Remote gaze-tracking system on basis of TightVNC

System description

Version:	1.0
Last updated:	22.02.10
Created:	02.11.09

Author	Date	Version	Comments
Andrey Shipilov	02.11.09	0.1	Create document
Andrey Shipilov	07.11.09	0.2	Messages description added
Andrey Shipilov	08.11.09	0.3	Structures description added. Messages description updated.
Andrey Shipilov	14.11.09	0.4	TightVNC Server and client description.
Andrey Shipilov	14.01.10	0.5	Architecture description
Andrey Shipilov	01.02.10	0.6	Document structure changed
Andrey Shipilov	07.02.10	0.7	ETI java client changes
Andrey Shipilov	15.02.10	0.8	ETI server changes
Andrey Shipilov	15.02.10	0.9	Drawing process description
Andrey Shipilov	22.02.10	1.0	Completing document

CONTENTS

GLOSSARY

- **ETI** – eye-tracking information;
- **ETI TightVNC –** extended version of TightVNC system, allowing processing eye-tracking information;
- **VNC** – Virtual Network Computing, using RFB protocol;
- **RFB** - Remote FrameBuffer. Simple protocol for remote access to graphical user interfaces;
- **ETU Driver** – eye-tracking universal driver. Software, providing API for unified access to different types of eye-tracking hardware;
- **JNI** – Java Native Interface. Technology allowing call C++ methods from Java code;
- **Fixation** – result of filtering eye-tracking samples, received from eye-tracker;
- **Nickname** – short client description, that will be shown next to its eye mark.

SYSTEM DESCRIPTION

ETI TightVNC is a remote gaze-tracking system on basis of TightVNC intended for collecting and displaying eye-tracking information on several clients, using the same shared workspace. This system is build on ideology of VNC. It is implemented as extension of open source VNC project TightVNC. Current version of ETI TightVNC allows only translating and displaying eye-tracking information, but it can be further improved to allow complex processing of gaze data on server side.

Architecture

ETI TightVNC system can be divided into four main parts for convenience:

- Eye-tracker;
- ETU Driver;
- ETI TightVNC Java client with JNI wrapper for ETU Driver;
- ETI TightVNC Server.

Eye-tracker and ETU Driver are optional and can be absent on some clients. In that case such clients will be able only to show gaze information, received from other clients.

System uses TCP\IP to exchange information between clients and server.

