# Neden ROOT?
# YEF Tarzı Düşünme

V. Erkcan Özcan
*University College London*


HPFBU Okulu, Adana, 30 Ocak 2009

# Amaç

- C++ ve ROOT'un temel kavramlarını öğrendik.
  - Grafik arayüzü, betik yazma, ağaç yapısı…
  - Pekiyi ROOT'u nasıl kullanıyoruz? Elinize bir başkasının hazırladığı bir ROOT dosyası çıkınca, ondan ne beklersiniz?

# C ve C++'da Diziler

```
root [0] float cdizisi[5] = { -1.2, 2.3, 0.1, 15.7, 2.72 } ;
root [1] cout << cdizisi[4] << endl;
2.72
root [2] cdizisi[0] = cdizisi[1] + cdizisi[2] ;
root [3] cout << cdizisi[0] << endl;
2.4
```

```
root [0] #include<vector>
root [1] std::vector<float> cArtiArtiVektoru ;
root [2] cArtiArtiVektoru.push_back(23.1);
root [3] cArtiArtiVektoru.push_back(12.7);
root [4] cArtiArtiVektoru.push_back(-1.7);
root [5] cout << cArtiArtiVektoru.size() << endl;
3
root [6] cout << cArtiArtiVektoru[1] << endl;
12.7
```

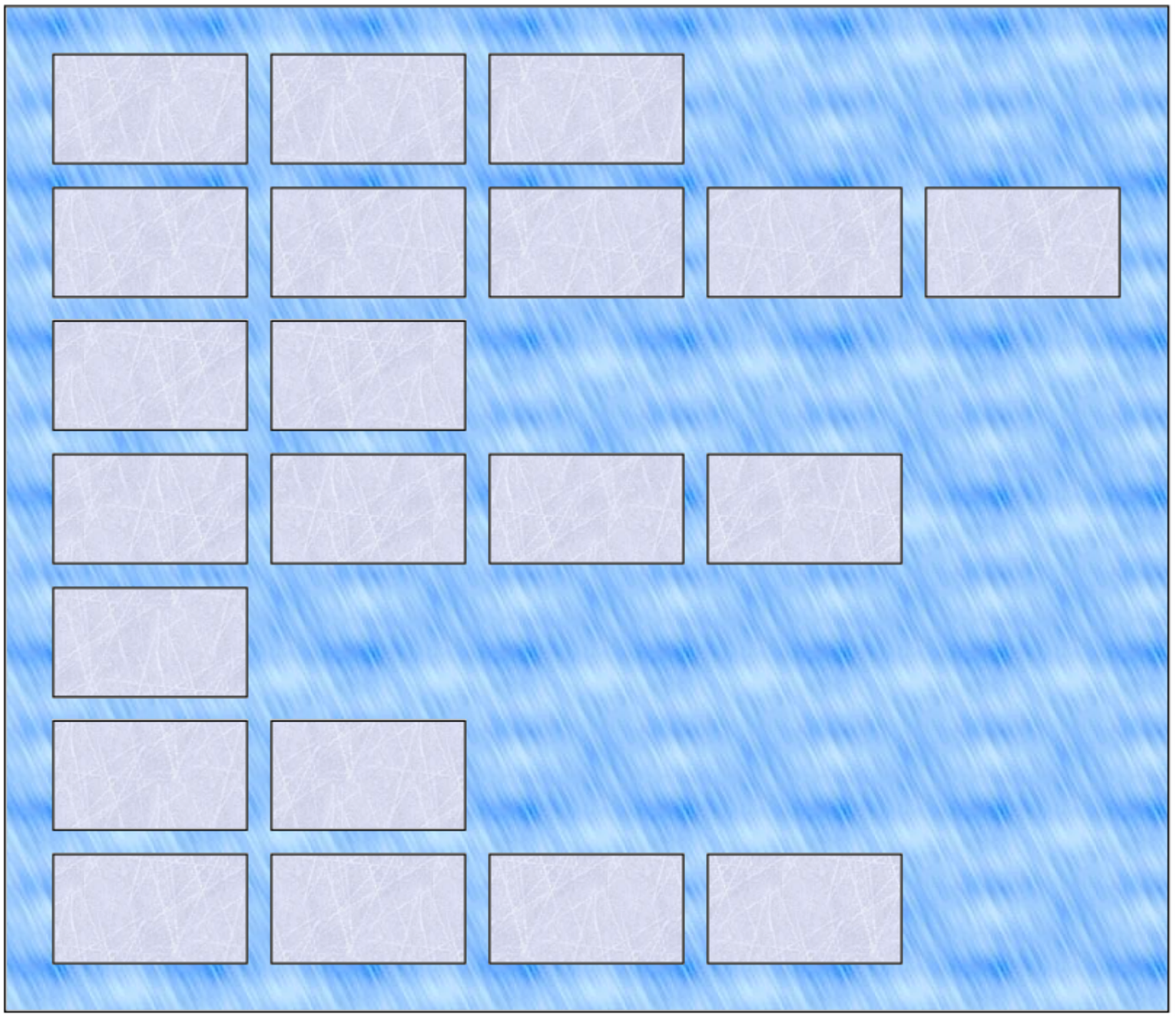
- C++'da tavsiye edilen metod STL class'larını kullanmaktır.

