

Development of a GUI to display signal as collected by the TOF-F detector of NA61

Cressida Cleland¹ under the supervision of Emil Kaptur² and Piotr Podlaski³

¹University College Dublin, Ireland; ²University of Silesia, Poland; ³University of Warsaw, Poland

Summer 2018

Abstract—This report outlines the steps taken in developing a GUI to display signal as captured by the TOF-F detector of NA61. The GUI contains 32 scintillators, a canvas for graphing the signal, buttons for inputting the run number and event number and buttons for switching between events. If the selected run and event have activated scintillators, these scintillators are displayed visually, along with the physical position of the particle detection on the scintillator. The GUI was programmed using PyQt4 and Python 2.7 libraries.

I. INTRODUCTION

NA61/SHINE (SPS Heavy Ion and Neutrino Experiment) is a fixed-target multi-experiment in CERN's North Area. Studies include hadron-proton collisions, hadron-nucleus collisions, and nucleus-nucleus collisions [1, 2]. The main goals of these experiments include studying the properties of the onset of deconfinement [3] and the search for the critical point of strongly interacting matter through the investigation of p+p, p+Pb, and nucleus-nucleus collisions, and precise measurement of hadron production for the improvement of calculations of the initial neutrino beam flux in long-baseline neutrino oscillation experiments [4, 5] and for more reliable simulations of cosmic ray showers [6, 7].

During NA61 data taking for the T2K (Tokai to Kamioka) neutrino oscillation experiment, low-momentum particles are produced and so a detector between the right and left time-of-flight detectors is required. Hence the time-of-flight-forward detector was constructed and placed between the ToF-R and ToF-L detectors. This provides full time-of-flight coverage of particles exiting the spectrometer.

The ToF-F detector consists of 80 scintillators, 32 of which collect data created from particle detections. Each scintillator has two photo-multipliers on each end, totaling 64 channels. The dimensions of each scintillator is $120 \times 10 \times 2.5 \text{ cm}^3$ [1].

II. APPLICATION

A. Getting familiar with PyQt4

As I had little experience with GUI programming, I had to take some time to familiarise myself with PyQt4. This involved following numerous tutorials online to learn how to create the elements I would eventually need in the final application. When I was confident I had a working program, my supervisor wrote me a script that acted as an interface between the raw data compiled and stored using C++ code, and any Python script I wrote. With this interface I could work with the NA61 data.

B. Stages of Completion

Implementing the actual NA61 data was not an issue. At this stage of the application, there were 32 scintillators on the screen. There was no visual indication of which of these were activated; the terminal simply printed whether or not each scintillator was activated, as in Figure 1. The next step was to visually represent this by drawing the scintillators in a different colour depending on whether or not it contained data above a certain threshold (to account for noise).

My next task was to make it so that the scintillators could be clicked and the signal collected by that scintillator could be graphed. To do this I created 32 separate buttons assigned to 32 separate functions. Admittedly this is a clunky way to implement such a feature and a lot of time was spent trying to figure out how to streamline this. The issues encountered will be discussed in §III.

I had no problems plotting the signals using `matplotlib`, however the graphs were plotted in new window. I had some incompatibility issues when it came to plotting the graphs in the same PyQt window. I tried to use a different graphing library intended to be used with

PyQt4, `pyqtgraph`, but then I had trouble incorporating a dedicated graphics window into my pre-existing main window. After showing my supervisor a few different attempts, he updated the libraries and gave me some assistance with my code, after which everything launched in the same window as intended.

