

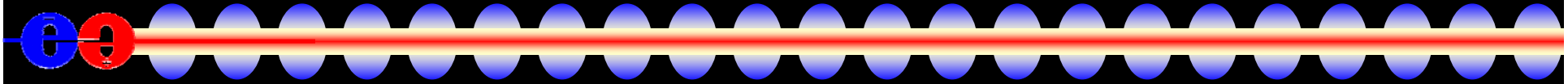
The Vertex

Norman Graf

SLAC

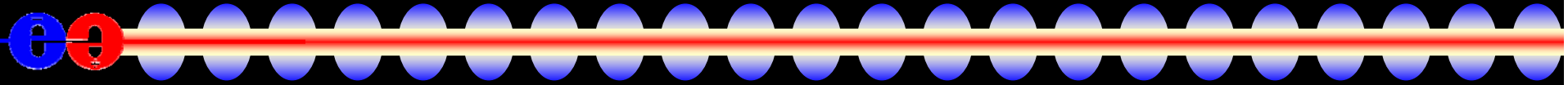
July 19, 2006

What is a Vertex?



- Within the context of HEP event reconstruction, a vertex is loosely understood as the point at which some aggregate of trajectories originates.
- Most naively, this can be treated as the intersection of reconstructed tracks resulting from charged particles.
- Most correctly, the vertex is considered as the point of origin of decay products of a particle and can thus be treated as a particle itself.

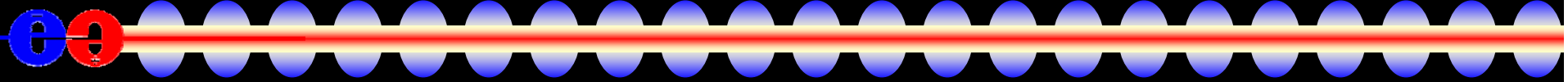
The Vertex as ReconstructedParticle



- The latter is the approach taken by the LC event data model and is why there is (to-date) no explicit Vertex class defined in either the EDM or in LCIO.
- The interface for ReconstructedParticle (RP) is intentionally minimal, and some functionality will need to be added.
 - Question is whether the RP can fully stand in for a Vertex or whether we need a new class in LCIO.

Vertex Proposal

ReconstructedParticle

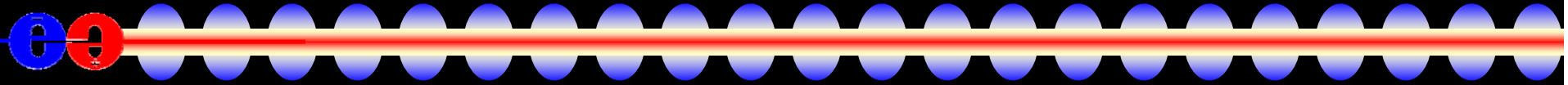


getMomentum()
getMass()
getCharge()
getPosition()
getCovMatrix()
getChi2()
getProbability()
getDistanceToPreviousVertex()
getErrorDistanceToPreviousVertex()
getParameters()
getTracks()
addTrack()
getPreviousVertex()

getMomentum()
getMass()
getCharge()
getReferencePoint()
getCovMatrix()

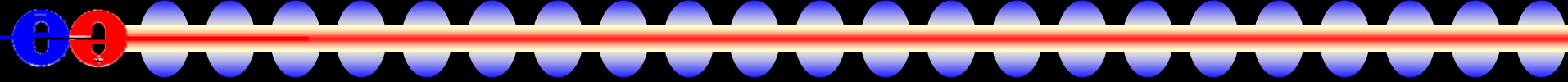
getTracks()

Vertex and RP similarities



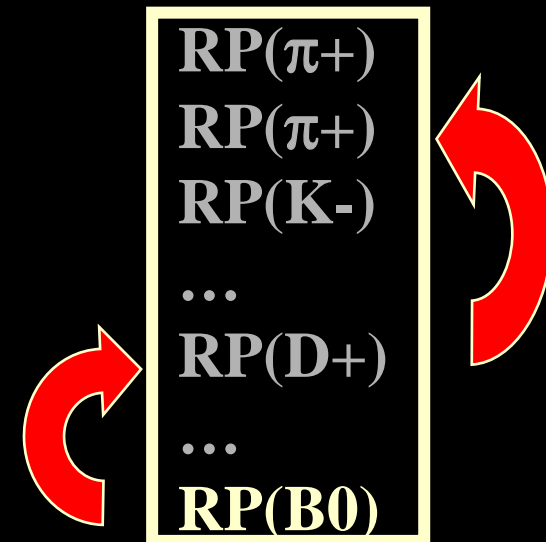
- As can be seen from the previous slide, there is a high degree of overlap between the proposed Vertex class and the existing RP class.
- Can the existing RP class be amended to incorporate the additional information/functionality required of the Vertex class?
- If not, and a new class IS needed, is the proposal complete?

