EPJ Web of Conferences **214**, 06015 (2019) https://doi.org/10.1051/epjconf/201921406015
CHEP 2018

gopaw, a refactoring of CERN-PAW by using the softinex tools.

Guy Barrand[1,*]

[1]Laboratoire de l'Accélérateur Linéaire, Université Paris-Sud, CNRS-IN2P3, Orsay, France

Abstract. gopaw is for the "Good Old PAW". We present the motivations to do such a program along with the technical choices to do it.

1 Introduction

A motivation to do gopaw came from the introduction of the HDF5 [1] file format in the Geant4 [2] analysis toolkit to store histograms and ntuples. See the related poster at the same CHEP Sofia conference about the introduction of HDF5 in Geant4, and we refer to [3], [4], [5], [6] for the Geant4 analysis toolkit. It is fine to produce such files, but having interactive analysis tools that understand them would be highly welcome. We already improved our GUI oriented ioda [7] program so that it knows these files, but being aware that a lot of HEP people prefer command typing, we look also for a keyboard oriented application. In HEP history, CERN-PAW [8] has been very popular and the minimal typing to open a file (H/FILE) and plot an histogram (H/PLOT) or do an ntuple projection (N/PRO), is rather appealing to do a refactoring of this program by using our softinex [11] technology. Another motivation of this work is also related to the long questioning of the author about the right API to do (HEP) physics; are we at the optimum with the today technologies? Can we improve that? We are going to discuss these points in the dedicated section 11.

2 The command engine

The command engine is nothing more than the KUIP one found in the "old" CERNLIB [9] mainly written in FORTRAN. It was one of the few components written in C at that time, and we took the source code from packlib/kuip of the CERNLIB/2006a release. We took also the command description files `kuip.cdf` and `paw.cdf` from here. Using all these permits at once to have the same basic features and behaviour than the original CERN-PAW FORTRAN implementation. The "only" thing to do was then to reimplement/refactor the "callback" code of these commands, knowing their parameters and possible values. In the original CERN-PAW, these callbacks were put in various files `pafith.F, pafitv.F, pafunc.F`, etc... whose names appear in the `pawcdf.cdf`. For design simplicity, we keep this similar structure by having `pafith.cpp, pafitv.cpp, pafunc.cpp`, etc... but by coding the callbacks in C++ in these files. In gopaw/1.0 [16], we implemented the callbacks needed to execute the examples `pawex1.kumac` up to `pawex24.kumac` by having a very close rendering than CERN-PAW. See figure 2. Then a lot (but not all) command and option callbacks are implemented in this first public release.

*e-mail: barrand@lal.in2p3.fr

3 Graphics

We use our `inlib/sg` scene graph manager for high level graphics. It has a similar architecture as the OpenInventor [10] one. For inlib see [11]. `inlib/sg` permits to setup a scene by specifying a tree of "nodes" as `inlib::sg::ortho` to define a camera, `inlib::sg::matrix` to define the position of a shape as an `inlib::sg::cube` or `inlib::sg::sphere`. The `inlib/sg` library of nodes is sufficiently rich now in order to have the node `inlib::sg::plotter` able to reproduce good part of the plotting rendering of CERN-PAW. It can also reproduce the style of CERN-ROOT by using the `root_def.kumac` script. See figure 3. (For ROOT, see [12]).

Final images are done by traversing a scene graph with a "renderer". For screens, we have a GL-ES renderer that permits to do graphics on MacOS, Linux, Windows, iOS and Android by using the OpenGL libraries available on all these platforms. For MacOS and iOS, the announcement by Apple, at their WWDC of June 2018, to deprecate their OpenGL library [13], in favour of their Metal proprietary rendering engine, compromises our GL-ES strategy, but we foresee to have, in a short future, a dedicated Metal renderer for the Apple devices.

We have also offscreen renderers to produce Postscript, png or jpeg files. These are available in gopaw through the PICTURE/PRINT command.

4 Python to replace COMIS

As for CERN-PAW, we can define functions to be plotted or to fill histograms, etc.. In CERN-PAW it was done by using the COMIS FORTRAN interpreter. In gopaw, we use a Python [14] kernel to do that. For example, in `pawex10.kumac`, CERN-PAW draws a fractal with the command:

```
fun2 10 mandel.f [1] -2.4 .8 [1] -1.2 1.2 ' '
```

whilst in gopaw it is done with:

```
fun2 10 mandel.py [1] -2.4 .8 [1] -1.2 1.2 ' '
```

We show in figure 4 and figure 5, the two versions of the mandel script.

