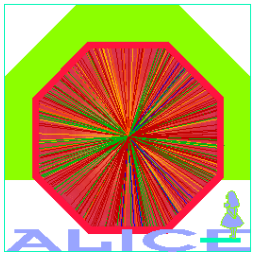
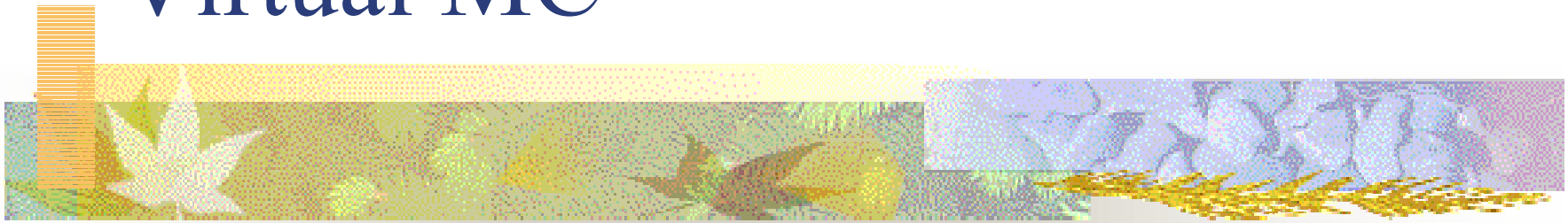


