Detector simulation with GSTAR and root4star

Tutorial at the Dubna regional STAR meeting, November 2003

Mikhail Kopytine (Kent State University)

Key design decisions made in STAR MC:

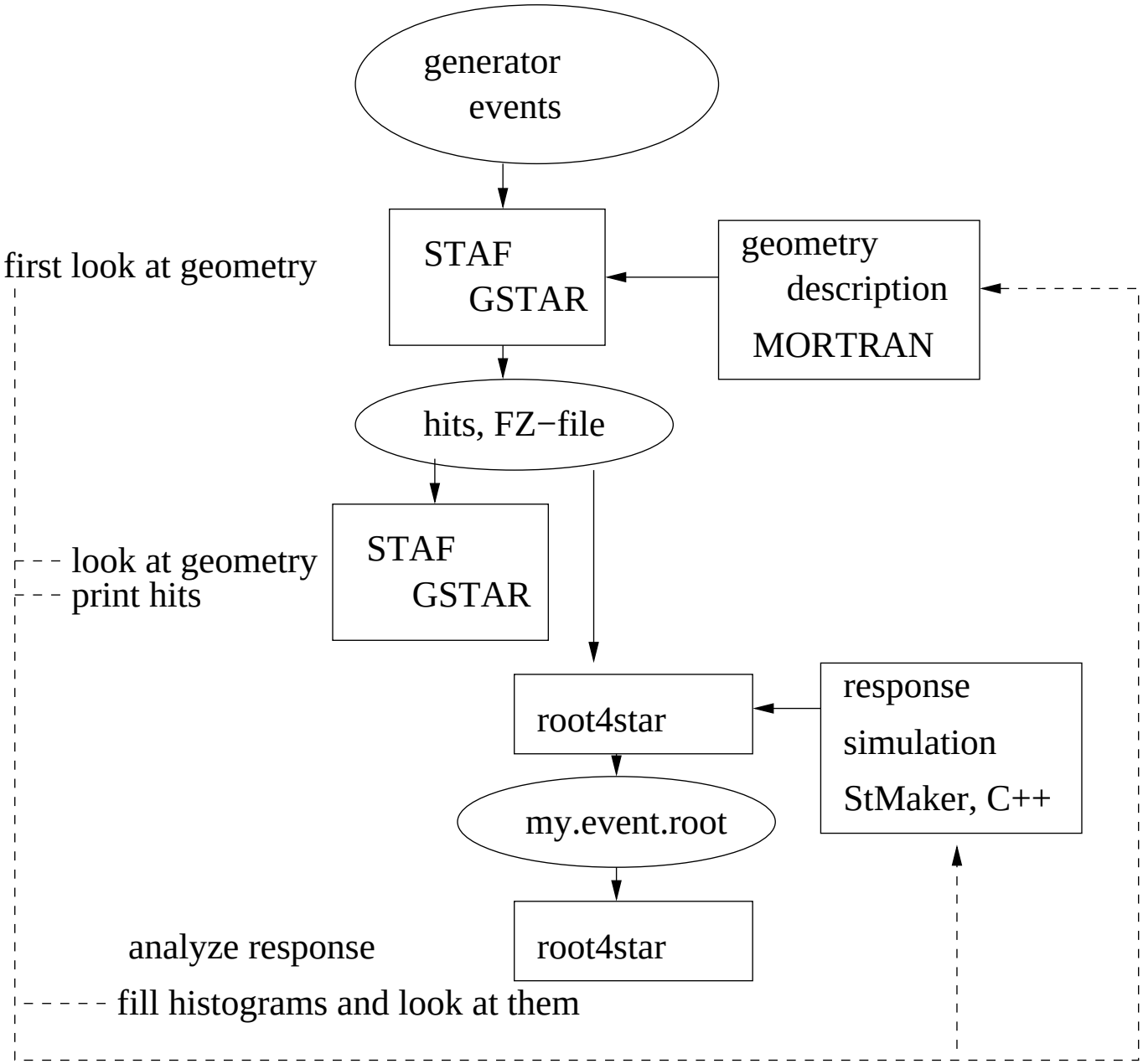
• GEANT 3.21 ⇒ age of FORTRAN: CERNLIB, PAW, ZEBRA (old good products)

