OO Conversion of the ALPHA BOS Bank Structure

/afs/cern.ch/aleph/project/database/cavanaug/PrototypeB/Applications

Richard Cavanaugh
Universität Heidelberg

December 6, 2000
ALPHA++ Meeting

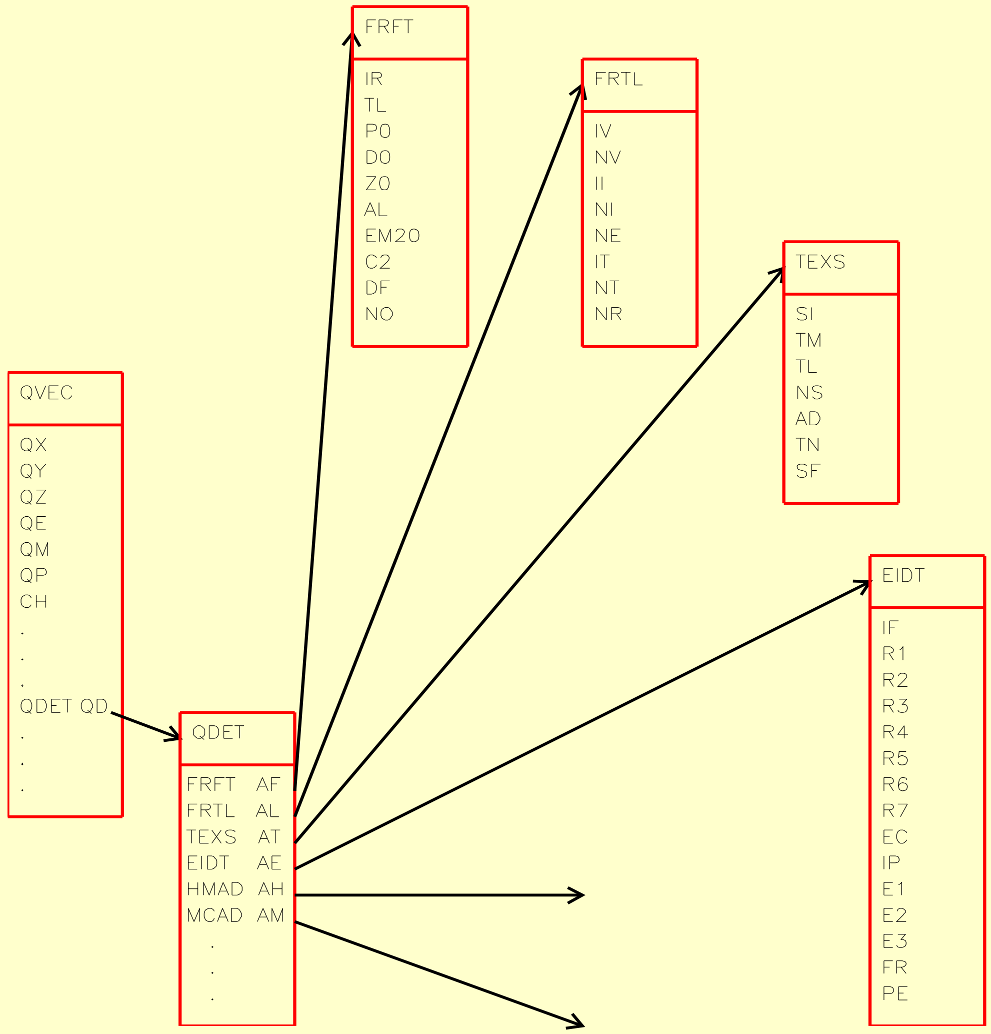
- New Tree-like Data Structure for QVEC
- New Monte Carlo Truth Class
- Conclusion

The ALPHA Data Structure Model

- The QVEC Bank is the fundamental "ALPHA track object"

