Lehrstuhl für Medieninformatik
Institut für Information und Medien, Sprache und Kultur
Universität Regensburg

Interaction Technology and Techniques
Assignment 11: Wiimote, PyQtGraph

Summer semester 2016

Submission due: Friday, 17. June 2016, 23:55 (extended deadline)

Hand in in groups of max. two.

Your task is to get comfortable with the WiiMote and PyQtGraph

11.1: A short introduction to Digital Signal Processing

Read at least chapters 14 and 15 of *The Scientist and Engineer's Guide to Digital Signal Processing*[1] - and earlier chapters if necessary for understanding.

Concisely answer the following questions:

- What is random noise?
- What does a low-pass filter do in general?
- Is a *moving average* filter a low-pass or a high-pass filter? Why?

Points

- **1** Good answer to first question
- **1** Good answer to second question
- **1** Good answer to third question

11.2: WiiMote as a digital bubble level

Read the source code for `wiimote.py` and have a look at `wiimote_demo.py`. Write a small Python application `level.py` that takes a Bluetooth MAC address as its only parameter. This application should turn your WiiMote into a digital bubble level with the following properties:

- the accelerometer measures inclination in X and Y axis
- the LEDs show direction and amount of deviation from the horizontal: `*---`, `**--`, `***-`, `****`, `-***`, `--**`, `---*`
- once the WiiMote is perfectly horizontal, all LEDs light up and the WiiMote rumbles once.
- the directional buttons allow for setting the axis of measurement to be used (X or Y).

The application should `import wiimote` similar to the `wiimote_demo.py` example.

Hint: Activating the rumble motor will mess with the accelerometer values. You might want to wait for a short time until you read and interpret them again.

Hand in the following file:

[1] http://www.dspguide.com/

level.py: a Python script that implements your digital bubble level.

(Please do not hand in `wiimote.py`.)

Points

- **1** The python script has been submitted, is not empty, and does not print out error messages.
- **2** The script correctly implements the features described above.
- **1** The script is well-structured and follows the Python style guide (PEP 8).

11.3: A custom PyQtGraph flowchart using the WiiMoteNode

Read the source code for `wiimote_node.py` and the PyQtGraph documentation[2]. Write a small Python application `analyze.py` that takes a Bluetooth MAC address as its only parameter. This application should generate a PyQtGraph flowchart with the following elements:

- a WiiMoteNode.
- a *BufferNode* (see `wiimote_node.py`) for each of the accelerometer channels,
- one or more of the default filter nodes
- three nodes that plot the accelerometer data
- *optionally:* a new *NormalVectorNode* that calculates the rotation around one axis from the accelerometer values of the other two axes and outputs a vector (i.e., two 2D points) that can be plotted by a `PlotWidget` to indicate the rotation (see video in GRIPS).

Your application should import `wiimote_node.py` and use the two nodes defined there.

Hand in the following file:

analyze.py: a Python script that implements this flowchart.

(Please do not hand in `wiimote.py` or `wiimote_node.py`.)

Points

- **1** The python script has been submitted, is not empty, and does not print out error messages.
- **2** The script correctly implements and displays a flowchart.
- **2** The script correctly reads accelerometer data from the Wiimote and plots it.
- **1** The script is well-structured and follows the Python style guide (PEP 8).
- **2** *Optional:* the script contains a working *NormalVectorNode* as described above.

Submission

Submit via GRIPS until the deadline

All files should use UTF-8 encoding and Unix line breaks. Python files should use spaces instead of tabs. If you need to submit further supporting files, please add a comment describing their use.

`Have Fun!`

[2] http://pyqtgraph.org/documentation/