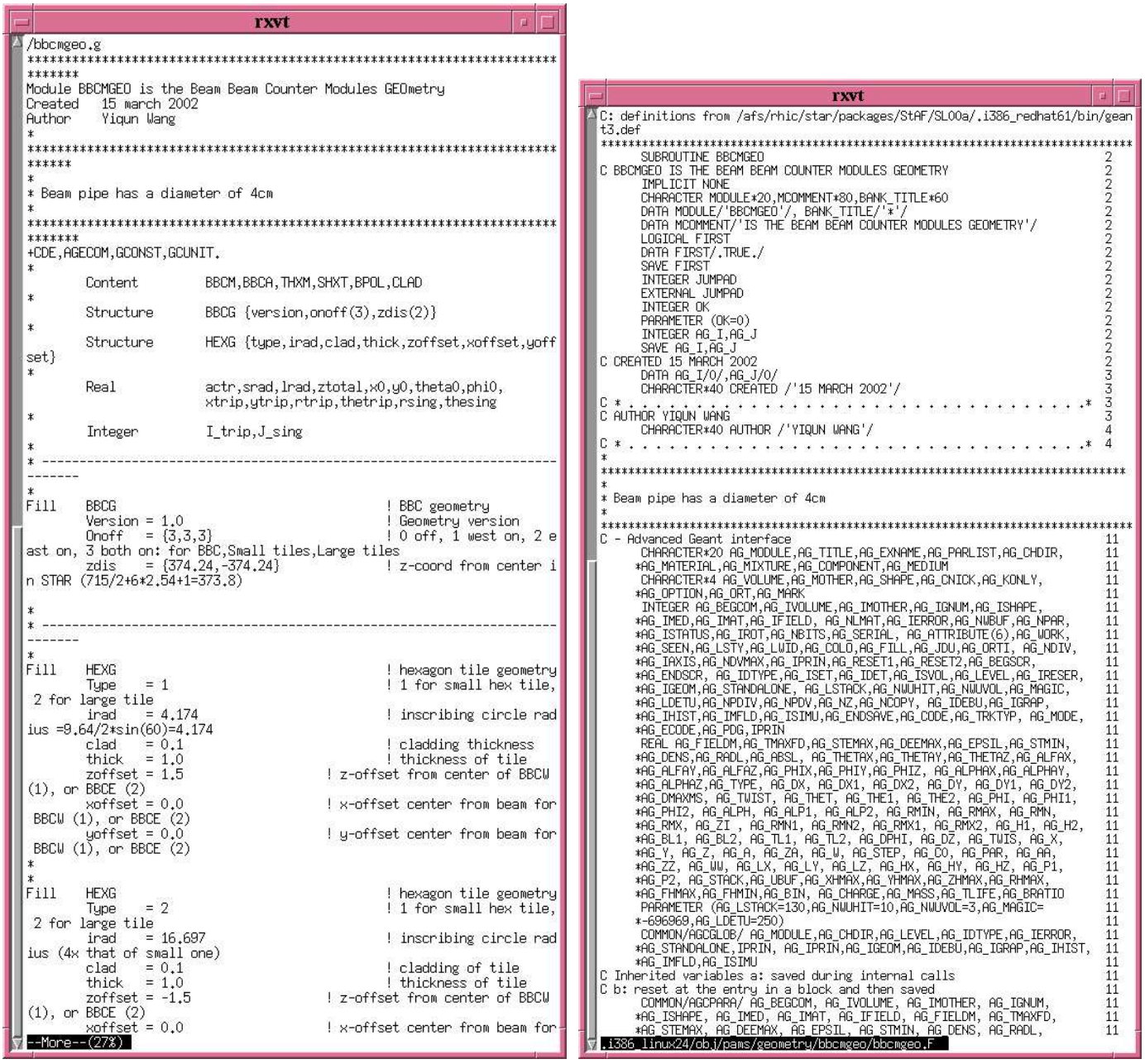
• detector geometry description relies on MORTRAN: a public domain FORTRAN preprocessor for structured programming.

• use root4star environment ("Big Full Chain" or BFC) for detector response (hit digitization) ⇒ independent second stage, based on StMaker (C++).

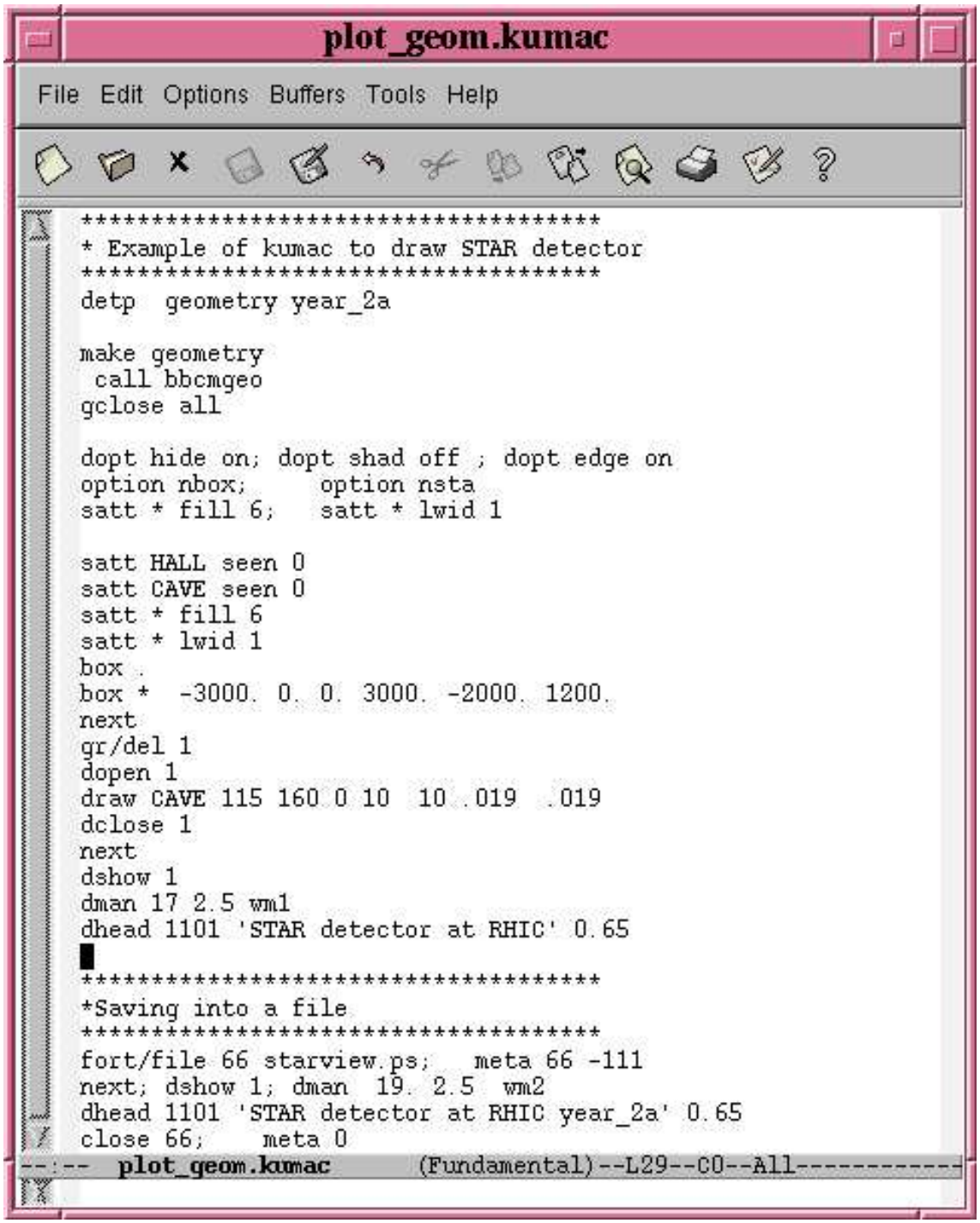
⇒ heterogeneous system; "hidden" tools to take care of legacy FORTRAN/C code; things to learn for non-ambidextrous FORTRAN/C++ users.

