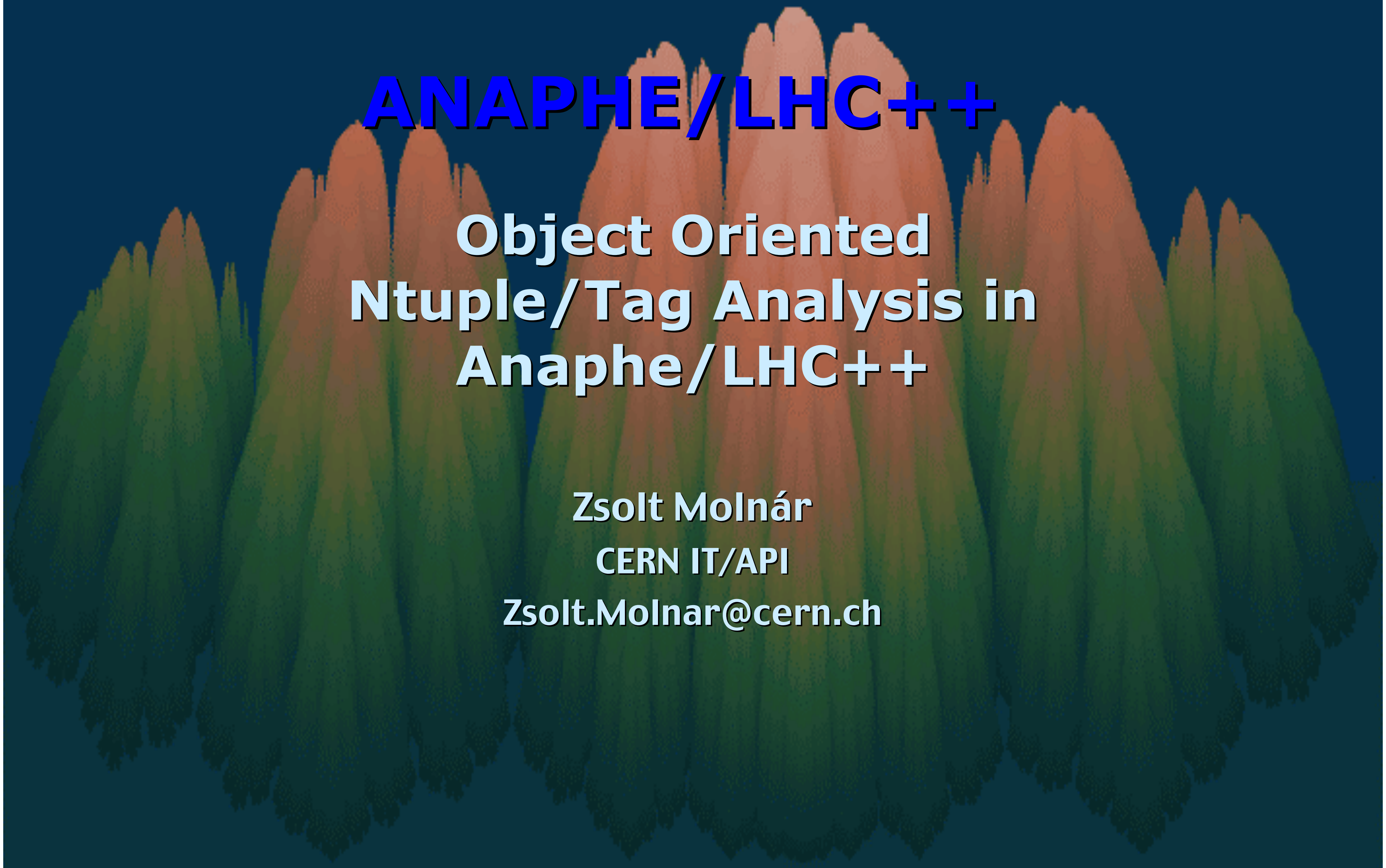
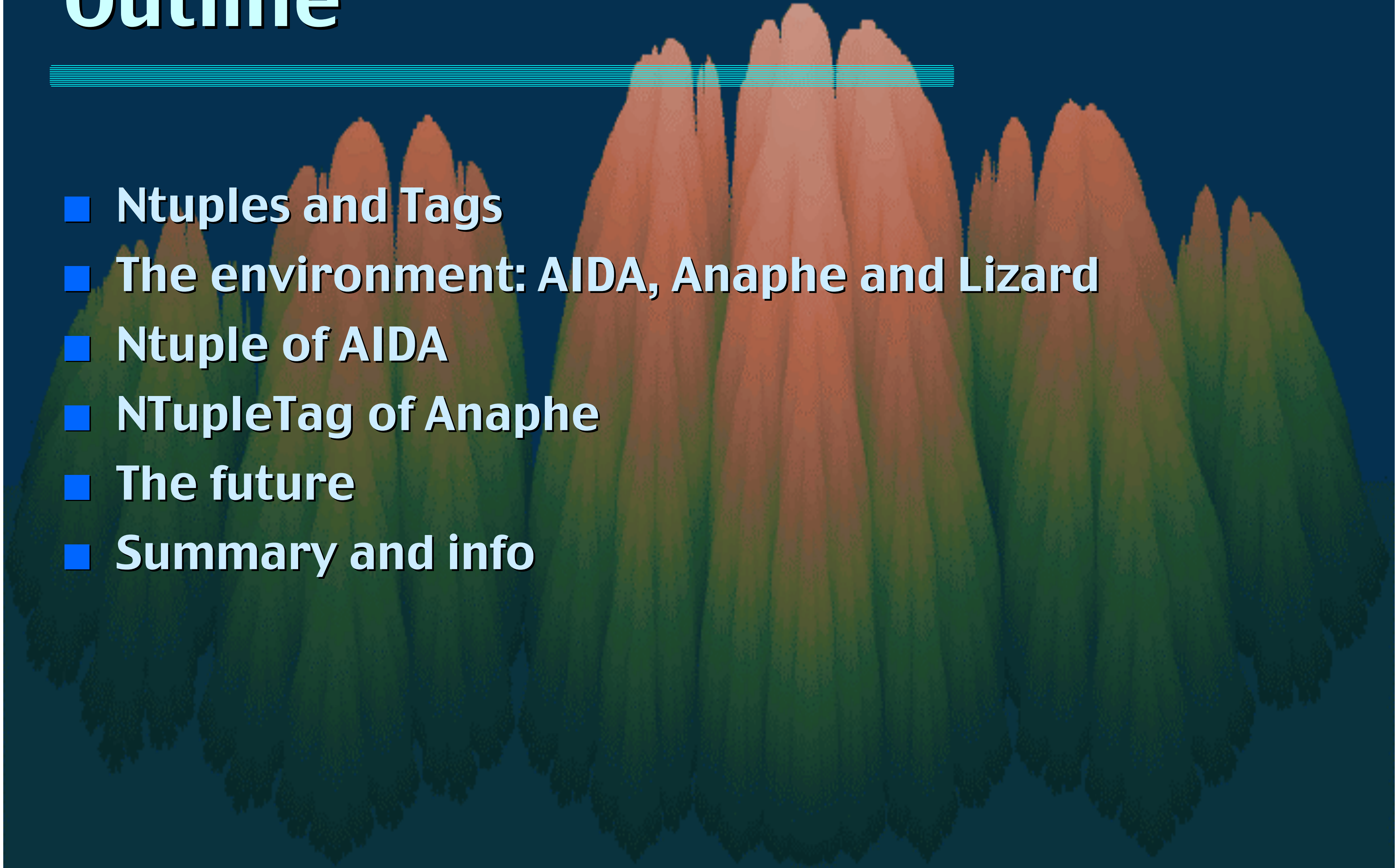
ANAPHE/LHC++

