TPC Flash ADC Simulation/Reconstruction In Marlin and LCIO

Jim Hunt
and Dan Peterson

Cornell University

LCWS: May 2007

CORNELL UNIVERSITY LEPP
LABORATORY FOR ELEMENTARY-PARTICLE PHYSICS

What Are We Trying To Do?

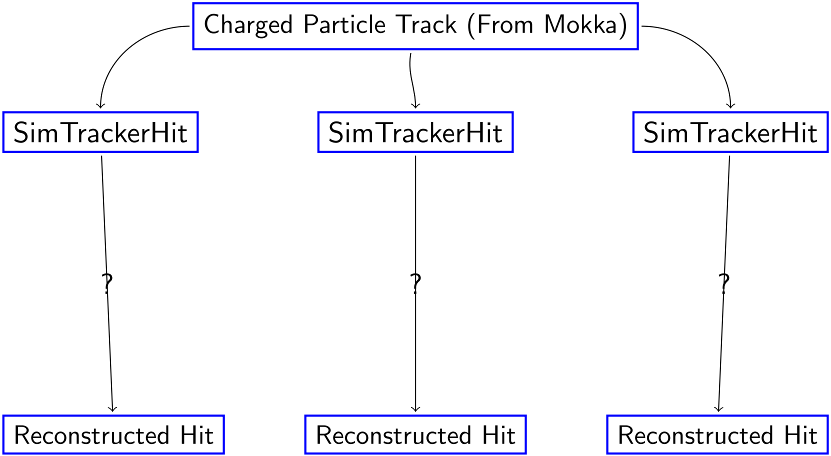
Charged Particle Track (From Mokka)

SimTrackerHit

SimTrackerHit

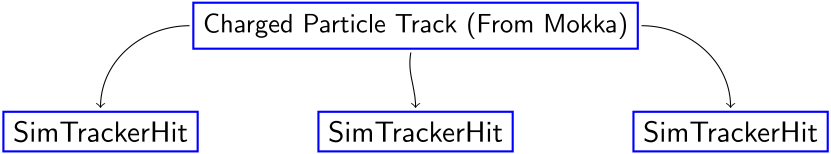
SimTrackerHit

What Are We Trying To Do?



Question: Can We Directly Make Reconstructed Hits?

What Are We Trying To Do?



No! TPCs Have Pads!

TPC Hits are not one-to-one with Reconstructed Hits.

What Are We Trying To Do?

Charged Particle Track (From Mokka)

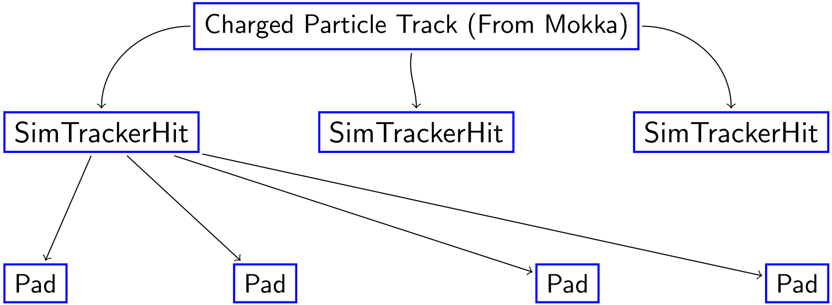
SimTrackerHit SimTrackerHit SimTrackerHit

No! TPCs Have Pads!

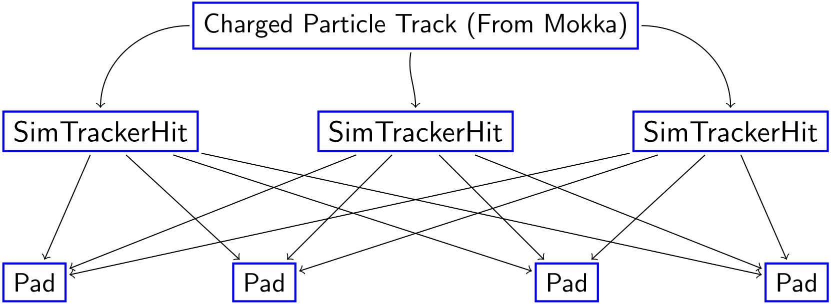
TPC Hits are not one-to-one with Reconstructed Hits.

(Silicon Hits are closer to one-to-one with Reconstructed Hits)

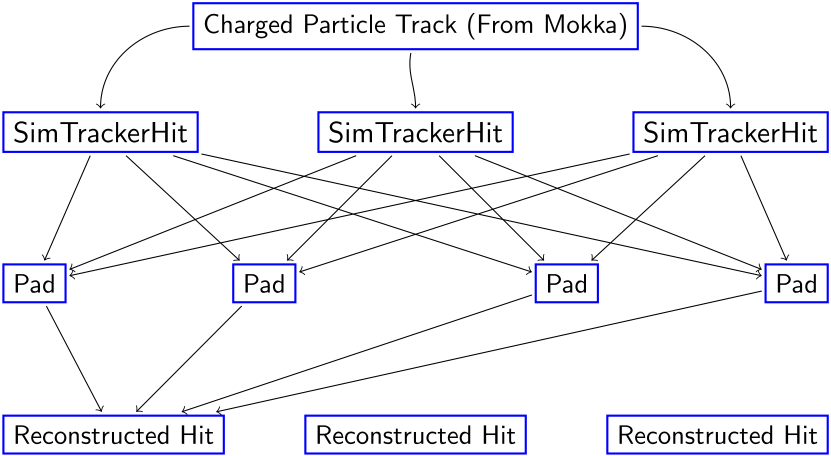
What Are We Trying To Do?

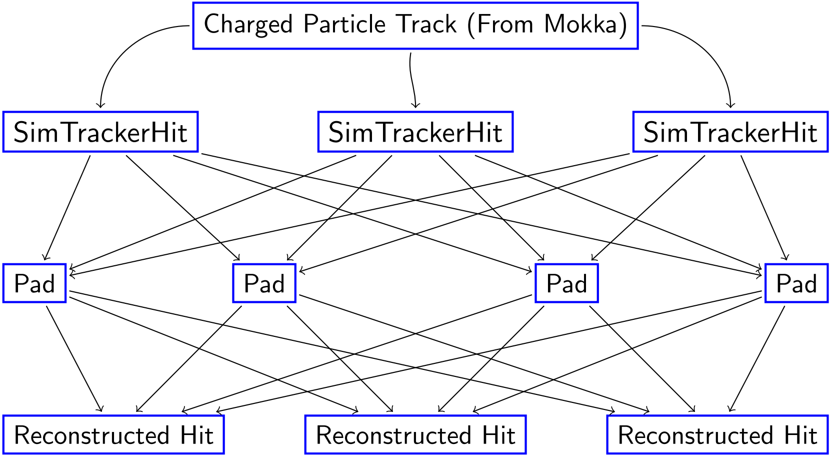


What Are We Trying To Do?



