Making a Mutli-Group Monte Carlo Program for C5G7MOX Benchmark Analysis

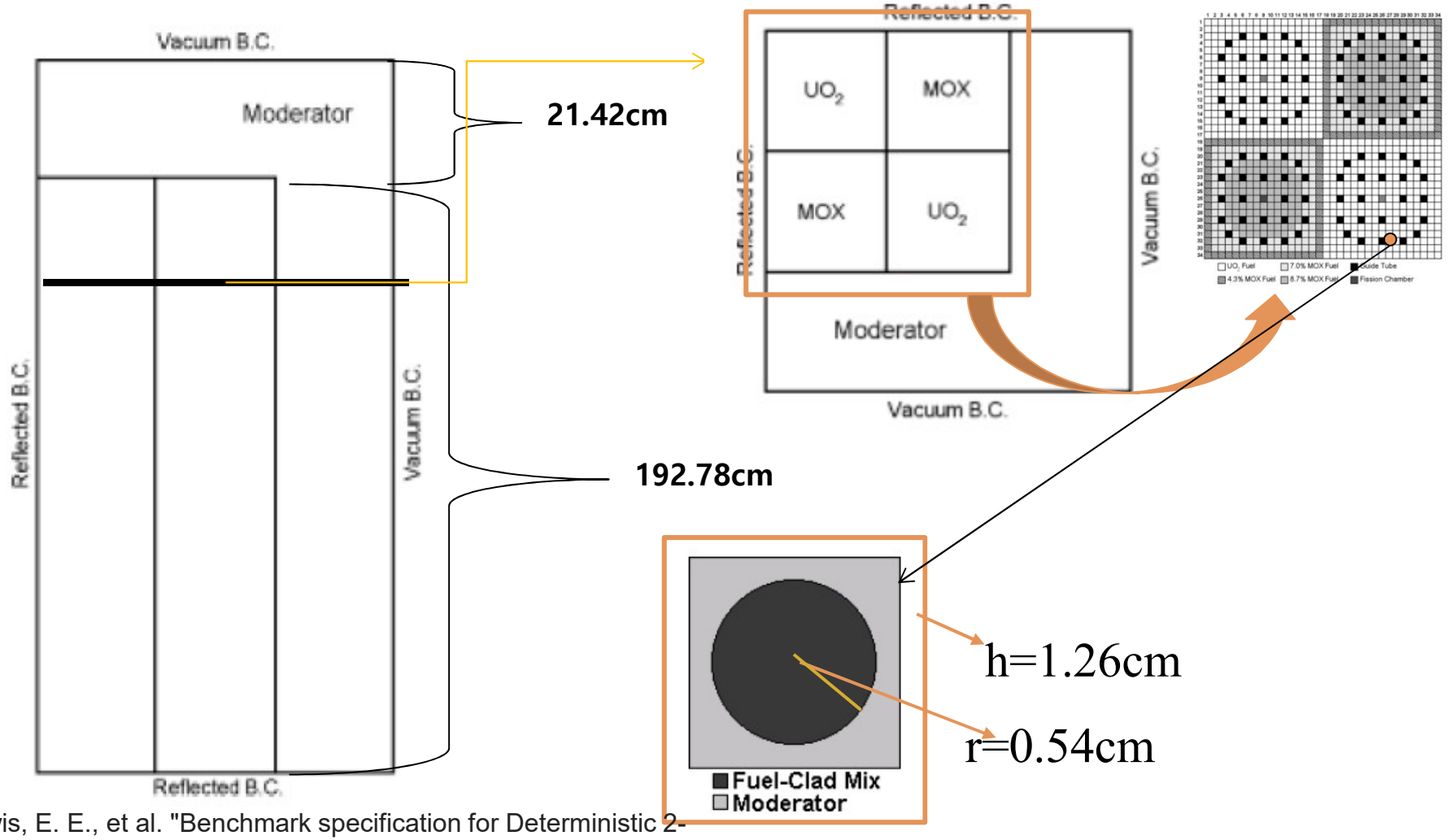
Shim, Hyung Jin

Nuclear Engineering Department,
Seoul National University

Requirements

- Calculate MC estimates of the following neutronics parameters for the C5G7MOX benchmark.
  - Particle balance information
  - Region-wise average flux and reaction rates from neutron sources of the first group at the center position
  - Criticality: k_{eff}
  - Region-wise average flux and reaction rates for the eigenvalue calculations
  - Kinetics parameters

C5G7MOX - Geometry



Lewis, E. E., et al. "Benchmark specification for Deterministic 2-D/3-D MOX fuel assembly transport calculations without spatial homogenization (C5G7 MOX)." *NEA/NSC* 280 (2001).