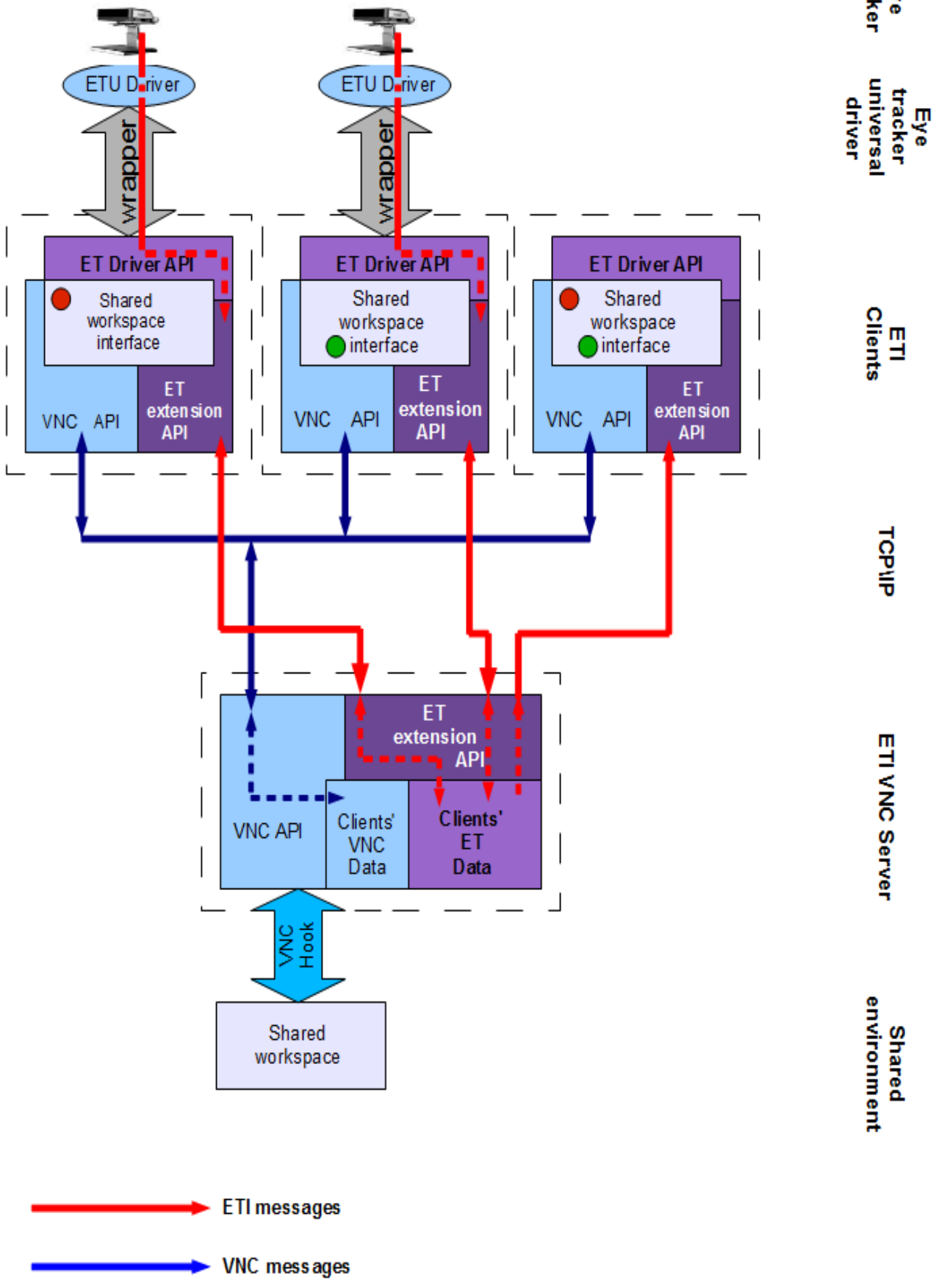


Illustration 1: Remote gaze-tracking system architecture

Gaze data processing

Overall process of collecting, processing and distributing of eye-tracking information:

1. Eye-tracker reads gaze data with defined frequency;
2. ETU Driver gets gaze data samples from eye-tracker;
3. Internal ETU Driver filter process acquired samples and generate fixations;
4. ETU Driver wrapper receives fixations from ETU Driver and send them to ETI TightVNC java client;
5. Client checks if fixation point is inside workspace area, translate coordinates and send it to the server;
6. Server receive and parse message and store received data to the clients' ETI collection;
7. After storing updated information server generate messages and distribute received gaze data to all other clients;
8. Each client parse message and store received gaze data into copy of clients' ETI collection;
9. After storing information client checks if current settings allow displaying of gaze data and redraw workspace image if needed.

