Keeping tracking hit information in Monte Carlo

Benedikt Zihlmann

Purpose

- Store information about the particle that caused a hit in the FDC and CDC! So that later in track reconstruction it is possible to identify a given hit used to build a track candidate or later in track fitting.
- The information I want to know is the particle type and the particle track number (GEANT).
- Use this to determine how many hits in a reconstructed track are caused by the reconstructed particle, how many hits are noise hits and how many hits were caused by other particles? Has one true track been split into several track candidates?

Monte Carlo basics

Do book keeping each track at each step:
In GUSTEP.F:

```
* Register hits in sensitive detectors here

  if (ISVOL.ne.0.or.ISTOP.ne.0.or.INWVOL.ge.2) then
    call savehits
  endif
```

- call gustep.F for each tracking step
- call savehits.F book keeping
  - hitFDC.c
  - hitCDC.c
  - hitFCAL.c
  - and more.....
- leaving sensitive Volume record data

FDC data

Code for FDC data recording:

```
if wire never got hit before then:
----------------------------------
wires->in[0].fdcAnodeTruthHits = ahits = make_s_FdcAnodeTruthHits(MAX_HITS);

or wire got hit before:
-----------------------
ahits = fdc->fdcChambers->in[0].fdcAnodeWires->in[0].fdcAnodeTruthHits;

then record the hit information:
--------------------------------
ahits->in[nhit].t = tdrift;
ahits->in[nhit].t_unsmeared=tdrift_unsmeared;
ahits->in[nhit].d = dradius;
ahits->in[nhit].dE = dE;
ahits->in[nhit].itrack = track;
ahits->in[nhit].ptype = ipart;

and the same thing for the strips:
----------------------------------
chits->in[nhit].t = tdrift;
chits->in[nhit].q = q;
chits->in[nhit].itrack = track;
chits->in[nhit].ptype = ipart;
```

FDC data

Code for CDC data recording:

```
if the straw was never hit before:
----------------------------------
straws->in[0].cdcStrawTruthHits = hits = make_s_CdcStrawTruthHits(MAX_HITS);


or if the straw was hit before:
-------------------------------
hits = cdc->cdcStraws->in[0].cdcStrawTruthHits;

hits->in[nhit].t = tdrift;
hits->in[nhit].dE = dEsum;
hits->in[nhit].d = dradius;
hits->in[nhit].itrack = track;
hits->in[nhit].ptype = ipart;
```

Note the word "Truth" in cdcStrawTruthHits!

For secondary particles “track” is zero unless it is put back on the stack and made primary

Put secondaries back on the stack and make the primaries

At each tracking step check for newly created particles (secondaries) and ...

```
c     make primary except if in calorimeter volume and not hadronic interaction
            iflgk(i) = 1
            cint = KCASE
c            print *, cchar
            rx = sqrt(VERT(1)**2+VERT(2)**2)
            if ( (((rx>65.) .and.((VERT(3)>-17.) .and.
     >           (VERT(3)<390.))).or.(VERT(3)>625.)).and.
     >           (cchar.ne.'HADR')) then
              iflgk(i) = 0
            endif
            call GSKING(i)
          endif
        enddo
       endif
```

$IFLGK(I) = 1$ means put track info also into *JKIN* data structure.

But don't do that for shower particles (calorimeter)!

Keeping track of primaries

Primaries: particles generated at the initial target vertex in DMCThrown

```
if (history == 0)
{
   int mark = (1<<30) + pointCount;
   void** twig = getTwig(&centralDCTree, mark);
   if (*twig == 0)
   {
      s_CentralDC_t* cdc = *twig = make_s_CentralDC();
      s_CdcTruthPoints_t* points = make_s_CdcTruthPoints(1);
      int a = thisInputEvent->physicsEvents->in[0].reactions->
                                              in[0].vertices->in[0].products->mult;
      points->in[0].primary = (stack <= a);
      points->in[0].track = track;
      points->in[0].t = t;
      points->in[0].z = x[2];
      points->in[0].r = sqrt(x[0]*x[0] + x[1]*x[1]);
      points->in[0].phi = atan2(x[1],x[0]);
      points->in[0].dradius = dradius;
      points->in[0].px = pin[0]*pin[4];
      points->in[0].py = pin[1]*pin[4];
      points->in[0].pz = pin[2]*pin[4];
      points->in[0].dEdx = dEdx;
      points->in[0].ptype = ipart;
      points->mult = 1;
      cdc->cdcTruthPoints = points;
      pointCount++;
   }
}
```

Data Model

MC Data structure for Drift Chambers:

```
<centralDC minOccurs="0">
  <cdcStraw maxOccurs="unbounded" minOccurs="0" ring="int" straw="int">
    <cdcStrawHit      dE="float" maxOccurs="unbounded" t="float" itrack="int"
                      d="float" ptype="int"/>
    <cdcStrawTruthHit dE="float" maxOccurs="unbounded" t="float" itrack="int"
                      d="float" ptype="int"/>
  </cdcStraw>
  <cdcTruthPoint dEdx="float" dradius="float" maxOccurs="unbounded" minOccurs="0"
                 phi="float" primary="boolean" ptype="int" px="float" py="float"
                 pz="float" r="float" t="float" track="int" z="float"/>
</centralDC>

<forwardDC minOccurs="0">
  <fdcChamber layer="int" maxOccurs="unbounded" module="int">
    <fdcAnodeWire maxOccurs="unbounded" minOccurs="0" wire="int">
      <fdcAnodeHit      dE="float" maxOccurs="unbounded" t="float" itrack="int"
                        d="float" ptype="int"/>
      <fdcAnodeTruthHit dE="float" maxOccurs="unbounded" t="float"
                        d="float" t_unsmeared="float" itrack="int" ptype="int"/>
    </fdcAnodeWire>
    <fdcCathodeStrip maxOccurs="unbounded" minOccurs="0" plane="int" strip="int">
      <fdcCathodeHit      maxOccurs="unbounded" q="float" t="float" itrack="int" ptype="int"/>
      <fdcCathodeTruthHit maxOccurs="unbounded" q="float" t="float" itrack="int" ptype="int"/>
    </fdcCathodeStrip>
    <fdcTruthPoint E="float" dEdx="float" dradius="float" maxOccurs="unbounded" minOccurs="0"
                   primary="boolean" ptype="int" px="float" py="float" pz="float" t="float"
                   track="int" x="float" y="float" z="float"/>
  </fdcChamber>
</forwardDC>
```

