How to make an application code using G4MT

Xin Dong and Gene Cooperma
High Performance Computing Lab
College of Computer and Information Science
Northeastern University
Boston, Massachusetts 02115
USA
{gene,xindong}@ccs.neu.edu

Geant4MT Tools for Implementation Support

- Transformation for Thread Safety (TTS)
  1. make each global or static variable thread-local
  2. independent threads lead to absolute thread-safety: any thread can call any function. No data race!
- Transformation for Memory Reduction (TMR)
  1. *relatively read-only data*: written to during its initialization and read-only during the computation of each event.
  2. share relatively read-only data, and replicate other data
- Debugging Tools
  1. compare the original program with the multi-threaded version
  2. runtime correctness: to serialize updates to shared data
- Malloc Non-standard Extension using a Thread-Private Heap (TPMalloc)
- Avoidance of Cache Coherence Bottlenecks

N02 Parallelization: Change List I

N02	ParN02	Comments	Action
exampleN02.cc	ParN02.cc	Parallel Application Main	TMR
	ParTopC.icc	Common Frame	Copy
	mymalloc.h	TPMalloc	Copy
	hjmalloc.c	TPMalloc	Copy
	mymalloc.c	TPMalloc	Copy
	tpmallocstub.h	TPMalloc	Copy
	tpmallocstub.c	TPMalloc	Copy
GNUmakefile		Link to TPMalloc	Copy + Change

N02 Parallelization: Change List II

N02/include	ParN02/include	Comments	Action
ExN02ChamberParameterisation.hh			
ExN02DetectorConstruction.hh			TMR
ExN02DetectorMessenger.hh			
ExN02EventAction.hh			
ExN02MagneticField.hh			
ExN02PhysicsList.hh			
ExN02PrimaryGeneratorAction.hh			
ExN02RunAction.hh			
ExN02SteppingAction.hh			
ExN02SteppingVerbose.hh			
ExN02TrackerHit.hh		global	TTS
ExN02TrackerSD.hh			
	G4MycoutDestination.icc	TPoutput	Copy
	GetTid.hh	TPoutput	Copy
	GetTid.icc	TPoutput	Copy
	ParRunManager.hh	Common Frame	Copy

N02 Parallelization: Change List III

N02/src	ParN02/src	Comments	Action
ExN02ChamberParameterisation.cc			
ExN02DetectorConstruction.cc			TMR
ExN02DetectorMessenger.cc			
ExN02EventAction.cc			
ExN02MagneticField.cc			
ExN02PhysicsList.cc			
ExN02PrimaryGeneratorAction.cc			
ExN02RunAction.cc			
ExN02SteppingAction.cc			
ExN02SteppingVerbose.cc			
ExN02TrackerHit.cc		global	TTS
ExN02TrackerSD.cc		static	TTS
	ParRunManager.cc	Common Frame	Copy + Change

Thread Local Storage (TLS): An Example

```
#include <stdio.h>
#include <pthread.h>
__thread int gvar = 0; //int gvar = 0;
void *increase(void *)
{   gvar++;
    printf(”Value in child thread: %d\n”, gvar);
}
int main(int argc, char* argv[])
{  pthread_t tid;
    printf(”Value in main thread: %d\n”, gvar);
    pthread_create( &tid, NULL, increase, NULL );
    pthread_join(tid, NULL);
    printf(”Value in main thread: %d\n”, gvar);
    return 0;
}
```

```
Value in main thread:  0
Value in child thread:  1
Value in main thread:  0
```

ExN02TrackerHit.hh Before TTS

```
extern G4Allocator<ExN02TrackerHit> ExN02TrackerHitAllocator;
inline void* ExN02TrackerHit::operator new(size_t)
{
    void *aHit;
    aHit = (void *) ExN02TrackerHitAllocator.MallocSingle();
    return aHit;
}

inline void ExN02TrackerHit::operator delete(void *aHit)
{
    ExN02TrackerHitAllocator.FreeSingle((ExN02TrackerHit*) aHit);
}
```

ExN02TrackerHit.hh After TTS

```
extern __thread G4Allocator<ExN02TrackerHit> *ExN02TrackerHitAllocator;
inline void* ExN02TrackerHit::operator new(size_t)
{
   if (!ExN02TrackerHitAllocator)
      ExN02TrackerHitAllocator = new G4Allocator<ExN02TrackerHit>;
   void *aHit;
   aHit = (void *) (*ExN02TrackerHitAllocator).MallocSingle();
   return aHit;
}

inline void ExN02TrackerHit::operator delete(void *aHit)
{
   if (!ExN02TrackerHitAllocator)
      ExN02TrackerHitAllocator = new G4Allocator<ExN02TrackerHit>;
   (*ExN02TrackerHitAllocator).FreeSingle((ExN02TrackerHit*) aHit);
}
```

TTS for Others

ExN02TrackerHit.cc before TTS:

```
G4Allocator<ExN02TrackerHit> ExN02TrackerHitAllocator;
```

ExN02TrackerHit.cc after TTS:

```
__thread  G4Allocator<ExN02TrackerHit>  *ExN02TrackerHitAllocator  = 0;
```

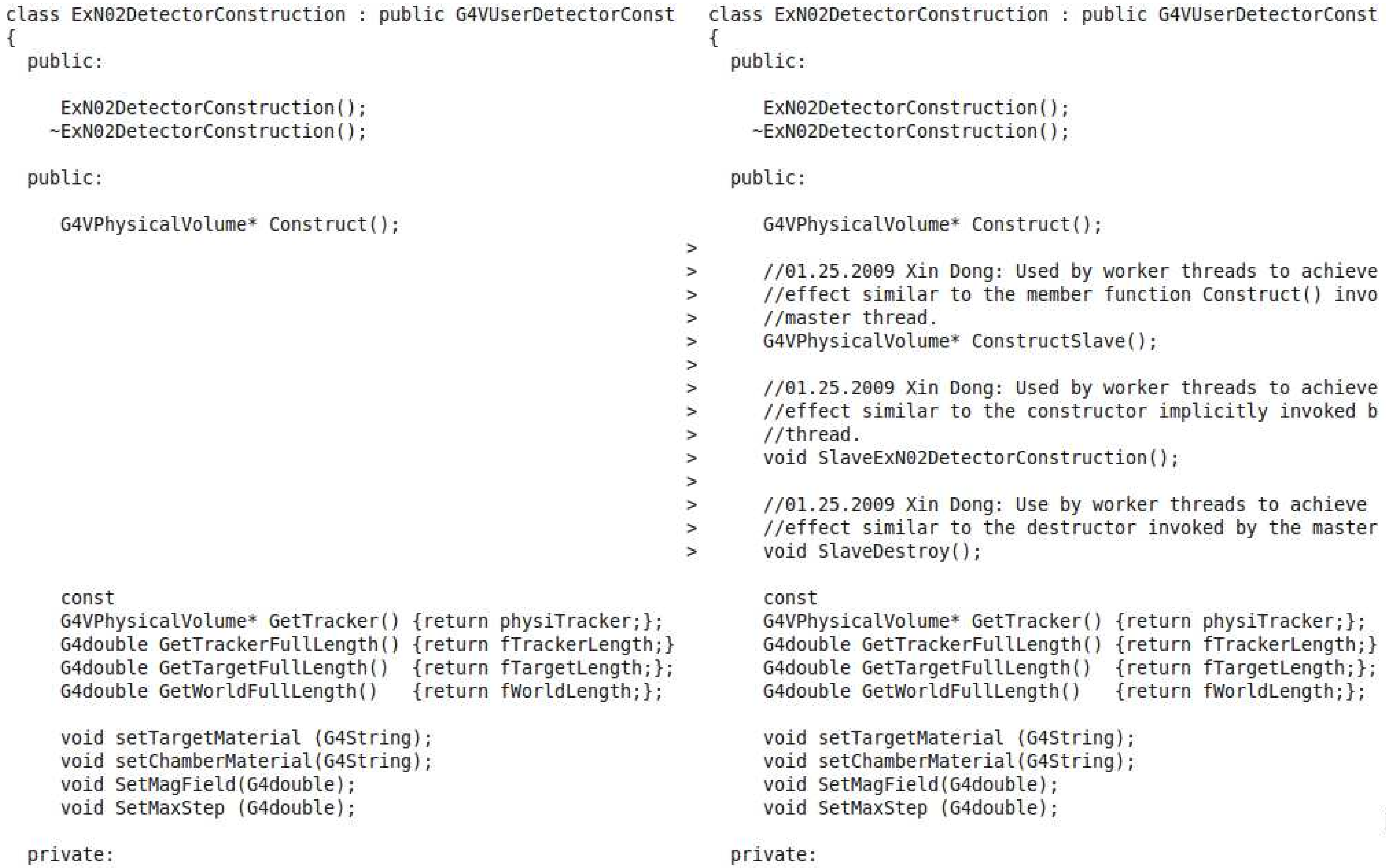
ExN02TrackerSD.cc before TTS:

```
static G4int HCID = -1;
```

ExN02TrackerSD.cc after TTS:

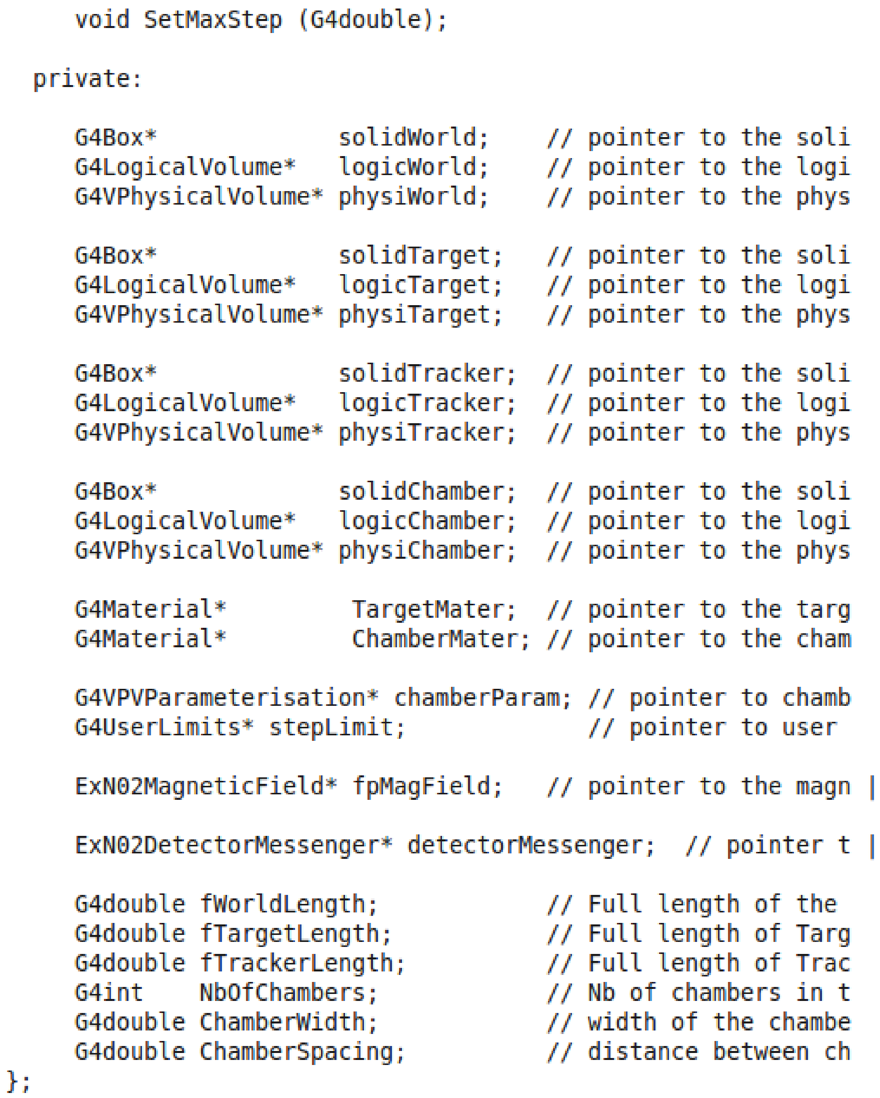
```
static __thread G4int HCID = -1;
```

TMR for ExN02DetectorConstruction.hh I

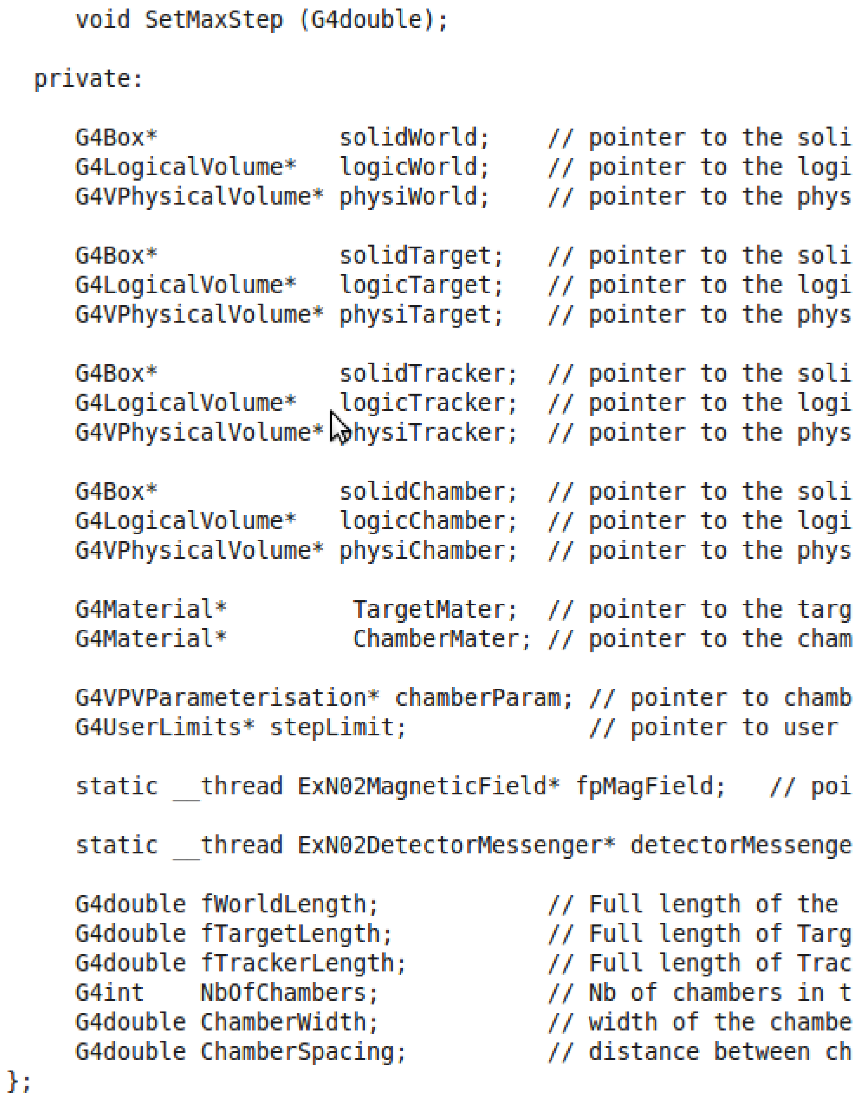


```
class ExN02DetectorConstruction : public G4VUserDetectorConst   class ExN02DetectorConstruction : public G4VUserDetectorConst
{                                                               {
  public:                                                         public:

     ExN02DetectorConstruction();                                    ExN02DetectorConstruction();
    ~ExN02DetectorConstruction();                                   ~ExN02DetectorConstruction();

  public:                                                         public:

     G4VPhysicalVolume* Construct();                                 G4VPhysicalVolume* Construct();
                                                              >
                                                              >      //01.25.2009 Xin Dong: Used by worker threads to achieve
                                                              >      //effect similar to the member function Construct() invo
                                                              >      //master thread.
                                                              >      G4VPhysicalVolume* ConstructSlave();
                                                              >
                                                              >      //01.25.2009 Xin Dong: Used by worker threads to achieve
                                                              >      //effect similar to the constructor implicitly invoked b
                                                              >      //thread.
                                                              >      void SlaveExN02DetectorConstruction();
                                                              >
                                                              >      //01.25.2009 Xin Dong: Use by worker threads to achieve
                                                              >      //effect similar to the destructor invoked by the master
                                                              >      void SlaveDestroy();

     const                                                           const
     G4VPhysicalVolume* GetTracker() {return physiTracker;};         G4VPhysicalVolume* GetTracker() {return physiTracker;};
     G4double GetTrackerFullLength() {return fTrackerLength;}        G4double GetTrackerFullLength() {return fTrackerLength;}
     G4double GetTargetFullLength()  {return fTargetLength;};        G4double GetTargetFullLength()  {return fTargetLength;};
     G4double GetWorldFullLength()   {return fWorldLength;};         G4double GetWorldFullLength()   {return fWorldLength;};

     void setTargetMaterial (G4String);                              void setTargetMaterial (G4String);
     void setChamberMaterial(G4String);                              void setChamberMaterial(G4String);
     void SetMagField(G4double);                                     void SetMagField(G4double);
     void SetMaxStep (G4double);                                     void SetMaxStep (G4double);

  private:                                                        private:
```

TMR for ExN02DetectorConstruction.hh II



```
     void SetMaxStep (G4double);

  private:

     G4Box*             solidWorld;    // pointer to the soli
     G4LogicalVolume*   logicWorld;    // pointer to the logi
     G4VPhysicalVolume* physiWorld;    // pointer to the phys

     G4Box*             solidTarget;   // pointer to the soli
     G4LogicalVolume*   logicTarget;   // pointer to the logi
     G4VPhysicalVolume* physiTarget;   // pointer to the phys

     G4Box*             solidTracker;  // pointer to the soli
     G4LogicalVolume*   logicTracker;  // pointer to the logi
     G4VPhysicalVolume* physiTracker;  // pointer to the phys

     G4Box*             solidChamber;  // pointer to the soli
     G4LogicalVolume*   logicChamber;  // pointer to the logi
     G4VPhysicalVolume* physiChamber;  // pointer to the phys

     G4Material*         TargetMater;  // pointer to the targ
     G4Material*         ChamberMater; // pointer to the cham

     G4VPVParameterisation* chamberParam; // pointer to chamb
     G4UserLimits* stepLimit;             // pointer to user

     ExN02MagneticField* fpMagField;   // pointer to the magn |

     ExN02DetectorMessenger* detectorMessenger;  // pointer t |

     G4double fWorldLength;            // Full length of the
     G4double fTargetLength;           // Full length of Targ
     G4double fTrackerLength;          // Full length of Trac
     G4int    NbOfChambers;            // Nb of chambers in t
     G4double ChamberWidth;            // width of the chambe
     G4double ChamberSpacing;          // distance between ch
};
```



```
     void SetMaxStep (G4double);

  private:

     G4Box*             solidWorld;    // pointer to the soli
     G4LogicalVolume*   logicWorld;    // pointer to the logi
     G4VPhysicalVolume* physiWorld;    // pointer to the phys

     G4Box*             solidTarget;   // pointer to the soli
     G4LogicalVolume*   logicTarget;   // pointer to the logi
     G4VPhysicalVolume* physiTarget;   // pointer to the phys

     G4Box*             solidTracker;  // pointer to the soli
     G4LogicalVolume*   logicTracker;  // pointer to the logi
     G4VPhysicalVolume* physiTracker;  // pointer to the phys

     G4Box*             solidChamber;  // pointer to the soli
     G4LogicalVolume*   logicChamber;  // pointer to the logi
     G4VPhysicalVolume* physiChamber;  // pointer to the phys

     G4Material*         TargetMater;  // pointer to the targ
     G4Material*         ChamberMater; // pointer to the cham

     G4VPVParameterisation* chamberParam; // pointer to chamb
     G4UserLimits* stepLimit;             // pointer to user

     static __thread ExN02MagneticField* fpMagField;   // poi

     static __thread ExN02DetectorMessenger* detectorMessenge

     G4double fWorldLength;            // Full length of the
     G4double fTargetLength;           // Full length of Targ
     G4double fTrackerLength;          // Full length of Trac
     G4int    NbOfChambers;            // Nb of chambers in t
     G4double ChamberWidth;            // width of the chambe
     G4double ChamberSpacing;          // distance between ch
};
```