Macroscopic XS's

Seven group, transport corrected, isotropic scattering cross-sections

Table 1a. UO_2 Fuel-Clad Macroscopic Cross-sections

	Total Cross-section	Transport Cross-section	Absorption Cross-section	Capture Cross-section	Fission Cross-section	Nu	Chi
Group 1	2.12450E-01	1.77949E-01	8.02480E-03	8.12740E-04	7.21206E-03	2.78145E+00	5.87910E-01
Group 2	3.55470E-01	3.29805E-01	3.71740E-03	2.89810E-03	8.19301E-04	2.47443E+00	4.11760E-01
Group 3	4.85540E-01	4.80388E-01	2.67690E-02	2.03158E-02	6.45320E-03	2.43383E+00	3.39060E-04
Group 4	5.59400E-01	5.54367E-01	9.62360E-02	7.76712E-02	1.85648E-02	2.43380E+00	1.17610E-07
Group 5	3.18030E-01	3.11801E-01	3.00200E-02	1.22116E-02	1.78084E-02	2.43380E+00	0.00000E+00
Group 6	4.01460E-01	3.95168E-01	1.11260E-01	2.82252E-02	8.30348E-02	2.43380E+00	0.00000E+00
Group 7	5.70610E-01	5.64406E-01	2.82780E-01	6.67760E-02	2.16004E-01	2.43380E+00	0.00000E+00

Scattering Block

	to Group 1	to Group 2	to Group 3	to Group 4	to Group 5	to Group 6	to Group 7
Group 1	1.27537E-01	4.23780E-02	9.43740E-06	5.51630E-09	0.00000E+00	0.00000E+00	0.00000E+00
Group 2	0.00000E+00	3.24456E-01	1.63140E-03	3.14270E-09	0.00000E+00	0.00000E+00	0.00000E+00
Group 3	0.00000E+00	0.00000E+00	4.50940E-01	2.67920E-03	0.00000E+00	0.00000E+00	0.00000E+00
Group 4	0.00000E+00	0.00000E+00	0.00000E+00	4.52565E-01	5.56640E-03	0.00000E+00	0.00000E+00
Group 5	0.00000E+00	0.00000E+00	0.00000E+00	1.25250E-04	2.71401E-01	1.02550E-02	1.00210E-08
Group 6	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	1.29680E-03	2.65802E-01	1.68090E-02
Group 7	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	8.54580E-03	2.73080E-01

Lewis, E. E., et al. "Benchmark specification for Deterministic 2-D/3-D MOX fuel assembly transport calculations without spatial homogenization (C5G7 MOX)." *NEA/NSC* 280 (2001).

Design of Input File Structure

```
Options (
    Mode FixedSource
    SrcN 1000
    //Center position
     InitialSource    PNT    115
    ImplicitCapture 1
);

Geometry (
    NCell 15
    Size 40
         10 10 10 10 10
         15 20 15
         10 10 10 10 10
         40
);
```

```
CXLibrary (
    //outer ref.
    CXTable 0 (
        SigCap  1.16801E-04
        SigN2N  8.57765E-07
        SigN3N  1.34304E-09
        SigFis  0.00000E+00
        nuSigF  0.00000E+00
        SigSca  2.65238E-01
    );

    //fuel #5
    CXTable 1 (
        SigCap  1.12041E-03
        SigN2N  3.50227E-06
        SigN3N  4.96426E-09
        SigFis  7.92237E-04
        nuSigF  2.32702E-03
        SigSca  2.95714E-01
    );
    …
);
```