After some rearranging of the different elements in the window, I moved on to the next task: graphically plotting the position of the detected particle in the scintillator. This was done by finding the time (from $t = 0$) at which the maximum amplitude of signal occurs, converting from nanoseconds to seconds, and multiplying by the speed of light. This

```
Scintillator 0 has no signal.  
Scintillator 1 has no signal.  
Scintillator 2 has no signal.  
Scintillator 3 has no signal.  
Scintillator 4 has no signal.  
Scintillator 5 has no signal.  
Scintillator 6 has signal.  
Scintillator 7 has signal.  
Scintillator 8 has no signal.  
Scintillator 9 has no signal.  
Scintillator 10 has no signal.  
Scintillator 11 has no signal.  
Scintillator 12 has no signal.  
Scintillator 13 has no signal.  
Scintillator 14 has signal.  
Scintillator 15 has no signal.  
Scintillator 16 has no signal.  
Scintillator 17 has no signal.  
Scintillator 18 has signal.  
Scintillator 19 has no signal.  
Scintillator 20 has signal.  
Scintillator 21 has signal.  
Scintillator 22 has no signal.  
Scintillator 23 has no signal.  
Scintillator 24 has no signal.  
Scintillator 25 has no signal.  
Scintillator 26 has signal.  
Scintillator 27 has no signal.  
Scintillator 28 has no signal.  
Scintillator 29 has no signal.  
Scintillator 30 has no signal.  
Scintillator 31 has no signal.
```

Fig. 1. The terminal printed which scintillators were active.



Fig. 2. The program on start-up.

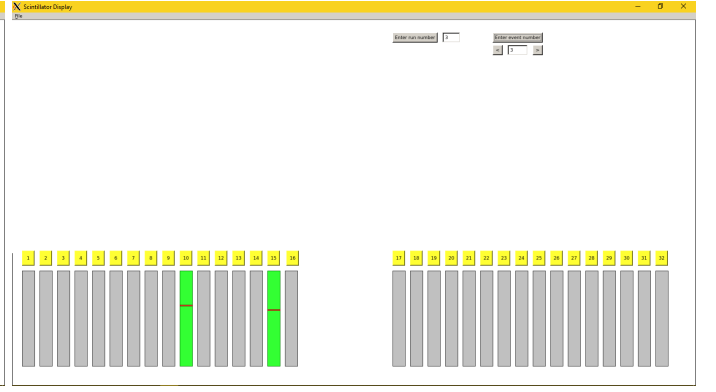


Fig. 4. The activated scintillators are displayed with the signal.

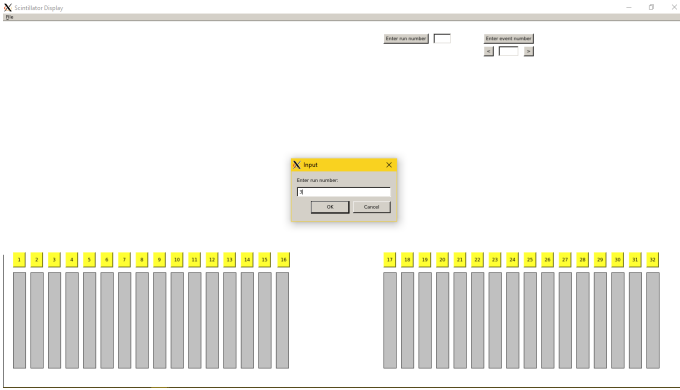


Fig. 3. The user-input for the run number.

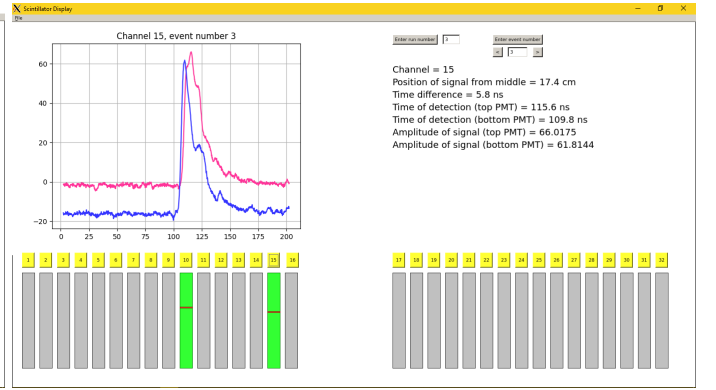


Fig. 5. A channel is selected and the signal is graphed.

‘distance’ is taken as the distance up the scintillator from the middle, with positive values having positive y-value.

The next task was to implement user input and navigation options. I implemented text input buttons so that the user could specify which run and which event to display. At this stage I had some errors on start-up since there was no data with which to draw the scintillators before the run or event numbers were input. After making it so that the program created null datasets, the program ran smoothly with no errors.

The program still has some bugs that I haven’t had time to fix. These will be discussed in §III.