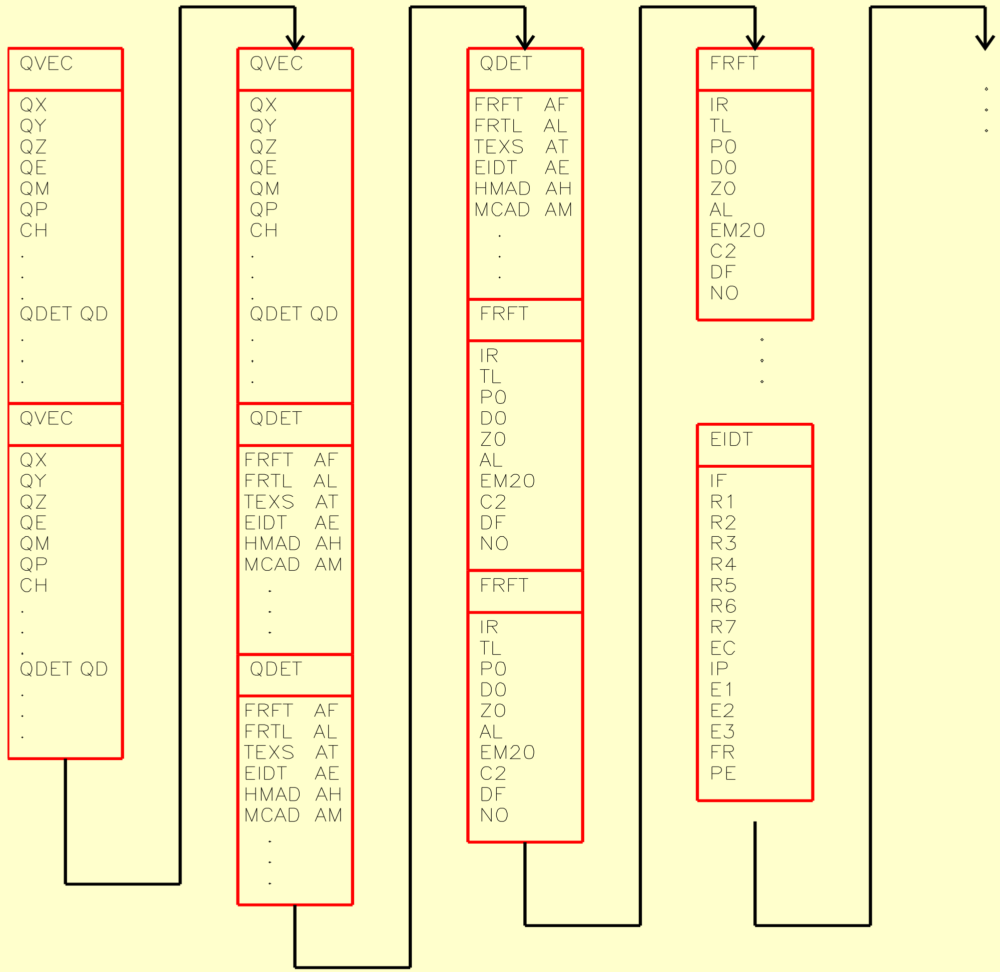
- Bank QVEC has a pointer to the detector information in Bank QDET and corresponds to a particular "track".

- Bank QDET then has pointers to other Banks: FRFT (Tracking), FRTL (Tracking), TEXS (dE/dx), etc.



BOS Implementation of the Model

- The BOS implementation of the model is essentially a linear "look-up" table.



Richard Cavanaugh
Universität Heidelberg

The ALPHA Data Structure Model

- **Conclusion:** FORTRAN ALPHA already has an OO like design, but a non-OO like implementation

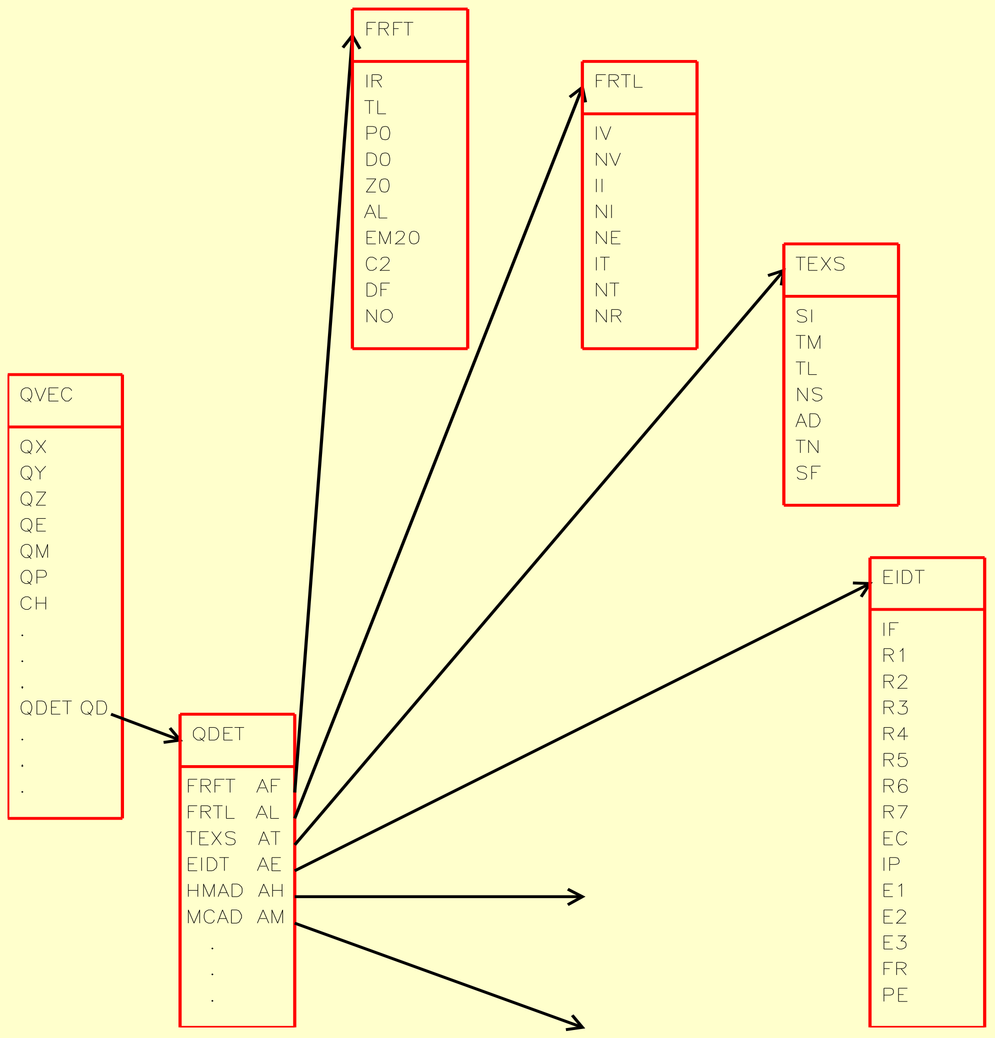
- **Idea:** Directly convert the ALPHA BOS Bank structure into a C++ tree-like data structure

