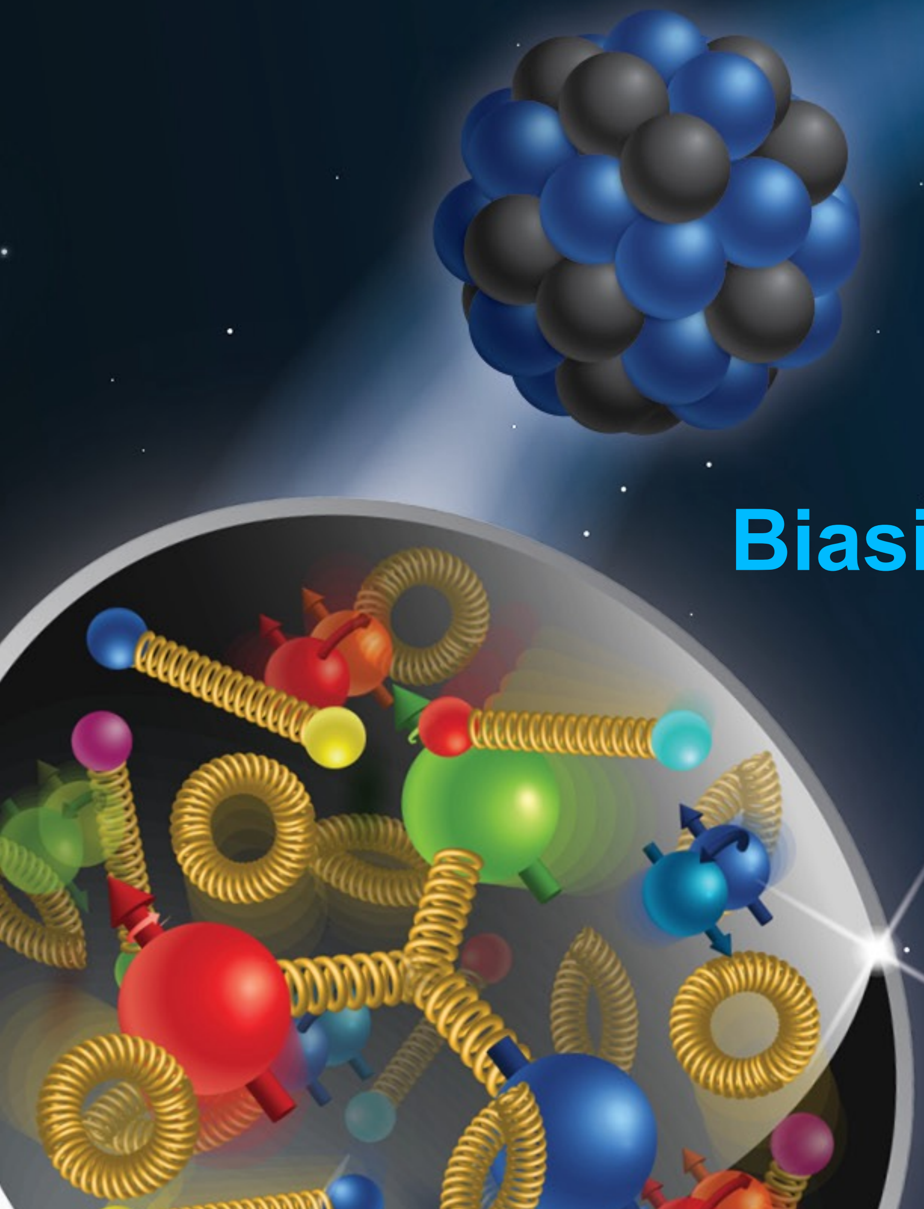


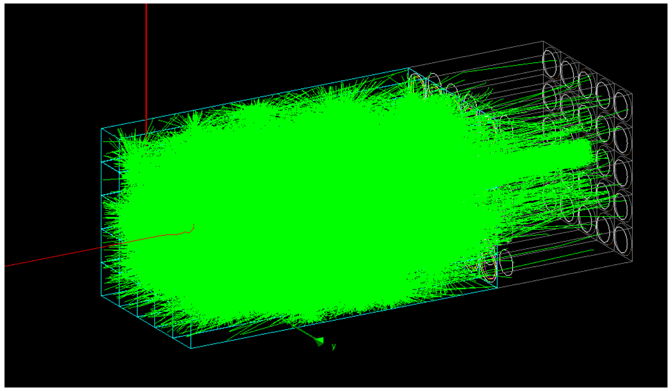
Biasing and Scoring

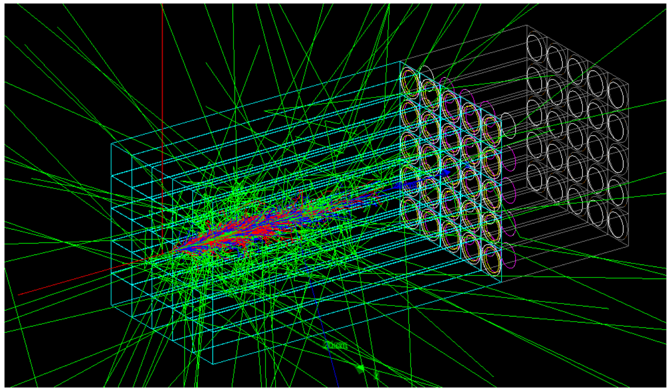
Makoto Asai (JLab/SCT)
asai@jlab.org

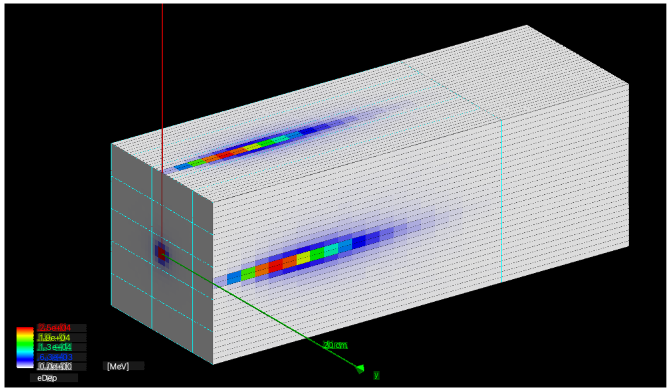
Preamble

- Biasing in general is a very powerful tool for boosting the simulation.
- Currently, one has to implement a dedicated sensitive detector to score, for example, energy deposition made by a biasing "process".
  - This restriction is somehow acceptable for a running experiment where detector configuration is fixed.
  - But it is not the case for detector designing phase where one typically tries various detector parameters and needs rather quick results.

5 GeV e-







- Full simulation with optical photon transport to photo-multipliers
  - 18.41 sec/event