This talk will show you how to make these steps in the code development:



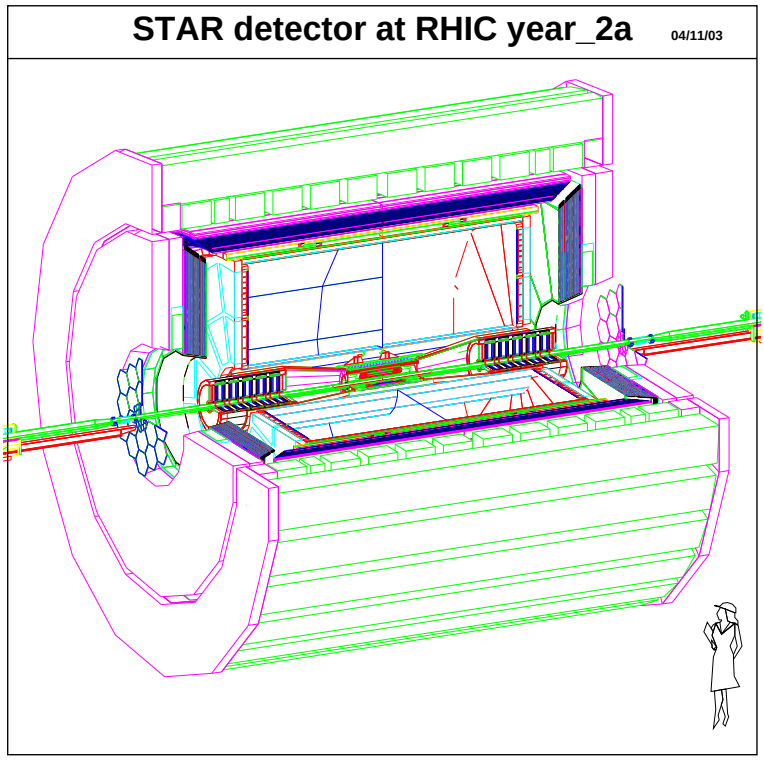


Detector description with Advanced Geant User Interface (left) and a machine-generated one in FORTRAN (right).



KUIP: Same syntax for command line typing and scripting.

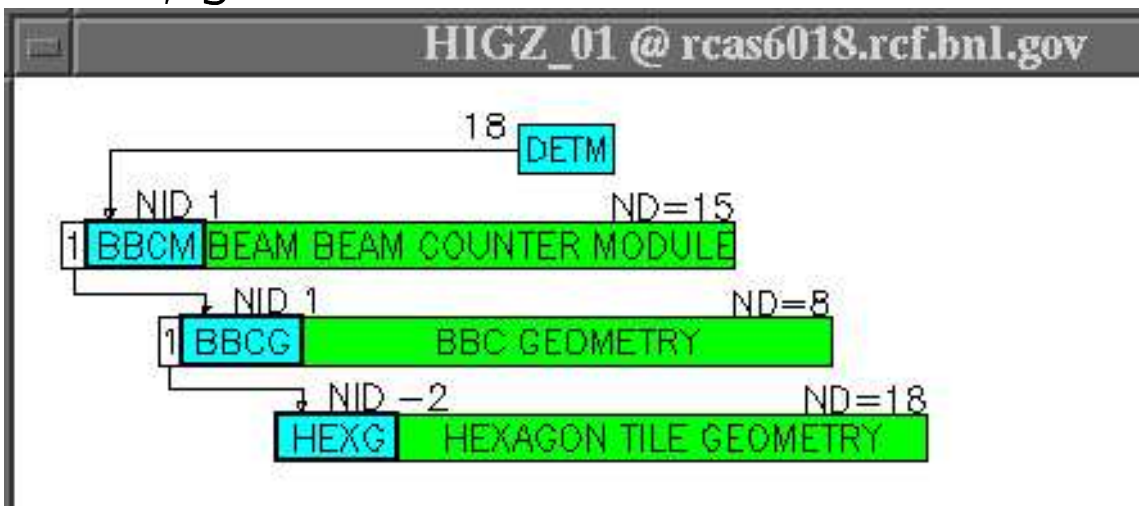
```
rxvt
rcas6018 /<4>kopytin/trigger/trigger/dAu/> staf -w 1
 ************************************************************
 * Starting     staf NwGEANT= 20000000 NwPAW= 2000000 *
 ************************pid=  4549************************
 ************************************************************
 *                                                          *
 *            W E L C O M E    to   staf                    *
 *                                                          *
 *        Version  2.00/04     7 October 2003               *
 *                                                          *
 ************************************************************
 Version 1.29/03 of HIGZ started
1*****  GEANT Version  3.21/14 Released on 19032002
0*****  Correction Cradle Version  0.1400
 *****  Initialising Staf libraries
 *****  RZMAKE. OLD RZ format selected for RZDOC
staf > exec plot_geom.kumac
 YEAR_2A: OLD ASYMPTOTIC STAR
```