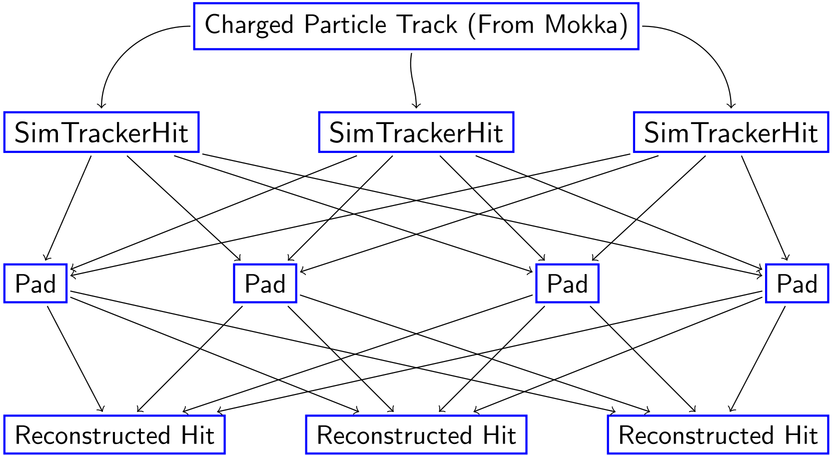
What Are We Trying To Do?





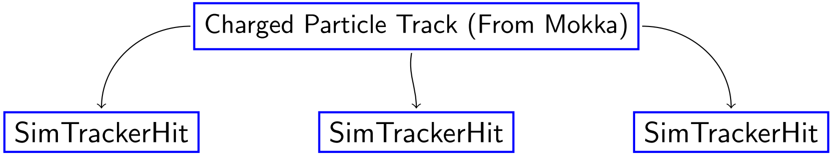
Charged Particle Track (From Mokka)
SimTrackerHit
SimTrackerHit
SimTrackerHit
Pad
Pad
Pad
Pad
Reconstructed Hit
Reconstructed Hit
Reconstructed Hit

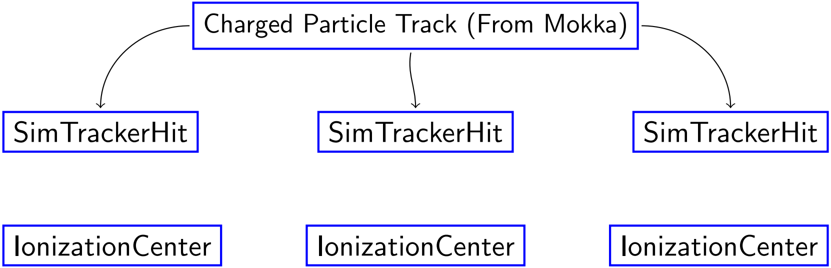
What Are We Trying To Do?



Goal: Simulate this structure in Marlin.

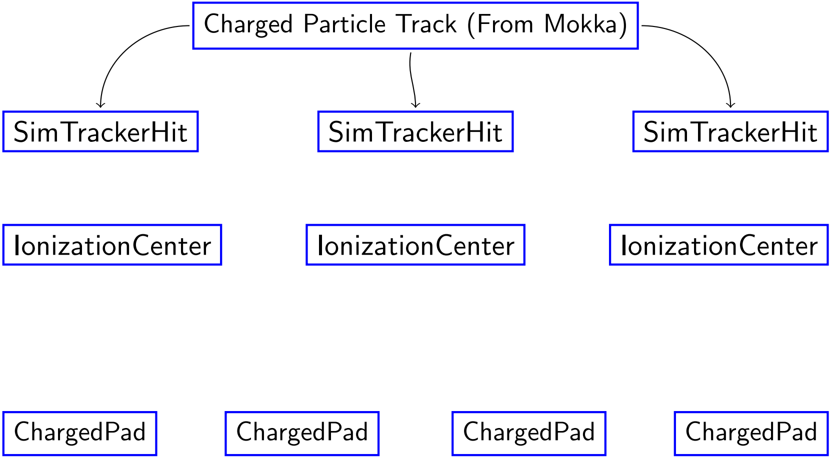
Custom LCIO Objects: IonizationCenters and ChargedPads



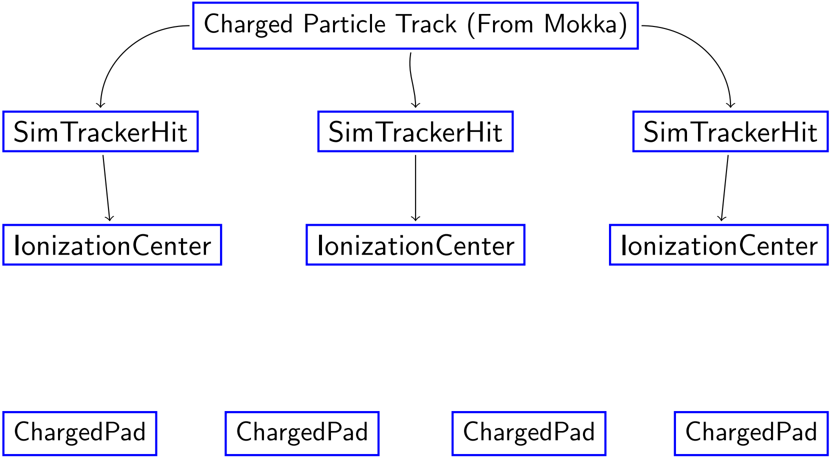


New Custom LCIO Object: IonizationCenter

Custom LCIO Objects: IonizationCenters and ChargedPads

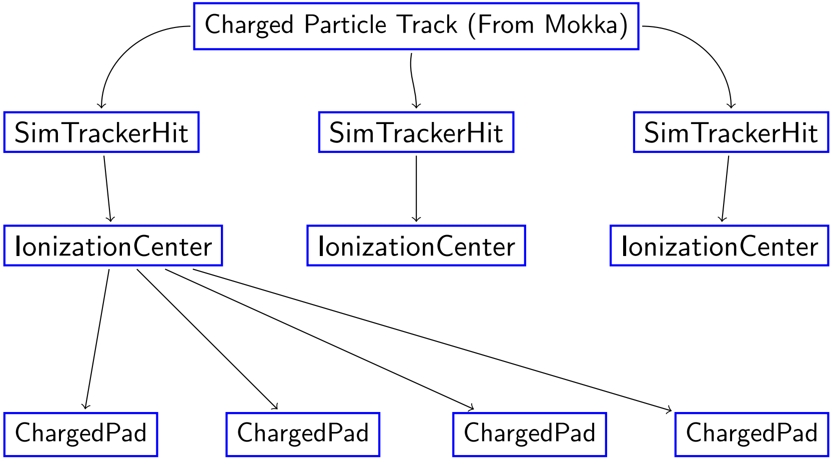


New Custom LCIO Object: ChargedPad



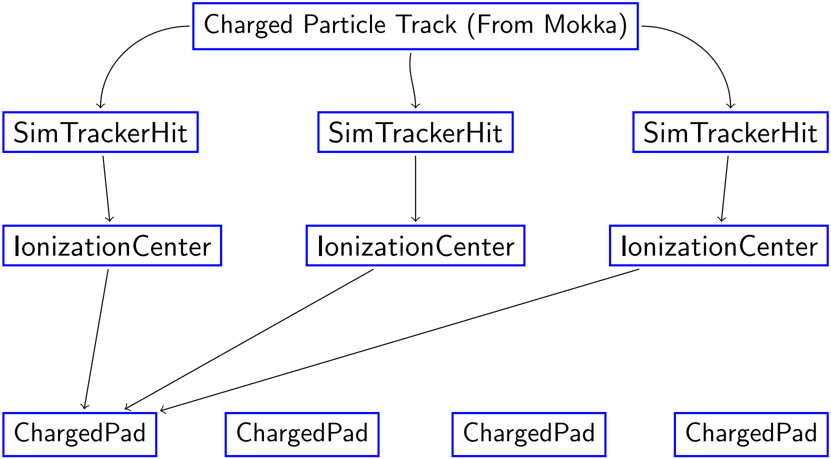
IonizationCenters are One-To-One with SimTrackerHits

Custom LCIO Objects: IonizationCenters and ChargedPads



IonizationCenters hold the One-To-Many Relationship.

