DEDEKTÖR TASARIMI & BENZETİMİ

Doç. Dr. Sinan KUDAY

Ankara Üniversitesi Fizik Bölümü
skuday@ankara.edu.tr

İÇİNDEKİLER

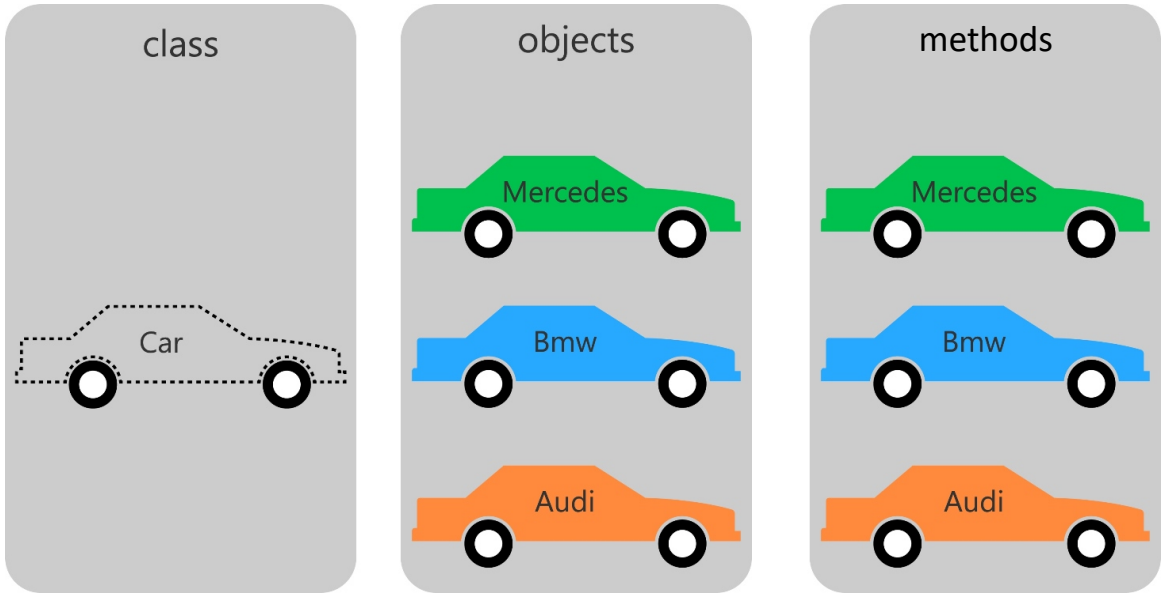
1. Dedektör Benzetimleri Terminolojisi
2. Geometrik Modelleme
3. Materyal ve Nesne Yönetimi
4. Fiziksel Etkileşimler

- Geant4 Kurulumu
- Geant4 Kod Yapısı
- Örnek Benzetimler

⇨ Ek Yansılar diğer sunumda (Geant4 için okuyunuz)

Kısım 1:
Detektör Benzetimleri Terminolojisi

Nesne (Obje) Yönelimli Programlama (Sınıf Yapısı)



keyword user-defined name

```
class ClassName
{ Access specifier:          //can be private,public or protected
  Data members;              // Variables to be used
  Member Functions() { }     //Methods to access data members
};                           // Class name ends with a semicolon
```

```
class CCar
{
private:
    int m_iNumberOfBolts;

public:

    int m_iCarCurrentSpeed;

    CCar()
    {
        m_iCarCurrentSpeed  =  0;
        m_iCurrentCarGear   =  0;
        m_iNumberOfBolts    =  78569;
    }

protected:

    int m_iCurrentCarGear;
    int GetCurrentSpeed() {return m_iCarCurrentSpeed;}
};
```

CCar Mercedes;
CCar Bmw;
CCar Audi;

<Sınıf Adı> Obje Adı;

Mercedes.GetCurrentSpeed();
Bmw.GetCurrentSpeed(); → «0»
Audi.GetCurrentSpeed();

Bmw.OpenBaggageDoor(); → ERROR: Undefined method in class !!!

→ Tüm Geant4 Kütüphanesi bu tür sınıflardan oluşur

Nesne (Obje) Yönelimli Programlama (Türetilmiş Sınıf Yapısı)

Temel sınıf sentaksı:

```
#include<iostream>
using namespace std;
class base {
 public:
base()
 {
 cout<<"base class constructor"<<endl;
  }
~base()
 {
 cout<<"base class destructor"<<endl;
   }
 };
```

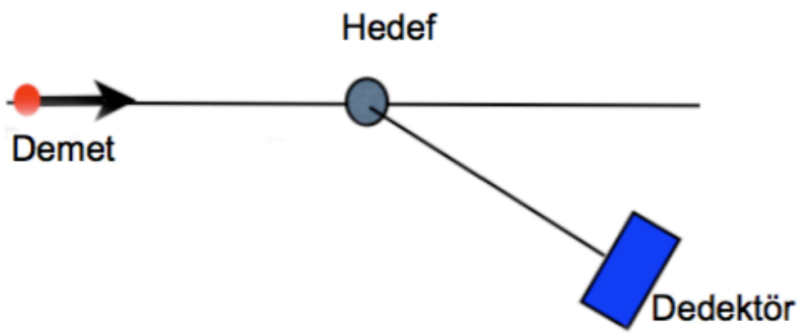
Temel sınıflar G4 kütüphanesinde zaten yaratılmıştır. Kod içinde #include base.hh şeklinde çağrılırlar.

Türetilmiş sınıf sentaksı:

```
class derived:public base
{
 public:
derived()
{
 cout<<"derived class constructor"<<endl;
  }
 ~derived()
  {
 cout<<"derived class destructor"<<endl;
   }
};
 int main()
{
  derived d;
  return 0;
}
```

G4 örneklerinin tamamı ve bizim yaratacağımız kodlarda src ve inc

Deneysel Benzetimlerde Nesneler



Basit bir nükleer fizik deneyinin benzetimi için gereken temel nesneler şunlardır:

1. Demet
2. Hedef
3. Dedektör
4. Etkileşim Alanları
5. Fiziksel Etkileşimler
6. Analiz

Deneysel Benzetimlerde Terminoloji

- Run, Event, Track, Step, Step point
- Track (İz) <-> Trajectory (yörünge); Step <-> Yörünge noktaları
- Process (At rest, along step, post step)
- Cut = Production Threshold (Üretim Eşik Değerleri)
- Sensitive detector (SD), Score, Hit, Hits collection,
- Bit, Byte, Word, Multithreaded

Geant4 Terminolojisi (RUN)

- Run: Çalışma (Koşma)
- Gerçek deneylerde; deney sisteminin yeterince ölçüm alıncaya kadar çalıştırılarak işletilmesi süresine bir çalışma (run) denir.
- Benzetimlerde de aynı terminoloji kullanılır.
- Geant4 için **"BeamOn"** komutu çalışmayı başlatan komuttur.
- Benzetimlerde **tek çalışma bir olay döngüsünün gerçekleştiği aralıktır.**
- **G4RunManager** sınıfı çalışmanın yönetimini ve temsilini sağlar.
- **G4Run** sınıfı kullanıcı tanımlı çalışmaların türetilmesini sağlar. (Bir çalışma objesi o çalışmadaki özet bilgileri içerebilir.)
- **G4RunUserAction** opsiyonel olarka kullanılabilir.

