Lecture 11

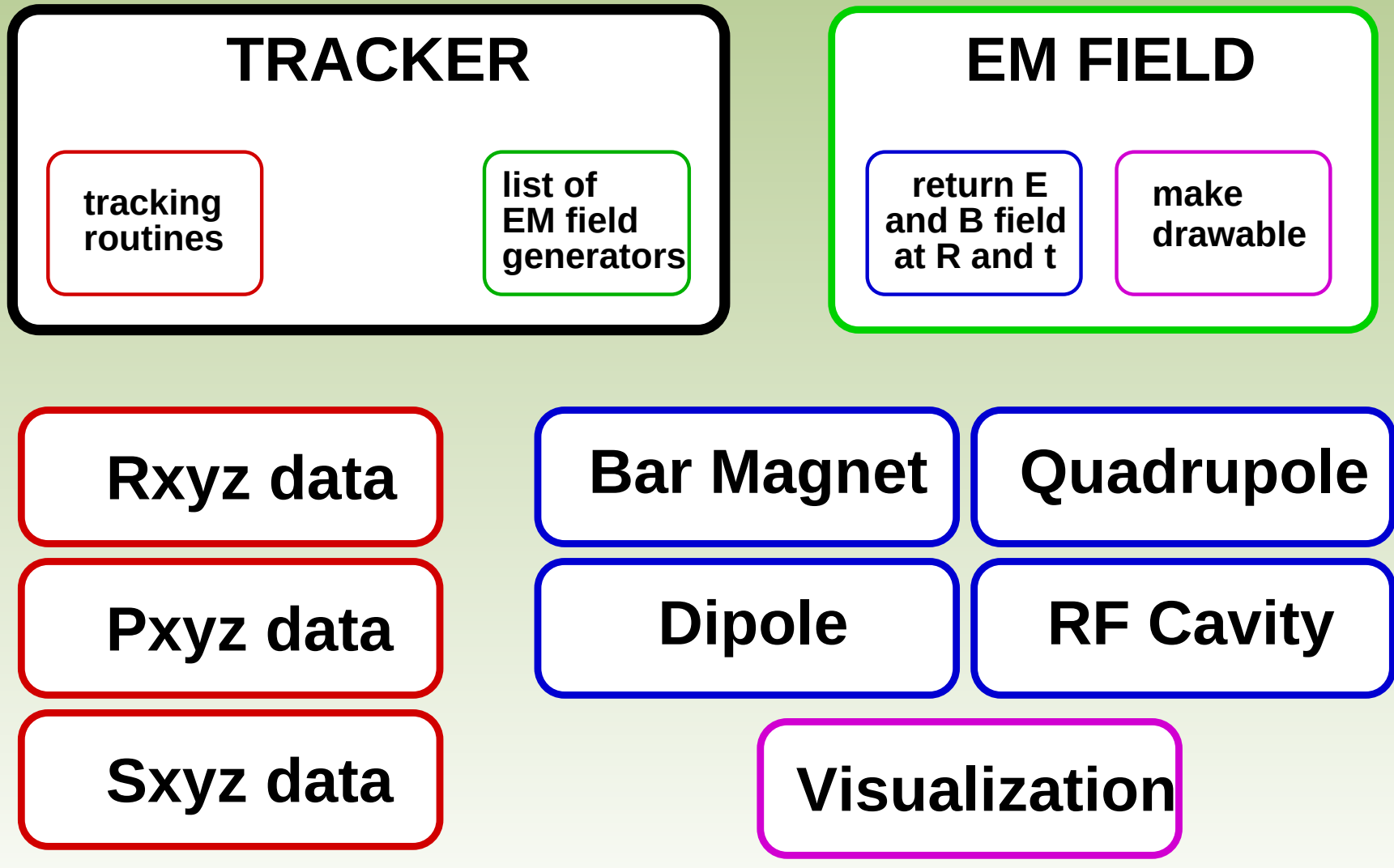
A ROOT-based program

Project goal

GOALS:

1. a general code that tracks charged particles through user-defined position and time dependent electro-magnetic fields.
2. easy to add new EM field geometries
3. some graphical display options