Object Oriented Ntuple/Tag Analysis in Anaphe/LHC++

Zsolt Molnár
CERN IT/API
Zsolt.Molnar@cern.ch



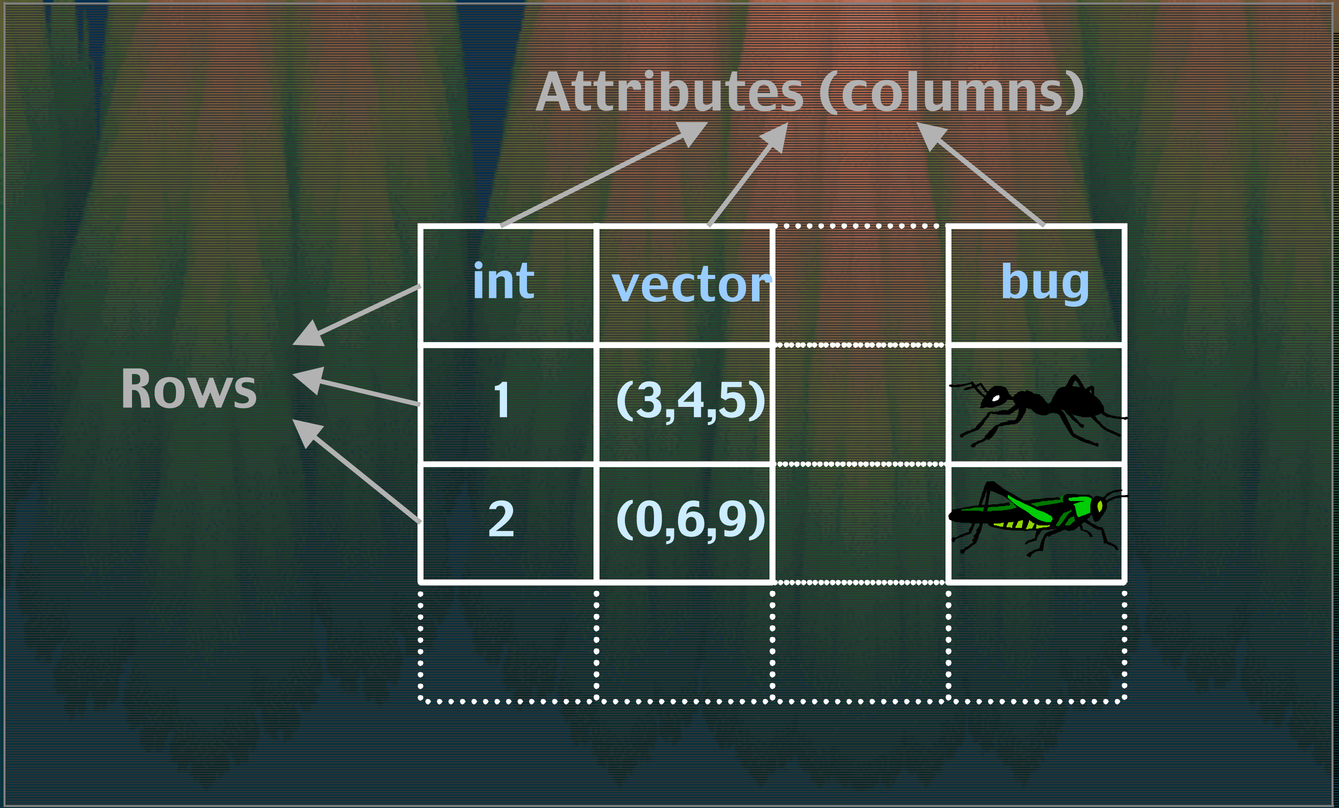
Outline



- Ntuples and Tags
- The environment: AIDA, Anaphe and Lizard
- Ntuple of AIDA
- NTupleTag of Anaphe
- The future
- Summary and info

Ntuple – definition

- Ntuple is a simple **table** consisting of rows and columns
  - types of columns can be different but fixed in a column



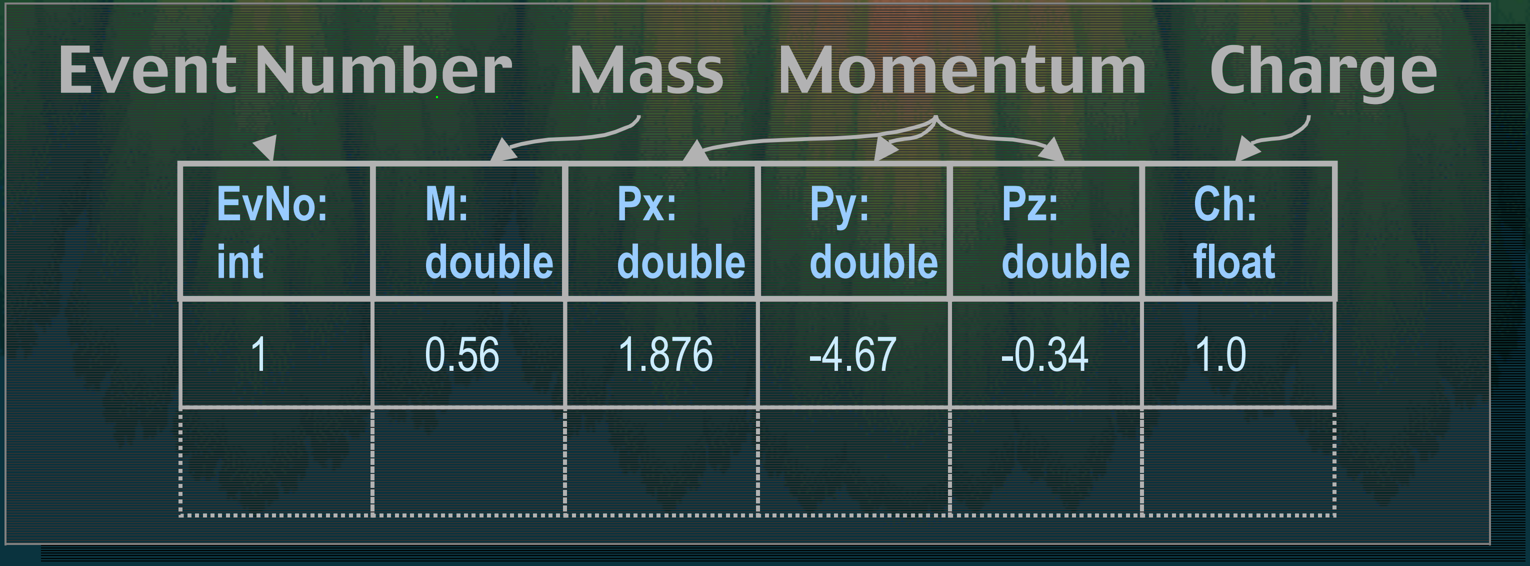
Attributes (columns)

Rows

int	vector		bug
1	(3,4,5)		
2	(0,6,9)		

Use of ntuples – the past

- Event data is stored in an experiment-specific hierarchical format.
- History: Re-clustered to obtain a more compact and more efficient representation -> ntuples (ex. HBOOK + PAW).
- Different experiments <-> different data models
- Ntuple has no direct connection with the original data



Event Number Mass Momentum Charge

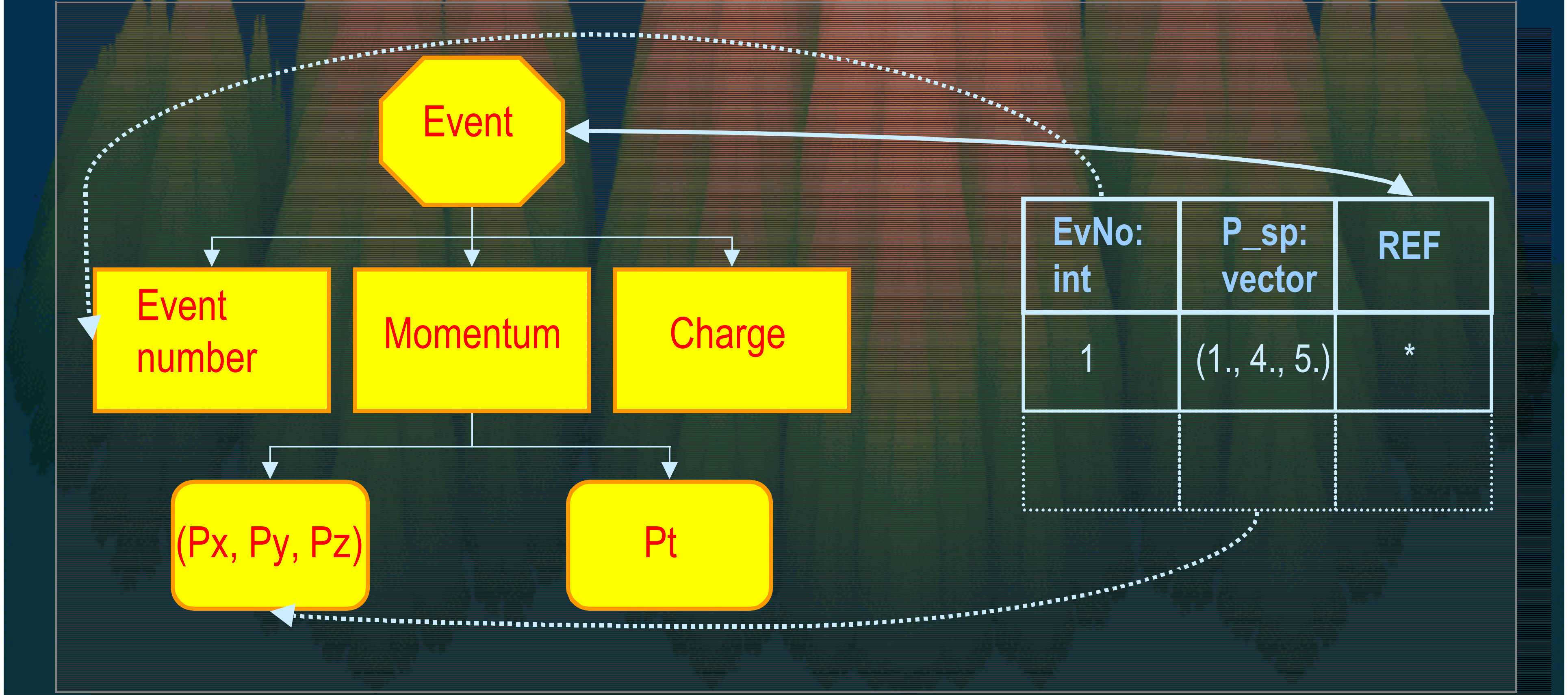
EvNo: int	M: double	Px: double	Py: double	Pz: double	Ch: float
1	0.56	1.876	-4.67	-0.34	1.0

