Unit Testing Derived G4VSensitiveDetector Classes

Liangliang Han, Shenjian Chen, Lei Zhang, John Chapman, Xiaozhong Huang

Nanjing University

Contents

- Introduction on Derived G4VSensitiveDetector Classes
- Unit test overview
- The structure of my test code
- Access non-public members
- CMakeLists.txt file and jobOptions file
- Test the member functions

Introduction on Derived G4VSensitiveDetector Classes

- Generate and collect hits
- Member variable

  WriteHandle<AtlasHitsVector> m_HitColl
- Member functions

  G4bool ProcessHits(G4Step*, G4TouchableHistory*)

  template <**class**... Args> **void** AddHit(Args&&... args)

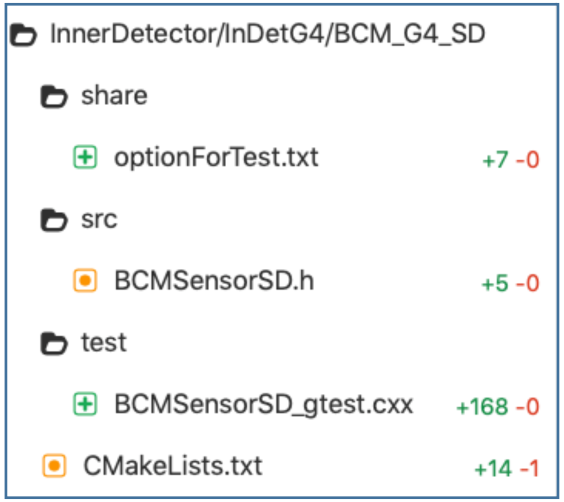
  void Initialize(G4HCofThisEvent *)

  void StartOfAthenaEvent ()

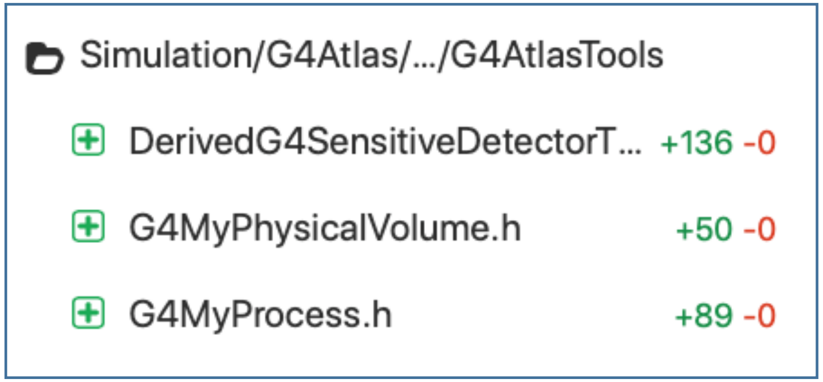
  void EndOfAthenaEvent()

Unit test overview

- The smallest unit is a member function of a class
- Google test as test framework
- Package structure

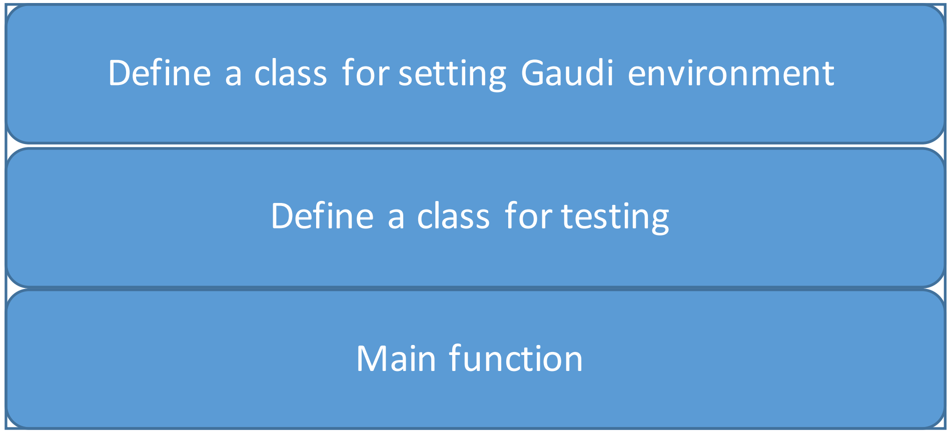


```
InnerDetector/InDetG4/BCM_G4_SD
  share
    optionForTest.txt        +7 -0
  src
    BCMSensorSD.h            +5 -0
  test
    BCMSensorSD_gtest.cxx  +168 -0
  CMakeLists.txt            +14 -1
```



```
Simulation/G4Atlas/.../G4AtlasTools
  DerivedG4SensitiveDetectorT...  +136 -0
  G4MyPhysicalVolume.h             +50 -0
  G4MyProcess.h                    +89 -0
```