Deficiencies in RP

- 
- Are `getChi2()` and `getProbability()` related?
 - If so, do we need both? Or do we need `getDOF()`?
 - The following all seem very non-OO for `Vertex`, since a `Vertex`, as defined, is composed of `Tracks`, and not `Vertices`. Relationship between *this* and `PreviousVertex` is ill-defined.
 - `getPreviousVertex()`
 - `getDistanceToPreviousVertex()`
 - `getErrorDistanceToPreviousVertex()`
 - Last 2 could be calculated from 2 `Vertex` objects.

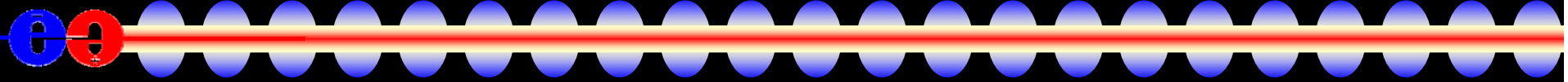
RP deficiencies, continued

- However, this construct (i.e. “pointing” to previous object) makes perfect sense within the context of a hierarchical ReconstructedParticle, since RP’s are themselves composed of RP’s, e.g.



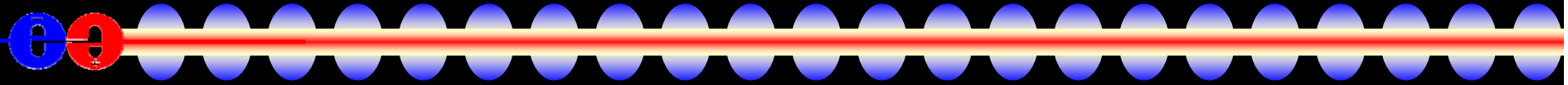
- Not clear what `getParameters()` refers to.

Vertex Deficiencies



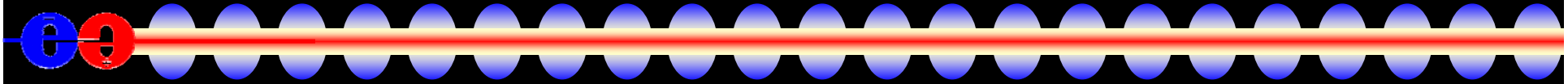
- If we are to have a dedicated Vertex class, then some amount of thought should be devoted to its construction.
- What does a Vertex represent?
- Of what is it composed?
- What functionality does it have?
- How does it interact with other classes?
- How should it be persisted?

What does a Vertex represent?



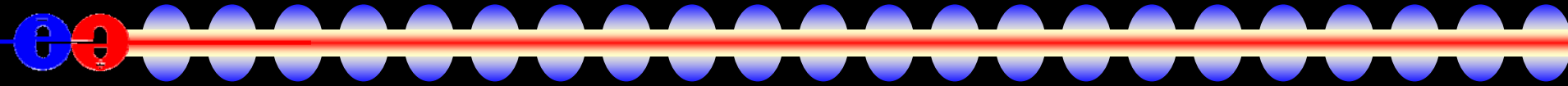
- The proposed Vertex class seems to represent simply the fitted point of intersection of reconstructed tracks representing charged particle trajectories.
- In the current LCIO EDM a Vertex IS a ReconstructedParticle. To-date we have not defined exactly how composite RP's are to be constructed. Clearly a vertex fit would be the most appropriate way to do this.
 - Requires covariance matrix on RP “reference point”.
 - what else?

Of what is a Vertex composed?

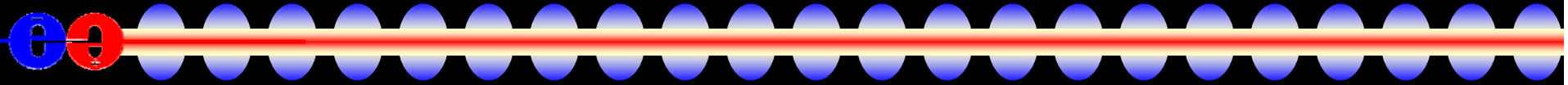


- The proposed Vertex class seems to be composed only of reconstructed tracks representing charged particle trajectories.
- It is clear that this will have to be expanded to include trajectories of neutral particles as well.
 - If not RP, we will need to introduce another class to represent either a charged or neutral trajectory.