Tags – the future

- An event means huge amount of data
- Regroup selected data replicas to an ntuple
  - replicas are stored as tags
- Use of general ntuple analyzer tool is possible
- Event data and tags are stored in the same federated DataBase.
- Maintain direct connection to the event
  - Links to the original data; works on the fly

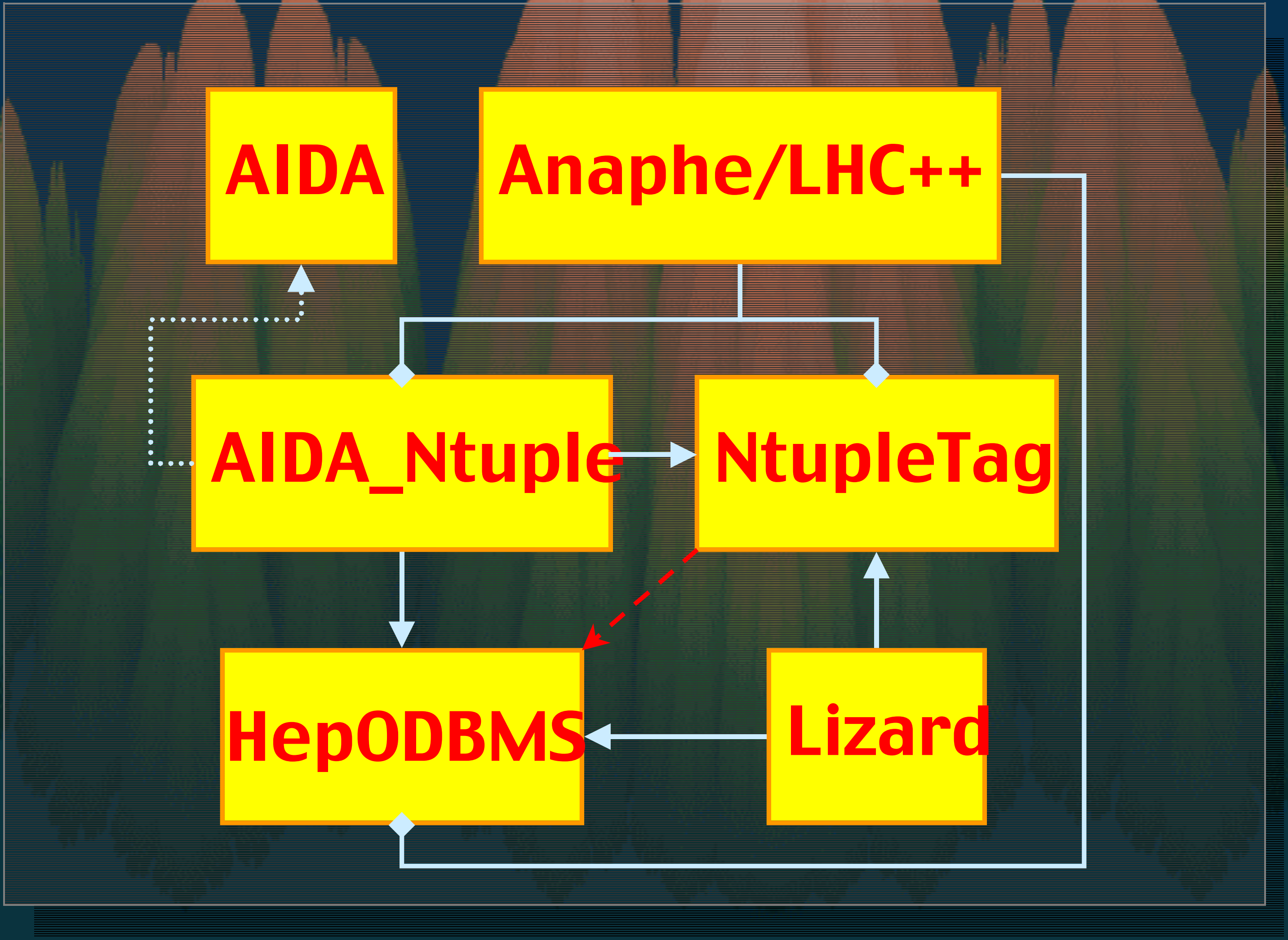
Tags – the future

Tag collection is:

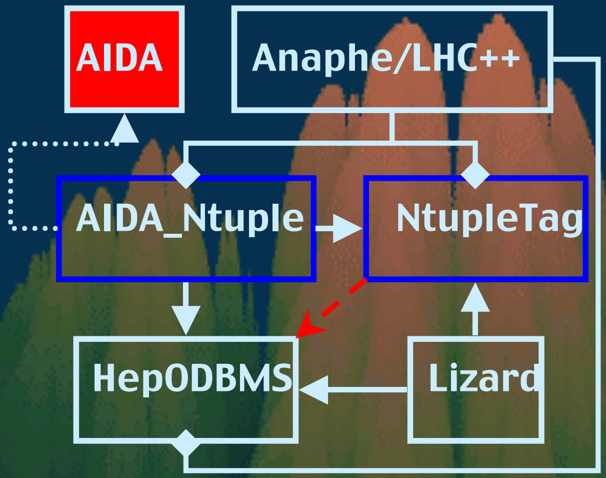


EvNo: int	P_sp: vector	REF
1	(1., 4., 5.)	*

Environment



AIDA



AIDA

Anaphe/LHC++

AIDA_Ntuple

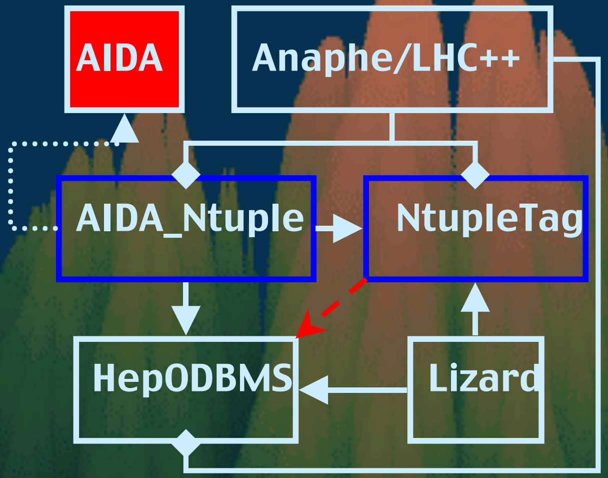
NtupleTag

HepODBMS

Lizard

- AIDA – Astronomical Image Data Archive
- AIDA on-line diabetes software simulator
- AIDA – Aerosols and Heterogeneous Chemistry in the Atmosphere
- AIDA: Agricultural Income Disaster Assistance
- AIDA -- *Abstract Interfaces for Data Analysis*
- AIDA – An opera from Verdi

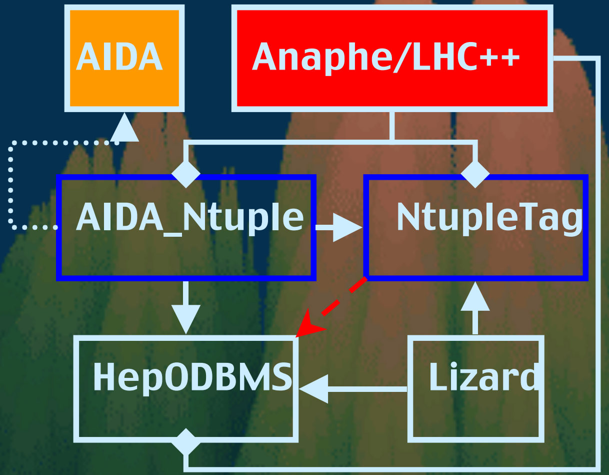
AIDA



- Has been formed to systematically design **interfaces** for components of data analysis tools.
- User group started on HepVis'99 conference