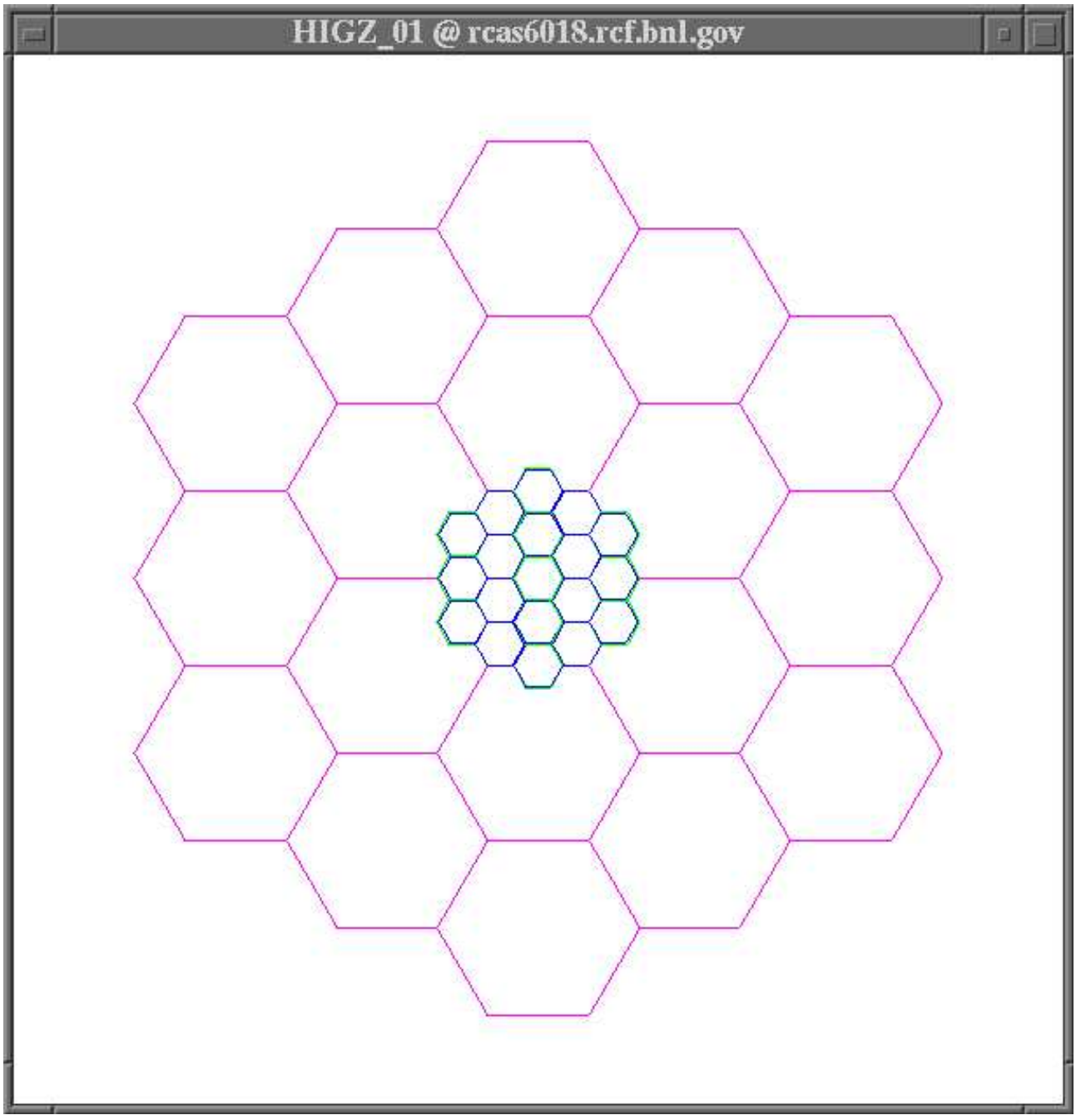


Looking at geometry in GSTAR:

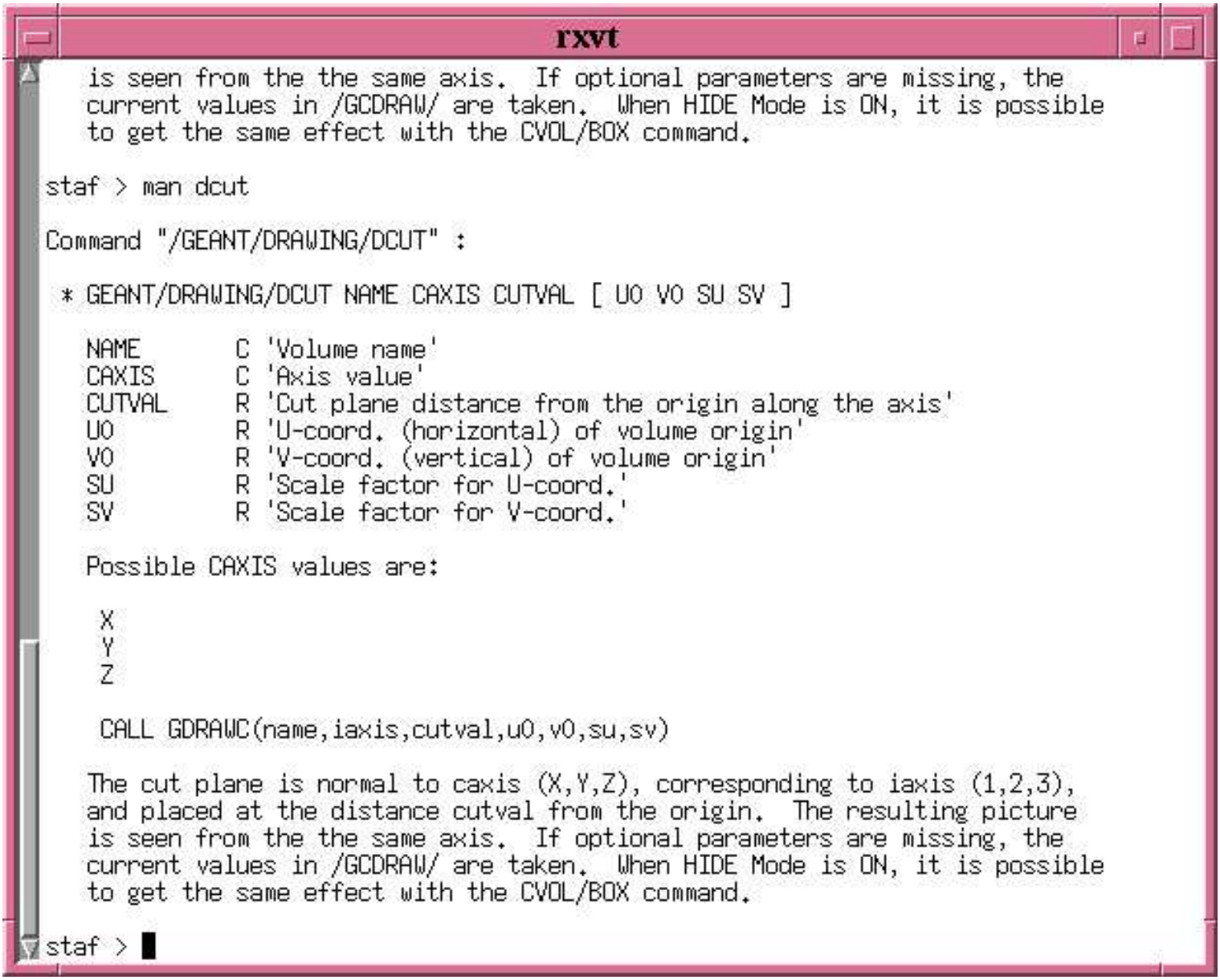
`staf > disp detm`

Use on-screen controls to navigate among sub-systems, go to BBC:

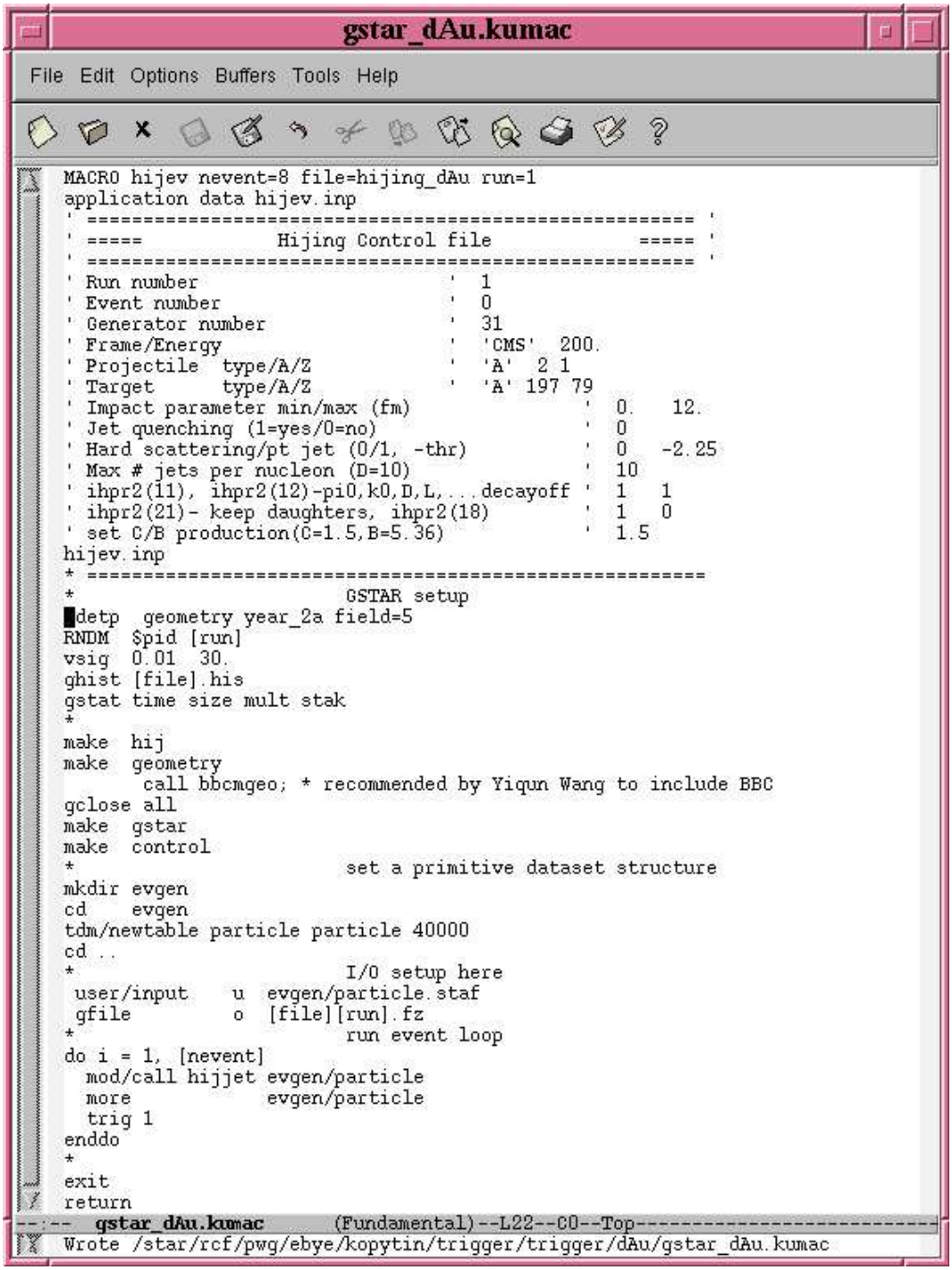





```
staf >  dcut BBCM Z 374.3  10  10 .1  .1
```



Use 'man' to learn more about dcut or any other command.



