# Hall C Software meeting 03/20/2012

- Note: These are two talks rolled into one
- 1. The outline of the talk for the software readiness review in June
- 2. Strategy on how to proceed writing the HallC++ code (expect nuts & bolts)

# Hall C Software meeting 03/20/2012

- This is done assuming that I will present something at the Software readiness meeting in June, 2012
- As the charge said "present outline" so that's what this document contains. Each topic listed in red (there are six of them) below will be developed on a slide/more in the actual talk

# Hall C Software meeting 03/20/2012

- Introduction.
  - Current Software: "Engine" – the Hall C in the 6 GeV era
- Justifying the transition from F77/PAW to C++/ROOT
- Transition strategy.
  - Man power & responsibilities.
  - Collaboration with Hall A
  - "Equivalence principle"

# Hall C Software meeting 03/20/2012

- Key Challenges/Milestones
- Current Status of the Hall C++ engine (insert here whatever catchy name we want to call the new software)
  - Checklist with things done, in progress, not started yet.
  - (Hopefully) a demo of the part of the C++ software that works by June

# Hall C Software meeting 03/20/2012

- Summary and Outlook
- Benchmarking
- Deployment
- Relationship with other Hall C software
- Continuous Collaboration with Hall A (other two Halls)

# Blank page

# HallC++ Strategy thoughts

Design features & goals, known constraints & limitations:

- Reproduce engine results* & speed
- User-centered features
  - Ease of use/familiarity for experienced Hall C users
  - Ease of maintaining/modifying/expanding code
- Synergy between Halls
- Time, man power, skill level

# HallC++ Strategy thoughts

*Critical design decisions (based on constraints…)*

- *Use C++/ROOT*
- *Maintain/Document engine (comparisons, etc.)*
- *Use as much of podd (Hall A C++/ROOT software) as possible*

# Issue #1: Naming (and not only) Convention

- Most podd classes are nicely/logically laid out and appropriate for their intended use
- Which means that we probably want similar set of Classes for hcana as well
- However they tend to be named: THa…
- ex: THaSpectrometer, THaInterface, THaScintillator – you get the idea

- There are a few potential problems/solutions that come to mind

# What's in a name?...

*In coding the Hall C equivalent of these Classes we could (I'll use the spectrometer class as an example):*

- *1. class THcSpectrometer: public THaSpectrometer {…*
  - *Discuss here pros and cons*
- *2. keep THaSpectrometer and just define a Hall C – specific constructor: void THaSpectrometer(…put here Hall C initialization stuff…)*
  - *Discuss pros and cons*
- *3. go back one class (or two) and do:*
- *class THcSpectrometer: public THaAparratus {…*
  - *Discuss pros and cons*

# GN's Take

*My $0.02:*

- *use (1) when possible…*
- *but not to the detriment of readability/ease of use.*
- *when (1) not feasible use (3).*
- *still glance at (1) as a model*
- *never use (2)!*
- *do not fight THa… (read it as "THis is a…" not as "The Hall A…"*

# An Example

- *Steve produced an THcInterface (which can read Hall C – style parameters in hcana). It works! :D*
- *THcInterface inherits from TRint*
- *class THcInterface: public TRint…*
- *One can get the same functionality by doing:*
- *class THcInterface: public THaInterface…*
- *…as shown on the next slide.*

# An Example

- sdiff -s hcana_orig/src/THcInterface.h hcana/src/THcInterface.h

```
#include "TRint.h"                                  |        //#include "TRint.h"
                                                    >        #include "THaInterface.h"
class THcInterface : public TRint {                 |        class THcInterface : public
THaInterface {
  static TClass* GetDecoder();                      | //static TClass* GetDecoder();
  static TClass* SetDecoder( TClass* c );           |          //static TClass* SetDecoder(
TClass* c );
```