# Ntuple Nedir?

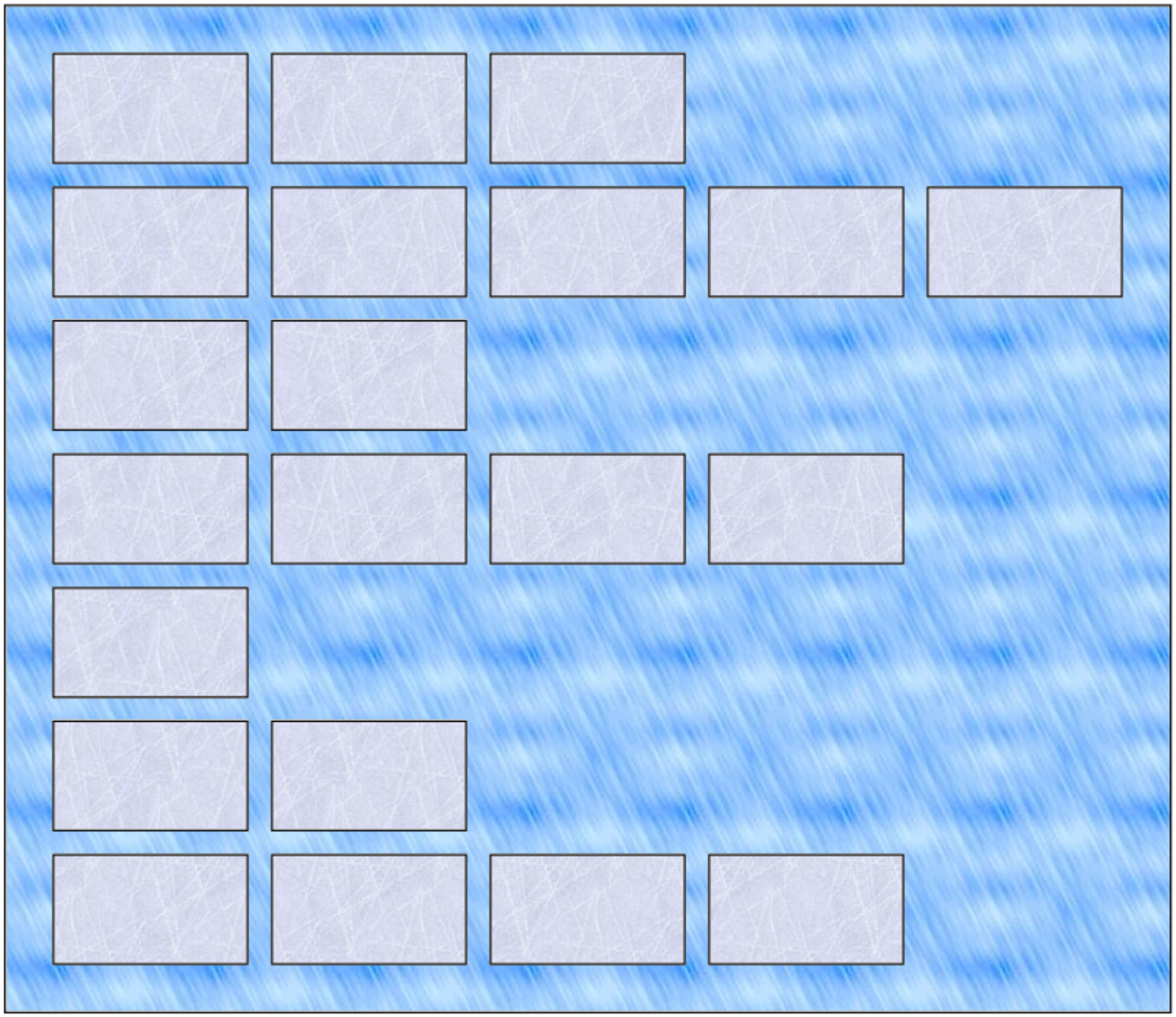
| | | |
|---|---|---|
| 496.748 | 498.62 | -4.54E-03 |
| 496.504 | 501.483 | -3.09E-02 |
| 497.019 | 457.184 | -1.70E-02 |
| 496.947 | 498.395 | -6.54E-03 |
| 499.831 | 500.914 | -1.07E-02 |
| 410.338 | 499.397 | -1.10E-02 |
| 497.194 | 429.197 | 7.89E-03 |
| 497.254 | 440.405 | -4.62E-03 |
| 499.506 | 500.951 | -9.64E-03 |
| 498.133 | 501.397 | -1.63E-02 |
| 496.696 | 499.5 | -1.63E-02 |
| 497.562 | 496.822 | 1.13E-02 |
| 500.659 | 498.569 | 1.31E-02 |
| 498.334 | 501.478 | -2.11E-02 |
| 499.619 | 500.319 | 8.70E-03 |
| 501.201 | 499.568 | 5.35E-03 |
| 498.229 | 500.383 | -1.05E-02 |
| 498.043 | 481.825 | 1.94E-02 |
| 500.057 | 501.24 | -1.89E-02 |
| 449.289 | 497.142 | -1.33E-02 |
| 496.761 | 496.953 | 4.73E-03 |
| 491.794 | 474.876 | -1.13E-05 |
| 500.099 | 481.557 | -4.88E-04 |

- En basit haliyle kolon kolon sayılar, "spreadsheet" gibi.
- Calypso'da ürettiğimiz e+e- olaylarında: birinci ve ikinci ışınların enerjileri ve çarpışma noktasının z-koordinatı.