Geant4 Terminolojisi (EVENT)

Bir olay, detektörde birçok Hit'e karşılık gelebilir !!

- Event: Olay - G4 benzetiminin en küçük birimidir.
- Benzetim başladığında öncül (primer) izler birleştirilerek istiflenir.
- İstiflenen izler tek tek ele alınarak ikincil (sekonder) izler belirlenir.
- Elde edilen tüm izlerin istiflenmesi bitinceye kadar işlem devam eder.
- Böylece "Tracking" işlemi yapılmış olur.
- **G4Event** sınıfı tek bir olayı temsil eder. İşlem sonunda aşağıdaki nesnelere sahip olur;

  ➡ Input: Öncül köşeler (primary vertices) ve parçacık listesi

  ➡ Output: Hit sayıları ve Yörüngeler (Trajectories)

- **G4EventManager** sınıfı bir olayın üretilmesini yönetir. Opsiyonel olarak kullanıcı tanımlı olaylar **G4UserEventAction** ile yönetilir.

Geant4 Terminolojisi (TRACK)

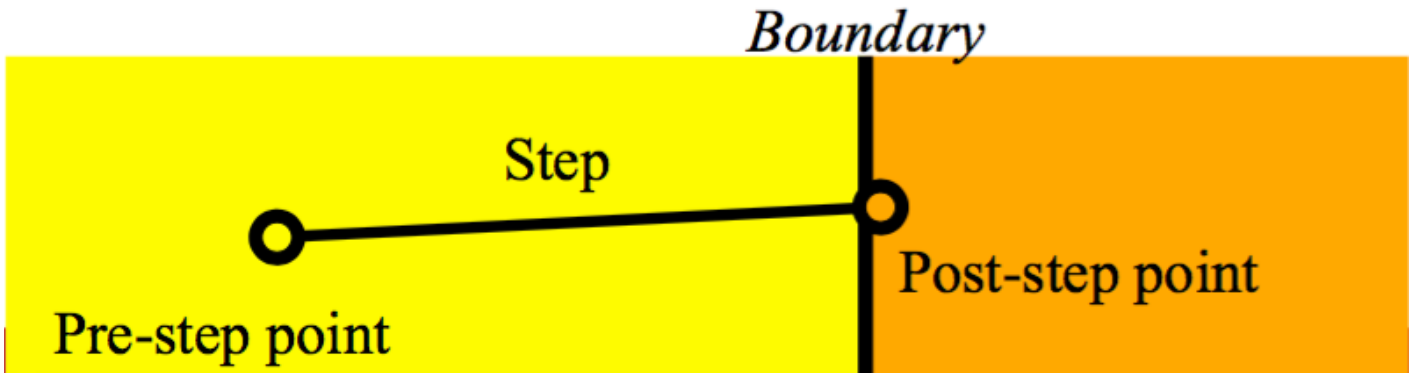
- Track: İz - G4 içinde bir parçacığın anlık (ekran) görüntüsüdür.
- Parçacığın o an içindeki kısa zaman aralığındaki fiziksel miktarıdır.
- Step tanımıyla karıştırılmamalıdır.
- İzler hızlıca işlemlenir ve kinetik enerjisi sıfıra düşünce silinir.
- Geometrilerin dışına çıkan, bozunmaya ve çarpışmaya uğrayan izler de silinebilir.
- **G4Track** sınıfı tek bir izi temsil eder. Bir olayın bitiminde hiçbir iz nesnesi devam edemez.

  ➡ Tüm izlerin kaydedilmesi ile yörünge (trajectory) sınıfı oluşur.
- **G4TrackingManager** sınıfı bir izin üretilmesini yönetir. Opsiyonel olarak kullanıcı tanımlı izler **G4UserTrackingAction** ile yönetilir.

Geant4 Terminolojisi (STEP)

Enerji kaybı ve ToF

- Step: Adım - G4 içinde bir parçacığın delta bilgisini taşır.
- Adım uçlarındaki noktalar boyunca hacim ve materyal bilgisi tutulur.
- Eğer noktalardan birisi sınıra denk geldiyse; sonraki hacime ait olarak kaydedilir.
- **G4Step** sınıfı tek bir adımı temsil eder.
- **G4SteppingManager** sınıfı bir adımın üretilmesini yönetir. Opsiyonel olarak kullanıcı tanımlı adımlar **G4UserSteppingAction** ile yönetilir.



Geant4 Terminolojisi (TRAJECTORY)

- Trajectory: Yörünge - G4 iz bilgisinin kaybolmadan depolandığı sınıftır.
- Adım uçlarındaki noktalar boyunca hacim ve materyal bilgisi tutulur.
- **G4Trajectory** sınıfı tek yörünge oluşumunu temsil eder.

  ➡ G4Track nesnesinin bazı bilgilerini kopyalar.
- **G4TrajectoryPoint** sınıfı yörünge noktasının oluşumunu temsil eder.

  ➡ G4Step nesnesinin bazı bilgilerini kopyalar.
- G4Trajectory nesnesi G4TrajectoryPoint sınıfından bir vektöre sahiptir.
- Event işlemleme sonucunda, G4Events nesnesi bir dizi G4Trajectory nesneleriyle dolmalıdır **(/tracking/storeTrajectory must be set to 1)**
- Programların çok fazla miktarda yörünge depolamamasına dikkat edilmelidir. Hafıza ve işlemleme yavaşlayabilir !!

Geant4 Terminolojisi (PARTICLE)

- Particle: Parçacık - G4 içinde 4 farklı sınıf yoluyla tanımlanır:
- **G4Track** sınıfı;
  - ➡ Pozisyon, geometrik yapı .vs.
  - ➡ Bu sınıf izi alınacak parçacığa ilişkilendirilmiş olmalıdır.
- **G4DynamicParticle** sınıfı;
  - ➡ Momentum, enerji, spin gibi parçacığın dinamik özellikleri saklanır.
  - ➡ Her G4 Track nesnesinin kendi G4DynamicParticle nesnesi vardır.
  - ➡ Tekil (bireysel) olarak parçacığın temsil edildiği sınıftır.
- **G4ParticleDefinition** sınıfı;
  - ➡ Yük, kütle, bozunum kanalı …vs. gibi statik bilgiler saklar.
  - ➡ G4ProcessManager nesnesine ait G4ParticleDefinition nesnesi var.
  - ➡ G4ParticleDefinition tip1 <——> G4DynamicParticle dtip1, dtip2..vs.

Geant4 Terminolojisi (SD)

- Verilen geometride, fizik etkileşimleri olup izler üretilirken ve olaylar ortaya çıkarken detektöre ulaşan sadece hit (entry) bilgisi vardır.
- Kullanıcılar tüm süreçten bilgi alabilmek için 3 yola başvurabilir;
  - Gömülü skorlama komutlarını kullanabilir.
  - İzlenen hacim için skorlama kullanabilir.
  - Detektör read-out (sensörü) ile hitlerin sayımı yapılmalıdır
    - ★ Bu yöntem için **G4VSensitiveDetector** nesnesi bir hacme atanmalıdır.
- Kullanıcı tanımlı (kendi oluşturduğu) sınıf veya yöntemleri de kullanabilir.
  - Bknz. **G4VUserTrackingAction**, **4VUserSteppingAction**

Geant4 Terminolojisi (UNITS)

- Dahili birim sistemleri G4 kodu içinde saklı olarak görünür. Fakat kısaltmalar aşikardır (trivial)
- Birimlendirilmek istenen her sayı ilgili **birim ismiyle çarpılır**.

```
radius = 10.0 * cm;

Mass = 100.0 * GeV;
```

- Dönüşümü istenen birimler için bölme işlemi yapılır.

```
    G4cout << eDep / MeV << " [MeV]" <<
G4endl;
```

- Kullanıcı arayüzünde /units/list komutu ile varsayılan birimler görüntülenebilir.
- Birimler belirtilmezse varsayılan birimlerde çıktı elde edilir.