- Only **interfaces**, **basic types** and types from **foundation libraries** (like STL) are allowed in the interfaces
- Only **pure virtual methods** are allowed
- Internals do not appear in the interface

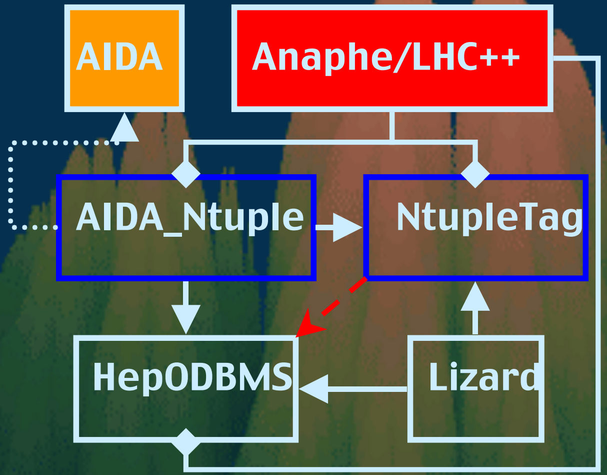
Anaphe/LHC++



ANAlysis for **PH**ysics **E**xperiments

On the Cretan Sea, the *Argo* was caught again in a terrible storm. Jason prayed to Apollo, who sent a flesh of light which showed them that they were very close to the small island of Anaphe. The Argonauts landed on the island and raised a shrine to Apollo, but instead of wine, they offered water. (Argonauts)

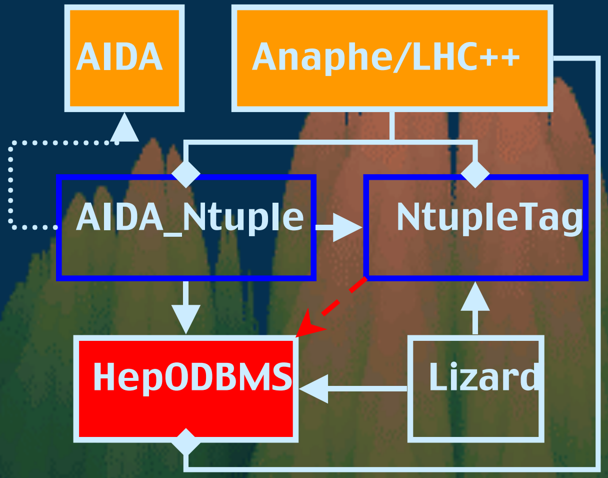
Anaphe/LHC++



- Replacement of CERNLIB
- Standard solutions are used
  - from industry and public domain
  - where appropriate
- Identify and provide key HEP-specific functions

- Primary focus is on C++-based solutions
- The Tag object model is a concept of ANAPHE

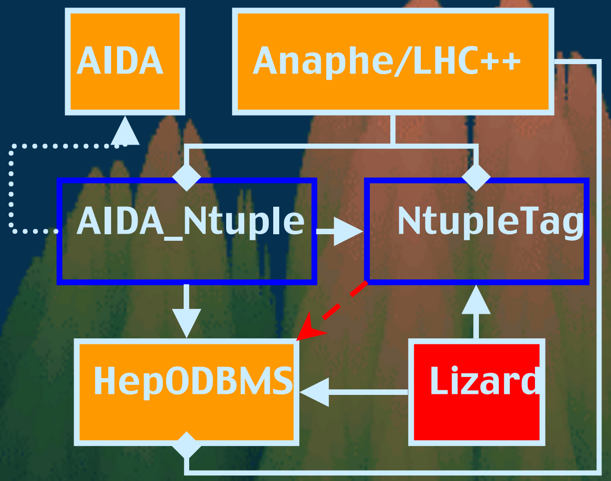
HepODBMS



- Provides a simplified and consistent interface to object database systems
- Offers features important to HEP
- Minimize dependencies on a given database vendor or release

- Features:
  - clustering and locking strategies
  - simplified database session and transaction control
  - event collections, tag database access
  - no significant performance or storage overhead.
  - current implementation is based upon Objectivity/DB
  - location independence
    - moving databases is hidden

Lizard – General



AIDA

Anaphe/LHC++

AIDA_Ntuple

NtupleTag

HepODBMS

Lizard

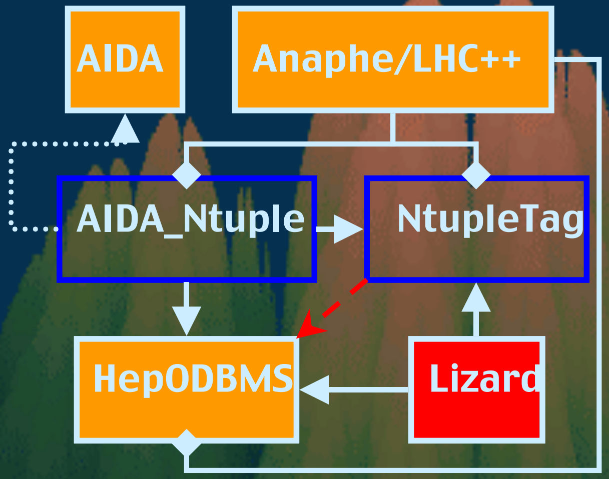
Lizard (noun):

1. Relatively long-bodied reptile with usually two pairs of legs and a tapering tail

2. A man who idles about in the lounges of hotels and bars in search of women who would support him

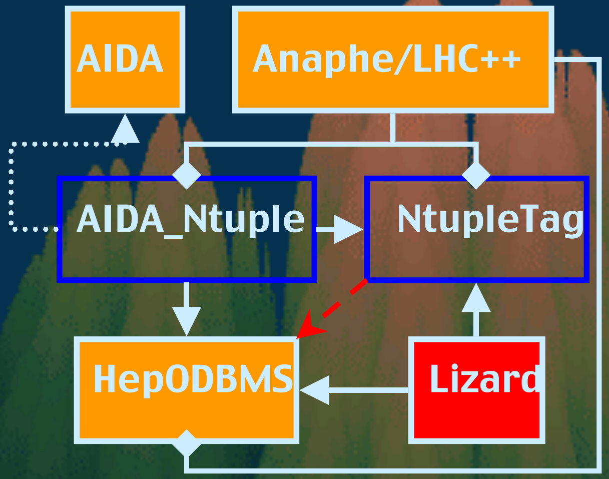
- An **Interactive Analysis Tool**
- Can be easily integrated in a **C++ based environment**
- Functionality is at least comparable with **PAW**
- First release is available since October, 2000

Lizard – Architecture



- **Weak coupling** between components
  - via pure **abstract interfaces**
  - implementation is in **separate dynamically loaded libraries**
  - **plugin** structure
- Relies on the set of **Anaphe libraries**
- **Components** are developed **independently**
- Keeping the structure **open for future extensions**
  - by the developers and by the users
- Using of design **patterns**

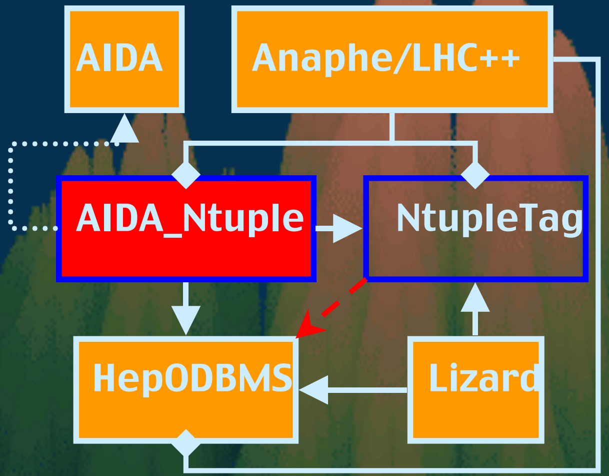
Lizard – Features



- Provides an environment for 'end user analysis'
  - Command-line interface to reach AIDA functionality
  - for physicist

- Usage of a scripting language (now Python)
- Compiling and executing user-generated code on the fly (Analyzer)
- visualization of data

