September 18, 2017. Available https://aframe.io

A-Frame. (2015-2017c). Sources of information on Animations in A-Frame software, version 0.7.0, Core API. Consulted September 18, 2017. Availablehttps://aframe.io/docs/0.7.0/core/animations.html

A-Frame. (2015 – 2017d). Sources of information on Primitives and Attributes in A-Frame software, version 0.7.0, Primitives. Consulted September 18, 2017. Available https://aframe.io/docs/0.7.0/primitives/a-box.html

A-Frame. (2015 -2017e). Sources of information on The Basics for Building a Scene in A-Frame software, version 0.7.0, Guides. Consulted September 18, 2017. Available https://aframe.io/docs/0.7.0/guides/building-a-basic-scene.html

Aframe.argonjs.io. (2017). Sources of information on Argon and A-Frame Examples. Last updated June

Avaialble https://artoolkit.org/documentation/doku.php?id=3_Marker_Training:marker_training

ARToolkit. (2016b). Sources of information on ARToolkit software. Last updated April 1, 2016. Consulted September 19, 2017. Available https://archive.artoolkit.org

ARToolkit5. (2016). Sources of information on ARToolkit5 software. Last updated November 5, 2017. Consulted September 19, 2017. Available https://github.com/artoolkit/artoolkit5

Etienne, A. (2017). "Area Learning with Multi-Markers in AR.js." *Medium.com*. Published June 13, 2017. Consulted September 25, 2017. Available https://medium.com/arjs/area-learning-with-multi-markers-in-ar-js-1ff03a2f9fbe

Etienne, J. (2017a). Sources of information on AR.js software for use with A-Frame software. Last updated September 30, 2017. Consulted September 20, 2017. Available https://github.com/jeromeetienne/AR.js/tree/master/aframe

Etienne, J. (2017b). "Creating Augmented Reality with AR.js and A-Frame." A-Frame Blog. July 11, 2017. Consulted September 25, 2017. Available https://aframe.io/blog/arjs/#different-type-of-markers-pattern-and-barcode

Etienne, J. (2017c). Sources of information on AR.js software. Last updated September 30, 2017. Consulted September 20, 2017. Available https://github.com/jeromeetienne/ar.js

Filipovic, E. (2009). "A Museum That is Not." *e-flux journal #04*. March, 2009. Consulted September 15, 2017. Available http://www.e-flux.com/journal/04/68554/a-museum-that-is-not

Khronos Group. (2018a). Sources of information on glTF Blender Exporter software. Last updated January 29, 2018. Consulted September 19, 2017. Available https://github.com/KhronosGroup/glTF-Blender-Exporter

Khronos Group. (2018b). Sources of information on glTF Sample Models, version 2.0. Last updated

Sketchfab. (2017). Sources of information on Sketchfab, a source for glTF models. Last updated January 30, 2018. Consulted September 19, 2017. Available https://sketchfab.com

Stewart, J. (2017). "Understanding Augmented Reality (AR) and App Development." *D-Zone*. March 9, 2017. Consulted December 5, 2017. Available https://dzone.com/articles/understanding-augment-reality-ar-and-app-developme

Three.js. (2017). Sources of information on Three.js software. Last updated December 18, 2017. Consulted September 21, 2017. Available https://threejs.org

WebRTC. (2017). Sources of information on WebRTC software. Last updated November 2, 2017. Consulted September 25, 2017. Available https://webrtc.org

ZapWorks (2018). Sources of information on ZapWorks software. Last updated January 1, 2018. Consulted January 15, 2018. Available https://zap.works

Cite as:
Leighton, Fred. "The portARble museum: Developing augmented reality for the Web using AR.js." *MW18: MW 2018*. Published January 15, 2018. Consulted January 28, 2022. https://mw18.mwconf.org/paper/the-portarble-museum-developing-augmented-reality-for-the-web-using-ar-js/