Custom LCIO Objects: IonizationCenters and ChargedPads



ChargedPads hold the Many-To-One Relationship.

What's in an IonizationCenter?

An IonizationCenter points back to the lcio::SimTrackerHit and holds an array of the ordered pairs (pad_number, charge).

```
class IonizationCenter : public CustomLCIOObject {
  ...
  class Charge {
  ...
    PadNumber pad_number;
    double charge;
  };
  fast_data_structures::array<Charge> m_charges;
  SimTrackerHit m_sim_tracker_hit;
}
```

What's in a ChargedPad?

A ChargedPad points to a PadGeometry::Pad (see later slides), holds Flash ADC Values and holds a list of IonizationCenters.

```
class ChargedPad : public CustomLCIOObject{
   ...
   fast_data_structures::sllist<IonizationCenter*>
                                          m_ionization_centers;
   fast_data_structures::array<unsigned int> m_adc_values;
   PadGeometry::Pad* m_pad;
}
```

The Algorithm

The Algorithm for creating IonizationCenters and ChargedPads in pseudo-code is:

```
for sim_hit in SimTrackerHits
    ion = new IonizationCenter
    ion.set_sim_tracker_hit(sim_hit)
    ion.generate the (pad_number,charge) ordered pairs

for pad in Pads
    charged_pad = new ChargedPad

for ion in IonizationCenters
    for pairs in ion
        charged_pad = get_charged_pad(pair.pad_number)
        charged_pad.add_ion(ion)
```

The user controls how charge is placed on pads with an

```
class IonizationCenterChargeDistribution {
      virtual double get_max_xy_radius(Vector &ionization_center)=0;
      virtual double get_relative_charge(Vector &ionization_center,
                                         PadGeometry::Pad* pad )=0;
      virtual double get_total_charge()=0;
};
```

The user controls how charge is placed on pads with an

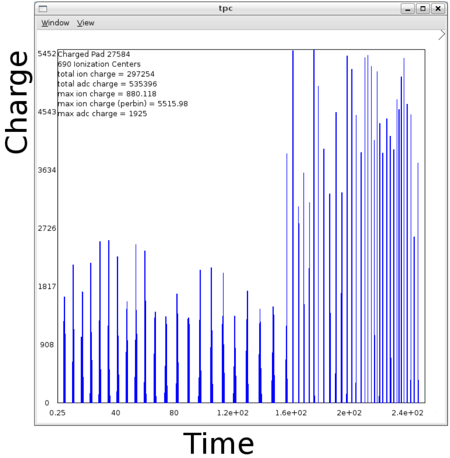
```
class IonizationCenterChargeDistribution {
      virtual double get_max_xy_radius(Vector &ionization_center)=0;
      virtual double get_relative_charge(Vector &ionization_center,
                                         PadGeometry::Pad* pad )=0;
      virtual double get_total_charge()=0;
};
```

The Algorithm in pseudo-code for each ion:

```
dist = get_user_defined_charged_distribution()
xy_radius = dist.get_max_xy_radius(ion.get_xy())
pads = get_pads_near(ion.get_xy(), xy_radius)
total_charge = 0
for pad in pads
 charge = dist.get_relative_charge(ion.get_xy(),pad)
 total_charge += charge
 pad.charge = charge
scale_correction = dist.get_total_charge()/total_charge
for pad in pads
  pad.charge *= scale_correction
```

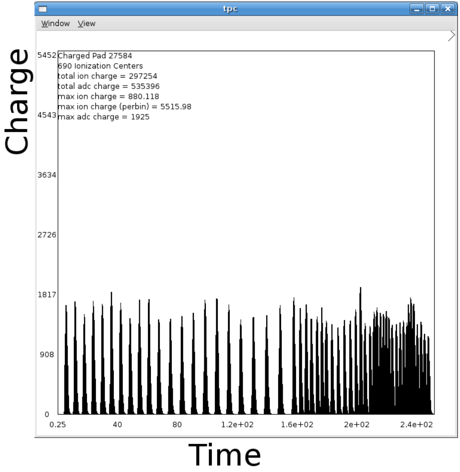
IonizationCenter charge vs time.

Below is an example of a highly populated pad (pad number 27584 from a Kaon event).

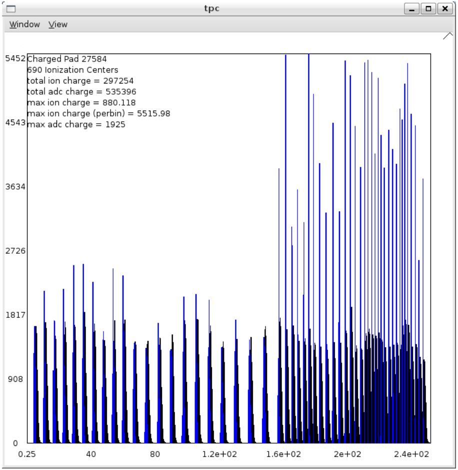


Generated Flash ADC values vs time

(generated dynamically by looping over the ChargedPad's IonizationCenters and putting in an exponential decay)



ChargedPadViewer (charge on one pad)



For the rest of the talk I'm going to show tools that allow the user to inspect events. The point is to figure out what made the time structure for this pad. All plots that follow are from this event.

A New PadGeometry

We could not use

The **GE**ometry **A**pi for **R**econstruction package

for Pad Geometry because it lacked the crucial get_pads_near(xy,xy_radius) method.[1]

[1] the closest match was PadRowLayout2D::getNearestPad

A New PadGeometry

We could not use

The **GE**ometry **A**pi for **R**econstruction package

for Pad Geometry because it lacked the crucial get_pads_near(xy,xy_radius) method.[1]

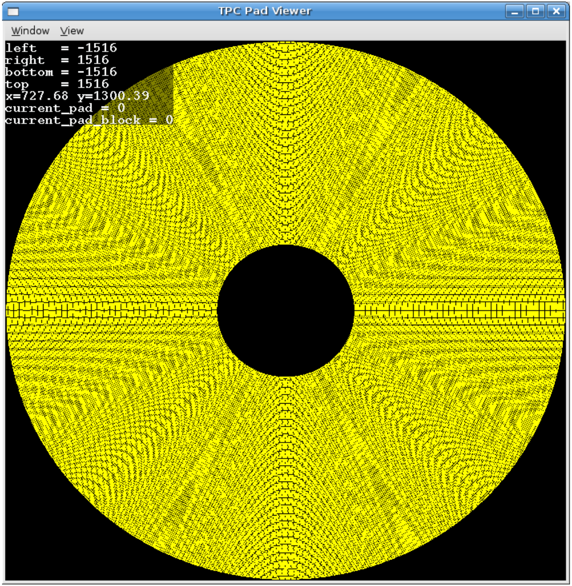
So instead I wrote a new PadGeometry package

- that is based on arbitrarily placed convex polygons,
- and optimized for speed by using aggressively inlined custom data structures.

