Enhanced visualization: assignment 6

2D and 3D rendering with OpenGL

I. 2D rendering

In this part we will study OpenGL code in its simplest stance: 2D rendering. Download the notebook *opengl2d.ipynb* from *moodle*.

A. Drawing a plot.

1. Run the cells of the notebook and comment the numbered commands (*Commands* #$1-7$).
2. Modify the code to draw a second simultaneous plot of a sine function.
3. Modify the plot to draw a random white Gaussian process with zero mean and standard deviation $\sigma = 0.1$.
4. Add a square grid with inter line space of $\delta x = \delta y = 1$.

ECG

1. Download the file *ecg.txt* from *moodle*[1]. Plot this electrocardiographic (ECG) signal in a graphic with the following specifications:
   - The window should correspond to 1 second of signal and it should simulate an online acquisition, with new samples being drawn on the right. You should keep a vector with only the visible part of the signal.
   - To simulate a continuous acquisition, once you have drawn all the samples you restart on the beginning of the signal (you will need to create a circular buffer).
   - The framebuffer refreshing rate should be the same as the one used for refreshing the screen of your computer. This will allow you to know how you should shift your circular buffer for each draw.
   - Draw the signal in standard ECG grid, specifications are given in

     `https://en.wikipedia.org/wiki/Electrocardiography#Electrocardiogram_grid`

     and the lines should be drawn in light pink color. The amplitudes should range from -0.5 mV to 5 mV. The ECG signal is the direct output of an analog to digital converter, its amplitude is quantized in integer values from 0 to 4095. You can consider that the quantizer is uniform (constant steps) with 0 mV corresponding to the value 2047 and each integer value step corresponding to a change of 0.01 mV.

[1] Source: `https://github.com/PIA-Group/BioSPPy`

II. 3D rendering

A. Rendering of basic objects. In this part we are going to test the rendering of simple 3D objects. Download the notebook *opengl3d.ipynb*.

1. Execute cells #1 – 4 and explain what each of these cells do. If pyOpenGL has been installed correctly you should see for Cell #1 and #2 what is displayed in Figure 1.
2. What are the input parameters of the function gluPerspective? What is its role? What is the difference between this function and glFrustum?
3. What does glRotatef do in Cells #3 and #4 ?
4. Design your own rotation matrix and replace glRotatef with glMatrixMult by using your matrix.
5. How can you control the speed of rotation of the objects? How can you control the axis of rotation?
6. In Cell #4, what does the command glEnable(GL_DEPTH_TEST) do? What happens if you comment this code line? Why?

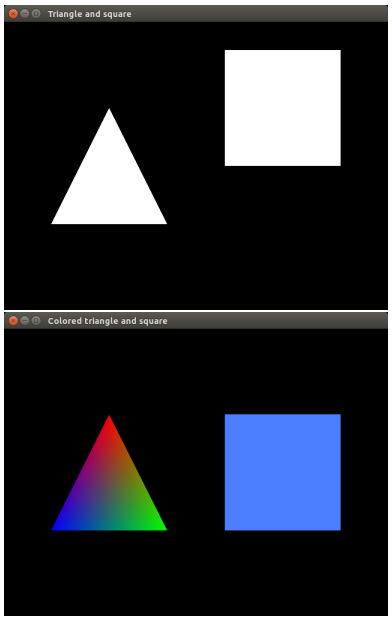

Figure 1: Cell #1 (top) and Cell #2 generated images with notebook *opengl3d.ipynb*.

B. Rendering of a robot arm. Download the notebook *robotarm.ipynb*[2]. This notebook simulates an articulated robot arm with a pincher claw.

1. Read and execute the code cell, you should see a window like in Figure 2.
2. Describe to which parts of the robot correspond Part #1 – 6 in the code (base, lower arm, middle arm, upper arm, hand, pincher claw).
3. What are the keys that should be pressed on the keyboard to move/rotate each part? Around what axis each part rotates and what is the increase in the angle of rotation each time we press a key?
4. Modify this code so that you are able to stretch or reduce the length of the upper arm part with the keyboard keys "R" and "r" respectively.
5. Modify this code so that the robot arm looks more like a human arm, that is, with two arm parts and a rotating hand with 5 fingers.

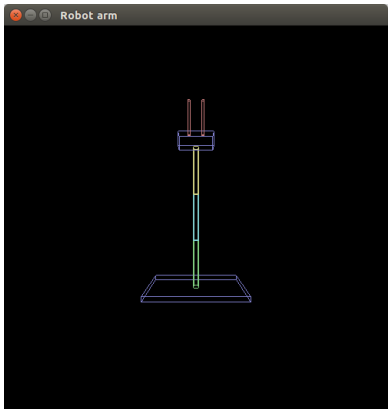

Figure 2: Articulated robot arm in OpenGL.

[2] This code is a modification of a code that you can find here `http://www.dreamincode.net/forums/topic/223384-robot-arm-opengl-python/`.