HIJING events can be generated “on the fly” during the GSTAR session.

An alternative: "teach" the event generator to store events in a STAR-wide standard "HEPTUP" format, based on a Row-Wise Ntuple – done for popular generators by Ron Longacre.

`$STAR/pams/gen/heptup.f`

implements the format.

```
* The following explanation was provided by Pavel Nevski:
* call HEPEvent
* (generator, run, Npart, B,F,Et,At, A1,Z1,A2,Z2 ) -new event
*     B -impact parameter
*     F -reaction plane angle
*     Et - energy
*     At - number of jets in HIJING, as used by Ron Longacre
* call HEPPart  (ipa,ist,pdg, moth,idau,pp, Ep,Am,vv,vt)
* - write new particle
*   ipa - particle number, 1-npart
*   ist - status (stable - 1, unstable -2 )
*   pdg - Particle data book code (pythia & hijing use it)
*   moth(2) - numbers(ipa) of this particle's parents
*   idau(2) - numbers(ipa) -""-                daughters
*   pp(3) - momentum px,py,pz
*   Ep    - energy
*   Am    - mass (particles may be off the mass shell)
*      momentum,energy and mass - in GeV
*   vv(3) - place of birth, vt - time of birth in mm and mm/c
*  can be simply 0*
* call HEPEnd (option) -close ntuple; compress on "z" option
```

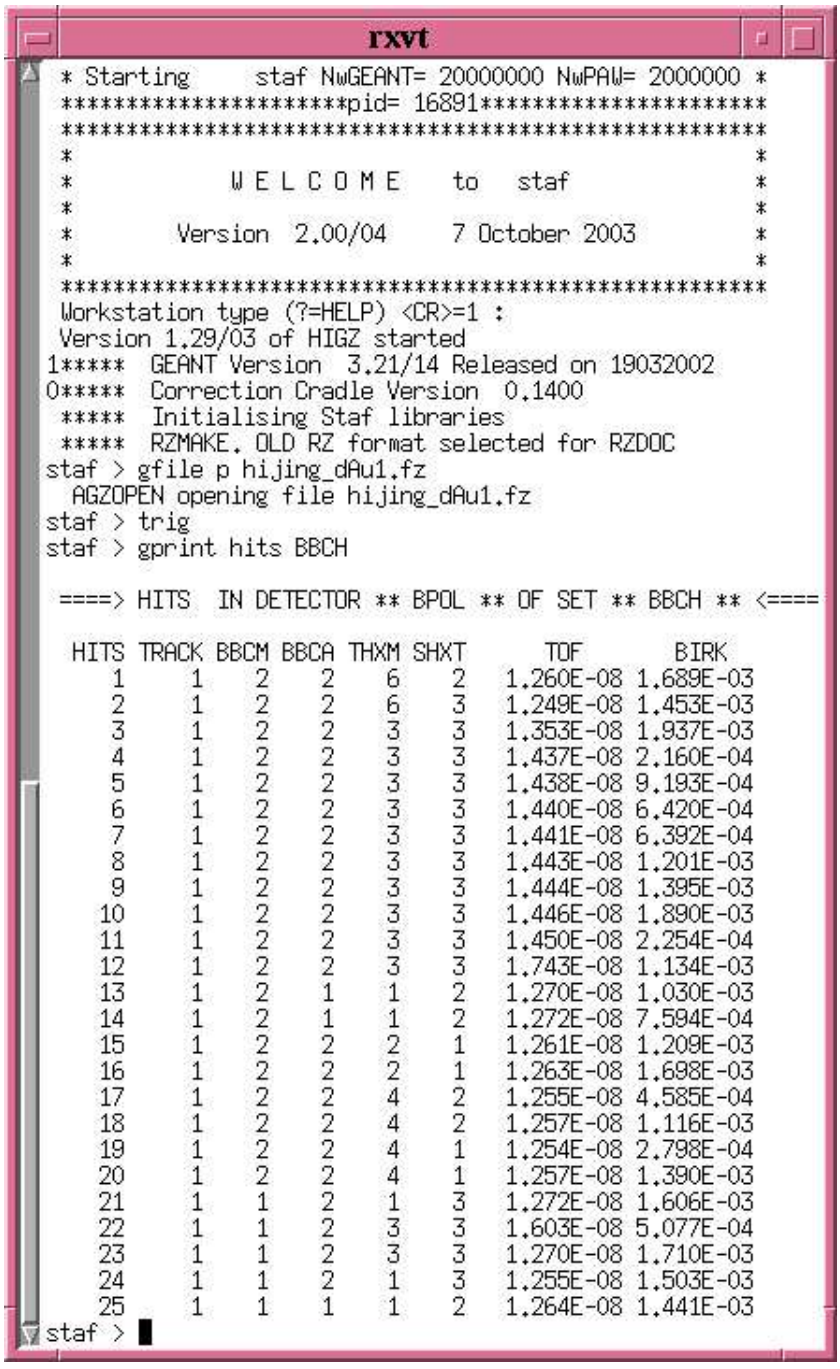
Read the events file with

```
staf> user/input u evgen.1.nt
```

Output from GSTAR – I use FZ; it can be read and fed into the "Big Full Chain" for hit digitization.

Example of a shell script to do response simulation, based on GEANT hits:

```
#! /bin/tcsh
root4star -q -b 'bfc1.C(1,100000,\
"C2003 tfs fzin big mwc trg halfField EmcSim",\
"hijing_dAu1.fz","hijing_dAu1")'
# This will create file hijing_dAu1.event.root,
# subject to standard analysis
```



Need more cross-checks? read FZ file with events into STAF. Geometry is kept in the file (persistent) ⇒ have full access (can draw). Hits are saved. Use `trig` to move from event to event, `gprint` to print hits.