서울대학교 노심핵설계실

Output for an Fixed-Source Calculation

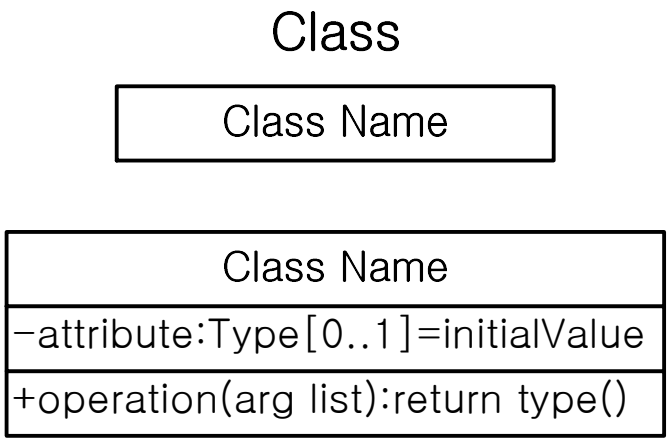
```
      *** Neutron Creation ***       |        *** Neutron Loss ***
    Source       :    1.00000e+000   |
    Fission Gain:     3.25535e+001   |   Fission Loss:     1.10786e+001
    (n,Xn)  Gain:     9.77826e-002   |   (n,Xn)  Loss:     4.88913e-002
                                     |   Capture Loss:     1.70707e+001
                                     |   Leakage Loss:     5.45311e+000
    ------------------------------------------------------------------
    Gain Sum     =    3.36513e+001   |   Loss Sum     =    3.36513e+001
    Multi Factor =    2.25238e+001(0.12172)


 *** Flux & Fission Rate Estimation ***
CELL #1 :
    Estimated Flux          = 4.84268e+001    Standard Deviation = 6.23332e+000
    Estimated Fission Rate  = 0.00000e+000    Standard Deviation = 0.00000e+000

CELL #2 :
    Estimated Flux          = 1.03650e+002    Standard Deviation = 1.34203e+001
    Estimated Fission Rate  = 8.21157e-002    Standard Deviation = 1.06320e-002

...
```

서울대학교 노심핵설계실

S/W Design – Methodology: UML

- What is UML?
  - The Unified Modeling Language (UML) is a family of graphical notations that help in describing and designing software systems, particularly software systems built using the object-oriented (OO) style.
  - Graphical modeling language have been around in the software industry for a long time. The fundamental driver behind them all is that programming language are not high enough level of abstraction to facilitate discussions about design.
  - The UML is a relatively open standard, controlled by the Object Management Group (OMG), an open consortium companies.
  - The UML was born out of the unification of the many object-oriented graphical modeling languages that thrived in the late 1980s and early 1990s.

UML – Attribute



- A class diagram is divided into three compartments:
  - the name of the class,
  - its attributes,
  - and its operations.

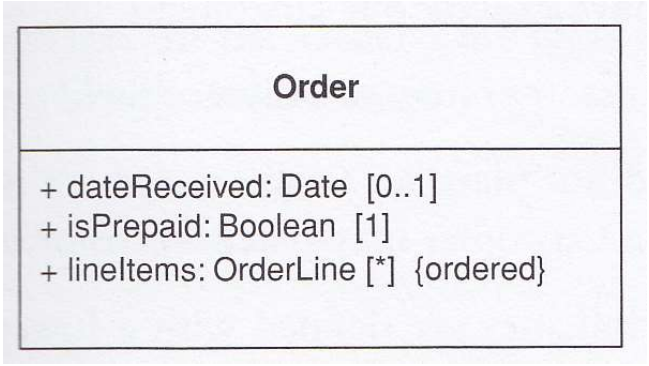
- Properties represent structural features of a class.
- Properties may appear in two quite distinct notations: attributes and associations.
- The full form of an attribute is:

  Visibility name: type multiplicity = default {property-string}

  ex) - name: String [1] = “Untitled” {readOnly}
  - The visibility marker indicates whether the attribute is public (+) or private (-).