The structure of my test code BCMSensorSD_gtest.cxx

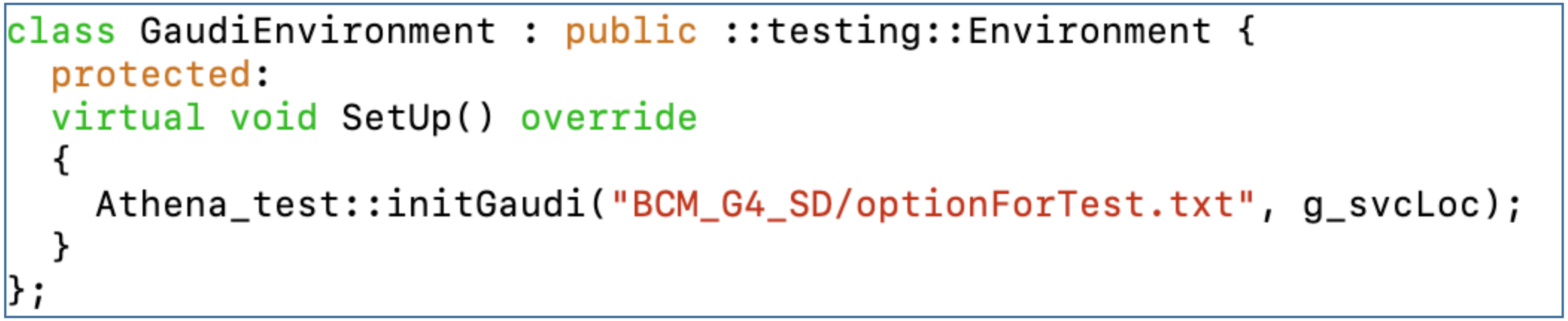

Define a class for setting Gaudi environment

Define a class for testing

Main function

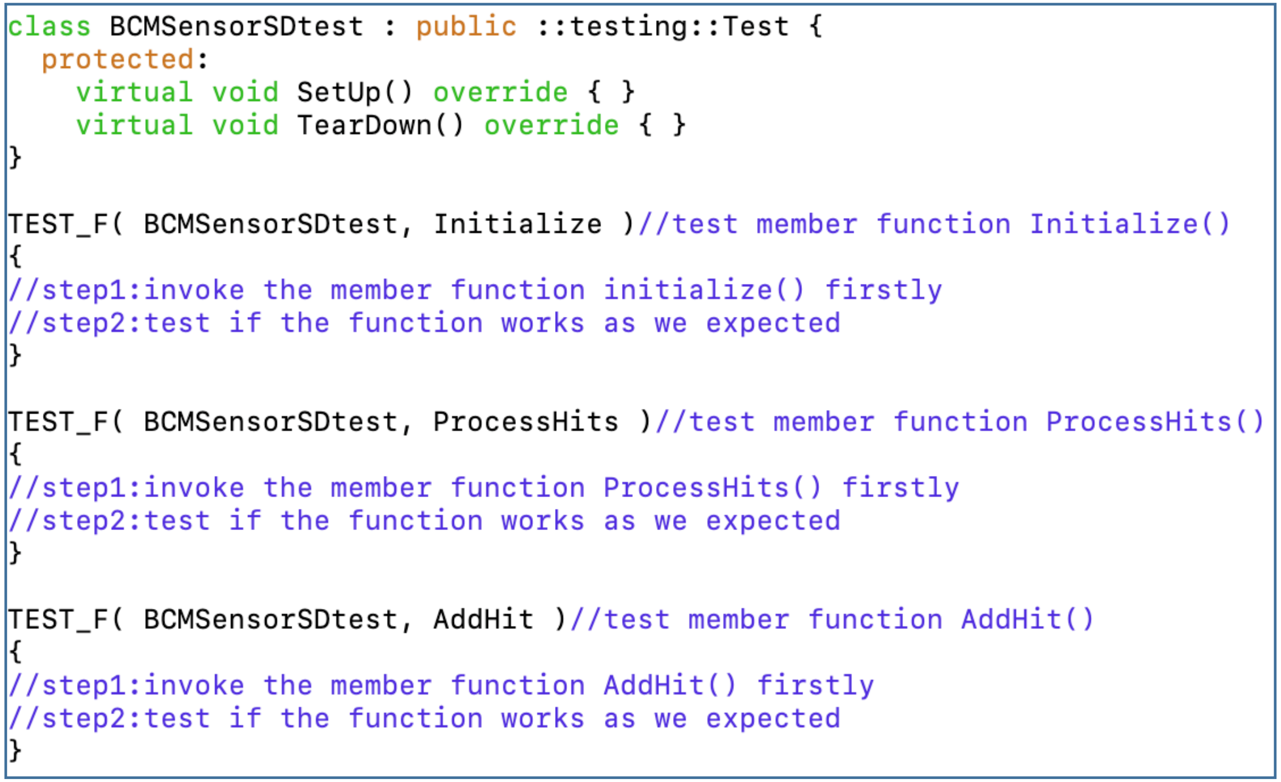
BCMSensorSD_gtest.cxx

