EEG Control (II)

Rocío Alegre Marzo
November 25th 2011

Glossary

- Affectiv Suite: emotional state.
- Cognitiv Suite: conscious thoughts.
- Expresiv Suite: facial expressions.

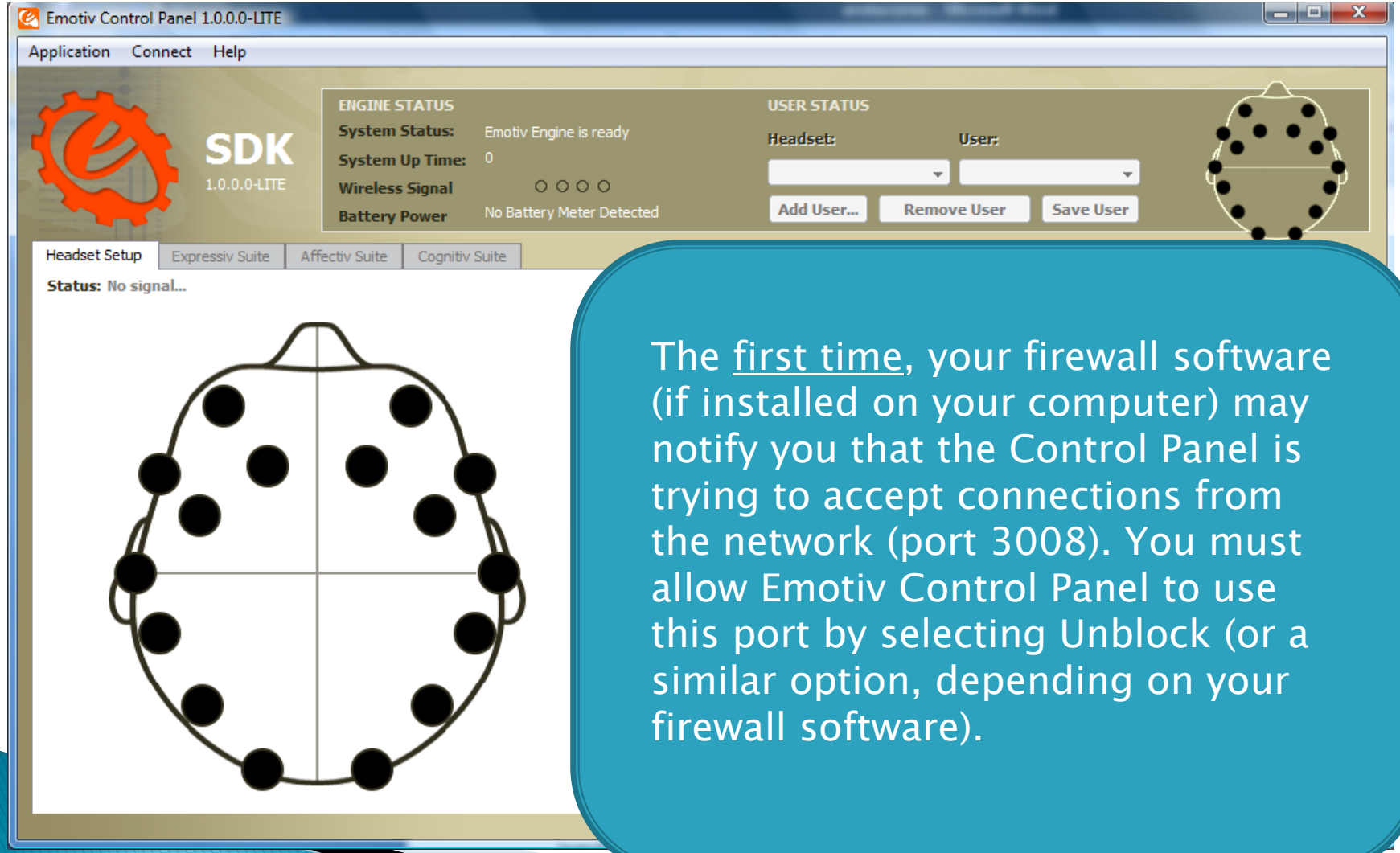
- Emotiv SDKLite™: A version of the Emotiv SDK that uses neuroheadset emulation to allow integration with new and existing software. Software developed will be compatible with the Emotiv EPOC™ headset.

Glossary

- EmoComposer™: An Emotiv EmoEngine™ emulator for development.
- EmoKey™: Tool to translate EmoStates™ into signals that emulate traditional input devices (such as keyboard).

- EML : EmoComposer™ Markup Language – an XML-based syntax that can be interpreted by EmoComposer to playback predefined EmoState values.
- EmoScript™: A text file containing EML, which can be interpreted by EmoComposer to automate the generation of predefined EmoStates.

Parts of the API: Emotiv Control Panel™

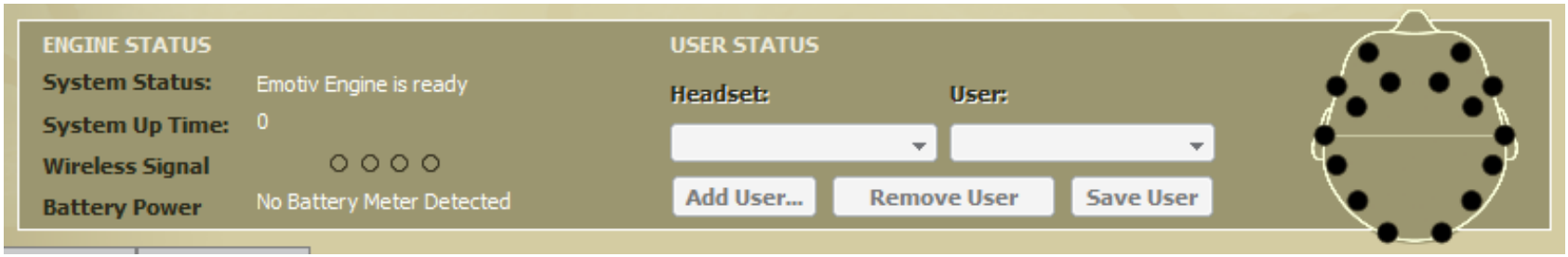


The first time, your firewall software (if installed on your computer) may notify you that the Control Panel is trying to accept connections from the network (port 3008). You must allow Emotiv Control Panel to use this port by selecting Unblock (or a similar option, depending on your firewall software).

Parts of the API: Emotiv Control Panel™

- Provides a GUI (graphical user interface) that interfaces with Emotiv EmoEngine through the Emotiv API.
- Interface showcases the EmoEngine's capabilities to decipher brain signals and present them in useful forms using Emotiv's detection suites.

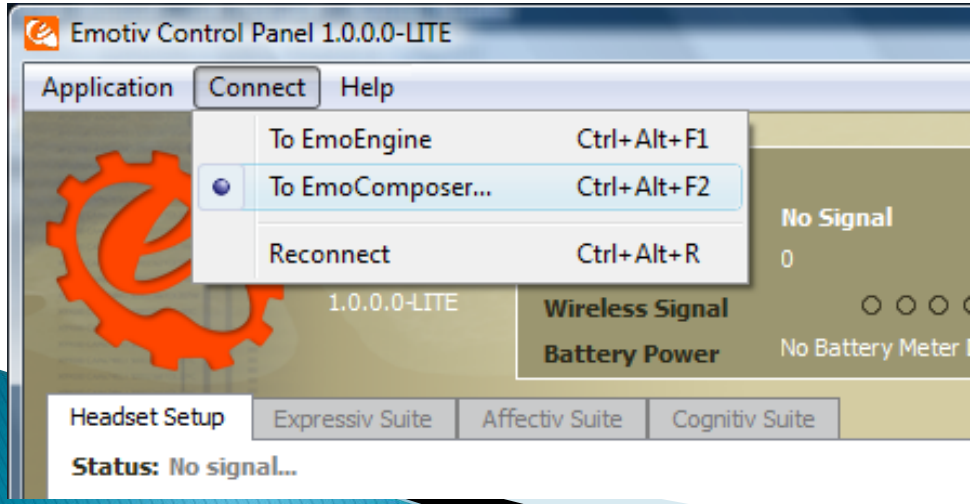
Parts of the API: Emotiv Control Panel™



- This is the EmoEngine Status Pane.
- Displays indicators that provide real-time information status and neuroheadset sensor contact quality.
- It also exposes user profile management controls.

Parts of the API: Emotiv Control Panel™

- May connect to EmoComposer, (emulator tool) from the *Connect* menu.
- SDKLite Developers: **you will need to change this menu setting and connect to EmoComposer.**
- EmoComposer should be launched prior to selecting this option in Control Panel.

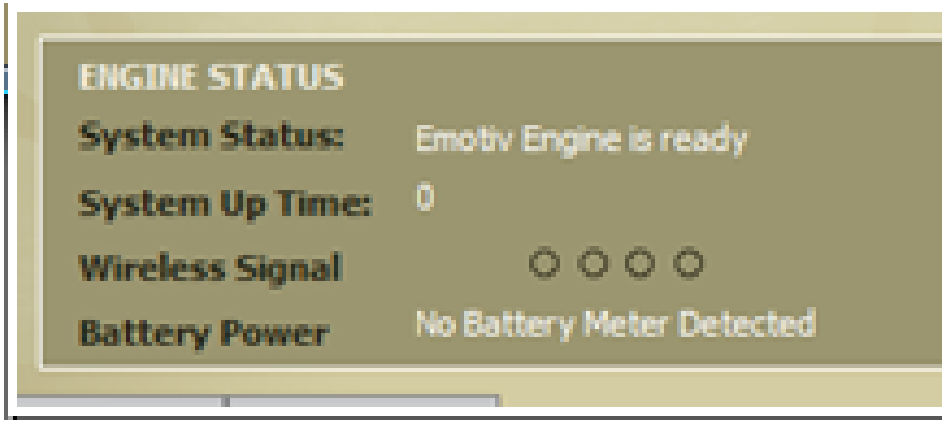


Host: 127.0.0.1

Port: 3008

Parts of the API: Emotiv Control Panel™

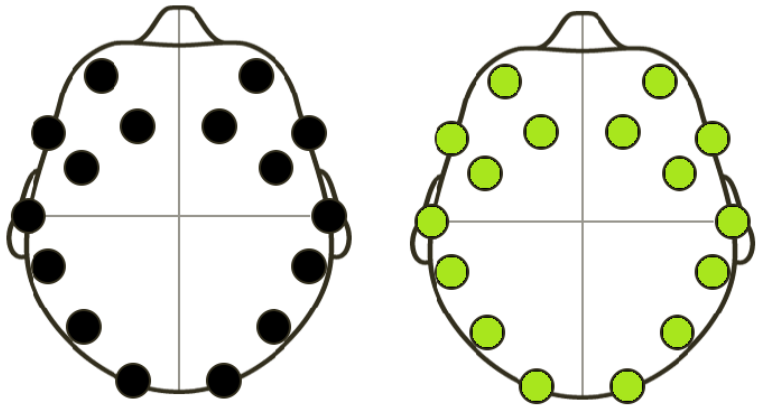
- *System Status*: A summary of the general EmoEngine status.
- *System Up Time*: The timestamp (in seconds) attached to the most recently received.
- *Wireless Signal*: This displays the quality of the connection between the neuroheadset and the Emotiv wireless USB receiver connected to your machine.
- *Battery Power*: Displays an approximation of the remaining charge in the neuroheadset's built-in battery.



Parts of the API: Emotiv Control Panel™

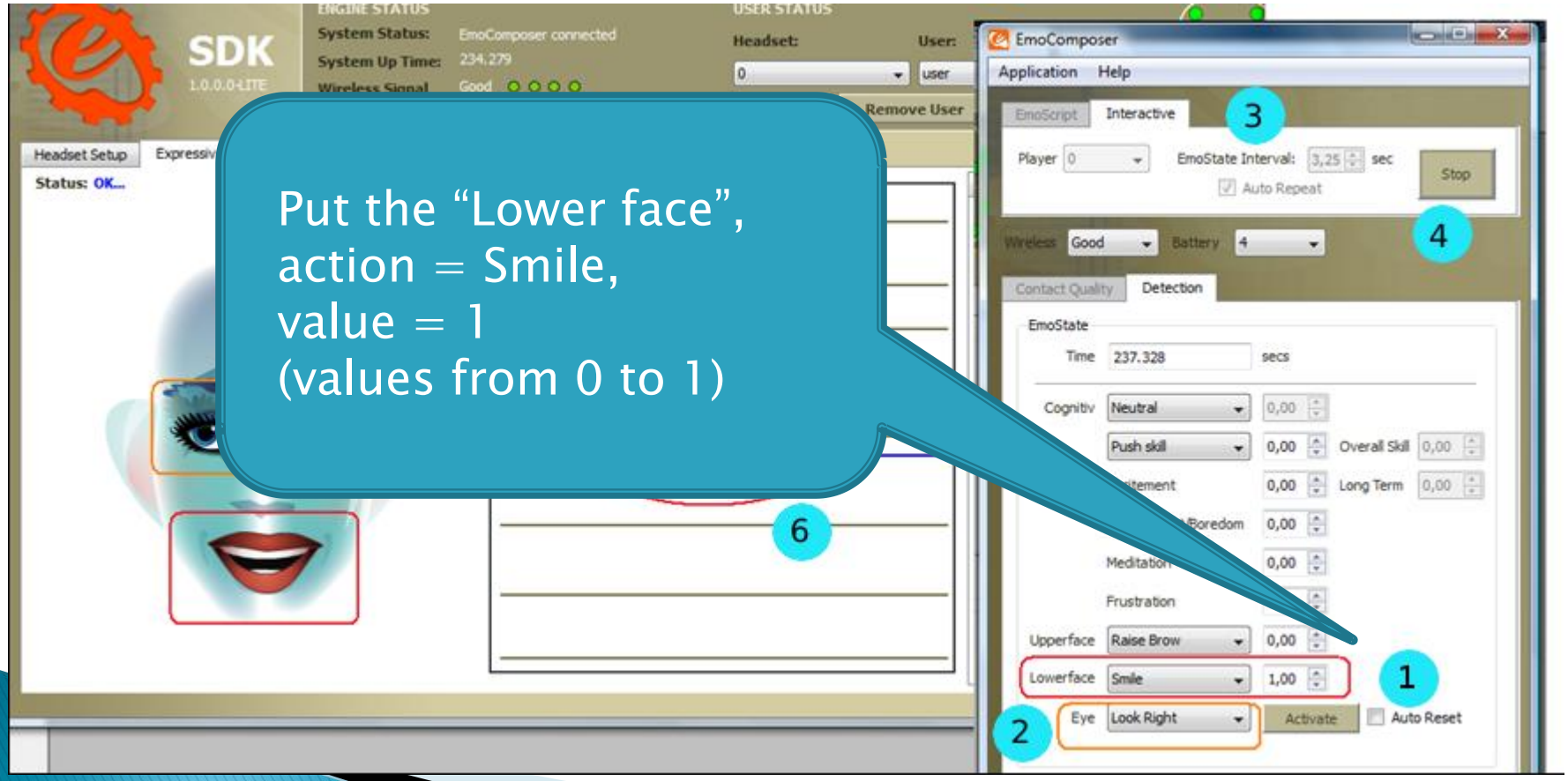
- Although the EmoEngine supports up to two real simultaneously connected neuroheadsets, Emotiv Control Panel only displays status information and detection results for a single neuroheadset at a time.
- Accurate detection results depend on good sensor contact and EEG signal quality. This display is a visual representation of the current contact quality of the individual neuroheadset sensors

