Two months SPiRIT TPC Software development timeline

	week 1 (jun)							week 2 (jun)							week 3 (jun)							week 4 (jun)						
	mon 9	tue 10	wed 11	thu 12	fri 13	sat 14	sun 15	mon 16	tue 17	wed 18	thu 19	fri 20	sat 21	sun 22	mon 23	tue 24	wed 25	thu 26	fri 27	sat 28	sun 29	mon 30	tue 1	wed 2	thu 3	fri 4	sat 5	sun 6
Assigned to:																												
Prabi	Cluster Finding and Integration of GEANT4 by Jerzy into SPiRIT														Clusterizer, Drifting Task and Integrate for Digitization output													
Genie	Rieman Tracking and Initialization Tracks																					Kalman and Residual Task using GENFIT						
Mike	Geometry of TPC							Event Generation and transportation using GEANT4 Simulation																				
Jung Woo	Clusterizer and Drifting Tasks																					Avalanche and Pad response Tasks						
Fengfeng	Integration of TPC Geometry/Materials and Mixure into Jerzys GEANT4 code																					Event Genration and Transportation in E and B Field using GEANT4						

	Week 5 (Jul)							Week 6							Week 7							Week 8						
	mon 7	tue 8	wed 9	thu 10	fri 11	sat 12	sun 13	mon 14	tue 15	wed 16	thu 17	fri 18	sat 19	sun 20	mon 21	tue 22	wed 23	thu 24	fri 25	sat 26	sun 27	mon 28	tue 29	wed 30	thu 31			
Assigned to:																												
Prabi	Clusterizer, Drifting Task and Integrate for Digitization output							Kalman filter using GENFIT														Review & Present Work(15 min)						
Genie	Kalman and Residual Task using GENFIT																					Review & Present Work(15 min)						
Mike	Integrate GEANT4 tasks and produce MC Simulation output																					Review & Present Work(15 min)						
Jung Woo	Avalanche and Pad response Tasks														Electronics Task							Review & Present Work(15 min)						
Fengfeng	Event Genration and Transportation in E and B Field using GEANT4																					Review & Present Work(15 min)						

FOPI Digitization

1. **Clusterizing**
   - Clusterizing is done before drifting electron.
   - Points are merged if they are close enough
   - Group of electron is made with random size(within range) and a group is stored as “primary cluster”.
     - Physical?
2. **Drifting Electron**
   - Diffusion is applied to each primary cluster. Not to single electron.
   - Rest of the method is very much same with LAMPS TPC digitization.

BACKUP

Digi Tasks

- **SetPersistence()** : saves the data in output file

TpcClusterizerTask::Exec

```
115. void
116. TpcClusterizerTask::Exec(Option_t* opt)
117. {
118.   // Reset output Array
119.   if(fprimArray==0) Fatal("TpcPrimCluster::Exec)","No PrimClusterArray");
120.   fprimArray->Delete();
121.
122.   Int_t np=fpointArray->GetEntriesFast();
123.   if(np<2){
124.     int evNb=(FairRun::Instance())->GetEventHeader()->GetMCEntryNumber();
125.     TString warning=Form("TpcClusterizerTask::Exec ev:%i ",evNb);
126.     //Warning("TpcClusterizerTask::Exec","Not enough Hits in Tpc for Digitization (<2)");
127.     Warning(warning,"Not enough Hits in Tpc for Digitization (<2)");
128.     return;
129.   }
130.
131.   if(fmereChargeConversion)  {
132.      ChargeConversion();
133.      return;
134.      //goodbye, you wretched world!
135.   }
136.
137.   TpcPoint* point;
138.   TpcPoint* theLastPoint;
139.   Int_t icluster=0;
140.   theLastPoint= (TpcPoint*)fpointArray->At(0);
141.
142.   for(int ip=1;ip<np;++ip){
143.     point=(TpcPoint*) fpointArray->At(ip);
144.     //point->Print();
145.
146.     // check if points are not too far from each other
147.     TVector3 p1;point->Position(p1);
148.     TVector3 p2;theLastPoint->Position(p2);
149.
150.     TVector3 d=p1-p2;
151.     if(d.Mag()>1){
152.       theLastPoint=point;
153.       continue;
154.     }
```

: check how many points there are from MC simulation. If number of points are less than 2, return and exit digitization.

