FAQ

FAQs, Issues and Troubleshooting

Updated Today (moments ago) by r.biedert

General

- **What is an eye tracker? What is a *brain* tracker?**

  An eye tracker is a device that reports on which part of the screen you look at. A brain tracker is just another name for EEG.

- **Do I need an eye tracker?**

  Just to get started right away: no. The framework comes with a simulator that can help you to develop apps without any specific hardware. However, in order to really test your apps, you should definitely use a real device.

- **What does 'Certified' mean in the downloads-section?**

  Certified means that (part of) this specific version has been successfully tested by an independent, external partner. The test details can be reviewed in the specific certificate, which will be listed below soon. In general we recommend the latest *Featured* version as it should contain most features and least bugs. However, we will always provide you with the latest certified version for fallback purposes as well.

  Please note that the certificate **does not guarantee any particular functionality or fitness**. You still download and use the software and information provided **at your own risk**.

- **What browser do you recommend?**

  Unfortunately there is no simple choice. Both supported ones (Chrome, Firefox, Safari also mostly works) have their strengths and weaknesses. Some are very fast for certain JavaScript operations (e.g., Chrome + Safari) while others (like sometimes Firefox 4) outperform the others in complex DOM/CSS manipulation scenarios.

Usage

- **Right now the gaze position always seems a bit off. What's wrong?**

  Please run the 'Browser Calibration' first.

- **How does mouse input work?**

  It's pretty straightforward: virtual eye tracking events are generated at the position of the mouse cursor. For example, if you hold your mouse still for a brief moment, a fixation should be detected. This way you can test interaction handlers like `onFixation`, without connecting to a tracker.

- **How can I use the simulator? Is it the same as using no tracking server and just the mouse?**

  The simulator is an adapter of the tracking server. You can enable it by specifying `gazeadapter:simulator` within the file `config.properties` (don't forget to restart the tracking server afterwards!). It differs from the mouse fallback in that it randomly generates fixations around the mouse position and, in addition, adds some noise. Data generated like this should look much more realistic. If you increase the fixation and measurement noise (both also done in the file `config.properties`) you should get a feeling how your app behaves under bad eye tracking conditions.

```
de.dfki.km.text20.trackingserver.eyes.remote.impl.TrackingServerRegistryImpl.adapter.id=gazeadapter:simulator
```

- **I have an eye tracker! How can I use it?**

  Have a look at the file `config.properties`. First you have to enable the Tobii adapter, then you have to specify the TET Address (`tobii.server`). You can get the TET Address from your *Tobii Browser.* If you have a 1750, or any non-networking device the TET Address is most likely `localhost`. Don't forget to restart the tracking server.

```
de.dfki.km.text20.trackingserver.eyes.remote.impl.TrackingServerRegistryImpl.adapter.id=gazeadapter:tobii
de.dfki.km.text20.trackingserver.eyes.adapter.impl.tobii.TobiiGazeAdapter.tobii.server=TX120-203-83900194.local.
```

- **I start the file trackingserver.exe, but it does not even show up in the task manager / the eye tracker still doesn't come up.**

That is a real problem then. Starting with release 1.2.0 the tracking server should generate two files: *error.log* and *output.log*. Have a look at these to check for more details, then refer back to this FAQ. In case your question can't be answered, you will receive help in our newsgroup!

- **VerifyError: Cannot inherit from final class (on Linux)**

  Appears to happen on Linux when started with OpenJDK. Try to use Sun's Java.

- **Using the Tobii adapter I get COM problems. It prints something about com4j ...**

  Starting with the SDK Version 5 Tobii changed some parts of the API. Depending on which version of the SDK you have you must set the `tobii.api` key to v5 for the new API (see below) or to v2 for older APIs.

```
de.dfki.km.text20.trackingserver.eyes.adapter.impl.tobii.TobiiGazeAdapter.tobii.api=v5
```

- **When I try to run the Launcher, `System.loadLibrary("edk")` throws `java.lang.UnsatisfiedLinkError` with the message "Can't find dependent libraries"**

  After adding all dependent DLLs to your dependencies directories, append the `dependencies` folder to your `PATH` variable. In Eclipse:

  - Run Configurations
  - Launcher
  - Environment
  - New
  - Name: `PATH`, set the value to `${env_var:PATH};dependencies`

  Also, don't forget to set java.library.path.