The General Idea



Main

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

#include "TROOT.h"
#include "TRint.h"
#include "TFile.h"

extern void InitGui();
VoidFuncPtr_t initfuncs[] = {InitGui, 0};
TROOT root("Rint", "The ROOT Interactive Interface", initfuncs);

int main(int argc, char **argv)
{
  // only use this program interactively, i.e. no batch running
  TRint *theApp = new TRint("ROOT example", &argc, argv);
  theApp->Run();
  return (0);
}
```

"Main" is called upon startup

TFieldTracker.h: Header

```
#ifndef __TFieldTracker_h__
#define __TFieldTracker_h__

#include "TList.h"
#include "TVector3.h"
#include "TLorentzVector.h"
#include "TPolyLine3D.h"
#include "TEMField.h"
#include "TBRIK.h"
#include "TGeometry.h"
#include "TNode.h"
#include "PhysicsConstants.h"
```

__TFieldTracker_h__: Compiler variable used to avoid loading file twice

Declare all classes to be used

TFieldTracker.h: Methods

```
class TFieldTracker
{
  public:
    TFieldTracker(Double_t dx, Double_t dy, Double_t dz);
    virtual ~TFieldTracker();

    void AddSource(TEMField* field);
    inline UInt_t getNSources(){return fListOfFieldSources->LastIndex()+1;};
    Bool_t Track(TVector3* xinit, TLorentzVector* pinit, TVector3* sinit,
                 TPolyLine3D* trajectory, TPolyLine3D* momentum, TPolyLine3D* spin);
    Bool_t GetBField(TVector3* X, TVector3* B, Double_t time);
    Bool_t GetEField(TVector3* X, TVector3* E, Double_t time);
    inline void SetDt(Double_t dt) {fDt = dt;};
    inline void SetT0(Double_t t) {fT0 = t;};
    inline Double_t GetDt() {return fDt;};
    inline void SetMaxNSteps(Int_t n) {fMaxNSteps = n;};
    TNode* GetWorldNode() { return fWorldNode; };
```

Constructor and Destructor are obligatory

TNode used to draw geometry

TPolyLine3D to store and draw results

TFieldTracker.h: Data Members

```
  protected:
    TList* fListOfFieldSources; //!
    Bool_t Step(TVector3* x0, TLorentzVector* p0, TVector3* s0,
                TVector3* x1, TLorentzVector* p1, TVector3* s1,
                Double_t time);
    Double_t fDt; //!
    Double_t fQ;  //! charge
    Double_t fG;  //! g-factor
    Double_t fT0; //! time at beginning of tracking
    Int_t fMaxNSteps; //!
    TGeometry* fTrackingSpace; //!
    TNode* fWorldNode; //!
    Double_t fDx,fDy,fDz; //!

    TVector3 *fxd,*fpd,*fsd; //! R, P and S three-vectors
    TVector3 *fE, *fB;  //! electric and magnetic field
    TVector3 *fkx1, *fkp1, *fks1, *fkx2, *fkp2, *fks2; | intermatiate
    TVector3 *fkx3, *fkp3, *fks3, *fkx4, *fkp4, *fks4; | RC variables
    TVector3* fOmega; //! intermediate RC variable

    TVector3 *currentX,  *nextX;
    TLorentzVector *currentP,  *nextP;
    TVector3 *currentS,  *nextS;
    TVector3* p0;

  ClassDef(TFieldTracker,1) // tracks particles through an EM field
};
#endif
```

ClassDef(name,version)
necessary for interactive use

TFieldTracker.cpp: The Beginning

Methods of TFieldTracker.h filled-in here!