- Define a class for setting Gaudi environment



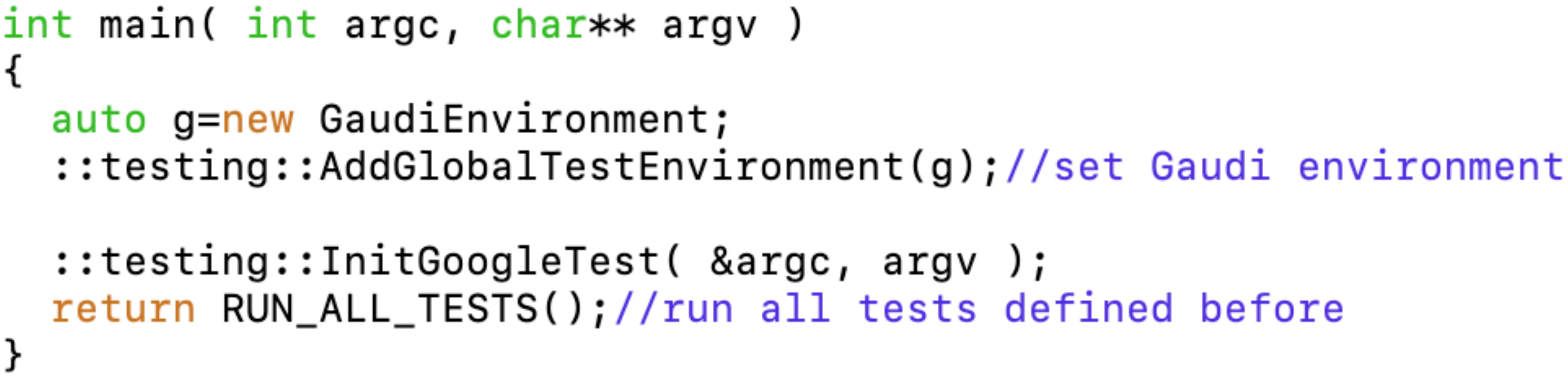
```
class GaudiEnvironment : public ::testing::Environment {
  protected:
  virtual void SetUp() override
  {
    Athena_test::initGaudi("BCM_G4_SD/optionForTest.txt", g_svcLoc);
  }
};
```