Black	No signal
Red	Very poor signal
Orange	Poor signal
Yellow	Fair signal
Green	Good signal



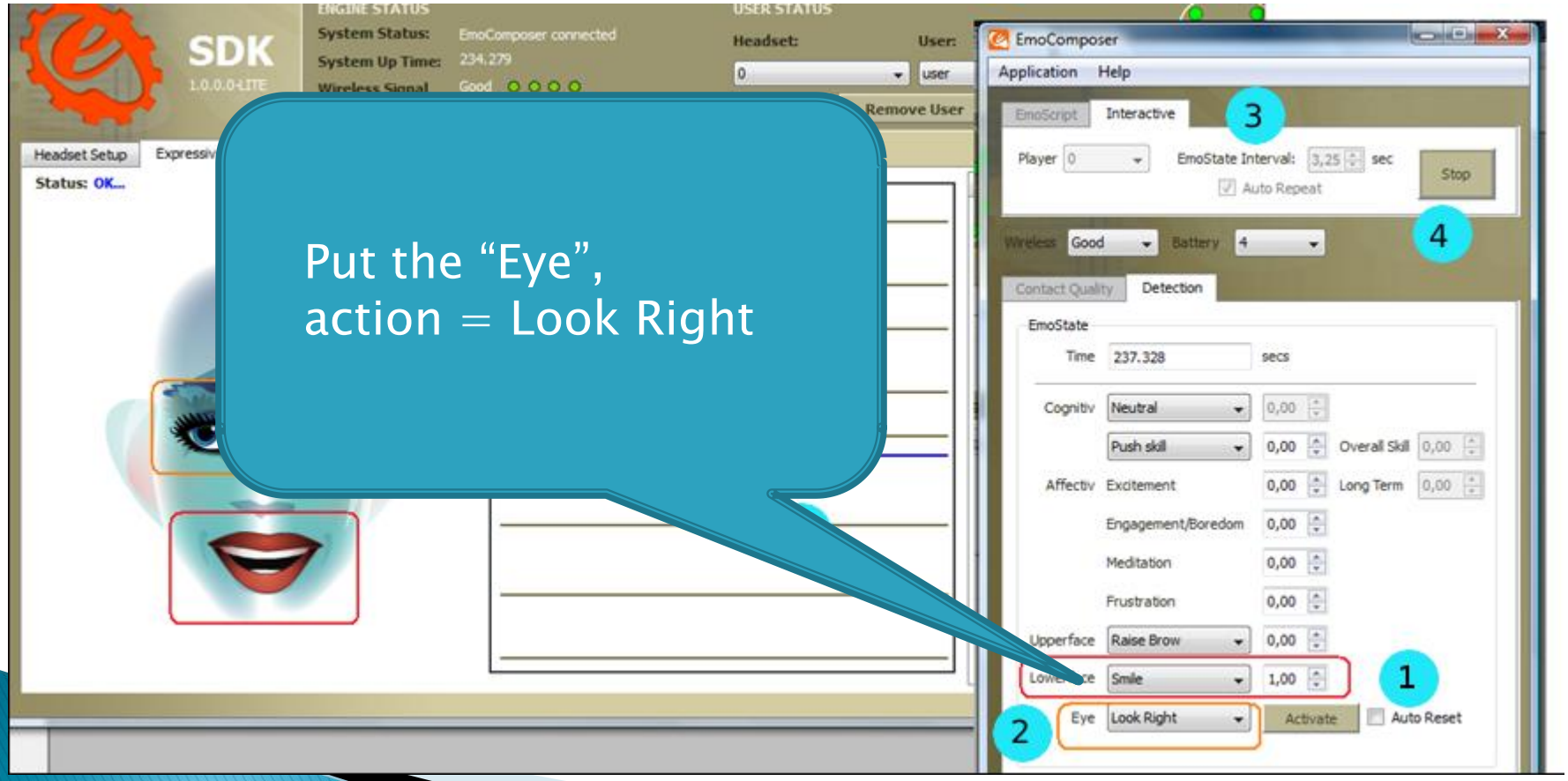
Parts of the API: Expressiv™ Suite

- One example: Look right and smile.



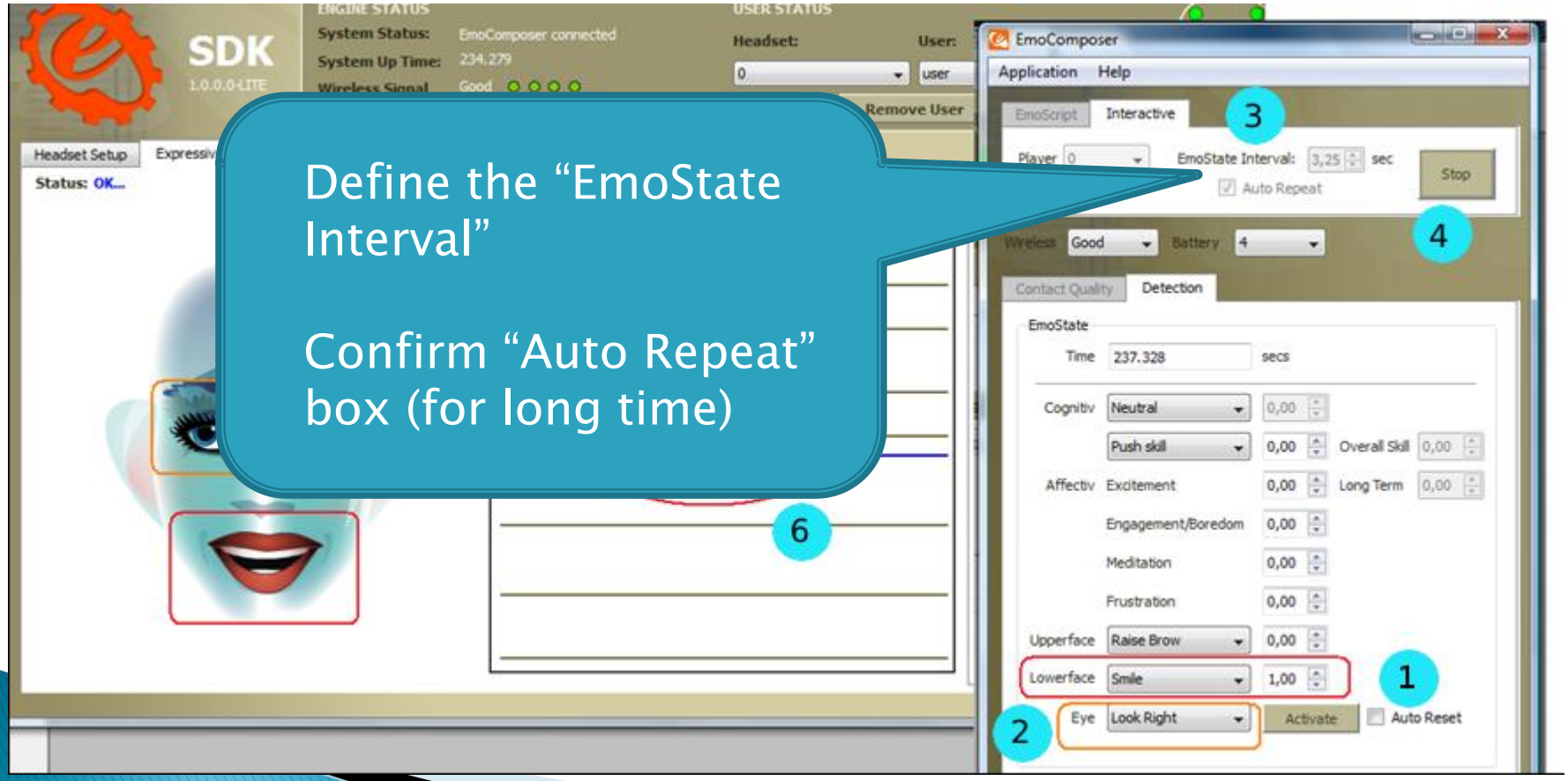
Parts of the API: Expressiv™ Suite

- One example: Look right and smile.



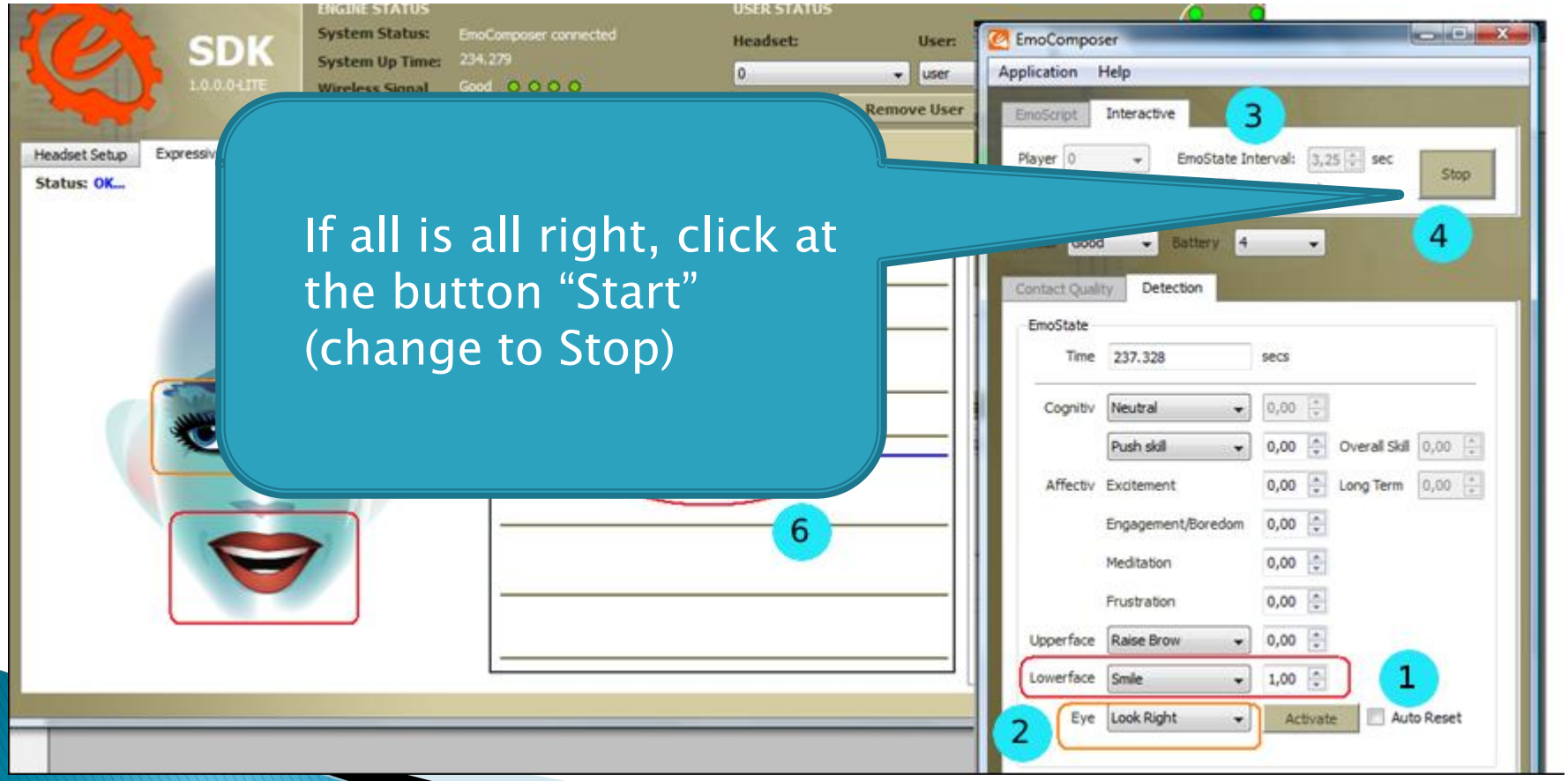
Parts of the API: Expressiv™ Suite

- One example: Look right and smile.



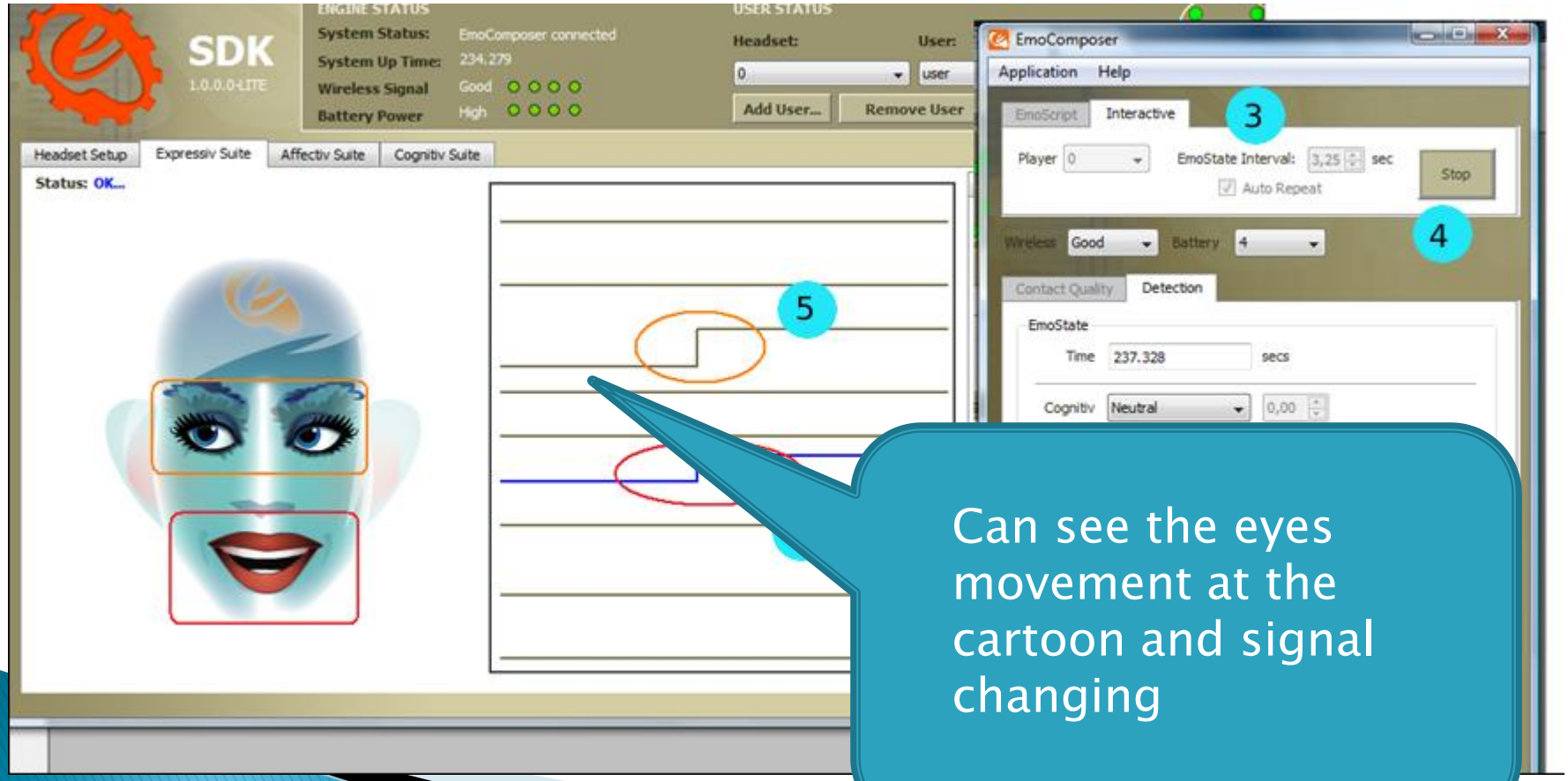
Parts of the API: Expressiv™ Suite

- One example: Look right and smile.



