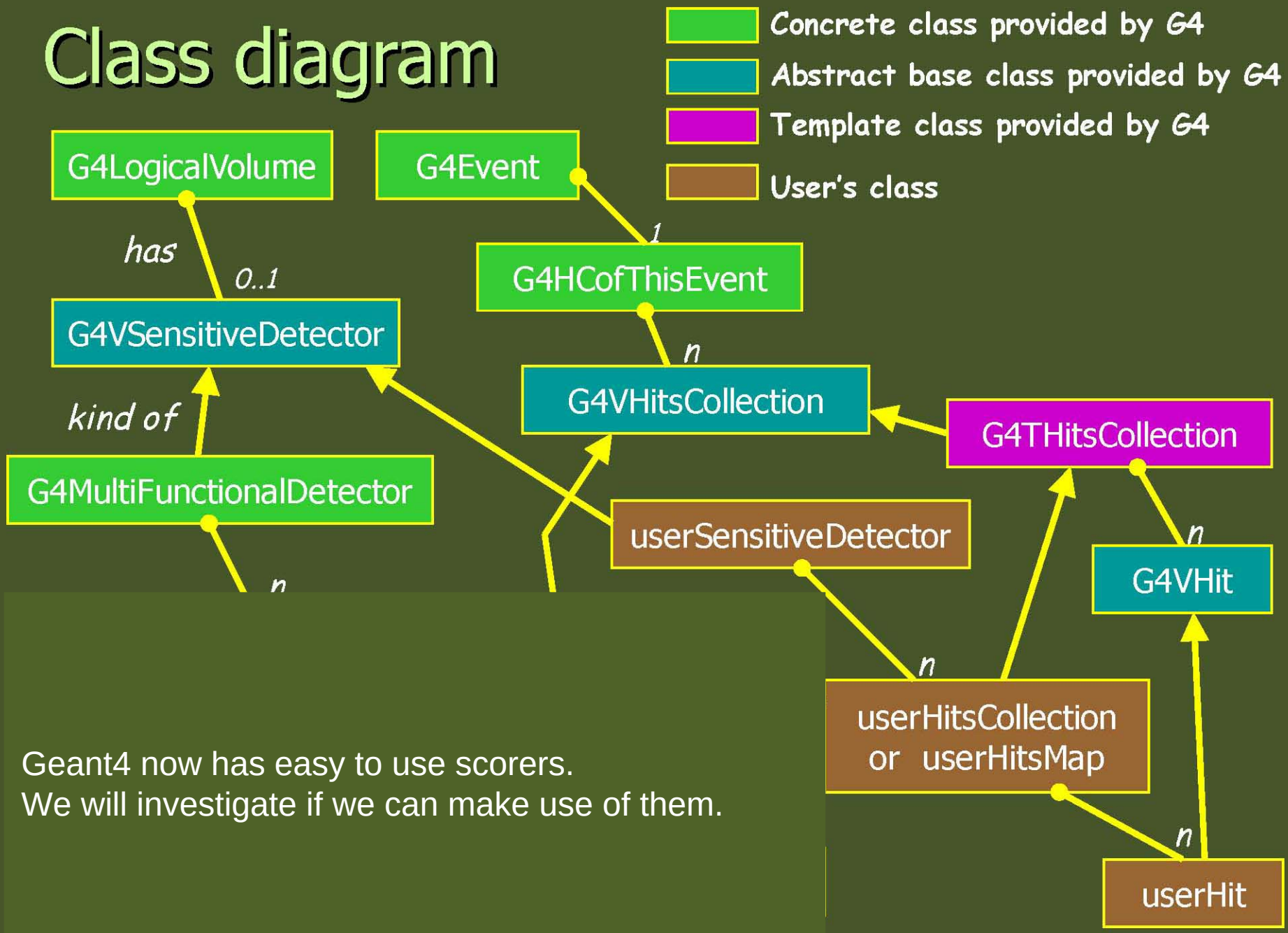
Hits and Digits

Geant4 introduction
2008/03/18
Nobu Katayama
KEK

Simulating the detector response

- Create hits in the sensitive detector
  - Record the information of physical interaction of a track in the sensitive region of the detector
- Hits should also be generated for the backgrounds
- Convert the hits information to “signal” from the sensors
- “Digitize” the information simulating the read-out electronics and DAQ

Class diagram



Geant4 now has easy to use scorers.
We will investigate if we can make use of them.