- Define a class for testing



```
class BCMSensorSDtest : public ::testing::Test {
  protected:
    virtual void SetUp() override { }
    virtual void TearDown() override { }
}

TEST_F( BCMSensorSDtest, Initialize )//test member function Initialize()
{
//step1:invoke the member function initialize() firstly
//step2:test if the function works as we expected
}

TEST_F( BCMSensorSDtest, ProcessHits )//test member function ProcessHits()
{
//step1:invoke the member function ProcessHits() firstly
//step2:test if the function works as we expected
}

TEST_F( BCMSensorSDtest, AddHit )//test member function AddHit()
{
//step1:invoke the member function AddHit() firstly
//step2:test if the function works as we expected
}
```

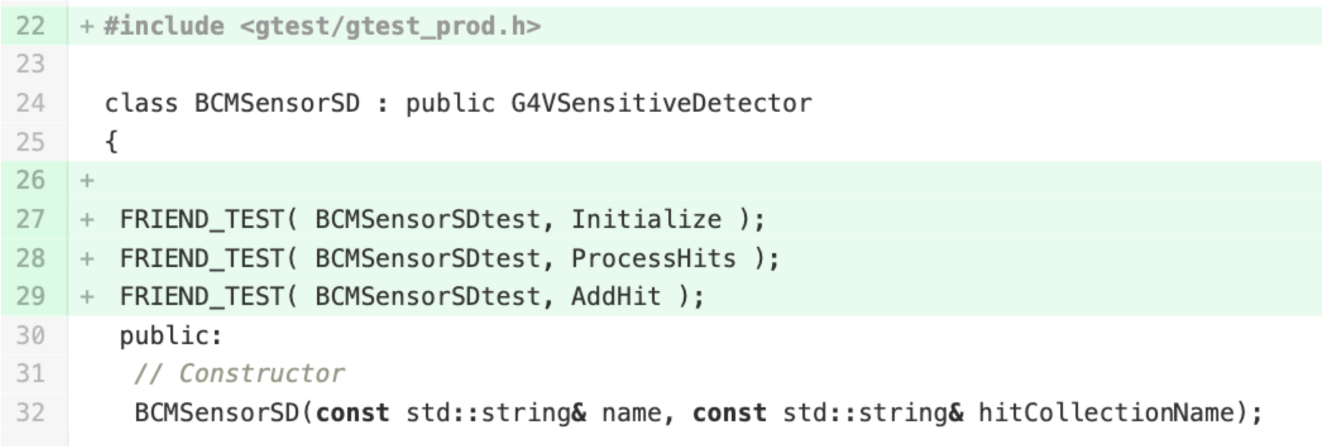
- Main function



```
int main( int argc, char** argv )
{
  auto g=new GaudiEnvironment;
  ::testing::AddGlobalTestEnvironment(g);//set Gaudi environment

  ::testing::InitGoogleTest( &argc, argv );
  return RUN_ALL_TESTS();//run all tests defined before
}
```

Access non-public members

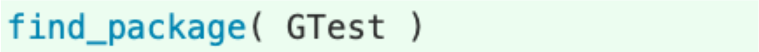
- In order to access the non-public members of tested class(i.e. BCMSensorSD) from test class(i.e. BCMSensorSDtest), add BCMSensorSDtest as a friend of BCMSensorSD.
- use the method provided by google test



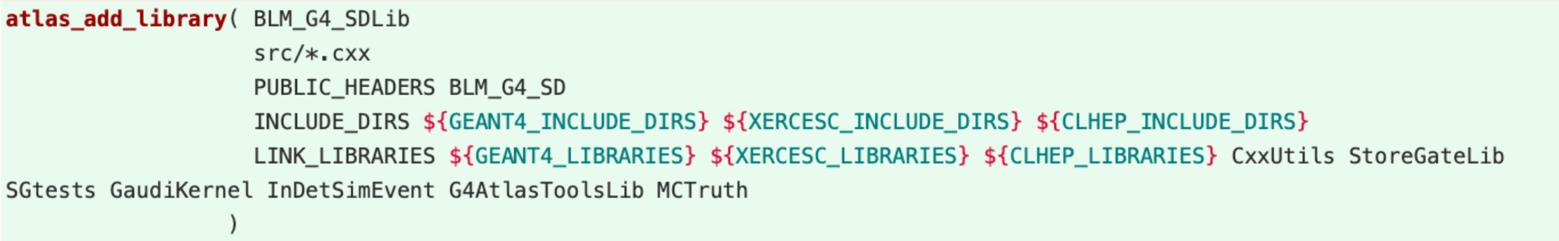
```
+ #include <gtest/gtest_prod.h>

  class BCMSensorSD : public G4VSensitiveDetector
  {
+
+  FRIEND_TEST( BCMSensorSDtest, Initialize );
+  FRIEND_TEST( BCMSensorSDtest, ProcessHits );
+  FRIEND_TEST( BCMSensorSDtest, AddHit );
   public:
    // Constructor
    BCMSensorSD(const std::string& name, const std::string& hitCollectionName);
```