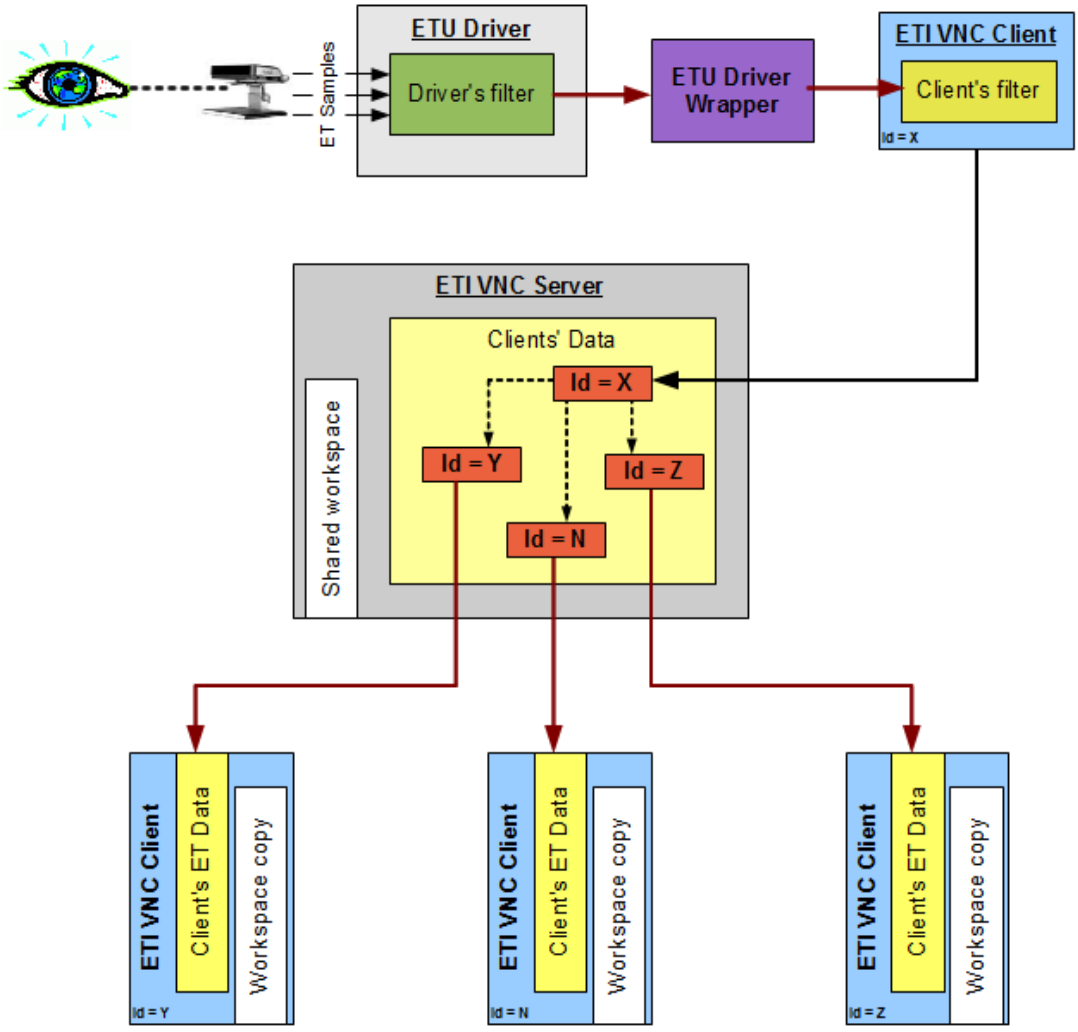


Illustration 2: Gaze data processing

ETI Gaze data components

Eye-tracking information that is used in the system include next components:

- Nickname – short description of the client, it will be shown next to the clients mark. It can be changed on each client independently;
- Description – extended description of the client. Clients cannot change descriptions of other gaze data sources;
- Coordinates of the gaze data mark;
- Graphical settings of gaze data mark.

Gaze data marks

ETI TightVNC java client outputs received eye-tracking information to the user, drawing marks on top of image of shared workspace. Parameters of each eye mark are stored in ETIItem class object (p.12).