Geant4 Terminolojisi (COUT / CIN)

- G4 tarafında tanımlı *iostream* nesneleri **G4cout** ve **G4cin** komutları kullanılarak dışa / içe aktarılır.

  ➡ **G4cerr** ve **G4endl** de ihtiyaca göre kullanılabilir.

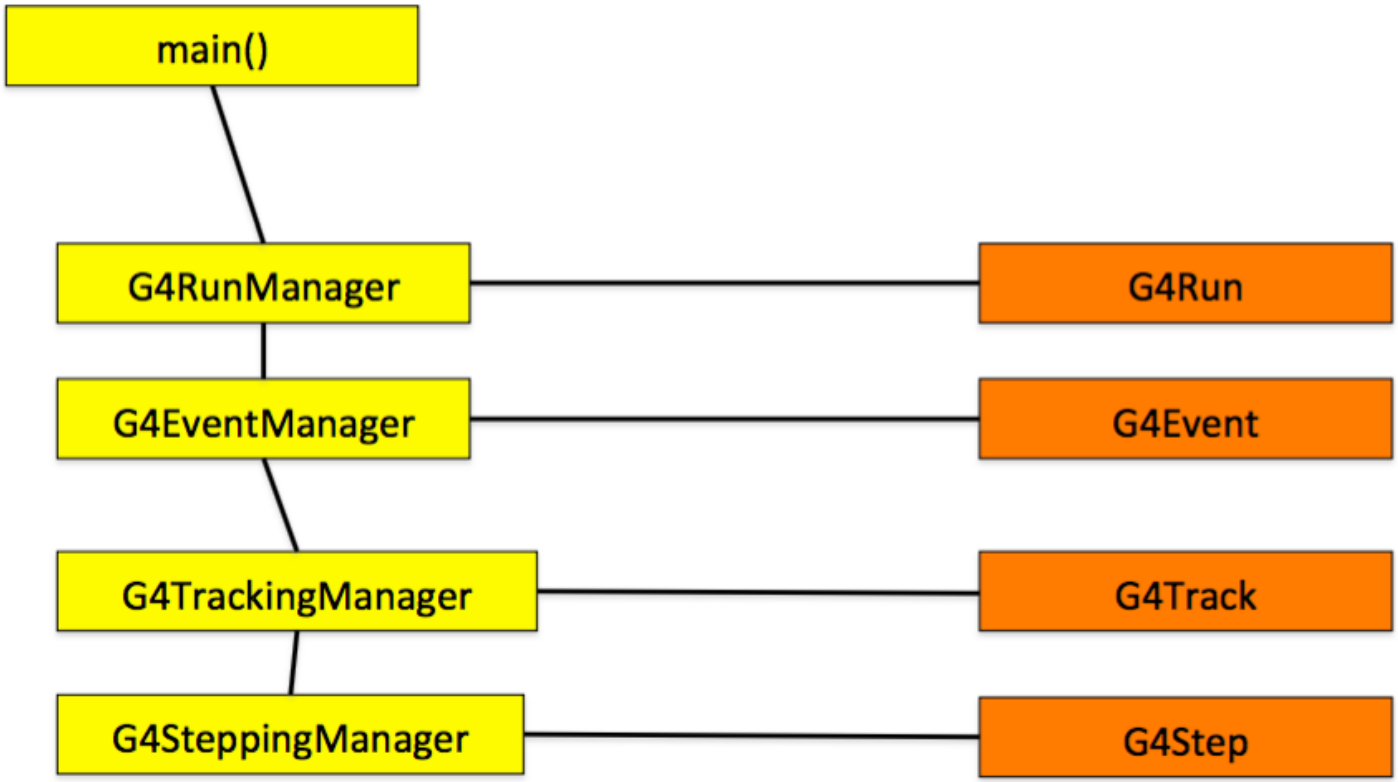
  Örn:

```
G4cout << "Merhaba Bodrum!!" << Gendl;
```

  - Kullanıcıların std::cout / std::cin kullanması G4 kodu içinde **uygun değildir** !!!
  - Dosya işlemleri için I/O bilinen C++ komutları geçerlidir.
- **G4cout** ve **G4cin** komutları özellikleri kaydedilen değerlerin takip edilmesi ve kodun doğrulanması açısından önemli olabilir!!

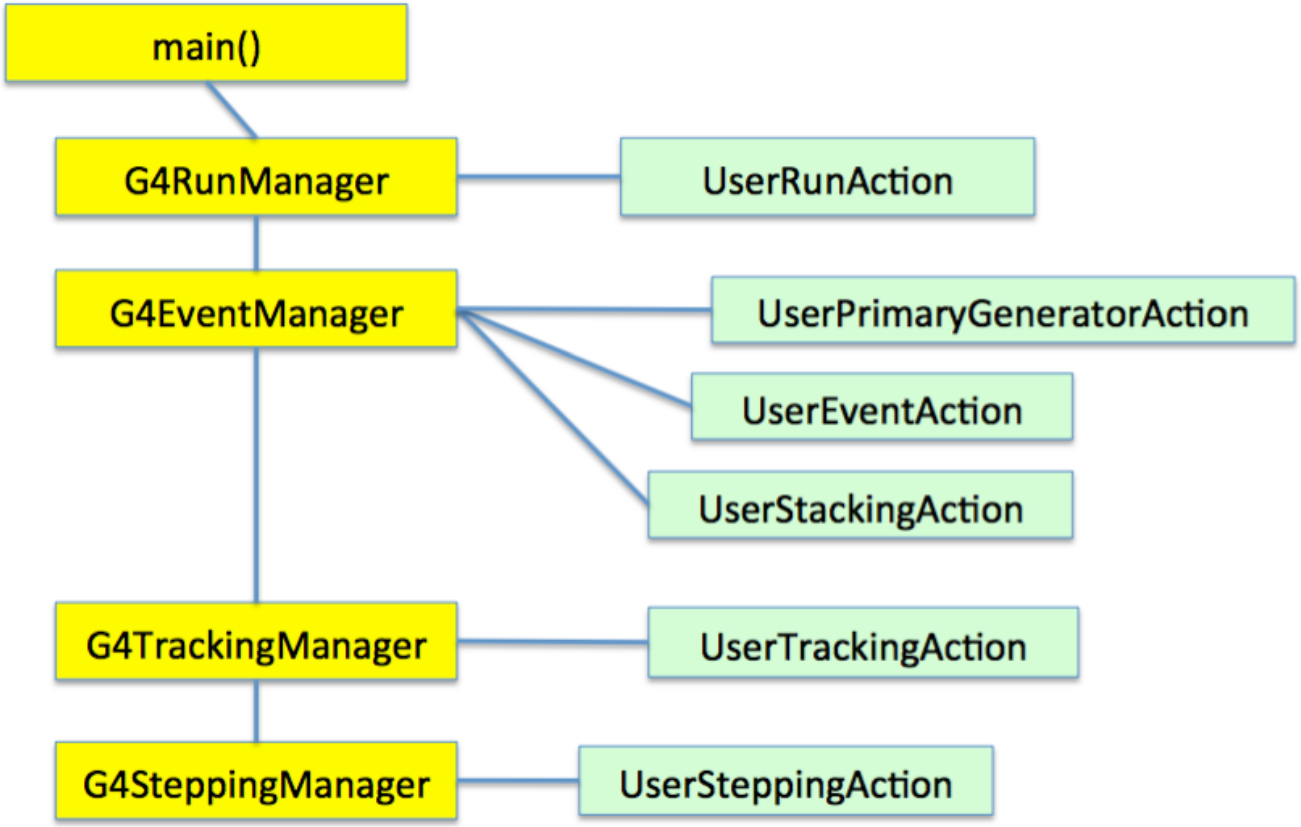
Geant4 Terminolojisi (PROCESS)

