Integration of Opticks with GeANT4

Yunlong Li, Keith Evans, Adam Davis, Evelina Gersabeck, Marco Gersabeck

University of Manchester
yunlong.li@cern.ch

June 25, 2021
Introduction: Opticks

- An open source project developed by Simon Blyth
- Integrates the NVIDIA OptiX GPU ray tracing techniques with GEANT4 simulations.

- Shows ability to accelerate optical photon simulation \( \sim 1000 \) times speedup compared to GEANT4 simulation
- CaTS(https://github.com/hanswenzel/CaTS): Ongoing project by Fermilab, formerly G4OpticksTest, a calorimeter and tracker simulation framework using Opticks with GEANT4 for the creation and propagation of optical photons
Introduction

- **Target**: ensure advanced Geant4 example works for complex geometries as well as current capabilities
- We had a meeting with Simon, Hans and other contributors
- So far, Hans mainly focuses on LAr specific work
- To incorporate this further, we focus on RICH, a much more complex geometry

A simple test of LXe box, PMTs on the surfaces, an Al ball inside the box, 511 MeV electrons injected

Confirms the speedup and allows us to move forward
Environment settings

- Opticks depends on a lot of externals packages and libraries
- All kinds of problems may appear when installing Opticks on a GPU machine
- Using the LHC Computing Grid (LCG) environment makes the installation much easier

//Add these lines to the opticks config file
export LCG_ENV=/cvmfs/sft.cern.ch/lcg
source $LCG_ENV/views/LCG_100/x86_64-centos7-gcc10-opt/setup.sh
opticks-prepend-prefix $LCG_ENV/releases/LCG_100/clhep/2.4.4.0/x86_64-centos7-gcc9-opt/
opticks-prepend-prefix $LCG_ENV/releases/LCG_100/XercesC/3.2.3/x86_64-centos7-gcc9-opt/
opticks-prepend-prefix $LCG_ENV/releases/LCG_100/Boost/1.75.0/x86_64-centos7-gcc9-opt/

- Build Opticks with Geant4.10.07.p01
RICH1 with Opticks

- Test Opticks with The Ring Imaging Chevenkov (RICH) detectors in the LHCb detector
- RICH1 is located before magnet.
RICH1 with Opticks

- Exported the RICH1 gdml file from LHCb detector geometry
- Ran into some problems with the geometry translation
- Translation of materials: naming conflict in the abbreviation of material names
- Translation of optical surfaces: polished and ground optical surfaces
- Translation of solids: polycone conversion
- Translation of boundary libraries ...
- A 2 GeV electron injected with a small angle between beam pipe
Summary and outlook

- We are ramping up at Manchester and contributing to the integration of Opticks and Geant4

- Plans:
  - Unit tests for materials, surfaces, solids and so on, in order to show that they work and what would need to be done
  - Further propagation of LHCb RICH geometry
  - Contribute to hanswenzel/CaTS development
  - Minimization of host-device transfers, memory management/batching
Backup
Abbreviation of material names

```cpp
//sysrap/SAbbrev.cc
void SAbbrev::init() {
    ... // Previous code
    for(unsigned i=0 ; i < names.size() ; i++) {
        const char* name = names[i].c_str();
        SASCII* n = new SASCII(name);
        ... // Previous code
        std::string ab;
        if( n->upper > 0 && n->number > 0 ) { // 1 or more upper and number
            int iu = n->first_upper_index;
            int in = n->first_number_index;
            ab = n->getTwoChar( iu < in ? iu : in , iu < in ? in : iu );
        }
        else if( n->upper == 1 && n->number > 0 ) { // 1 upper and number
            int iu = n->first_upper_index;
            int in = n->first_number_index;
            ab = n->getTwoChar( iu < in ? iu : in , iu < in ? in : iu );
        }
        else if( n->upper >= 2 ) { // more than one uppercase : form abbrev from all the uppercase chars
            ab = n->getFirstUpper(n->upper);
        }
        else { // more than one uppercase
            ab = n->getFirst(2);
        }
        ... // Previous code
        assert( is_now_free && "failed to abbreviate" );
        abbrev.push_back(ab);
    }
}
```
const char* GOpticalSurface::Finish(unsigned finish)
{
    const char* s = NULL
switch(finish)
{
    case 0: s = polished ; break ;
    case 1: s = polishedfrontpainted ; break ;
    case 2: s = polishedbackpainted ; break ;
    case 3: s = ground ; break ;
    case 4: s = groundfrontpainted ; break ;
    case 5: s = groundbackpainted ; break ;
    default: assert(0 && "unexpected OpticalSurface finish"); break ;
}
return s ;
}

//ggeo/GOpticalSurface.cc
bool GOpticalSurface::isSpecular() const
{
    if(strncmp(m_finish,"0",strlen(m_finish))==0) return true ;
    if(strncmp(m_finish,"1",strlen(m_finish))==0) return true ;
    // used by JUNO.Mirror_opsurf m_finish 1
    if(strncmp(m_finish,"3",strlen(m_finish))==0) return false ;
    if(strncmp(m_finish,"4",strlen(m_finish))==0) return false ;
    LOG(info) << "GOpticalSurface::isSpecular""_
            " \_m_shortname""_
            " ( m_shortname ? m_shortname : "-" )
            " _m_finish""_
            " ( m_finish ? m_finish : "-" )
    ;
    assert(0 && "expecting m_finish to be 0:polished or 3:ground");
    return false ;
}
Backup

Polycone conversion

//extg4/X4Solid.cc
const bool X4Solid::
    convertPolycone_duplicate_py_inner_omission
    = true ;

void X4Solid::convertPolycone()
{
    ... const G4Polycone* const solid = static_cast<
        const G4Polycone*>(m_solid);
    assert(solid);
    const G4PolyconeHistorical* ph = solid->
        GetOriginalParameters() ;

    float startphi = ph->Start_angle/degree ;
    float deltaphi = ph->Opening_angle/degree ;
    assert( startphi == 0.f && deltaphi == 360.f
    );

    unsigned nz = ph->Num_z_planes ;
    ...
}

Polycons (PCon) are implemented in GEANT4 through the G4Polycone class:

In the picture:

phiStart = 1/4*Pi, phiTotal = 3/2*Pi, numZPlanes = 9,
rInner = { 0, 0, 0, 0, 0, 0, 0, 0, 0 },
rOuter = { 0, 10, 10, 5 , 5, 10 , 10 , 2, 2 },
z = { 5, 7, 9, 11, 25, 27, 29, 31, 35 }