Virtual MC



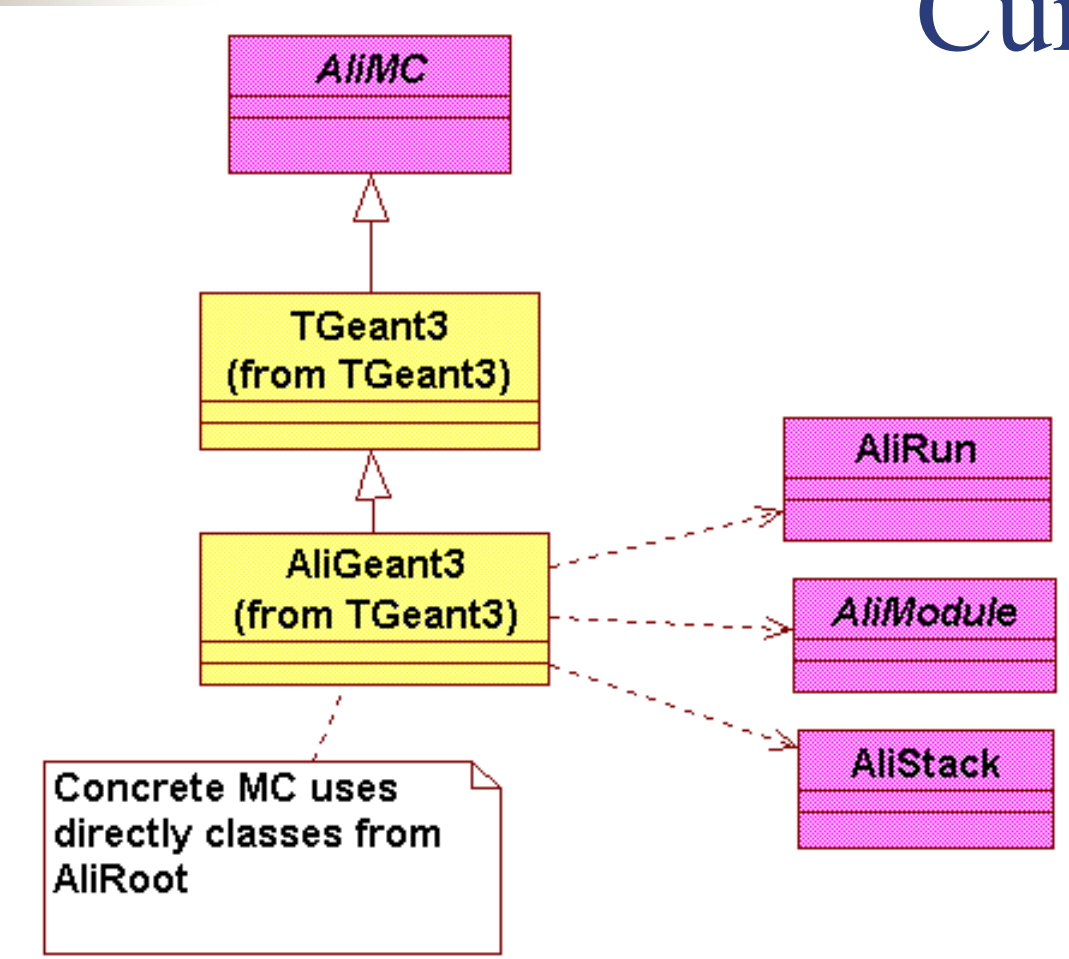
I. Hrivnacova, IPN, Orsay

ALICE Offline Meeting, CERN, 17 June 2002

Outline

- MC in ROOT
  - Generalisation of calls from MC to AliRoot
  - New interfaces for MC
- MC Implementations
  - geant3, geant4_mc
- MC Examples

Current Design



- Implementations of the MC interface (both for G3 and G4) require classes from AliRoot
  - AliRun, AliStack, …

New Interfaces

- Request:

  Make MC independent from AliRoot and usable in other experiment framework

- New interfaces for the calls from MC to the experiment framework has been introduced

MC Application

- Generalization of calls from MC to AliRun
  - Build materials, geometry
  - Generate primaries
  - Begin/end of event, primary track, track
  - Stepping
- Interface:

  TVirtualMCApplication

```
class TVirtualMCApplication : public TNamed
{
  public:
    // ... skipped

    virtual void ConstructGeometry() = 0;
    virtual void InitGeometry() = 0;
    virtual void GeneratePrimaries() = 0;
    virtual void BeginEvent() = 0;
    virtual void BeginPrimary() = 0;
    virtual void PreTrack() = 0;
    virtual void Stepping() = 0;
    virtual void PostTrack() = 0;
    virtual void FinishPrimary() = 0;
    virtual void FinishEvent() = 0;

    virtual Double_t TrackingRmax() const = 0;
    virtual Double_t TrackingZmax() const = 0;
    virtual void Field(const Double_t* x, Double_t* b) const = 0;
};
```

MC Stack

- Current design:

  G3 – Geant3 stack replaced with AliStack

  G4 – Primary particles are loaded from AliStack and transformed to G4 objects; G4 secondary particles objects are transformed to TParticles objects and saved in AliStack
- New design:

  TVirtualMCStack

  – the interface to the application stack

```
class TVirtualMCStack : public TObject {
  public:
    // ... skipped

    // methods
    virtual void  SetTrack(Int_t done, Int_t parent, Int_t pdg,
                           Double_t px, Double_t py, Double_t pz, Double_t e,
                           Double_t vx, Double_t vy, Double_t vz, Double_t tof,
                           Double_t polx, Double_t poly, Double_t polz,
                           TMCProcess mech, Int_t& ntr, Double_t weight,
                           Int_t is) = 0;
    virtual TParticle* GetNextTrack(Int_t& track)  = 0;
    virtual TParticle* GetPrimaryForTracking(Int_t i) = 0;

    // set methods
    virtual void  SetCurrentTrack(Int_t track) = 0;

    // get methods
    virtual Int_t  GetNtrack() const = 0;
    virtual Int_t  GetNprimary() const = 0;
    virtual Int_t  CurrentTrack() const = 0;
};
```

MC Decayer

- Current design:

  AliDecayer - abstract base class for the external decayer

- New design:

  Renamed to TVirtualMCDecayer

```
class TVirtualMCDecayer : public TObject
{
  public:
   // ... skipped

    virtual void      Init() = 0;
    virtual void      Decay(Int_t idpart, TLorentzVector* p) = 0;
    virtual Int_t     ImportParticles(TClonesArray *particles) = 0;
    virtual void      SetForceDecay(Decay_t type) = 0;
    virtual void      ForceDecay() = 0;
    virtual Float_t  GetPartialBranchingRatio(Int_t ipart) = 0;
    virtual Float_t  GetLifetime(Int_t kf) = 0;
    virtual void      ReadDecayTable() = 0;
};
```

