**How to Use the App**

Ensure that the webcam on the laptop is activated/switched on. To launch the application, open the folder "mechamarkers" and run "mechamarkers.exe" (on Mac it is just a regular app). You should see the webcam feed. Identified markers will appear as red rectangles with an ID number. We are using OpenCV's ArUco marker library. The panel on the left of the program enables you to modify the marker detection parameters. Hovering your cursor over a parameter will reveal a tooltip with a description of each parameter.

*Important parameters to note:*

[Camera Index]: Use this to toggle between multiple cameras that are connected to the computer.

[Flip Camera]: Check to flip the camera image. This is to detect markers that are mirrored.

[White Balance]: Changing this value might help improve image contrast for marker detection.

[minMarkerPerimeterRate]: This sets the smallest marker size the system will try to detect, based on a percentage of the image's width.

[minMarkerPerimeterRate]: This sets the largest marker size the system will try to detect, based on a percentage of the image's width.

[perspectiveRemoveIgnoredMarginPerCell]: Increasing this value might improve marker detection but also increase the chance of false positives.

[maxErroneousBitsInBorderRate]: Increasing this value might improve marker detection but also increase the chance of false positives.

[errorCorrectionRate]: Increasing this value might improve marker detection but also increase the chance of false positives.

**Making Your own markers and best practices**

To generate markers we suggest you use [this webpage](http://chev.me/arucogen/). Copy the svg from that page and then print it at whatever size you desire. We have found that detection is most consistent when markers are printed in black and white and have at least a 2.5mm white border around them. We also recommend that you don’t use marker 17 since it is often mistaken by the detection system for other elements in the environment. Presently we are limiting the system to only detect markers 0 - 50, so you cannot use any outside of that range.

**Using the Input Generator**

To actually get your markers to be viewed by our app as inputs rather than just ids and positions, you must assign them to Input Group configs in the Input Generator. A link to the Input Generator is located in the upper left corner of the main page of the desktop app. There you will see three buttons, load a previously stored config, save the current config, and add a group to the current config. Groups are automatically given a name which you can change to anything you would like.

You will also notice the camera video output to the right with markers outlined in a white border. Once you create a group you can set it’s Anchor (the identifying marker for the group) by clicking on the anchor field then a marker in the video stream.

To add inputs to a group, simply click the add input button. This will create a generic input that will not be saved until you give it one of the four input types from the dropdown. From there you can set the Actor (the identifying marker for the input) the same way you did for the group. Each marker has its own properties, one of the most important being relative position. Make sure you click the “set” button for relative position so that the Input Generator can calculate this value which we use under the hood. Additionally, slider inputs are special and require you to set two relative positions, a start and an end, by manually placing the marker at the start and end of the slider track before calculating each.

**Developing with Javascript**

Currently there are two ways we support creating web apps with Javascript. Through a local server and using Glitch. For those interested in working with a local server, you will need to install [Node.js](https://nodejs.org/en/) on your computer and then follow the setup guide in the readme from [this repository](https://github.com/atlas-acme-lab/mechamarkers-boilerplate), this also contains an API overview. If you just want to test an idea, we have created a starter project using the online code editor Glitch which you can [fork here](https://glitch.com/edit/#!/mechamarkers-starter). **Out of these two options we recommend you start with Glitch.**

**Developing with Processing**

<https://github.com/clementzheng/mechamarkers-processing>

**Source Code**

For those interested here is the source. Although documentation is incomplete on how to build it yourself right now so ask Peter if you wish to make your own build.

* [Detection Server Source](https://github.com/Petroochio/mecha-markers-detection)
* [Desktop App Source](https://github.com/atlas-acme-lab/mechamarkers-app)