Note that we added the command:

```
/SESSION/PYTHON <file>.py
```

in order to execute other python scripts. In gopaw/1.x, the Python kernel which is used is the one of a Python/2.7 release.

5 On the fly compilation and loading

To define functions, we have also an "on the fly compilation and plugin loading" mechanism, which is available if a compiler is correctly declared to gopaw. In fact, if a FORTRAN compiler is declared, someone can even recover the CERN-PAW COMIS way of doing by having FORTRAN scripts. Obviously, having compiled functions permits to be much more effective than manipulating interpreted ones.

6 File formats

gopaw can read ROOT and HDF5 files produced by the Geant4 analysis toolkit. It can also read FITS files used in astronomy. The opening of a file is done with the same command:

```
h/file <unit id> <file name>
```

For the ROOT file format, we use our light inlib classes to read these files. For HDF5, we rely on an external installation of the HDF5 library. For HDF5 and the ROOT file format, which have a tree directory structure to organise data, the command CDIR permits to change the current directory in the file, and LDIR permits to list the objects in a file directory. Being positioned on a directory, the famous command:

```
h/plot <storage id>
```

permits to read and plot and histogram. Ntuple commands, as N/PLOT, work on ntuples found in ROOT files, HDF5 files, but also on table TBL objects found in FITS files. For FITS, we have also the IMAGE_HDU command extension to show (astronomical) images stored in a file. See figure 1.

Note that gopaw is not able to read HBOOK files since we have no C library to do that, and that we do not want to tie to any FORTRAN code and the old CERNLIB anymore.

7 Fitting

Fitting is done by using a refactoring of the code of `ROOT/TMinuit` (copyright respected), extracted from ROOT, put in `inlib::f2cmn`, and arranged to depend only on the standard C/C++ libraries.

8 VECTOR and SIGMA

VECTOR handling and SIGMA are done with the `inlib::array` template class, which is an effective multidimensional array template over `std::vector`. The SIGMA command is done with the little `exlib::yacc::cexpr_eval` single C instruction interpreter. Something like V1*V2 loops directly on the two `std::vectors` in an effective way.

9 Platforms

Portability is a high priority for us. gopaw runs on MacOS, Linux, Windows-10 and we have also a docker container on docker hub [15]. The core engine (including KUIP and Python) can be built also on Android and iOS and we have embedded a “kumac reader” in our ioda application. On Android and iOS, we have not yet finished the work to have an effective way to input commands from a virtual keyboard, and then have not yet a gopaw application on the Apple AppStore and GooglePlay stores. Having a gopaw application on an iPad makes definitely sense for us and it remains a nice target to work on.

10 Availability

Public releases are deposited on GitHub at [16] and the web pages are at [17].

11 Discussion. A dedicated analysis language?

Why doing gopaw? Beside the motivation to read Geant4 HDF5 files, it is also to try to answer a more fundamental question: what is the best user API to do (HEP) physics? At the arrival of ROOT in the 1990's, a user told us: with CERN-PAW I had the feeling to do physics, with ROOT I have now the feeling to type C++. There is some truth here. Especially for someone, as the author, involved in interactive and graphics software engineering for physics. If we look at the today software situation in HEP, it is particularly shocking to see what a new comer has to learn of technicalities before being operational in manipulating data to get a plot ready for publication. We even do not want to mention the pain to build an event display exploiting local graphics capabilities of devices that people have in hands now (laptops, tablets, smartphones). The overall computing environment should help and not be in the way for someone wanting to "think physics". With gopaw, we do not necessarily want to return to KUIP and CERN-PAW commands, but we would like to point out that we are missing a piece in physicist-computer interaction, by remembering that twenty years ago, from a keyboard, it was more easy to open a file, read a histogram and do a plot, that with what is proposed today.

The `RDataFrame` of ROOT, presented in this conference, could be seen as a super N/PLOT command. Here too, as twenty years ago with CERN-PAW, the "internal looping on data" is hidden from a user. The fact that the ROOT team recovers now this behaviour is a clear sign that we start to identify the lacking pieces. The usage of functional programming found in Scala and exploited in a system as Spark points in a similar direction; here too "internal looping" is no more seen/programmed by a user. (The vocabulary in Spark is "map", 'reduce' and, if used on tabular data, is similar to the "ntuple" and "projection" used in HEP).

