ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-WEPAF003

BEAMLINE ARCHITECT

Josiah Denton Kunz[*], Lizette Michelle Romero, Caleb Matthew Conrad
Anderson University, Anderson, IN, USA

Abstract

Beamline Architect is a new particle accelerator simulation interface. Currently, two of the most widely used tools in this field are G4beamline and COSY Infinity. While these codes are fast and quite accurate, sometimes their interfaces can be time-consuming for students to learn, particularly undergraduate students or students whose primary field is not accelerator physics. Without Beamline Architect, each code has its own high-level language that must be manually written into a file and then executed on the command line. Moreover, sometimes the use of both simulation tools is warranted in order to check for consistency between the codes. Writing the codes by hand or translating between software can sometimes be cumbersome, even for experts. Furthermore, knowledge of an additional language, such as Python, is required in order to analyze the outputs of the codes (which may be in different formats from one another). Beamline Architect is a tool that provides a graphical user interface to G4beamline and COSY Infinity. This lets the user build a particle accelerator channel in 3D with or without using code. The channel may then be saved, exported, translated, or run. Any output data will be plotted in Beamline Architect using Python, since it is both flexible aesthetically and quite standard in the particle accelerator community. For undergraduate and non-accelerator students, Beamline Architect allows a hands-on experience with accelerator simulations. Some applications for these students include health physics radiation dosimetry problems, medical imaging mechanics, security scanner simulations, and (of course) accelerator channel design for particle physics experiments. For experts, Beamline Architect provides visual confirmation of the channel and a faster, more consistent way of cross-referencing results between the codes.

INTRODUCTION

Beamline Architect is a new particle accelerator simulation interface. Its basic goal is to serve as a graphical user interface for several well-known text-based particle accelerator codes, such as G4beamline [1] and COSY Infinity [2].

As a demonstration, Fig. 1 is a COSY channel containing a magnetic quadrupole, a central magnetic solenoid, and another magnetic quadrupole, with their centers at z = -0.276, 0.307, and 0.482 m, respectively. In Beamline Architect, constructing the channel involves simply dragging the element and customizing the element via popup.

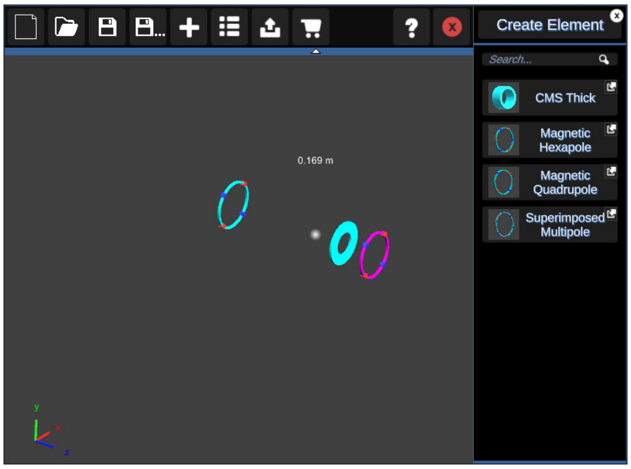


Figure 1: Example screenshot of Beamline Architect in COSY mode.

The channel may then be exported to a text file and run in COSY[1]. Since the exported file in its entirety is too large to be reproduced here, below is a snippet of the placing of the elements.

```
{INITIAL OFFSET}
DL -0.2805422;

{MQ named MQ1 and color RGBA(0.000, 1.000, 1.000, 1.000)}
MQ 0.01 0.1 0.1 ;
DL 0.5724507;

{CMST named CMST1 and color RGBA(0.000, 1.000, 1.000, 1.000)}
CMST 0.01 0 0 0.1 ;
DL 0.1647326;

{MQ named MQ2 and color RGBA(1.000, 0.000, 1.000, 1.000)}
MQ 0.01 0.1 0.1 ;
```

BASIC IMPLEMENTATIONS

At its core, Beamline Architect uses intuitive menus and popups to allow the user to construct a particle accelerator channel. The channel settings and user preferences are saved as a Beamline Architect channel at the user's choice of directory.