TMR for ExN02DetectorConstruction.cc I

```
__thread ExN02MagneticField* ExN02DetectorConstruction::fpMagField =
0;

__thread ExN02DetectorMessenger* ExN02DetectorConstruction::detectorMessenger
= 0;

void ExN02DetectorConstruction::SlaveExN02DetectorConstruction()
{
    fpMagField = new ExN02MagneticField();
    detectorMessenger = new ExN02DetectorMessenger(this);
}
```

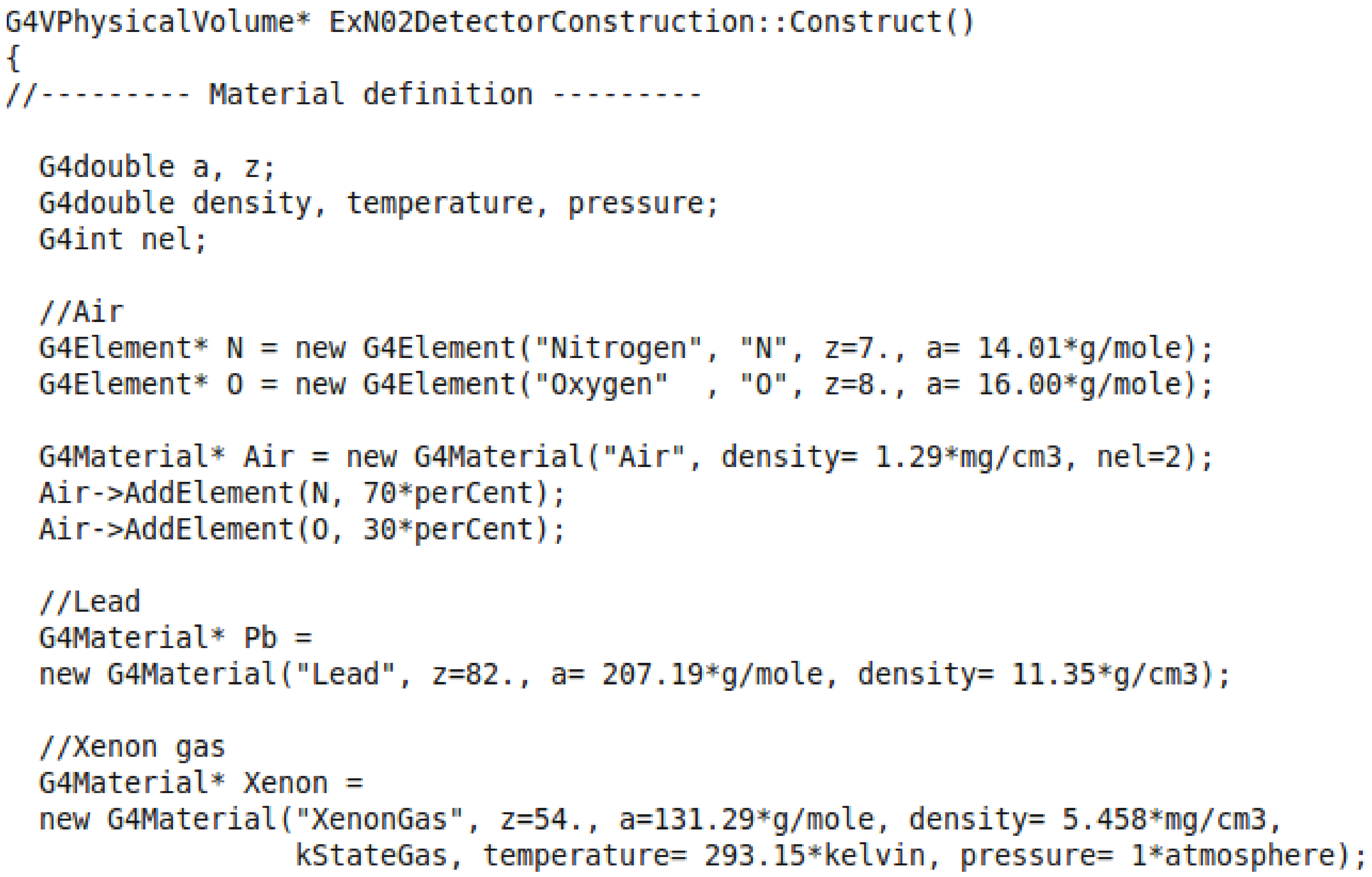
TMR for ExN02DetectorConstruction.cc II

```
ExN02DetectorConstruction::ExN02DetectorConstruction()
:solidWorld(0), logicWorld(0), physiWorld(0),
solidTarget(0), logicTarget(0), physiTarget(0),
solidTracker(0),logicTracker(0),physiTracker(0),
solidChamber(0),logicChamber(0),physiChamber(0),
TargetMater(0), ChamberMater(0),chamberParam(0),
stepLimit(0),
fWorldLength(0.), fTargetLength(0.), fTrackerLength(0.),
NbOfChambers(0) , ChamberWidth(0.), ChamberSpacing(0.)
{
   fpMagField = new ExN02MagneticField();
   detectorMessenger = new ExN02DetectorMessenger(this);
}
```

TMR for ExN02DetectorConstruction.cc III

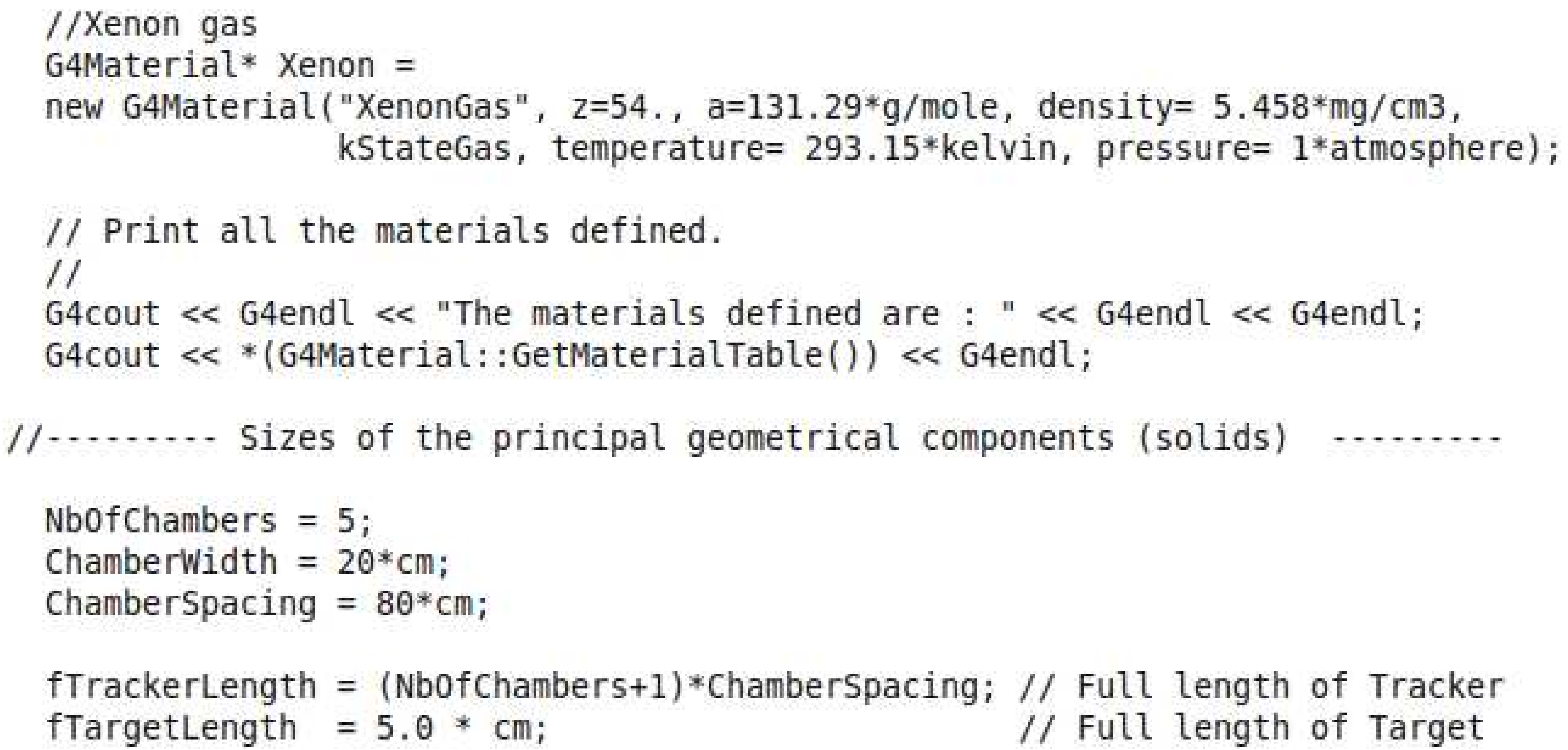
```
void ExN02DetectorConstruction::SlaveDestroy()
{
   delete fpMagField;
   delete detectorMessenger;
}

ExN02DetectorConstruction:: ExN02DetectorConstruction()
{
   delete fpMagField;
   delete stepLimit;
   delete chamberParam;
   delete detectorMessenger;
}
```