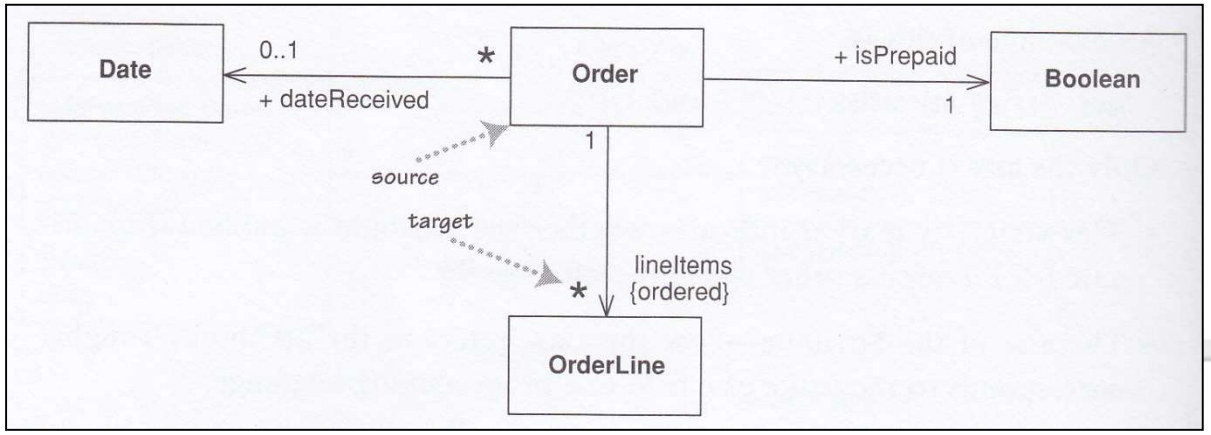
UML - Associations

- Much of the same information that you can show on an attribute appears on an association.
- The below figures show the same properties represented in the two different notations.



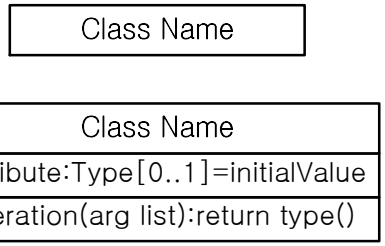
< Showing properties of an order as attributes >