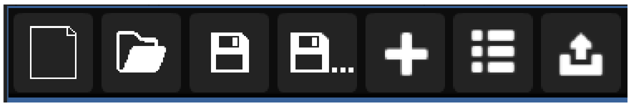
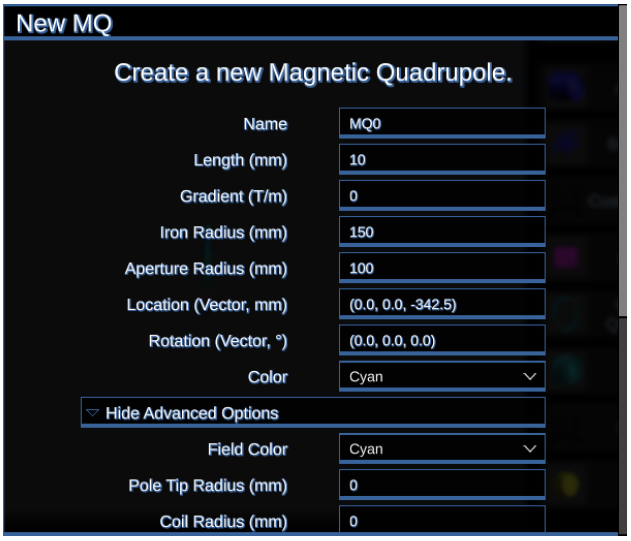


Figure 2: Top menu system for Beamline Architect. Symbols are from right to left: new channel, open channel, save channel, save channel as, new channel element, view current channel elements, and export to file.

[*] jdkunz@anderson.edu

[1] A distribution of COSY is not included with Beamline Architect.

Figure 2 shows the top menu system as it appears to the user. These are the primary actions that the user may take. The first set of four buttons are standard operations for the channel: new, open, save, and save as. The next two buttons pertain to the supported elements, be they particle accelerator elements (e.g., magnetic hexapole) or logic elements (e.g., for loop). With these two buttons, the user may add a new element or view a list of elements already in the channel. The last button is to export this channel so that it may be read by its native code (e.g., a `.fox` file for COSY).


New MQ

Create a new Magnetic Quadrupole.

Name	MQ0
Length (mm)	10
Gradient (T/m)	0
Iron Radius (mm)	150
Aperture Radius (mm)	100
Location (Vector, mm)	(0.0, 0.0, -342.5)
Rotation (Vector, °)	(0.0, 0.0, 0.0)
Color	Cyan

Hide Advanced Options

Field Color	Cyan
Pole Tip Radius (mm)	0
Coil Radius (mm)	0

Figure 3: Example popup menu for the G4beamline magnetic quadrupole element in Beamline Architect.

Once the users has selected an element, they may drag it into the channel. The popup menu (Figure 3) allows the user to customize the element. Several major options are listed with defaults, with all others in the advanced options menu.

G4BEAMLINE

G4beamline [1] is a particle tracking simulation software that can be used to simulate a multitude of different elements. G4beamline has a large library of known materials that help it to accurately model the way particles interact with matter. It is built on top of Geant4, which is a toolkit developed by CERN, and offers visualization of elements and eliminates the need for the user to know how to program in C++.

G4beamline has a long list of predefined elements that the user can choose from. To run a simulation with these elements, the user is required to write a text file containing all of the parameters of all of the elements that they wish to use. After all of the elements are defined, the user needs to place them in their proper position for them to run in G4beamline.

G4BEAMLINE IMPLEMENTATION

Not all undergraduate physics students are prepared to learn a unique language in order to run one particle simulation. Beamline Architect removes the need to write a text file or run anything from the command line. Beamline Architech offers a stylized user interface that is easy to use. The user only needs to change the parameters to match their needs, or use the defaults.

One of the greatest features of G4beamline is its customiz-ability. Most elements have a plethora of optional parameters that can be tweaked to the user's specifications. However, due to the large number of parameters, when writing the code the user may not be aware of what options the element has.

Beamline Architect separates every G4beamline parame-ter into one of two categories: mandatory or optional (see, e.g., Figure 3). Mandatory parameters are clearly laid out and typically have a default value assigned. Optional param-eters can be found under the advanced options toggle and are assigned default values as specified in the G4beamline manual.

Figure 4 shows a channel constructed in Beamline Architect and, below that, exported as a `.g4bl` file.