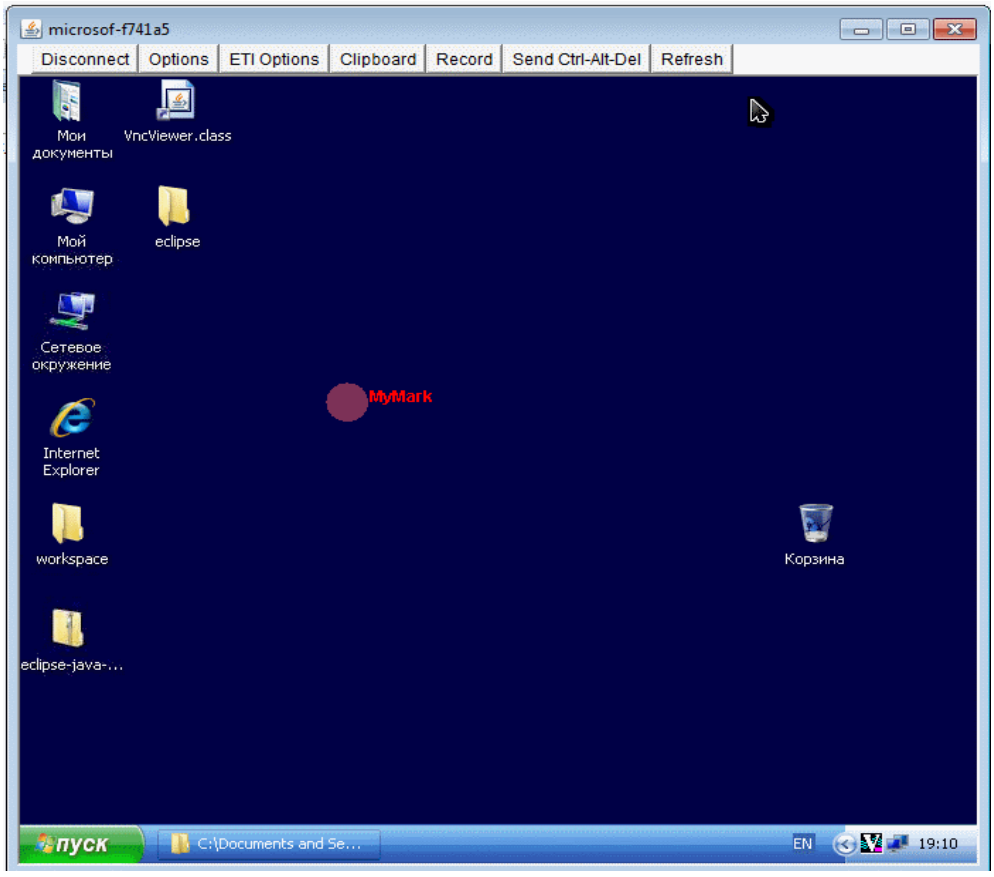


Illustration 3: Example of user interface

ETI mark consist of two components – graphical shape and text:

Illustration 4: Gaze data mark

Current version of ETI edition of TightVNC java client implements only one type of shape – transparent circle of some color, but client can be further improved to imlement posibility to draw other types of marks.

EXTENSION OF TIGHTVNC SOFTWARE

ETI TightVNC Server

ETI TightVNC server is based on TightVNC server 1.3.10 for Windows.

New data types

ETI edition of TightVNC implements several new structures, used to store gaze data.

- **etiItem**

  Store information about one eye-tracking point – coordinates of the point and ID of the client that own this ETI mark. Size - 6 bytes.

X (2 byte)
Y (2 byte)
Client ID (2 byte)

- **etiClient**

  Store information about one eye-tracking client. Contain information about client mark default colour. Size - 6 bytes.

Show Flag (2 byte)
R (2 byte)
G (2 byte)
B (2 byte)
A (2 byte)
Client ID (2 byte)

- **VncEYEClientsMap**

  std::map containing pairs <vncClientId, etiClient>;

- **vncEYEPosMap**

  std::map containing pairs <vncClientId, etiItem>.

New data collections

Server stores copy of all eye-tracking information passed through it. Two main collections, used for that:

- m_eyePositions – map of last received gaze data points from all clients;
- m_eyeClients – map of all clients that send gaze data information.

New messaging routines

ETI TightVNC server implements several wrapper functions, that can be used to distribute gaze information:

- AddClientToTrackingList(vncClientId id) – add client to tracking list and send it's Id to all other clients;
- RemoveClientFromTrackingList(vncClientId id) – remove client from tracking list;
- ResentClients(vncClientId send_to_id) – send to the client information about all records in current tracking list;
- UpdateEyePos(CARD16 x, CARD16 y, vncClientId id) – distribute between clients information about new gaze data point, received from one of the eye-tracking sources;
- SetNick(vncClientId id, LPSTR nick) – notify all clients about changing nick name on one of them;
- SetDesc(vncClientId id, LPSTR desc) – notify all clients about changing description on one of them.

ETI TightVNC Client

ETI TightVNC client is based on TightVNC java client version 1.3.10. It implements all functionality, presented in original client and adds some specific routines for gaze data processing.

New classes

ETI TightVNC java client implements several new classes, used for storing and processing eye-tracking information:

- ***ETIItem*** – class, used to store gaze information, received from one client;
- ***ETIInfo*** – stores all eye-tracking information, received from all other clients and handles the map of gaze data marks;
- ***ETMarkShape*** – information about graphical representation of the eye-tracking information mark, that will be shown to the user;
- ***ETIOptionsFrame*** – frame, showing list of all connected sources of gaze data and settings of current client;
- ***ETIItemOptions*** – frame, showing options of one eye-tracking mark, showed to the user;
- ***ETUDConnector*** – class for connecting to the wrapper and receiving gaze data points from the ETU Driver. JNI technology, used to implement wrapper strictly depends on the names of the classes and methods, that are used to call C++ code. Any changes to this class should be carefully planned as they can lead to impossibility to interact with eye-tracker.