Steps needed for a full integration of a detector into GEANT and response simulation.

Step 1: Add the geometry and materials description: `pams/geometry/bbcmgeo/bbcmgeo.g`

Step 2: Select the appropriate hit type for the detector. For the BBC, it is a calorimetric hit with time-of-flight information; therefore, I used `g2t_ctf_hit`. `pams/sim/g2t/g2t_bbc.F` and `pams/sim/g2t/g2t_bbc.idl` are new files which define the structure of the hit.

Step 3: `g2t_volume_id.g` needs to be modified. This function returns and defines **volume ID** with a separate scheme for every subsystem; response simulation needs to know about the scheme and use it consistently when it makes decisions about volume elements. In GEANT, volume ID is encoded as an array representing the path to the volume, where array positions correspond to the levels of the hierarchical tree of volumes, and the array elements are IDs of the embedding volumes. Typically, a volume ID encodes this hierarchical information within a single integer number where different decimal places correspond to the levels of the tree. See example...

Diff for /pams/sim/g2t/g2t_volume_id.g between version 1.37 and 1.38

version 1.37, 2001/09/06 00:22:00	version 1.38, 2002/10/16 19:12:44
Line 1	**Line 1**
* Id * Log	* Id * Log
	* Revision 1.38 2002/10/16 19:12:44 kopytin * Volume ID for BBC elements added. If changed, will affect StBbcSimulationMaker. *
* Revision 1.37 2001/09/06 00:22:00 nevski * new svt geometry numbering intrroduced *	* Revision 1.37 2001/09/06 00:22:00 nevski * new svt geometry numbering intrroduced *
Line 530	**Line 533**
+ numbv(4)*100 + numbv(5)	+ numbv(4)*100 + numbv(5)
16	*16*
	else if (Csys=='bbc') then * Mikhail Kopytine for the BBC group * BBC has 4 levels: west/east, annulus, triple module, single module volume_id = numbv(1)*1000 + numbv(2)*100 + numbv(3)*10 + numbv(4) *17*

else
print *,' G2T warning: volume ',Csys,' not found '
endif
	else
	print *,' G2T warning: volume ',Csys,' not found '
	endif

Legend:
Removed from v.1.37
changed lines
Added in v.1.38

Colored Diff Show

Step 4: In the ROOT/StMaker (C++) land: `St_geant_Maker.cxx` needs to be modified to read the hits from the new subsystem. **Need to know** the name of the hit (BBCH for BBC) and name of GEANT sensitive element (BPOL for BBC). The infrastructure generates a wrapper `g2t/St_g2t_bbc_Module.h`, based on `g2t_bbc.F` and `g2t_bbc.idl` – do not forget to include it!

Diff for /StRoot/St_geant_Maker/St_geant_Maker.cxx between version 1.73 and 1.74

version 1.73, 2002/08/26 14:21:21	version 1.74, 2002/10/16 20:39:23
Line 1	**Line 1**
// $Id: St_geant_Maker.cxx,v 1.37 1999/04/19 06:29:30 nevski Exp	// $Id: St_geant_Maker.cxx,v 1.37 1999/04/19 06:29:30 nevski Exp
// Id	// Id
// Log	// Log
// Revision 1.73 2002/08/26 14:21:21 jeromel	// Revision 1.74 2002/10/16 20:39:23 kopytin
// changing 'PHMD' to 'PHMH'	// Added code to read out BBC GSTAR tables. Needed by StBbcSimulationMaker
//	//
// Revision 1.72 2002/06/17 16:12:43 perev	// Revision 1.72 2002/06/17 16:12:43 perev
// fix wrong geant time	// fix wrong geant time
Line 296	**Line 296**
#include "g2t/St_g2t_zdc_Module.h"	#include "g2t/St_g2t_zdc_Module.h"
#include "g2t/St_g2t_vpd_Module.h"	#include "g2t/St_g2t_vpd_Module.h"
#include "g2t/St_g2t_pmd_Module.h"	#include "g2t/St_g2t_pmd_Module.h"
	#include "g2t/St_g2t_bbc_Module.h"
#include "StarCallf77.h"	#include "StarCallf77.h"
#ifdef F77_NAME	#ifdef F77_NAME
#define geometry F77_NAME(geometry,GEOMETRY)	#define geometry F77_NAME(geometry,GEOMETRY)
Line 636	**Line 640**
iRes = g2t_zdc(g2t_track,g2t_zdc_hit);	iRes = g2t_zdc(g2t_track,g2t_zdc_hit);
// ==========	// ==========
}	}
	geant3->Gfnhit("BBCH","BPOL", nhits);
	if (nhits>0)
	{
	St_g2t_ctf_hit *g2t_bbc_hit = new St_g2t_ctf_hit("g2t_bbc_hit",nhits);
	m_DataSet->Add(g2t_bbc_hit);
	iRes = g2t_bbc(g2t_track,g2t_bbc_hit);
	// ==========
	}
//----------all bloody detectors done----------//	//----------all bloody detectors done----------//

Step 5: retrieving data from a "table" in the digitization phase...

```
//_____________________________________________________________________
Int_t StBbcSimulationMaker::Make()
{
/// Make - this method is called in loop for each event

 TDataSet* ds= GetInputDS("geant");
 StEvent* event = (StEvent*)GetInputDS("StEvent");
 St_g2t_ctf_hit* g2t_bbc_hit = (St_g2t_ctf_hit*)ds->Find("g2t_bbc_hit");

 if (g2t_bbc_hit)
   {
     short nBBChits = g2t_bbc_hit->GetNRows();
     BbcTOF TOFdata;
     BbcDE DEdata;
     for (short iBBChit=0; iBBChit<nBBChits; iBBChit++)
       {
         float De = g2t_bbc_hit->operator[](iBBChit).de;
         float TOF = g2t_bbc_hit->operator[](iBBChit).tof;
         short Vid =  g2t_bbc_hit->operator[](iBBChit).volume_id;

         short PMTid = Geant2PMT[Vid]-1;


         DEdata.AddDE(PMTid,De);
         TOFdata.AddTOF(PMTid,TOF);
       }
```