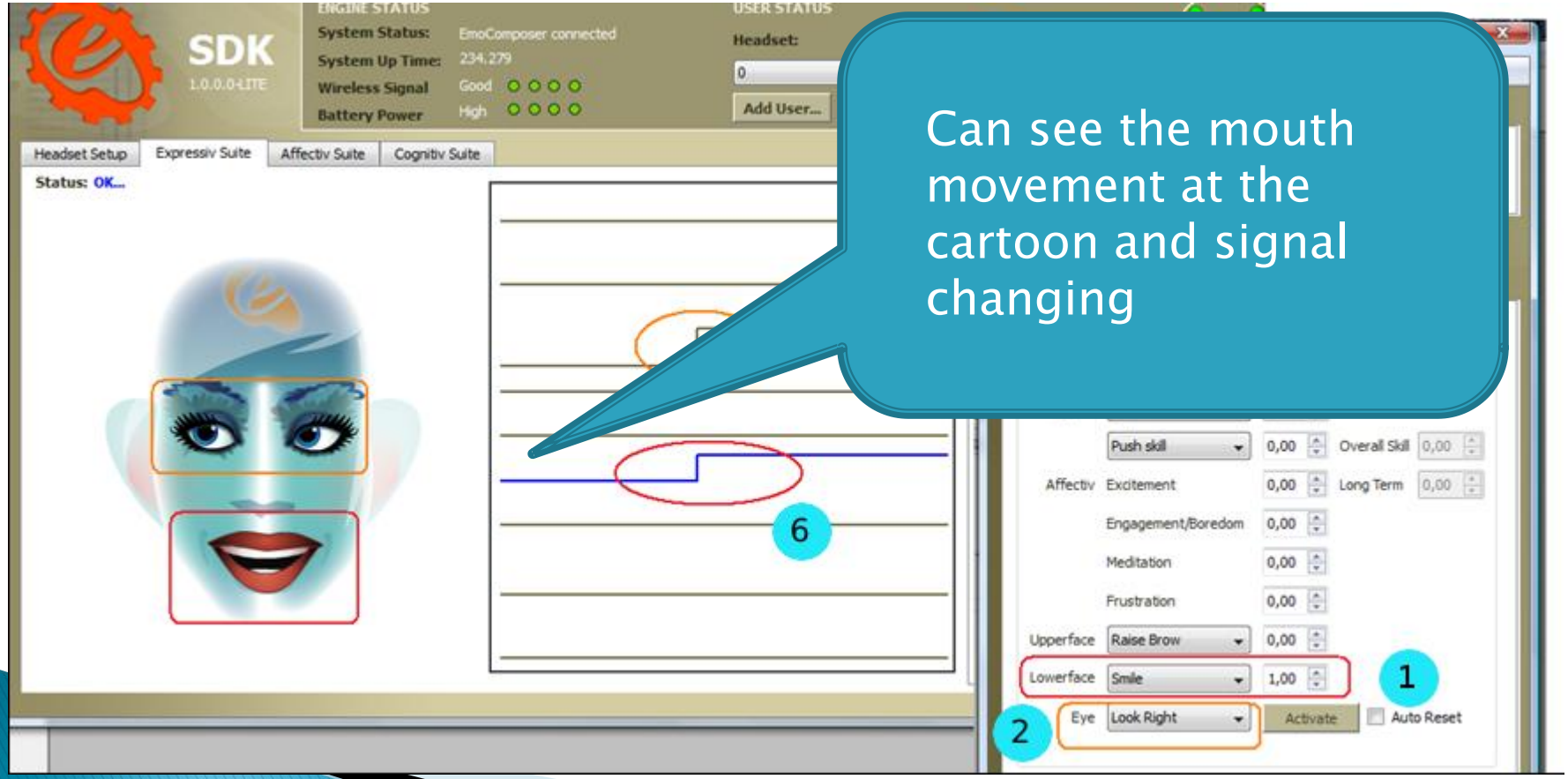
Parts of the API: Expressiv™ Suite

- One example: Look right and smile.



Parts of the API: Expressiv™ Suite

- One example: Look right and smile.



Parts of the API: Expressiv™ Suite

- Interpreting graphs:
  - *Blink*: low level = non-blink state / high level = a blink.
  - *Right Wink / Left Wink*: share a common graph line.

    Center level = no wink / low level = left wink /

    high level = right wink.
  - *Look Right / Left*: share a common graph line and a single sensitivity slider control.

    A center level = looking straight ahead / low level = looking left / high level = looking right.
  - *Raise Brow*: low level = no expression detected / high level = maximum level of expression detected.

    The graph level will increase or decrease depending on the level of expression detected.

Parts of the API: Expressiv™ Suite

- Interpreting graphs:
  - *Furrow Brow*: low level = no expression detected / high level = a maximum level of expression detected.

    The graph level will increase or decrease depending on the level of expression detected.
  - *Smile*: low level = no expression detected / high level = a maximum level of expression detected.

    The graph level will increase or decrease depending on the level of expression detected.
  - *Clench*: low level = no expression detected, high level = a maximum level of expression detected. The graph level will increase or decrease depending on the level of expression detected.

Parts of the API: Expressiv™ Suite

- Interpreting graphs:
  - *Right Smirk / Left Smirk*: share a common graph line.

    A center level = no smirk / low level = a left smirk / high level = a right smirk.
  - *Laugh*: low level = no expression detected / high level = a maximum level of expression detected.

    The graph level will increase or decrease depending on the level of expression detected.

Parts of the API: Expressiv™ Suite

- ***Sensitivity Adjustment Panel***:
  - This is controlled through sliders to the right of corresponding graph.
  - For each facial expression, check the performance of the detection. If you feel that the Expressiv detection is not responding readily to a particular expression, then increase the sensitivity for that expression.

Parts of the API: Expressiv™ Suite

- ***Training Panel***:
  - Requires the user to train the system by performing the desired action before it can be detected.
  - *Trained Signature*, the system will only detect actions for which the user has supplied training data.
  - **Not all** Expressiv expressions can be trained.
    In particular, eye and eyelid-related expressions (i.e. “blink”, “wink”, “look left”, and “look right”) can not be trained and always rely on the *Universal Signature*.

Parts of the API: Affectiv™ Suite

- Reports real time changes in the subjective emotions experienced by the user.

- Offers 5 distinct Affectiv detections:

  Engagement/Boredom
  +Frustation
  Instantaneous Excitement
  +Meditation
  Long-Term Excitement

- That characteristics are universal in nature and don't require an explicit training or signature-building step on the part of the user.
- It is very important that a new user profile is selected when a new user puts on the neuroheadset.

Parts of the API: Affectiv™ Suite

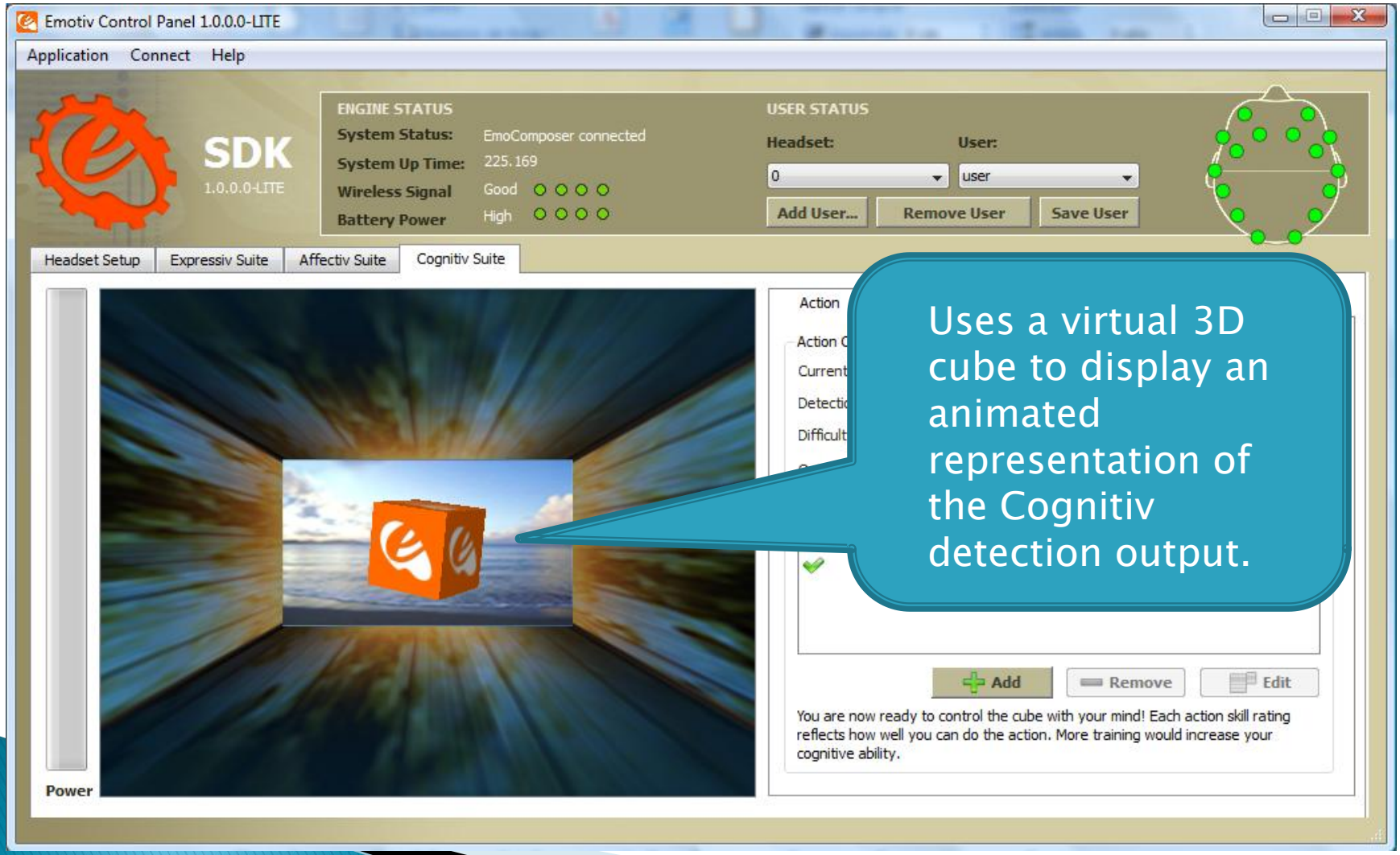
Parts of the API: Affectiv™ Suite

- *Detection Details:*
  - *Excitement* is characterized by activation in the sympathetic nervous system which results in a range of physiological responses including pupil dilation, eye widening, sweat gland stimulation, heart rate and muscle tension increases, blood diversion, and digestive inhibition.
  - *Engagement* is characterized by increased physiological arousal and **beta** waves along with attenuated **alpha** waves. The opposite pole of this detection is referred to as "*Boredom*"

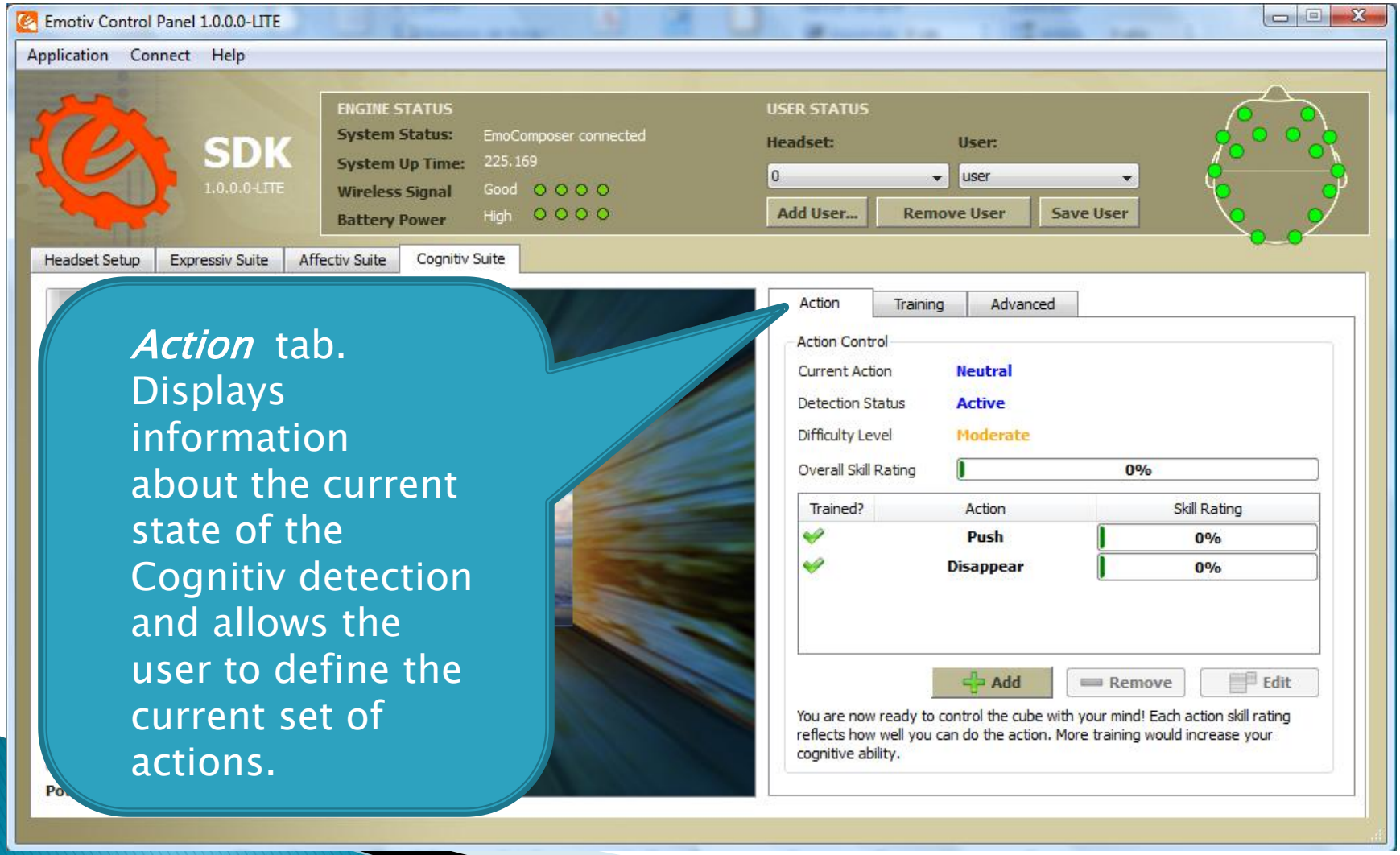
Parts of the API: Cognitiv™ Suite

- The Cognitiv detection suite evaluates a user's real time brainwave activity to discern the user's conscious intent to perform distinct physical actions on a real or virtual object.
- Is designed to work with up to 13 different actions: 6 directional movements (push, pull, left, right, up and down) and 6 rotations (clockwise, counter-clockwise, left, right, forward and backward) plus one additional action that exists only in the realm of the user's imagination: disappear.
- NOTE: allows the user to choose ONLY up to 4 actions that can be recognized at any given time.