TMR for ExN02DetectorConstruction.cc I



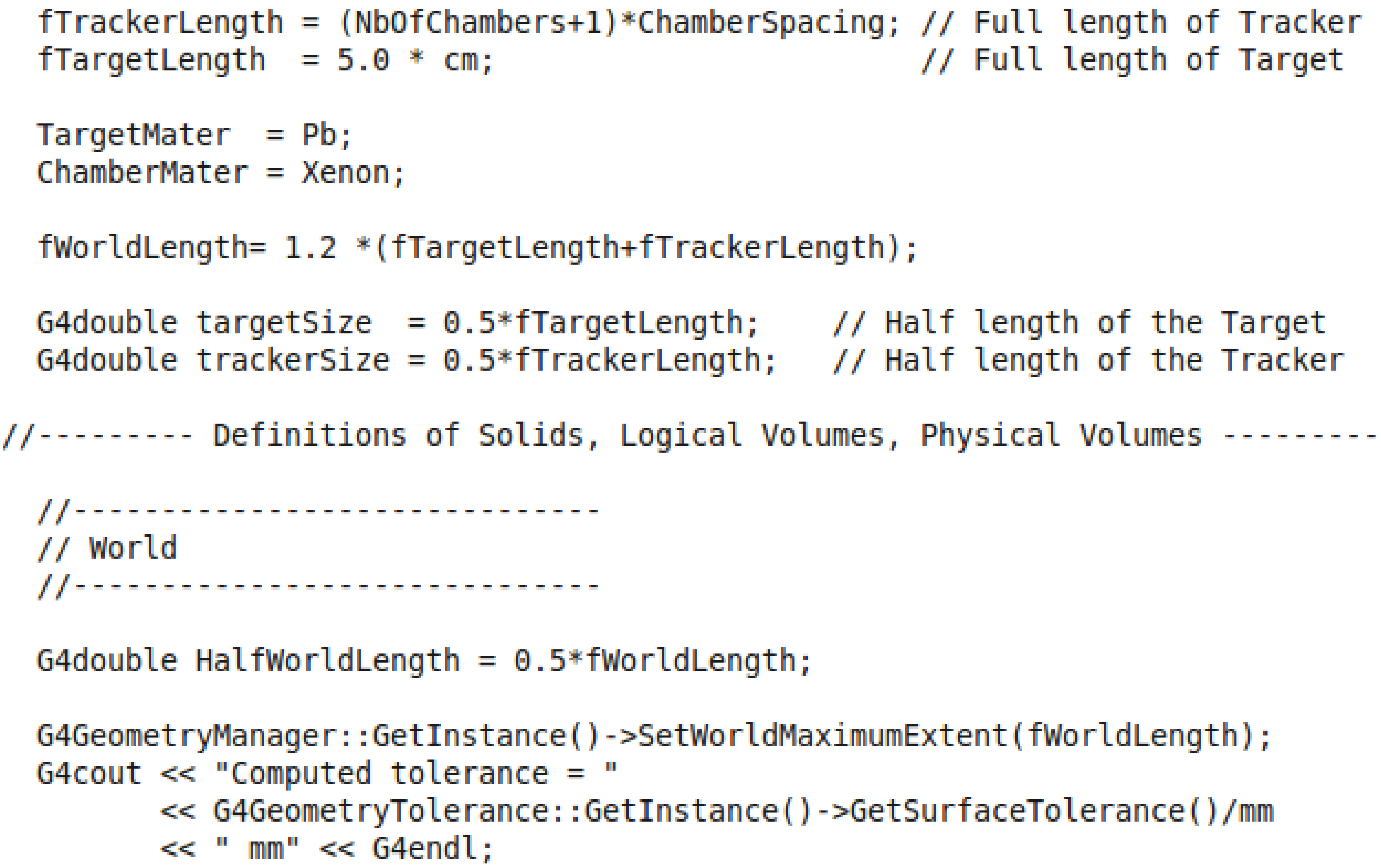
```
G4VPhysicalVolume* ExN02DetectorConstruction::Construct()
{
//--------- Material definition ---------

  G4double a, z;
  G4double density, temperature, pressure;
  G4int nel;

  //Air
  G4Element* N = new G4Element("Nitrogen", "N", z=7., a= 14.01*g/mole);
  G4Element* O = new G4Element("Oxygen"  , "O", z=8., a= 16.00*g/mole);

  G4Material* Air = new G4Material("Air", density= 1.29*mg/cm3, nel=2);
  Air->AddElement(N, 70*perCent);
  Air->AddElement(O, 30*perCent);

  //Lead
  G4Material* Pb =
  new G4Material("Lead", z=82., a= 207.19*g/mole, density= 11.35*g/cm3);

  //Xenon gas
  G4Material* Xenon =
  new G4Material("XenonGas", z=54., a=131.29*g/mole, density= 5.458*mg/cm3,
                 kStateGas, temperature= 293.15*kelvin, pressure= 1*atmosphere);
```

TMR for ExN02DetectorConstruction.cc II



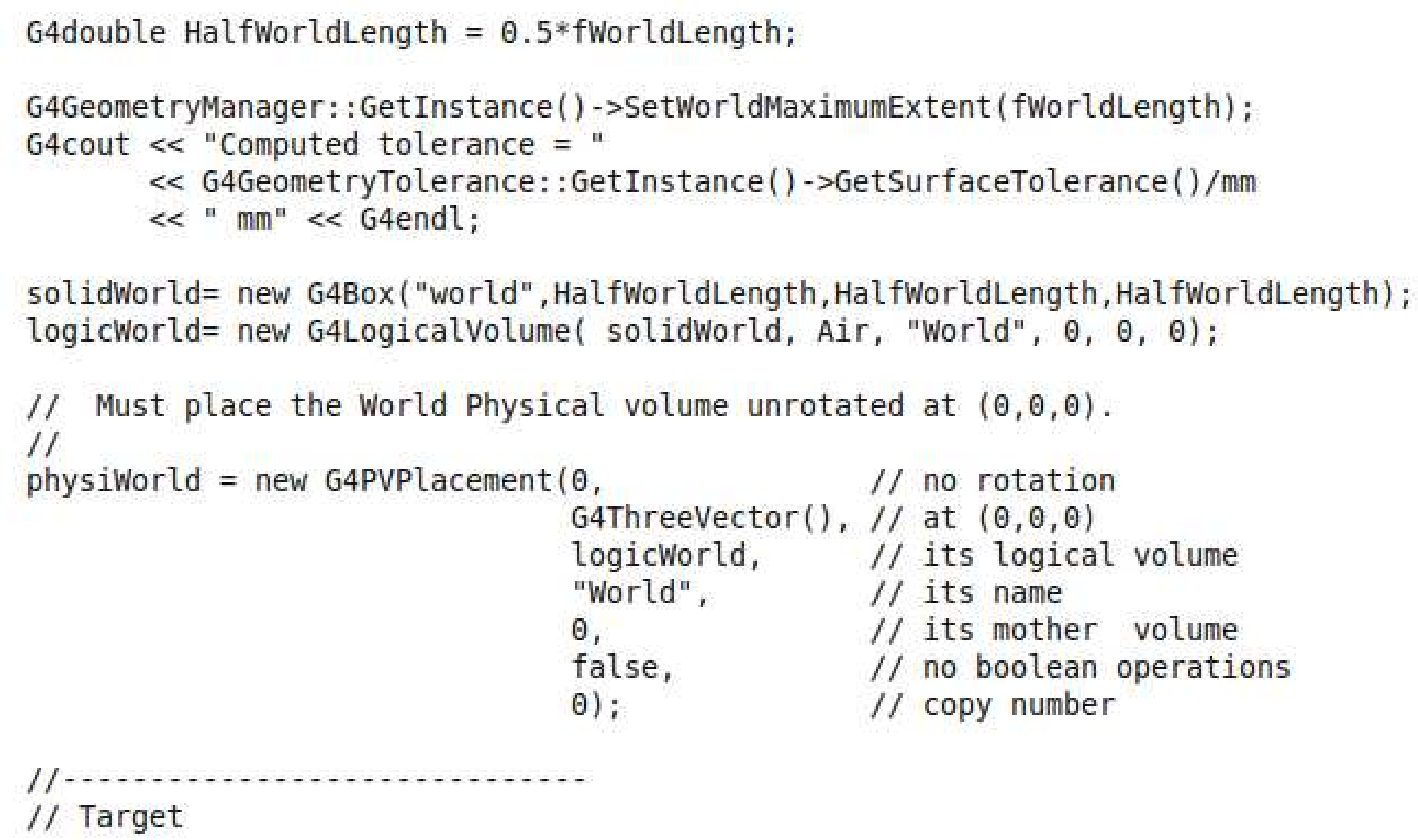
```
  //Xenon gas
  G4Material* Xenon =
  new G4Material("XenonGas", z=54., a=131.29*g/mole, density= 5.458*mg/cm3,
                 kStateGas, temperature= 293.15*kelvin, pressure= 1*atmosphere);

  // Print all the materials defined.
  //
  G4cout << G4endl << "The materials defined are : " << G4endl << G4endl;
  G4cout << *(G4Material::GetMaterialTable()) << G4endl;

//--------- Sizes of the principal geometrical components (solids)  ---------

  NbOfChambers = 5;
  ChamberWidth = 20*cm;
  ChamberSpacing = 80*cm;

  fTrackerLength = (NbOfChambers+1)*ChamberSpacing; // Full length of Tracker
  fTargetLength  = 5.0 * cm;                        // Full length of Target
```