- **Advantages:**
  - Minimises FORTRAN: Cut FORTRAN out at the level of copying the FORTRAN BOS Banks to the C++ Object Tree
  - Natural: All data used by ALPHA is available to ALPHA++ in a transparent way.

- **Example:** Implementation of Monte Carlo Truth information becomes almost trivial.

C++ Data Structure Model and Implementation

- QVEC is the fundamental *class*
- QVEC has a pointer to the detector information *class*: QDET
- The *class* QDET then has pointers to *classes*: FRFT (Tracking), FRTL (Tracking), TEXS (dE/dx), etc.
- AlphaBanks fills everything into an STL vector of QVEC objects



Richard Cavanaugh
Universität Heidelberg

Accessing information in the Object Tree

Stick to the naming convention of ALPHA (well documented).

For example, to access the number of VDET hits for a charged track

- `qvec` is of type `QVEC`
- `QVEC` has a data member `QD` of type `QDET`
- `QDET` has a data member `AL` of type `FRTL`
- `FRTL` has a data member `NV` of type `int`
- Thus,

```
Num_VDET_hits = qvec -> QD -> AL -> NV;
```

What has changed?

Slightly Modified:

- `AlObject.h`

- `AlTrack.h,   AlEflw.h,   AlGamp.h`
- `AlTrack.cpp, AlEflw.cpp, AlGamp.cpp`

- `QvecBase.h`
- `QvecBase.cpp`

Completely changed:

- `AlphaBanks.h` –simplier and more concise
- `AlphaBanks.cpp` –fills the Object Tree and the other objects

New:

- `BankClasses.h` –Class definitions for nodes in the Tree
- `BankClasses.cpp` –fills and attaches nodes in the Object Tree

- `BankPatterns.h` –Patterns which overlay the BOS memory structure

Important OO considerations

WORK COMPLETED:

Just as in the earlier ALPHA++ versions, the `qvec` object tree is only accessable through objects derived from QvecBase:

- AlTrack
- AlEflw
- AlGamp
- etc.

This forces the user to access all information only through designated objects and prevents the user from having direct access to `qvec`.

WORK IN PROGRESS:

write protect `qvec` against accidental modification...this is not yet implemented...suggestions are welcome!