Parts of the API: Cognitiv™ Suite

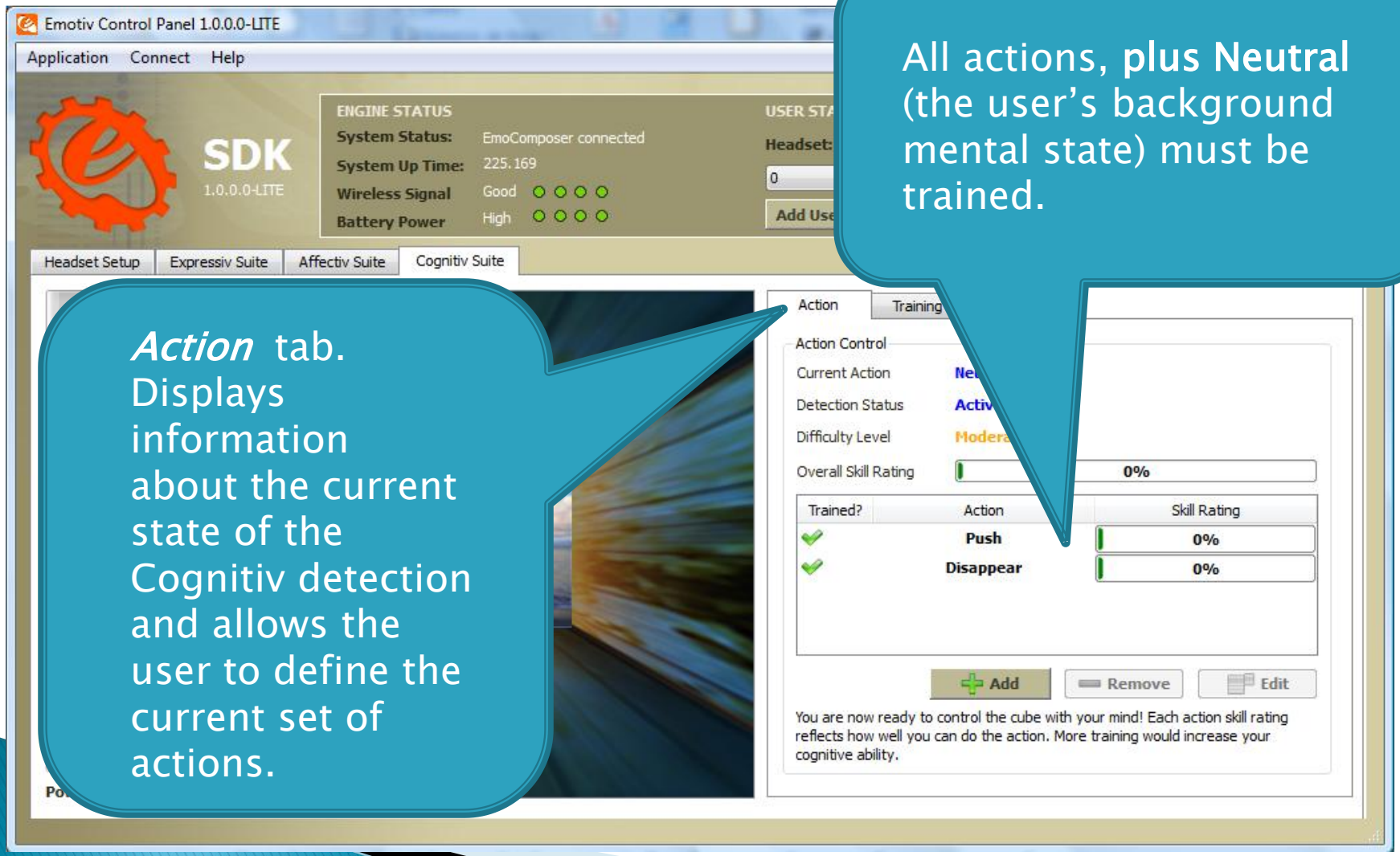


Parts of the API: Cognitiv™ Suite



Action tab. Displays information about the current state of the Cognitiv detection and allows the user to define the current set of actions.

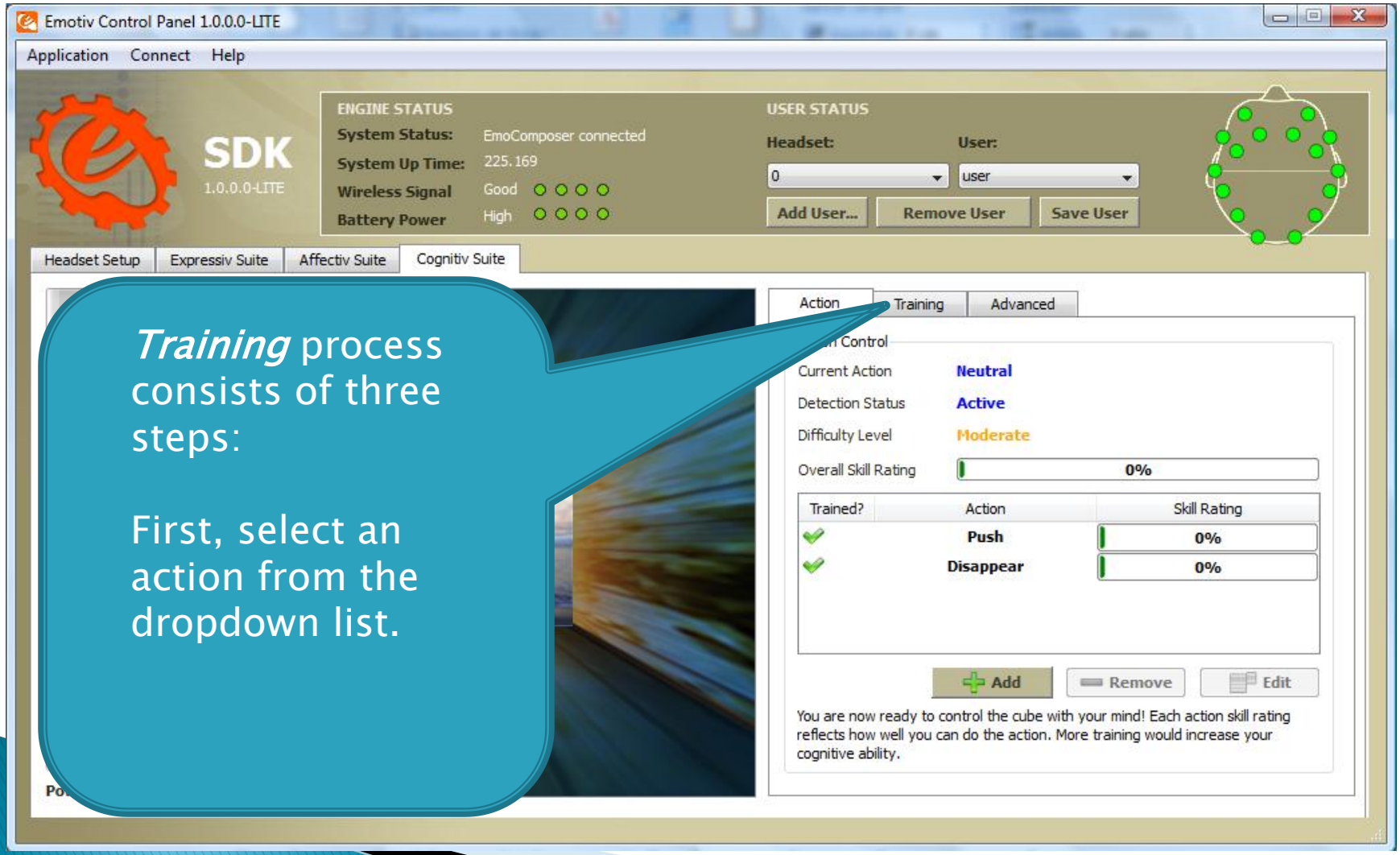
Parts of the API: Cognitiv™ Suite



All actions, **plus Neutral** (the user's background mental state) must be trained.

Action tab. Displays information about the current state of the Cognitiv detection and allows the user to define the current set of actions.

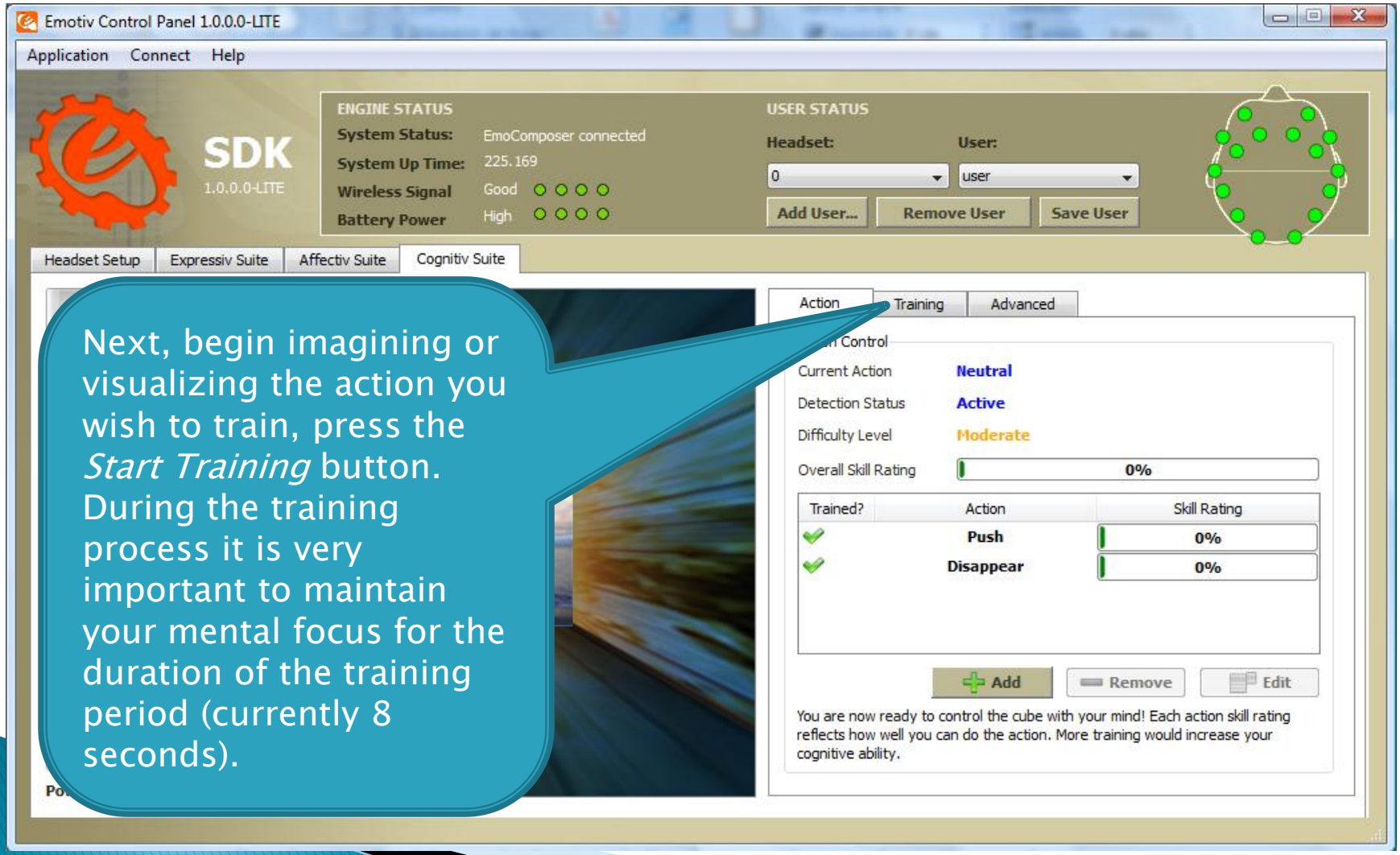
Parts of the API: Cognitiv™ Suite



Training process consists of three steps:

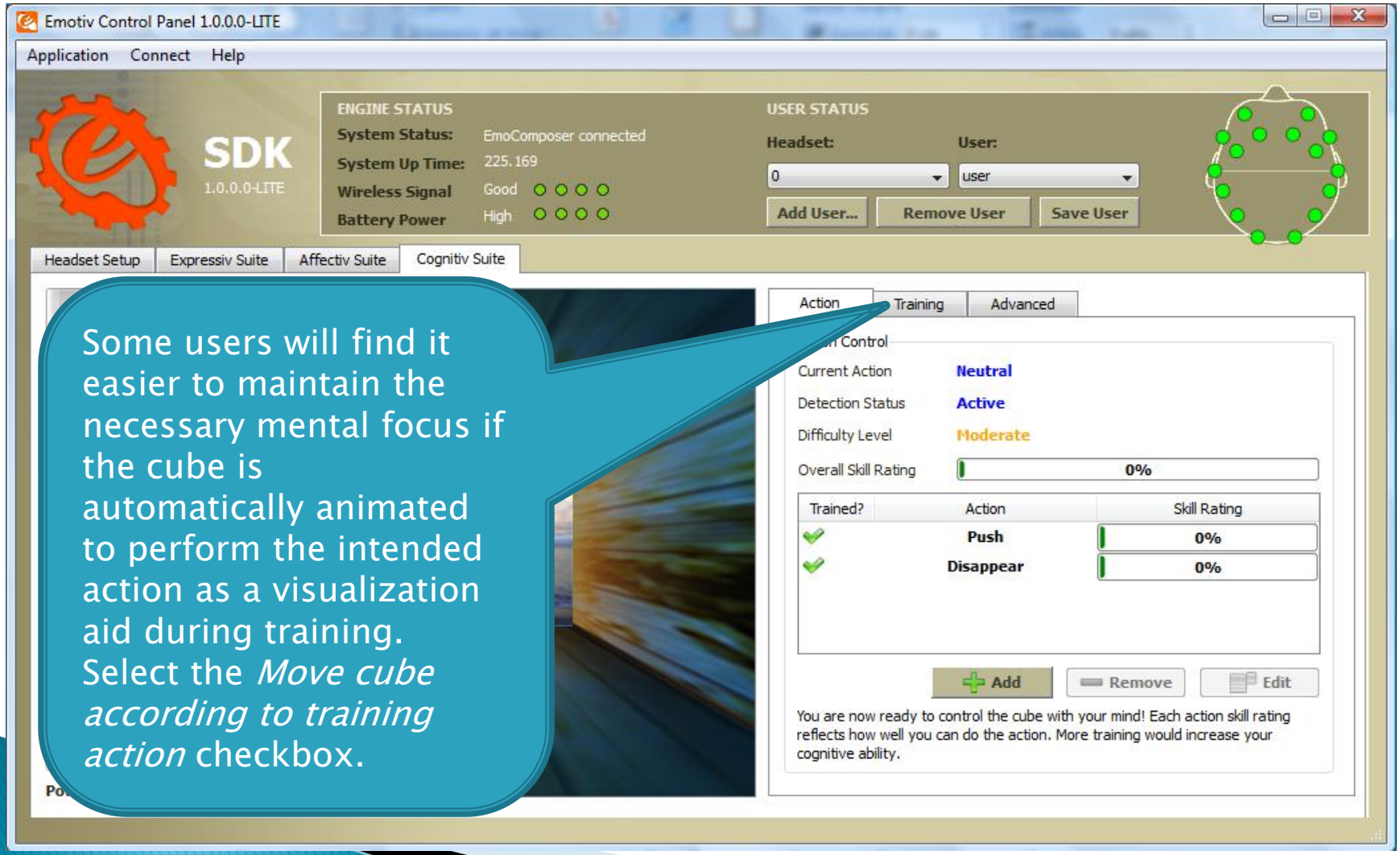
First, select an action from the dropdown list.

Parts of the API: Cognitiv™ Suite



Next, begin imagining or visualizing the action you wish to train, press the *Start Training* button. During the training process it is very important to maintain your mental focus for the duration of the training period (currently 8 seconds).