TMR for ExN02DetectorConstruction.cc III



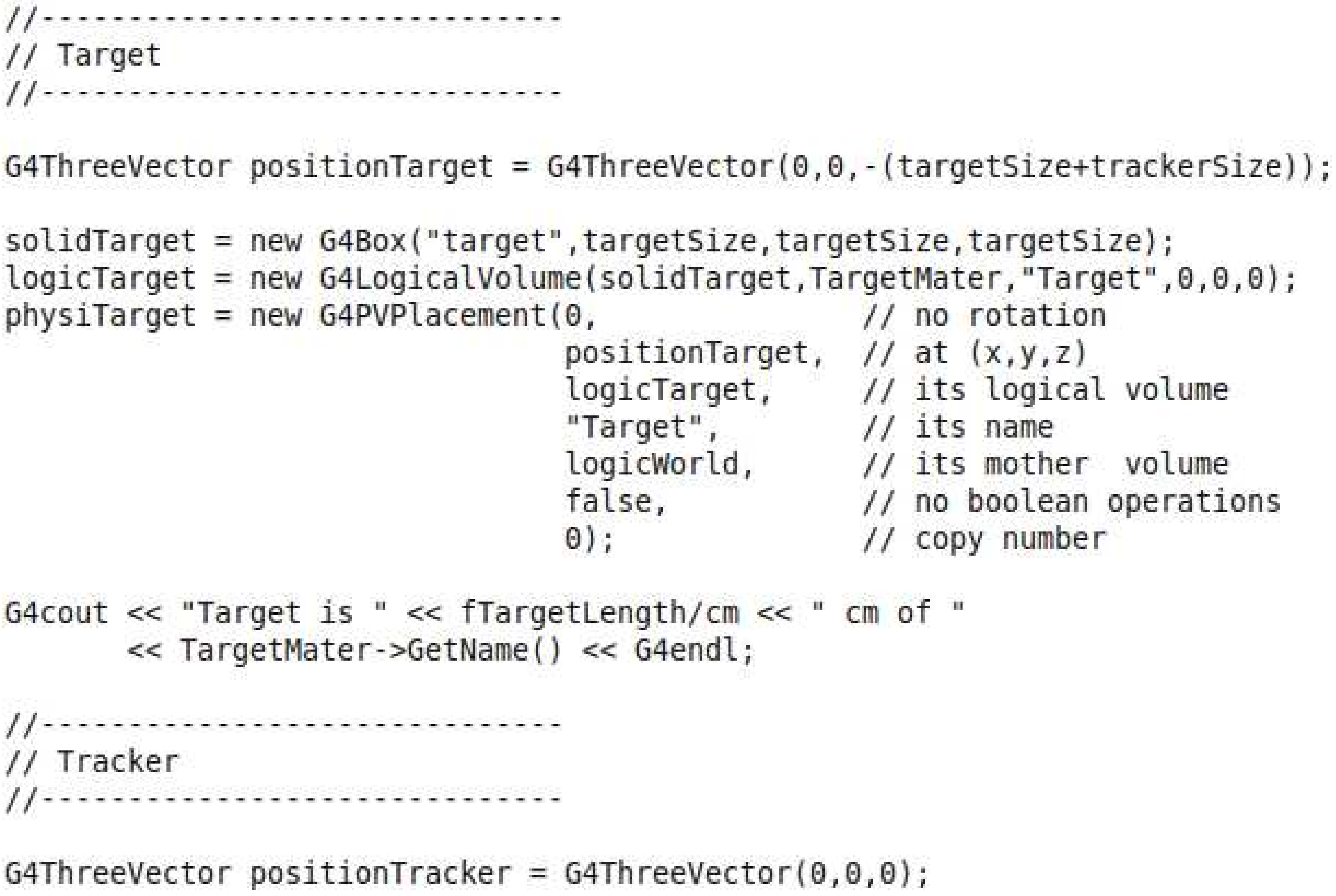
```
  fTrackerLength = (NbOfChambers+1)*ChamberSpacing; // Full length of Tracker
  fTargetLength  = 5.0 * cm;                         // Full length of Target

  TargetMater  = Pb;
  ChamberMater = Xenon;

  fWorldLength= 1.2 *(fTargetLength+fTrackerLength);

  G4double targetSize  = 0.5*fTargetLength;     // Half length of the Target
  G4double trackerSize = 0.5*fTrackerLength;    // Half length of the Tracker

//--------- Definitions of Solids, Logical Volumes, Physical Volumes ---------

  //------------------------------
  // World
  //------------------------------

  G4double HalfWorldLength = 0.5*fWorldLength;

  G4GeometryManager::GetInstance()->SetWorldMaximumExtent(fWorldLength);
  G4cout << "Computed tolerance = "
         << G4GeometryTolerance::GetInstance()->GetSurfaceTolerance()/mm
         << " mm" << G4endl;
```

TMR for ExN02DetectorConstruction.cc IV



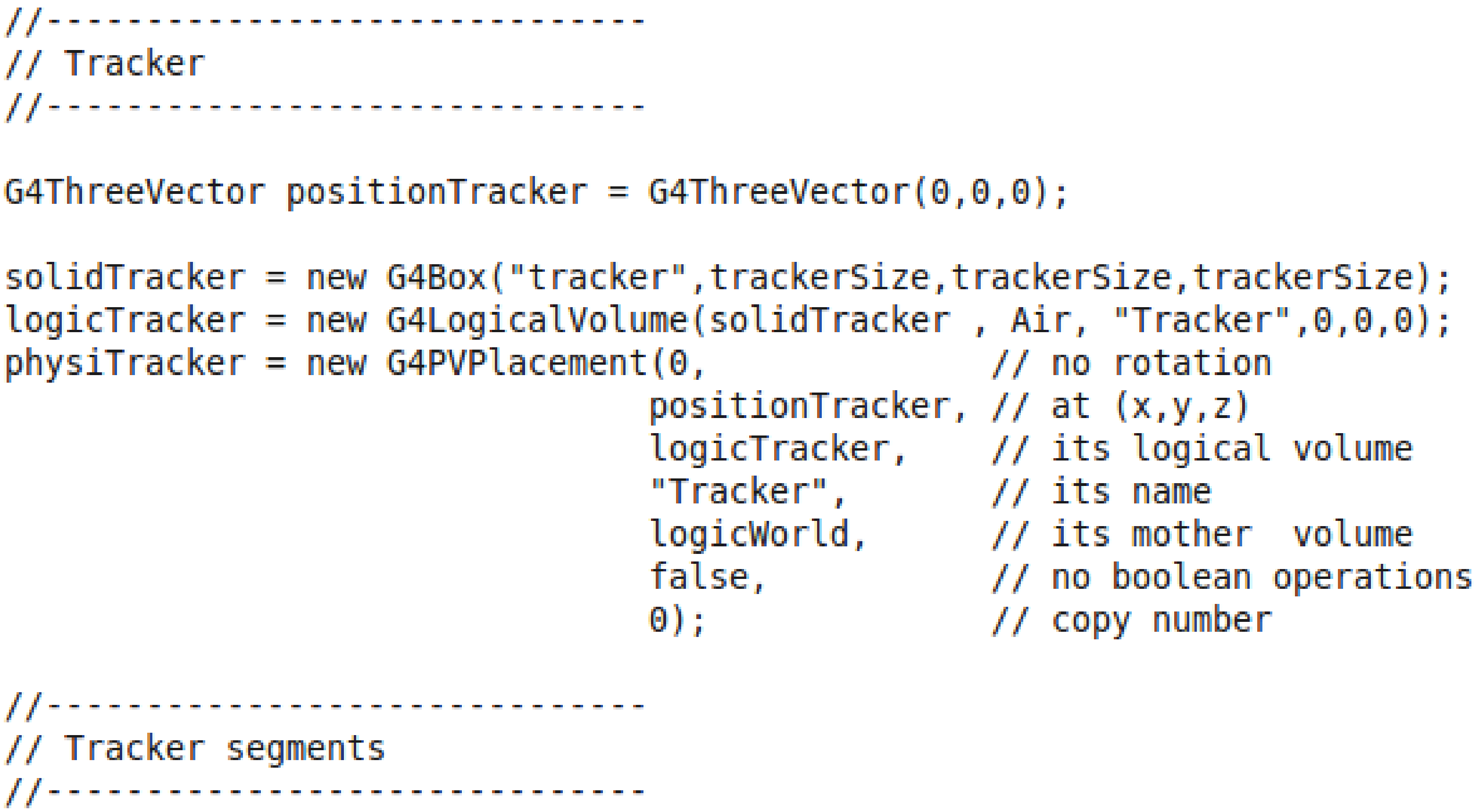
```
G4double HalfWorldLength = 0.5*fWorldLength;

G4GeometryManager::GetInstance()->SetWorldMaximumExtent(fWorldLength);
G4cout << "Computed tolerance = "
       << G4GeometryTolerance::GetInstance()->GetSurfaceTolerance()/mm
       << " mm" << G4endl;

solidWorld= new G4Box("world",HalfWorldLength,HalfWorldLength,HalfWorldLength);
logicWorld= new G4LogicalVolume( solidWorld, Air, "World", 0, 0, 0);

//  Must place the World Physical volume unrotated at (0,0,0).
//
physiWorld = new G4PVPlacement(0,               // no rotation
                               G4ThreeVector(), // at (0,0,0)
                               logicWorld,      // its logical volume
                               "World",         // its name
                               0,               // its mother  volume
                               false,           // no boolean operations
                               0);              // copy number

//------------------------------
// Target
```