CMakeLists.txt file

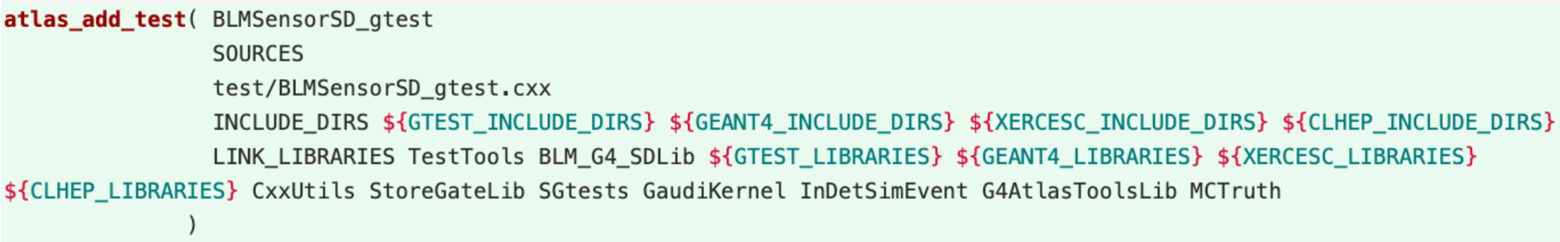
- Add the following lines:



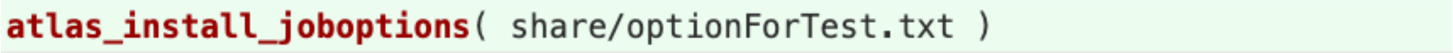
```
find_package( GTest )
```



```
atlas_add_library( BLM_G4_SDLib
                   src/*.cxx
                   PUBLIC_HEADERS BLM_G4_SD
                   INCLUDE_DIRS ${GEANT4_INCLUDE_DIRS} ${XERCESC_INCLUDE_DIRS} ${CLHEP_INCLUDE_DIRS}
                   LINK_LIBRARIES ${GEANT4_LIBRARIES} ${XERCESC_LIBRARIES} ${CLHEP_LIBRARIES} CxxUtils StoreGateLib
SGtests GaudiKernel InDetSimEvent G4AtlasToolsLib MCTruth
                 )
```

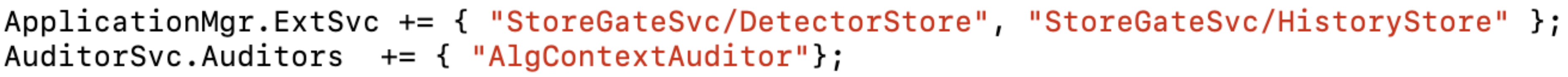


```
atlas_add_test( BLMSensorSD_gtest
                SOURCES
                test/BLMSensorSD_gtest.cxx
                INCLUDE_DIRS ${GTEST_INCLUDE_DIRS} ${GEANT4_INCLUDE_DIRS} ${XERCESC_INCLUDE_DIRS} ${CLHEP_INCLUDE_DIRS}
                LINK_LIBRARIES TestTools BLM_G4_SDLib ${GTEST_LIBRARIES} ${GEANT4_LIBRARIES} ${XERCESC_LIBRARIES}
${CLHEP_LIBRARIES} CxxUtils StoreGateLib SGtests GaudiKernel InDetSimEvent G4AtlasToolsLib MCTruth
              )
```



```
atlas_install_joboptions( share/optionForTest.txt )
```