Parts of the API: Cognitiv™ Suite



Some users will find it easier to maintain the necessary mental focus if the cube is automatically animated to perform the intended action as a visualization aid during training. Select the *Move cube according to training action* checkbox.

Parts of the API: Cognitiv™ Suite

- *Training Neutral:*

- The Neutral "action" refers to the user's passive mental state; one that isn't associated with any of the selected Cognitiv actions.
- Typically this means engaging in passive mental activities such as reading or just relaxing.

- ***Advanced*** tab: It is strongly recommended that you only change these settings with the guidance of Emotiv personnel

Parts of the API: Cognitiv™ Suite

- *Tips:*
  - Most users typically achieve their best results after training each action several times.
  - Overtraining can sometimes produce a decrease in accuracy, this may also indicate a lack of consistency and mental fatigue.
  - If it becomes hard for you to return to neutral, try refreshing your mental state by momentarily shifting your focus away from the screen and relaxing.

Emotiv SDK Tools: EmoKey™

- Translates Emotiv detection results to predefined sequences of keystrokes according to logical rules defined by the user through the EmoKey user interface.
- Can be saved for later reuse.
- Communicates with Emotiv EmoEngine in the same manner as would a third-party application: by using the Emotiv API exposed by edk.dll.

Emotiv SDK Tools: EmoKey™

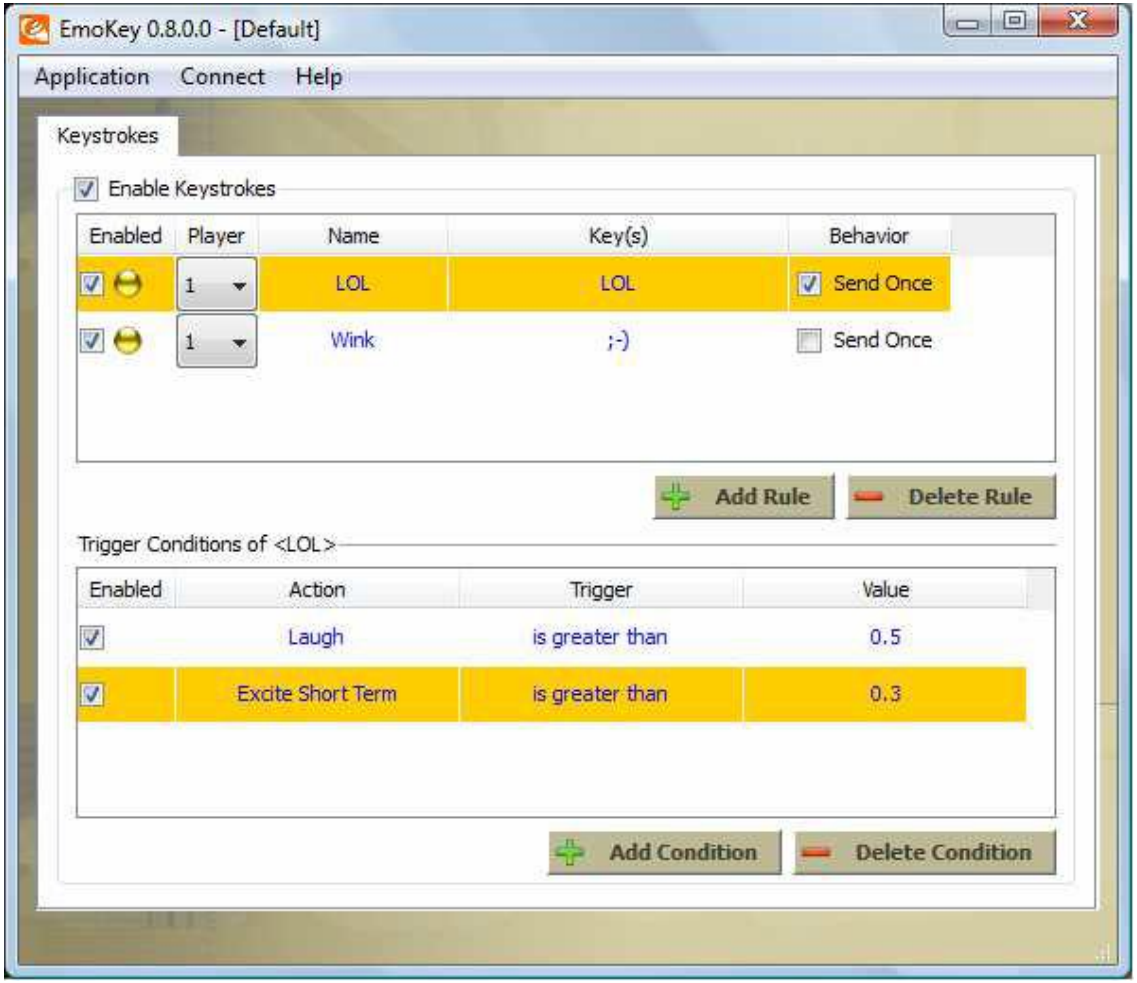
- *Connecting EmoKey to Emotiv EmoEngine:*

- By default, EmoKey will attempt to connect to Emotiv Control Panel when the application launches.

- Can also be connected to EmoComposer. Useful when creating and testing a new EmoKey Mapping.

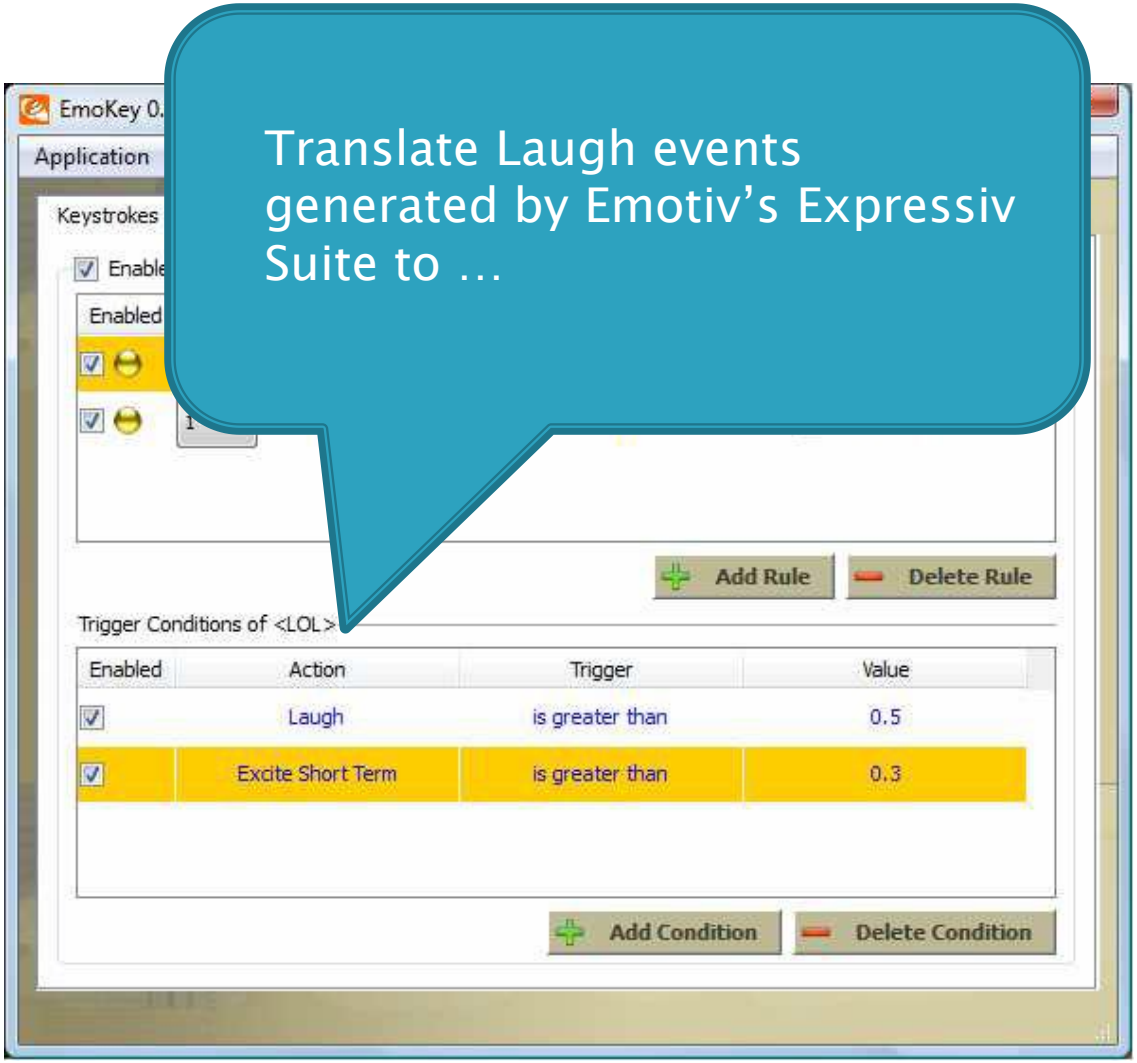
Emotiv SDK Tools: EmoKey™

- *Configuring EmoKey Rules:*



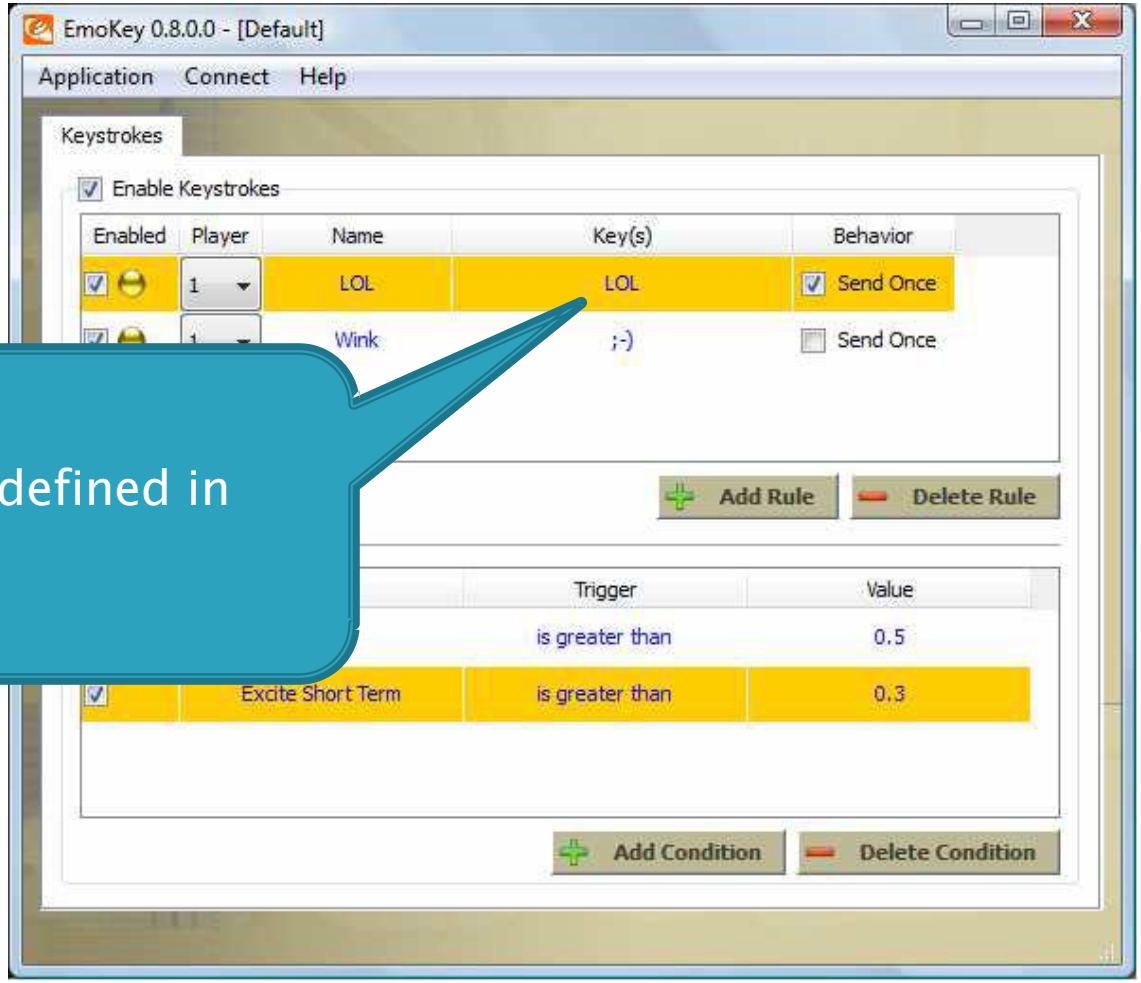
Emotiv SDK Tools: EmoKey™

- *Configuring EmoKey Rules:*



Emotiv SDK Tools: EmoKey™

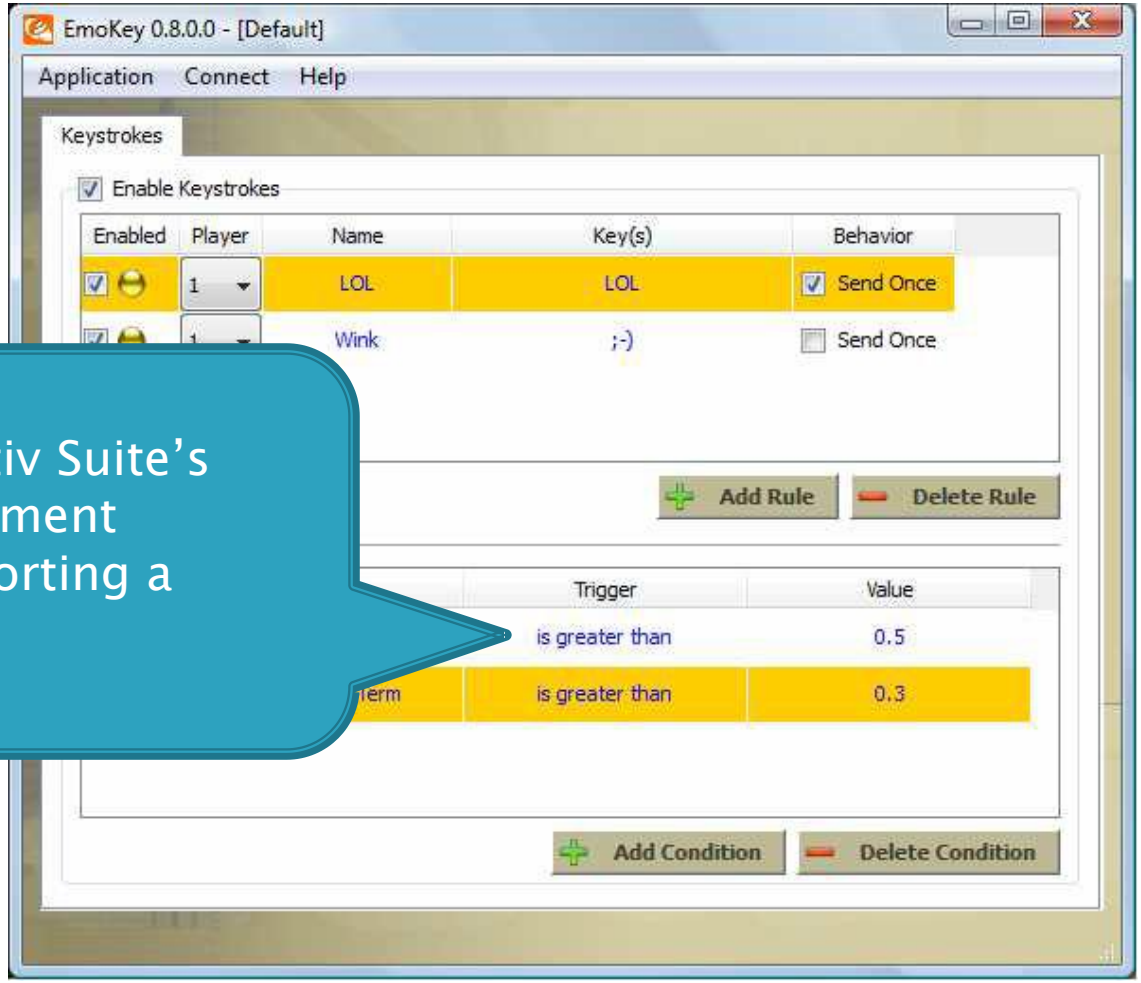
- ***Configuring EmoKey Rules:***



… to the text “LOL”, defined in “Key(s)”