AIDA_Ntuple

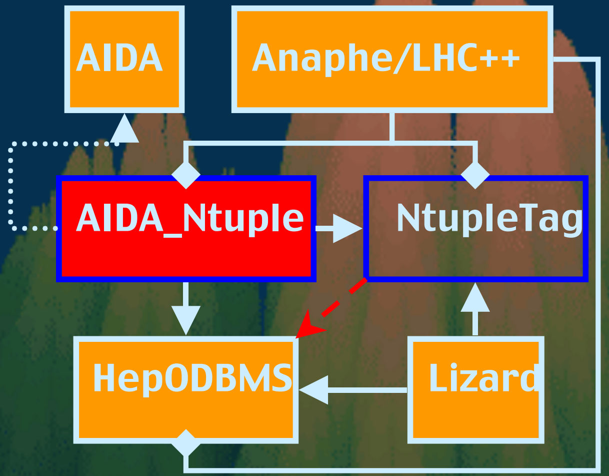


- A **command line interface** is provided by Lizard

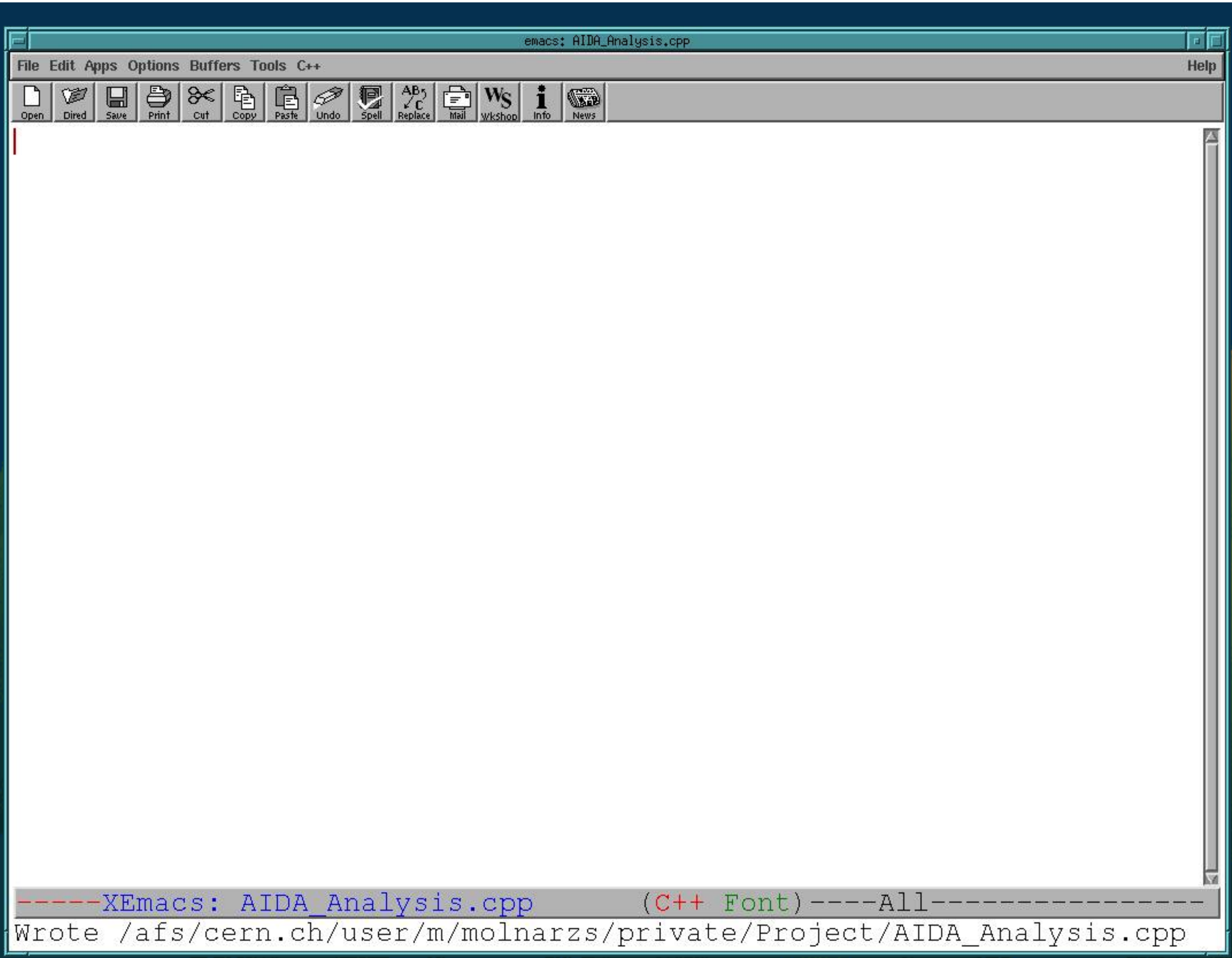
- **High level** ntuple analysis
  - for end users
- High level **ntuple analysis**
  - no dependencies on implementation present in the interfaces (tags, persistency, etc.)

Zsolt Molnár, CERN/IT, Zsolt.Molnar@cern.ch

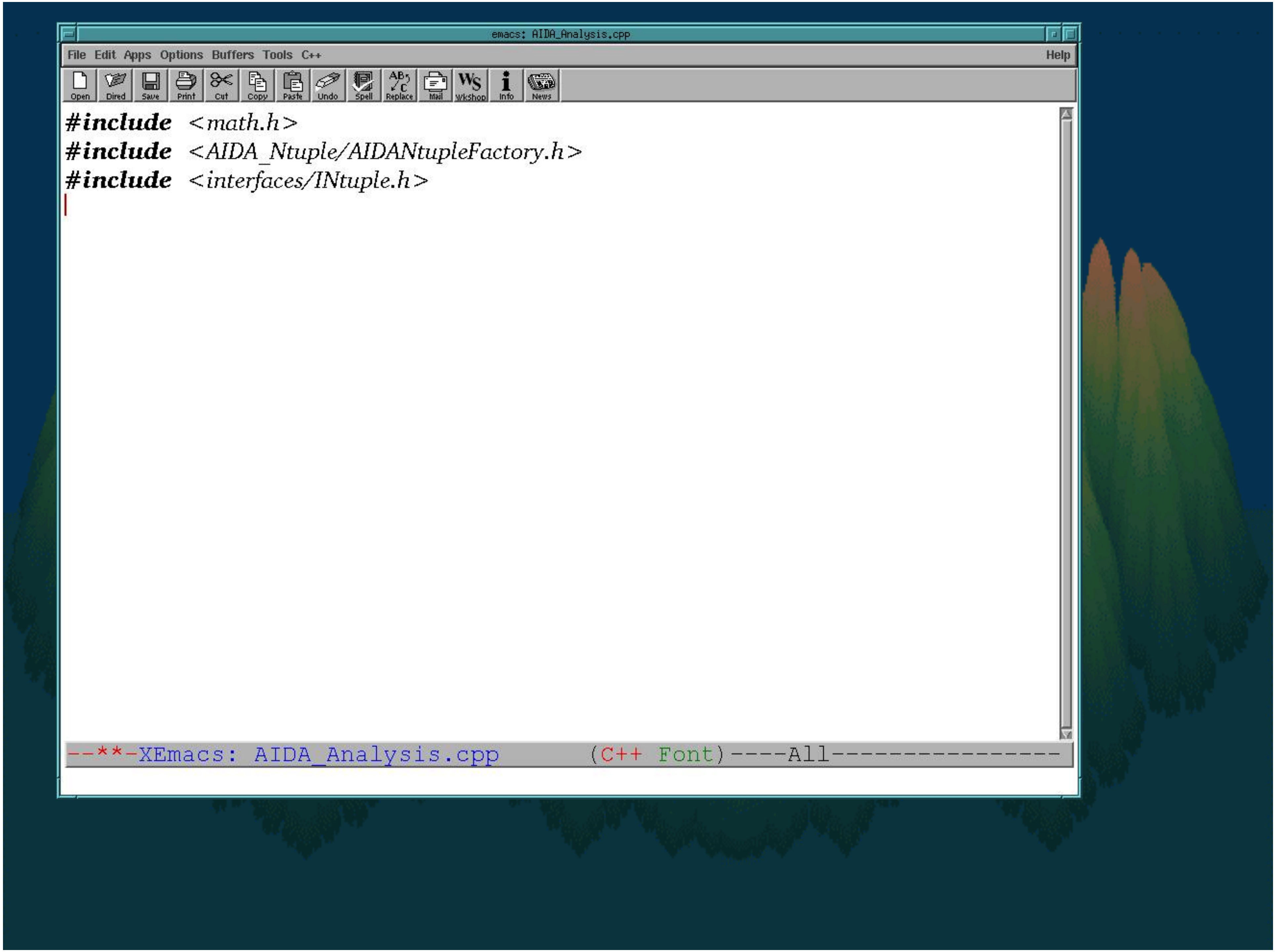
AIDA_Ntuple



- Uses the manager->factory->baseClass pattern
- The implementation
  - uses the low level ntuple functionality of NtupleTag
- Analysis:
  - project (scan) arbitrary mathematical expressions of attributes using selections (cuts)
  - cut is an arbitrary logical-valued mathematical expression of attributes
  - expressions and cuts are expressed in C++ syntax
  - example: `(sin(Energy) > 0.8)||(sin(Energy) <= 0.2)`