[1] the closest match was PadRowLayout2D::getNearestPad

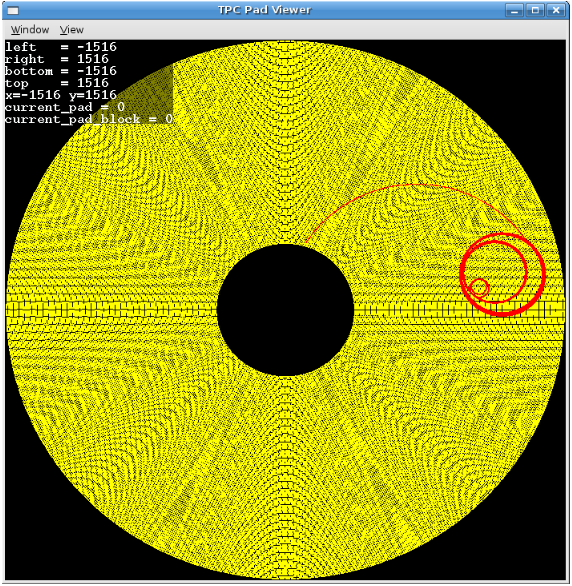
PadGeometry Details

Here's our 2D event display:



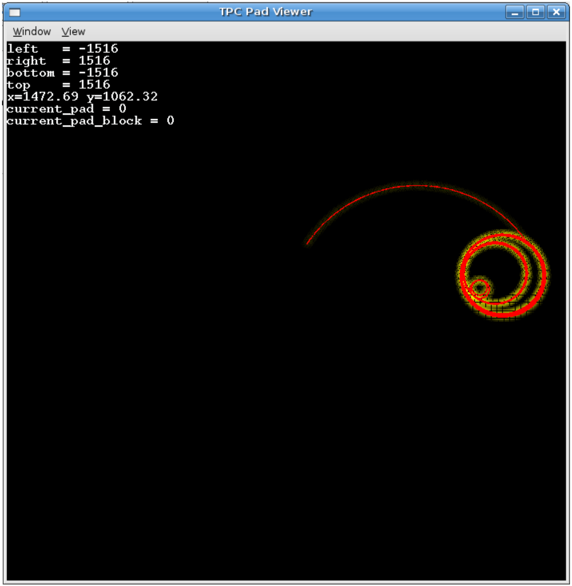
PadGeometry Details

With Tracks From an Event (SimTrackerHits are red):



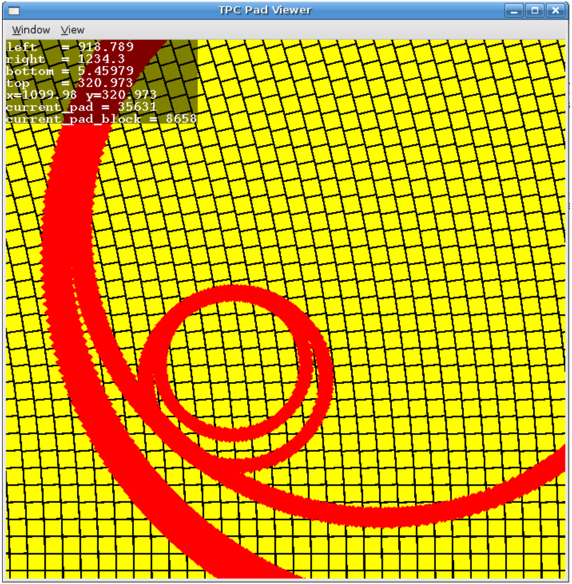
PadGeometry Details

The intensity of the yellow color scales with integrated charge on each pad:

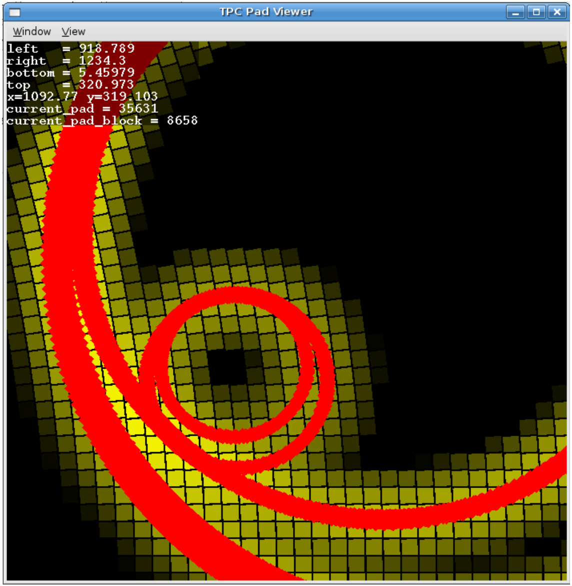


PadGeometry Details

Now zoom in :

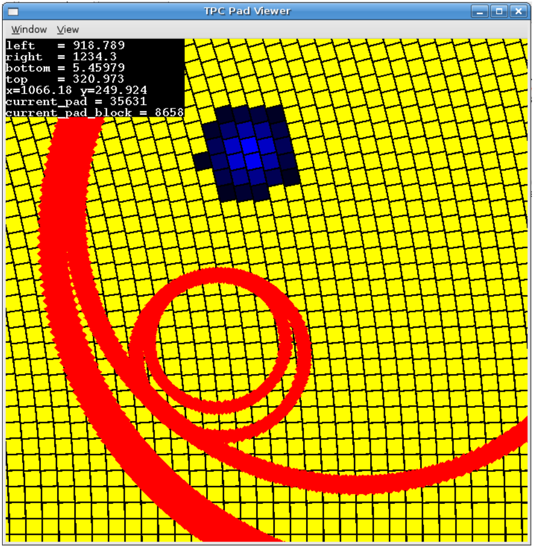


With integrated charged coloring:

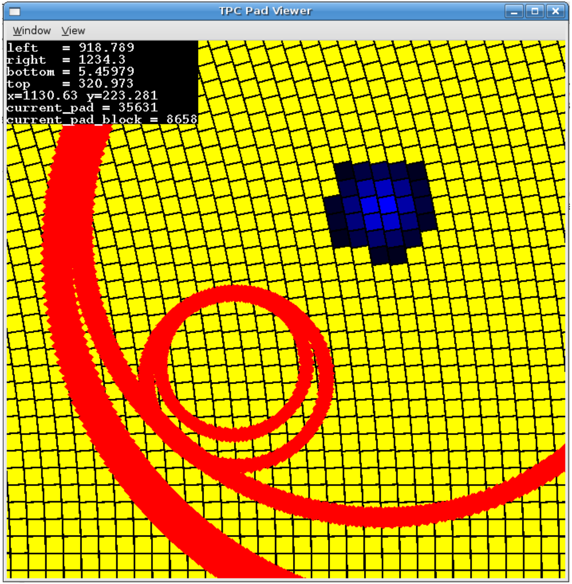


PadGeometry Details

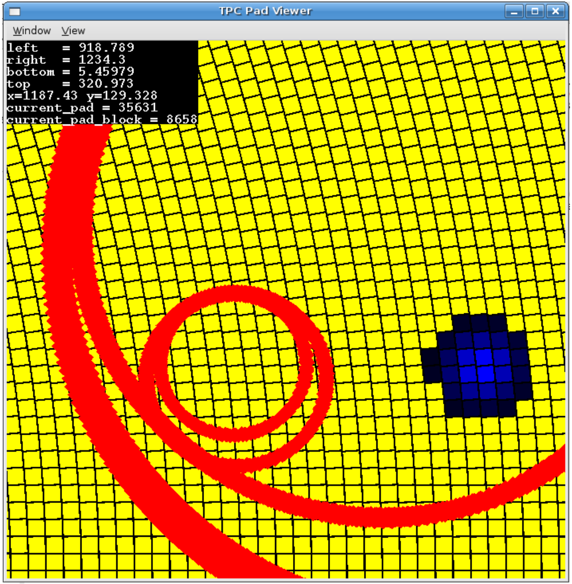
How does this package work? For example, suppose you want to project charge from and IonizationCenter at $z = 0$ to the position of the mouse:



PadGeometry Details

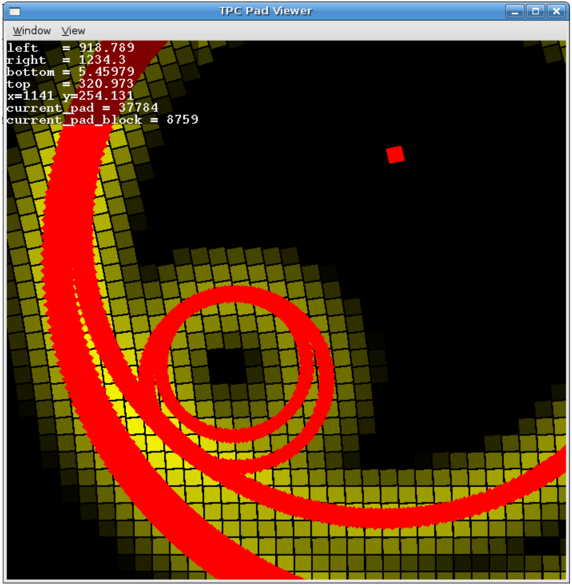


PadGeometry Details



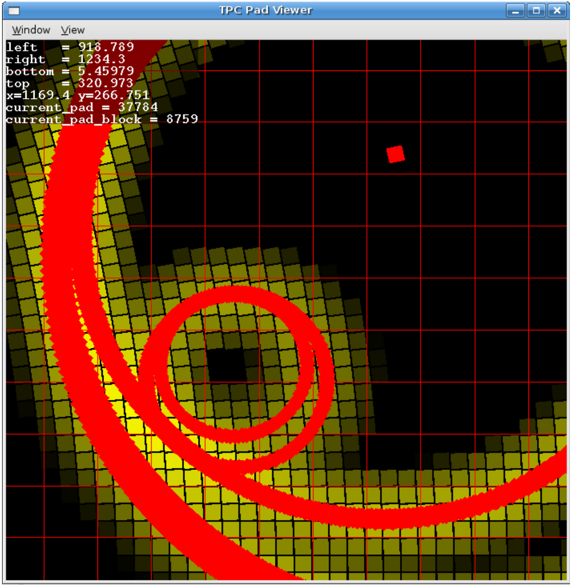
PadGeometry Details

A simpler example is how the package finds the pad at some xy-location:



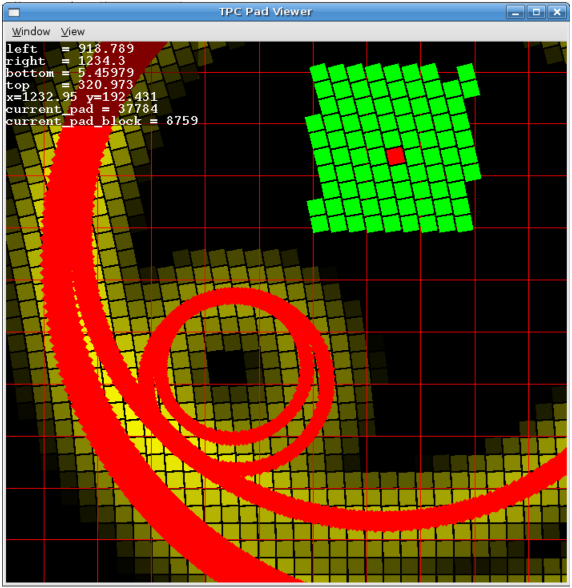
PadGeometry Details

It has an xy-grid index:



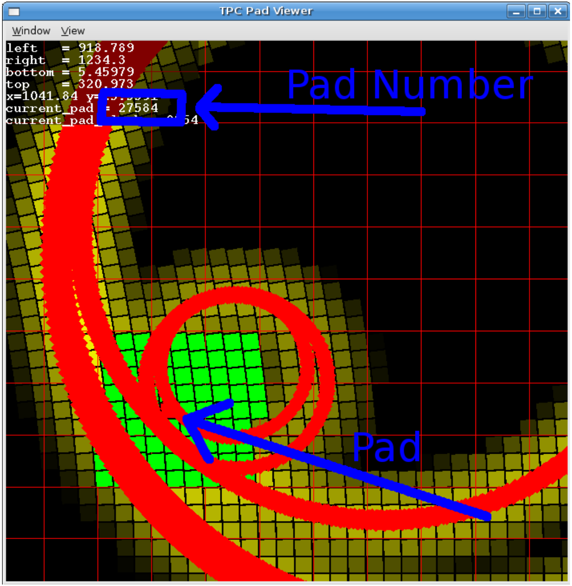
PadGeometry Details

And scans pads within one grid spacing:



PadGeometry Details

Here's the pad whose flash adc was displayed earlier:



A Transition...

That was a new 2D EventDisplay.

But for good diagnostics we really need 3D ...

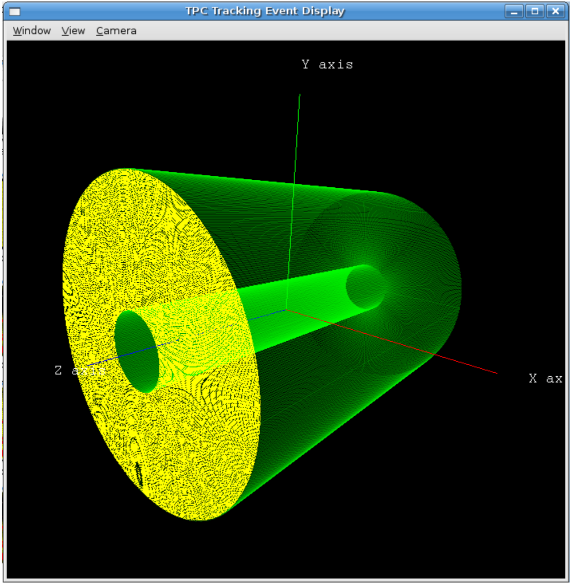
Run the new tpc python program:

```
~/tpc_tracking>  tpc

        A TPC Tracking Environment

http://www.lns.cornell.edu/~jmh263/tpc

tpc version: trunk (svnversion: 195M)


select an lcio file from the list:

 [1] kaon_example.slcio

enter # 1
opening   ~/tpc_tracking/data_files/kaon_example.slcio ...
executing ~/tpc_tracking/data_files/kaon_example.py ...
tpc python>
```