MC

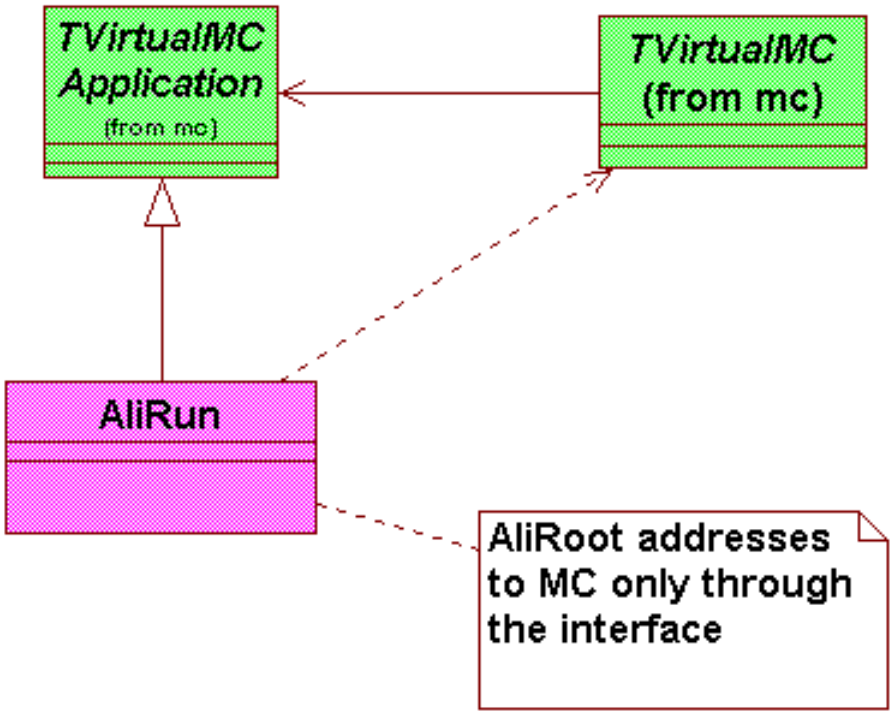
- AliMC renamed to TVirtualMC
- Inheritance from AliRndm replaced with composition of TRandom
  - TRandom – added functions from AliRndm
  - User code: gMC – > GetRandom() –> Random();
- Added data members:
  - TVirtualMCApplication* fApplication; //! User MC application
  - TVirtualMCStack* fStack; //! Particles stack
  - TVirtualMCDecayer* fDecayer; //! External decayer
  - TRandom* fRandom; //! Random number generator