< Showing properties of an order as associations >

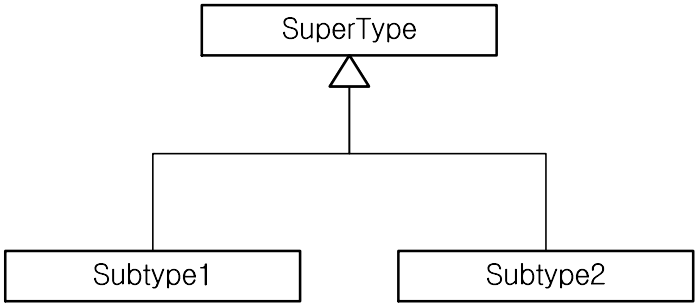


UML Rules

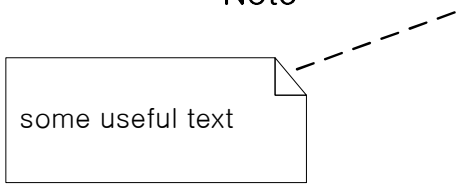
Class



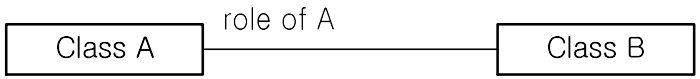
Generalization



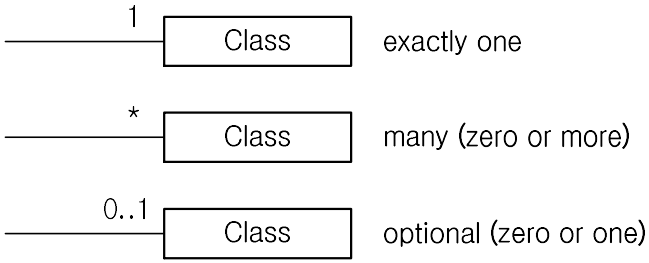
Note

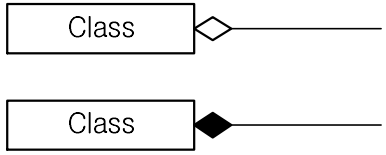


Association



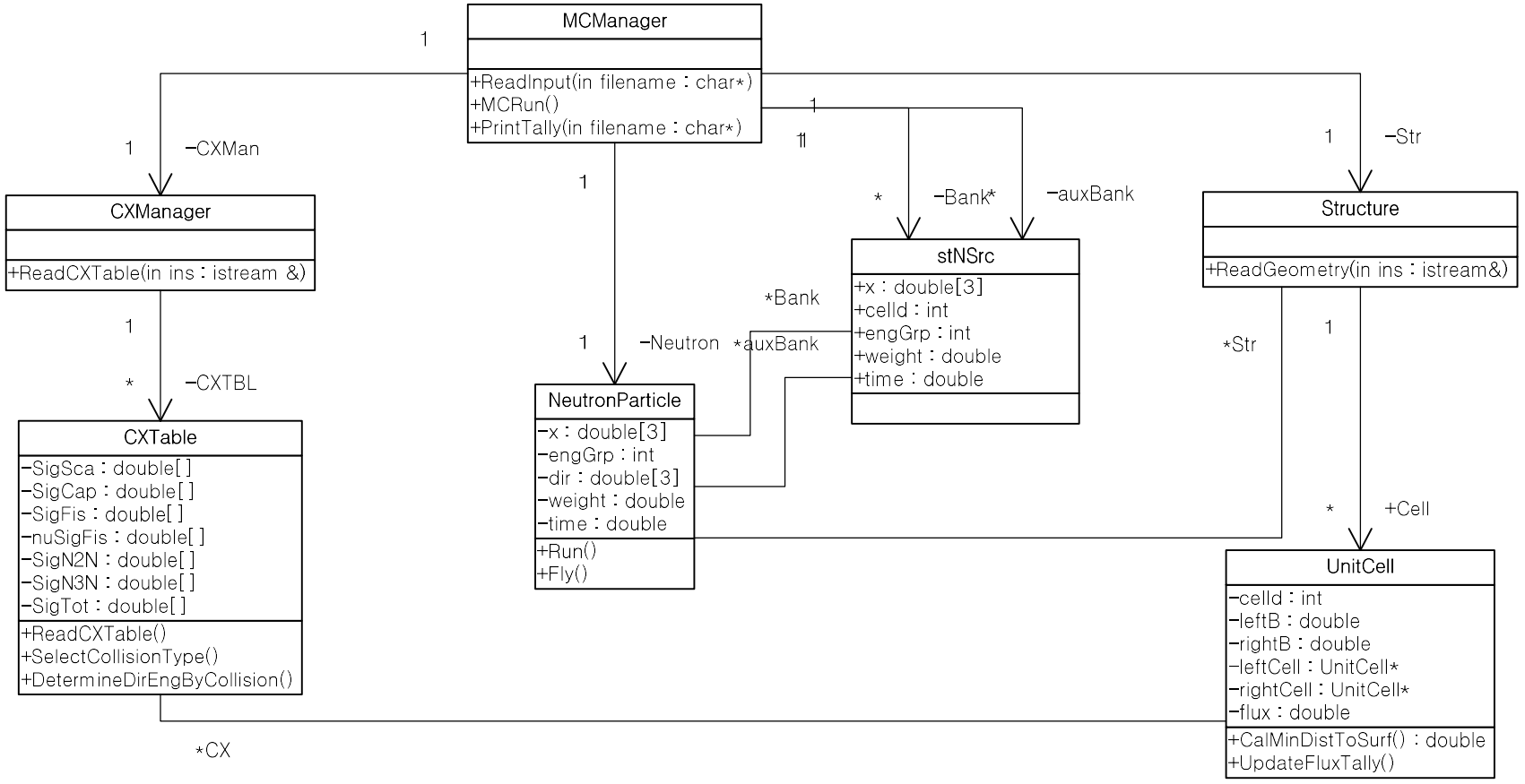
Multiplicities





Navigability

S/W Design – Class Diagram



Main Function

```
#include "mcman.h"

int main(void)
{
    MCManager MCMan; //MC Manager object

    //Read Input
    MCMan.ReadInput("input.txt"); // Input file name = "input.txt"

    //Execute MC simulation
    MCMan.MCRun();

    //Print Output
    MCMan.PrintTally("output.txt"); // Output file name = "output.txt"

    return 0;
}
```

