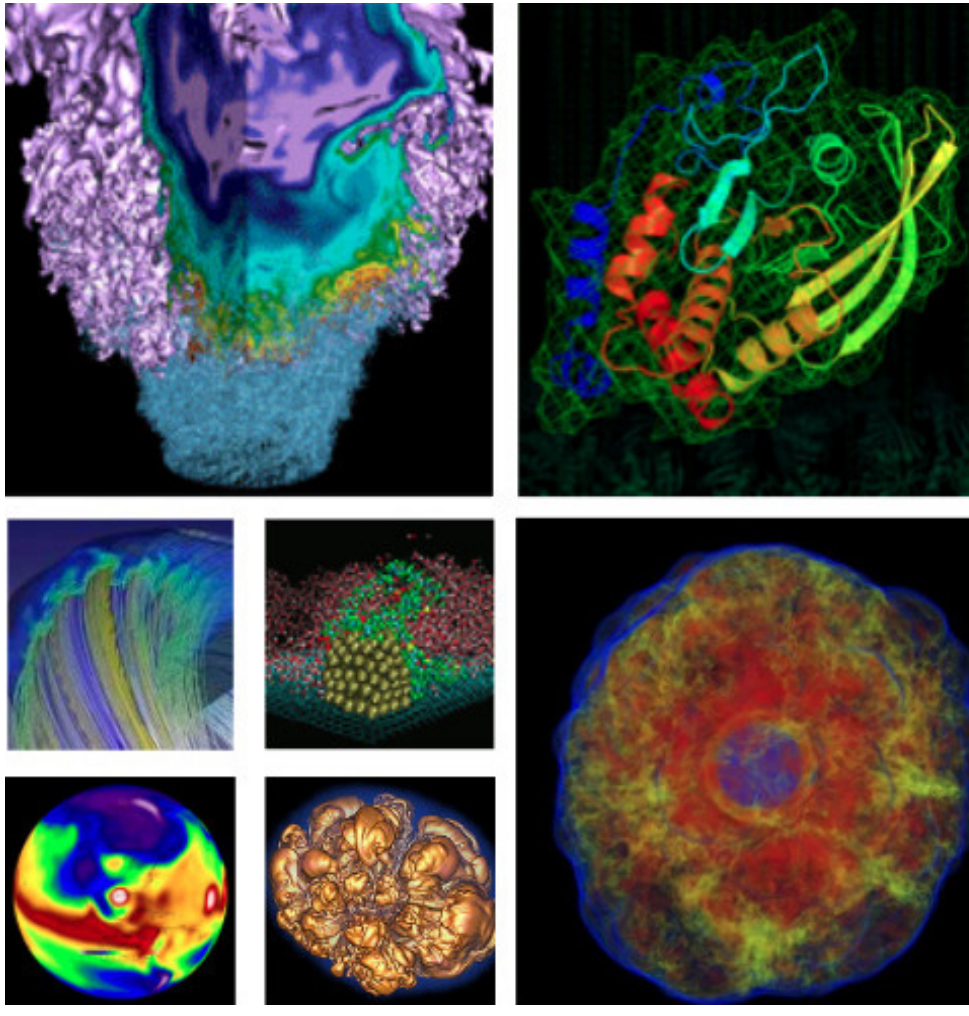
DICOM2 Extended Example

Using another example as library and demonstration of hits + statistics



National Energy Research Scientific Computing Center

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Jonathan R. Madsen

✉ jrmadsen@lbl.gov

National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory

August 30, 2018

- Build `DICOM/src` as library, build `DICOM.cc` as exe and link to "DICOM" library
- `include(CMakePackageConfigHelpers)`
  - `configure_package_config_file`
  - `write_basic_package_version_file`
- DicomUtilities.cmake – macro for building a library
  - Handles whether Geant4 built with shared or static libraries or both
  - Consistent tagging *i.e.*, `{_geant4_lib_use_suffix}`
- DICOMConfig.cmake
  - Standard type of installation file for `find_package(DICOM)`
  - Default installation path: same as Geant4 installation

- Currently just changes the primary generator action

```
int main(int argc, char** argv)
{
    // ... essentially same implementation as original DICOM example

    // except for User action initialization
    runManager->SetUserInitialization(new Dicom2ActionInitialization());
}

void Dicom2ActionInitialization::Build() const
{
    // from regular DICOM example
    SetUserAction(new DicomRunAction());

    // from regular DICOM example
    SetUserAction(new DicomEventAction());

    // from DICOM2 example
    Dicom2PrimaryGeneratorAction* pgAction = new Dicom2PrimaryGeneratorAction();
    SetUserAction(pgAction);
}
```

- Customize run implementation
  - `Dicom2RunAction`
  - `Dicom2Run`
- Demonstrate new scoring features for run accumulation scoring and different map/vector types
  - `G4THits{Map,UnorderedMap}<G4StatAnalysis>`
  - `G4THits{Vector,Deque}<G4double>`
  - `G4THits{Vector,Deque}<double, std::vector<double>>`
  - `G4THits{Vector,Deque}<G4StatAnalysis>`
  - `G4VTHits{Vector,Deque}<G4StatAnalysis, std::vector<G4StatAnalysis>>`

```
# voxel energy deposit

3375    128.74763 [sigma: 58.600106 | error: 0.64368613 | coeff: 0.64368613 |
                   eff: 1 | fom: 0.16090163 | hits: 2 )] keV

3377    2385.9604 [sigma: 1371.4691 | error: 0.99559666 | coeff: 0.99559666 |
                   eff: 1 | fom: 0.06725768 | hits: 3 )] keV

3378    68.157218 [sigma: 21.679894 | error: 0.55094206 | coeff: 0.55094206 |
                   eff: 1 | fom: 0.21963264 | hits: 3 )] keV

3379    2460.0649 [sigma: 1072.674 | error: 0.97500358 | coeff: 0.97500358 |
                   eff: 1 | fom: 0.070128786 | hits: 5 )] keV

# ...

5773    11291.75 [sigma: 265.31359 | error: 0.18649557 | coeff: 0.18649557 |
                  eff: 1 | fom: 1.7969789 | hits: 63 )] keV

5774    8466.0497 [sigma: 246.29098 | error: 0.18167702 | coeff: 0.18167702 |
                   eff: 1 | fom: 1.8935641 | hits: 39 )] keV

5775    11929.397 [sigma: 348.82999 | error: 0.17544725 | coeff: 0.17544725 |
                   eff: 1 | fom: 2.0304246 | hits: 36 )] keV
```

- MCNP always provides statistics by default
- Monte Carlo simulations are statistical simulations
- Many users, very reasonably, want statistics for their answers
- We don't feature statistics in our basic examples even though statistics are a *basic* part of Monte Carlo simulations
- Because we only feature statistics in extended and advanced examples, many users think they have to calculate their own statistics