Chemistry examples

*Geant4-DNA tutorial - G4 10.2 *beta*,*
August, 2015
Hiroshima
https://goo.gl/T7tl8L

Hoang Tran, Sylvain Meylan, Vaclav Stepan,
Yann Perrot,
Mathieu Karamitros, Sébastien Incerti

The examples

- **chem1**: Simple activation of the chemistry module
- **chem2**: Usage of TimeStepAction in the chemistry module
- **chem3**: Activate the full interactivity with the chemistry module

NB: Be aware that the released code is still a prototype

chem1

Initializing & starting the chemistry

chem1: classes

chem1.cc	Main file ⇒ creation of runManager; actionInitialization; detectorConstruction; physicsList
→ **ActionInitialization**	**Build()** ⇒ creation of primaryGeneratorAction & **stackingAction**
→ **DetectorConstruction**	Definition of the geometry
→ **PhysicsList**	Choice of the physics & chemistry lists ⇒ G4EmDNAPhysics & **G4EmDNAChemistry**
→ **PrimaryGeneratorAction**	Choice of the primary particle
→ **StackingAction**	In simple words, in Geant4, when no more “physical tracks” remain, the method **StackingAction::NewStage** is called

chem1: Activate chemistry

Minimum commands to start ...

File	Method/function	Commands
src/ PhysicsList.cc	**PhysicsList:: PhysicsList()**	//Load the default chemistry constructor **RegisterConstructor("G4EmDNAChemistry");**
src/ StackingAction.cc	**StackingAction:: NewStage()**	//Start handling of chemistry tracks **G4DNAChemistryManager::Instance()->Run();**

chem1: usage of macros

/scheduler/verbose 1
#up to 4

/scheduler/endTime 1 us
stop at 1 microsecond

Or alternatively
#**/scheduler/maxStepsNumber 10**
stop after 10 steps

for debugging
#**/scheduler/maxNullTimeSteps 10**
stop after 10 null time steps
#(computed time steps = 0)

All macro commands can be browsed through the Qt interface, just do:
$ **chem1 -gui**

Under virtual machines, in order to launch the Qt interface, you have to do:
$ **chem1 -gui -novis**