But then, why not doing an extra quantum leap by going toward a specific dedicated language? Some language "HEP data oriented" where the keywords would be de facto "run", "event", "cut", "histogram", "ntuple", "fit" and "plot" beside, or in place of, generic keywords as "for", "if", "class". And then, why not a voice API ? "Hi gopaw, apply a pt cut of 10 GeV on the last run and show me the plot". Science fiction? Quite not, if there is an adequate language at hand; the technology is around the corner to be able to do that.

12 Conclusions

We presented the first release of gopaw to help exploiting HDF5 files produced by using Geant4 analysis tools. This CERN-PAW refactoring is done in C++. It keeps the C KUIP found in CERNLIB along with the command file descriptions of CERN-PAW, which ensure a similar behaviour as the original FORTRAN program. It comes with a new graphics layer based on a scene graph manager co working with two renderers; a GL-ES one for displaying on screen, and an offscreen one to produce Postscript, png or jpeg files. It uses a Python 2.7 kernel to script functions, the "f2c' of MINUIT (extracted from ROOT) to do fitting, a new C++ implementation of VECTOR commands and the single instruction interpreter `exlib::yacc::cexpr_eval` to replace SIGMA. gopaw has also extensions to read histograms and ntuples in a ROOT file, and exploit tables and images found in FITS files used in astronomy. We discussed also the opportunity to reintroduce an interactive program driven by a command system with commands "born from HEP physics", and this in a context where the trend is to go toward exposing users toward more and more technicalities and over-complexity that, we think, do not go in the direction to improve the physicist-machine interaction.

References

[1] https://portal.hdfgroup.org
[2] Agostinelli S et al, *Nucl. Instrum, and Methods*, **A506**, 250-303 (2003)
Allison J et al, *IEEE Transactions on Nuclear Science*, **53 No. 1**, 270-278 (2006)
Allison J et al, *Nuclear Instruments and Methods in Physics Research*, **A835**, 186-225 (2016).
[3] Hrivnacova I, Barrand G, *J. Phys.: Conf. Ser.*, **898**,042018 (2017)
[4] Hrivnacova I, *J. Phys.: Conf. Ser.*, **513**,022014 (2014)
[5] Barrand G, *J. Phys.: Conf. Ser.*, **513**, 022002 (2014)
[6] https://geant4.web.cern.ch. See the User's Guide for Application Developers section.
[7] https://github.com/gbarrand/ioda
[8] http://cern.ch/paw
[9] http://cern.ch/cernlib
[10] https://www.openinventor.com
[11] http://softinex.lal.in2p3.fr
[12] http://root.cern.ch
[13] https://developer.apple.com/videos/play/wwdc2018/604
[14] https://www.python.org
[15] https://hub.docker.com
[16] https://github.com/gbarrand/gopaw
[17] https://gbarrand.github.io