MC ++

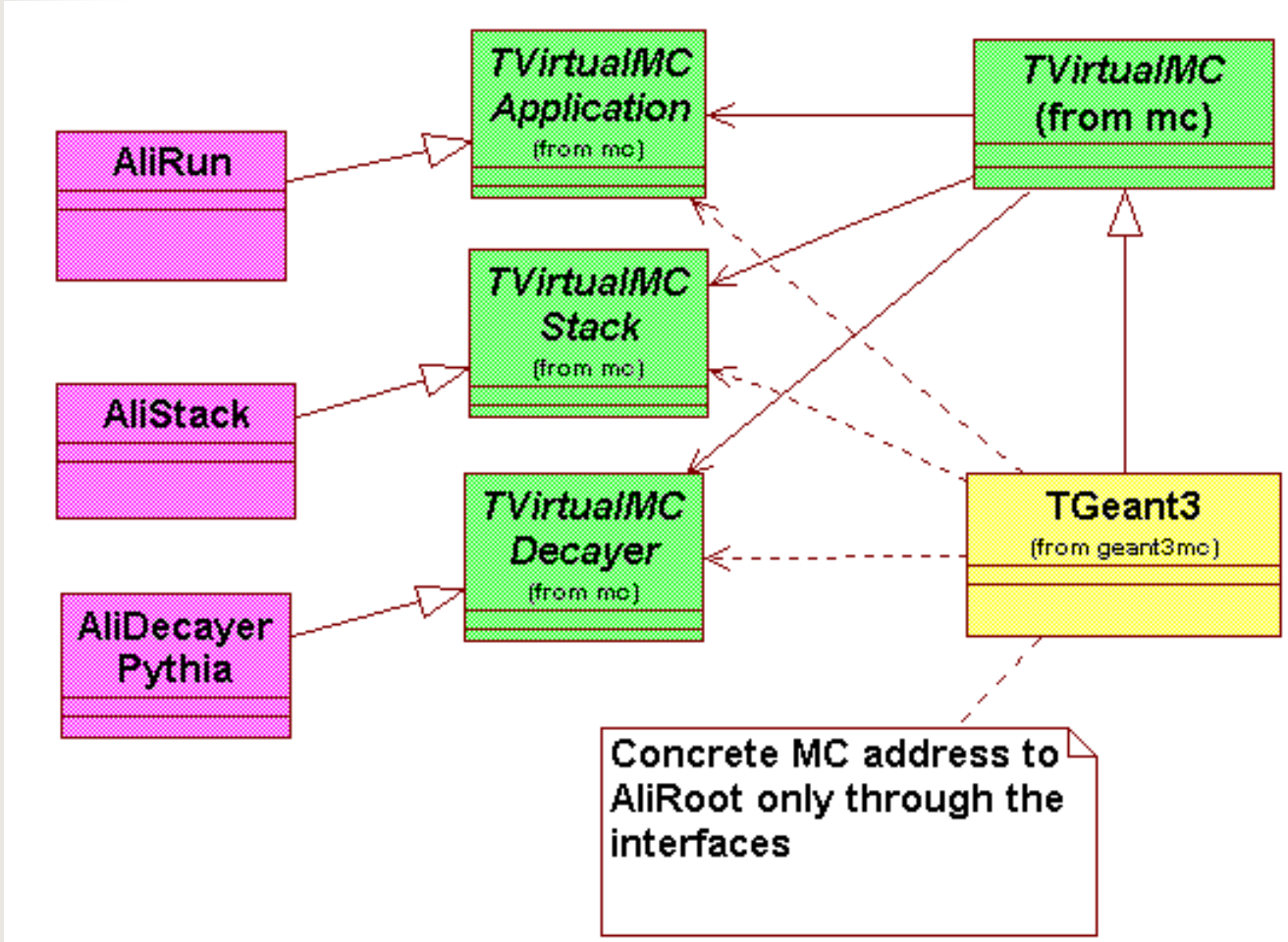
- Double precision
  - Overloaded functions with single and double precision
    - virtual Int_t Gsvolu(const char* name, const char* shape, Int_t nmed, Float_t* upar, Int_t np) = 0;
    - virtual Int_t Gsvolu(const char* name, const char* shape, Int_t nmed, Double_t* upar, Int_t np) = 0;
  - Enables to define geometry in a user code with double precision
    - Geant3 – conversion to single precision
    - Geant4, Fluka + Flugg - in double precision

```
class TVirtualMC : public TNamed
{
 public:
   // methods for building/management of geometry
   // methods for physics management
   // methods for step management
   // control methods
   // other (then geometry/step/run management) methods
   // set methods
   // get methods

 protected:
   TVirtualMCApplication*  fApplication; //! User MC application

 private:
   // … skipped
   TVirtualMCStack*       fStack;     //! Particles stack
   TVirtualMCDecayer*  fDecayer;  //! External decayer
   TRandom*                fRandom;  //! Random number generator
};
```

New Design


TVirtualMC Application
(from mc)
TVirtualMC (from mc)
AliRun
AliRoot addresses to MC only through the interface

New Design ++


TVirtualMC Application
(from mc)
TVirtualMC (from mc)
AliRun
TVirtualMC Stack
(from mc)
AliStack
TVirtualMC Decayer
(from mc)
TGeant3
(from geant3mc)
AliDecayer Pythia
Concrete MC address to AliRoot only through the interfaces

mc in ROOT

- The interfaces:

  TVirtualMC, TVirtualMCApplication, TVirtualMCStack, TVirtualMCDecayer

- Other files

  TMCProcess, TPDGCode – enum types

- Library (compiled on demand)

  libMC.so

MC Implementations

- Available for MCs:
  - Geant3, Geant4
- Requirements
  - *Standalone downloadable*
  - *Standalone compilable*
  - *Dynamically loadable libraries*
  - *Maintained in CVS*

Geant3

- geant3
  - New structured module
  - From AliRoot: MINICERN, GEANT321, TGeant3
- Contains:
  - geant321, geant3mc, minicern – source code
  - Makefile
  - config – platform dependent makefiles
  - tmp, lib – created at compilation time

Geant3 ++

- Libraries:
  - libgeant321.so - merged geant321 + geant3mc
    - Bi-directional dependence between geant321 and geant3mc does not allow to separate libraries in two
  - libminicern.so
  - dummy libraries:

    libG3Dummy.so, libG3mcDummy.so

Geant4_MC

- geant4_mc
  - From AliRoot: TGeant4
- File structure and makefiles system follow Geant4 structure
  - Geant4 environment setting transparent to geant4_mc
- Contains:
  - source – source code
  - config – general makefiles
  - include, tmp, lib – created at compilation time
  - examples – examples of usage of MC