x 154

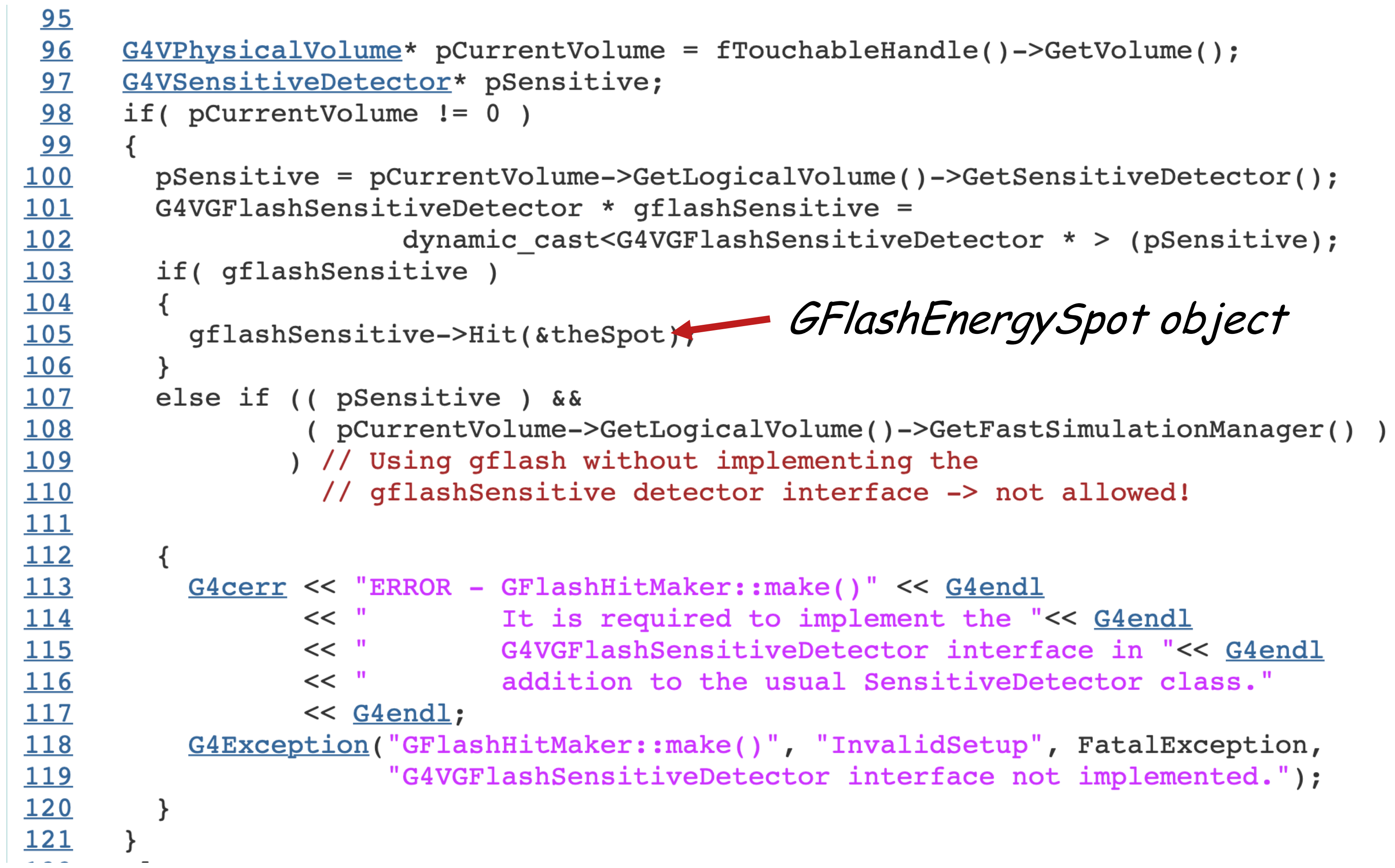
- Full simulation without optical photon transport
  - 0.119 sec/event

x 137

- Shower parameterization with GFlash
  - 0.00087 sec/event

Jefferson Lab

GFlashHitMaker::make() – version 11.0.p02



```

G4VPhysicalVolume* pCurrentVolume = fTouchableHandle()->GetVolume();
G4VSensitiveDetector* pSensitive;
if( pCurrentVolume != 0 )
{
  pSensitive = pCurrentVolume->GetLogicalVolume()->GetSensitiveDetector();
  G4VGFlashSensitiveDetector * gflashSensitive =
                 dynamic_cast<G4VGFlashSensitiveDetector * > (pSensitive);
  if( gflashSensitive )
  {
    gflashSensitive->Hit(&theSpot);
  }
  else if (( pSensitive ) &&
           ( pCurrentVolume->GetLogicalVolume()->GetFastSimulationManager() )
          ) // Using gflash without implementing the
            // gflashSensitive detector interface -> not allowed!

  {
    G4cerr << "ERROR - GFlashHitMaker::make()" << G4endl
           << "         It is required to implement the "<< G4endl
           << "         G4VGFlashSensitiveDetector interface in "<< G4endl
           << "         addition to the usual SensitiveDetector class."
           << G4endl;
    G4Exception("GFlashHitMaker::make()", "InvalidSetup", FatalException,
                "G4VGFlashSensitiveDetector interface not implemented.");
  }
}
```

GFlashEnergySpot object

G4FastSimHitMaker::make() – version 11.0.p03

```
  G4VSensitiveDetector* sensitive;
  if(currentVolume != 0)
  {
    sensitive = currentVolume->GetLogicalVolume()->GetSensitiveDetector();
    G4VFastSimSensitiveDetector* fastSimSensitive =
      dynamic_cast<G4VFastSimSensitiveDetector*>(sensitive);
    if(fastSimSensitive)
    {
      fastSimSensitive->Hit(&aHit, &aTrack, &fTouchableHandle);
    }
    else if(sensitive &&
            currentVolume->GetLogicalVolume()->GetFastSimulationManager())
    {
      G4cerr << "ERROR - G4FastSimHitMaker::make()" << G4endl
             << "        It is required to derive from the " << G4endl
             << "        G4VFastSimSensitiveDetector in " << G4endl
             << "        addition to the usual G4VSensitiveDetector class."
             << G4endl;
      G4Exception("G4FastSimHitMaker::make()", "InvalidSetup", FatalException,
                  "G4VFastSimSensitiveDetector interface not implemented.");
    }
  }
```