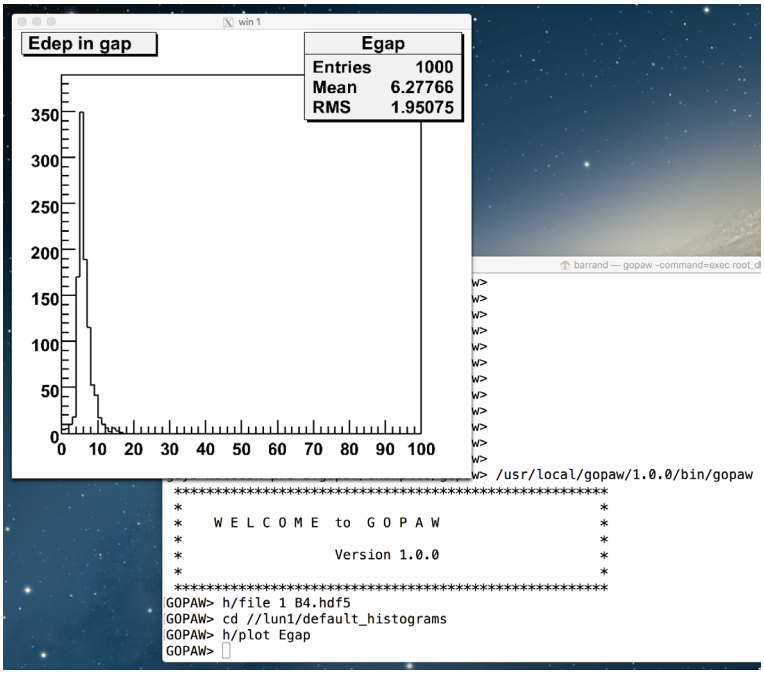

Figure 1. gopaw on the B4 hdf5 file

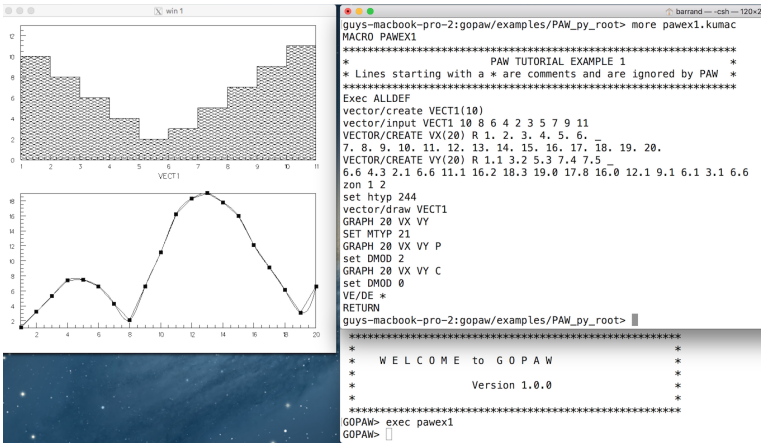

Figure 2. pawex1.kumac with gopaw

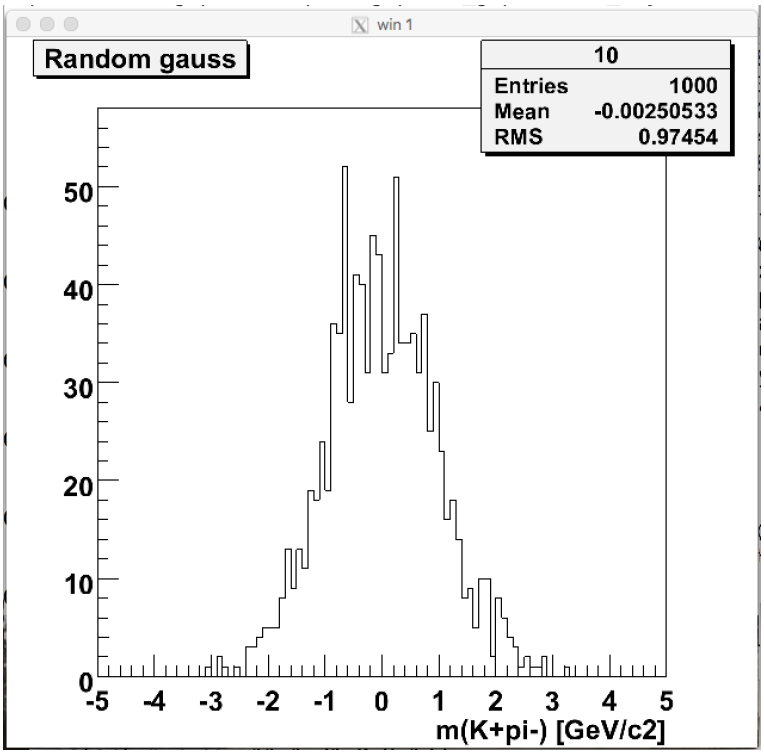

Figure 3. ROOT style

```
      REAL FUNCTION MANDEL(XP)
      DIMENSION XP(2)
      DATA NMAX/30/
      X=XP(1)
      Y=XP(2)
      XX=0.
      YY=0.
            DO 30 N=1,NMAX
               TT=XX*XX-YY*YY+X
               YY=2.*XX*YY+Y
               XX=TT
               IF (4..LT.XX*XX+YY*YY) GO TO 1
30          CONTINUE
 1          MANDEL=FLOAT(N)/FLOAT(NMAX)
      END
```

Figure 4. PAW mandel.f COMIS script

```
def mandel(XP,YP):
  NMAX = 30
  XX = 0
  YY = 0
  for N in range(1,NMAX+1) :
    TT = XX*XX-YY*YY+XP
    YY = 2.0*XX*YY+YP
    XX = TT
    if 4.0 < (XX*XX+YY*YY) :
      break
  return (1.0*N)/(1.0*NMAX)
```

Figure 5. gopaw mandel.py Python script