**Progetto stage - Demo AR**

**Realtà aumentata (AR) sul web**

Ridotta in minimi termini: tecnica per permettere di aggiungere (in tempo reale) degli oggetti virtuali ad immagini provenienti da una sorgente di acquisizione video (es. webcam)

Come realizzare un’esperienza di realtà aumentata (AR) sul web (Marker Based):

* Possedere uno o più **marker**, ovvero delle immagini speciali che saranno utilizzate dal software come punti di riferimento per decidere dove posizionare gli elementi 3D all’interno dell’immagine ripresa dalla fotocamera/webcam
* Creare uno o più **oggetti 3D** da associare ai marker. Questi oggetti possono essere creati:
  + Sfruttando delle primitive messe a disposizione da A-Frame
  + Attraverso software di modellazione 3D (es. Unity, Blender, …)
* Scrivere una **pagina web** di codice che permetta di catturare l’ingresso video dentro un canvas e indicare, ad esempio ad AR.js, quali sono i marker da riconoscere
* Eseguire la pagina web generata attraverso un **web server** che “serva” la modalità HTTPS, dato che l’accesso alla webcam avviene esclusivamente se la connessione è protetta

AR examples

* Sito lamborghini: <https://www.lamborghini.com/sites/it-en/files/js_assets/experience/assets/ar/index.html?utm_source=3dplatform&utm_medium=qrcode&utm_campaign=augmented_reality_huracan_sto>

**AR.js**

AR.js è una libreria Javascript per la Realtà Aumentata (o Augmented Reality, AR) sul Web.

AR.js offre tre differenti tipologie di realtà aumentata:

* **Image Tracking**: in questo caso quando la camera trova un’immagine 2D è possibile mostrare su di essa (o vicino) del contenuto. Questo contenuto possono essere immagini 2D, immagini 3D, GIF, video 2D, …
* **Location Based AR**: vengono utilizzati luoghi e posizioni del mondo reale per mostrare della realtà aumentata attraverso il dispositivo dell’utente.
* **Marker Tracking**: in questo caso quando la camera trova un marker è possibile mostrare su di essa (o vicino) del contenuto, come nell’Image Tracking.

AR.js è una soluzione puramente web, non serve quindi nessuna installazione ed è scritta in Javascript basandosi su three.js, A-frame e jsartoolkit5.

Funziona inoltre su tutti i telefoni con webgl e webrtc.

**Test AR.js**

La tipologia di AR utilizzata è Marker Tracking.

Creazione sito web locale per test

* Ri-creazione del sito web <https://stemkoski.github.io/AR-Examples/> in locale (tramite github <https://github.com/stemkoski/AR-Examples> ) per vederlo runnato su un server locale (utilizzato XAMPP e creazione di una nuova directory virtuale)

Creazione pagina web di esempio AR.js-three.js: test-torusknot

* Test di visualizzazione di Torus Knot (utilizzo webcam del pc + immagine del marker pattern Hiro sul cellulare per una corretta visualizzazione)
* Utilizzo di three.js
* File test-torusknot.html

Creazione sample pagina web AR.js: ar\_sample.html (funzionamento di AR.js)

* Commentato la pagina web per spiegare funzionamento

Attraverso la creazione di pagine di testing con AR.js si può già notare, seppur in modo semplice, l’utilizzo del web per vivere un’esperienza di realtà aumentata. Infatti spostando la sorgente video oppure muovendo il marker si vede che l’oggetto 3D si adatta al movimento (sinistra-destra/rimpicciolamento-ingrandimento).

**Three.ar.js**

Three.ar.js è una libreria three.js di supporto utilizzata per sviluppare esperienze di realtà aumentata attraverso **WebARonARKit** e **WebARonARCore**.

WebARonARKit e WebARonARCore sono delle app sperimentali, rispettivamente per iOS e Android, che permettono agli sviluppatori la creazione di realtà aumentata utilizzando il web.

**A-Frame**

A-Frame è un framework web per lo sviluppo di esperienze di realtà virtuale (VR).

A-Frame si basa su HTML, ma il suo punto di forza è che fornisce una struttura dichiarativa, estensibile e componibile a **three.js**.