More extended description of these classes can be fond in Java-doc to the ETI TightVNC java client source code.

New RFB messages

TightVNC java client includes description of RFB messages, that are processed between server and client. ETI version of TightVNC also includes

description of new messages, used for gaze data transmitting. New messages are described in section “Extension to RFB protocol” (p.15).

New RFB messaging function

ETI TightVNC java client implements several wrapper functions, that can be used to send information to the server, easily:

- writeNick(String text) – send nickname of the current client to the server;
- writeDesc(String text) – send description of the current client to the server;
- writeETIEyeMoved(int x, int y) – send to the server new coordinates, received from the current clients's eye-tracker;
- writeAddMeToList() - request to add the current client to the list of clients, sending gaze data. In case if client send actual eye-tracking daa, this message is not needed, client will be added to the list automatically;
- writeRemoveMeFromList() - request to delete client from the list of tracked clients;
- writeResentClientsList() - request to resend all list of tracked clients stored on server.

More extended description of these functions can be fond in Java-doc to the ETI TightVNC java client source code.

New drawing process

Original TightVNC client stores and processes only one image that is shown to the user. Client receive update messages from the server, draw them to the in-memory image and then output it to the user interface.

ETI edition of TightVNC assumes the output of additional graphical information. User should be able to control it independently from the server and other clients. Approach to the drawing of graphical information by client was changed to implement this requirement. ETI edition of TightVNC java client stores and processes two in-memory images:

- original TightVNC image, containing graphical information about remote desktop;
- image of eye-tracking marks.

Client merge these two images before output to the user interface. Such approach allows defining output options for each client independently. Any changes on the main image do not influence on drawing of eye-marks.

Use of additional layer during drawing user interface image allows easy adding some preprocessing logic (for example, removing trembling of eye-mark).

Illustration 5: ETI TightVNC drawing approach

This change makes it difficult to update the version of TightVNC java client, used as a basis for implemeting ETI edition of this software.

EXTENSION TO RFB PROTOCOL

ETI edition of TightVNC implements all functionality, that was implemented in the original system. In addition it allows collecting, processing and distributing eye-tracking information among clients. VNC systems use RFB protocol to exchange information. TightVNC implements RFB protocol and adds several custom messages. ETI version of TightVNC on its turn adds several own messages for processing eye-tracking information. All messages can be divided into two groups:

- TightVNC messages (standard VNC and extended TightVNC messages);
- ETI messages, used for distributing eye-tracking information.

RFB protocol messages includes two groups of messages:

- server-to-client messages;
- client-to-server messages.

ETI extension adds messages to both groups. All messages are aligned to 32-bits, to make it easier to send, receive and convert numbers.

Server To Client messages

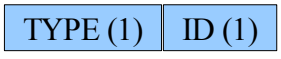
Server controls all communications between clients. After receiving new message from the client, server process it and propagate to other clients if needed.

etiEyePositionUpdateMsg

Send new eye-tracking information to the client. This message consists of a header giving the number of eye points followed by ***etiItem*** structures. The header is padded so that together with the type byte it is an exact multiple of 4 bytes. Currently server process and send ETI points by one. Size of the header of the message – 4 bytes.

etiRemoveTrackClientMsg

Send request to remove client from the list of tracked entities. It can be used to clean clients tracking lists from disconected eye-trackers. This message includes only ID of the client that should not be tracked anymore. Size of the message – 2 bytes.



etiSendClientNickMsg

Send request to change the nickname, shown next to the mark of the client.This message includes ID of the client and length of the nickname, followed by the text that should be sent to the recepient. Size of the message header – 8 bytes.

TYPE (1)	Pad (1)	ID (2)	LENGTH (4)	Text with LENGTH chars....

etiSendClientDescMsg

Send request to change the description, shown next to the mark of the client.This message includes ID of the client and length of the description, followed by the text that should be sent to the recepient. Size of the message header – 8 bytes.

TYPE (1)	Pad (1)	ID (2)	LENGTH (4)	Text with LENGTH chars....

Client To Server messages

Client send to the server information, received from eye-tracking device, and requests to changes its parameters, available for other clients.

etiEyeMovedMsg

Send new position of the eye, received from eye-tracking device, to the server. This message include x,y coordinates of eye mark. Size of the header of the message – 6 bytes.