```
#include "TFieldTracker.h"

ClassImp(TFieldTracker)

// unit of length is [cm]
// the following conversion constants are needed to get
// p[MeV/c] instead of p[kg.m/s]
// t[ns]    instead of t[s]
// q[e]     instead of q[C]
// E[V/m]   (SI unit)
// B[Gs]    instead of B[T]
static const Double_t Efac = 2.99792458e-7; // TMath::C()*1e-15
static const Double_t Bfac = 8.9875518e-3;  // TMath::C()*TMath::C()*1e-19

TFieldTracker::TFieldTracker(Double_t dx, Double_t dy, Double_t dz)
{
```

ClassImp(name)
necessary to generate docs

Initialize all variables

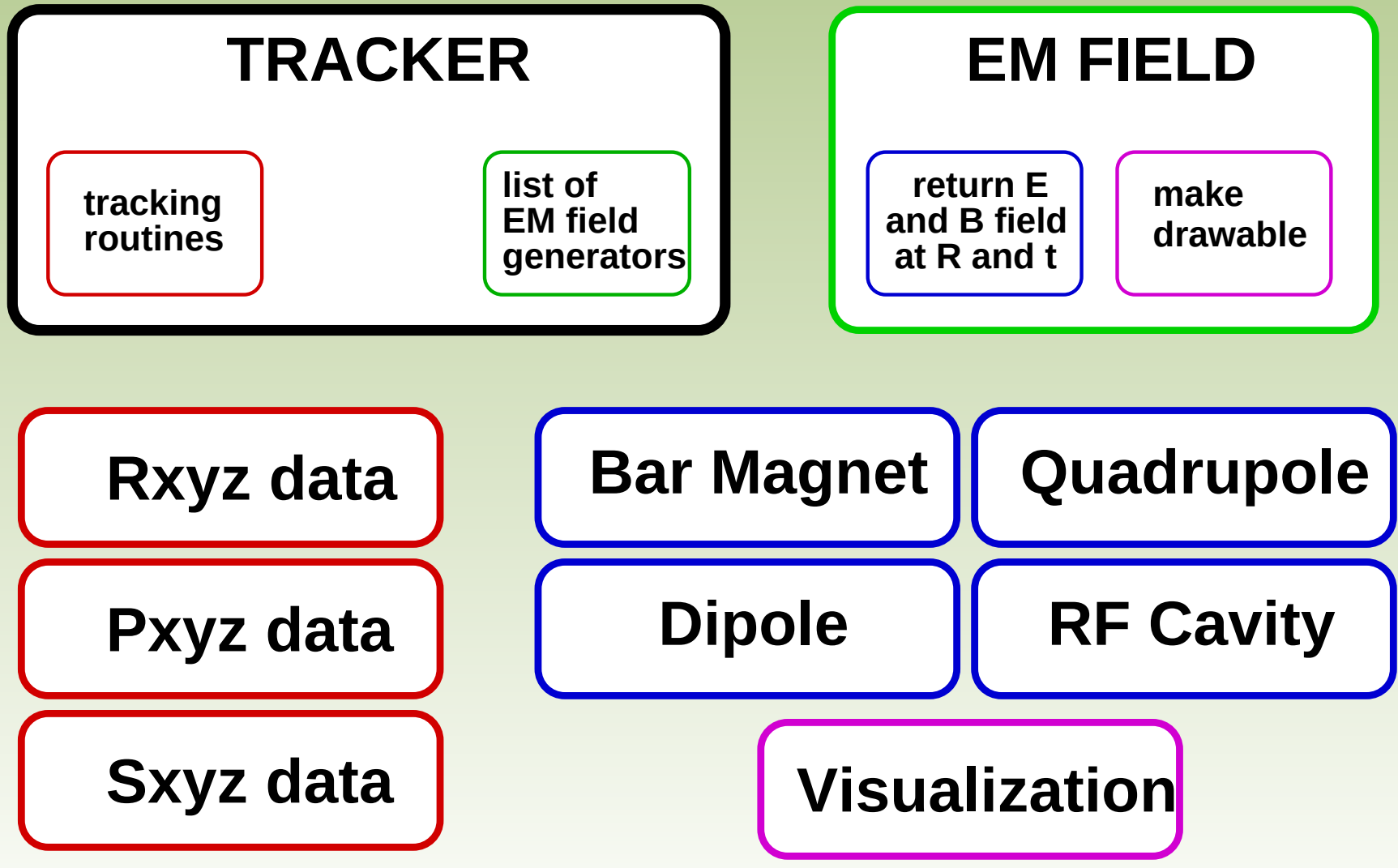
TFieldTracker.cpp: Selected Pieces