The 3D EventDisplay

```
tpc python> e=tpc.create_event_display()
```

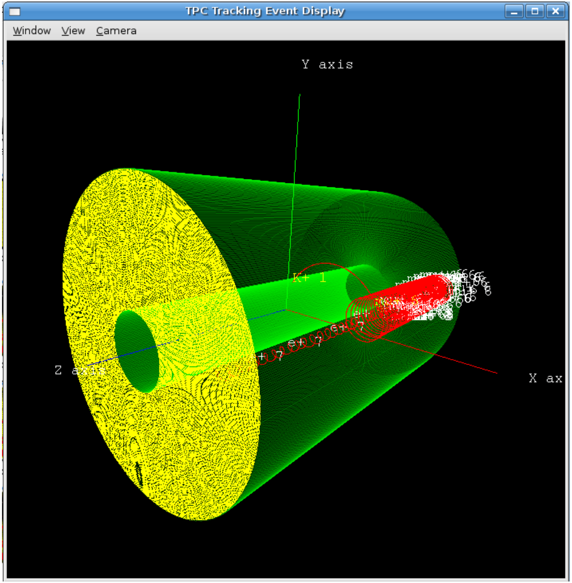
The 3D EventDisplay



The 3D EventDisplay

```
tpc python> tpc.event.read_lcevent_from_file()
```

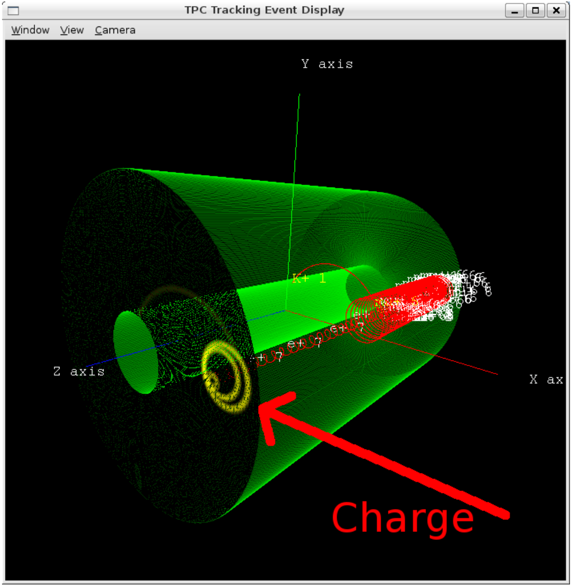
Now tracks with labels are visible:



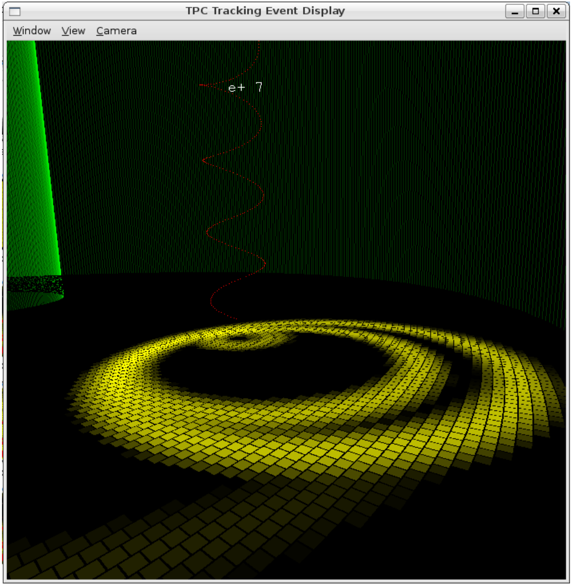
Put charge on the pads:

```
tpc python> tpc.event.ionization_centers.create()
tpc python> tpc.event.ionization_centers.generate_charge_information()
tpc python> tpc.event.charged_pads.create()
tpc python> tpc.event.ionization_centers.put_charge_on_pads()
tpc python> tpc.event.refresh_gui()
```

The 3D EventDisplay



The 3D EventDisplay

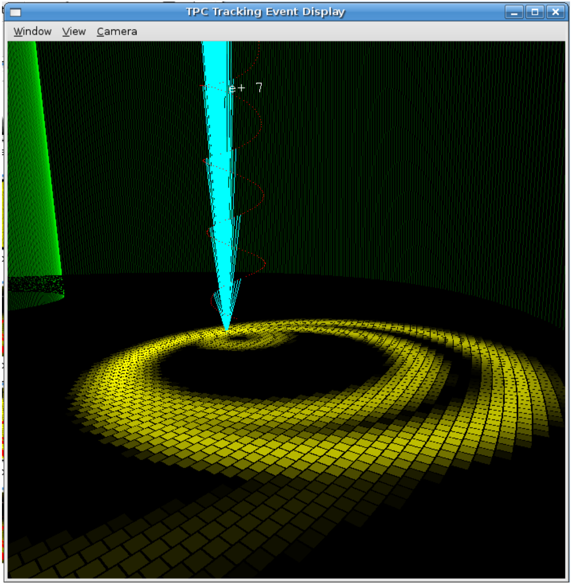


Now highlight the Many-To-One Relationship between IonizationCenters and one Pad.

```
tpc python> e.set_current_charged_pad(
... tpc.event.charged_pads.get_charged_pad(27584))
tpc python> tpc.event.refresh_gui()
```

The 3D EventDisplay

Many-To-One

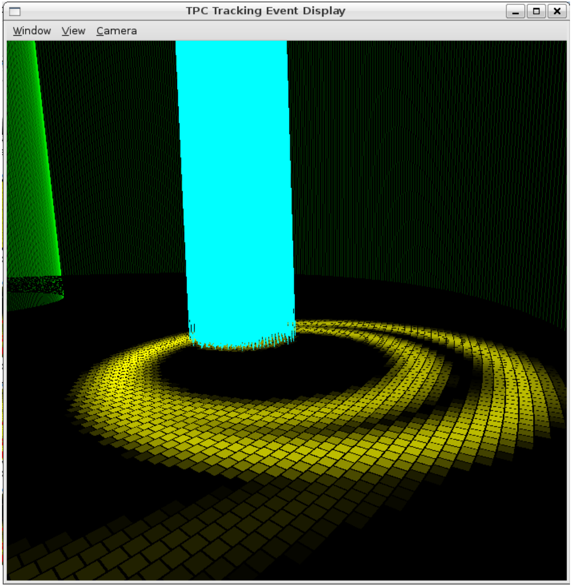


Another example: highlight the IonizationCenters-To-ChargedPads relationships for one Track.

```
tpc python> positron=tpc.event.mc_particles.get_particle(7)
tpc python> track=tpc.event.sim_tracks.get_track(positron)
tpc python> e.set_current_sim_track(track)
tpc python> tpc.event.refresh_gui()
```