# Veri Kaydı Üzerine

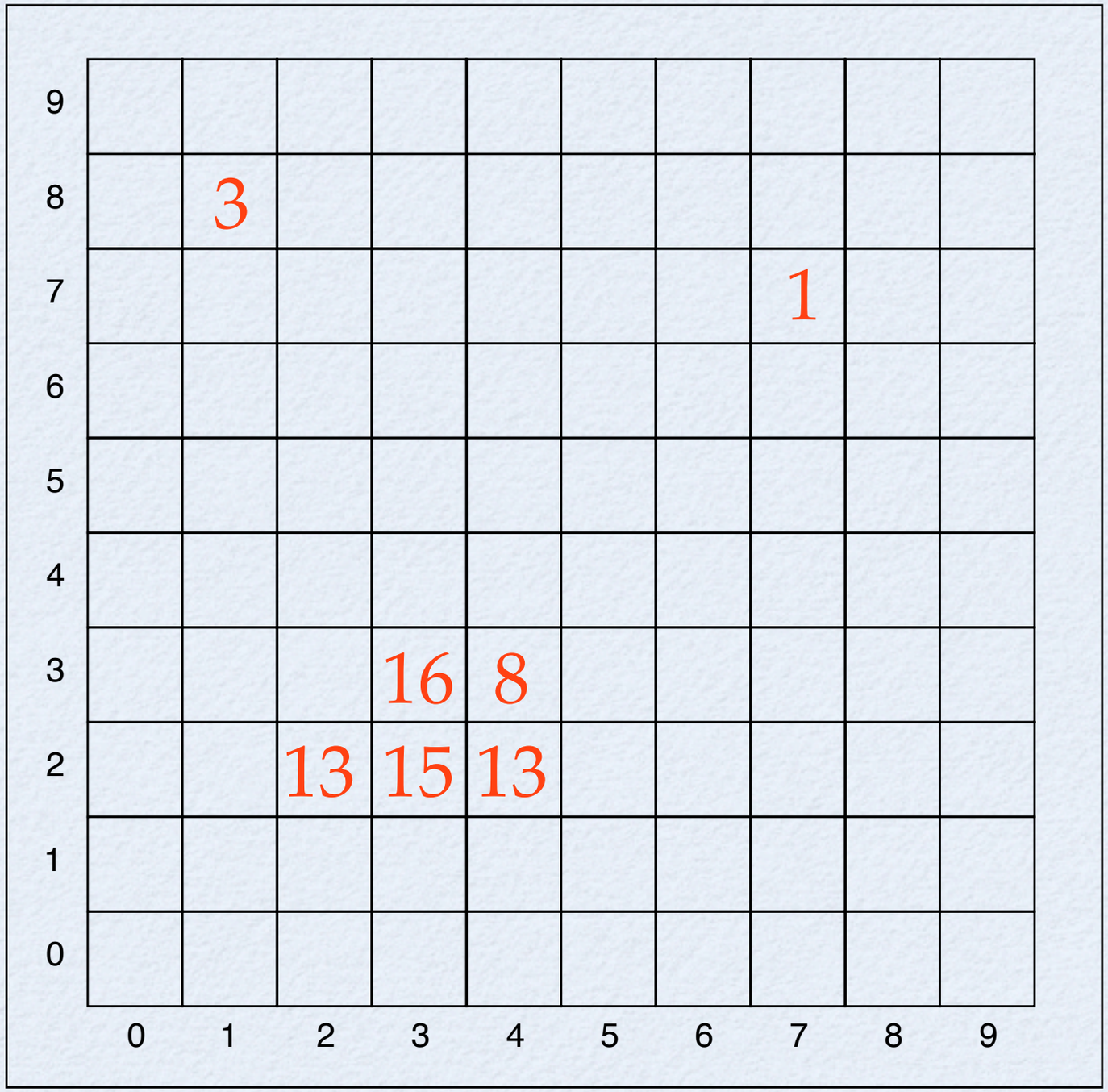


# Veri Kaydı Üzerine



- Eğer ölçümler farklı sayılarda veri girişine sebep oluyorsa, o zaman indeksleme yapabiliriz.
- Ayrı bir değişken kaç tane veri olduğunu tutar.

# Gerçek Bir Uyguluma




- Bir 10×10 silikon piksel algıcımız olsun. Bunun kalibrasyonu için üzerine sabit enerjili bir demet yolluyoruz. Demet aynı anda birden fazla pikseli aydınlatıyor. Ölçüm alıyoruz, sinyal veren piksellerin x-y koordinatlarını ve sinyalin büyüklüğünü birer dizide kaydediyoruz.

