A Geant4 simulation for the CrystalBall@MAMI Setup

Derek Glazier, University of Edinburgh

20th December 2006

1 Introduction

As advances in Geant3 particle tracking were discontinued a number of years ago, it is clear that to achieve as realistic a simulation as possible a move to Geant4 is required. In particular simulation of hadronic interactions is thought to be much improved with G4 and detailed validation exercises have been carried out. G4 uses object-orientated C++ programming.

The code included here contains the first (quite large) steps in producing a G4 simulation for the CB, TAPS, PID, MWPC detector setup for the first round of experiments at MAMI. Switching to the upcoming beamtime geometry should in principle be straightforward although is not currently implemented.

This report will outline the structure and the status of the simulation, named *A2*, as well as giving some instructions for installing and running, and finally list some thoughts on what still needs to be done.

2 Installation

Installation of this simulation requires a pre-installed release of G4. Users should refer to the G4 webpage for details. This simulation was developed using Geant4.8.1.p01 and I would recommend this version is installed. I have briefly tried compiling and running with Geant4.7.1, but the visualisation failed. In addition to the standard compilation users should also compile the hadronic physics list libraries. (from $G4INSTALL, cd physics_lists/hadronic, make. If you are unable to install these libraries then comment out the line #include "LHEP_BIC.hh" and replace runManager->SetUserInitialization(new LHEP_BIC); with runManager->SetUserInitialization(new A2PhysicsList); in A2.cc. And remove, include hadronic_lists.gmk from GNUMakefile) .

After the G4 system is installed the user is recommended to try and compile a few of the G4 examples.

The code package can currently be downloaded from http://nuclear.physics.gla.ac.uk/~dglazier/A2. The user can then unpack A2 into their G4WORKDIR and make. A2 uses a

ROOT based input/output and so the variable $ROOTSYS should be defined. The executable A2 should be found in $G4WORKDIR/bin/$G4SYSTEM/.

How to run the simulation is explained in sec. 5.

3 The A2 Package

The A2 directory contains all the additional code required to make the simulation work including some example macros for running the simulation in either batch or interactive mode. The directories contain :

docs	the user manuals, currently this document!
cbsim	relevant files from cbsim and the ROOT conversions
macros	macro files to control the batch and interactive running of the simulation
include	the .hh files for the new A2 classes
src	the .cc files for the new A2 classes
pics	some graphics created by A2

The file A2.cc controls the simulation by creating the A2 classes and assigning them to the G4 manager classes. It then allows an interactive session to be run if no initial macro is given, i.e just the command A2 is given. Or runs in batch mode if a run macro is given, e.g. A2 macros/doppi0.mac.

The files GNUmakefile and hadronic_lists.gmk are required for compilation. The file G4History.macro is produced by A2 and saves the interactive commands given.

The files CrystalConvert.in and taps.dat come from cbsim and allow a meaningful readout of the detectors.

4 Geometry

The A2 class A2DetectorConstruction is responsible for all of the geometry built. Currently it can call on three detector setups the Crystal Ball, TAPS and the PID. The Wire Chambers are still to be implemented.

Each detector has it's own class and A2DetectorConstruction decides whether to build it based on a setup macro, see sec.5.1.

A2DetectorConstruction is also responsible for creating all of the materials used in building the detectors.

4.1 The Crystal Ball

The geometry used for the Crystal Ball and surrounding material was taken from the cbsim file of Sasha. This can be found in the A2/cbsim directory as ugeom_sasha.F.

4.1.1 Software import from ugeom.F

Two methods were tried to export the crystal ball geometry directly from the cbsim file ugeom.F

The first was to use the G3toG4 utilities which are part of the G4 libraries. This requires an .rz file of the geometry to be produced by cbsim[1]. The first problem with this method was the Trap shape used in G4 does not accept sides with zero length, which had been used in cbsim to construct the triangular faces of the crystals. However Ken Livingston showed that this could be bypassed by a simple fix in the G4Trap class. After this G4 was able to produce a reasonable visualised likeness to the CB and TAPS, however attempts to track particles through the CB failed as they never underwent any interactions.

Fortran .rz file can be converted to ROOT readable format via the ROOT utility rz2root, so a second attempt to pass cbsim to G4 was made via ROOT's xml interface. This failed due to a mismatch in the syntax of the ROOT xml output and G4 xml input files. However even if such a scheme had been successful it almost certainly would have also failed to track the particles (for reasons explained below).