Two issues

- GFlashEnergySpot, G4FastHit
  - Somewhat equivalent to G4Step but with single step-point
  - We can live with G4Step with one G4StepPoint object assigned to both Pre- and Post-step-point.
    - Problem in G4Step destructor is fixed at v11.0.p03.

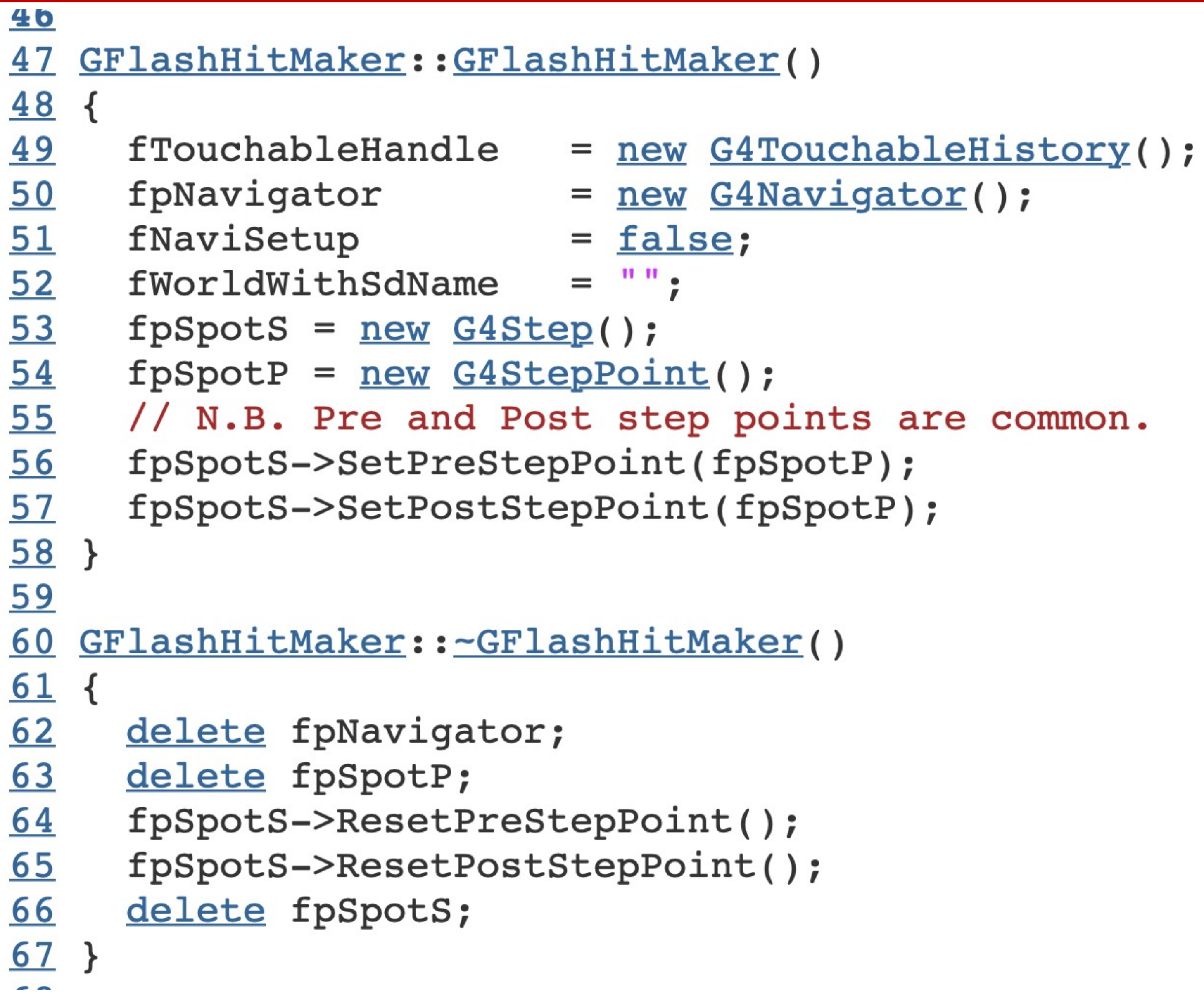
- GFlashHitMaker, G4FastSimHitMaker
  - How to set (the name of) the parallel world(s) is the issue.

```
void myDetectorDescription::ConstructSD()
{
  auto gflashModel = new GFlashShowerModel("GFlashModel",localRegion);
  auto param = new GFlashHomoShowerParameterisation(fDetectorMater);
  gflashModel->SetParameterisation(*param);

  auto particleBounds = new GFlashParticleBounds();
  gflashModel->SetParticleBounds(*particleBounds);

  auto hitMaker = new GFlashHitMaker();
  hitMaker->SetNameOfWorldWithSD(componentName);
  gflashModel->SetHitMaker(*hitMaker);
}
```