- **My browser reports 'too much recursion' on the console when starting an eye tracking app. What can I do?**

  We know about this issue, but don't have a good solution, it might be related to your computer's performance. For most people this issue never appears, for some only if they try very hard (e.g., reloading the page quickly over and over again) and for a few it strikes instantly. In the latter case one user reported (who used a weak netbook) that after upgrading the RAM to 2GB the problem went away.

Development

- **Can I participate? Can I submit patches?**

  By all means, yes! However, before you start, please contact us through the discussion groups. The basic requirements for patches are that they are well documented, free of compiler warnings at high settings (enable all warning in Java/Eclipse), and integrate naturally in the rest of the framework (i.e., adhere to the coding standards of the rest of the project).

- **How can I write extensions?**

  Extensions are written as Java Simple Plugin Framework-plugins. They are a very powerful mechanism that goes way beyond what we presented here. More information on how to write extensions will be provided later, but for the moment have a look at this small example. Export it as a JAR and add it to the configuration. To access Text 2.0 internal functionality, please `@Inject` the needed modules, or obtain them through the `InformationBroker`.

```
@PluginImplementation
public class SomeExtension implements Extension {

    @Init
    public void init() {
        System.out.println("I am there!");
    }

    @ExtensionMethod // (since 1.4)
    public String someMethod(int x) {
        return "You said " + x;
    }
}
```

- **How can I use extensions?**

  To use extensions from a web application, you have specify them in the configuration first, then (after calling `text20.core.init()` and having waited for the plugin to `INITIALIZE`, you may access them through the connector:

```
// Specify an extension
text20.connector.config.extensions = ["extensions/yodeling.jar"];
```

```
// After INITIALIZE was signalled
text20.connector.extensions.someMethod(123)
```

- **I receive too many fixations! My fixation data appears to be wrong.**

  Fixation listeners are called at least three times: When the fixation starts, when it continues and when it ends. In most cases you should check that you operate on `FixationEventType.FIXATION_END` events only. Otherwise you will receive multiple fixation objects for one true fixation.

PEEP Specific

- **The gaze point is off (like 200px or so)**

  Happens when Processing renders 'fullscreen', but the rendering surface doesn't stretch the whole screen (looks as if embedded into a black border). Either go to windowed mode, or go real fullscreen with maxed surface. Anyway, we'll try to fix this bug in one of the next releases.

- **What is the difference between device.x and device.eyes.rawX?**

  As stated above, device.x corresponds to the latest fixation, while device.eyes.rawX corresponds to the latest point of measurement. Many measurement points will be merged into a single fixation. This means device.x gives you a stable eye position for a few hundred milliseconds while the raw measurements are nosier but arrive with almost no delay. The decision which feature is more useful depends on the task, so better try both and compare them.

- **No eye tracking (or simulation) events arrive.**

  The most likely reason is that you don't have any network adapter installed. The plugin uses ZeroConf to discover the tracking server, however, the library we use appears to have some troubles in loopback-only mode. Try to plug in a network cable (or enable WiFi) and try again.

- **It takes a few seconds until the first events arrive ...**

  We know, sorry. The discovery mechanism is not optimized yet. We are optimistic however that we can decrease this initialization time to significantly less than a second.

Limitations

- **What are the limitations of the recording / replay?**

  Right now a couple of constraints apply. First of all you have to spanify and register all elements you want to track. Also, if your page is highly dynamic (elements moving around a lot) your replay probably won't resemble what it looked like on the screen. In addition the recording and replay only supports text nodes right now. Lastly it doesn't like or handle CSS transforms properly.

  To make it short: the smaller your page is (overall), the more text it contains (percentage) and the less it moves (in terms of DHTML) the better. However, that's only for recording. Interaction with `onFixation` and so forth should work just fine with images, DHTML and any kind of transforms.

Background

- **So, what is a fixation? And what is this other thing called saccade?**

  Eyes either move, or they stand still(*). When you look at some text or image, your eyes do not glide over it smoothly. Instead, for some time they hover over a small area (this is called a fixation) or they travel very quickly to the next hovering point (this is called a saccade). (*While looking eyes never stand still perfectly, however, for this purpose these are merely technical details.)

▶ Sign in to add a comment