Sensitive detector and Hit

- Each Logical Volume can have a pointer to a sensitive detector.
  - Then this volume becomes sensitive
- Hit is a snapshot of the physical interaction of a track or an accumulation of interactions of tracks in the sensitive region of your detector
- A sensitive detector creates hit(s) using the information given in G4Step object. The user has to provide his/her own implementation of the detector response.
- Hit objects, which are still the user's class objects, are collected in a G4Event object at the end of an event

B4SensitiveDetectorBase

- inherits from G4VSensitiveDetector
- is a placeholder for (multiple) Hits collections
- has a pointer to DetBase
- The user should derive your own SensitiveDetector class from this class and implement CreateCollection, Initialize, ProcessHits and AddbgOne functions

Step, StepPoint and Touchable

- When a track trajectory goes through a sensitive detector, It invokes SensitiveDetector::ProcessHits(G4Step, G4TouchableHistory)
- G4Step has PreStepPoint from which
  - Position in world coordinate system
  - Material
  - Track etc.
- G4TouchableHistory is a vector of information for each geometrical hierarchy with
  - copy number
  - transformation/rotation to its mother etc.

B4CDC_SensitiveDetector::ProcessHits(G4Step * aStep, G4TouchableHistory *) {

```
const G4double edep = aStep->GetTotalEnergyDeposit();
  if (edep == 0.) return false;
  //...Get step information...
  const G4Track & t = * aStep->GetTrack();
  const G4double charge = t.GetDefinition()-
     >GetPDGCharge();
  if (charge == 0.) return false;
  const G4VPhysicalVolume & v = * t.GetVolume();
  const G4StepPoint & in = * aStep->GetPreStepPoint();
  const G4StepPoint & out = * aStep->GetPostStepPoint();
  const G4ThreeVector & posIn = in.GetPosition();
  const G4ThreeVector & posOut = out.GetPosition();
  const G4ThreeVector & mom = t.GetMomentum();
static const B4CDC *cdc(NULL);
if(NULL==cdc) {
    //     cdc = dynamic_cast<const B4CDC
      *>(B4DetectorConstruction::Instance()->det("cdc"));
    cdc = dynamic_cast<const B4CDC *>(getDetBase());
}
 //...Get layer ID...
 const unsigned layerId = v.GetCopyNo();
 static const B4CDC_GeometryDB *cdcgp(NULL);
 cdcgp = &(cdc->geometryDB());
 const B4CDC_GeometryDB & cdcg(*cdcgp);
 //...Calculate cell IDs...
 const unsigned idIn = cdcg.cellId(layerId, posIn);
 const unsigned idOut = cdcg.cellId(layerId, posOut);
 //...Calculate drift length...
 const bool magneticField = false;
 const B4CDC_Layer & l = cdcg.layer(layerId);
 std::vector<unsigned> wires = WireId(idIn, idOut,
    l.nWires());
 for (unsigned i = 0; i < wires.size(); i++) {
  const B4CDC_Wire & w = * l[wires[i]];
  double distance = 0;
  // for each cell in phi, calculate the drift distance
  // and create hits
  B4CDC_Hit * hit = new B4CDC_Hit(w, distance);
  getHcdc<B4CDC_HitsCollection>()->insert(hit);
```

User Hits class

- Hit is a user-defined class derived from G4VHit
- You can store various types information by implementing your own concrete Hit class. For example:
  - Position and time of the step
  - Momentum and energy of the track
  - Energy deposition of the step
  - Geometrical information
  - or any combination of above
- Hit objects of a concrete hit class must be stored in a dedicated collection which is instantiated from G4THitsCollection template class
- The collection will be associated to a G4Event object via G4HCofThisEvent
- Hits collections are accessible
  - through G4Event at the end of event
  - through G4SDManager during processing an event