GFlashHitMaker::make() – version 11.0.p03

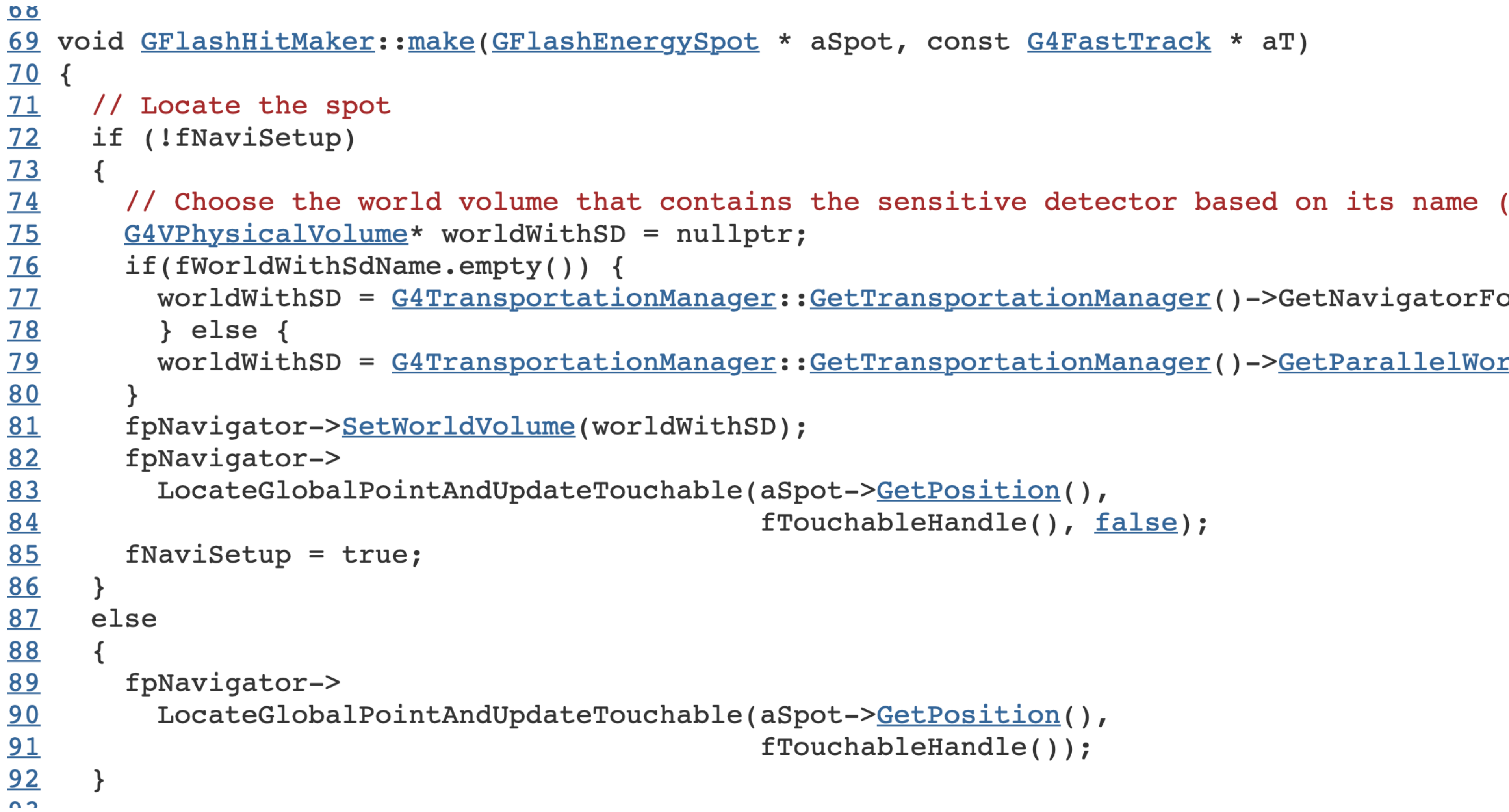
```cpp

  G4VPhysicalVolume* pCurrentVolume = fTouchableHandle()->GetVolume();
  G4VSensitiveDetector* pSensitive;
  if( pCurrentVolume != 0 )
  {
    pSensitive = pCurrentVolume->GetLogicalVolume()->GetSensitiveDetector();
    G4VGFlashSensitiveDetector * gflashSensitive =
                   dynamic_cast<G4VGFlashSensitiveDetector * > (pSensitive);
    if( gflashSensitive )
    {
      // set spot information:
      G4GFlashSpot theSpot(aSpot, aT, fTouchableHandle);
      gflashSensitive->Hit(&theSpot);
    }
    else if( pSensitive )
    {
      fpSpotS->SetTotalEnergyDeposit(aSpot->GetEnergy());
      fpSpotS->SetTrack(const_cast<G4Track*>(aT->GetPrimaryTrack()));
      fpSpotP->SetWeight(aT->GetPrimaryTrack()->GetWeight());
      fpSpotP->SetPosition(aSpot->GetPosition());
      fpSpotP->SetGlobalTime(aT->GetPrimaryTrack()->GetGlobalTime());
      fpSpotP->SetLocalTime(aT->GetPrimaryTrack()->GetLocalTime());
      fpSpotP->SetProperTime(aT->GetPrimaryTrack()->GetProperTime());
      fpSpotP->SetTouchableHandle(fTouchableHandle);
      fpSpotP->SetProcessDefinedStep(fpProcess);
      fpSpotP->SetStepStatus(fUserDefinedLimit);
      pSensitive->Hit(fpSpotS);
    }
  }
```