jobOptions file optionForTest.txt for setting Gaudi environment



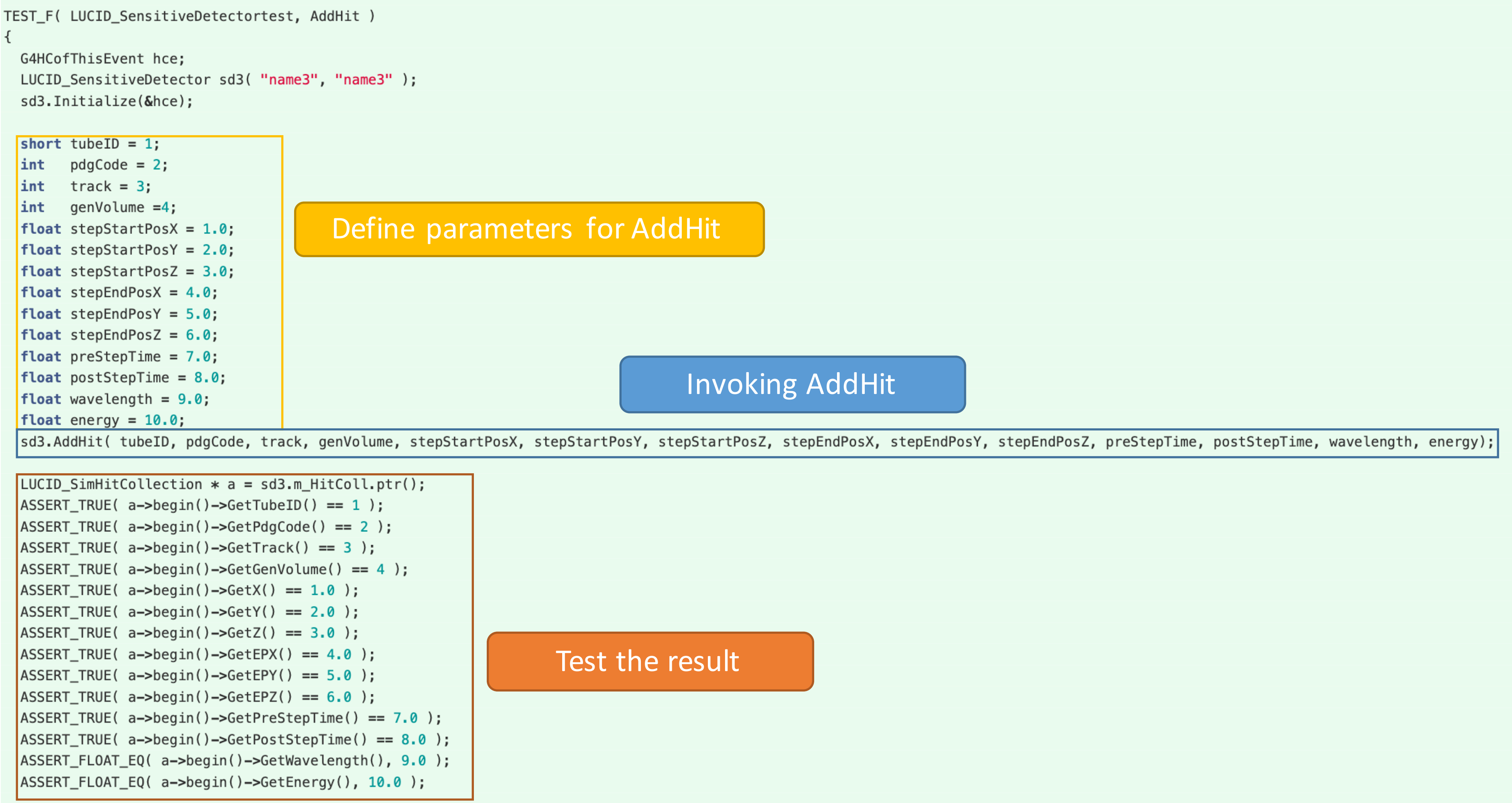
```
ApplicationMgr.ExtSvc += { "StoreGateSvc/DetectorStore", "StoreGateSvc/HistoryStore" };
AuditorSvc.Auditors  += { "AlgContextAuditor"};
```

