Extended Reality
Marker based augmented reality

Jean-Yves Didier

Lab goals :
— Getting used to WebGL and Three.js ;
— Being able to create and animate a scene graph.

Students's requirements :
— Being able to write a program, either in javascript or C language.

Environment's requirements :
— Webbrowser supporting WebGL (Mozilla Firefox / Google Chrome / Microsoft Edge) [1] ;
— Text editor with syntax highlighting [2].

Session's preparation

For this lab, we will be using several javascript libraries :
— ARUCO.js is a javascript port for a marker localisation and recognition library named ARUCO : `https://github.com/jcmellado/js-aruco`
— ARCS.js, component-oriented programming framework for augmented reality : `http://arcs.ibisc.univ-evry.fr`
— Three.js to handle scene graphs. Its documentation can be recovered there : `http://threejs.org/docs/`
— For this, we will use javascript as a programming language. For the ones who are not comfortable with it, you can use the following resources :
`http://eloquentjavascript.net/`
`http://www.w3schools.com/js/`

Main files and directories of the application are given in table 1. They contain component source code upon which you will develop. The webpage to load into browser is named `aruco.html`. This page also loads an application description (file `arcsapp.json`), instantiates components and runs application.

Components use signal/slot paradigm to communicate together. Slots correspond to inputs and signals to outputs that can transmit data or objects. Components signals are connected to other component slots in order to form a processing chain. Component list is given in table 2 and main data stream is represented in figure 1.

Question 6 (switching from pre-recorded video to live camera) can be performed at any moment during lab session, as soon as you have the required hardware (external webcam or internal camera).

1. Browser ordered by preference and compliance, from best to worst.
2. It can be done online with our own CodeLab : `https://deptgi.iup.univ-evry.fr/codelab/`.

Name	**Content**
`build`	Scripts for starting ARCS.js engine
`components`	Application component directory
`deps`	Required dependencies (scripts)
`docs`	Documentation (engine and component API)
`tests/aruco`	Application directory
`tests/aruco/aruco.html`	HTML page that bootstrap application
`tests/aruco/arcsapp.json`	Description/configuration of the application

TABLE 1 – Main files and directories of the application

Name	**Type**	**Function**
animator	Animator	Animation loop iterator
detector	ArucoDetector	Marker detection
locator	MarkerLocator	Pose estimation
objectLoader	OBJLoader	3D object loader
objectTransform	ObjectTransform	3D object spatial transform
video	VideoSource	Video stream acquisition
video	LiveSource	Webcam stream acquisition
viewer	ARViewer	Compositor for images and 3D scences
windowresize	WindowEvent	browser window event detection (resize)
ts	TokenSender	Synchronisation with statemachine
sm	StateMachine	Statemachine controller (related to ARCS engine)

TABLE 2 – Application components and related functions

To do

1. Detect maerkers

In file `arucodetector.js`, complete the component so that it can detect markers (documentation of js-aruco library might guide you).

2. Apply the result of pose estimation

One markers have been detected, corresponding objects are transmitted to `locator` component. It estimates the pose of each marker, that is to say its position and orientation respectively to the camera.

You must fill in the code for method `setExtrinsics(...)` within file `arviewer.js`. You will adjust camera position and orientation accordingly to poses associated to markers. We consider that the marker share the same coordinate system as the scene (including its origin).

At the end of this step, your application should react to the marker with the #17 id, as represented in figure 3. What can you see ? Is the registration accurate ?

3. Real camera frustum computation

Intrinsics parameters of virtual camera should match the real ones. You will have to adjust virtual camera accordingly. In order to achieve this, you will have to introduce some code in method `setIntrinsics(...)` from `arviewer.js` file.

Is the registration better ?

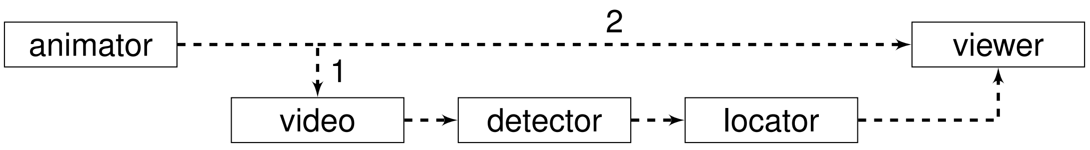


FIGURE 1 – Application data stream

4. Pose estimation dual problem

The ultimate goal is to be able to track and move several objects at the same time (see next question). Instead of modifying camera's position and orientation, we will modify position and orientation of each object.

First, modify application description (file `arcsapp.json`). Replace the line :

```
{ "source": "locator", "signal": "onLocatedMarkers","destination": "viewer",
"slot": "setExtrinsics"}
```

by

```
{ "source": "locator", "signal": "onLocatedMarkers", "destination":
"objectTransform", "slot": "setTransform"}
```

Complete `setTransform(...)` method from `objecttransform.js` file so that the object is moved according to the estimated pose of marker #17.

5. Clone objects for two independent markers

Modify components and application description so that two characters can be registered on two different markers (#17 and #321) at the same time.

You will be able to use video `webcam2.webm` to test multi-markers tracking. If you would like to display other 3d models, you can go on the `https://poly.google.com/blocks` website in order to download the one you want, provided that the file format is OBJ.