```
void TFieldTracker::AddSource(TEMField* field)
{
  fListOfFieldSources->Add(field);
  TNode* fieldNode = field->GetNode();
  if (fieldNode) fieldNode->SetParent(fWorldNode);
  // printf("Adding object at %p\n",field);
}
```

```
Bool_t TFieldTracker::GetBField(TVector3* X, TVector3* B, Double_t time)
{
  B->SetXYZ(0,0,0);
  Int_t nSources = fListOfFieldSources->GetSize();
  if (nSources<=0) return kFALSE;

  TVector3* dummyB =  new TVector3(0,0,0);
  Bool_t allSuccesFull = kTRUE;
  for (Int_t source = 0; source<nSources; source++)
  {
    Bool_t SuccesFull =
      ((TEMField*)fListOfFieldSources->At(source))->GetBField(X,dummyB,time);
    if (SuccesFull) (*B) += (*dummyB);
    allSuccesFull &= SuccesFull;
  }
  delete dummyB;
  return allSuccesFull;
}
```

Back to the General Idea



TEMField.h

All field generators classes inherit from TEMField!

Virtual functions:
must be filled in elsewhere

```
#ifndef __TEMField_h__
#define __TEMField_h__

#include "TVector3.h"
#include "TNode.h"

class TEMField : public TObject
{
  public:
    TEMField();
    virtual ~TEMField();
    inline virtual Bool_t GetBField(TVector3* X, TVector3* B, Double_t time) {return kFALSE;};
    inline virtual Bool_t GetEField(TVector3* X, TVector3* E, Double_t time) {return kFALSE;};
    inline virtual TNode* GetNode() {return fNode;};
  protected:
    TNode* fNode;

  ClassDef(TEMField,1) // Generic Electro-Magnetic Field
};

#endif
```

TEMField.cpp

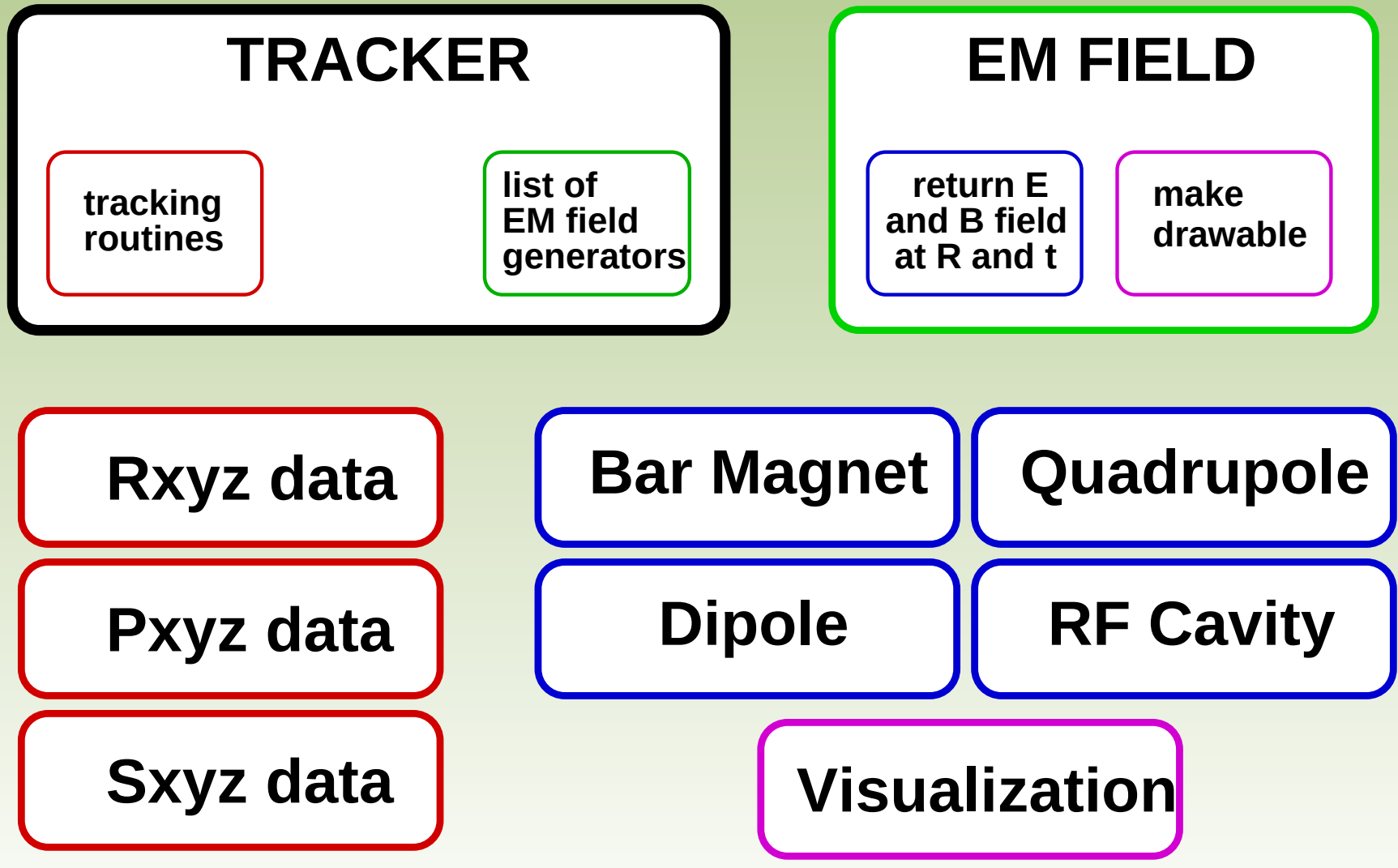
doesn't do much ...