The 3D EventDisplay

IonizationCenters-To-ChargedPads (Many-To-Many) for one Track

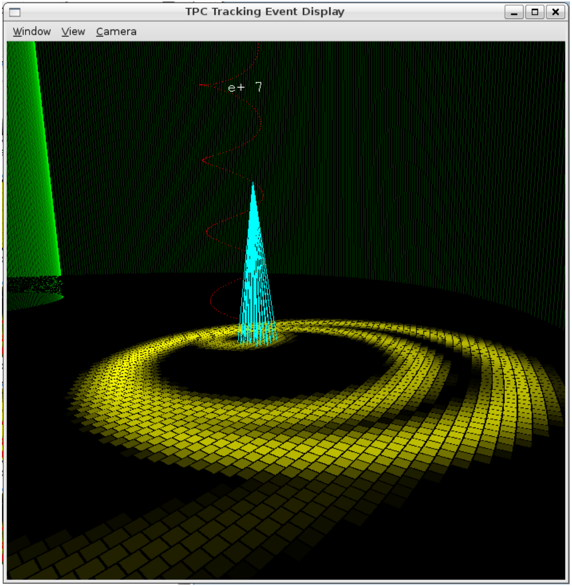


And the One-To-Many relationship between One IonizationCenter and Many ChargedPads:

```
tpc python> tpc.event.ionization_centers.get_nionization_centers()
19104
tpc python> e.set_current_ionization_center(
... tpc.event.ionization_centers.get_ionization_center(19000))
tpc python> tpc.event.refresh_gui()
```

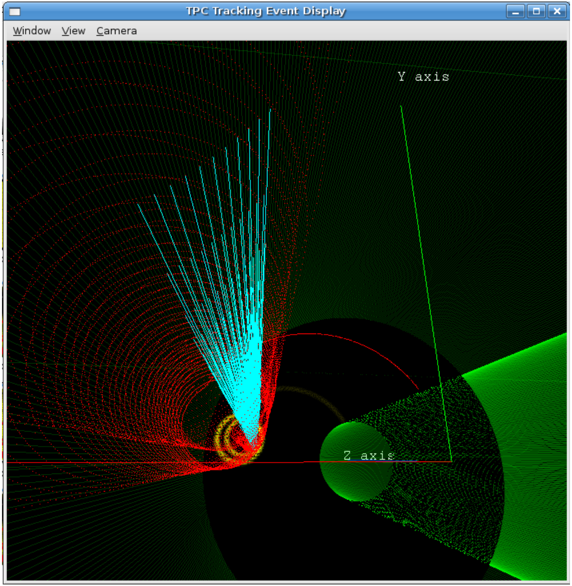
The 3D EventDisplay

One-To-Many.



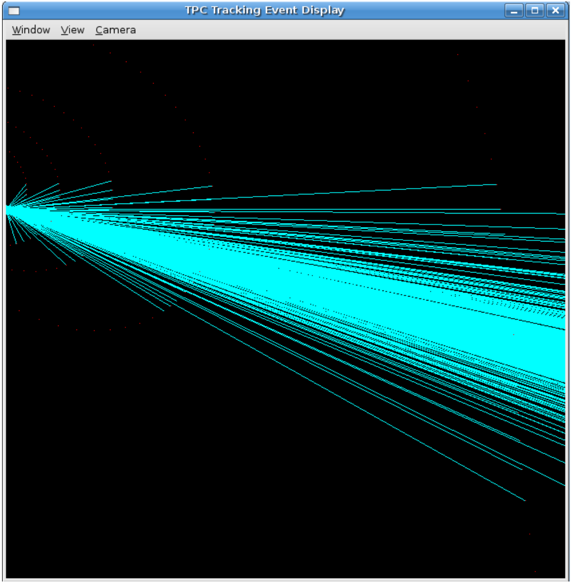
The 3D EventDisplay

other views

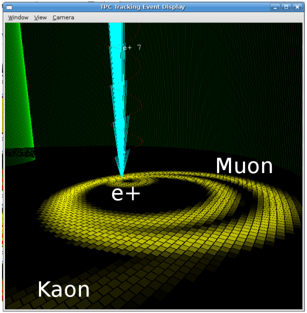


The 3D EventDisplay

other views

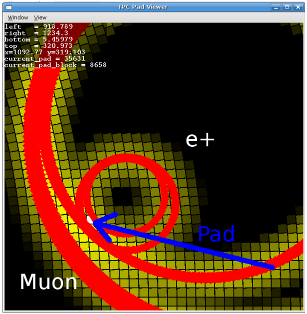


So what's in the event?



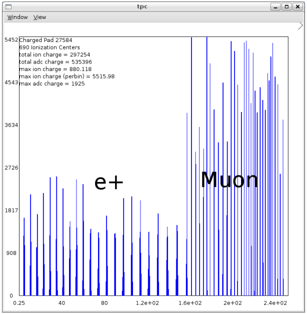

```
tpc python> tpc.event.mc_particles.get_particle(1).print_decays(True)
K+  -> pi0 nu_mu mu+ e- e-
 pi0  -> gamma gamma
 mu+  -> e+ nu_e nu_mu~
  e+  -> gamma
```

So what's in the event?




```
tpc python> tpc.event.mc_particles.get_particle(1).print_decays(True)
K+  -> pi0 nu_mu mu+ e- e-
 pi0  -> gamma gamma
 mu+  -> e+ nu_e nu_mu~
  e+  -> gamma
```

So what's in the event?




```
tpc python> tpc.event.mc_particles.get_particle(1).print_decays(True)
K+  -> pi0 nu_mu mu+ e- e-
 pi0  -> gamma gamma
 mu+  -> e+ nu_e nu_mu~
  e+  -> gamma
```

Here is our working example of a Marlin Processor.

CustomMarlinProcessor is a thin wrapper around marlin::Processor.

```
class PutChargeOnPadsProcessor :
        public tpc_tracking::CustomMarlinProcessor {

    CUSTOM_MARLIN_PROCESSOR_IMPL(PutChargeOnPadsProcessor)

public:

    void processEvent(lcio::LCEvent *lc_event);
    // etc...
};
```

The Marlin Interface: PutChargeOnPadsProcessor

- The user no longer has to supply a newProcessor() call.
- Event::Sync synchronizes lcio::LCEvent with tpc_tracking::Event.

```
void PutChargeOnPadsProcessor::processEvent(lcio::LCEvent *lc_event) {
    using tpc_tracking::Event;
    using tpc_tracking::globals::event;

    Event::Sync sync(lc_event);

    event->ionization_centers.create();
    event->charged_pads.create();
    event->ionization_centers.generate_charge_information();
    event->ionization_centers.put_charge_on_pads();
}
```

Conclusions

We now have

- a new Event Display (3D and 2D),
- a new PadGeometry package,
- a mechanism to add custom objects to LCIO[2]
- marlin processors that move charge from IonizationCenters to ChargedPads,
- and a python interface that glues everything together.

[2] for details see `http://www.lns.cornell.edu/~jmh263/tpc`

Conclusions

Future plans:

- Allow the user to control the amplifier characteristics (Polymorphism).
- Implement Flash ADC Time Reconstruction and store the result in LCIO (Probably a CustomLCIOObject).
- Plan how to integrate this project into the Marlin/MarlinTPC/LCIO/GEAR code base.

