Py4ET 2013: Accessing Eye Sample Data from ioDataStore

ioDataStore Access

In [40]:
```
# psychopy.iohub related imports.....
#
import psychopy.iohub
from psychopy.iohub.datastore.util import displayDataFileSelectionDialog, ExperimentD
from psychopy.iohub import EventConstants

# A couple general Python imports....
#
import os
import numpy as np

# Useful define for which sections of ipython notebook code should only run
# when a code cell is explicitly launched, not just loaded.
#
script =  __name__ == '__main__'
```

ExperimentDataAccessUtility Class

- Contains current ioHub Device Event Reading Functionality
- Simple Event Access API
- Access Events using Same Type Constants and Event Attributes as are Used During On-Line Event Access.
- When combined with Experiment Runtime use of ExperimentVariableProvider class, Events Access can be Filtered by:
    - Dependent and Independent Conditions
    - Session and Trial IDs
    - Other Variables Calculated at Runtime, e.g. Trial Start and End Times, Stimulus Onset and Offset Times, etc.

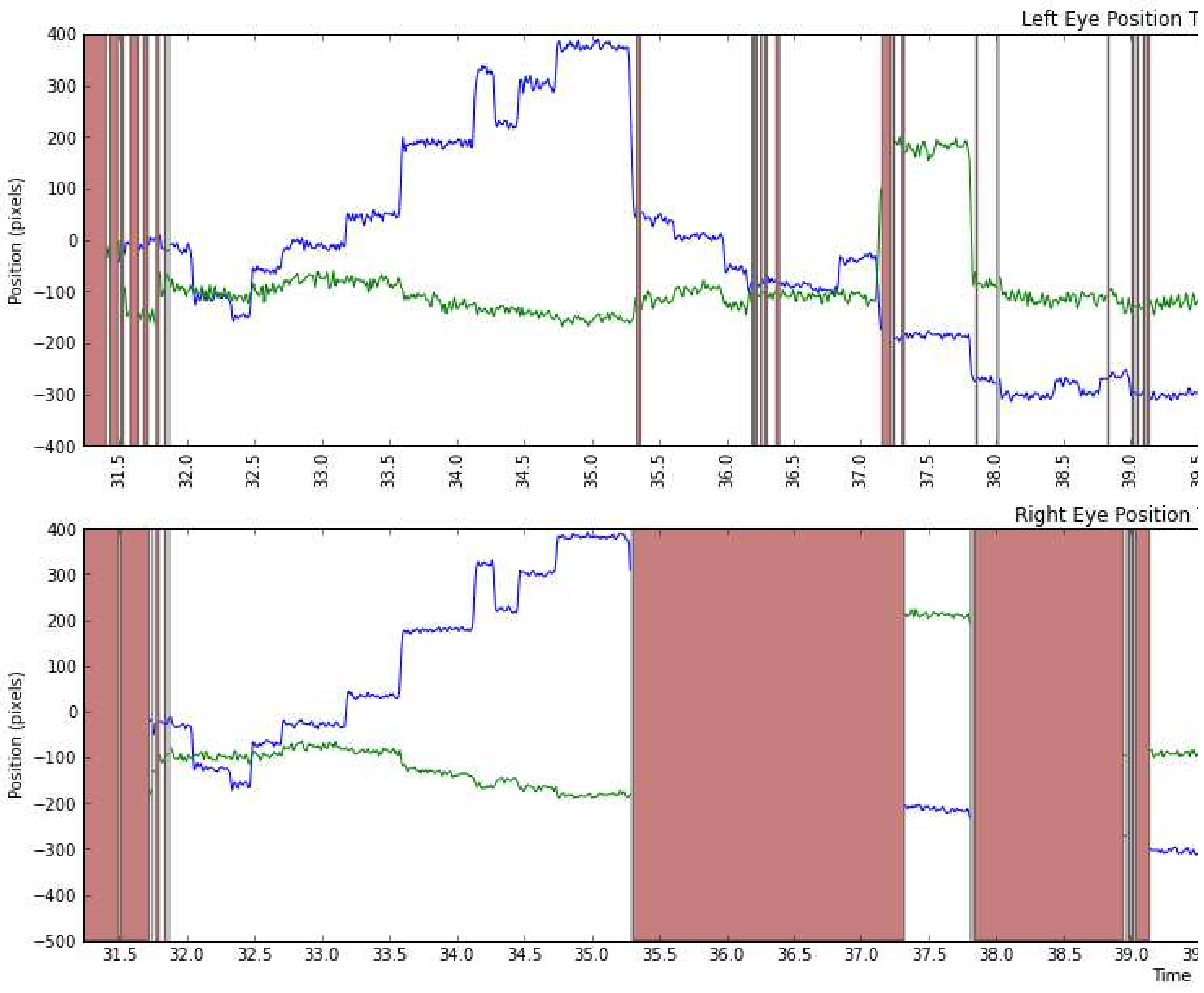
Example of Accessing Eye Sample Event Data