Data Model

MC Data structure for Drift Chambers:

```
<centralDC minOccurs="0">
  <cdcStraw maxOccurs="unbounded" minOccurs="0" ring="int" straw="int">
    <cdcStrawHit      dE="float" maxOccurs="unbounded" t="float" itrack="int"
                      d="float" ptype="int"/>
    <cdcStrawTruthHit dE="float" maxOccurs="unbounded" t="float" itrack="int"
                      d="float" ptype="int"/>
  </cdcStraw>
  <cdcTruthPoint dEdx="float" dradius="float" maxOccurs="unbounded" minOccurs="0"
                 phi="float" primary="boolean" ptype="int" px="float" py="float"
                 pz="float" r="float" t="float" track="int" z="float"/>
</centralDC>

<forwardDC minOccurs="0">
  <fdcChamber layer="int" maxOccurs="unbounded" module="int">
    <fdcAnodeWire maxOccurs="unbounded" minOccurs="0" wire="int">
      <fdcAnodeHit      dE="float" maxOccurs="unbounded" t="float" itrack="int"
                        d="float" ptype="int"/>
      <fdcAnodeTruthHit dE="float" maxOccurs="unbounded" t="float"
                        d="float" t_unsmeared="float" itrack="int" ptype="int"/>
    </fdcAnodeWire>
    <fdcCathodeStrip maxOccurs="unbounded" minOccurs="0" plane="int" strip="int">
      <fdcCathodeHit      maxOccurs="unbounded" q="float" t="float" itrack="int" ptype="int"/>
      <fdcCathodeTruthHit maxOccurs="unbounded" q="float" t="float" itrack="int" ptype="int"/>
    </fdcCathodeStrip>
    <fdcTruthPoint E="float" dEdx="float" dradius="float" maxOccurs="unbounded" minOccurs="0"
                   primary="boolean" ptype="int" px="float" py="float" pz="float" t="float"
                   track="int" x="float" y="float" z="float"/>
  </fdcChamber>
</forwardDC>
```

MC Data in DANA

Where does the MC data go in DANA?
Translation happens in

`src/libraries/HDDM/DEventSourceHDDMGenerator.cc`

- cdcStrawHit $\rightarrow$ **DCDCHit**
- fdcAnodeHit $\rightarrow$ **DFDCHit**
- fdcCathodeHit $\rightarrow$ **DFDCHit**
- **DMCTrackHit** list of all DetectorTruth**Point**

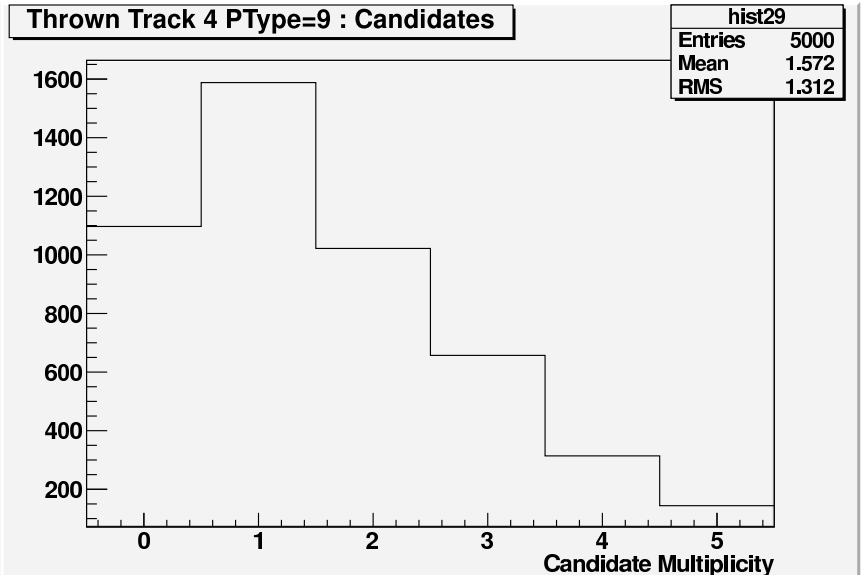
DMCTrackHits are matched as associated objects with CDCHits and FDCPseudoHits that are used o build track candidates.
Note: Matching is not 100%! For a 100% match would need XXXTruthHit as associated object.

Tracking diagnostics

Use plugin trackanal like:

```
hd_ana --plugin=trackanal foo.hddm
```

- Works ONLY with data where all events have the same topology!
- Output is a root tree that can be analyzed with check2 root code to generate histograms.
- Example: How many track candidates do I get from the hits of generated track number 4?



What to do?

A Only remove itrack and ptype from MC smeared data structure only.

B Remove itrack and ptype from MC data altogether (unsmeared and smeared)