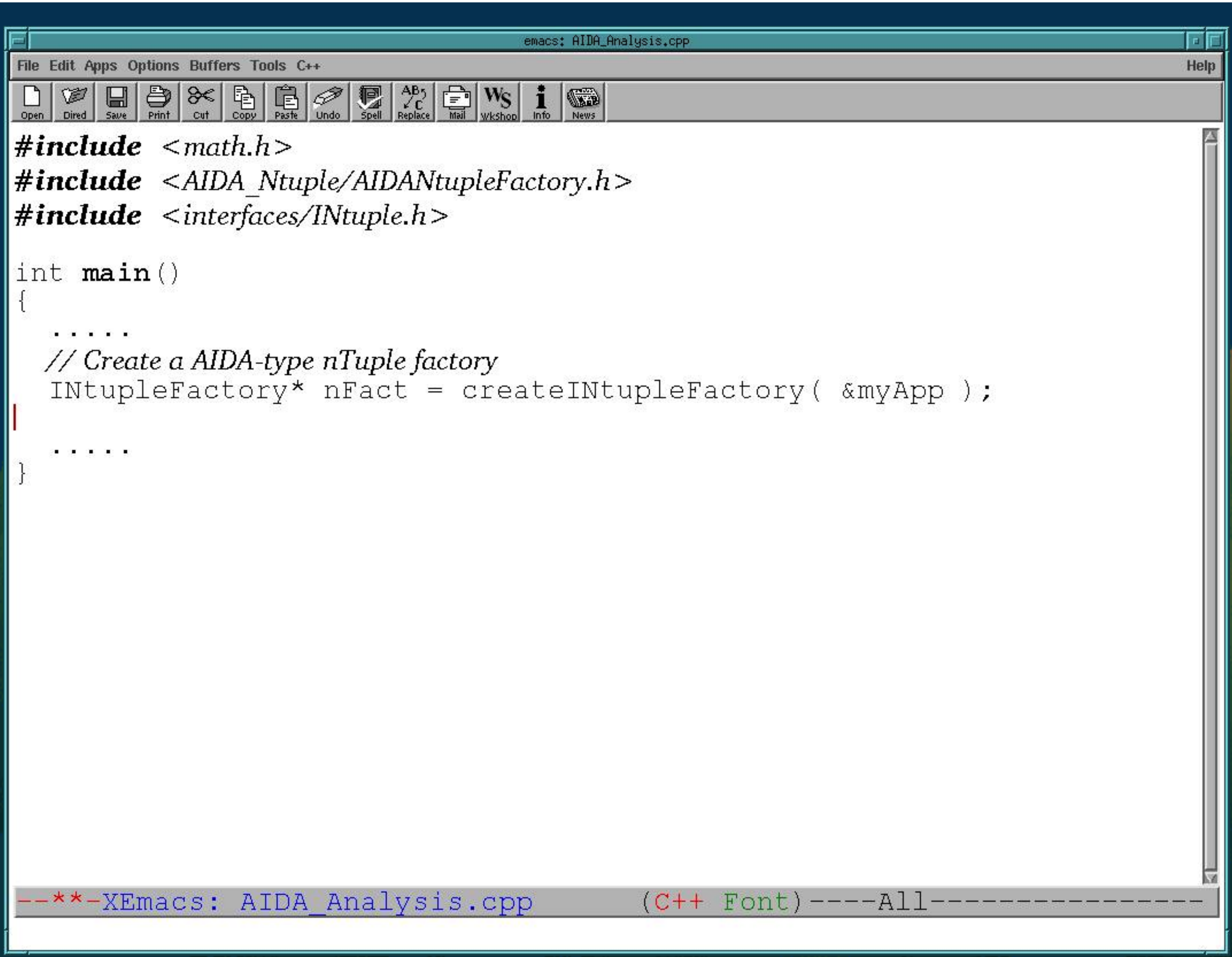

emacs: AIDA_Analysis.cpp
File Edit Apps Options Buffers Tools C++
Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail WkShop Info News

```
-----XEmacs: AIDA_Analysis.cpp        (C++ Font)----All----------------
Wrote /afs/cern.ch/user/m/molnarzs/private/Project/AIDA_Analysis.cpp
```

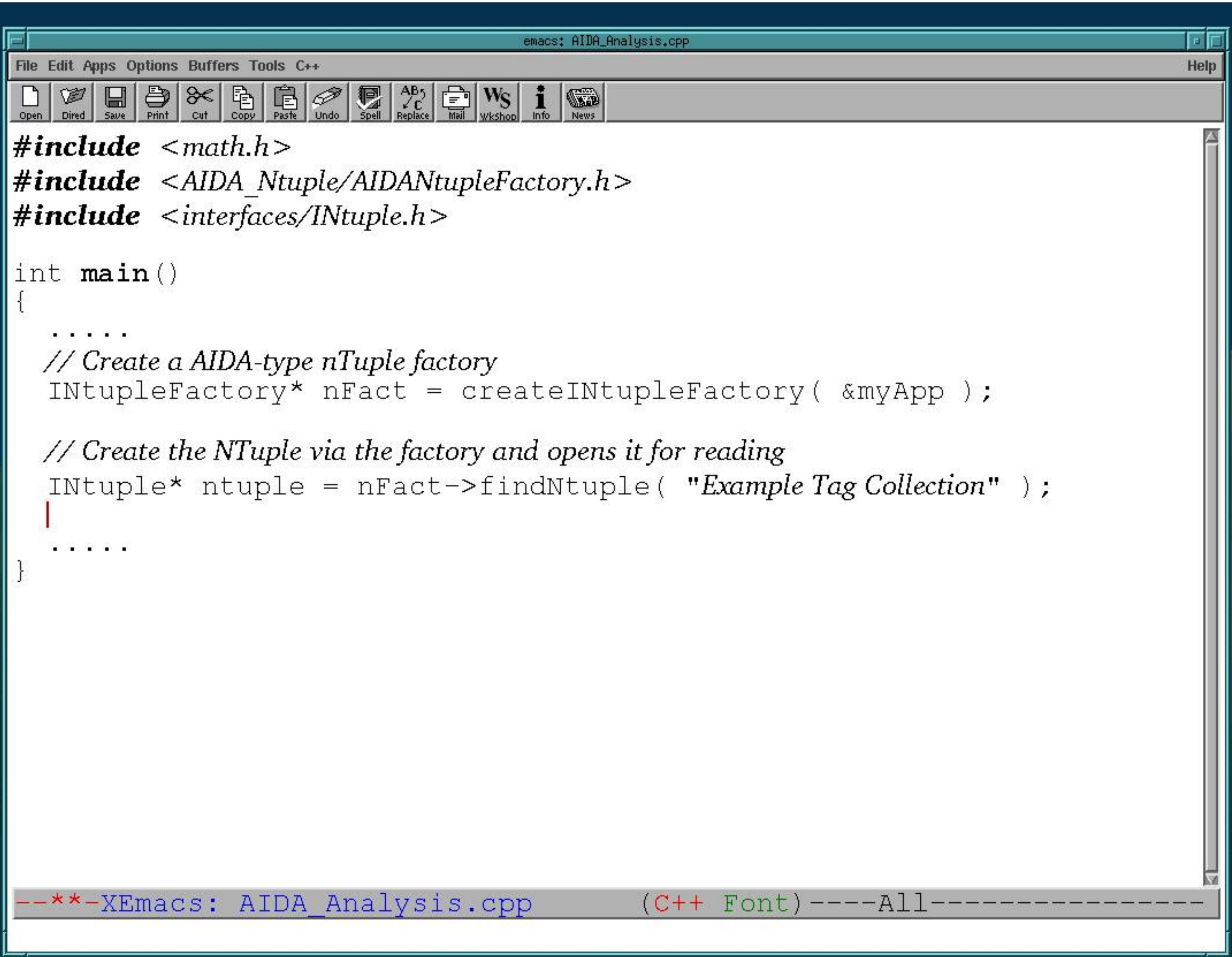


```
#include <math.h>
#include <AIDA_Ntuple/AIDANtupleFactory.h>
#include <interfaces/INtuple.h>
```

```
--**-XEmacs: AIDA_Analysis.cpp      (C++ Font)----All----------------
```


emacs: AIDA_Analysis.cpp
File Edit Apps Options Buffers Tools C++
Help
Open
Dired
Save
Print
Cut
Copy
Paste
Undo
Spell
Replace
Mail
WkShop
Info
News
--**-XEmacs: AIDA_Analysis.cpp (C++ Font)----All----------------