4.1.2 Manual import

After attempting the two software conversions I decided the best solution would be to manually rewrite the Fortran code in C++ using the G4 libraries. This was realised with the A2DetCrystalBall class. When writing the detector I used both the files ugeom_sasha.F and cbsasha.C (the ROOT conversion) in A2/cbsim for guidance. To simplify matters for myself I decided not to rewrite the prism.F function, which calculates the crystal, major and minor triangle parameters, but to just take the calculated results from the exported ROOT file cbsasha.C.

Initially I followed the cbsim recipe for placing the crystals first in Minor triangles, then the Minors in Major triangles, then the Majors in Hemispheres, then the Hemispheres in the Ball Volume. The crystals were again given a Trap volume, however instead of setting one side=0cm I have instead set it to 1nm, so G4 will not complain about the zero length.

This produced a CB that looked as expected. However, again when I fired the particles no interactions occurred in the ball!

The problem lies with the way G4 handles overlapping volumes. The fact the two hemispheres nearly exactly overlap meant G4 was unable to determine which hemisphere's daughter volumes, i.e Major triangles, it was in.

To overcome this I got rid of the 2 hemisphere volumes and place the Major triangles directly in the ball volume and applied the rotation and translation of the hemispheres directly to the Major Triangles. This appeared to give success as particles fired at the ball now interacted.

[1] To do this you need to add the line: call grfile(21,"crystalball.rz","ON") after call GGCLOS in ugeom.F

However it turned out the problem still existed close to the equator of the ball, the reason being that Major triangles here exist for both the upper and lower halves.

My eventual solution to the overlaps problem was to place each crystal individually into the World volume. Each crystal is rotated and translated by the product of the transforms for it's Minor, Major and Hemisphere.

Every crystal now tracks particles!

The next issue to resolve was for the cut crystals in the tunnel regions. Previously this was realised by placing one overriding volume CCUT in the ball. For G4 I chose to subtract off this volume from each individual crystal before placing it in the World volume. This is done via the G4SubtractionSolid class, and just needs the same translation and rotation information as used in placing the crystals and so prove to be relatively straightforward. Unfortunately the visualisation software I have tried has not been able to draw these crystals with the volumes cut to check the procedure. I was able to prove to myself that it did work using the G4UnionSolid class and by firing particles into the region that should be cut.

4.2 TAPS

The TAPS detector was implemented from a straight interpretation of the ugeom_taps.F code. The class is called A2DetTAPS. One slight change I have made is to readout every taps veto detector that fires, previously it was only read if it's corresponding BaF2 crystal fired. Someone can inform me if I should change back.

Figure 1 shows a visualisation of the CB-TAPS detectors.

4.3 The PID

The implementation for the PID varies from the original, as a more realistic right angular wedge shape is used for the scintillator shape, rather than a symmetric trapezoid. The effect of the change should be small.

5 Running the simulation

5.1 Detector Setup

Is controlled via the macros/DetectorSetup.mac file. Users can edit this to build any subset of detector systems, by switching the 1's to 0's. By default :

/A2/det/useCB 1
/A2/det/useTAPS 1
/A2/det/usePID 1

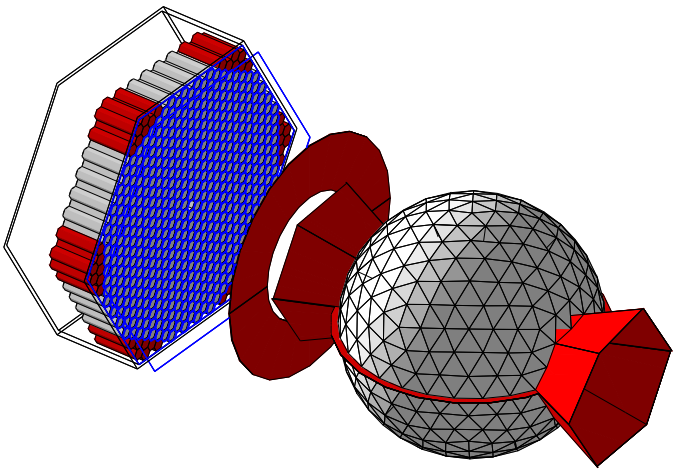

Figure 1: A picture of the CB-TAPS setup using DAWN visualisation. Unfortunately the drawing is not perfect, but this is due to the visualisation driver not colouring the right areas and not the actual geometry. Note, TAPS vetos are in blue and dummies are in red.