On the left hand side
⇒ macros with explanations

To obtain all possible command line options:
$ **chem1 -h**

chem2

Time stepping

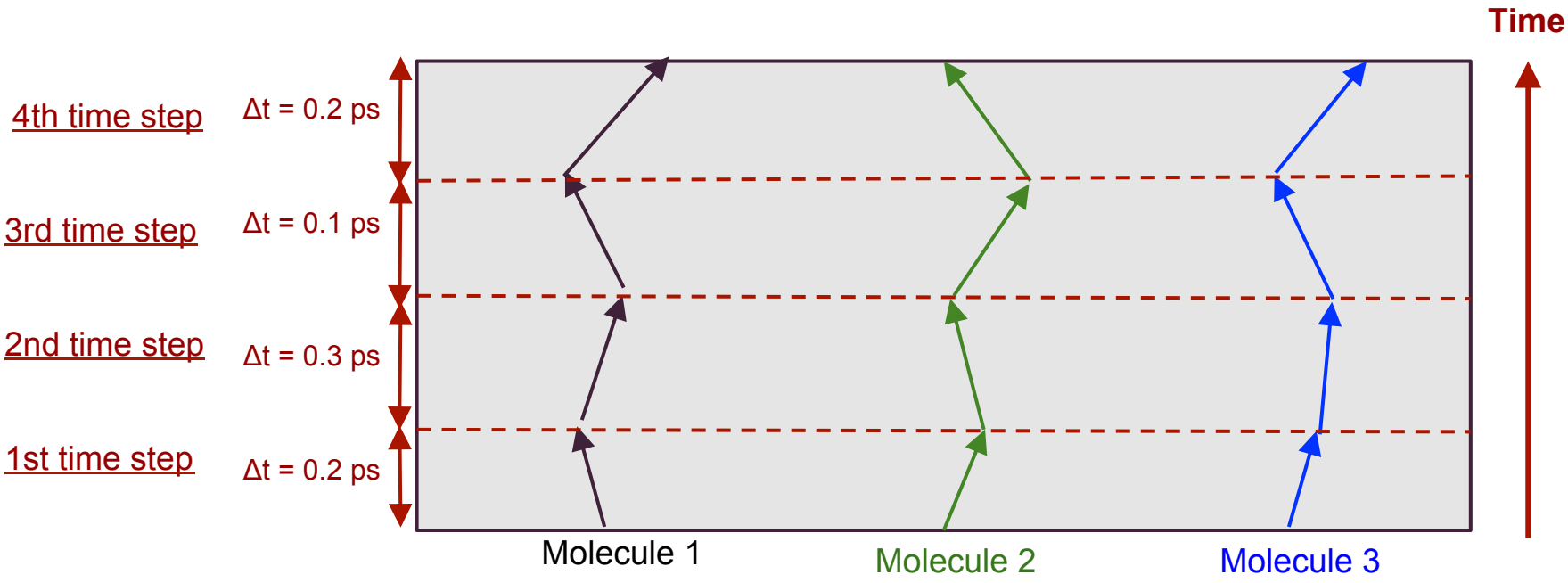
chem2: classes

chem2.cc	Main file ⇒ creation of runManager; actionInitialization; detectorConstruction; physicsList
→ **ActionInitialization**	**Include TimeStepAction**
→ **DetectorConstruction**	Definition of the geometry
→ **PhysicsList**	Choice of the physics & chemistry lists ⇒ G4EmDNAPhysics & **G4EmDNAChemistry**
→ **PrimaryGeneratorAction**	Choice of the primary particle
→ **StackingAction**	In simple words, in Geant4, when no more “physical tracks” remain, the method **StackingAction::NewStage** is called
→ **TimeStepAction**	**Allow the user to set minimal time step values and to retrieve informations from a given time step (molecule names, reaction products, etc…).**

chem2: TimeStep definition

Time step: **An amount of time when molecules will all move (and maybe react).**

Dynamically decided thanks to the implemented MolecularStepByStepModel.



chem2: Class methods

src/TimeStepAction.cc

In ActionInitilialisation: G4Scheduler::Instance()->SetUserAction(new TimeStepAction());

Methods

- **TimeStepAction():** Constructor of the TimeStepAction class. Inside it you can set the minimal time steps of your simulation.

AddTimeStep(1 * picosecond, 0.1 * picosecond);	*During the first simulated picosecond the minimal time step will be of 0.1 picosecond. If molecules are too close and can react before that time limit: brownian bridge.*
AddTimeStep(10 * picosecond, 1 * picosecond);	*From 1 ps to 10 ps in simulation time, the minimal time step will be of 1 ps.*

chem2: Class methods

src/TimeStepAction.cc

Methods

- **StartProcessing()**: Beginning of the chemistry simulation.
- **EndProcessing()**: End of the chemistry simulation.

- **UserPreTimeStepAction()**: If the user wants to do something before the start of the current time step.
- **UserPostTimeStepAction()**: If the user wants to do something after the end of the current time step. Called once after stepping all the tracks.
- **UserReactionAction(Reactif1, Reactif2, Products)**: will be called just after a reaction happened.

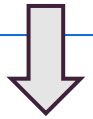
The output you see comes from the verbose setting in ActionInitialisation:
G4Scheduler::Instance()->SetVerbose(1);

chem2: Print the time spent during the last time step

src/TimeStepAction.cc

```
#include “G4Scheduler.hh”

void TimeStepAction::UserPostTimeStepAction()
{
  G4cout << G4Scheduler::Instance()->GetTimeStep() << G4endl;
  G4cout << G4ITTrackHolder::Instance()->GetNTracks() << G4endl;
}
```



G4Scheduler::Instance() gives access to some relevant methods:

- GetGlobalTime() → access to the current global time
- GetEndTime() → access to the final time of the simulation
- EndTracking() → stops the simulation
- GetMaxNbSteps() → if the user has set max nb of simulated steps

Other classes of interest

- **G4ITTrackHolder** → gives access to the tracks (e.g. GetNTracks, GetMainList(MOLECULE_ID) …)
- **G4MoleculeTable** → gives access to general molecule information (still in development)

chem2: Print the track ID of a solvated electrons

src/TimeStepAction.cc

```
#include “G4Scheduler.hh”
#include “G4MoleculeTable.hh”
#include “G4Molecule.hh”
#include “G4ITTrackHolder.hh”
#include “G4TrackList.hh”

void TimeStepAction::UserPostTimeStepAction()
{
  G4cout << “Current time step= ” << G4Scheduler::Instance()->GetTimeStep() << G4endl;

  int e_aq_id = G4MoleculeTable::Instance()->GetMoleculeModel(“e_aq”)->GetMoleculeID();
  G4TrackList* e_aq_list = G4ITTrackHolder::Instance()->GetMainList(e_aq_id);

  G4cout << “Nb of e_aq= ”<< e_aq_list->size() << G4endl;

  G4TrackList::iterator it = e_aq_list->begin();
  G4TrackList::iterator end = e_aq_list->end();

  for(;it!=end;++it)
  {
    G4cout << “trackID= ”<< (*it)->GetTrackID() << G4endl;
  }
}
```