Geant4_MC ++

- Libraries:
  - libgeant4mc.so
  - Optionally granular libraries can be created

AliGeant3

- Class AliGeant3 merged back in TGeant3
  - After introducing new interfaces (MC application, MC stack) – no dependence on AliRoot
- AliGeant3GUI
  - All Ali* classes for Geant3 GUI

AliGeant4

- Introducing new interfaces (MC application, MC stack) enabled to move ~18 classes in geant4_mc
- Kept classes
  - Stacking action
  - Set of classes for interactive selection of modules
  - Set of classes for interactive particle gun
- Kept current structure
  - config, doc, macro

Distribution

- CVS
  - mc – ROOT server
  - geant3, geant4_mc – ALICE server
    - 2 new packages parallel with AliRoot
- ROOT Web
  - Distribution of source:

    geant3.tar.gz, geant4_mc.tar.gz
  - Libraries for supported platforms
  - Updates with each Root minor release

Other Interfaces Implementation

- To be implemented by user

  MyMCApplication : TVirtualMCApplication -mandatory

  MyMCStack : TVirtualMCStack - mandatory

  MyMCDecayer : TVirtualMCDecayer - optional

- AliRoot:

  AliRun : TVirtualMCApplication

  AliStack : TVirtualMCStack

  AliDecayerPythia : TVirtualMCDecayer

MC Examples – E01

- Geant4 novice example N01 rewritten with usage of the Virtual Monte Carlo
- Builds simple geometry, tracks geantino and prints info at each step
- Classes:
  - Ex01MCApplication, Ex01MCStack, Ex01Particle
- Macros:
  - run_g3, run_g4

E01 with G3

```
{
 // Load basic libraries
  gSystem->Load("libPhysics");
  gSystem->Load("libMC");
  gSystem->Load("$(MCINSTALL)/lib/Linux-g++/libexample01");

  // Load Geant3 libraries
  // ... skipped

  // MC application
  Ex01MCApplication* appl
    = new Ex01MCApplication("Example01", "The example01 MC application");

  appl->InitMC("g3Config.C");
  appl->RunMC(1);
}
```

g3Config.C

```
void Config()
{
  cout << "Create geant3" << endl;
  new  TGeant3("C++ Interface to Geant3");
}
```

E01 with G4

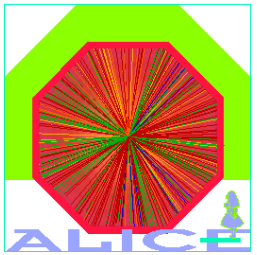
```
{
 // Load basic libraries
  gSystem->Load("libPhysics");
  gSystem->Load("libMC");
  gSystem->Load("$(MCINSTALL)/lib/Linux-g++/libexample01");

  // Load Geant4 libraries
  // ... skipped

  // MC application
  Ex01MCApplication* appl
    = new Ex01MCApplication("Example01", "The example01 MC application");

  appl->InitMC("g4Config.C");
  appl->RunMC(1);
}
```

g4Config.C

```
void Config()  {

    // RunConfiguration for Geant4
    TG4RunConfiguration* runConfiguration  = new TG4RunConfiguration();

    // TGeant4
    new TGeant4("TGeant4", "The Geant4 Monte Carlo", runConfiguration);
}
```

MC Examples – E02

- Geant4 novice example N02 rewritten with usage of the Virtual Monte Carlo
- Builds parameterised geometry with magnetic field, tracks proton, registers hits in a tracker and saves hits and kinematics in a Root file
- Classes:
  - Ex02MCApplication, Ex02MCStack, Ex02Particle, Ex02DetectorConstruction, Ex02ChamberParameterisation, Ex02MagneticField, Ex02TrackerSD, Ex02TrackerHit, Ex02RootManager
- Macros:
  - run_g3, run_g4



Geant4 Integration

I. Hrivnacova, IPN, Orsay

ALICE Offline Meeting, CERN, 18 March 2002

Geant4 4.1

- Release 4.1 – June 2002
- No patch to Geant4 for AliRoot + Geant4 needed
  - Patch for reflection
    - G4ReflectedSolid - supports CSG and CSG-like solids
  - Patch for stacking
    - Added needed method in G4StackManager
    - AliRoot stacking mechanism can be achieved with G4 design
  - Patch for optical boundary process
    - Added access method to process status
- Patch to Geant4 for AliRoot + Fluka + Flugg still needed
  - Patch for G4Navigator
    - A special function added for Flugg