: **'fmereChargeConversion' is by default false**. Can be set with SetMereChargeConversion(). if 'fmereChargeConversion' is true, Run ChargeConversion() and return. So rest of the Exec is not used. if fmereChargeConversion==kFALSE ↓

: define two points to compare; 'point' and 'theLastPoint'.
>> theLastPoint = TpcPoint(0)

: loop over all the TPC points from MC simulation.
>> point = TpcPoint(1)
position of 'point' and 'theLastPoint' is set as p1 and p2.

: calculate magnitude of (p1-p2), which is magnitude of vector between current point and last(just before) point. if magnitude of two points are larger than 1(unit is in mm(?)), two points are far enough so, continue 'for' loop. If they are smaller than 1, that is they are too close, so go on.

```
155.    //check if hits ly on the same track
156.     if(point->GetTrackID()==theLastPoint->GetTrackID()){
157.       double dE=point->GetEnergyLoss()*1E9; //convert from GeV to eV
158.
159.       //Step 0: calculate the overall amount of charge, produced
160.       if(dE<0){
161.         Error("TpcClusterizerTask::Exec","Note: particle:: negative Energy loss!");
162.         theLastPoint=point;
163.         continue;
164.       }
165.       unsigned int q_total =(unsigned int)floor(fabs(dE / fgas->W()));
166.       unsigned int q_cluster=0;
167.       unsigned int ncluster=0;
168.       //Step 1: Create Clusters
169.
170.       //while still charge not used-up distribute charge into next cluster
171.
172.       while(q_total>0){
173.         //roll dice for next clustersize
174.         q_cluster=fgas->GetRandomCS(gRandom->Uniform());
175.         if(q_cluster>q_total)q_cluster=q_total;
176.         q_total-=q_cluster;
177.         // create cluster
178.         Int_t size = fprimArray->GetEntriesFast();
179.         TpcPrimaryCluster* clus
180.         =new((*fprimArray)[size]) TpcPrimaryCluster(point->GetTime(),
181.                                                     q_cluster,
182.                                                     TVector3(0,0,0),
183.                                                     point->GetTrackID(),
184.                                                     ip,
185.                                                     point->GetSecID());
186.         clus->setIndex(size);
187.
188.         ++ncluster;
189.       }// finish loop for cluster creation
190.
191.       //Step 2: Distribute Clusters along track segment
192.       LinearInterpolPolicy().Interpolate(theLastPoint,point,fprimArray,icluster,ncluster);
193.       icluster+=ncluster;
194.     }//end check for same track
195.     theLastPoint=point;
196.   } // finish loop over GHits
197.
198.   std::cout<<"TpcClusterizer:: "<<fprimArray->GetEntriesFast()<<" clusters created"<<std::endl;
199.
200.   return;
201. }
```

: if current point and last track is in the same track,

: convert unit of energy from GeV to eV

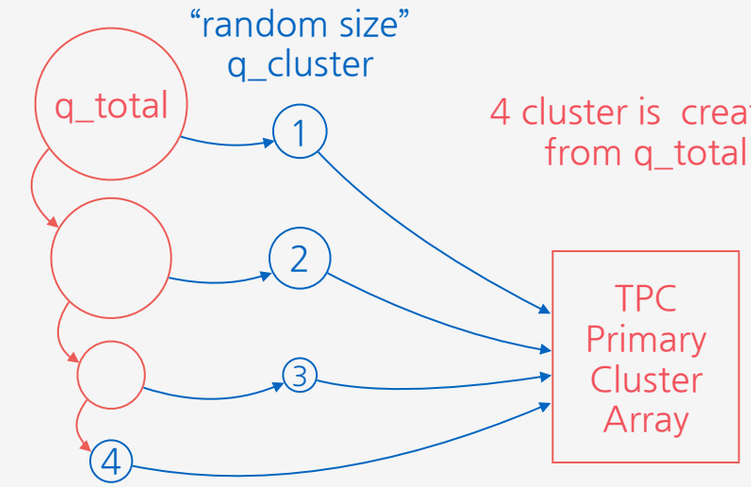
: if energy loss is negative, send error and continue 'for' loop