TMR for ExN02DetectorConstruction.cc V



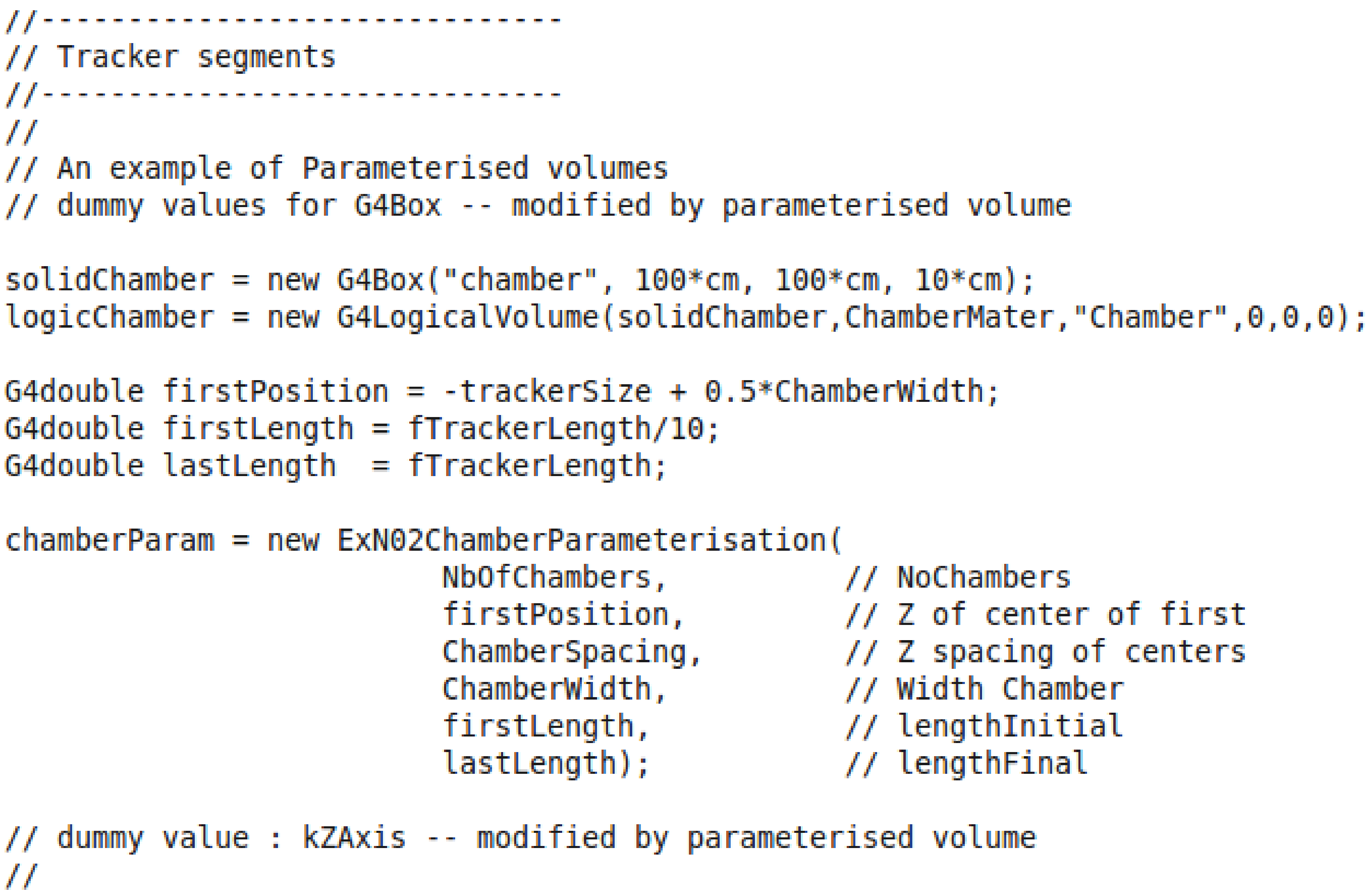
```
//------------------------------
// Target
//------------------------------

G4ThreeVector positionTarget = G4ThreeVector(0,0,-(targetSize+trackerSize));

solidTarget = new G4Box("target",targetSize,targetSize,targetSize);
logicTarget = new G4LogicalVolume(solidTarget,TargetMater,"Target",0,0,0);
physiTarget = new G4PVPlacement(0,               // no rotation
                                positionTarget,  // at (x,y,z)
                                logicTarget,     // its logical volume
                                "Target",        // its name
                                logicWorld,      // its mother  volume
                                false,           // no boolean operations
                                0);              // copy number

G4cout << "Target is " << fTargetLength/cm << " cm of "
       << TargetMater->GetName() << G4endl;

//------------------------------
// Tracker
//------------------------------

G4ThreeVector positionTracker = G4ThreeVector(0,0,0);
```

TMR for ExN02DetectorConstruction.cc VI



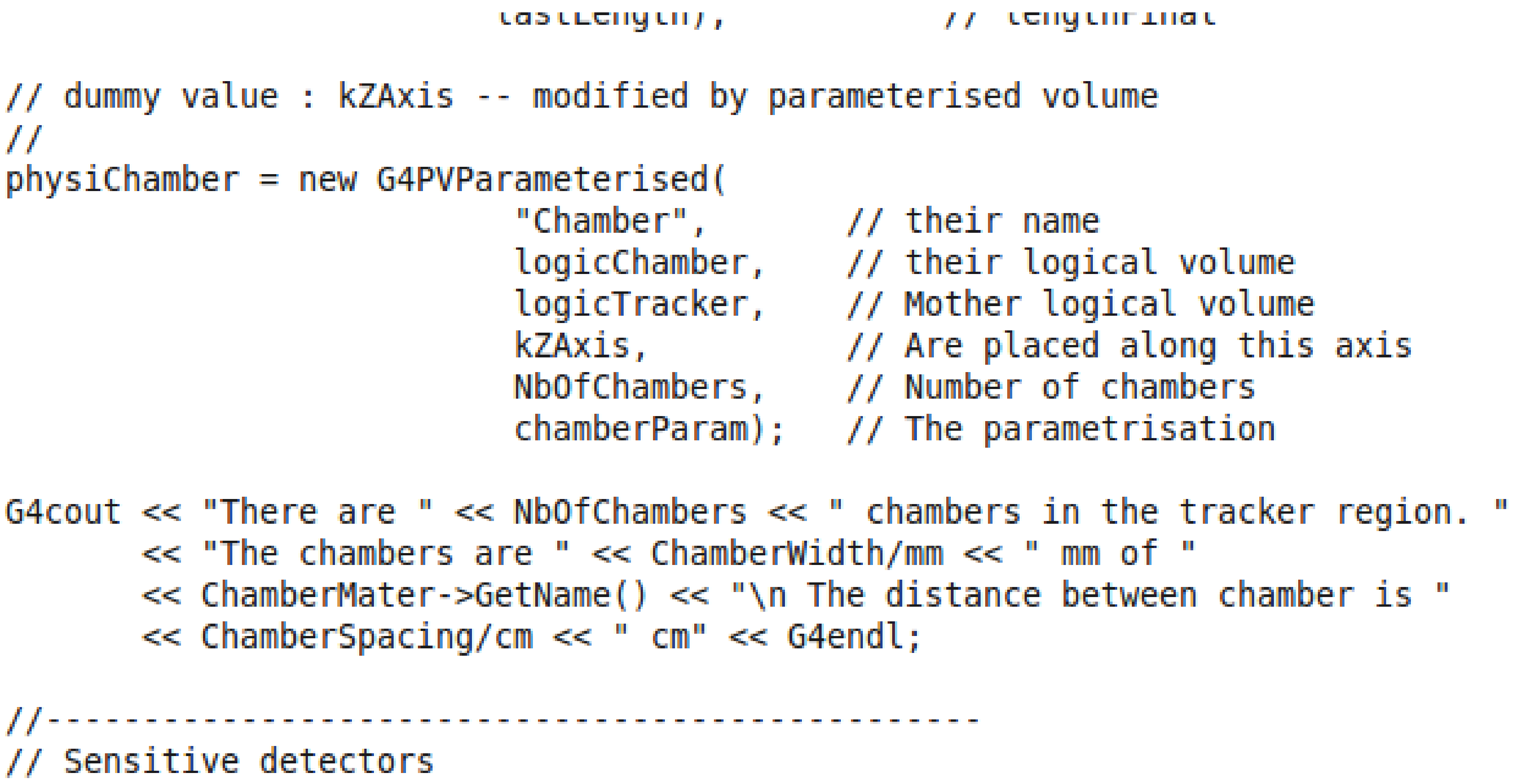
```
//------------------------------
// Tracker
//------------------------------

G4ThreeVector positionTracker = G4ThreeVector(0,0,0);

solidTracker = new G4Box("tracker",trackerSize,trackerSize,trackerSize);
logicTracker = new G4LogicalVolume(solidTracker , Air, "Tracker",0,0,0);
physiTracker = new G4PVPlacement(0,              // no rotation
                                positionTracker, // at (x,y,z)
                                logicTracker,    // its logical volume
                                "Tracker",       // its name
                                logicWorld,      // its mother  volume
                                false,           // no boolean operations
                                0);              // copy number

//------------------------------
// Tracker segments
//------------------------------
```