# *Inheriting from THaInterface greatly reduces overhead in THcInterface:*

```
sdiff -s THcInterface.cxx ../../hcana_orig/src/ > ../../txt2

  THaInterface( appClassName, argc, argv, options, numOptions |   TRint( appClassName, argc, argv, options, numOptions, kTRUE
  SetPrompt("hcana [%d] ");                                   |          SetPrompt("analyzer [%d] ");
  ///  gHaVars    = new THaVarList;                            |          gHaVars     = new THaVarList;
  ///gHaCuts     = new THaCutList( gHcVars );                 |          gHaCuts     = new THaCutList( gHaVars );
  ///gHcApps     = new TList;                                 |          gHaApps     = new TList;
  ///gHcScalers = new TList;                                  |          gHaScalers = new TList;
  ///gHcPhysics = new TList;                                  |          gHaPhysics = new TList;
  ///  gHcDecoder = THaCodaDecoder::Class();                  |          gHaDecoder = THaCodaDecoder::Class();
  ///  gHaDB       = new THaFileDB();                         |          //  gHaDB       = new THaFileDB();
  ///gHcTextvars = new THaTextvars;                           |          gHaTextvars = new THaTextvars;
  ///  TTree::SetMaxTreeSize(1500000000);                     |          TTree::SetMaxTreeSize(1500000000);
                                                             |          TString s = gSystem->Getenv("ANALYZER");
  /*                                                         |          if( s.IsNull() ) {
    TString s = gSystem->Getenv("ANALYZER");                 |            s = HA_INCLUDEPATH;
    if( s.IsNull() ) {                                       <
  s = HA_INCLUDEPATH;                                        <
  // Give preference to $ANALYZER/include                    |            // Give preference to $ANALYZER/include
  TString p = s+"/include";                                  |            TString p = s+"/include";
  void* dp = gSystem->OpenDirectory(p);                      |            void* dp = gSystem->OpenDirectory(p);
  if( dp ) {                                                 |            if( dp ) {
  gSystem->FreeDirectory(dp);                                |              gSystem->FreeDirectory(dp);
  s = p;                                                     |              s = p;
  } else                                                     |            } else
  s = s+"/src " + s+"/hana_decode " + s+"/hana_scaler";      |              s = s+"/src " + s+"/hana_decode " +
s+"/hana_scaler";
  */                                                         |
/*                                                           <
  return gHcDecoder;                                         |          return gHaDecoder;
  gHcDecoder = c;                                            |          gHaDecoder = c;
  return gHcDecoder;                                         |          return gHaDecoder;
*/                                                           |
```

# An Example (IV)

- *The added benefit is that we get access to the variables (and methods!) defined in THaInterface:*

```
ifarm1101> ./hcana
Gabriel Niculescu: Welcome to ROOT
  ************************************************
  *                                              *
  *           W E L C O M E  to  the             *
  *         H A L L C ++  A N A L Y Z E R        *
  *                                              *
  *        Release     1.5.22        Mar 20 2012 *
  *  Based on ROOT  5.30/02          Sep 21 2011 *
  *                                              *
  *           For information visit              *
  *        http://hallaweb.jlab.org/root/        *
  *                                              *
  ************************************************

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
hcana [0] gHa

gHaVars
gHaCuts
gHaApps
gHaScalers
gHaPhysics
gHaRun
gHaDB
gHaTextvars
hcana [0] gHcParms
```

# Another Example

- *In some cases we might be better off (IMHO) inheriting not from the similar-name Hall A class but from it's ancestor(s)*
- *Take for instance* **THaSpectrometer**
- *At first one would say: "let's derive THcSpectrometer from THaSpectrometer!"*
- *There are number of methods that certainly make sense to have in Hall C:*