- main() çalışması sequantial mod için;



Geant4 Terminolojisi (PROCESS)

- main() çalışması sequantial mod için kullanıcı tanımlı eylemler (action) ile;



Geant4 Terminolojisi (main.cc kodu nasıl olmalı?)

- Tipik bir G4 uygulamasında; int main() içinde ;
  - Geometrik kurulumu tanımlayın;
    - Materyal ve Hacimler
  - Fiziksel kurulumu tanımlayın;
    - Parçacıklar, fiziksel etkileşimler, modeller …vs.
    - Üretim eşik değerleri
  - Olayı ve olay döngüsünü tanımlayın;
    - Öncül izlerin (primary tracks) üretimi
    - Olaylardaki size gerekli bilgileri kaydedin
  - Çıktılarını uyun formatta yazdırın
  - Görselleştirme işlemleri (opsiyonel)

Sonu ....Messenger (Örn:G4DetectorMessenger) ile biten tüm sınıflar ve kod dosyaları, makro kartlarında yer alacak komutların tanımlandığı kodlardır.

Yani macro komutları ile C++ komutları arasındaki bağlantı ...Messenger kodlarında tanımlanır.

Kısım 2:
Geometrik Modelleme

Geometri Oluşturma

- Geometriler, oluşturacağımız SANAL DENEYlerin başlangıcıdır.
- Geometri oluşturma; deneyde kullanılacak **materyallerin ve yerlerinin** tanımlanması işlemidir.
- Benzetimin **her adımında izlenen yol üzerindeki** materyaller (parçacık) kontrol edilir / ötelenebilir / etkileşime girebilir.
- Basit geometriler (kutu, silindir, küre) <u>işlemleme süresini kısaltır</u>.
- Ayrıntılı benzetimlerde karmaşık/gelişmiş geometriler kullanılabilir.
- İlk geometri tanımları DETEKTÖR inşaasında karşımıza çıkar.

<u>Not:</u> Gelişmiş Geometrilerden sonraki sunumda bahsedilecektir.

Geometri Oluşturma

- Tanımlara geometrik şeklin boyutlarını belirleyerek başlayın:

Box 5m x 2m x 10m
Sphere r=10m
Cylinder r = 5m, h = 10 m …

- Hangi özelliklere sahip olmalı?

Material
Any B- or E-fields?
Is it sensitive, i.e. used for read-out?

- Hangi Konumda ?

Single placement?
Or repeatedly as described by some function?

Solids

↓

Logical Volumes

↓

Physical Volumes

Geometri Oluşturma

- Solid (katı) geometrilerden oluşan tanımlamalardır

Kutu, küre, silindir tanımlamalarından karmaşık geometriler de elde edilebilir !!

- Hazır fonksiyonlardır: G4Box, G4Cone, …vs.
- Boyut argümanlarına ihtiyaç duyarlar

Boyutlar birimleri (cm, mm ?) ile birlilkte verilmelidir !!

Solids

↓

Logical Volumes

↓

Physical Volumes

Geometri Oluşturma

Aşağıdaki sıra ile tanımlanırlar:

- Katı geometriye bir materyal eklenir
- Katı geometriye E veya B alanları eklenir
- Katı geometri hassas (sensitive) yapılır
- Kullanıcı limitleri belirlenir
- Fiziksel kız hacimler belirlenir.
  (physical daughter volumes)

Solids

↓

Logical Volumes

↓

Physical Volumes

Geometri Oluşturma

Aşağıdaki sıra ile tanımlanırlar:

- Hacmin konumu ve rotasyonu belirtilir.
- Aynı hacmin replikası veya tekrarı belirtilir.

Solids

↓

Logical Volumes

↓

Physical Volumes

Örnek:

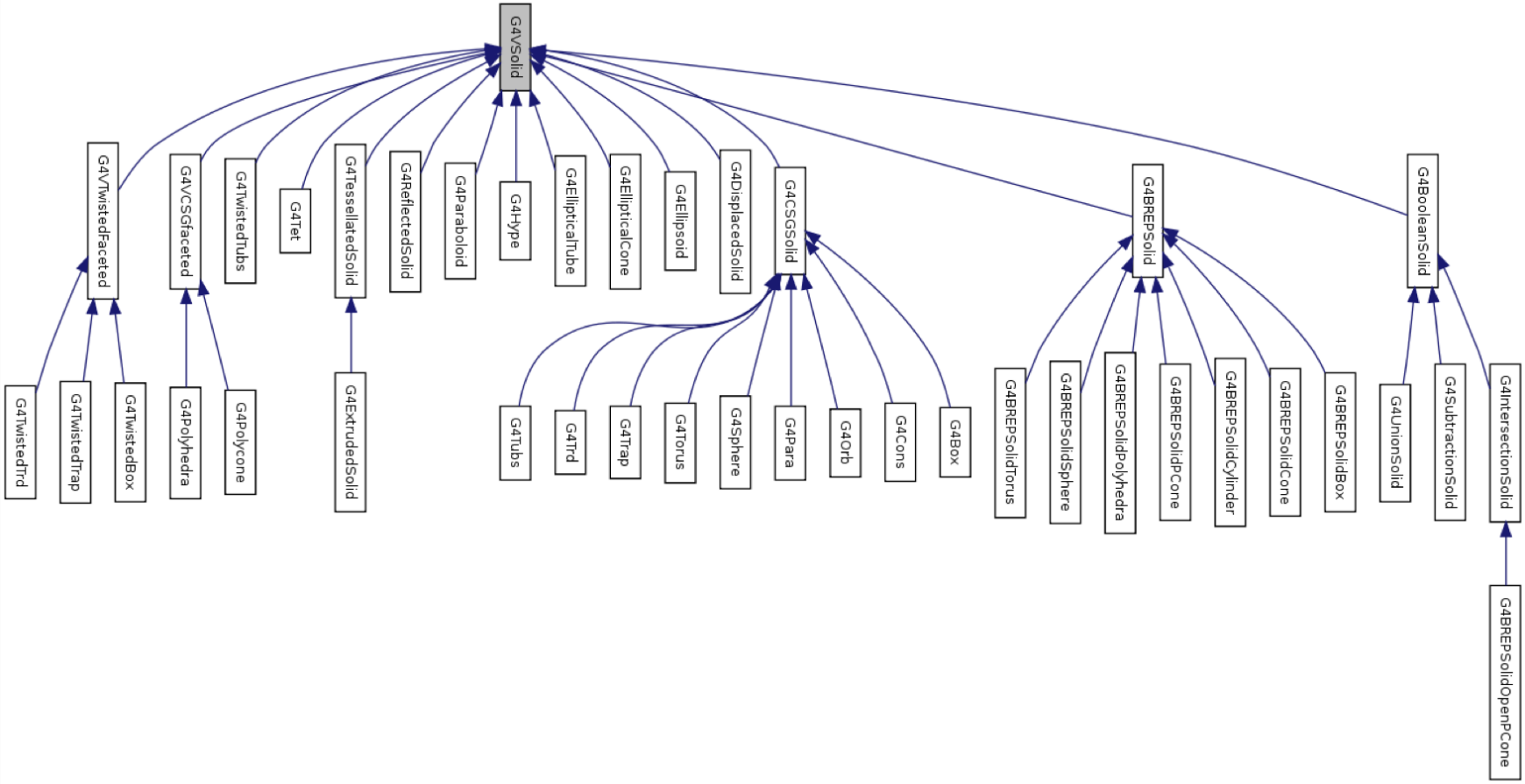
```
G4Box* boxSolid = new G4Box("aBox", 1.0*m, 1.0*m, 1.0*m);
```

```
G4LogicalVolume* boxLogic = new G4LogicalVolume(BoxSolid, BoxMaterial,
"Box1Logic")
```

```
G4ThreeVector pos(1*m, 1*cm, 1*mm);
G4RotationMatrix* rot = 0;
G4VPhysicalVolume* boxPhys = new G4PVPlacement(rot, 0, BoxLogic, "Box1Placed",
World, pMany = 0, copyNo = 0, surfChk = true);
```