TYPE (1)	Pad (1)	X (2)	Y (2)

etiAddMeToTrackMsg

Send request to update lists of tracked clients. Currently this message is not used. Tracked clients lists are updated automatically when new ET-information received. Only type of the message is send, as server already have the Id of the client. Size of the message – 1 bytes.

TYPE (1)

etiRemoveMeFromTrackMsg

Send request to update lists of tracked clients. This message can be send before closing client – so it won't be shown on other clients. Only type of the message is send, as server already have the Id of the client. Size of the message – 1 bytes.

TYPE (1)

etiResentTracksMsg

Send request to resent all trackable clients. This message can be used to sinchronise lists of ET-information between server and client. Only type of the message is send, as server already have the Id of the client. Size of the message – 1 bytes.

etiClientSetNickMsg

Send request to change nickname text assosiated with the client. Message consists of header (type of the message and length of following text) and text. Message padded to 32*n bits, as the largest variable is 32 bits (4 bytes). Size of the message – 8 bytes.

etiClientSetDescMsg

Send request to change description text assosiated with the client. Message consists of header (type of the message and length of following text) and text. Message padded to 32*n bits, as the largest variable is 32 bits (4 bytes). Size of the message – 8 bytes.

ETU DRIVER WRAPPER

ETU Driver is implemented on C++. It is required to use wrapper, using JNI, to connect this driver to the ETI TightVNC Client, written on Java.

Architecture

This wrapper includes an ETU Driver instance and a set of functions, written using JNI technology. This functions allow controlling ETU Driver from Java client and translate gaze data, received from eye-tracker back to the Java environment, using predefined callback routine.

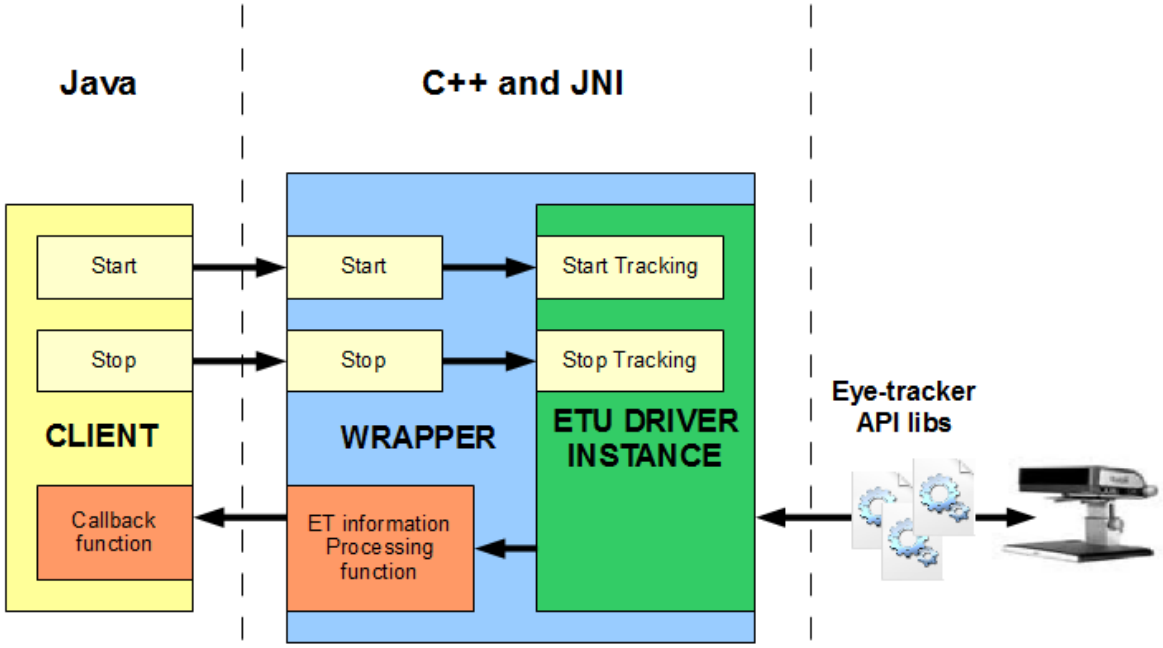


Illustration 6: ETU Driver Wrapper architecture

Interface methods

Wrapper methods, that can be invoked from Java client, include:

- *Java_ETUDConnector_Init(JNIEnv *env, jclass c)* – initialize ETU Driver instance;
- *JNICALL Java_ETUDConnector_Start(JNIEnv *env, jclass c)* – create a new ETU Driver controlling window and start listening for gaze data from eye-tracking hardware;
- *JNICALL Java_ETUDConnector_Stop(JNIEnv *env, jclass c)* – stop listening for gaze data from eye-tracking hardware;
- *JNICALL Java_ETUDConnector_ShowOptions(JNIEnv *env, jclass c)* – open standard ETU Driver options window;
- *JNICALL Java_ETUDConnector_Calibrate(JNIEnv *env, jclass c)* – start standard ETU Driver calibration routine;
- *JNICALL Java_ETUDConnector_getEyePosition(JNIEnv *env, jclass c)* – send to Java callback function last received fixation.

Limitations

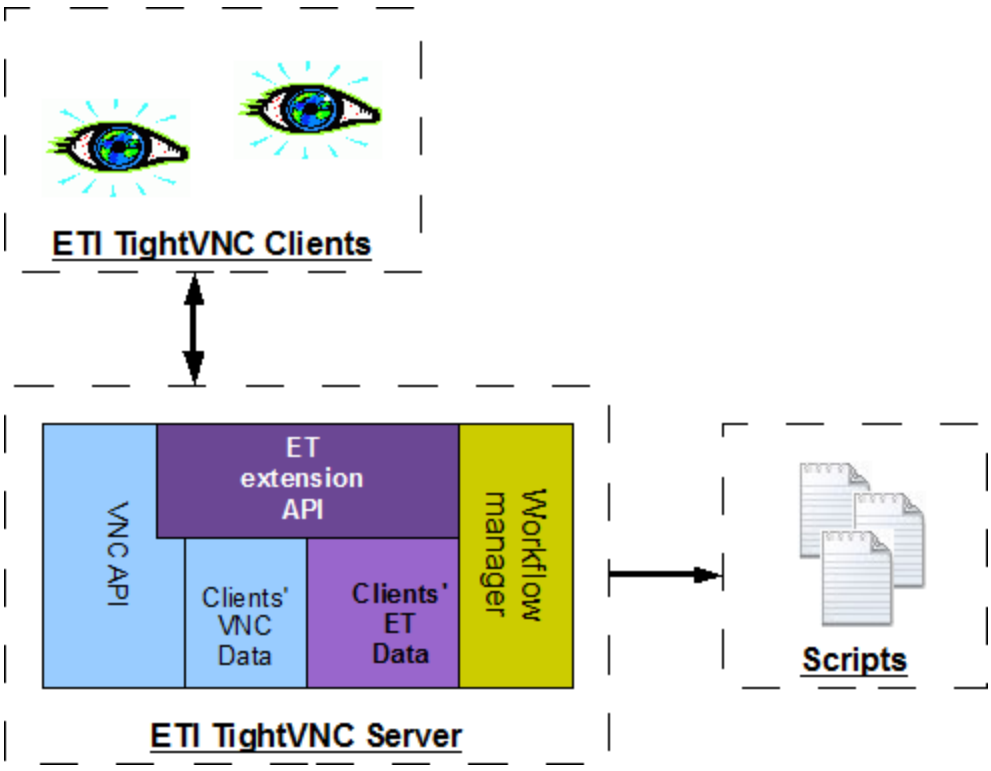
Implementation traits lead to some limitations in use of ETU Driver wrapper:

- ETU Driver Wrapper is strictly bounded to the Java client class names. In case of changes in source codes of ETI TightVNC Java client it should be checked, if wrapper is still operable.
- It is required to create dialog box during listening to the gaze information from eye-tracker. Otherwise, ETU Driver thread became suspended and do not send any gaze data to the wrapper.
- Java client requires access to the wrapper library. It restricts the use of Java client as a simple java applet, loading from web page.

FURTHER IMPROVEMENTS

Complex data processing

It is possible to implement "workflow manager" on server side to process different scripts. That will make it easier to implement different logic for eye-tracking data processing. Supposed architecture of the server will be:



One of approaches that can be used:

- Scripts are implemented using some language like python;
- Server starts scripts sending it eye-tracking information as an input.