Emotiv SDK Tools: EmoKey™

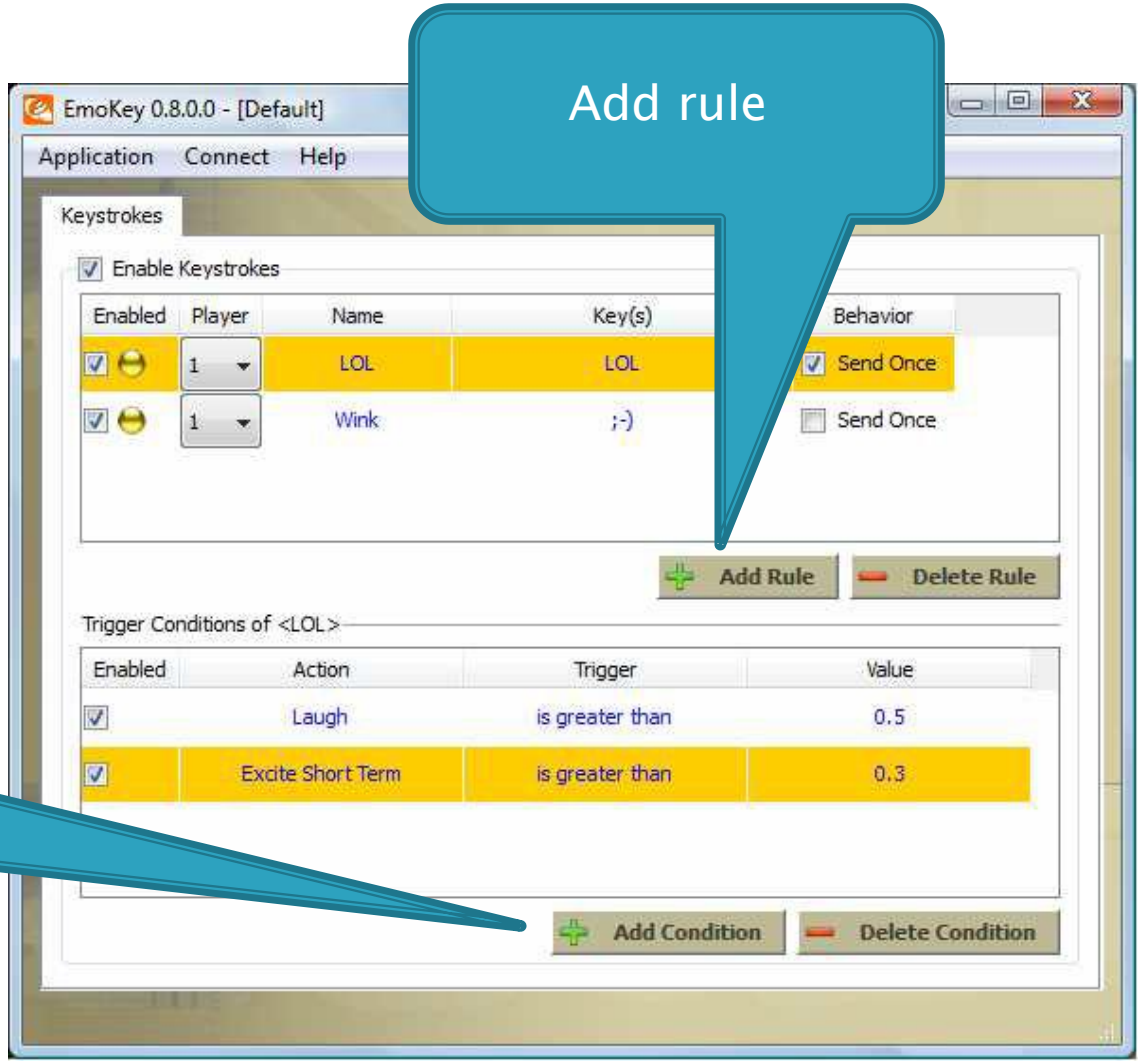
- ***Configuring EmoKey Rules:***



As long as the Affectiv Suite's Instantaneous Excitement detection is also reporting a score > 0.5

Emotiv SDK Tools: EmoKey™

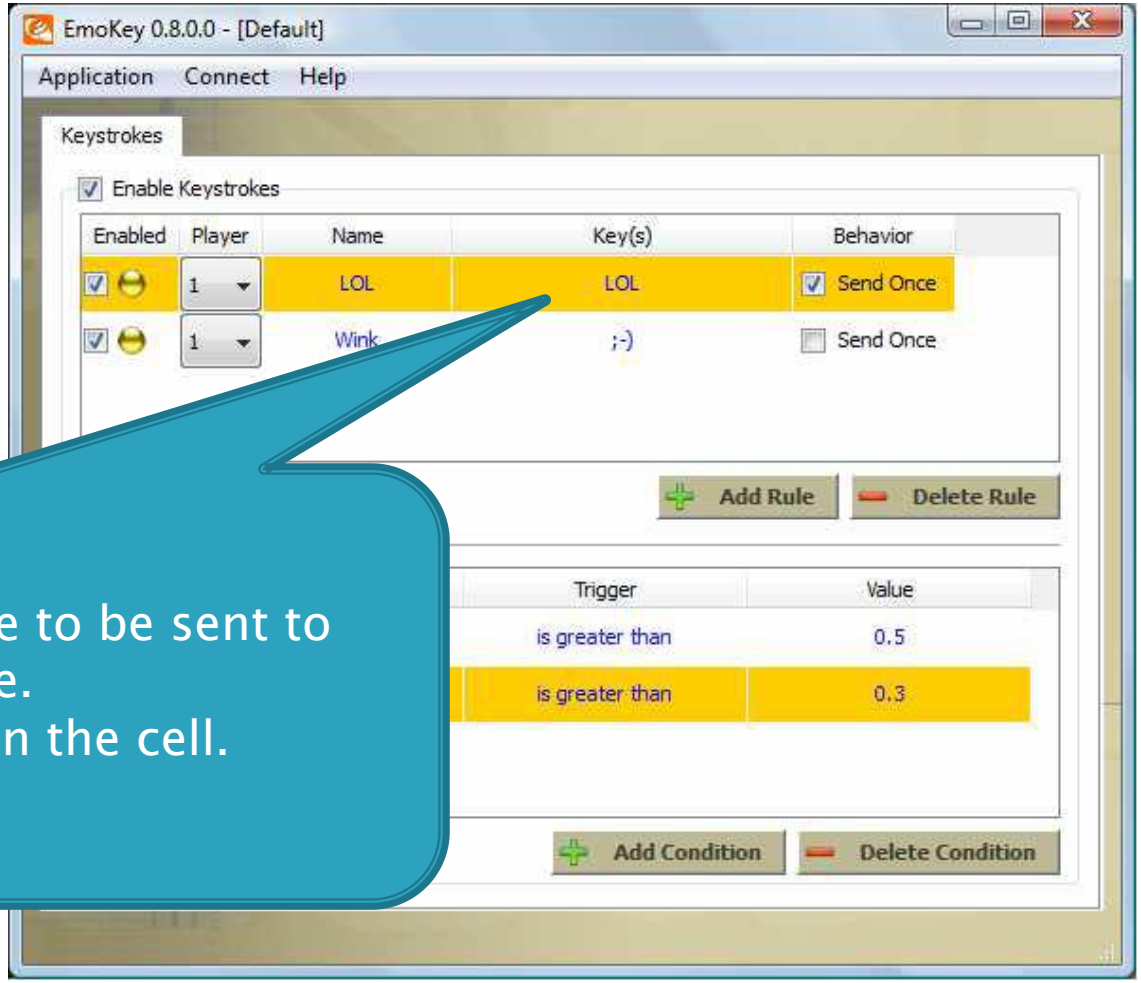
- ***Configuring EmoKey Rules:***



Add trigger conditions to the selected rule

Emotiv SDK Tools: EmoKey™

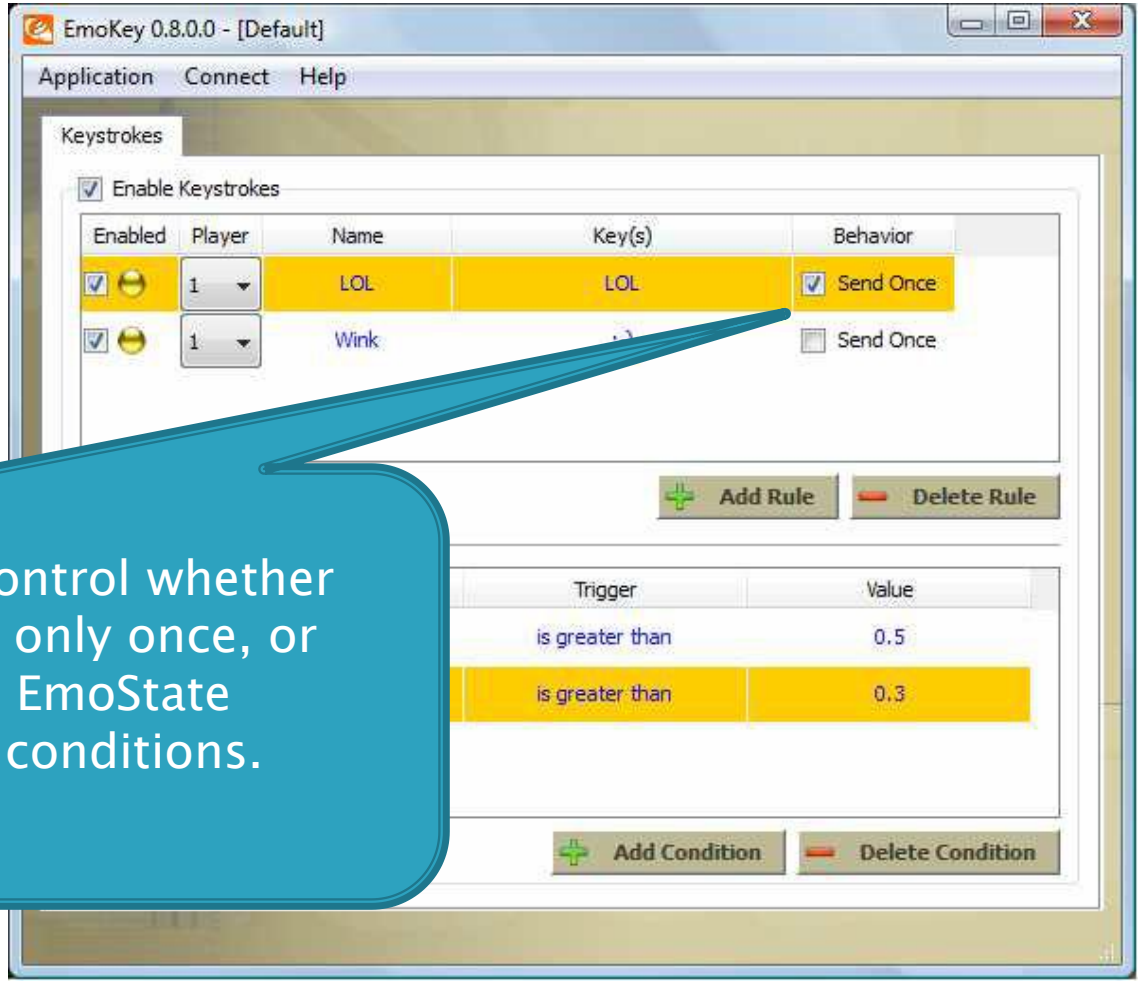
- *Configuring EmoKey Rules:*



Key , Keystroke sequence to be sent to the Windows input queue.
Edit by double clicking on the cell.

Emotiv SDK Tools: EmoKey™

- ***Configuring EmoKey Rules:***



Behavior Checkbox, to control whether the key sequence is sent only once, or repeatedly, each time an EmoState update satisfies the rule conditions.

Emotiv SDK Tools: EmoKey™

- ***Configuring EmoKey Rules:***
  - Emulates a Windows-compatible keyboard and sends keyboard input to the Windows operating system's input queue.
  - The application with the input focus will receive the emulated keystrokes or other window you define.
  - EmoKey is run in the background.

Emotiv SDK Tools: EmoKey™

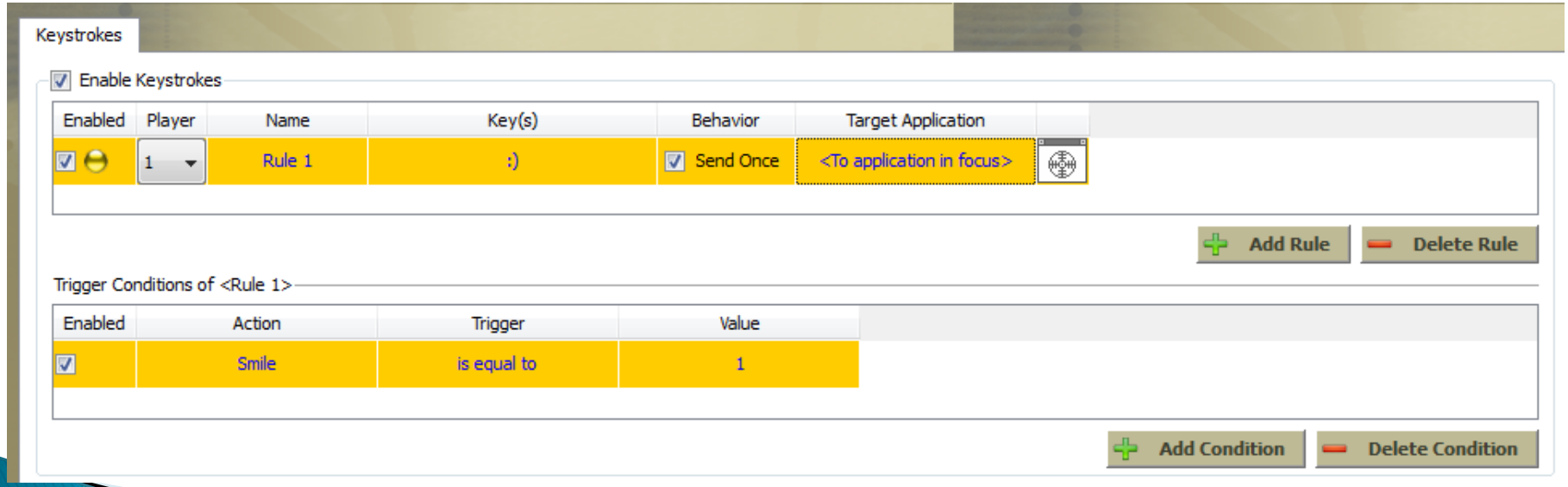
- *Configuring EmoKey Rules:*

Hot keys or special keyboard keys: any combination of these, and another keystroke.