ReadInput

```
//Read input file
void MCManager::ReadInput(char *filename)
{
    int i;
    char card[CARDLEN];

    ifstream inFp(filename); //Make input file stream

    //Read input
    while(inFp.peek()!=EOF)
    {
        inFp>>card;

        //Read "Options" section
        if(!strcmp(card, "Options")) this->ReadOptions(inFp);
        //Read "CXLibrary" section
        else if(!strcmp(card, "CXLibrary")) CXMan.ReadCXTables(inFp);
        //Read "Geometry" section
        else if(!strcmp(card, "Geometry")) Str.ReadGeometry(inFp);
    }

    //Allocate memories
    for(i=0; i<NUM_CELL; i++) Str.Cell[i].AllocateMemory(totCycNum,NUM_CELL);
}
```

MCRun

```
//Simulate neutron by MC
void MCManager::MCRun(void)
{
    int i;

    cout.precision(5);
    cout.setf(ios::fixed, ios::floatfield);

    //Link objects
    Neutron.LinkStructure(&Str);
    Neutron.LinkBank(&Bank);
    Neutron.LinkAuxBank(&auxBank);
    for(i=0; i<NUM_CELL; i++) {
        Str.GetCell(i)->CX = CXMan.GetCXTable(i);
    }

    if(mode==MODE_FIXED) MCRunInFixedMode();
    else                 MCRunInEigenMode();
}
```

MCRunInFixedMode

```
//MC Run by fixed-source calculation
void MCManager::MCRunInFixedMode(void)
{
    int i;
    stNSrc nData; //data for a source neutron
    stNSrc sData;
    int percent = 0;

    //Run as much as 'nSrc'
    for(i=0; i<nSrc; i++) {
        //Generate a source
        GenerateSource(nData);
        //Simulate neutron
        Neutron.Run(mode, true, bImpCap, nData.weight, nData);
        while(auxBank.size()>0) {
            //Read source data from source bank
            sData = auxBank.front(); auxBank.pop();
            Neutron.Run(mode, true, bImpCap, sData.weight, sData);
        }
        Str.UpdateTallySquare();

        //Print execution percentage
        if( 100.0*i/nSrc > percent+1.0 ) {
            percent = int(100.0*i/nSrc);
            cout<<percent<<" % done"<<endl;
        }
    }
}
```

Neutron.Run (1/4)

```
//Run particle simulation
void NeutronParticle::Run(int mode, bool bTally, bool bImpCap,
                          double wgt, stNSrc nData, bool bSecond)
{
    int i;
    int state;
    int colType;
    stNSrc sData;
    double preWeight;

    //Set current data
    srcCell = nData.celId;
    Cell=Str->GetCell(nData.celId); //Current Cell

    x[0] = nData.x; x[1] = nData.y; x[2] = nData.z; //location
    engGrp = nData.engGrp; //energy
    SamplingMethod::SampleIsotropicDir(dir); //direction
    weight = wgt; //weight
    time = nData.time;

…
```

Neutron.Run (2/4)

```
do {
      //Simulate neutron flight
      state = Fly( bTally );

      //Collide with a nuclide
      if( state == IN_CELL )
      {
          preWeight = weight;

          //Select collision type
          colType = Cell->CX->SelectCollisionTypeIncludingFission( bImpCap, engGrp, weight );

          //If collision type is 'absorption', kill neutron.
          if(colType == REA_CAP) state = THE_END;
          //If collision type is 'removal', sample new energy and direction
          else if(colType == REA_SCA) Cell->CX->DetermineDirEngByCollision(engGrp, dir);
          …
```

Neutron.Run (3/4)

```
else if(colType == REA_FIS) {
     int srcParticleNum;

     //Sample source particle number
     srcParticleNum = (int)( Cell->CX->GetNu(engGrp) + RNG.GetRN() );

     for(i=0; i<srcParticleNum-1; i++) {
          //Set source data
          sData.x = x[0];
          sData.y = x[1];
          sData.z = x[2];
          sData.weight = weight;
          sData.engGrp = 0; // Energy group number of fission sources are assumed to be 0.
          sData.celId = Cell->GetCellId();
          sData.time = time;

          //Store fission source at bank
          auxBank->push(sData);
     }
     Cell->CX->DetermineDirEngByCollision(engGrp, dir);
 }
```

Neutron.Run (4/4)

```
            //If collision type is '(n,2n)', keep neutron
            else if(colType == REA_N2N) {
            …
            }
            //If collision type is '(n,3n)', keep neutron
            else if(colType == REA_N3N) {
            …
            }
        }
    }while(state!=THE_END);
```