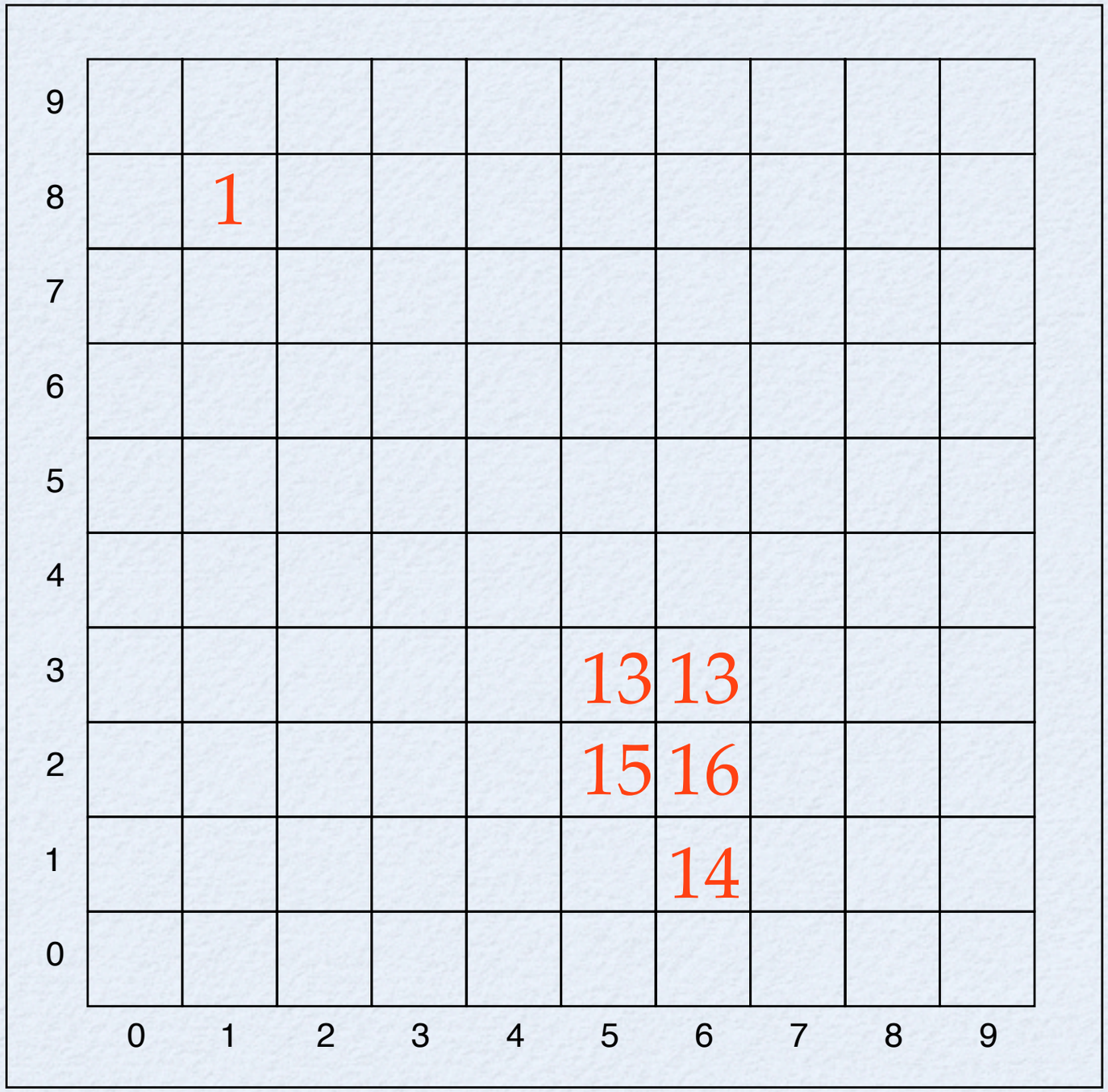
Ölçüm 1

x[7]={2,3,4,3,4,1,7}

y[7]={2,2,2,3,3,8,7}

e[7]={13,15,13,16,8,3,1}

# Gerçek Bir Uyguluma




- Bir 10×10 silikon piksel algıcımız olsun. Bunun kalibrasyonu için üzerine sabit enerjili bir demet yolluyoruz. Demet aynı anda birden fazla pikseli aydınlatıyor. Ölçüm alıyoruz, sinyal veren piksellerin x-y koordinatlarını ve sinyalin büyüklüğünü birer dizide kaydediyoruz.