```
#include "TEMField.h"
ClassImp(TEMField)

TEMField::TEMField() : TObject()
{ fNode = 0; }

TEMField::~TEMField()
{ }
```

The General Idea once More



TBarMagnet.h

Actual implementation of an EM-field: a barmagnet

TBarMagnet inherits from TEM-Field

```
#ifndef __TBarMagnet_h__
#define __TBarMagnet_h__

#include "TEMField.h"
#include "TBRIK.h"
#include "TRotMatrix.h"

class TBarMagnet : public TEMField
{
  public:
    TBarMagnet(TVector3* pos, TVector3* B, TVector3* x,
               Double_t dx, Double_t dy, Double_t dz);
    ~TBarMagnet();
    Bool_t GetBField(TVector3* X, TVector3* B);
    Bool_t GetBField(TVector3* X, TVector3* B, Double_t time);
    inline Bool_t GetEField(TVector3* X, TVector3* E) {return kFALSE;};
    inline Bool_t GetEField(TVector3* X, TVector3* E,Double_t time) {return kFALSE;};
    void Info();
  protected:
    TVector3* fPos; //!
    TVector3* fXaxis; //!
    TVector3* fYaxis; //!
    TVector3* fZaxis; //!
    Double_t fB; //!
    Double_t fDx,fDy,fDz; //!
    static Int_t fInstanceNumber; //!

  ClassDef(TBarMagnet,1) // calculates the magnetic field for a bar magnet
};

#endif
```

TBarMagnet.cpp: Visualization

```
#include "TBarMagnet.h"

#include "PhysicsConstants.h"

ClassImp(TBarMagnet)

Int_t TBarMagnet::fInstanceNumber = -1;

TBarMagnet::TBarMagnet(TVector3* pos, TVector3* B, TVector3* x,
      Double_t dx, Double_t dy, Double_t dz) : TEMField()
{
  ....
  Char_t BrikName[200];
  Char_t NodeName[200];
  Char_t RotMName[200];
  sprintf(BrikName,"Bfixmagnet%02d",fInstanceNumber);
  TBRIK* mworld = new TBRIK(BrikName,BrikName,"void",fDx,fDy,fDz);
  sprintf(RotMName,"RMfixmagnet%02d",fInstanceNumber);
  TRotMatrix* rotM = new TRotMatrix(RotMName,RotMName,
        fXaxis->Theta()*180/TMath::Pi(),fXaxis->Phi()*180/TMath::Pi(),
        fYaxis->Theta()*180/TMath::Pi(),fYaxis->Phi()*180/TMath::Pi(),
        fZaxis->Theta()*180/TMath::Pi(),fZaxis->Phi()*180/TMath::Pi());
  sprintf(NodeName,"fixmagnet%02d",fInstanceNumber);
  fNode = new TNode(NodeName,NodeName, BrikName);
  printf("Made node with name: %s\n",NodeName);
  fNode->SetPosition(fPos->X(),fPos->Y(),fPos->Z());
  fNode->SetMatrix(rotM);
  ....
}
```

every instance of TBarMagnet sees fInstanceNumber!

TNode can be drawn

Linkdef.h