6. Switching from a recorded video to live camera stream (for those who can)

In `html` file, comment the video source and suppress its `loop` attribute. In file `arcsapp.json`, replace line : `"video": {"type": "VideoSource"}` by : `"video": {"type": "LiveSource"}`

A Developping for web browsers

A.1 Browser configuration and security

These past years, security of web browsers have been strengthened. These features are in the interest of end-users but not necessarily of developpers. In order to run our simulation we might have to lift some browser restrictions.

A.1.1 Mozilla Firefox

1. In address bar, type `about:config`,then accept the risk ;
2. Look for key `security.fileuri.strict_origin_policy` and change its value to`false`.

A.1.2 Google chrome

Launch browser with the switch `--allow-file-access-from-files`.
Under windows one way to achieve this is :

1. close all *Chrome* windows ;
2. copy *Chrome* icon ;
3. modify its properties (right click, properties menu) ;
4. in dialog box, select *Shortcut* tab ;
5. modify the target and add the switch `--allow-file-access-from-files` ;
6. cliquer sur *Ok* ;
7. launch chrome using the new shortcut icon.

A.1.3 Microsoft Edge

Since Microsoft Edge is using the same restrictions as Google Chrome, the same process will apply.

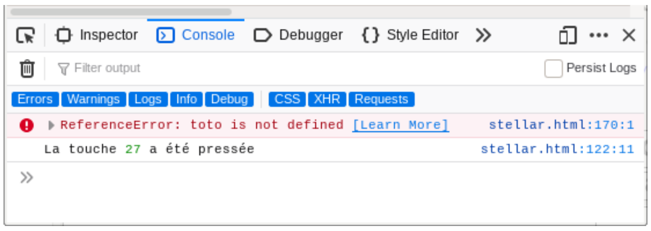
A.2 Enabling development tools

Recent web browser also have development tools in order to speed up development process such as a console that display information, warning or error messages as we can see in figure 2 page suivante.
One can read :
— An error message stating that a *toto* variable is used but is not defined in file *stellar.html* at line 170 ;
— An information message *key 27 has been pressed*.
Table 3 tells how to enable development tools in your browser.

Finally, if you want to display your own message, you can use the instruction `console.log(param1, param2, ...)` that can display almost everything that is passed as a parameter.

Browser	Key combo	By the navigation bar
Mozilla Firefox	Ctrl Shift ⇑ K	≡ › Web Development › Web Console
Google Chrome	Ctrl Shift ⇑ I	⋮ › More tools › Development tools
Microsoft Edge	F12	··· › Development tools

TABLE 3 – Activating development tools in browsers



```
ReferenceError: toto is not defined [Learn More]        stellar.html:170:1
La touche 27 a été pressée                             stellar.html:122:11
```

FIGURE 2 – Screenshot of browser development tools (Firefox browser).

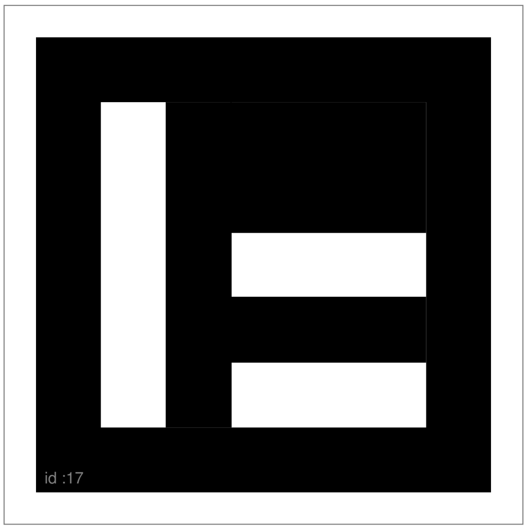
A.3 Keys and keycodes matching

Table 4 gives the keycodes corresponding to the keys we will use during the lab. A complete list can be obtained at the address :
`http://gcctech.org/csc/javascript/javascript_keycodes.htm`

Key	Keycode	Key	Keycode
Page ↑	33	Page ↓	34
←	37	↑	38
→	39	↓	40
C	67	F	70

TABLE 4 – Keys and keycodes mapping

B Markers to use during the lab session



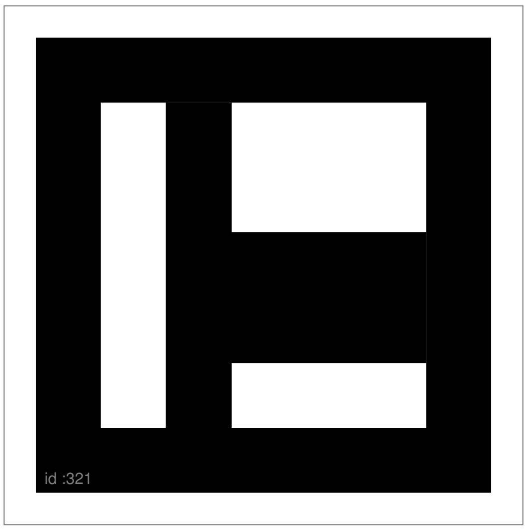


FIGURE 3 – Markers to use during the lab session