chem3

User interactivity and visualization

Extensions to chem2

chem3.cc	Main file ⇒ creation of runManager; actionInitialization; detectorConstruction; physicsList
→ **ActionInitialization**	Extended to include ITTrackingInteractivity()
→ **DetectorConstruction**	World as semitransparent sphere now
→ **ITSteppingAction**	To retrieve information from a given “step” (several steps in one time step). **Same as SteppingAction for physics stage.**
→ **ITTrackingAction**	To retrieve information from a given track. **Same as TrackingAction for physics stage.**
→ **ITTrackingInteractivity**	**Mandatory to visualize chemical tracks and use stepping/tracking actions**

chem3: Class methods

src/ITSteppingAction.cc

src/ITTrackingAction.cc

In ActionInitilialisation:

```
ITTrackingInteractivity* itInteractivity = new ITTrackingInteractivity();
itInteractivity->SetUserAction(new ITSteppingAction);
itInteractivity->SetUserAction(new ITTrackingAction);
G4Scheduler::Instance()->SetInteractivity(itInteractivity);
```

- **To use the standard UserSteppingAction and UserTrackingAction, one has to declare it through ITTrackingInteractivity and then send the ITTrackingInteractivity to G4Scheduler**

Note this useful method: **GetMolecule(track)->GetName()**
It can be used in pre or post user actions.

chem3: Class methods

src/ITTrackingInteractivity.cc

Methods

The methods here should not be changed by the user.

Executing the example

```
$ ./chem3 -gui tcsh
```

```
# Send one primary particle in
/control/execute beam.in
```

```
# Let's try a longer track
/gun/particle proton
/gun/energy 10 MeV
/run/beamOn 1
```

Note: Complain from the navigator may appear (solvated electrons exit the volume where the original electron was located). This is fixed for release 10.2
(Note to Geant4 developers: the fix is not yet on the trunk)

Executing the example: GUI

Note: Be aware that under virtual machines, Qt and OpenGL may cause troubles. To solve this issue follow this instructions

Replace the vis.mac in chem3 by this one

```
./chem3 -gui -novis
```

In the GUI:

```
/vis/open OGL
/control/execute vis.mac
/control/execute beam.in
```

Thank you :)

Exercice

Print the number of H_3O^+ & °OH after every 10 time steps

Hints

The additional slides might be useful to know how the molecules are referenced

What's needed:

G4MoleculeTable → Get a molecule model

G4Molecule → Get a molecule ID

G4ITTrackHolder → Use a molecule ID to retrieve a G4TrackList through the method GetMainList(MOLECULE_ID)

G4TrackList → Get the size (= number of tracks)

Exercice - Solution

Print the number of H3O+ & °OH after every 10 time steps

Solution

```
#include “G4Scheduler.hh”
#include “G4MoleculeTable.hh”
#include “G4Molecule.hh”
#include “G4ITTrackHolder.hh”
#include “G4TrackList.hh”

void TimeStepAction::UserPostTimeStepAction()
{
  if(G4Scheduler::Instance()->GetNbSteps() % 10 != 0) return;

  int H3O_id = G4MoleculeTable::Instance()->GetMoleculeModel(“H3Op”)->GetMoleculeID();
  int OH_id = G4MoleculeTable::Instance()->GetMoleculeModel(“OH”)->GetMoleculeID();
  G4TrackList* H3O_list = G4ITTrackHolder::Instance()->GetMainList(H3O_id);
  G4TrackList* OH_list = G4ITTrackHolder::Instance()->GetMainList(OH_id);

  G4cout << “Nb of H3O= ” << H3O_list->size() << G4endl;
  G4cout << “Nb of OH= ” << OH_list->size() << G4endl;
}
```

Additional slides
Define species & reactions

Valid for G4 10.2 Beta - NOT valid for G4 10.2

G4EmDNAChemistry

```
#include "G4VPhysicsConstructor.hh"
#include "G4VUserChemistryList.hh"
#include "globals.hh"
class G4DNAMolecularReactionTable;
class G4EmDNAChemistry : public G4VUserChemistryList
{
public :
        G4EmDNAChemistry();
        virtual ~G4EmDNAChemistry();
        virtual void ConstructMolecule();
        virtual void ConstructProcess();
        virtual void ConstructDissociationChannels();
        virtual void ConstructReactionTable(G4DNAMolecularReactionTable* reactionTable);
        virtual void ConstructTimeStepModel(G4DNAMolecularReactionTable*
reactionTable);
};
```

```
void G4EmDNAChemistry::ConstructMolecule() {
  //----------------------------------
  // Create the definition
  G4H2O::Definition();
  G4Hydrogen::Definition();
  G4H3O::Definition();
  G4OH::Definition();
  G4Electron_aq::Definition();
  G4H2O2::Definition();
  G4H2::Definition();
  //----------------------------------
 // Define molecule model
  G4MoleculeTable::Instance()->CreateMoleculeModel("H3Op", G4H3O::Definition());
  G4Molecule* OHm = G4MoleculeTable::Instance()->          CreateMoleculeModel(
                                        "OHm", // just a tag to store and retrieve from G4MoleculeTable
                                                                    G4OH::Definition(),
                                      1, 5.0e-9 * (m2 / s)); // define new diffusion coefficient
  OHm->SetMass(17.0079 * g / Avogadro * c_squared);  // Set a new mass
  G4MoleculeTable::Instance()->CreateMoleculeModel("OH", G4OH::Definition());
  G4MoleculeTable::Instance()->CreateMoleculeModel("e_aq",
                                G4Electron_aq::Definition());
  G4MoleculeTable::Instance()->CreateMoleculeModel("H",
                                G4Hydrogen::Definition());
  G4MoleculeTable::Instance()->CreateMoleculeModel("H2", G4H2::Definition());
}
```