GFlashHitMaker::make() – version 11.0.p03



```
GFlashHitMaker::GFlashHitMaker()
{
  fTouchableHandle    = new G4TouchableHistory();
  fpNavigator         = new G4Navigator();
  fNaviSetup          = false;
  fWorldWithSdName    = "";
  fpSpotS = new G4Step();
  fpSpotP = new G4StepPoint();
  // N.B. Pre and Post step points are common.
  fpSpotS->SetPreStepPoint(fpSpotP);
  fpSpotS->SetPostStepPoint(fpSpotP);
}

GFlashHitMaker::~GFlashHitMaker()
{
  delete fpNavigator;
  delete fpSpotP;
  fpSpotS->ResetPreStepPoint();
  fpSpotS->ResetPostStepPoint();
  delete fpSpotS;
}
```

GFlashHitMaker::make() : parallel world for scoring



```
void GFlashHitMaker::make(GFlashEnergySpot * aSpot, const G4FastTrack * aT)
{
  // Locate the spot
  if (!fNaviSetup)
  {
    // Choose the world volume that contains the sensitive detector based on its name (
    G4VPhysicalVolume* worldWithSD = nullptr;
    if(fWorldWithSdName.empty()) {
      worldWithSD = G4TransportationManager::GetTransportationManager()->GetNavigatorFo
      } else {
      worldWithSD = G4TransportationManager::GetTransportationManager()->GetParallelWor
    }
    fpNavigator->SetWorldVolume(worldWithSD);
    fpNavigator->
      LocateGlobalPointAndUpdateTouchable(aSpot->GetPosition(),
                                          fTouchableHandle(), false);
    fNaviSetup = true;
  }
  else
  {
    fpNavigator->
      LocateGlobalPointAndUpdateTouchable(aSpot->GetPosition(),
                                          fTouchableHandle());
  }
```

Jefferson Lab
Thomas Jefferson National Accelerator Facility

Discussion

- Who, how, when,…

U.S. DEPARTMENT OF ENERGY Office of Science JSA