5.2 mkin Input

Users should obtain an mkin event generator file containing the reaction particles of interest. This file should be converted to ROOT via the h2root utility. 2 examples of such files are available alongside this release, kin_pi0p_100000.root and mkin_pin_400mev.root. The setup of A2 to read such files can be done either in batch or interactive mode using the same commands. Using the reaction $\gamma p \rightarrow p\pi^0$ as an example (e.g. kin_pi0p_100000.root), the command sequence follows :

The number of particles to be tracked, 1 proton 2 photons
/A2/generator/NToBeTracked 3
The mkin indexes of the particles to be tracked
The proton
/A2/generator/Track 2
The 2 photons
/A2/generator/Track 3
/A2/generator/Track 4
Note, the π^0 will not be tracked as it's index 1, is not given.
The mkin file, converted to root, to be read
/A2/generator/InputFile kin_pi0p_100000.root
The particles in the file and those to be tracked will be output to the screen:
A2PrimaryGeneratorAction::SetUpRootInput You have chosen to track 3 generated particles.
A2PrimaryGeneratorAction::SetUpRootInput, adding a pi0 as index 1
A2PrimaryGeneratorAction::SetUpRootInput, adding a proton as index 2
A2PrimaryGeneratorAction::SetUpRootInput, adding a gamma as index 3
A2PrimaryGeneratorAction::SetUpRootInput, adding a gamma as index 4
Will track proton with index 2
Will track gamma with index 3
Will track gamma with index 4

5.3 Interactive Mode

This allows visualisation of the detectors and supports standard G4 gun/ commands to define the beam or can read an mkin generated file. To run in this mode simply type A2 from the A2 directory. Any detector set in the DetectorSetup.mac is drawn in the World volume. The initial visualisation setup is defined in macros/vis.mac. Currently I prefer the OGLIX style viewer, which requires command line or macro based commands. Additionally the OGLIXm viewer (comment in line /vis/open OGLIXm) allows changes in the picture using a GUI system, although this can crash if too many commands are given. Users are encouraged to try the many other viewers available.

To run an mkin event follow the commands in sec. 5.2. Alternatively define a beam using the gun/commands :

/gun/particle proton
/gun/energy 100 MeV

/gun/direction 1 0.1 0

With either method then type /run/beamOn N to run N events. The tracks will be drawn on detector picture. In the case of the ball the crystals hit will be coloured with an intensity depending on the energy deposit. Alternatively the command /A2/event/drawHitOpt option will change the colour coding to depend on either the (option=)time or (option=)depth of the hit. This command has to be run prior to the event. In the case of TAPS and the PID currently the whole detector will be coloured if any element is hit.

For a list of all possible commands users can type help at the Idle> prompt. This includes all standard G4 commands for visualisation, runs, events, geometry... As well as the new commands for A2, found in the A2 help directory.

5.4 Batch Mode

Running in batch mode is straightforward, the executable is given a series of commands by a macro file which are implemented before the run. Currently this will contain the mkin setup commands from sec. 5.2, and an output file to write the tracked ntuple. The latter is done via the command :

/A2/event/setOutputFile test.root

For example to track the mkin input file kin_pi0p_100000.root, change the path to this file in macros/doppi0.mac to wherever you have saved the file and give a suitable output file name. The from the A2 directory type :

A2 macros/doppi0.mac

A large amount of set up info will scroll down the screen giving info on the geometry, physics processes etc. If the output file does not exist the simulation will start tracking. If it does exist the user will be prompted to overwrite or supply a new file name.

On my laptop the example above takes around 1 hour to run.

The output file can then be read into AcquRoot as in the case of the cbsim output.

6 Things to be done...

- Include the wire chambers. This should be done with some detector readout capability. Something like a twisted tube section might do the trick.
- Include the target geometries.
- Include PID support rings.
- Include TAPS and PID 2007
- Optimise tracking algorithm via particle threshold cuts and G4Regions.
- Investigation of different physics processes.
- Validation exercises, e.g. particle detection efficiencies, neutron energy deposits...