Key press duration and delay times: some applications, especially games, are sensitive to the timing of key presses.

Emotiv SDK Tools: EmoKey™

- ***Saving Rules to an EmoKey Mapping file:***
  - EmoKey allows you to save the current set of rule definitions to an EmoKey Mapping file.
  - Example:



Emotiv SDK Tools: EmoKey™

- rule1.ekm:

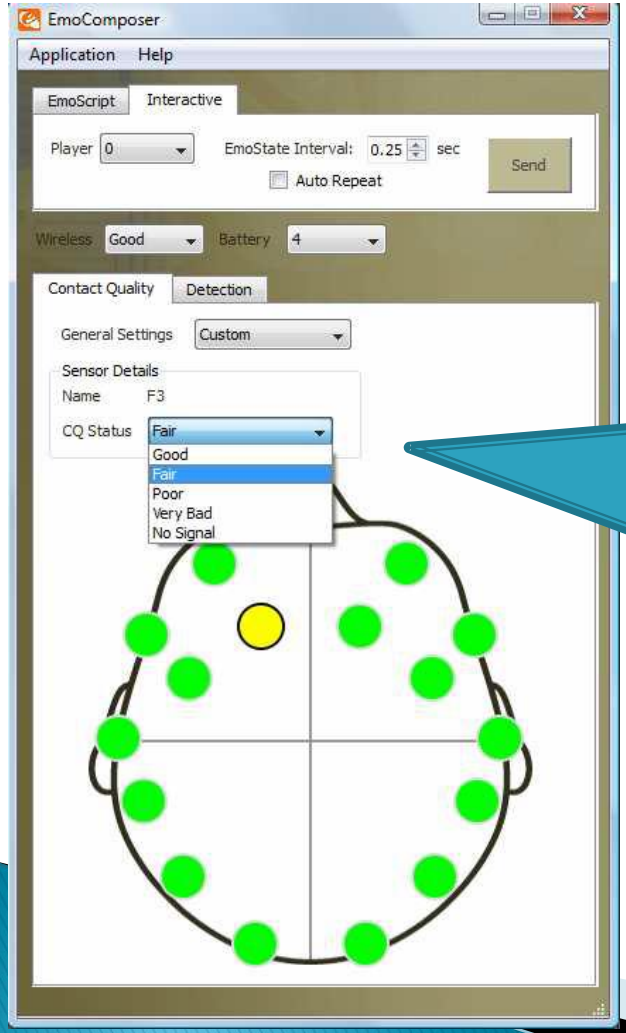
```
<EmoMappingModel mappingName="rule1.ekm" > NAME
   <EmoMapping enabled="1" name="Rule 1" keys=":)"
   sendOnce="1" sendToFocus="1" targetApp=""(WHICH
   APPLICATION WINDOW) sendHotKeys="0" ctrlKey="0" altKey="0"
   shiftKey="0" winKey="0"  hotKey="0" holdKey="0"
   holdTime="20" triggerDelayTime="20" (VARIABLES)player="0">
      <EmoStateCondition> (TRIGGER)
         <enabled value="1" />
         <action value="7" />(7=Smile)
         <threshold type="double" value="1" />
         <actionRule value="0" />(0=is equal to)
      </EmoStateCondition>
   </EmoMapping>
</EmoMappingModel>
```

Emotiv SDK Tools: EmoComposer™ usage

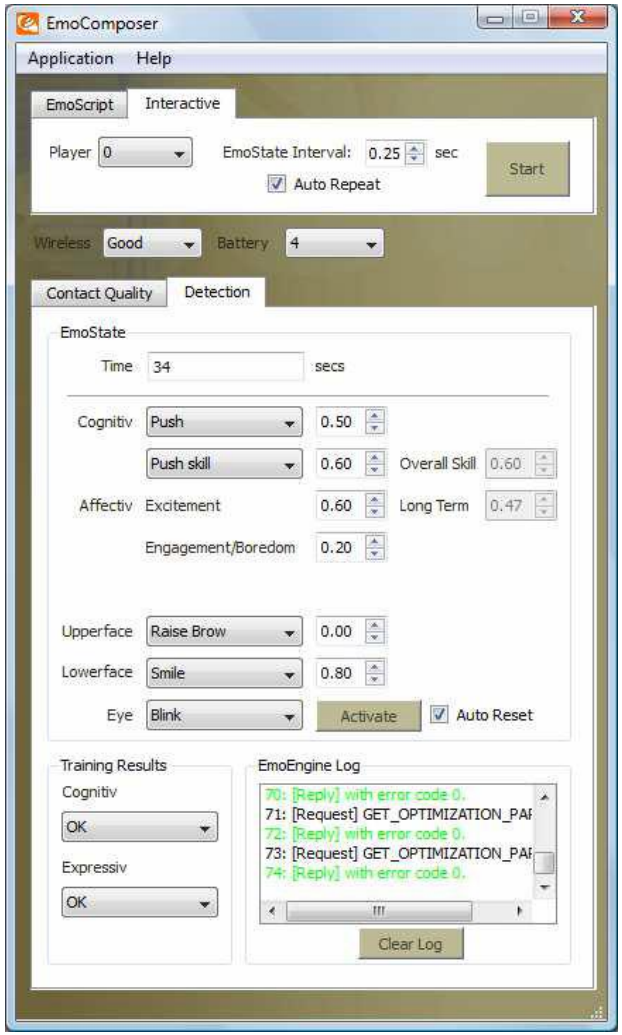
- Allows you to send user-defined EmoStates™ to Emotiv Control Panel, EmoKey, or any other application that makes use of the Emotiv API.
- Two modes:
  - Interactive mode
  - EmoScript mode
- SDKLite users will rely on EmoComposer to simulate the behavior of Emotiv EmoEngine and Emotiv neuroheadsets.

Emotiv SDK Tools: EmoComposer™ usage

- *Interactive mode:*

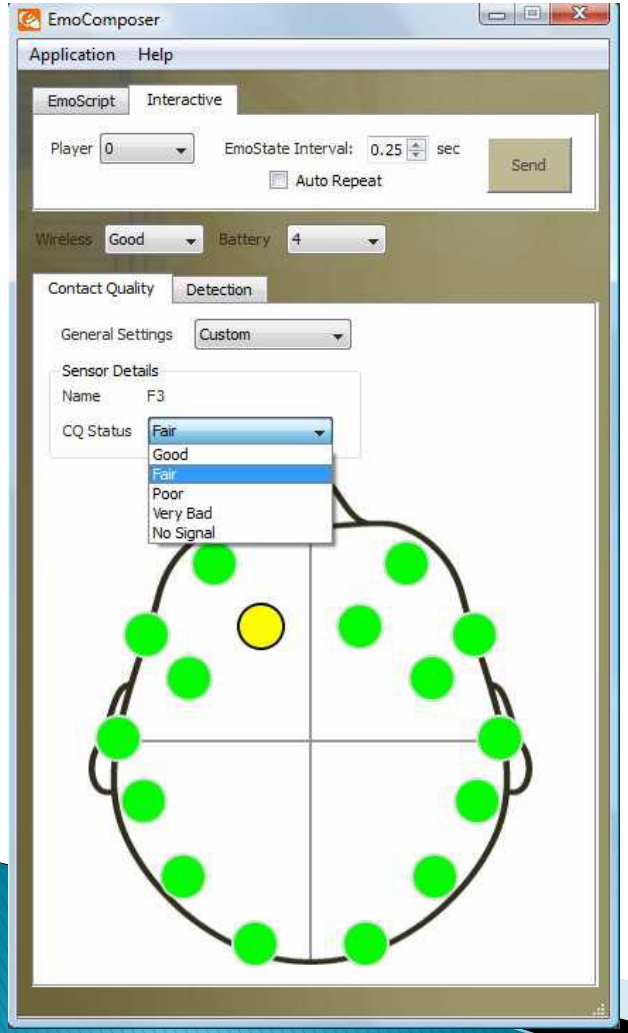


Reported contact quality for each sensor

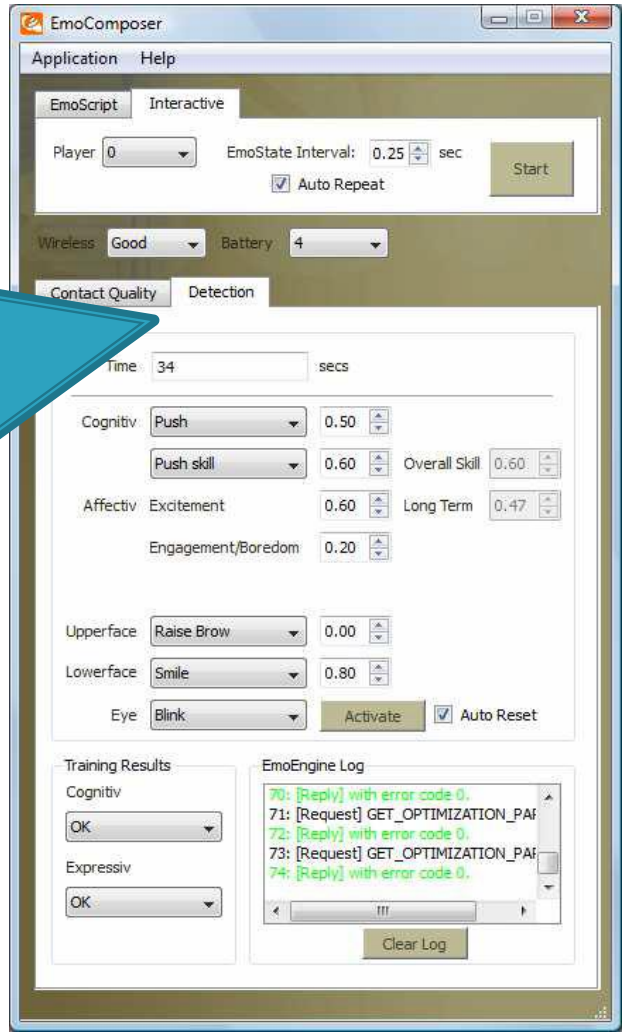


Emotiv SDK Tools: EmoComposer™ usage

- *Interactive mode:*

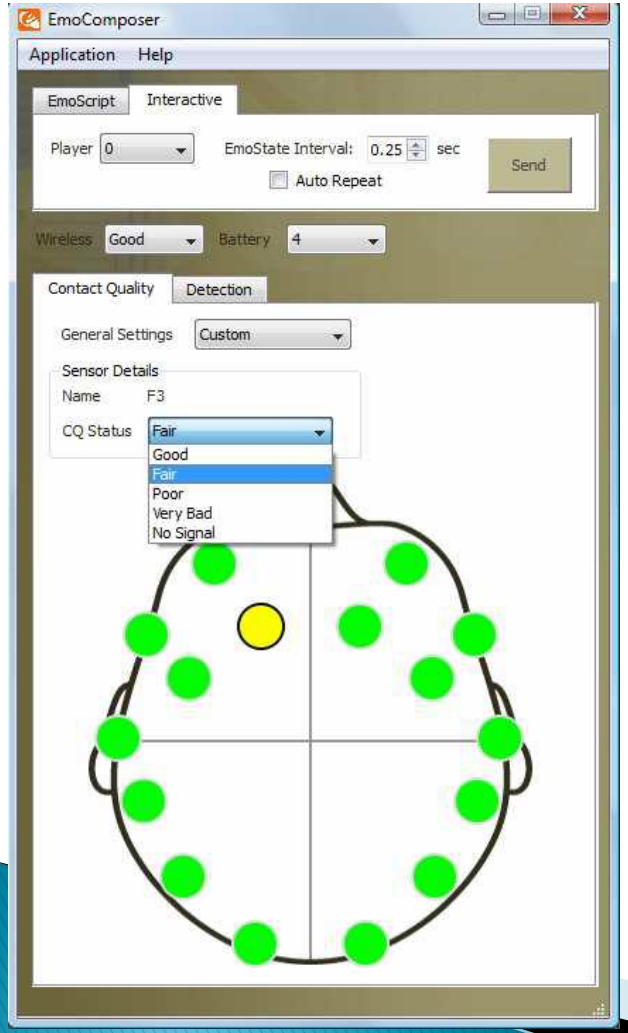


Interactively control EmoState™ detection values and training result values.

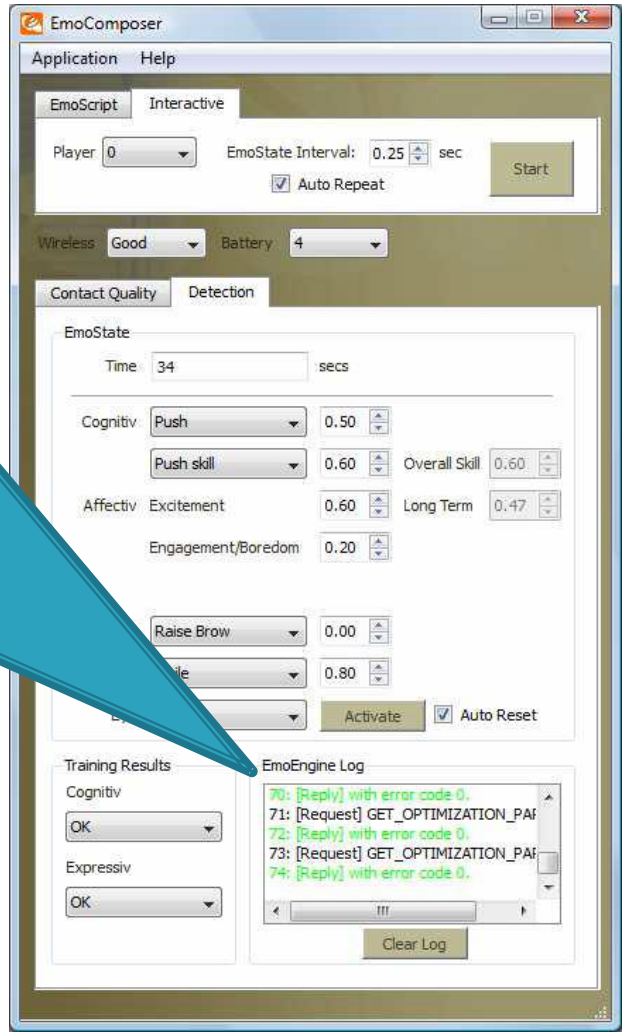


Emotiv SDK Tools: EmoComposer™ usage

- *Interactive mode:*

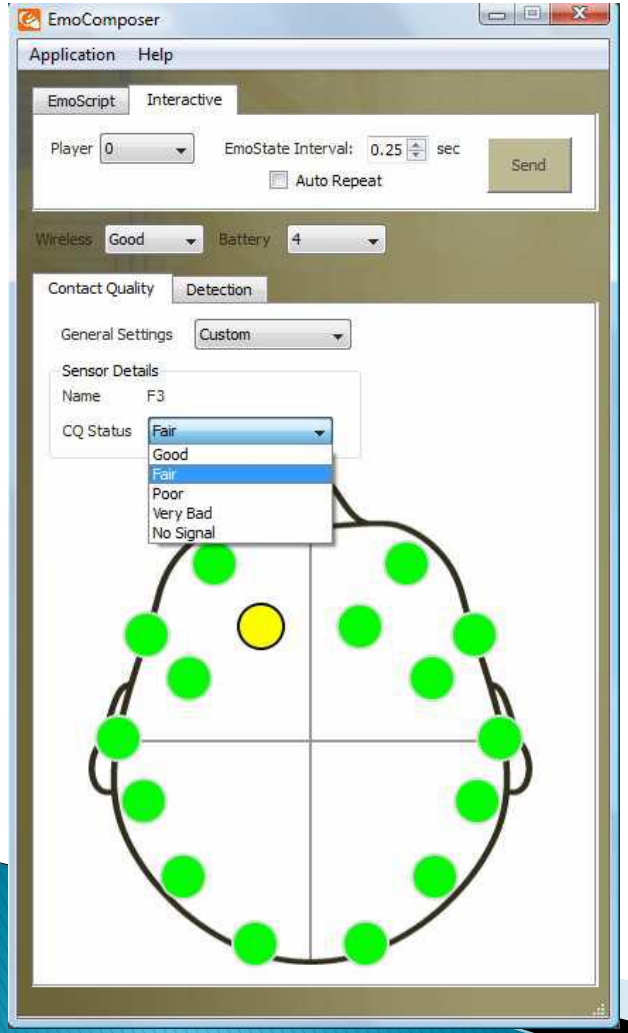


Give developers a clearer picture about how processes requests generated by various Emotiv API functions.