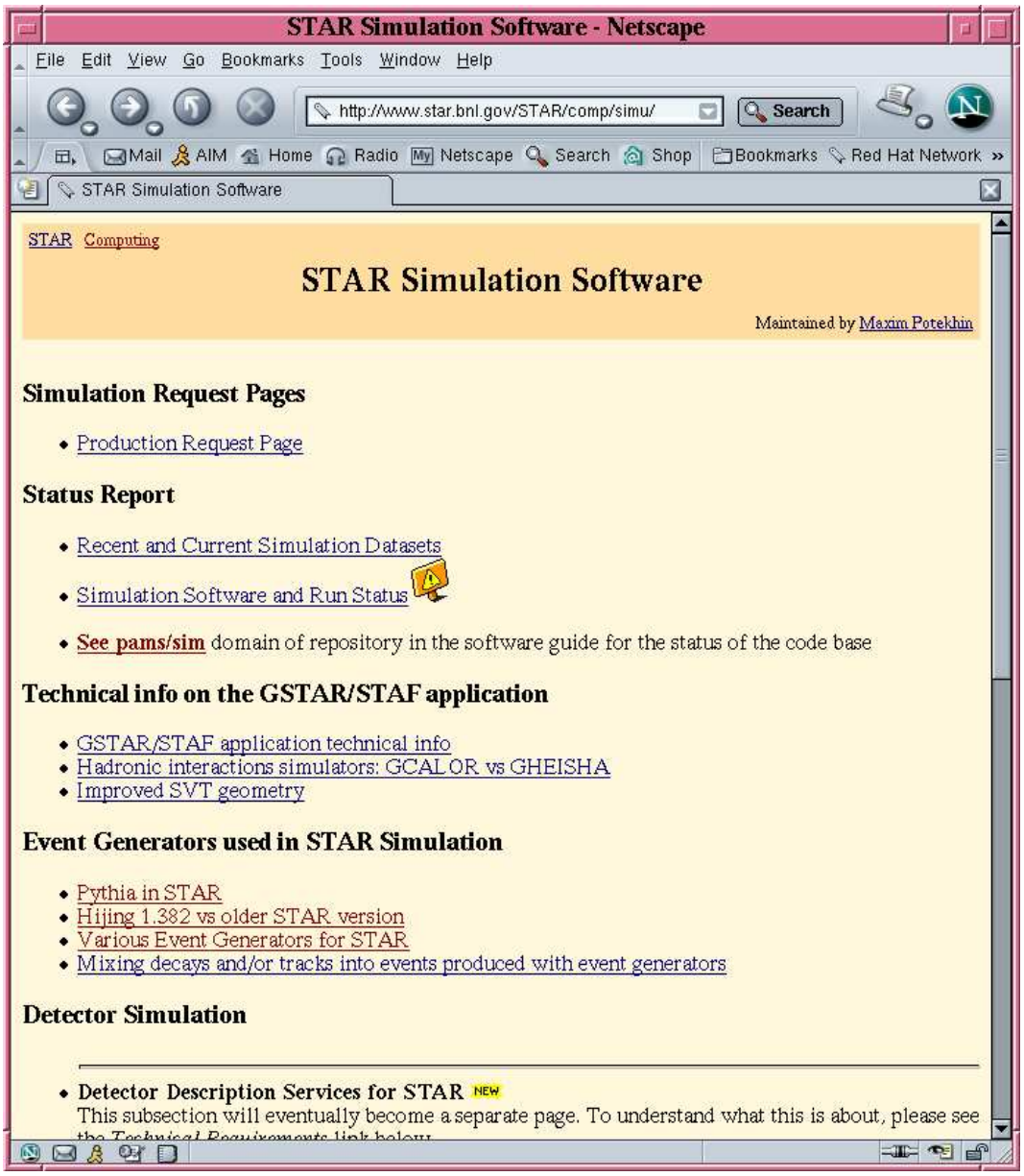
... and filling StEvent data structures (imitating calibrated output).

```
     StTriggerDetectorCollection* myTrig = event->triggerDetectorCollection();
     StBbcTriggerDetector& myBbc = myTrig->bbc();


     for (u_short iPMT = 0; iPMT<NPMT2; iPMT++)
       {
          short ADC = DEdata.GetADC(iPMT);
#ifdef BbcSimQa
          //      QaBbcPmtAdc->Fill(iPMT,ADC);
          short Vid = PMT2Geant[iPMT+1];

          if (Vid<2000) {QaBbcWestVid->Fill(ADC);}
          if (Vid>2000) {QaBbcEastVid->Fill(ADC);}
          if (iPMT<NPMT1) {QaBbcEastPmt->Fill(ADC);}
          if (NPMT1<=iPMT && iPMT<NPMT2) {QaBbcWestPmt->Fill(ADC);}
          QaBbcPmtTime->Fill(iPMT,TOFdata.GetTOF(iPMT));
          QaBbcPmtdE->Fill(iPMT,DEdata.GetDE(iPMT));
#endif
          myBbc.setAdc(iPMT, ADC);
          if (IsSmall(iPMT))
            {
              short TDC = TOFdata.GetTDC(iPMT);
              myBbc.setTdc(iPMT, TDC);
            }
        }
   }
else
  {
    gMessMgr->Info
      ("MLK StBbcSimulationMaker::Make() could not inst g2t_bbc_hit\n");
  }

 return kStOK;
}
```



http://www.star.bnl.gov/STAR/comp/simu/

Credits: Pavel Nevski, Maxim Potekhin.