```
#ifdef __CINT__

#pragma link off all globals;
#pragma link off all classes;
#pragma link off all functions;

#pragma link C++ class TBarMagnet;
#pragma link C++ class TDipole;
#pragma link C++ class TEMField;
#pragma link C++ class TFieldTracker;
#pragma link C++ class TFourierTransform;
#pragma link C++ class TMichel;
#pragma link C++ class TQuad;
#pragma link C++ class TRFCavity;
#pragma link C++ class TSeparator;
#pragma link C++ class TUniformEfield;

#endif
```

Necessary for I/O and interactive use (dictionary generator)
HINT: just copy and paste ...

List all your new classes here

files.make

Used by the make-file to identify all pieces of source code

```
DIRS += src/

SOURCES += main.cpp TBarMagnet.cpp TDipole.cpp TEMField.cpp \
TFieldTracker.cpp TFourierTransform.cpp TMichel.cpp TQuad.cpp \
TRFCavity.cpp TSeparator.cpp TUniformEfield.cpp fourier.cpp \
ft.cpp realfft.cpp w.cpp

HEADERS += PhysicsConstants.h TBarMagnet.h TDipole.h TEMField.h \
TFieldTracker.h TFourierTransform.h TMichel.h TQuad.h \
TRFCavity.h TSeparator.h TUniformEfield.h complex.h w.h \
LinkDef.h
```

Linkdef.h must be last!!!

Makefile

Be careful editing a Makefile

```
# Makefile for Tracker under Linux (also works under CYGWIN

# Include the lists of source files
include ../src/files.make

...

# Set up the executable file to be made by this makefile
EXE := tracker
all : $(EXE)

# ROOT libraries: needs root-config
ROOTLIBS := $(shell root-config --glibs) -lMinuit
ROOTINCLUDES := $(shell root-config --prefix)/include
ROOTCFLAGS := $(shell root-config --cflags)

...

# Rule to build the dictionary file, if there is one
$(DICTIONARY) : $(HEADERS) $(LINKDEF)
@echo Building dictionary...
@rootcint -f $(DICTIONARY) -c $(CCINCLUDEFLAGS) $^

# Rule to build a C++ file
%.o : %.cpp
g++ $(CFLAGS) -c $<

...
```

root-config is used to find Libs

rootcint generates dictionary

Compilation

```
KVIQ75:objects:758>make
Makefile:54: depend: No such file or directory
Building dictionary...
Making dependencies...
g++ -march=pentiumpro -O2 -g -Wall -fPIC -pthread -I/usr/local/share/root/pro/include
-I../src/ -I/usr/local/ share/root/pro/include  -c ../src/main.cpp
.....
g++ -march=pentiumpro -O2 -g -Wall -fPIC -pthread -I/usr/local/share/root/pro/include
-I../src/ -I/usr/local/share/root/pro/include  -c Dictionary.cpp
Linking...
g++ -g -o tracker  main.o TBarMagnet.o TDipole.o TEMField.o TFieldTracker.o
TFourierTransform.o TMichel.o TQuad.o TRFCavity.o TSeparator.o TUniformEfield.o
fourier.o ft.o realfft.o w.o Dictionary.o -L/usr/local/share/root/pro/lib -lCore
-lCint -lHist -lGraf -lGraf3d -lGpad -lTree -lRint -lPostscript -lMatrix -lPhysics
-lGui -pthread -lm -ldl -rdynamic -lMinuit
KVIQ75:objects:759> tracker
```