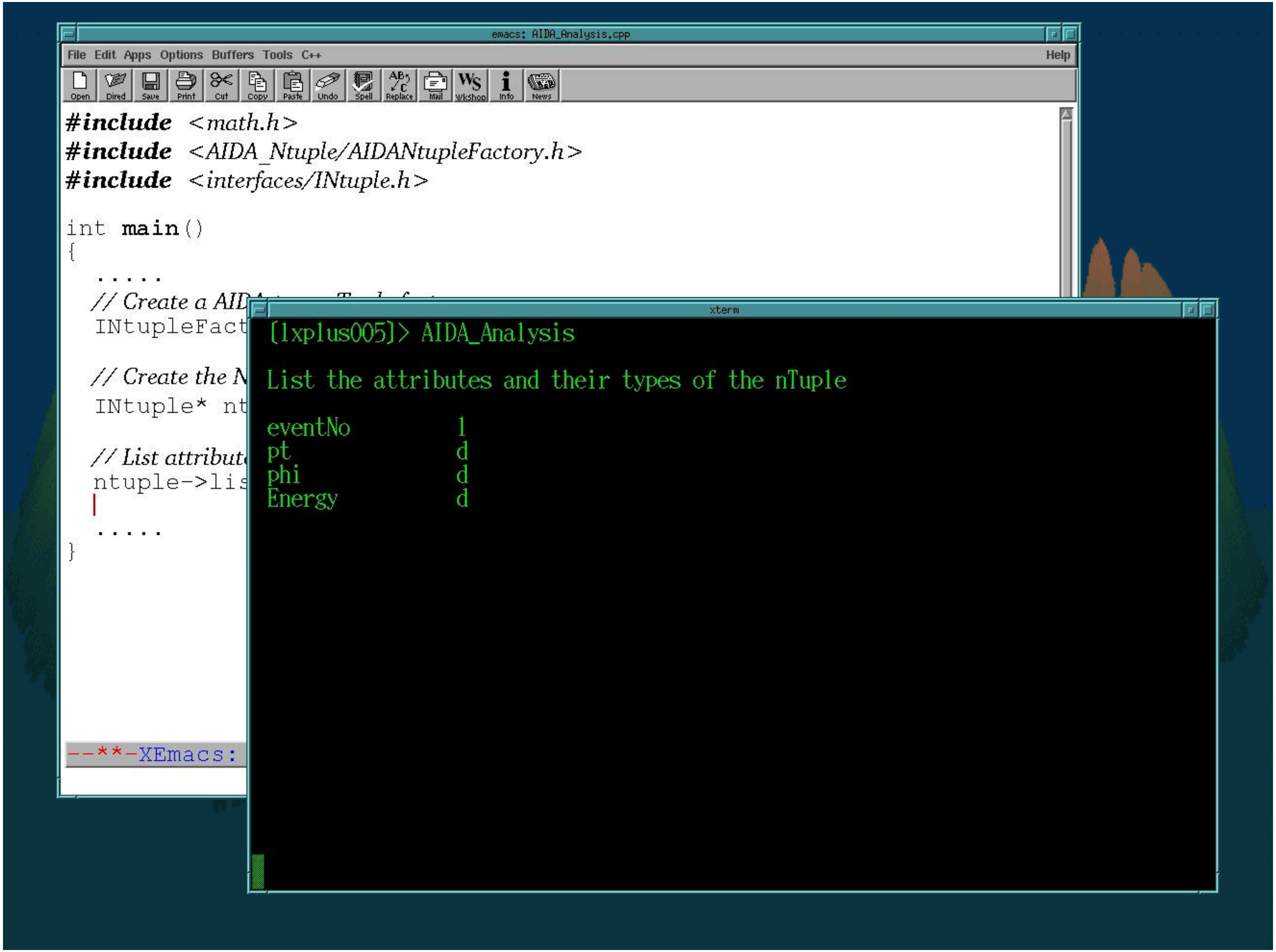
```
#include <math.h>
#include <AIDA_Ntuple/AIDANtupleFactory.h>
#include <interfaces/INtuple.h>

int main()
{
   .....
  // Create a AIDA-type nTuple factory
   INtupleFactory* nFact = createINtupleFactory( &myApp );

   .....
}
```



emacs: AIDA_Analysis.cpp
File Edit Apps Options Buffers Tools C++ Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail WkShop Info News

```
#include <math.h>
#include <AIDA_Ntuple/AIDANtupleFactory.h>
#include <interfaces/INtuple.h>

int main()
{
  .....
  // Create a AIDA-type nTuple factory
  INtupleFactory* nFact = createINtupleFactory( &myApp );

  // Create the NTuple via the factory and opens it for reading
  INtuple* ntuple = nFact->findNtuple( "Example Tag Collection" );

  .....
}
```

--**-XEmacs: AIDA_Analysis.cpp (C++ Font)----All----------------

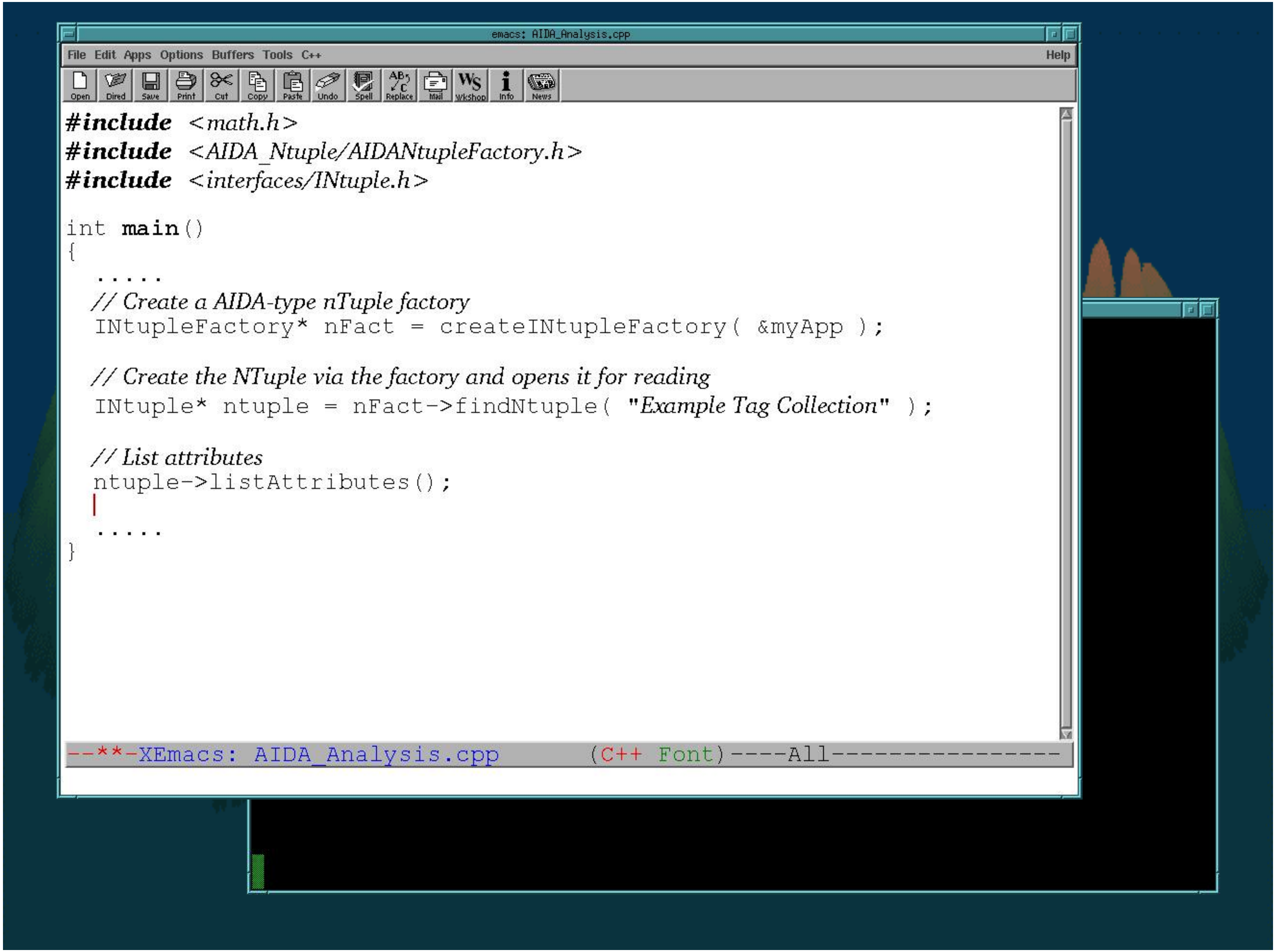


emacs: AIDA_Analysis.cpp
File Edit Apps Options Buffers Tools C++ Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail WkShop Info News
xterm

```
#include <math.h>
#include <AIDA_Ntuple/AIDANtupleFactory.h>
#include <interfaces/INtuple.h>

int main()
{
   .....
   // Create a AID
   INtupleFact

   // Create the N
   INtuple* nt

   // List attribut
   ntuple->lis

   .....
}
```

```
--**-XEmacs:
```