Useful commands

Create a new molecule

G4MoleculeDefintion* myMoleculeDefinition = **G4MoleculeTable::Instance**()->**CreateMoleculeDefinition**(
"NewMolDef", // name of the new molecule
5.0e-9 * (m2 / s)); // default diffusion coefficient
// WARNING: the definition of diffusion coefficient will be moved to processes

Create a "molecule model"

G4MoleculeTable::Instance()->CreateMoleculeModel(
"myMoleculeModel", // name of the new molecule
myMoleculeDefinition,
-1, // FACULTATIVE: create a model with a given charge
3.0e-8 * (m2 / s));
// FACULTATIVE: define a diffusion coefficient for the molecule of the selected charge
// WARNING: in the future, the definition of diffusion coefficient will be moved to processes and will handle multiple materials

Add reactions

```
void G4EmDNAChemistry::ConstructReactionTable(G4DNAMolecularReactionTable*
                        theReactionTable)
{
 G4Molecule* OHm = G4MoleculeTable::Instance()->GetMoleculeModel("OHm");
 G4Molecule* OH = G4MoleculeTable::Instance()->GetMoleculeModel("OH");
 G4Molecule* e_aq = G4MoleculeTable::Instance()->GetMoleculeModel("e_aq");
 G4Molecule* H2 = G4MoleculeTable::Instance()->GetMoleculeModel("H2");

 //------------------------------------------------------------
 // e_aq + e_aq + 2H2O -> H2 + 2OH-
 G4DNAMolecularReactionData* reactionData = new G4DNAMolecularReactionData(
              0.5e10 * (1e-3 * m3 / (mole * s)),  // reaction rate constant
       e_aq,   // reactant 1
        e_aq); // reactant 2
 reactionData->AddProduct(OHm);  // product 1
 reactionData->AddProduct(OHm); // product 2
 reactionData->AddProduct(H2);    // product 3
 theReactionTable->SetReaction(reactionData);  // add to the reaction table

...
```

Additional slides chem3

Visualization

Using OpenGL for track visualization

```
#vis.mac: Enabling the time slices, see Drawing by time
/vis/modeling/trajectories/drawByParticleID-0/default/setTimeSliceInterval 0.1 ns

# Manually setting the start and end time of tracks to show
/vis/ogl/set/startTime 0 s
/vis/ogl/set/endTime 1 ps

# Filtering the track works as on normal tracks /vis/filtering/trajectories/create/
particleFilter
/vis/filtering/trajectories/particleFilter-0/add e-
/vis/filtering/trajectories/particleFilter-0/add e_aq
```

Please refer to excellent guides by Joseph Perl for more information.

vis.mac

```
# To show many tracks, but consume more memory
/vis/ogl/set/displayListLimit 10000000

# To draw chemical molecule trajectories
/vis/modeling/trajectories/create/drawByParticleID
/vis/modeling/trajectories/drawByParticleID-0/default/setDrawStepPts false
/vis/modeling/trajectories/drawByParticleID-0/default/setStepPtsSize 1

# ...in brave colors
/vis/modeling/trajectories/drawByParticleID-0/set OH magenta
/vis/modeling/trajectories/drawByParticleID-0/set H3O yellow
/vis/modeling/trajectories/drawByParticleID-0/set e_aq blue
/vis/modeling/trajectories/drawByParticleID-0/set H2O2 green
/vis/modeling/trajectories/drawByParticleID-0/set H white
```

Please refer to excellent guides by Joseph Perl for more information.

Scripting time steps visualization and camera movements using loops

physics.mac

```
/control/alias endTime 0.01

/vis/modeling/trajectories/drawByParticleID-0/default/setDrawStepPts true

/vis/ogl/set/fade 0

/vis/ogl/set/startTime 0 ps
/control/divide timeStep {endTime} 100.
/control/loop physics.loop startTime 0.0 {endTime} {timeStep}
```

physics.loop:

```
/vis/ogl/set/endTime {startTime} ps
```

Please refer to excellent guides by Joseph Perl for more information.