TMR for ExN02DetectorConstruction.cc VII



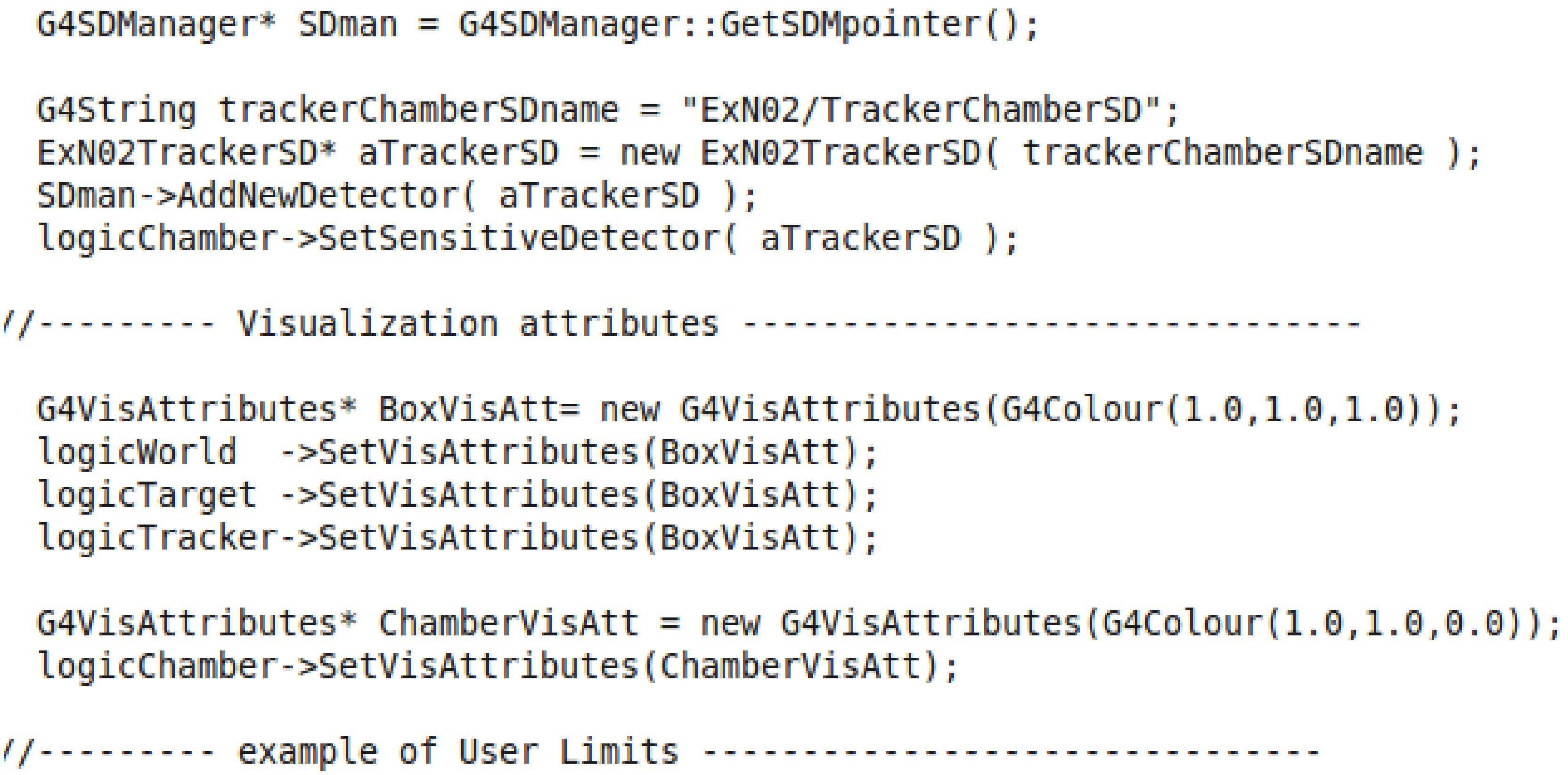
```
//------------------------------
// Tracker segments
//------------------------------
//
// An example of Parameterised volumes
// dummy values for G4Box -- modified by parameterised volume

solidChamber = new G4Box("chamber", 100*cm, 100*cm, 10*cm);
logicChamber = new G4LogicalVolume(solidChamber,ChamberMater,"Chamber",0,0,0);

G4double firstPosition = -trackerSize + 0.5*ChamberWidth;
G4double firstLength = fTrackerLength/10;
G4double lastLength  = fTrackerLength;

chamberParam = new ExN02ChamberParameterisation(
                         NbOfChambers,          // NoChambers
                         firstPosition,         // Z of center of first
                         ChamberSpacing,        // Z spacing of centers
                         ChamberWidth,          // Width Chamber
                         firstLength,           // lengthInitial
                         lastLength);           // lengthFinal

// dummy value : kZAxis -- modified by parameterised volume
//
```

TMR for ExN02DetectorConstruction.cc VIII



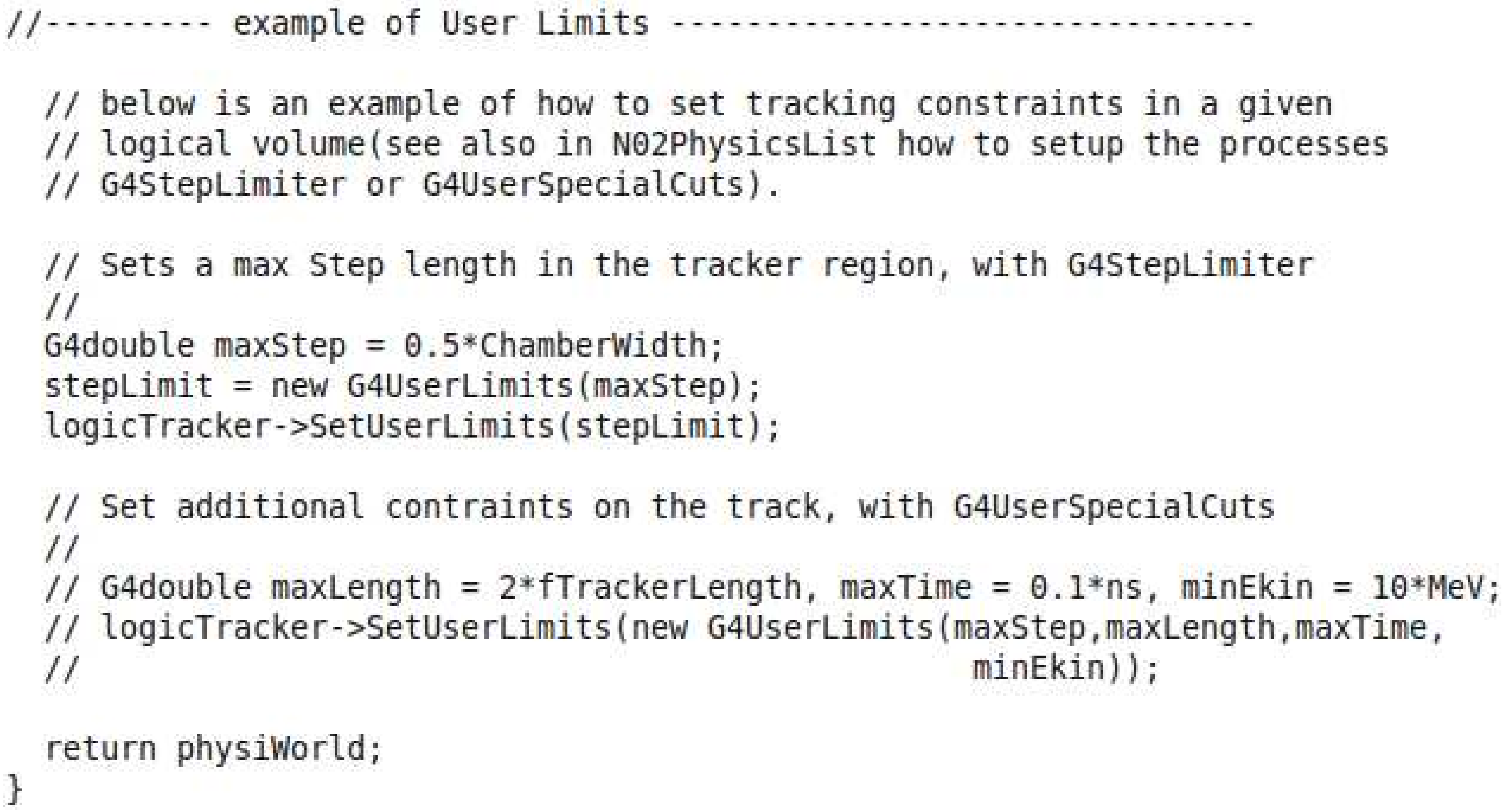
```
                         lastLength),           // lengthFinal

// dummy value : kZAxis -- modified by parameterised volume
//
physiChamber = new G4PVParameterised(
                          "Chamber",       // their name
                          logicChamber,    // their logical volume
                          logicTracker,    // Mother logical volume
                          kZAxis,          // Are placed along this axis
                          NbOfChambers,    // Number of chambers
                          chamberParam);   // The parametrisation

G4cout << "There are " << NbOfChambers << " chambers in the tracker region. "
       << "The chambers are " << ChamberWidth/mm << " mm of "
       << ChamberMater->GetName() << "\n The distance between chamber is "
       << ChamberSpacing/cm << " cm" << G4endl;

//--------------------------------------------------
// Sensitive detectors
```