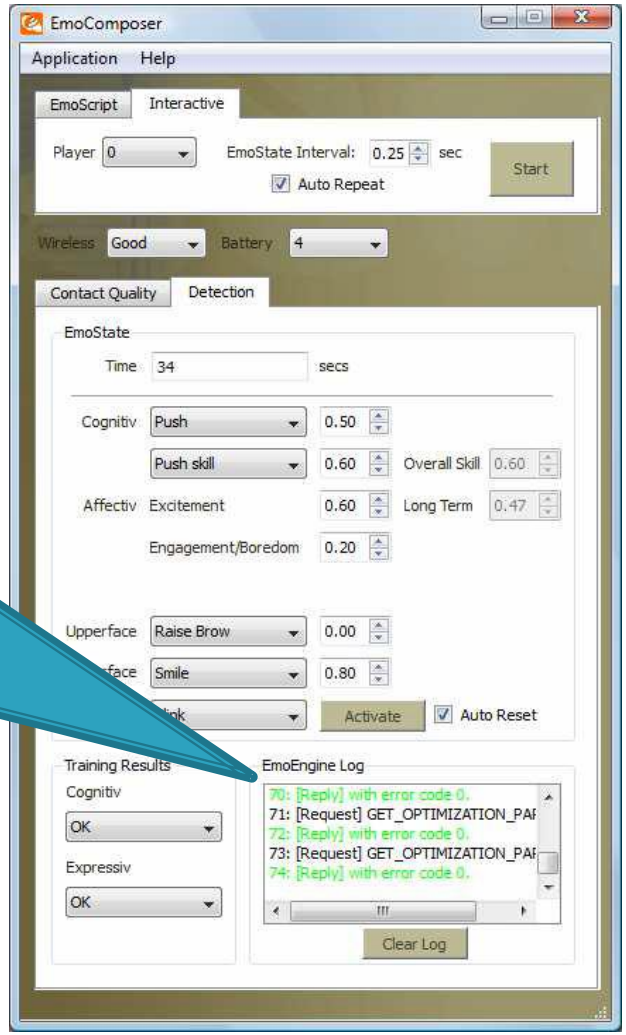


Emotiv SDK Tools: EmoComposer™ usage

- *Interactive mode:*



3 different output types:
- Request
- Reply
- CogResult or ExpResult



Emotiv SDK Tools: EmoComposer™ usage

- The multitude of API functions are translated to roughly a dozen different strings intended to allow the Emotiv SDK developer to see that an API function call has been serviced.
- These strings include:

  PROFILE_ADD_USER, PROFILE_CHANGE_USER, PROFILE_REMOVE_USER, PROFILE_LIST_USER, PROFILE_GET_CURRENT_USER, PROFILE_LOAD, PROFILE_SAVE, EXPRESSIV_GET, EXPRESSIV_SET, AFFECTIV_GET, AFFECTIV_SET, COGNITIV_SET and COGNITIV_GET.

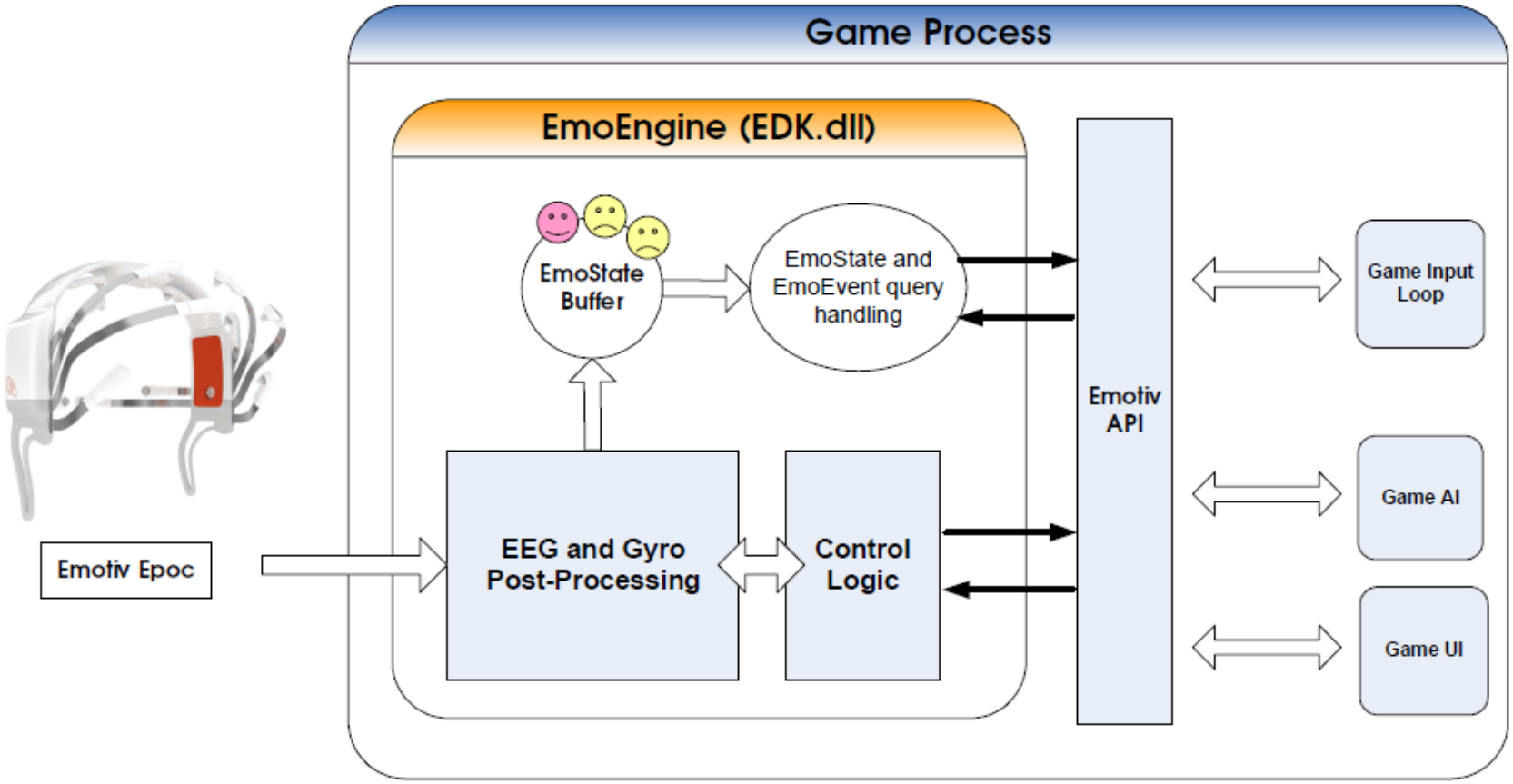
Emotiv SDK Tools: EmoComposer™ usage

- ***EmoScript Mode:***
  - EmoScript files are written in EML (EmoComposer™ Markup Language). EML documents are XML documents that can be interpreted by EmoComposer.
  - Note that these EmoScript values are not interactive and can not be modified by the user (use the Interactive mode for this instead).

Programming with the Emotiv SDK

- Program in C++ and compiled with Microsoft Visual Studio 2005 (Visual Studio 2008 is also supported).
- The Emotiv API is exposed as an ANSI C interface that is declared in 3 header files (edk.h, EmoStateDLL.h, edkErrorCode.h) and implemented in 2 Windows DLLs (edk.dll and edk_utils.dll).
- Applications simply include edk.h and link with edk.dll.
- Emotiv API functions that modify or retrieve EmoEngine settings are prefixed with “EE_.”
- An EmoState is an opaque data structure that contains the current state of the Emotiv detections, which, in turn, reflect the user’s facial, emotional and cognitive state. (“ES_.”)

Programming with the Emotiv SDK



Programming with the Emotiv SDK

Must establish a connection to the EmoEngine by calling EE_EngineConnect or *EE_EngineRemoteConnect*.

Events that can be retrieved by calling *EE_EngineGetNextEvent().* For near real-time responsiveness should poll for new EmoStates at least 10-15 times per second.

Before your application terminates, the connection to EmoEngine should be explicitly closed by calling *EE_EngineDisconnect().*

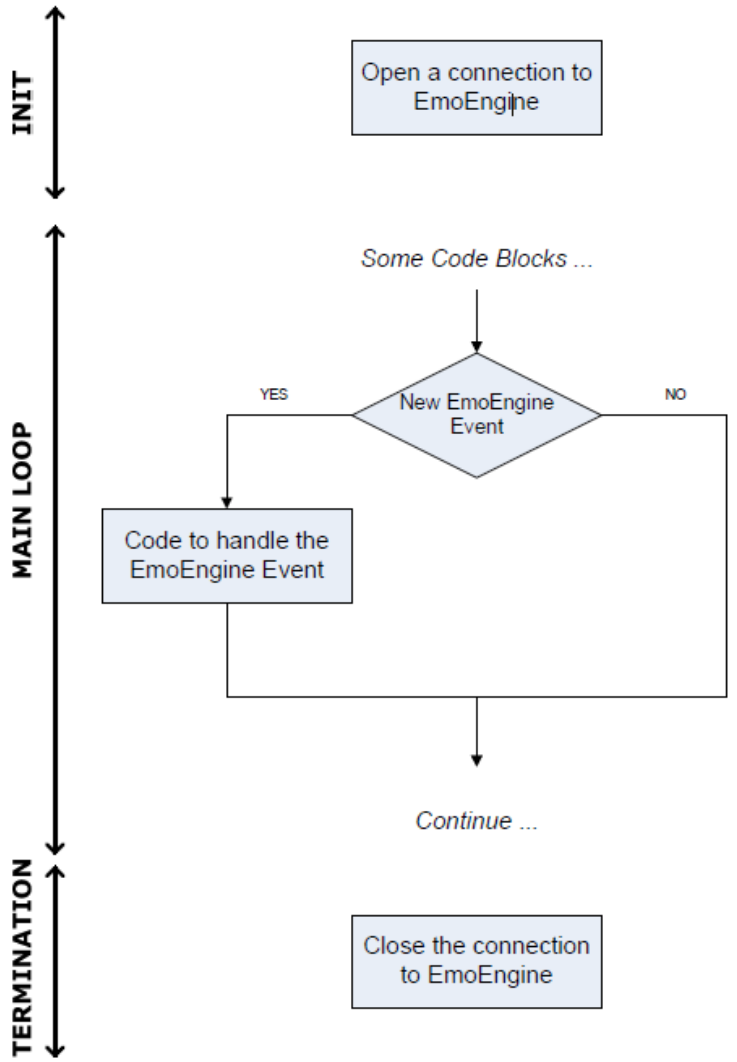


Figure 21 *Using the API to communicate with the EmoEngine*

Programming with the Emotiv SDK

- 3 main categories of EmoEngine events:
  - Hardware-related events: when users connect or disconnect Emotiv input devices to the computer (e.g. EE_UserAdded).
  - New EmoState events: changes in the user's facial, cognitive and emotional state. retrieve the updated EmoState by calling EE_EmoEngineEventGetEmoState(). (e.g. EE_EmoStateUpdated).
  - Suite-specific events: training and configuring the Cognitiv and Expressiv detection suites (e.g. EE_CognitivEvent).

- NOTE: A complete list of all EmoEngine events can be found in Appendix 3 (User Manual)

Programming with the Emotiv SDK

- **Scenarios Supported by EE_EngineRemoteConnect**: (in place of EE_EngineConnect())
- Developing with Emotiv SDKLite; not include an Emotiv headset so all Emotiv API function calls communicate with EmoComposer. It listens on port 1726 so an application that wishes to connect to an instance of EmoComposer running on the same computer must call EE_EngineRemoteConnect(127.0.0.1, 1726).
- Testing application's behavior, the same connection.
- Developer wants to speed the development process by beginning his application integration with the EmoEngine and the Emotiv headset without having to construct all of the UI and application logic required. Emotiv Control Panel can act as a proxy for either the real, headset-integrated EmoEngine or EmoComposer. Control Panel listens on port 3008 so, must call EE_EngineRemoteConnect(127.0.0.1, 3008).

Programming with the Emotiv SDK

```
// ... print some instructions...
std::string input;
std::getline(std::cin, input, '\n');
option = atoi(input.c_str());

switch (option) {
    case 1: {
          if (EE_EngineConnect() != EDK_OK) {
                throw exception("Emotiv Engine start up failed.");
          }
          break;
    }
    case 2: {
          std::cout << "Target IP of EmoComposer? [127.0.0.1] ";
          std::getline(std::cin, input, '\n');
          if (input.empty()) {
                input = std::string("127.0.0.1");
          }
          if (EE_EngineRemoteConnect(input.c_str(), 1726) != EDK_OK){
                throw exception("Cannot connect to EmoComposer!");
          }
          break;
    }
    default:
          throw exception("Invalid option...");
          break;
}
```

Listing 1 *Connect to the EmoEngine*

Programming with the Emotiv SDK

```
EmoEngineEventHandle eEvent= EE_EmoEngineEventCreate();
EmoStateHandle eState      = EE_EmoStateCreate();
unsigned int userID        = 0;
while (...) {
int state = EE_EngineGetNextEvent(eEvent);
// New event needs to be handled
if (state == EDK_OK) {
          EE_Event_t eventType = EE_EmoEngineEventGetType(eEvent);
          EE_EmoEngineEventGetUserId(eEvent, &userID);
          // Log the EmoState if it has been updated
          if (eventType == EE_EmoStateUpdated) {
                // New EmoState from user
                EE_EmoEngineEventGetEmoState(eEvent, eState);
                // Log the new EmoState
                logEmoState(ofs, userID, eState, writeHeader);
                writeHeader = false;
          }
    }
}
```

Listing 2 *Buffer creation and management*

Programming with the Emotiv SDK

- Before the end of the program, EE_EngineDisconnect() is called to terminate the connection with the EmoEngine and free up resources associated with the connection.
- The user should also call EE_EmoStateFree() and EE_EmoEngineEventFree() to free up memory allocated for the EmoState buffer and EmoEngineEventHandle.

```
EE_EngineDisconnect();
EE_EmoStateFree(eState);
EE_EmoEngineEventFree(eEvent);
```

Listing 3 *Disconnecting from the EmoEngine*

References

- http://emotiv.com/store/sdk/edition/sdklite/

  SDKLite developers will download the compressed file Emotiv_SDKLite_v1.0.x.exe, which contains both the SDKLite software and the User Manual.
- https://jira.ai2.upv.es/confluence/display/LOTO/2011/10/25/WGM+35.+EEG+Control+%28I%29

  Previous presentation