*floor(x) : round down x to 1st digit.
*fabs(x) : compute absolute value x.

g_total is round down value of absolute value of [energy loss] / [effective ionization energy(in eV) 'W'] which is same as **number of electrons produced** in this point

: get random cluster size 'q_cluster'(**study is need on how random cluster size is calculated**). make cluster and save the cluster into data set. subtract q_cluster from q_total until q_total is 0.

ex)



4 cluster is created from q_total

TpcClusterizerTask::ChargeConversion

```
202. void TpcClusterizerTask::ChargeConversion()
203. {
204.   //hardcoded value from ALICE paper:
205.   //const Float_t w_ion = 35.97e-9; //mean energy for pair creation
206.
207.   Float_t w_ion = fgas->W()*1.e-9;
208.
209.   Int_t np=fpointArray->GetEntriesFast();
210.   for(int ip=1;ip<np;++ip)
211.   {
212.     TpcPoint* point=(TpcPoint*) fpointArray->At(ip);
213.     //Do no clustering just convert energy deposition to ionisation
214.
215.     //is this assuming some actual, valid process done by GEANT
216.     //(e.g. vibration mode, ...) or do we lose something here (F.B.)??
217.     if(point->GetEnergyLoss() < fPoti)
218.       continue;
219.
220.     int nel = int(floor(((point->GetEnergyLoss())-fPoti)/w_ion)) + 1;
221.
222.     //nel=TMath::Min(nel,300); // 300 electrons corresponds to 10 keV
223.
224.     Int_t size = fprimArray->GetEntriesFast();
225.     TpcPrimaryCluster* clus=new((*fprimArray)[size])TpcPrimaryCluster(point->GetTime(),
226.                                                 nel,
227.                                                 TVector3(point->GetX(),
228.                                                       point->GetY(),
229.                                                       point->GetZ()),
230.                                                 point->GetTrackID(),
231.                                                 ip,
232.                                                 point->GetSecID());
233.         clus->setIndex(size);
234.   }
235. }
```

: get W (effective ionization energy). convert unit from eV to GeV.

: get W (effective ionization energy). convert unit from eV to GeV.

: fPoti is first ionization potential. Can be set with SetFirstPoti() and by default, it has value of 20.77e-9 GeV. So if energy loss is smaller than fPoti, no ionization occur.

: number of electron produced is calculated by ([energy loss] - [first ionization potential]) / [effective ionization energy]

Cluster is made only inside the Tpc point which is different from method in Exec().

In header file, it says about 'MereChargeConversion' :
"This has to be set if the ALICE Monte Carlo is activated in TpcDetector".
And about 'fPoti' :
"first ionization potential, used in ALICE charge conversion".
So It has something to do with using ALICE MC.

FOPI digi macro is using this method

TpcClusterizerTask for SPiRIT

- Nothing to change(?)

TpcDriftTask

- **QAPlotCollection* fqa**
  - **In digi macro >>>**

```
TpcDriftTask* tpcDrifter = new TpcDriftTask();

QAPlotCollection* QAPlotCol = new QAPlotCollection();
tpcDrifter -> SetQAPlotcol(QAPlotCol);

// after fRun -> Run()
tpcDrifter -> WriteHistograms();
```

  - This will save the histogram of x,y-shifts of electron in data file.

- **TpcGas* fgas** = fpar -> getGas()
  - drift velocity : fgas -> VDrift()
  - attachment coefficient(?) : fgas -> k()
  - longitudinal diffusion coefficient : fgas -> Dl()
  - transversal diffusion coefficient : fgas -> Dt()

TpcDriftTask

- **TpcDigiPar* fPar** = (TpcDigiPar*) db -> getContainer("TpcDigiPar")
  - Gas : fpar -> getGas()
  - Diffusion along Longitudinal : fpar -> getDiffuseL()
  - Diffusion along Transversal : fpar -> getDiffuseT()
  - Attachment(?) : fpar -> getAttach()
  - GemPosition is contained in parameter file but no function was made to call. We can always make one.