What Vertex information is needed?

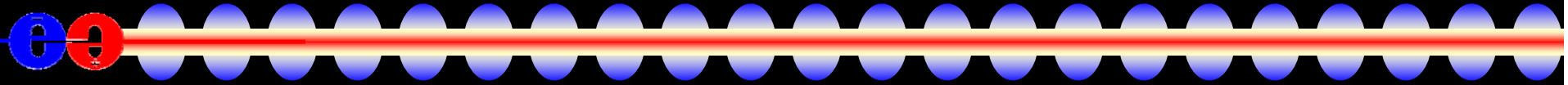
- 
- The vertex position (x) and covariance matrix.
 - The vertex “geometrical momentum” (q) and cov.
 - Covariance terms between x and q .
 - mass? If so, requires mass hypothesis for constituents.
 - OK for RP, but do we now add mass to Trajectory?
 - Constraining constituents to originate from a common vertex improves each trajectory measurement but also introduces covariance terms between all of them
 - do we want to save improved tracks and full cov matrix?
 - Information on constraints applied in fit (see later)

What functionality does it have?



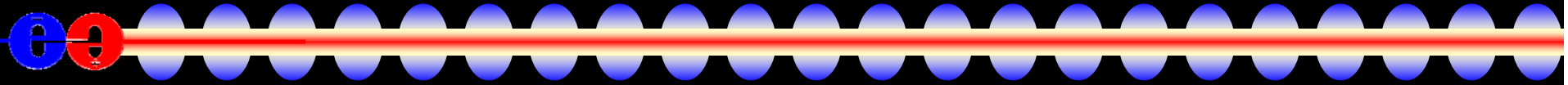
- It is not clear how the proposed Vertex class is to be used. It seems to be seen as simply a mechanism for flavor-tagging jets.
- How would the primary Interaction Point be handled?
- How are conversion pairs, or Vee's handled?
- In the current model, the hierarchy of RP's represents the sequential decay/interaction of previous RP's.

Interaction with other classes.



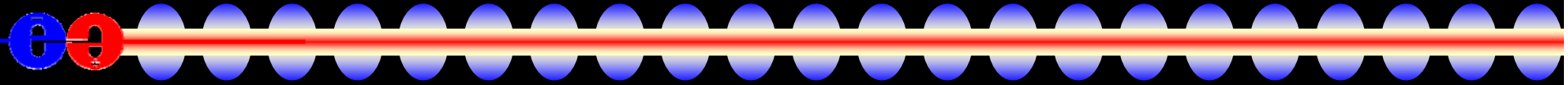
- In order to accommodate both charged and neutral particles, a new class (Trajectory?) would have to be introduced. Vertex would then be composed of these objects.
- Is the Vertex then also a Trajectory?
- Is the Vertex object a constituent of the RP?
- All handled naturally if one does not introduce an explicit new class, but treats a vertex as a particle.

Thoughts on Fitting



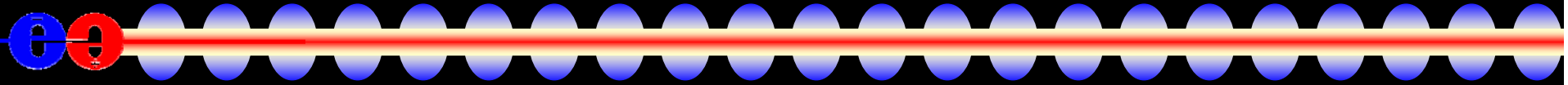
- Clearly, one has to accommodate the possibility of constraints when fitting a common vertex.
 - Constraint to a prior vertex position.
 - Often referred to as a beam-constraint.
 - Constraint to a particle mass
 - Constraint to a particular direction
 - either to the IP, or a previous vertex.
- All affect the fit, reducing the number of degrees of freedom.
- Will need to discuss how to address this issue.

Vertexing in org.lcsim



- The ZvTop Vertex Finding algorithm has been implemented in Java since Snowmass 2001.
 - Updated for org.lcsim by Jan Strube.
- Found vertices good enough for flavor-tagging.
- We are now concentrating on the full vertex fit, including neutrals.
 - will compare “pT-corrected” mass to full fit mass including neutrals in the jet.

Additional Information



- There is an ongoing discussion of this topic on the forum.
- Please see:
 - [Vertex discussion in lcio @ forum.linearcollider.org](http://forum.linearcollider.org)