**Test A-Frame**

* Creazione pagina web semplice A-frame (Hello WebVR) con l’aggiunta di alcuni elementi 3D
* Aprire l’inspector (cntrl + alt + i) per la visualizzazione e il testing dello spostamento dei vari elementi creati
* Creazione pagina web <https://glitch.com/edit/#!/aframe-model-viewer?path=index.html%3A4%3A4> in locale per testing

Successivamente le cose iniziano a diventare più complesse, complice anche il fatto della documentazione non troppo dettagliata di AR.js. C’è da ricordare però che stiamo parlando di cutting-edge technology ovvero una tecnologia nuova e presente da poco sul web.

**Marker personalizzati**

Possono essere creati dei marker personalizzati, rendendoli appropriati (es. “brandizzandoli”).

Per la creazione di un nuovo marker posso utilizzare il sito <https://jeromeetienne.github.io/AR.js/three.js/examples/marker-training/examples/generator.html>

Considerazioni:

* Non tutti i tipi di immagine vanno bene per creare un marker che venga poi riconosciuto
* Le immagini non devono essere troppo complesse
* Possono esserci colori, ma deve esserci contrasto tra sfondo e simboli
* File .patt -> File che verrà utilizzato da AR.js per riconoscere il pattern del marker
  + 4 matrici bidimensionali che contengono 16x48 valori, in cui sono rappresentati i canali RGB
  + Solitamente file di pattern non validi hanno tutti 0 come valore (immagine completamente nera)
* Pattern ratio <= 0.75 (risultati migliori)

**Test Marker personalizzati**

Creazione di vari marker sfondo bianco/grigio

* Creazione di 1 marker con sfondo bianco -> Riconoscimento NOT SUCCESS
* Creazione di 3 marker con sfonfo grigio (più o meno dettagliati) -> Riconoscimento NOT SUCCESS

Creazione marker4

* Creazione di 1 marker (marker4) personalizzato (Max) con sfondo bianco
* Utilizzo di marker1 (uguale a marker4 ma con sfondo grigio) come immagine da riconosce
* Riconoscimento del software SUCCESS

Creazione “Company Marker”: Vegan Solutions (logo vegan/s)

* Creazione marker aziendale (come marker4)

**Far funzionare XAMPP sul cellulare**

Tutto il codice delle pagine web per essere testato viene eseguito su XAMPP (web server Apache), per testare il tutto anche da mobile (esperienza migliore) XAMPP deve essere raggiungibile anche da mobile.

Quando pc e smartphone sono collegati alla stessa rete WIFI:

* Modificare file di xampp -> httpd-xampp.conf (Require all granted)
* Restart XAMPP e avviare Apache web server
* Creare una Inbound Rule in Windows Firewall selezionando la porta corretta (80 nel mio caso) e permettendo la connessione
* Controllare da dispositivi nella rete l’indirizzo IP del pc con XAMPP (es. 192.168.4.73)

Tutto molto bello ma rimane il problema dell’https, ovvero per avere l’accesso alla fotocamera serve una connessione protetta.

* Soluzione migliore -> <https://ngrok.com/> (one command for an instant, secure URL to your localhost server through any NAT or firewall)
* Soluzione palesemente adottata da me -> mettere https davanti all’url 😊

**Riferimenti**

Realtà aumentata con AR.js: <https://devsware.wordpress.com/2018/07/25/realta-aumentata-con-ar-js/>

* <https://aframe.io/blog/arjs/>

Documentazione ufficiale AR.js: <https://ar-js-org.github.io/AR.js-Docs/>

Github organization AR.js: <https://github.com/ar-js-org>

Github three.ar.js: <https://github.com/google-ar/three.ar.js?files=1>

Documentazione ufficiale A-Frame: <https://aframe.io/>

Github A-Frame Inspector: <https://github.com/aframevr/aframe-inspector>

Documentazione ufficiale three.js: <https://threejs.org/>

Xampp da mobile: <https://people.utm.my/shaharil/access-pc-localhost-xampp-server-from-mobile/>

* <https://github.com/jeromeetienne/AR.js/issues/463>

**Link utili**

Color piker W3Cschool: <https://www.w3schools.com/colors/colors_picker.asp>

Esempi tesi AR

* <https://webthesis.biblio.polito.it/13182/1/tesi.pdf>
* <http://tesi.cab.unipd.it/57105/1/Elena_Rostellato_2017.pdf>
* <http://dspace.unive.it/bitstream/handle/10579/10683/842282-1210441.pdf?sequence=2>
* …