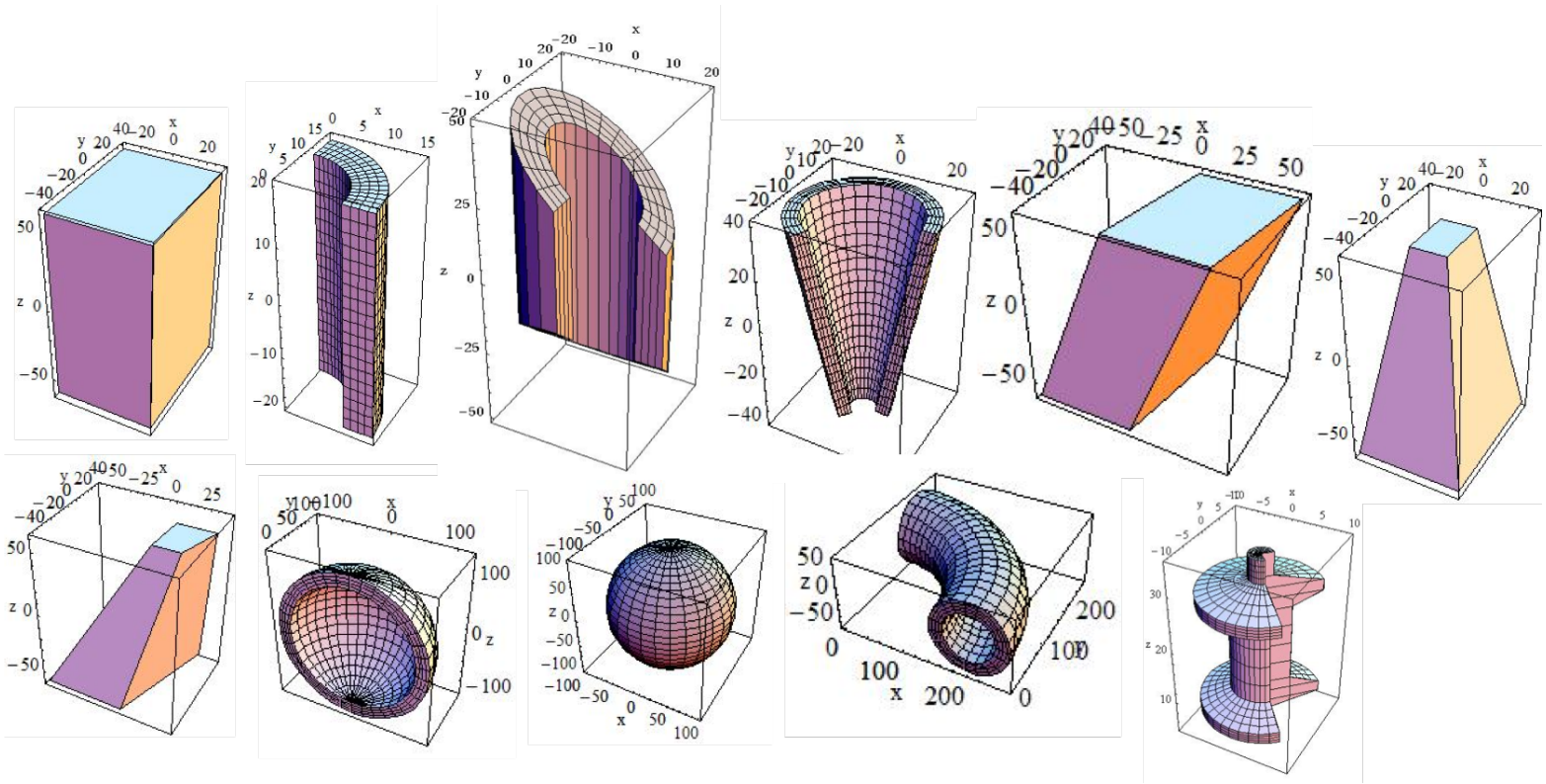
Katı Geometriler

- Tüm katı şekiller G4VSolid sınıfıyla türetilir / oluşturulur.
- Tanımlanması bağlantılı bazı fonksiyonları kullanır hale getirir



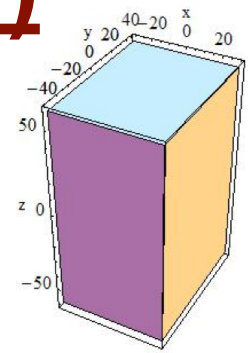
Katı Geometriler

- Constructed Solid Geometries (CSG)
- Boundary Represented Shapes (BREPs)
- Complex shapes by parameterization
- Boolean operations also possible

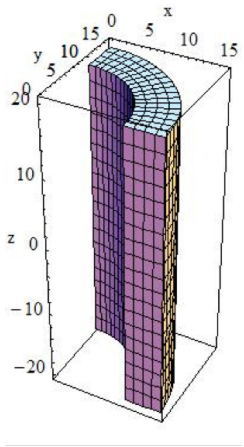


Katı Geometriler (Örnekler)