Test the member function G4bool ProcessHits(G4Step*, G4TouchableHistory*)

- The key step is to provide a very complete G4Step object
  - ➢set a G4StepPoint object as preStepPoint
  - ➢set a G4StepPoint object as postStepPoint
  - ➢set a G4Track object
  - ➢set other variables like step length, total deposit energy, ……
- G4Step object setting code was written as a function
- after invoking the function, hit(s) has been stored. Then test if the hit(s) information is the same with what we expected

Test other member functions

- Read the member function
- Provide proper parameters for it
- Invoke it
- Test if the result is as we expected



```
TEST_F( LUCID_SensitiveDetectortest, AddHit )
{
  G4HCofThisEvent hce;
  LUCID_SensitiveDetector sd3( "name3", "name3" );
  sd3.Initialize(&hce);

  short tubeID = 1;
  int   pdgCode = 2;
  int   track = 3;
  int   genVolume =4;
  float stepStartPosX = 1.0;
  float stepStartPosY = 2.0;
  float stepStartPosZ = 3.0;
  float stepEndPosX = 4.0;
  float stepEndPosY = 5.0;
  float stepEndPosZ = 6.0;
  float preStepTime = 7.0;
  float postStepTime = 8.0;
  float wavelength = 9.0;
  float energy = 10.0;
  sd3.AddHit( tubeID, pdgCode, track, genVolume, stepStartPosX, stepStartPosY, stepStartPosZ, stepEndPosX, stepEndPosY, stepEndPosZ, preStepTime, postStepTime, wavelength, energy);

  LUCID_SimHitCollection * a = sd3.m_HitColl.ptr();
  ASSERT_TRUE( a->begin()->GetTubeID() == 1 );
  ASSERT_TRUE( a->begin()->GetPdgCode() == 2 );
  ASSERT_TRUE( a->begin()->GetTrack() == 3 );
  ASSERT_TRUE( a->begin()->GetGenVolume() == 4 );
  ASSERT_TRUE( a->begin()->GetX() == 1.0 );
  ASSERT_TRUE( a->begin()->GetY() == 2.0 );
  ASSERT_TRUE( a->begin()->GetZ() == 3.0 );
  ASSERT_TRUE( a->begin()->GetEPX() == 4.0 );
  ASSERT_TRUE( a->begin()->GetEPY() == 5.0 );
  ASSERT_TRUE( a->begin()->GetEPZ() == 6.0 );
  ASSERT_TRUE( a->begin()->GetPreStepTime() == 7.0 );
  ASSERT_TRUE( a->begin()->GetPostStepTime() == 8.0 );
  ASSERT_FLOAT_EQ( a->begin()->GetWavelength(), 9.0 );
  ASSERT_FLOAT_EQ( a->begin()->GetEnergy(), 10.0 );
```

Define parameters for AddHit

Invoking AddHit

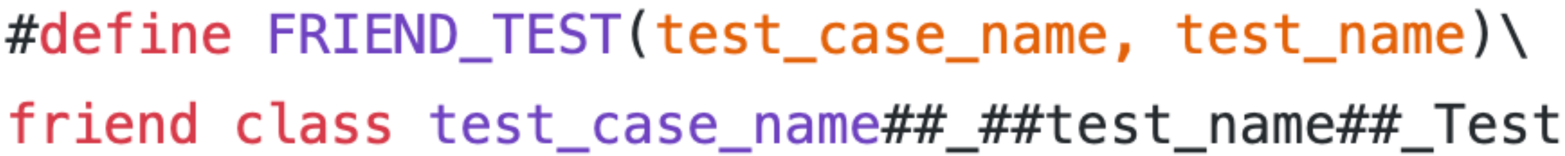
Test the result

The classes that I have written test code for

- AFP_SensitiveDetector
- AFP_SiDSensitiveDetector
- AFP_TDSensitiveDetector
- SctSensorSD
- SctSensor_CTB
- ZDC_StripSD
- ZDC_PixelSD
- TRTSensitiveDetector
- PixelSensorSD
- LUCID_SensitiveDetector
- ALFA_SensitiveDetector
- BCMSensorSD
- BLMSensorSD