C. Completed program

Figures 2, 3, 4 and 5 show the completed program and how it can be used.

On start-up, as shown in Figure 2, all 32 scintillators are displayed as grey since there is no data yet. The user can click the ‘Enter run number’ button or the ‘Enter event number’ button to input the run number and event number respectively. Nothing changes on the screen until both numbers are input.

A new window is opened for the text input as shown in Figure 3.

The activated scintillators are lit up in green as Figure 4 shows. The user can click one of the yellow buttons to

graph the signal of that scintillator and to see the relevant information, as shown in Figure 5.

III. ISSUES

- As for the scintillator buttons, I first tried to create a loop that would attach one button to each channel, that would then pass each channel through the plotting function. For whatever reason, I got errors when trying to call the function with the channel as an argument. I believe this is to do with the way that the `QPushButton` class operates, but I decided I would implement individual buttons with individual functions and try to neaten the code later if I had time.
- The position of the signal as graphed on the scintillators is partly arbitrarily calculated. As a placeholder, the time difference is multiplied by c and then scaled by 10 to make the position difference visible on the program, however this isn’t accurate as the signal wouldn’t move at exactly c through the scintillator. Finding out the refractive index of the scintillator and applying that in the calculation would fix this issue.
- One major bug that I don’t immediately know how to fix is that once one of the graphs is plotted, the input and navigation buttons don’t work anymore. Since all the other graphing buttons work still, I imagine this may have something to do with the way the other buttons

are refreshed and updated, but I would have to spend considerable time looking into how to fix it.

IV. POTENTIAL IMPROVEMENTS

If I had more time to work on this project there are some minor things I would change to improve the program. Along with fixing the bugs listed in §III, I would streamline the program overall to provide more customisability (i.e. remove any hard-coded parameters). The structure of the code is rather messy considering how different elements were added at different times, so I would completely overhaul the code to rewrite it in a way that makes structural sense. Cosmetically the program could be improved; the input and navigation buttons aren't very appealing. Also, the position of each element is hardcoded—given more time I would rewrite the code using the `QLayout` class to avoid any scaling issues in the future.

V. SUMMARY

Over the course of eight weeks I became reasonably familiar with PyQt4 and its classes, a framework with which I had no previous experience. I wrote a program that takes NA61 data as input from the user, plots the signal of the selected scintillator, and gives relevant information of that scintillator. Given more time I would fix some minor bugs and make some more general improvements.

ACKNOWLEDGMENT

This program would not have been completed without the guidance and support of my supervisors Emil and Piotr, nor without the support of the entire NA61 team. I've learned so much in such a short time, not only about GUI programming but also about what it's like to work in a collaboration such as NA61. For all this and more I'd like to thank everyone I've worked with at NA61 and CERN in general. I also have to thank the Julius Baer Group for kindly donating to CERN & Society and for making my summer program possible.

REFERENCES

- [1] N. Abgrall et al. [NA61/SHINE Collaboration], *NA61/SHINE facility at the CERN SPS: beams and detector system*, (2014) [arXiv:1401.4699v1].
- [2] N. Antoniou et al. [NA61/SHINE Collaboration], *Study of the hadron production in hadron-nucleus and nucleus-nucleus collisions at CERN SPS*, CERN-SPSC-2006-034.
- [3] M. Gazdzicki, M. Gorenstein and P. Seyboth, *Onset of deconfinement in nucleus-nucleus collisions: Review for pedestrians and experts*, Acta Phys. Pol. **B 42** (2011) 307.
- [4] Y. Itow et al. [T2K Collaboration], *The JHF-Kamioka neutrino project*, (2001) [arXiv:hep-ex/0106019].
- [5] N. Abgrall et al. [NA61/SHINE Collaboration], *NA61/SHINE plans beyond the approved program*, CERN-SPSC-2012-022; SPSC-P-330-ADD-6.
- [6] J. Abraham et al. [Pierre Auger Collaboration], *Properties and performance of the prototype instrument for the Pierre Auger Observatory*, Nucl. Instr. Meth. **A 523** (2004) 50.
- [7] T. Antoni et al. [KASCADE Collaboration], *The cosmic-ray experiment KASCADE*, Nucl. Instr. Meth. **A 513** (2003) 90.