Marker Based

Markers can be of three, different types:

- Hiro
- Barcode
- Pattern.

To learn more about markers, please read this articles:

- AR.js basic Marker Based tutorial and Markers explanation (https://medium.com/chialab-open-source/ar-js-the-simpliest-way-to-get-cross-browser-ar-on-the-web-8f670dd45462)
- Deliver AR.js experiences using only QRCodes (Markers inside QRCodes) (https://medium.com/chialab-open-source/how-to-deliver-ar-on-the-web-only-with-a-qr-code-e24b7b61f8cb).

TL:DR

- Hiro Marker is the default one, not very useful actually
- Barcode markers are auto-generated markers, from matrix computations. Learn more on the above articles on how to use them. If you need the full list of barcode markers, here it is (https://github.com/nicolocarpignoli/artoolkit-barcode-markers-collection)
- Pattern markers are custom ones, created starting from an image (very simple, hight contrast), loaded by the user.

⚡ You can create your Pattern Markers with this tool (https://ar-js-org.github.io/AR.js/three.js/examples/marker-training/examples/generator.html). It will generate an image to scan and a `.patt` file, to be loaded on the AR.js web app, in order for it to recognise the marker when running.

How to choose good images for Pattern Markers

Markers have a black border and high contrast shapes. Lately, we have added also white border markers with black background, altough the classic ones, with black border, behave better.

Here's an article explaining all good practice on how to choose good images to be used to generate custom markers: 10 tips to enhance your AR.js app (https://medium.com/chialab-open-source/10-tips-to-enhance-your-ar-js-app-8b44c6faffca).

API Reference for Marker Based

A-Frame

`<a-marker/>`

Here are the attributes for this entity

Attribute	Description	Component Mapping
type	type of marker - ['pattern', 'barcode', 'unknown']	artoolkitmarker.type
size	size of the marker in meter	artoolkitmarker.size
url	url of the pattern - IIF type='pattern'	artoolkitmarker.patternUrl
value	value of the barcode - IIF type='barcode'	artoolkitmarker.barcodeValue
preset	parameters preset - ['hiro', 'kanji']	artoolkitmarker.preset
emitevents	emits 'markerFound' and 'markerLost' events - ['true', 'false']	-
smooth	turn on/off camera smoothing - ['true', 'false'] - default: false	-
smoothCount	number of matrices to smooth tracking over, more = smoother but slower follow - default: 5	-
smoothTolerance	distance tolerance for smoothing, if smoothThreshold # of matrices are under tolerance, tracking will stay still - default: 0.01	-
smoothThreshold	threshold for smoothing, will keep still unless enough matrices are over tolerance - default: 2	-

three.js

threex-artoolkit

threex.artoolkit is the three.js extension to easily handle artoolkit (https://github.com/artoolkitx/jsartoolkit5).

Architecture

threex.artoolkit is composed of 3 classes

- `THREEx.ArToolkitSource` : It is the image which is analyzed to do the position tracking. It can be the webcam, a video or even an image

- `THREEx.ArToolkitContext`: It is the main engine. It will actually find the marker position in the image source.
- `THREEx.ArMarkerControls`: it controls the position of the marker It use the classical three.js controls API (https://github.com/mrdoob/three.js/tree/master/examples/js/controls). It will make sure to position your content right on top of the marker.

THREEx.ArMarkerControls

```
var parameters = {
  // size of the marker in meter
  size: 1,
  // type of marker - ['pattern', 'barcode', 'unknown' ]
  type: "unknown",
  // url of the pattern - IIF type='pattern'
  patternUrl: null,
  // value of the barcode - IIF type='barcode'
  barcodeValue: null,
  // change matrix mode - [modelViewMatrix, cameraTransformMatrix]
  changeMatrixMode: "modelViewMatrix",
  // turn on/off camera smoothing
  smooth: true,
  // number of matrices to smooth tracking over, more = smoother but slower follow
  smoothCount: 5,
  // distance tolerance for smoothing, if smoothThreshold # of matrices are under tolera
nce, tracking will stay still
  smoothTolerance: 0.01,
  // threshold for smoothing, will keep still unless enough matrices are over tolerance
  smoothThreshold: 2
};
```

THREEx.ArToolkitContext

```
var parameters = {
    // debug - true if one should display artoolkit debug canvas, false otherwise
    debug: false,
    // the mode of detection - ['color', 'color_and_matrix', 'mono', 'mono_and_matrix']
    detectionMode: 'color_and_matrix',
    // type of matrix code - valid iif detectionMode end with 'matrix' - [3x3, 3x3_HAMMI
NG63, 3x3_PARITY65, 4x4, 4x4_BCH_13_9_3, 4x4_BCH_13_5_5]
    matrixCodeType: '3x3',
    // Pattern ratio for custom markers
    patternRatio: 0.5
    // Labeling mode for markers - ['black_region', 'white_region']
    // black_region: Black bordered markers on a white background, white_region: White b
ordered markers on a black background
    labelingMode: 'black_region',

    // url of the camera parameters
    cameraParametersUrl: THREEx.ArToolkitContext.baseURL + '../data/data/camera_para.da
t',

    // tune the maximum rate of pose detection in the source image
    maxDetectionRate: 60,
    // resolution of at which we detect pose in the source image
    canvasWidth: 640,
    canvasHeight: 480,

    // enable image smoothing or not for canvas copy - default to true
    // https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/imageSm
oothingEnabled
    imageSmoothingEnabled : true,
}
```

THREEx.ArToolkitSource

```
var parameters = {
  // type of source - ['webcam', 'image', 'video']
  sourceType: "webcam",
  // url of the source - valid if sourceType = image|video
  sourceUrl: null,

  // resolution of at which we initialize the source image
  sourceWidth: 640,
  sourceHeight: 480,
  // resolution displayed for the source
  displayWidth: 640,
  displayHeight: 480
};
```