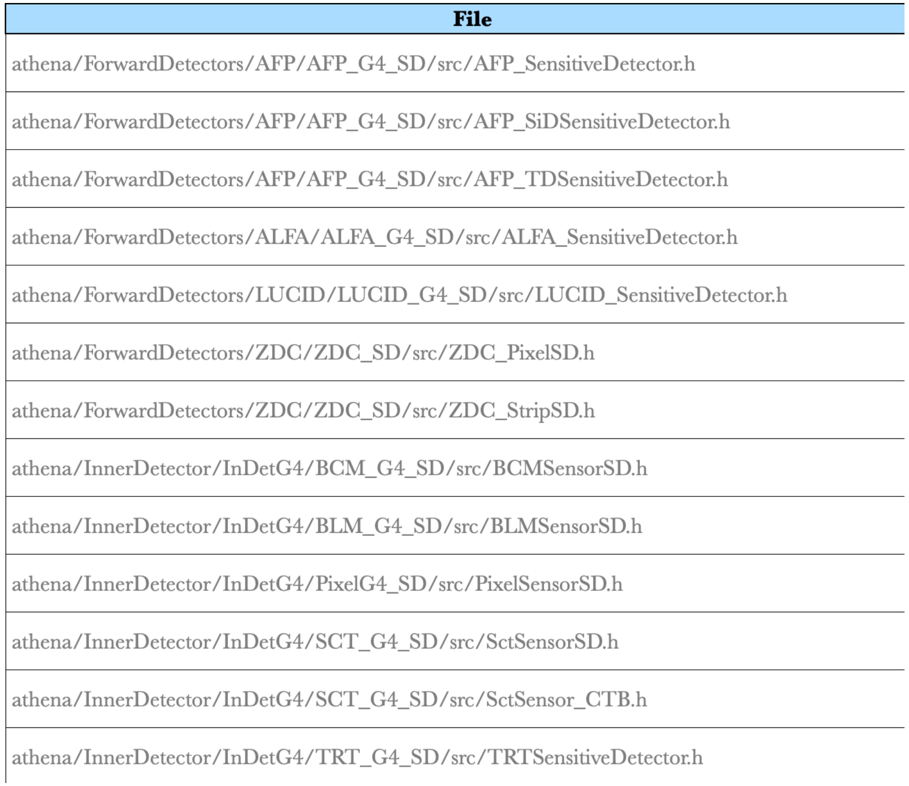
BACKUP

File gtest/gtest_prod.h



```
#define FRIEND_TEST(test_case_name, test_name)\
friend class test_case_name##_##test_name##_Test
```

The classes tested



File
athena/ForwardDetectors/AFP/AFP_G4_SD/src/AFP_SensitiveDetector.h
athena/ForwardDetectors/AFP/AFP_G4_SD/src/AFP_SiDSensitiveDetector.h
athena/ForwardDetectors/AFP/AFP_G4_SD/src/AFP_TDSensitiveDetector.h
athena/ForwardDetectors/ALFA/ALFA_G4_SD/src/ALFA_SensitiveDetector.h
athena/ForwardDetectors/LUCID/LUCID_G4_SD/src/LUCID_SensitiveDetector.h
athena/ForwardDetectors/ZDC/ZDC_SD/src/ZDC_PixelSD.h
athena/ForwardDetectors/ZDC/ZDC_SD/src/ZDC_StripSD.h
athena/InnerDetector/InDetG4/BCM_G4_SD/src/BCMSensorSD.h
athena/InnerDetector/InDetG4/BLM_G4_SD/src/BLMSensorSD.h
athena/InnerDetector/InDetG4/PixelG4_SD/src/PixelSensorSD.h
athena/InnerDetector/InDetG4/SCT_G4_SD/src/SctSensorSD.h
athena/InnerDetector/InDetG4/SCT_G4_SD/src/SctSensor_CTB.h
athena/InnerDetector/InDetG4/TRT_G4_SD/src/TRTSensitiveDetector.h

G4Step object setting function

G4MyPhysicalVolume.h

G4MyProcess.h