B4CDC_Hit

```
class B4CDC_Hit : public G4VHit {
  friend class B4CDC_Digi;
 public:
B4CDC_Hit(const B4CDC_Wire &, double driftLength);
  ~B4CDC_Hit() {}
  void Print() {}
  void Draw() {}
  Belle::Panther_ID Store(void) const;

  inline void * operator new(size_t);
  inline void operator delete(void *aHit);

  struct sort_functor_cmp_hits {
    bool operator() (const Superb::B4CDC_Hit * const &a,
      const Superb::B4CDC_Hit * const &b) const;
  };
  G4bool SameWire(const B4CDC_Hit &h) const {
    return _wire->id() == h._wire->id();
  }
 private:
  const B4CDC_Wire * _wire;
  const G4double _driftLength;
  unsigned _state;
  static Belle::Reccdc_wirhit_Manager &mgr;
  static Belle::Geocdc_wire_Manager &geo_mgr;
};
```

```
// special new/deletes
 extern G4Allocator<B4CDC_Hit> B4CDC_HitAllocator;
 inline void* B4CDC_Hit::operator new(size_t) {
   void *aHit;
   aHit = (void*) B4CDC_HitAllocator.MallocSingle();
   return aHit;
 }
 inline void B4CDC_Hit::operator delete(void *aHit) {
   B4CDC_HitAllocator.FreeSingle((B4CDC_Hit*) aHit);
 }
```

```
// G4THitsCollection
 typedef G4THitsCollection<Superb::B4CDC_Hit>
       B4CDC_HitsCollection;
```

Digitizer and Digit

- Digit represents a detector output (e.g. ADC/TDC count, trigger signal, etc.)
- Digit is created with one or more hits and/or other digits by a user's concrete implementation derived from **`G4VDigitizerModule`**. (B4DigitizerBase for Belle)
- In contradiction to the sensitive detector which is accessed at tracking time automatically, the digitize() method of each **`G4VDigitizerModule`** must be explicitly invoked by the user’s code (e.g. at EventAction)

B4DigitizerBase

- inherits from G4VDigitizerModule
- is a placeholder for (multiple) digiCollections
- has a pointer to B4SensitiveDetectorBase
- The user should derive your own Digitizer class from this class and implement Digitize and Store functions
- Digitizers are created (for now) in B4EventAction::BeginOfEventAction and Digitize/Store are called from EndOfEventAction

B4CDC_Digitizer::Digitize() {

```
const B4DetBase &base(*(sd()->getDetBase()));
  const B4CDC_HitsCollection *hcdc = getHc<B4CDC_HitsCollection>(base);
  if(hcdc) {
    m_digiCollection.clear();
    m_digiCollection.push_back(new B4CDC_DigiCollection( base.DGname(),
     collectionName[0] ));
    std::sort(hcdc->GetVector()->begin(), hcdc->GetVector()->end(),
              B4CDC_Hit::sort_functor_cmp_hits());
    const B4CDC_Hit *lastHit(NULL);
    B4CDC_Digi *dg(NULL);
    for(std::vector<B4CDC_Hit*>::const_iterator it=hcdc->GetVector()->begin();
      it != hcdc->GetVector()->end(); ++it) {
     if(lastHit==NULL || !(*it)->SameWire(*lastHit)) {
      dg = new B4CDC_Digi(**it);
      getDc<B4CDC_DigiCollection> ()->insert(dg);
     } else if(lastHit!=NULL && (*it)->SameWire(*lastHit) && dg!=NULL) {
      dg->SetAdc(5678+dg->GetAdc());
     }
     lastHit = *it;
    }
    StoreDigiCollection( m_digiCollection[0] );
}
```

User Digi class

- Ideally the digi class should have the same information as real detector readout such as ADC/TDC information with smearing, electronics noise, backgrounds, non linearity simulated, through pipelined electronics/time window and sparsification and other software in the read-out systems
- In reality we do as detailed as we can at any given time in the development in order to satisfy the need for the particular simulation jobs
  - For example, in CDC, smearing, efficiency, multiple hits in a cell are simulated at this moment

Writing them out

- For now we use panther format
  - write tdf file if necessary or fill the existing tables
- For reading and writing panther table, see
  - http://belle.kek.jp/group/software/slides/Panther/soft_PANTHER.html
- For now we might use hits information in the reconstruction software as “digitize” and “undigitize” may not be important for the kind of things we are now looking at

Summary

- I think you can
  - Start with what you need
  - Complete the chain (simulation $\rightarrow$ reconstruction $\rightarrow$ physics)
  - Refine as necessary
- If you would like a different approach, please let us know