TpcDriftTask::Exec()

```
156. void
157. TpcDriftTask::Exec(Option_t* opt)
158. {
159.   // Reset output Array
160.   if(fdriftedArray==0) Fatal("TpcPrimCluster::Exec)","No DriftedElectronArray");
161.   fdriftedArray->Delete();
162.
163.   //loop over incoming electrons
164.   Int_t nc=fprimArray->GetEntriesFast();
165.
166.   for(int ic=0;ic<nc;++ic){
167.     TpcPrimaryCluster* pcl=(TpcPrimaryCluster*)fprimArray->At(ic);
168.
169.     if(fphicut){
170.       double phi=pcl->pos().Phi();
171.       if(phi<fphimin || phi>fphimax)continue;
172.     }
173.     //create single electrons
174.     Int_t q=pcl->q();
175.     for(Int_t ie=0;ie<q;++ie){
176.
177.       //calculate drift time
178.       double driftl=pcl->z();
179.       if(driftl<0)continue;
180.       //attachment
181.       if(fattach){
182.         if ( exp( -driftl * fgas->k() ) < gRandom->Uniform())
183.           continue;
184.       }
185.       //diffusion
186.       double dx=0;double dy=0; double dt=0;
187.       dt=driftl/fDriftVel;
188.       if(fdiffuseL){
189.         double sigmal = fgas->Dl() * sqrt(driftl);
190.         dt+=gRandom->Gaus(0,sigmal)/fDriftVel;
191.       }
```

: get number of entires from primary cluster array

: loop over all primary cluster

: get primary cluster

: if there is phi cut,
1) get angle phi from primary cluster
2) 'continue' loop if angle phi of cluster is out of phi cut range.
phi cut can be set with SetPhiCut(phimin, phimax)

: get charge 'q' from the primary cluster. Charge represents number of electron produced, so loop over number of electron 'q'.

1) calculate drift length
2) if drift length is smaller then 0, continue
3) attachment ???

drift time 'dt' is defined as drift length divided by velocity.

: **fdiffuseL' is by default true**, and I have found no where to change it out side the code. Sigma of longitudinal direction 'sigmal' is defined by = Dl × √(drift length along longitudinal direction) where Dl is longitudinal diffusion coefficient. Then drift time is summed with diffused length(by random)

TpcDriftTask::Exec()

```
192.        if(fdiffuseT){
193.          double sigmat = fgas->Dt() * sqrt(driftl);
194.          dx+=gRandom->Gaus(0,sigmat);
195.          dy+=gRandom->Gaus(0,sigmat);
196.        }
197.        //drift distortions
198.        if(fdistort){
199.          double posX = pcl->x();
200.          double posY = pcl->y();
201.          double posZ = pcl->z();
202.          TVector3 value = fdevmap->value(TVector3(posX, posY, posZ));
203.          dx+=value.X();
204.          dy+=value.Y();
205.          dt+=value.Z() / fDriftVel;
206.        }
207.        Int_t size = fdriftedArray->GetEntriesFast();
208.        TpcDriftedElectron* myElectron = new((*fdriftedArray)[size]) TpcDriftedElectron(pcl->x()+dx,
209.                                                                                        pcl->y()+dy,
210.                                                                                        pcl->t()+dt,
211.                                                                                        pcl);
212.        myElectron->setIndex(size);
213.        myElectron->setDist(dx,dy,dt);
214.
215.        //feeding the tracking Histograms with this electrons' data
216.        FillHistograms(dx, dy, driftl);
217.        //std::cout<<"x="<<pcl->x()<<"  y="<< pcl->y()<<std::endl;
218.      } // end loop over electrons
219.
220.    } // end loop over clusters
221.    if (fVerbose) std::cout<<fdriftedArray->GetEntriesFast()<<" electrons arriving at readout" <<std::endl;
222.    return;
223. }
```

: **'fdiffuseT' is by default true**, but initialize again in Init() from fpar.
Sigma of transversal direction 'sigmat' is defined by = Dt × √(drift length along transversal direction) where Dt is transversal diffusion coefficient.
dx and dy is summed with diffused length each.

: **'fdistort' is by default false**. Can be set with SetDistort()
!!! what is fdevmap(TpcDevmapCyl)?

All informations are saved in data set 'TpcDriftedElectron' (TClonesArray)

TpcDriftTask for SPiRIT

- Calculation of Drift length should be changed.
- y is used for drift length.
- Position x, y >> Position x, z.
- If gas is changed, parameter sets should be changed such as diffusion coefficients.