Conclusions

For Details, Tutorials, and More visit

http://www.lns.cornell.edu/~jmh263/tpc

Extra Slides

An Object Hierarchy

tpc.event is a wrapper around lcio::LCEvent.

```
namespace tpc_tracking {
 class Event {

   lcio::LCEvent *m_lc_current_event;
 };
}
```

An Object Hierarchy

- tpc.event.*s objects represent Collections.
- the collections hold objects that may wrap lcio objects.

```
namespace tpc_tracking {
 class MCParticle;         // lcio::MCParticle
 class SimTrackerHit;      // lcio::SimTrackerHit
 class SimTrack;           // [ uses tpc_tracking::SimTrackerHit  ]
 class IonizationCenters; // custom lcio object
 class ChargedPads;        // custom lcio object

 class Event {
   MCParticles mc_particles
   SimTrackerHits sim_tracker_hits
   SimTracks sim_tracks;
   IonizationCenters ionization_centers;
   ChargePads charged_pads;
 };
};
```

An Object Hierarchy

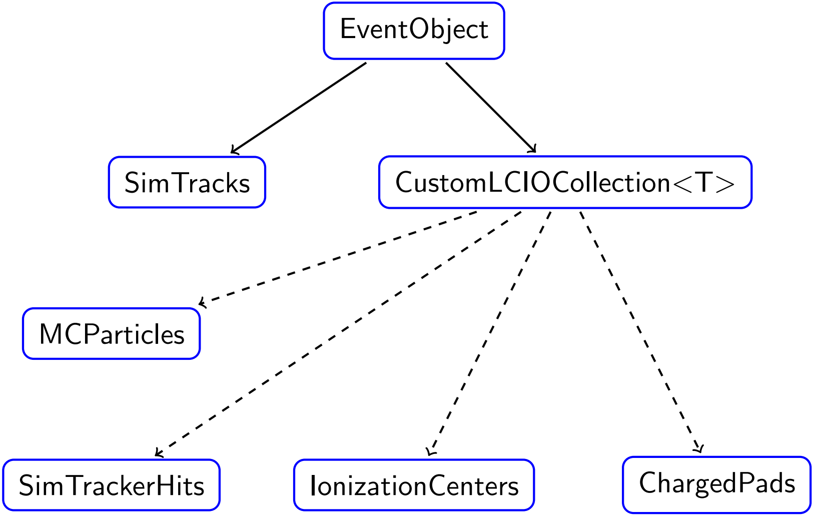
light-weight object:

```
namespace tpc_tracking {
 class MCParticle  {

  lcio::MCParticle* m_pMCParticle;
 };
}
```

heavy-weight object:

```
namespace tpc_tracking {
 class SimTrack {

  std::vector<SimTrackerHitNumber> m_sim_tracker_hits;
 };
}
```

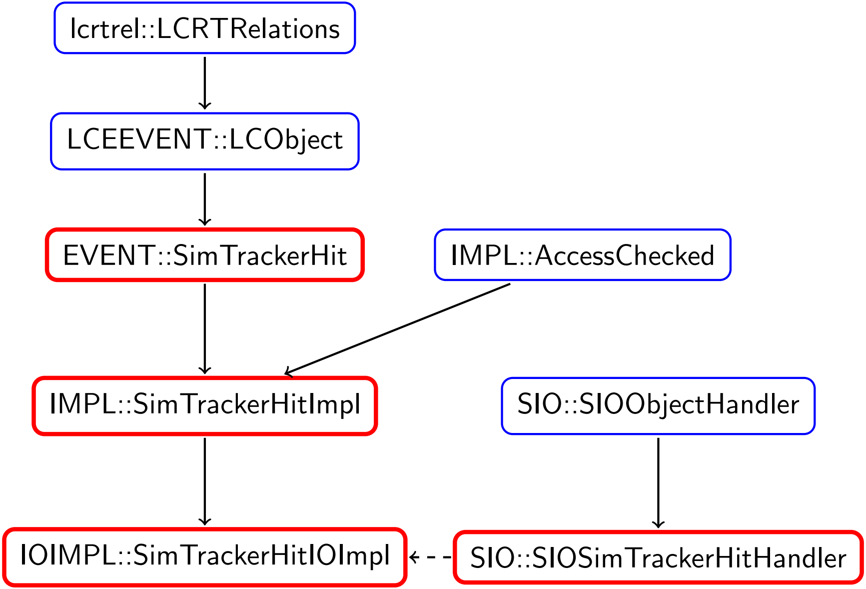
EventObject and CustomLCIOCollection<T>



EventObject and CustomLCIOCollection<T>

```
namespace tpc_tracking {

//  LCIO Collections:

class EventObject;
class SimTracks : public EventObject;

template<typename T>
class CustomLCIOCollection<T> : public EventObject;

class MCParticles : public CustomLCIOCollection<MCParticle>;
class SimTrackerHits : public CustomLCIOCollection<SimTrackerHit>;
class IonizationCenters:public CustomLCIOCollection<IonizationCenter>;
class ChargedPads : public CustomLCIOCollection<ChargedPad>;
};
```

CustomLCIOObject: lcio::SimTrackerHits details



CustomLCIOObject: IonizationCenter.h

- The preprocessor generates an SIOObjectHandler.
- SIO is hidden in CustomLCIOObject::Stream.

```
namespace tpc_tracking {
 class IonizationCenter : public CustomLCIOObject {

    bool read(Stream &stream);
    bool write(Stream &stream);
 };
  DECLARE_CUSTOM_LCIO_OBJECT(IonizationCenter);
};
```

CustomLCIOObject: IonizationCenter.cc

- The preprocessor generates an SIOObjectHandler.
- SIO is hidden in CustomLCIOObject::Stream.

```
namespace tpc_tracking {
    DEFINE_CUSTOM_LCIO_OBJECT(IonizationCenter);
    bool IonizationCenter::read(Stream &stream)
    {
        stream >> m_sim_tracker_hit;
        stream >> m_bGeneratedChargeInformation;
        if (m_bGeneratedChargeInformation) {
            stream >> m_charges;
        }
        return true;
    }
    bool IonizationCenter::write(Stream &stream)
    {
        stream << m_sim_tracker_hit;
        stream << m_bGeneratedChargeInformation;
        if (m_bGeneratedChargeInformation) {
            stream << m_charges;
        }
        return true;
    }
};
```

What happened to SIO_PTR and SIO_PTAG?

```
unsigned int CustomLCIOObject::read(SIO_stream* sio_stream)
{
    Stream stream(sio_stream);
    stream. declare_pointer_target(this) ;
    if (read(stream))
        return SIO_BLOCK_SUCCESS;
    else
        return SIO_BLOCK_NOTFOUND; // see FAQ in SIO manual;
}

unsigned int CustomLCIOObject::write(SIO_stream* sio_stream)
{
    Stream stream(sio_stream);
    stream. declare_pointer_target(this);
    if (write(stream))
        return SIO_BLOCK_SUCCESS;
    else
        return SIO_BLOCK_NOTFOUND; // see FAQ in SIO manual;
}
```

Where is GEAR? What About General Setting?

- As explained earlier, I did not use Gear.
- General Settings are stored in a *Settings* object.
- They are initialized at start-up; remember kaon_example.py:

```
tpc.settings.set_tpc_sim_tracker_hits_collection_name('tpc07_TPC')
tpc.settings.set_tpc_half_length(2037.5);
tpc.settings.set_tpc_min_radius(371);
tpc.settings.set_tpc_max_radius(1516);

tpc.settings.set_time_start(0.25);
tpc.settings.set_time_per_bin(0.25);

tpc.settings.set_nbins(1000);

...

tpc.pad_geometry.create_circular_pattern2(10,10,0.9)

tpc.settings.
   create_diffusion_ionization_center_charge_distribution(10000,20,30)
```