# Another Example

```
class THaSpectrometer : public THaApparatus, public THaTrackingModule,
                        public THaVertexModule {

public:
  virtual ~THaSpectrometer();

  // Main functions
  virtual void            Clear( Option_t* opt="")
  virtual Int_t           CoarseTrack();
  virtual Int_t           CoarseReconstruct();
  virtual Int_t           Track();
  virtual Int_t           Reconstruct();
  virtual Int_t           CalcPID();
  virtual Int_t           FindVertices( TClones
  virtual Int_t           TrackCalc() = 0;

  // Auxiliary functions
  virtual Int_t           AddDetector( THaDetec
  virtual Int_t           AddPidParticle( const
                                          const
                                          Doubl
  virtual void            DefinePidParticles();
  virtual Int t           DefineVariables( EMod

          Int_t           GetNpidParticles() co
          Int_t           GetNpidDetectors() co
  const   THaParticleInfo* GetPidParticleInfo( I
  const   THaPidDetector* GetPidDetector( Int_t
          Int_t           GetNTracks()  const {
          TClonesArray*   GetTracks()   const {
          TClonesArray*   GetTrackPID() const {
  virtual const TVector3& GetVertex()   const;
  virtual Bool_t          HasVertex() const;

          Bool_t          IsDone( UInt_t stage
          Bool_t          IsPID() const       {

          THaTrack*       GetGoldenTrack() const { return fGoldenTrack; }

          void            SetGoldenTrack( THaTrack* t ) { fGoldenTrack = t; }
          void            SetPID( Bool_t b = kTRUE )    { fPID = b; }

// The following is specific to small-acceptance pointing spectrometers
// using spectrometer-specific coordinates such as TRANSPORT
const     TRotation&      GetToLabRot() const { return fToLabRot; }
const     TRotation&      GetToTraRot() const { return fToTraRot; }
const     TVector3&       GetPointingOffset() const { return fPointingOffset; }
          Double_t        GetThetaGeo() const { return fThetaGeo; }
          Double_t        GetPhiGeo()   const { return fPhiGeo; }
          Double_t        GetThetaSph() const { return fThetaSph; }
          Double_t        GetPhiSph()   const { return fPhiSph; }
          Double_t        GetPcentral() const { return fPcentral; }
          Double_t        GetCollDist() const { return fCollDist; }

virtual void              TrackToLab( THaTrack& track, TVector3& pvect ) const;
virtual void              TransportToLab( Double_t p, Double_t th,
                                          Double_t ph, TVector3& pvect ) const;
virtual void              LabToTransport( const TVector3& vertex,
                                          const TVector3& pvect,
                                          TVector3& tvertex,
                                          Double_t* ray ) const;
          void            LabToTransport( const TVector3& vertex,
                                          const TVector3& pvect,
                                          Double_t* ray ) const;
```

# Another Example

```
        THaTrack*        GetGoldenTrack() const { return fGoldenTrack; }

        void             SetGoldenTrack( THaTrack* t ) { fGoldenTrack = t; }
        void             SetPID( Bool_t b = kTRUE )    { fPID = b; }

// The following is specific to small-acceptance pointing spectrometers
// using spectrometer-specific coordinates such as TRANSPORT
const   TRotation&       GetToLabRot() const { return fToLabRot; }
const   TRotation&       GetToTraRot() const { return fToTraRot; }
const   TVector3&        GetPointingOffset() const { return fPointingOffset; }
        Double_t         GetThetaGeo() const { return fThetaGeo; }
        Double_t         GetPhiGeo()   const { return fPhiGeo; }
        Double_t         GetThetaSph() const { return fThetaSph; }
        Double_t         GetPhiSph()   const { return fPhiSph; }
        Double_t         GetPcentral() const { return fPcentral; }
        Double_t         GetCollDist() const { return fCollDist; }

virtual void             TrackToLab( THaTrack& track, TVector3& pvect ) const;
virtual void             TransportToLab( Double_t p, Double_t th,
                                         Double_t ph, TVector3& pvect ) const;
virtual void             LabToTransport( const TVector3& vertex,
                                         const TVector3& pvect,
                                         TVector3& tvertex,
                                         Double_t* ray ) const;
        void             LabToTransport( const TVector3& vertex,
                                         const TVector3& pvect,
                                         Double_t* ray ) const;
```

*<- Here is the missing line (and others like it)*

*For a lot of these we either do not have the Hall C equivalent, or it's name is totally different than Hall A's*

- *My $0.02 is that this is a good candidate to use (3) rather than (1)*

# (instead of) Conclusion

- I hope we can get a (small) set of "rules/guidelines" that should be applied in the process of coding hcana
- I outlined (proposals for) some of them
- I will feel a lot more comfortable if said r/g are group decision.