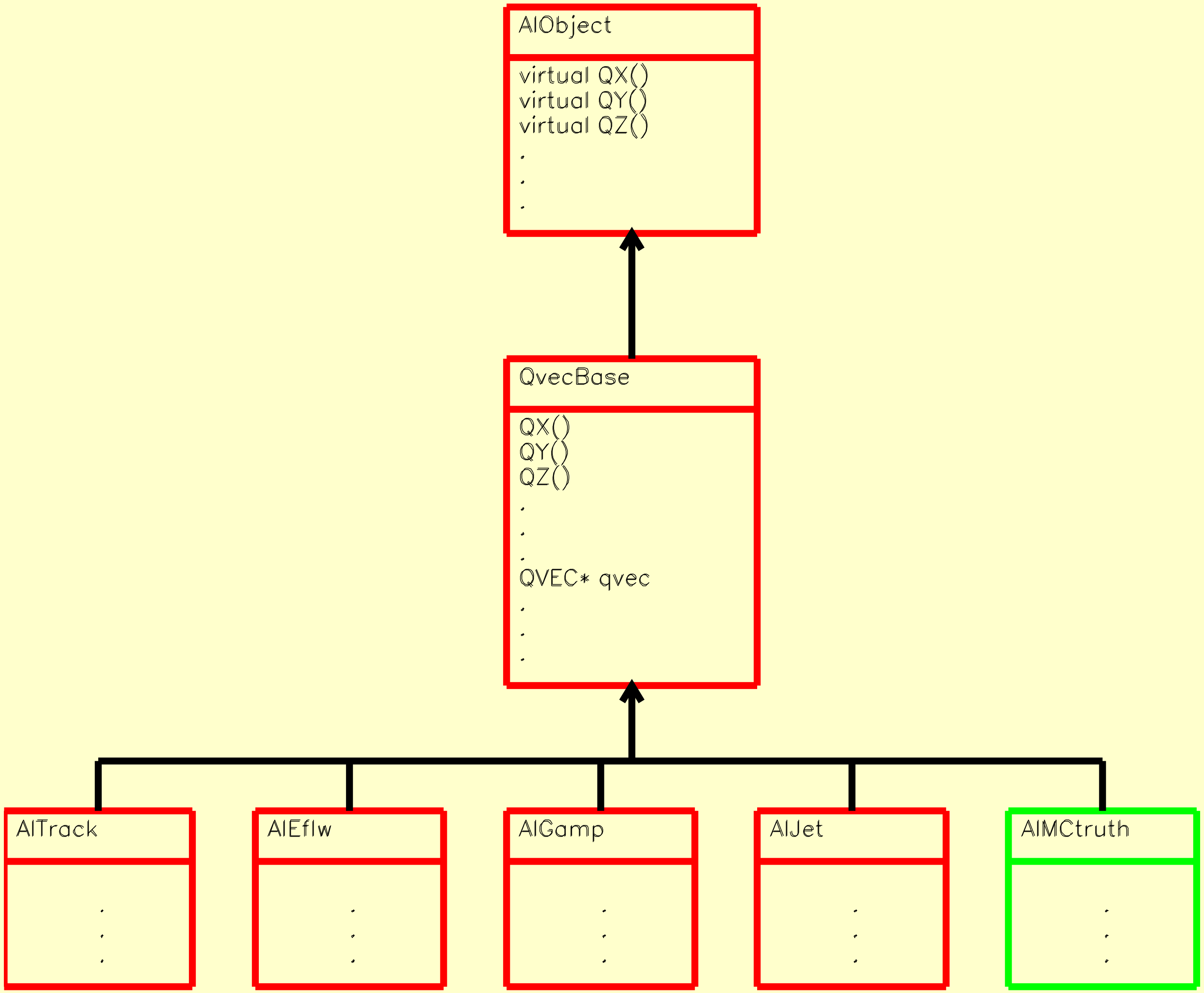
Monte Carlo Truth Implementation

Add a class which inherits from QvecBase called AlMCtruth which:

- has an AlephVector of pointers to Mother(s)
- has an AlephVector of pointers to Daughters
- privides its Monte Carlo Particle Identification Code

```
class AlMCtruth : public QvecBase
{
public:
   AlMCtruth();                                        // constructor
   AlMCtruth(const AlMCtruth&);                        // copy constructor
   AlMCtruth& operator= (const AlMCtruth& OldM);       // assignment operator
  ~AlMCtruth();                                        // destructor
   ALEPHTYPE TYPE(){return MCTRUTH;}                   // get the object type
   void setMotherVector(   AlephVector<AlMCtruth*> );// set mothers
   void setDaughterVector( AlephVector<AlMCtruth*> );// set daughters
   AlephVector<AlMCtruth*> getMotherVector();          // get mothers
   AlephVector<AlMCtruth*> getDaughterVector();        // get daughters
   int PA() { return qvec -> PA; }                     // MC particle code
   int NO() { return qvec -> NO; }                     // number of Mothers
   int ND() { return qvec -> ND; }                     // number of Daughters
   char* name();                                       // name of particle
private:
   AlephVector<AlMCtruth*> Mother;                     // vector of pointers to mother(s)
   AlephVector<AlMCtruth*> Daughter;                   // vector of pointers to daughters
};
```

Monte Carlo Truth Implementation



Conclusion

- The ALPHA BOS Banks have been converted into a **C++ tree-like data structure**.

- This **defines a border**, beyond which ALPHA++ is independent of FORTRAN.

- **Backward compatibility** has almost been achieved:
  - The implementation is transparent to the user
  - AlTrack, AlEflw, AlGamp, etc., need to be slightly modified

- A new **Monte Carlo Truth class** has been developed using the new tree-like data structure.

Richard Cavanaugh
Universität Heidelberg

Future

- Implement a C++ tree-like data structure for QVRT
- Provide additional protection against modifying `qvec` and `qvrt`
- Di-fermion physics analysis already under development...