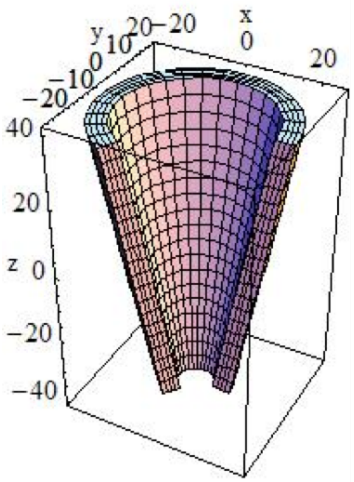
```
G4VSolid* boxSolid = new G4Box("aBox", 1.0*m, 1.0*m,
1.0*m);
```



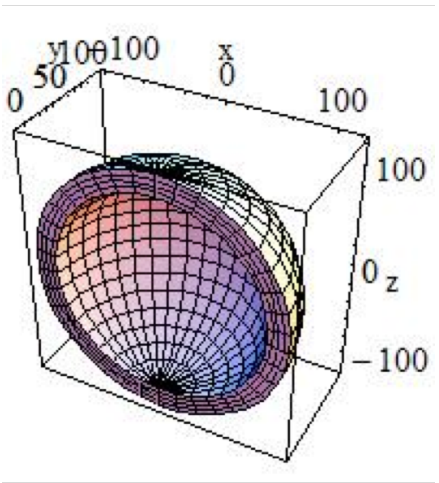
```
G4VSolid* tubeSolid = new G4Tubs("aTube",
                1.0*m, //inner radius (0 possible)
                2.0*m, //outer radius
                4.0*m  // half – height
                0.*deg, 360.*deg); //segment angles
```



```
G4VSolid* coneSolid = new G4Cons("aCone",
                1.0*m, 1.5*m, //inner/outer radius 1
                2.0*m, 2.5*m, //inner/outer radius 2
                4.0*m  // half – height
                0.*deg, 360.*deg); //segment angles
```



```
G4VSolid* sphereSolid = new G4Sphere("aSphere",
                1.0*m, 1.5*m //inner/outer radius
                0.*deg, 360.*deg, // phi
                0.*deg, 180.*deg); // theta
```



Mantıksal Hacimler (Logical Volumes)

- Katı Geometriler (solids) malzemelerin sadece boyutunu ve şeklini belirt
  Oysa bir deneyde sadece bu parametreleri bilmek yetmez.
- Hacimlere aşağıdaki özellikler verilebilir;

  i. Materyal cinsi

  ii. E / B Alanlar

  iii. Görselleştirme Atıfları

  iv. Hassasiyet

  v. Kız hacimlerin pozisyonları

  vi. Bölgeler / Lezyonlar
- Mantıksal Hacimler pozisyon ve rotasyonları tanımlamaz (Fiziksel nicelikler;; fiziksel hacim içinde tanımlanır.)
- Mantıksal Hacimler aynı tipteki fiziksel hacimler arasında paylaştırılabilir

Mantıksal Hacimler (Logical Volumes)

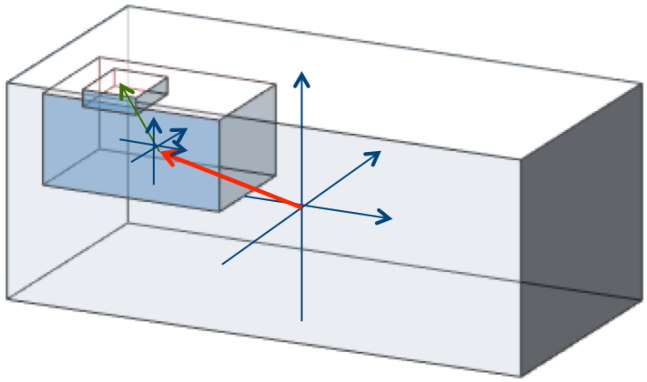
- Tanımlandıktan sonra mantıksal hacimlerin kaydı yapılmış olur.
- Kullanılması için başlatılması, fiziksel / katı bir hacime paylaştırılması gereklidir.
- Kullanılmazlarsa tanımlanmaları hatalı olsa bile program çalışır.
  (Derleme sırasında warning satırları gelir)

Örnek:

```
G4LogicalVolume(G4VSolid* pSolid,        //pointer to solid, must to be 0
                G4Material* pMat,        //pointer to material, must not be 0
                const G4String& name,    //name of this logical volume
                G4FieldManager* pFieldMgr = 0, //pointer to field manager
                G4VSensitiveDetector* pSensDet = 0, //sensitive detector
                G4UserLimits* pUserLimits = 0,  // user limits
                G4bool optimise = true);        // optimization on
```

Fiziksel Hacimler (Physical Volumes)

- Oluşturulan katı ve mantıksal hacimlerin yerleşimi veya hareketi fiziksel hacim ile tanımlanır.
- Geant4 benzetimleri hacim geometrilerinin hiyerarşik yapısıyla oluşturulur:
  - En üst: Kök Hacim (veya World olarak da isimlendirilir)
  - İçeride: Ana hacimler (Tüm diğer kız hacimler Ana hacmin koordinat eksenini baz alırlar)
  - Daha içeride: Kız hacimler (Aynı ana hacme bağlı birden fazla olabilir)
- Fiziksel Hacimler **G4VPhysicalVolume** sınıfıyla üretilirler.
- Diğer hacimler gerçek fiziksel hacmi temsil ediyorsa **G4PVPlacement** kullanılır.



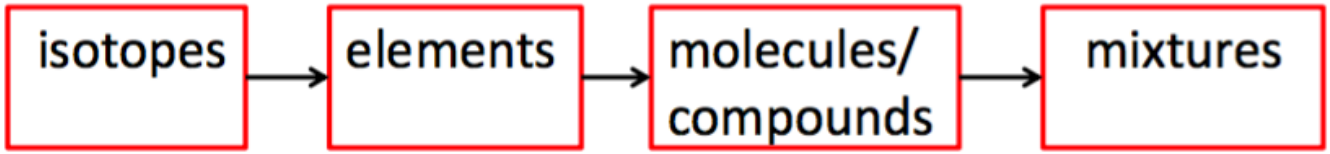
Fiziksel Hacimler (Örnek)

```
G4VSolid* boxSolid= new G4Box("aBox", 1.0*m, 1.0*m, 1.0*m);

G4LogicalVolume* boxLogic = new G4LogicalVolume(boxSolid, boxMat, "logicBox");

G4RotationMatrix* xRot = new G4RotationMatrix();
xRot->rotateX(M_PI/4*rad);

G4ThreeVector yTrans(0., 1.*m, 0.);

//constructor 1
G4VPhysicalVolume* boxPhys = new G4PVPlacement(xRot, yTrans, boxLogic,
"physicBox", motherLog, 0, copyNo, true);

//constructor 2
G4VPhysicalVolume* boxPhys = new G4PVPlacement(G4Transform3D(xRot, yTrans),
boxLogic, "physicBox", motherLog, 0, copyNo, true);

//constructor 3
G4VPhysicalVolume* boxPhys = new G4PVPlacement(xRot, yTrans, boxLogic,
"physicBox", motherPhys, 0, copyNo, true);
```

Kısım 3:
Materyal ve Nesne Yönetimi

Materyal Nesneleri ve Yönetimi

- G4 materyalleri tanım olarak şunlar olabilir: İzotoplar, Elementler, Bileşikler (Compounds-Molecules), Alaşımlar (Alloys)



- Fiziksel durumları: Katı, sıvı veya gaz olabilir. (Plazma yok!)
- Çeşitli değerlerdeki basınç, sıcaklık veya yoğunluk durumlarında olabilirler.
- Şekildeki sırasıyla **G4Isotope, G4Element** ve **G4Material** sınıfları kullanılı
- Karışımlar / Alaşımlar bu 3 sınıfın nesneleri kullanılarak oluşturulabilir.