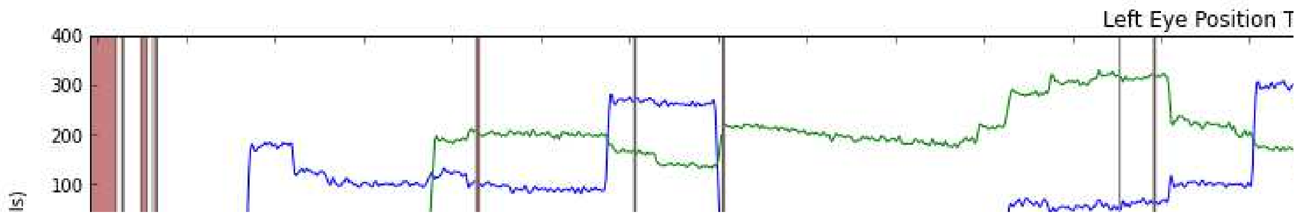
In [41]:
```
if script:
    # Select the ioDataStore hdf5 file to process.
    # (Function defined below)
    dpath,dfile=openDataStore()

    # Create an instance of the ExperimentDataAccessUtility class
    # for the selected DataStore file. This allows us to access data
    # in the file based on Device Event names and attributes.
    #
    dataAccessUtil=ExperimentDataAccessUtility(dpath,dfile, experimentCode=None,sessi

    # Get the trial condition variables for each experiment session
    # saved to the ioDataStore file. Returns a list of ConditionVariable
    # named tuples, usually one per trial that occurred during the experiment runtime
    #
    trial_conditions=dataAccessUtil.getConditionVariables()

    # Retrieve a subset of the BINOCULAR_EYE_SAMPLE event attributes, for events that
    # between each time period defined by the TRIAL_START and TRIAL_END trial variabl
    # in the trial_conditions data table.
    #
    session_trial_sample_data=dataAccessUtil.getEventAttributeValues(EventConstants.B
                                ['time','left_gaze_x','left_gaze_y','right_gaze_x','l
                                 'right_gaze_y','right_pupil_measure1','status'],
                                conditionVariablesFilter=None,
                                startConditions={'time':('>=','@TRIAL_START@')},
                                endConditions={'time':('<=','@TRIAL_END@')})

    # For each entry in the trial_conditions data table (i.e. trial),
    # set missing data appropriately for the given eye tracker hardware used
    # , in this case a Tobii system. (from blinks, eye occlusion, etc)
    #
    for t,tsamples in enumerate(session_trial_sample_data):
        tsamples.left_gaze_x[tsamples.status//10>=2]=np.NaN
        tsamples.left_gaze_y[tsamples.status//10>=2]=np.NaN
        tsamples.left_pupil_measure1[tsamples.status//10>=2]=0
        tsamples.right_gaze_x[tsamples.status%10>=2]=np.NaN
        tsamples.right_gaze_y[tsamples.status%10>=2]=np.NaN
        tsamples.right_pupil_measure1[tsamples.status%10>=2]=0

def here():
    pass

def openDataStore():
    data_file_path= displayDataFileSelectionDialog(psychopy.iohub.module_directory(he
    if data_file_path is None:
        sys.exit(0)
    return os.path.split(data_file_path)
```

Plotting Retrieved Eye Sample Data

In [42]:
```
# Data Plotting Imports and Setup
#
import matplotlib.cm as cm
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import matplotlib.transforms as mtransforms

orginal_plt_width=None
orginal_plt_height=None
plt.rcParams['figure.figsize'] =6,4
if orginal_plt_width is None:
    orginal_plt_width,orginal_plt_height=plt.rcParams['figure.figsize']
    plt.rcParams['figure.figsize'] =orginal_plt_width*4,orginal_plt_height*2.5

# Example function for plotting eye sample position data from binocular data recordin
#
def plotSampleTraces(trial_id,tsamples):
    fig = plt.figure()
    ax2 = fig.add_subplot(212)
    ax1 = fig.add_subplot(211,sharex=ax2)
    ax1.plot(tsamples.time,tsamples.left_gaze_x,label='Horizontal Position')
    ax1.plot(tsamples.time,tsamples.left_gaze_y,label='Vertical Position')
    ax2.plot(tsamples.time,tsamples.right_gaze_x,label='Horizontal Position')
    ax2.plot(tsamples.time,tsamples.right_gaze_y,label='Vertical Position')
    ax2.set_xlabel('Time')
    ax1.set_ylabel('Position (pixels)')
    ax2.set_ylabel('Position (pixels)')
    ax1.set_title("Left Eye Position Traces: Trial %d"%(trial_id,))
    ax2.set_title("Right Eye Position Traces: Trial %d"%(trial_id,))
    tmin=tsamples.time.min()//1
    tmax=tsamples.time.max()//1+1
    #trange=tmax-tmin
    plt.xticks(np.arange(tmin,tmax,0.5),rotation='vertical')

    trans1 = mtransforms.blended_transform_factory(ax1.transData, ax1.transAxes)
    trans2 = mtransforms.blended_transform_factory(ax2.transData, ax2.transAxes)
    ax1.fill_between(tsamples.time, 0, 1, where=tsamples.left_pupil_measure1==0, face
    ax2.fill_between(tsamples.time, 0, 1, where=tsamples.right_pupil_measure1==0, fac

    plt.legend()
    plt.show()
```

In [43]:
```
# Actual Plotting Loop....
#
if script:
    # For each entry in the trial_conditions data table plot the left and right
    # eye position data, with graphics depicting any periods of eye data loss
    # (from blinks, eye occlusion, etc).
    #
    for t,tsamples in enumerate(session_trial_sample_data):
        plotSampleTraces(t+1,tsamples)

    # Done creating plots, reset figure size back to original state
    #
    plt.rcParams['figure.figsize'] =orginal_plt_width,orginal_plt_height

    # Close the data file that was used.
    #
    dataAccessUtil.close()
```





That's It!