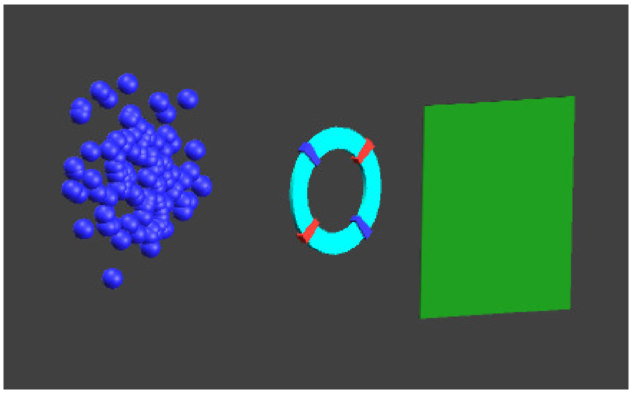


Figure 4: Example G4beamline channel constructed in Beamline Architect. Elements are a beam start, a magnetic quadrupole, and a detector.

```
#Physics
#-------
physics FTFP_BERT minRangeCut=1.0 maxTime=1000000

#Beam named Beam0
#----------------
beam gaussian particle=proton nEvents=1000 beamZ=-500 \
        sigmaX=100 sigmaY=100 meanMomentum=200
polycone Beam0 outerRadius=50.000,0 z=0,50.000 \
        color=0.000,0.000,1.000,1.000
place Beam0 z=-500

#MQ named MQ0
#------------
genericquad MQ0 apertureRadius=100 ironRadius=150 \
       ironLength=10 color=0.000,1.000,1.000,1.000
place MQ0 z=-123.7

#DET named DET0
#--------------
detector DET0 height=500 width=500 \
        color=0.000,1.000,0.000,0.500
place DET0 z=431.9
```

COSY INFINITY

COSY Infinity [2] is a powerful software language used for the simulation of particle optical and accelerator physics environments. This system uses the transfer map method and differential-algebraic (DA) techniques to allow for very efficient calculations and understanding of nonlinear motion. In its most basic level, it allows for the computation of the maps of standard beamline elements. This includes fringe fields, wedge absorbers and system parameters to arbitrary orders. It can also be very versatile and work for very large and complex mapping codes, which allows for many ways to efficiently manipulate and analyze the maps.

COSY Infinity's approach for the user interface is to phrase various tasks in terms of a standard scripting language environment called COSYScript. For users in need of special-purpose features, it can be very powerful, as it allows the use of DA and others as built-in types. Most commands are calls to previously defined procedures, and if desired, the user can create new commands by defining procedures of their own. The commands within COSY Infinity are simplistic and consist of a few letters which are abbreviations for the words that describe the action of the procedure. In addition to the commands that describe the particle-optical elements, there are commands that instruct the code on what to do.

COSY IMPLEMENTATION

Even though COSY is be fast and accurate, its interface can be somewhat cumbersome for students to learn, particularly those whose primary field is not accelerator physics. This this is because the systems code language must be manually written into a file and then executed on the command line. Beamline Architect as a tool provides a graphical user interface for COSY Infinity, letting users build particle accelerator channels in 3D. The user can decide to build with or without using code, then the channels can be saved, exported, translated, or run.

Some of the COSY Infinity elements that were implemented into Beamline Architect include: magnetic quadrupole and hexapole, superimposed multipole, a thick central magnetic solenoid (CMST), and fringe fields. Since fringe fields are modelled as Enge functions, an interface to construct Enge functions was also implemented. Likewise, a polynomial interface was implemented in preparation for adding wedge absorbers into Beamline Architect.

OTHER IMPLEMENTATIONS

There are several other advanced systems and features that are of little immediate consequence to the first-time user. These features are:

- **Variables.** There are several variable types that the user may use repeatedly. Examples of variable types that the user can create are: colors, Enge functions, and polynomials.
- **Error handling.** Users are commonly prevented from entering in text which may prevent the accelerator code (G4beamline, COSY) from functioning. Examples of data types that error handling pertains to are: positive numerals, vectors, valid G4beamline absorber files, and conflicting variable names.
- **Native manual description.** For every element, there exists a more detailed description from the native code's user manual.

CURRENT CHALLENGES

The greatest challenge thus far is the sheer amount of labor hours that the project requires. While the bulk of the project's logic is finished, there are only a few elements that are actually implemented.

One core tenet of Beamline Architect is that it is "code agnostic". Therefore, though the user may be able to construct and even run the simulation, there is currently no way to get useful information from it. An interface to Python is the proposed solution, but proving to be somewhat difficult.

REFERENCES

[1] Tom Roberts. G4beamline. `http://www.muonsinternal.com/muons3/G4beamline`, 2014. Version 2.15w.

[2] M. Berz and K. Makino. *COSY Infinity Beam Physics Manual*, 2013. Version 9.1.