Ölçüm 2

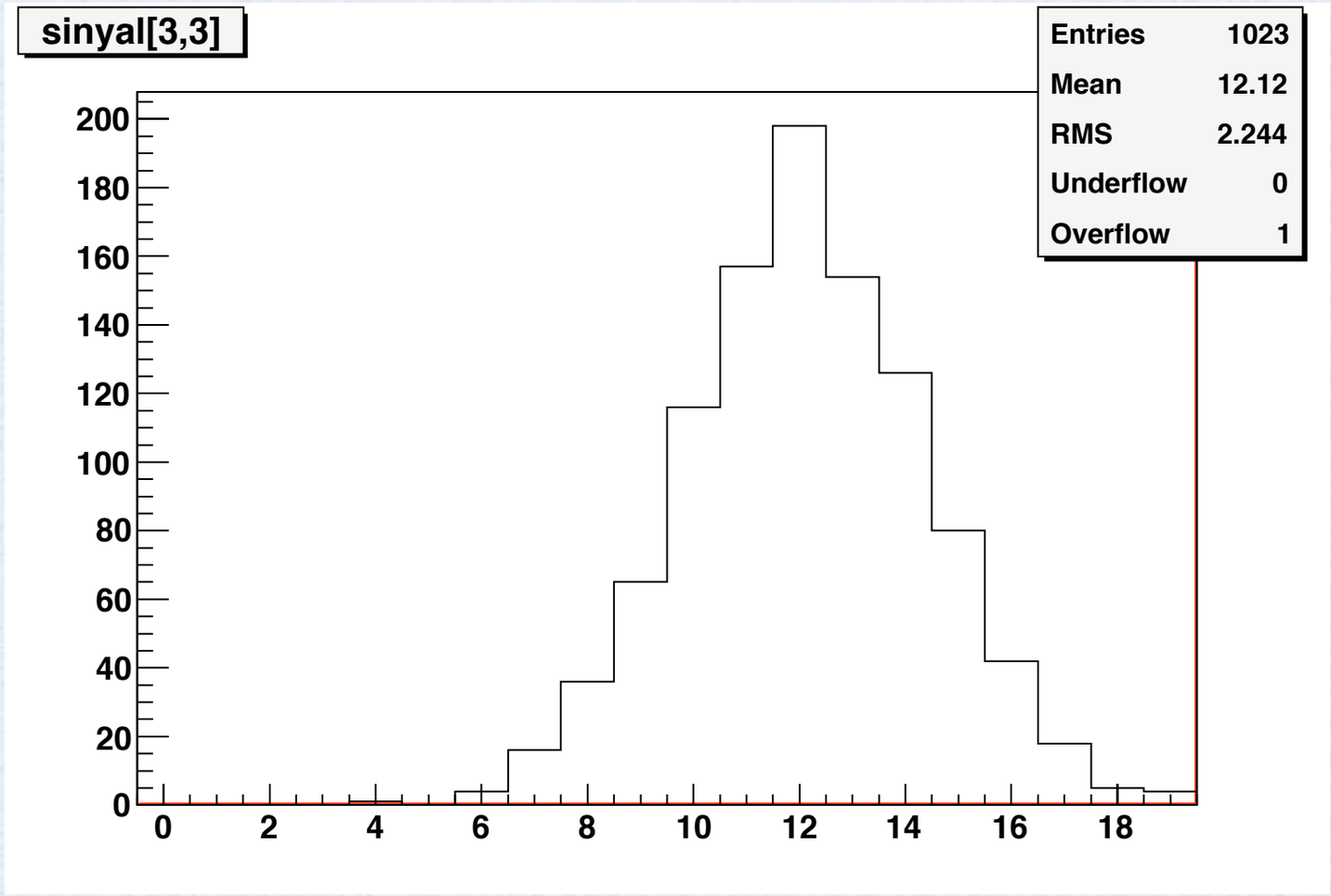
x[6]={1,5,5,6,6,6}

y[6]={8,3,2,3,2,1}

e[6]={1,13,15,13,16,14}

# Gerçek Bir Uyguluma II

- Bizden istenen her bir pikselin sinyal dağılımını çıkartmak. Yani deneyin sonunda her bir hücre için bir sinyal histogramı çıkmalı.