TMR for ExN02DetectorConstruction.cc IX



```
  G4SDManager* SDman = G4SDManager::GetSDMpointer();

  G4String trackerChamberSDname = "ExN02/TrackerChamberSD";
  ExN02TrackerSD* aTrackerSD = new ExN02TrackerSD( trackerChamberSDname );
  SDman->AddNewDetector( aTrackerSD );
  logicChamber->SetSensitiveDetector( aTrackerSD );

//--------- Visualization attributes -------------------------------

  G4VisAttributes* BoxVisAtt= new G4VisAttributes(G4Colour(1.0,1.0,1.0));
  logicWorld  ->SetVisAttributes(BoxVisAtt);
  logicTarget ->SetVisAttributes(BoxVisAtt);
  logicTracker->SetVisAttributes(BoxVisAtt);

  G4VisAttributes* ChamberVisAtt = new G4VisAttributes(G4Colour(1.0,1.0,0.0));
  logicChamber->SetVisAttributes(ChamberVisAtt);

//--------- example of User Limits -------------------------------
```

TMR for ExN02DetectorConstruction.cc X



```
//--------- example of User Limits -------------------------------

  // below is an example of how to set tracking constraints in a given
  // logical volume(see also in N02PhysicsList how to setup the processes
  // G4StepLimiter or G4UserSpecialCuts).

  // Sets a max Step length in the tracker region, with G4StepLimiter
  //
  G4double maxStep = 0.5*ChamberWidth;
  stepLimit = new G4UserLimits(maxStep);
  logicTracker->SetUserLimits(stepLimit);

  // Set additional contraints on the track, with G4UserSpecialCuts
  //
  // G4double maxLength = 2*fTrackerLength, maxTime = 0.1*ns, minEkin = 10*MeV;
  // logicTracker->SetUserLimits(new G4UserLimits(maxStep,maxLength,maxTime,
  //                                               minEkin));

  return physiWorld;
}
```

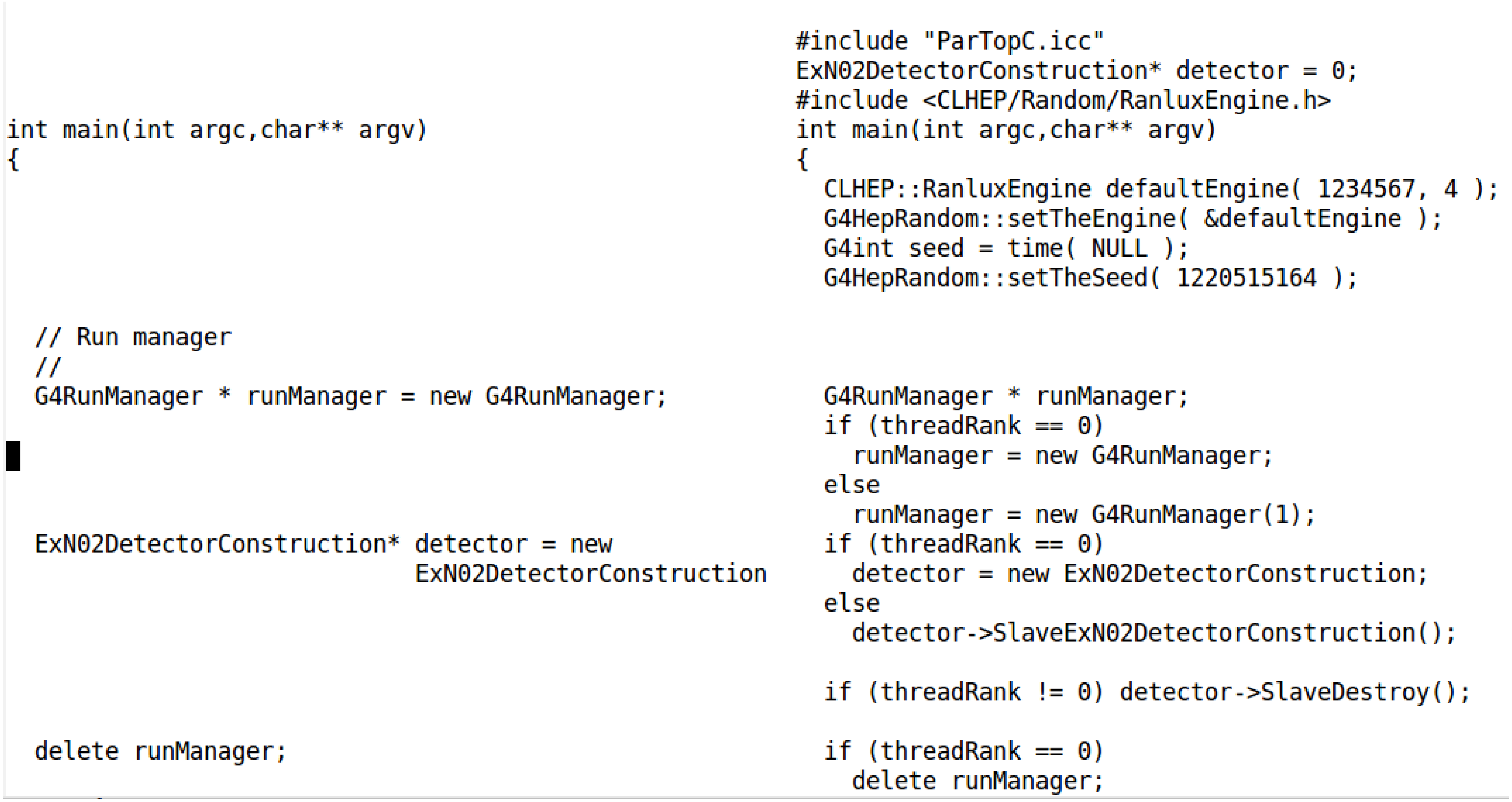
TMR for ExN02DetectorConstruction.cc XI

```
G4VPhysicalVolume* ExN02DetectorConstruction::ConstructSlave()
{
   // Sensitive detectors
   G4SDManager* SDman = G4SDManager::GetSDMpointer();
   G4String trackerChamberSDname = ”ExN02/TrackerChamberSD”;
   ExN02TrackerSD* aTrackerSD = new ExN02TrackerSD( trackerChamberSDname )
   SDman->AddNewDetector( aTrackerSD );
   logicChamber->SetSensitiveDetector( aTrackerSD );

   // Visualization attributes
   G4VisAttributes* BoxVisAtt= new G4VisAttributes(G4Colour(1.0,1.0,1.0));
   logicWorld ->SetVisAttributes(BoxVisAtt);
   logicTarget ->SetVisAttributes(BoxVisAtt);
   logicTracker->SetVisAttributes(BoxVisAtt);

   G4VisAttributes* ChamberVisAtt = new G4VisAttributes(G4Colour(1.0,1.0,0.0));
   logicChamber->SetVisAttributes(ChamberVisAtt);
```

TMR for ExN02DetectorConstruction.cc XII

```
  G4double maxStep = 0.5*ChamberWidth;
  stepLimit = new G4UserLimits(maxStep);
  logicTracker->SetUserLimits(stepLimit);

  return physiWorld;
}
```

TMR for The Application Main



```
                                                        #include "ParTopC.icc"
                                                        ExN02DetectorConstruction* detector = 0;
                                                        #include <CLHEP/Random/RanluxEngine.h>
int main(int argc,char** argv)                          int main(int argc,char** argv)
{                                                       {
                                                          CLHEP::RanluxEngine defaultEngine( 1234567, 4 );
                                                          G4HepRandom::setTheEngine( &defaultEngine );
                                                          G4int seed = time( NULL );
                                                          G4HepRandom::setTheSeed( 1220515164 );

  // Run manager
  //
  G4RunManager * runManager = new G4RunManager;           G4RunManager * runManager;
                                                          if (threadRank == 0)
                                                            runManager = new G4RunManager;
                                                          else
                                                            runManager = new G4RunManager(1);
  ExN02DetectorConstruction* detector = new               if (threadRank == 0)
                             ExN02DetectorConstruction      detector = new ExN02DetectorConstruction;
                                                          else
                                                            detector->SlaveExN02DetectorConstruction();

                                                          if (threadRank != 0) detector->SlaveDestroy();

  delete runManager;                                      if (threadRank == 0)
                                                            delete runManager;
```

Questions

Thank You.