Materyal Nesneleri ve Yönetimi

- Kullanıcılar kendi element (tümü olduğu halde) veya materyallerini yaratabilirler.
- G4 girilen bir materyal için sadece **yoğunluk (density)** bilgisini zorunlu tutmaktadır.
- Geri kalan bilgiler opsiyoneldir: fiziksel durum, sıcaklık, basınç (Aksi belirtilmediği sürece Standart koşullar geçerlidir.)
- Bazı anormal durumlar tanımlıdır: Standart Koşul dışındaki gazlar, yüksek basınçta katılar, çok düşük yoğunluklu sıvılar …vs.
- NIST Materyal veritabanı kullanılabilir. Önemli bir kolaylıktır!

- Tüm tanımlı materyal listesi: http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/ForApplicationDeveloper/html/Appendix/materialNames.html

Materyal Nesneleri ve Yönetimi

G4 ile elementlerin kod içi tanımlanması:

- Materyalin atom sayısı, kütle numarası, yoğunluğu verilmelidir.

```
G4double density = 4.506*g/cm3;
G4double a = 47.867*g/mole;
G4Material* ti = new G4Material("pureTitanium", z=22, a, density);
```

- Vakum tanımlanması veya herhangi bir gazın ortama yerleştirilmesi:

```
G4NISTManager* manager = G4NISTManager::GetPointer;
G4Material* vacuum = manager->FindOrBuildMaterial("G4_Galactic");
```

Materyal Nesneleri ve Yönetimi

G4 ile moleküllerin kod içi tanımlanması:

- Molekül atom sayısı, kütle numarası, yoğunluğu ve yapısı verilmelidir.

```
G4double a = 1.01*g/mole;
G4Element* elH = new G4Element("Hydrogen", "H", z=1, a);
a = 16.00*g/mole;
G4Element* elO =  new G4Element("Oxygen", "O", z=8, a);

G4double density = 1.0*g/cm3;
G4int ncomp = 2;
G4Material* H2O = new G4Material("Water", density, ncomp);
G4int nAtoms;
H2O->AddElement(elH, nAtoms=2);
H2O->AddElement(elO, nAtoms=1);
```

Materyal Nesneleri ve Yönetimi

G4 ile alaşım / karışımların kod içi tanımlanması:

- Alaşımdaki elementlerin atom sayısı, kütle numarası, yoğunluğu ve alaşımın yapısı verilmelidir.

```
G4Element* elC = ... ;        // define carbon
G4Material* H2O = ... ;       // define molecule (previous page)
G4Material* SiO2 = ... ;      // define another molecule

G4double density = 0.20*g/cm3;
G4int ncomp = 3;
G4double fracMass;
G4Material* Aerog = new G4Material("Aerogel", density, ncomp);

Aerog->AddMaterial(SiO2, fracMass = 62.5*perCent);
Aerog->AddMaterial(H2O, fracMass = 37.4*perCent);
Aerog->AddElement(elC, fracMass = 0.1*perCent);
```

Materyal Nesneleri ve Yönetimi

G4 ile izotopların kod içi tanımlanması:

- Aslında tanımlanan elementler de doğal birer izotoptur. Yoğunluk değerleri farklı girilse bile doğal çokluğuyla (abundance) kabul edilir.
- Doğal olmayan elementler ve farklı değerlerin girilmesi;

```
G4int z; G4int a;
G4Isotope* u235 = new G4Isotope("U235", z=92, a=235.,
    235.044*g/mole);
G4Isotope* u238 = new G4Isotope("U238", z=92, a=238.,
    238.051*g/mole);
```

G4Isotope sınıfıyla hiç olmadıkları kabul edilerek türetildiler !

Materyal Nesneleri ve Yönetimi

G4 ile bileşiklerin kod içi tanımlanması:

- Örnek: Nükleer reaktör için uranyum yakıtı (fuel)

```
G4int ncomp;
G4double abundance;
G4Element* enrichedU = new G4Element("enrichedU", ncomp=2);
enrichedU->AddIsotope(u235, abundance=5.0*perCent);
enrichedU->AddIsotope(u238, abundance=95.0*perCent);

G4Element* elF = new G4Element("Fluorine", "F", 9., 18.998*g/mole);

G4double density;
G4Material* fuel = new G4Material("NuclearFuel", density = 5.09*g/cm3,
                        ncomp=2, kStateGas,  640*kelvin, 1.5e7*pascal) ;
fuel->AddElement(elF, 6);
fuel->AddElement(enrichedU, 1);
```

Kısım 4:
Fiziksel Etkileşimler

Fizik Listesi

- **Fizik listeleri;** İki temel tanımlamadan sorumludur;

  1. Benzetimdeki tüm Parçacıklar **2.** Parçacıkların dahil olduğu süreçler/ etkileşimler

- Fizik listeleri ile ilgili en az 3 zorunlu sınıf tanımlamasında yapılmalıdır.

- **G4RunManager** sınıfında uygulamanın en başında fizik listelerinin nasıl ele alınacağı tanımlanmalıdır.

- **Kullanıcılar, benzetimin hangi fiziksel etkileşimleri içerdiği konusunda bilgi sahibi olmalıdır. Aksi takdirde; benzetimler GERÇEKÇİ olamaz** !!

- "Fizik fiziktir. Geant4, standart olarak tüm fizik etkileşmelerini içermelidir" görüşü neden yanlıştır?

  1. Tüm etkileşmeler için **hesaplama zamanı ve performans** sorunu vardır.
  2. Etkileşmeler için yapılabilecek **farklı yaklaşımlar ve kısıtlamalar** vardır.
  3. Benzetimlerin yapılma amacında **belli etkileşimler** baskındır/önemlidir.

Fizik Listeleri

Geant4 içeriğinde yer alan fiziksel süreçler:

- **EM physics**
  – “standard” processes valid from ~ 1 keV to ~PeV
  – “low energy” valid from 250 eV to ~PeV
  – optical photons

- **Weak interaction physics**
  – decay of subatomic particles
  – radioactive decay of nuclei

- **Hadronic physics**
  – pure strong interaction physics valid from 0 to ~TeV
  – electro- and gamma-nuclear valid from 10 MeV to ~TeV

Fizik Listesi (G4VUserPhysicsList sınıfı)

- Tüm fizik listeleri bu sınıf yoluyla türetilebilir.
  – daha sonra runManager objesine kaydedilir. (registering)
- Örnek

```
class MyPhysicsList: public G4VUserPhysicsList {
    public:
        MyPhysicsList();
        ~MyPhysicsList();
        void ConstructParticle();
        void ConstructProcess();
        void SetCuts();
}
```

- Kullanıcılar şu 3 methotu zorunlu olarak tanımlamalıdır:

ConstructParticle, ConstructProcess, SetCuts

Fizik Listesi (G4VUserPhysicsList sınıfı)

Tanımlanması gereken metodlar:

ConstructParticle() – Simülasyonda beklenen tüm parçacık tanımlarını bu metod ile yapınız. Örn:

```
        exPhysicsList::exPhysicsList() :
G4VModularPhysicsList(){
                 particleList = new G4DecayPhysics();}
        void exPhysicsList::ConstructParticle(){
                 particleList->ConstructParticle();}
```

ConstructProcess() – Her bir parçacık için, tüm fiziksel süreçleri bu metodlar içinde tanımlayınız. Örn:

```
       void exPhysicsList::ConstructProcess(){
                 particleList->ConstructProcess();}
```

SetParticleCuts() – Menzil kesilimlerini özellikle ikincil parçacık üretimleri için bu metod içinde tanımlayınız. Örn:

```
       void exPhysicsList::SetCutForElectron(G4double cut){
                 cutForElectron = cut;
                 SetParticleCuts(cutForElectron,
G4Electron::Electron());}
```

Fizik Listesi (G4VModularPhysicsList sınıfı)

- **Geant4** kurulumuyla gelen hazır fizik listelerinin eklenmesini sağlar.
- Daha basit ve güvenilir bir yöntemdir.
- Dezavantajları; benzetim amacına uygun olmayan etkileşimlerin de dahil olması, hata alma durumunda düzeltmenin zorluğu, optimizasyon zorluğu
- G4VModularPhysicsList özellikleri;
  - G4UserPhysicsList sınıfından türetilir.
  - AddTransportation() metodu otomatik tüm kayıtlı parçacıkları çağırır
  - Fizik modülleri tanımlanabilir: EM fiziği, Hadronik fizik, Optik ….vs.

Fizik Listesi (G4VModularPhysicsList sınıfı)

Constructor:

```
MyModPhysList::MyModPhysList(): G4VModularPhysicsList() {

      defaultCutValue = 0.7*mm;
      RegisterPhysics(new ProtonPhysics() );

// all physics processes having to do with protons

      RegisterPhysics(new ElectronPhysics() );

// all physics processes having to do with electrons

      RegisterPhysics(new DecayPhysics() );

// physics of unstable particles

}
```

SetCuts:

```
void MyModPhysList::SetCuts() { SetCutsWithDefault(); }
```

Fizik Listesi (Paketler)

- There are currently 19 packaged physics lists available
  - but you will likely be interested in only a few, namely the "reference" physics lists
  - many physics lists are either developmental or customized in some way, and so not very useful to new users
- All but one of the packaged physics lists use templates
  - the LBE physics list is the old-style "flat" list without templates or physics builders
- 6 reference physics lists:
  • FTFP_BERT, FTFP_BERT_HP
  • QGSP_BERT, QGSP_BERT_HP, QGSP_BIC • QGSP_FTFP_BERT

Fizik Listesi (Paket İsim Kısaltmaları)

- The following acronyms refer to various hadronic options

- QGS -> Quark Gluon String model (>~20 GeV)
- FTF->Fritiofstringmodel(>~5GeV)
- BIC -> Binary Cascade (<~ 10 GeV)
- BERT -> Bertini-style cascade (<~ 10 GeV)
- HP -> High Precision neutron model (< 20 MeV)

- P -> G4Precompound model used for de-excitation

- EM options designated by

- no suffix : standard EM physics
- EMV suffix : older but faster EM processes
- other suffixes for other EM options

Fizik Listesi (Referans Fizik Listeleri)

- FTFP_BERT
  - recommended by Geant4 for HEP
  - contains all standard EM processes
  - uses Bertini-style cascade for hadrons < 5 GeV
  - uses FTF (Fritiof) model for high energies (> 4 GeV)
- QGSP_BERT
  - all standard EM processes
  - Bertini-style cascade up to 9.9 GeV
  - QGS model for high energies (> ~18 GeV) – FTF in between
- QGSP_BIC
  - same as QGSP_BERT, but replaces Bertini cascade with Binary cascade and G4Precompound model
  - recommended for use at energies below 200 MeV (many medical applications)

Fizik Listesi (Referans Fizik Listeleri)

- FTFP_BERT_HP
  - same as FTFP_BERT, but with high precision neutron model used for neutrons below 20 MeV
  - significantly slower than FTFP_BERT when full thermal cross sections used (there's an option to turn this off)
  - for radiation protection and shielding applications

- Shielding
  - based on FTFP_BERT_HP with improved neutron cross sections from JENDL
  - bejer ion interactions using QMD model
  - currently used by SuperCDMS dark matter search
  - recommended for: – shielding applications – space physics – HEP

Fizik Listesi (Referans Fizik Listeleri)

- FTFP_INCLXX, FTFP_INCLXX_HP
  - like FTFP_BERT, but with BERT replaced by INCL++ cascade model

- QBBC
  - uses both BERT and BIC cascade models
    - latest coherent elastic scattering
  - neutronXS models (faster CPU-wise)

- QGSP_BIC_HP
  - same as QGSP_BIC, but with high precision neutron model used for neutrons below 20 MeV
  - recommended for radiation protection, shielding and medical applications

Fizik Listesi (Referans Fizik Listeleri)

Ne zaman, hangi fizik listesi kullanmalı ??

- Eğer ana parçacık enerjiniz 5 GeV altındaysa; (Örn: klinik proton ışını ~

  150 MeV olabilir)
  - BIC veya BERT içeren bir fizik listesi tercih ediniz.
  - Örn: QGSP_BIC, QGSP_BERT, FTFP_BERT ….vs.

- Eğer “neutron transportation” önemliyse;
  – HP kısaltması içeren bir fizik listesi seçin
  – Örn: QGSP_HP, QGSP_HP, FTFP_HP ….vs.

- Bragg Eğrileri elde etmek için EMV veya EMX kısaltmalı fizik listelerini seçin.
- Radyoaktif bozunum, Optik foton transport, detaylı spektrumlar için LBE kısaltmalı fizik listelerini seçin.

Örnek

Örnek: exampleB1.cc dosyasında tanımlanan fizik listesi objeleri ve diğerleri

```
int main(int argc,char** argv)
{
  // Detect interactive mode (if no arguments) and define UI sessi
  //
  G4UIExecutive* ui = 0;
  if ( argc == 1 ) {
    ui = new G4UIExecutive(argc, argv);
  }

  // Choose the Random engine
  G4Random::setTheEngine(new CLHEP::RanecuEngine);

  // Construct the default run manager
  //
#ifdef G4MULTITHREADED
  G4MTRunManager* runManager = new G4MTRunManager;
#else
  G4RunManager* runManager = new G4RunManager;
#endif

  // Set mandatory initialization classes
  // Detector construction
  runManager->SetUserInitialization(new B1DetectorConstruction());
  // Physics list
  G4VModularPhysicsList* physicsList = new QBBC;
  physicsList->SetVerboseLevel(1);
  runManager->SetUserInitialization(physicsList);
  // User action initialization
  runManager->SetUserInitialization(new B1ActionInitialization());
  // Initialize visualization
  G4VisManager* visManager = new G4VisExecutive;
  // G4VisExecutive can take a verbosity argument - see /vis/verbo
guidance.
  // G4VisManager* visManager = new G4VisExecutive("Quiet");
  visManager->Initialize();
  // Get the pointer to the User Interface manager
  G4UImanager* UImanager = G4UImanager::GetUIpointer();
```

Teşekkürler….