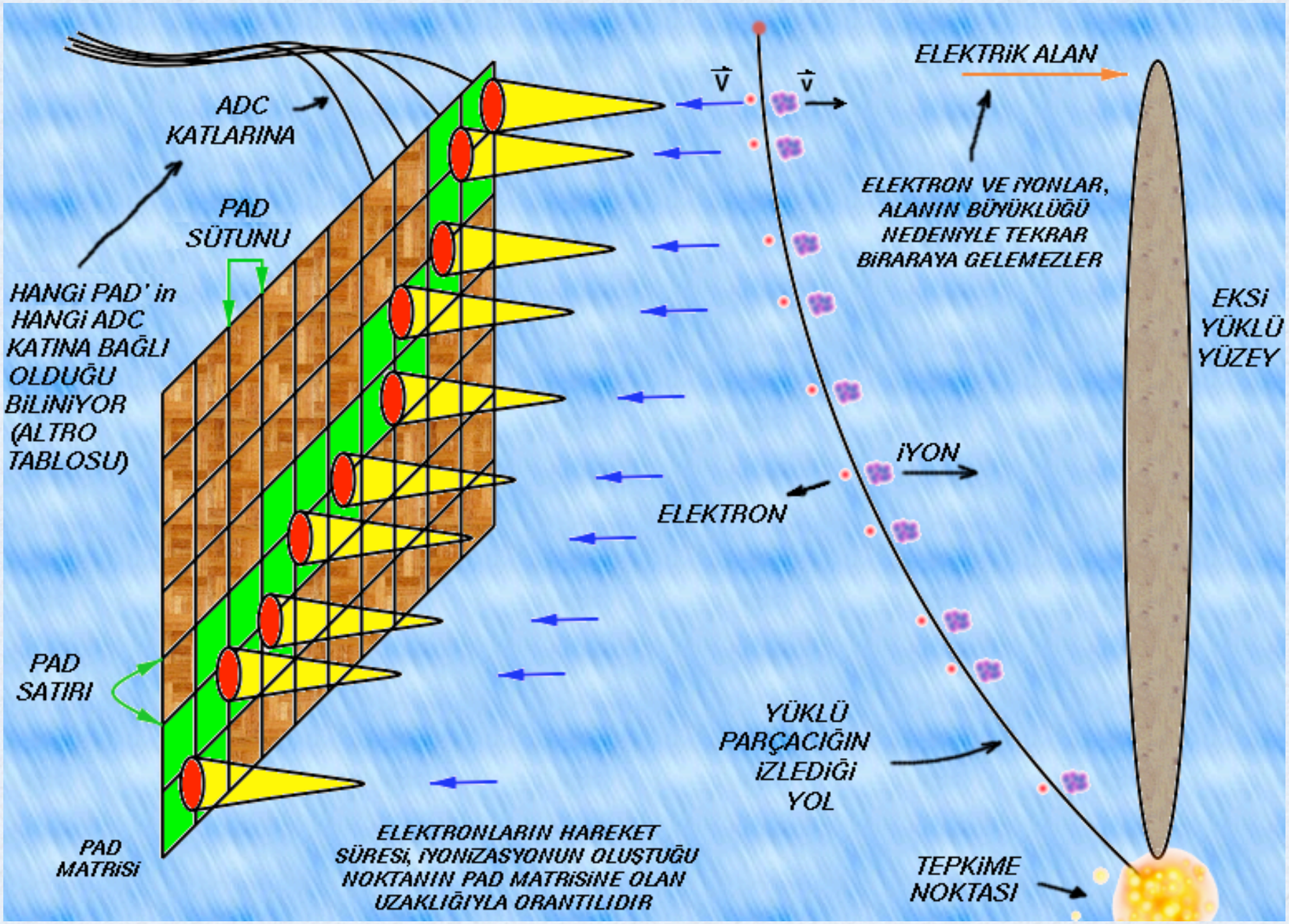


# Gerçek Bir Uygulama III

- Bu işi yapan sahte-kod (pseudo-code).

```
TH1F *sinyal_histo[10][10]  // yüz tane histogram
for ( i = tüm ölçümler ) {
   for ( j = bu ölçümdeki sinyal olan pikseller ) {
      int piksel_x = x[j], piksel_y = y[j]
      sinyal_histo[ piksel_x, piksel_y ] -> Fill( e[j] )
   }
}
```

# Gerçek İkinci Uyguluma




- Bu yaklaşım hakikaten çok sık.
- Örneğin Alice'in TPC'si. (Kaynak Özgür Çobanoğlu).

# Kendi Root Sinifiniz

```
#ifndef __UCVEKTOR__
#define __UCVEKTOR__

#include <cmath>

#include <TObject.h>

class UcVektor : public TObject {

 private:

  double m_r, m_theta, m_z;

 public:

  UcVektor() { m_r = 0; m_theta = 0; m_z = 0; }
  UcVektor(double r,  double theta,  double z);

  double r()     const {return m_r;}
  double theta() const {return m_theta;}
  double z()     const {return m_z;}

  // Kartezyen koordinatlar
  double x()    const {return m_r * cos(m_theta);}
  double y()    const {return m_r * sin(m_theta);}

  ClassDef(UcVektor,1);

};

#endif
```

```
#include "UcVektor.h"

#if !defined(__CINT__)
ClassImp(UcVektor)
#endif

UcVektor::UcVektor(double r, double theta, double z):
  m_r(r), m_theta(theta), m_z(z)
{ }
```

- Kendi ROOT class'ınızı oluşturmak çok kolay. Bu sekilde r, theta ve z için ayrı ayrı diziler yaratmaktansa, sadece bir UcVektor dizisi kullanabiliriz.
- Bir root paylasimli nesnesi lazım. root'un icinden:
  - `.L UcVektor.cxx++`

# Root Sinifimiz TTree'de

```
void agacyap() {

  gSystem->Load("UcVektor_cxx.so");
  UcVektor* v3; // = new UcVektor();

  Int_t j;

  TFile * dosya = new TFile("test.root", "RECREATE");
  TTree * agac = new TTree("agac", "test agacim");

  agac->Branch("olaynumarasi", & j, "j/I");
  agac->Branch("bir_ucvektor", & v3);

  for (j = 1; j <= 1000; j++) {

    // x-y duzleminde, herhangi bir yone bakan birim vektor yarat
    v3 = new UcVektor( 1, gRandom->Uniform(-3.14,3.14), 0 );

    agac->Fill();
    delete v3;

  }
  agac->Write();
  delete dosya;
}
```

- Simdi sinifimizi kullanan bir ROOT betiği yazalım ve çalıştıralım.
- Root içinden:
  - `.x agacyap.C`

# x(), y() Bilen TTree

- Dosyamızı Root ile bundan sonra her açışımızda bir uyarı görüyoruz. Root bize bu ağacın özel olduğunu hatırlatıyor. Biz de daha önce yaratmış olduğumuz nesne kütüğünü yüklüyoruz.

```
$ root -l test.root
root [0]
Attaching file test.root as _file0...
Warning in <TClass::TClass>: no dictionary for class UcVektor is available
root [1]  gSystem->Load("UcVektor_cxx.so");
```

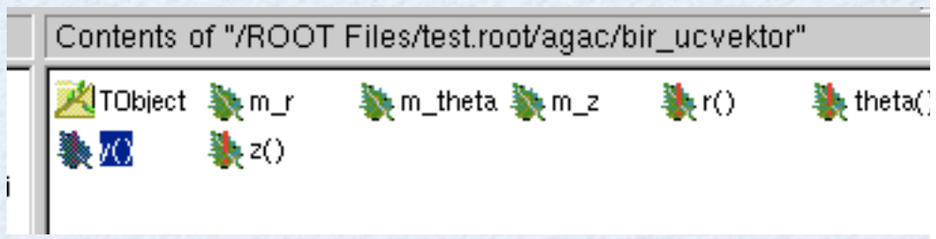
- Ve… x(), y() hesapları artık doğrudan…

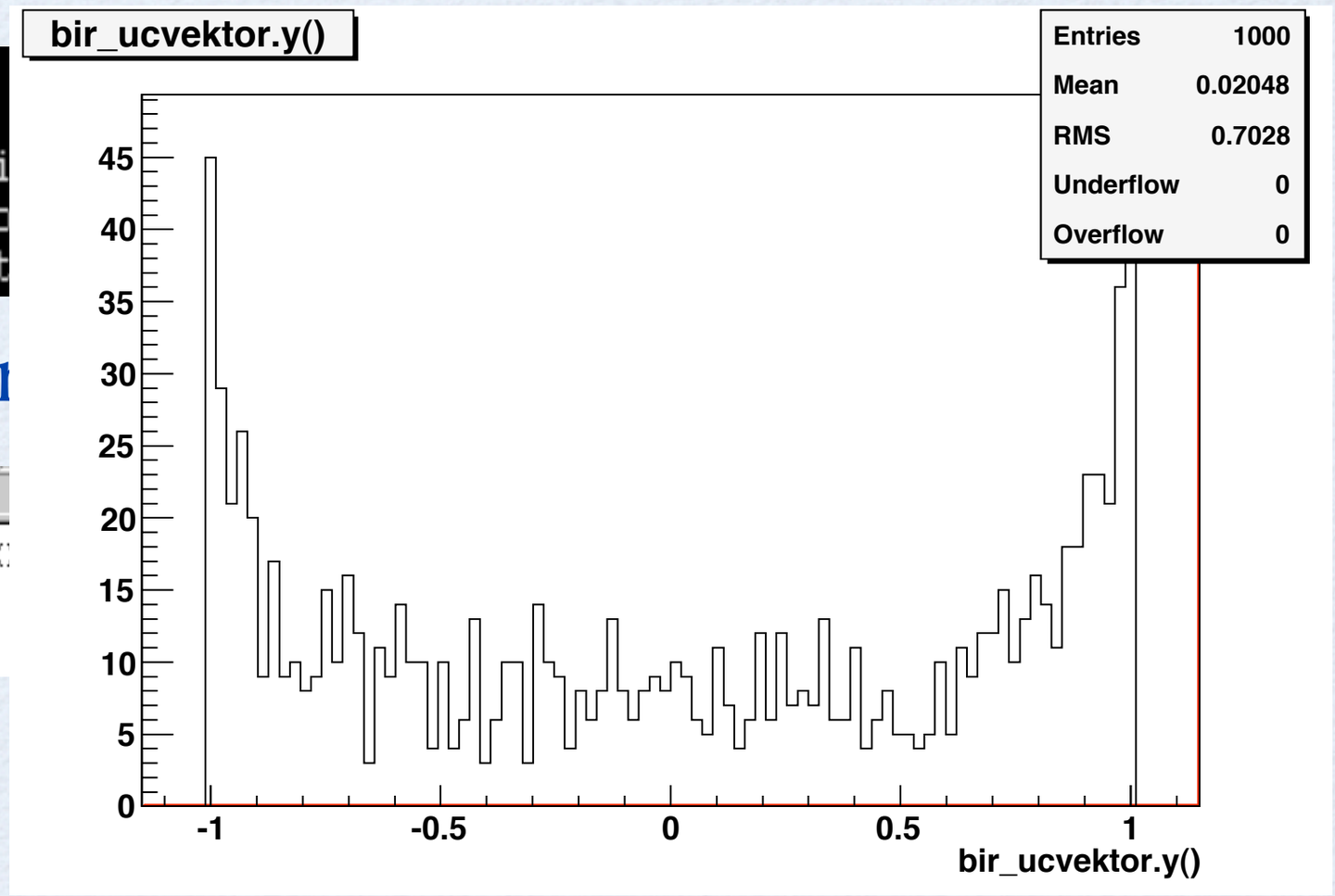
# x(), y() Bilen TTree

- Dosyamızı Root ile bundan sonra her açışımızda bir uyarı görüyoruz. Root bize bu ağacın özel olduğunu hatırlatıyor. Biz de daha önce yaratmış olduğumuz nesne kütüğünü yüklüyoruz.

```
$ root -l test.root
root [0]
Attaching file test.root as _fi
Warning in <TClass::TClass>: no
root [1]  gSystem->Load("UcVekt
```

- Ve… x(), y() hesaplar

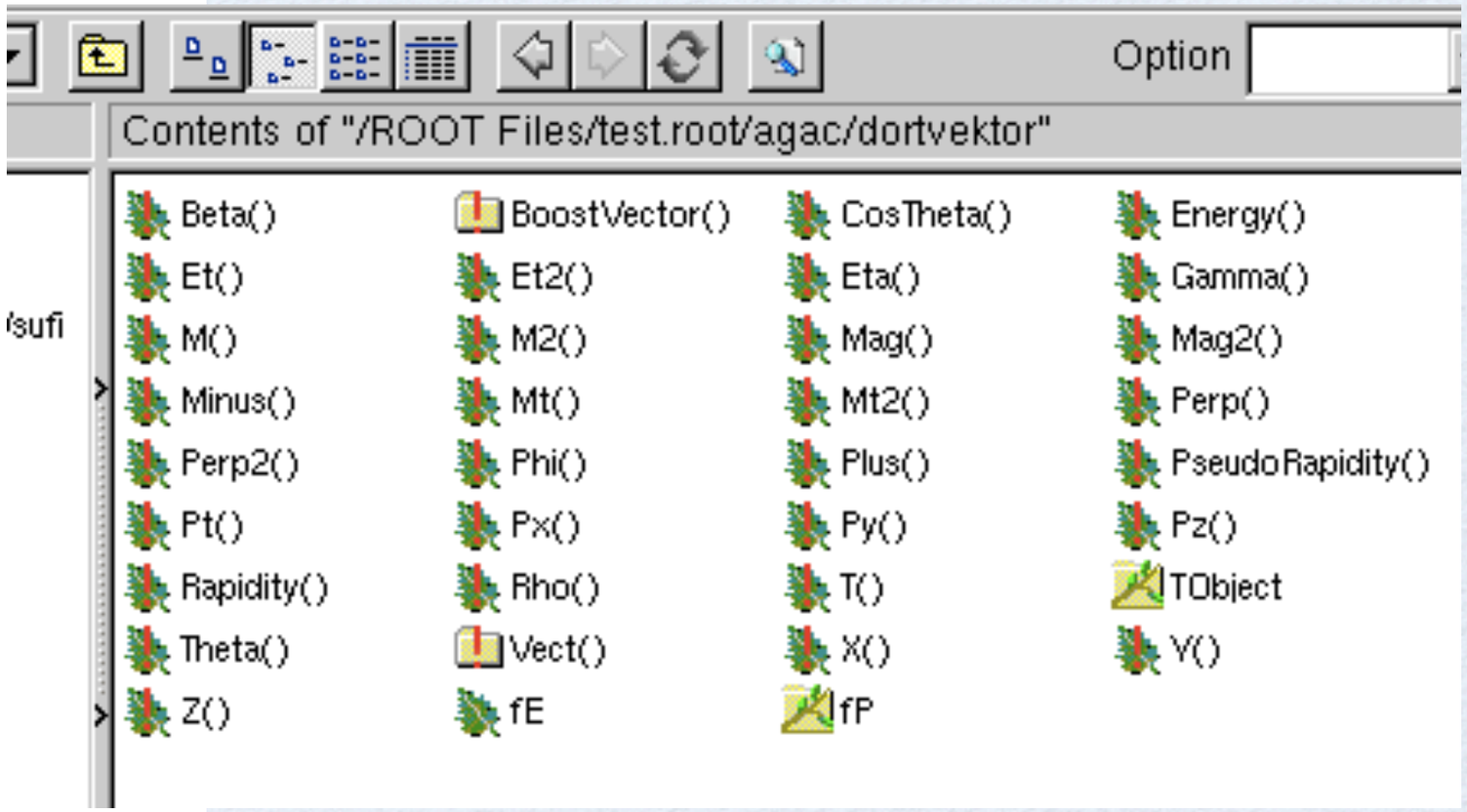


# TLorentzVector

```
                TLorentzVector (const TLorentzVector& lorentzvector)
                TLorentzVector (const TVector3& vector3, Double_t t)
                TLorentzVector (Double_t x = 0.0, Double_t y = 0.0, Double_t
        virtual ~TLorentzVector ()
      Double_t Angle (const TVector3& v) const
      Double_t Beta () const
          void Boost (const TVector3& b)
          void Boost (Double_t, Double_t, Double_t)
     TVector3 BoostVector () const
static TClass* Class ()
      Double_t CosTheta () const
      Double_t DeltaPhi (const TLorentzVector& v) const
      Double_t DeltaR (const TLorentzVector& v) const
      Double_t Dot (const TLorentzVector& q) const
      Double_t DrEtaPhi (const TLorentzVector& v) const
      Double_t E () const
      Double_t Energy () const
      Double_t Et () const
      Double_t Et (const TVector3& v) const
      Double_t Et2 () const
      Double_t Et2 (const TVector3& v) const
      Double_t Eta () const
     TVector2 EtaPhiVector ()
      Double_t Gamma () const
          void GetXYZT (Double_t* carray) const
          void GetXYZT (Float_t* carray) const
virtual TClass* IsA () const
      Double_t M () const
      Double_t M2 () const
      Double_t Mag () const
      Double_t Mag2 () const
      Double_t Minus () const
      Double_t Mt () const
      Double_t Mt2 () const
        Bool_t operator!= (const TLorentzVector& q) const
      Double_t operator() (int i) const
```

- Aynı oyunları TLorentzVector ile de oynayabiliriz.



# Ödev

- x(), y() bilen TTree sayfasındaki y dağılımını neden öyle?
- ROOT'a dayalı bir sürü ek paketler var. Bunlardan TMVA ne yapar acaba?
- UcVektor class'ını g++ ile derleyin.

# FAZLADAN LEBLEBİ

# TLorentzVector'lu TTree Yapma Örneği

- TLorentzVector sayfasındaki TTree'yi yapma yolum.

```
{
  TLorentzVector* v4 = new TLorentzVector();

  TFile * dosya = new TFile("test.root", "RECREATE");
  TTree * agac = new TTree("agac", "test agacim");

  Int_t j;

  agac->Branch("olaynumarasi", & j, "j/I");
  agac->Branch("dortvektor", & v4);

  for (j = 1; j <= 1000; j++) {

    // Bir foton uret. Enine momentumu 1 GeV'den baslayip birer birer artsin.
    // Eta'si Gaussian, phi'si duz bir sekilde dagilsin.

    v4->SetPtEtaPhiM( j, gRandom->Gaus(0,2), gRandom->Uniform(-3.14,3.14), 0 );
    agac->Fill();

  }
  agac->Write();
}
```