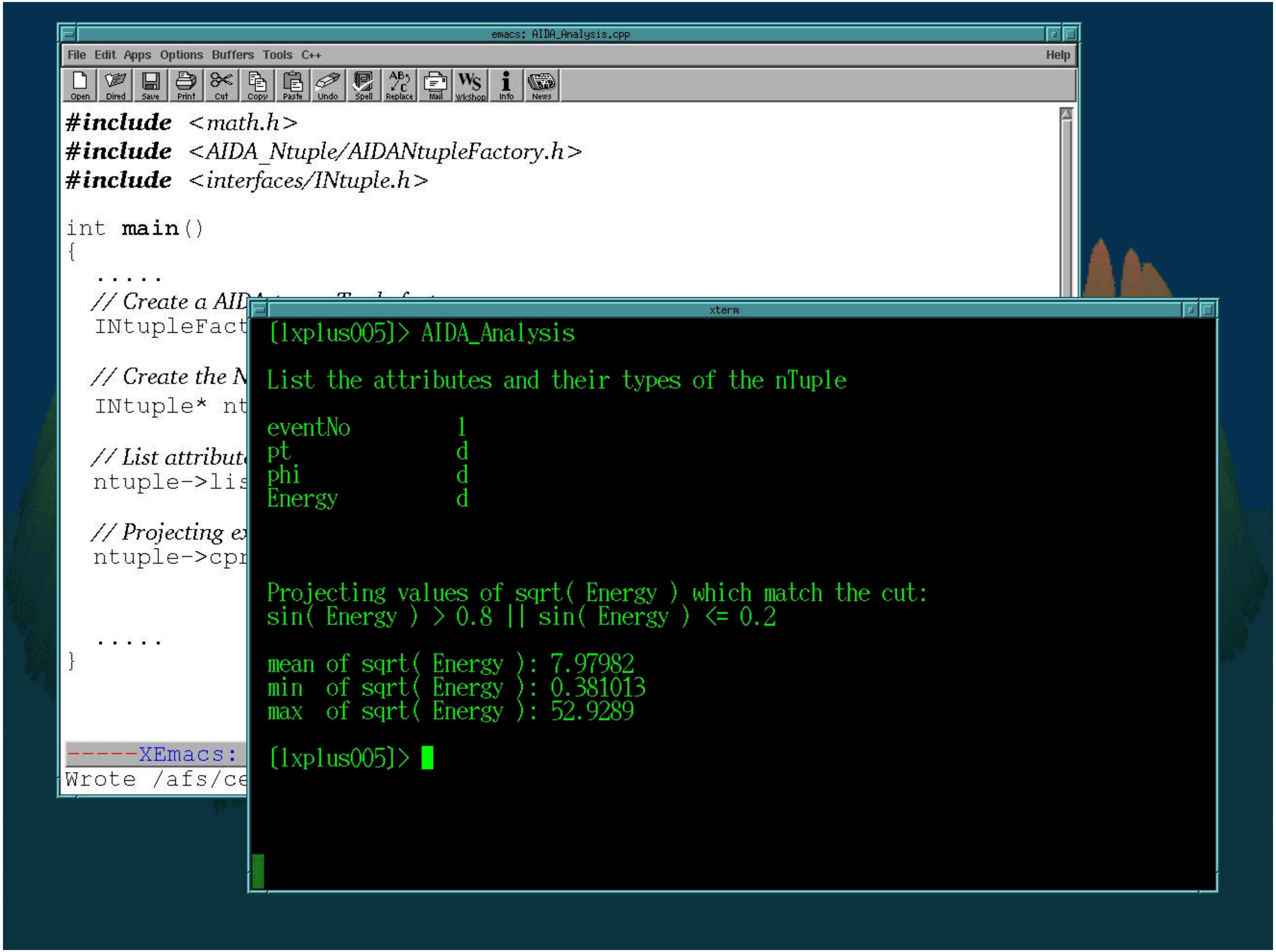
```
[lxplus005]> AIDA_Analysis

List the attributes and their types of the nTuple

eventNo         l
pt              d
phi             d
Energy          d
```



emacs: AIDA_Analysis.cpp
File Edit Apps Options Buffers Tools C++ Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail WkShop Info News

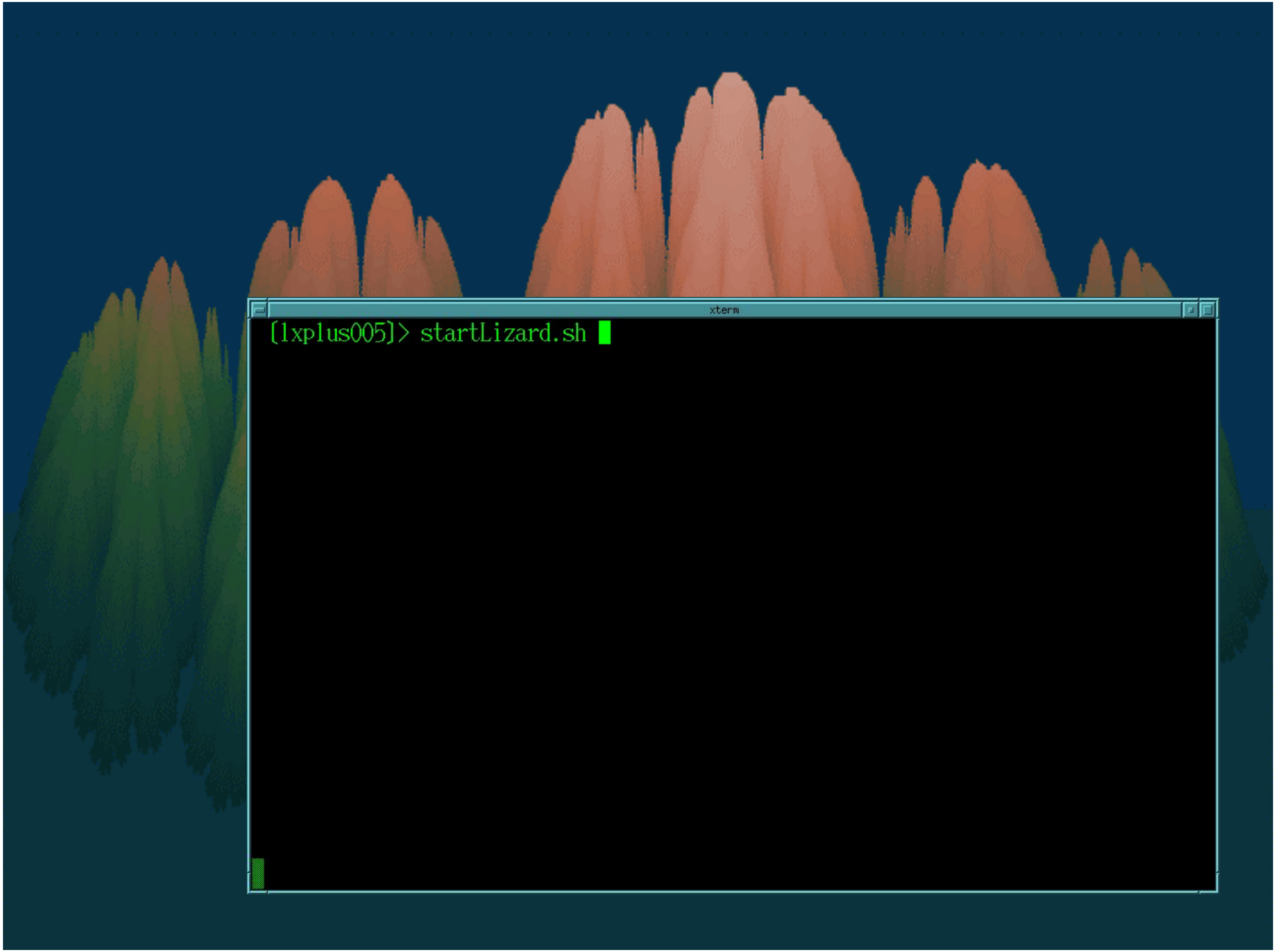
```
#include <math.h>
#include <AIDA_Ntuple/AIDANtupleFactory.h>
#include <interfaces/INtuple.h>

int main()
{
   .....
   // Create a AIDA-type nTuple factory
   INtupleFactory* nFact = createINtupleFactory( &myApp );

   // Create the NTuple via the factory and opens it for reading
   INtuple* ntuple = nFact->findNtuple( "Example Tag Collection" );

   // List attributes
   ntuple->listAttributes();

   .....
}
```

--**-XEmacs: AIDA_Analysis.cpp   (C++ Font)----All----------------

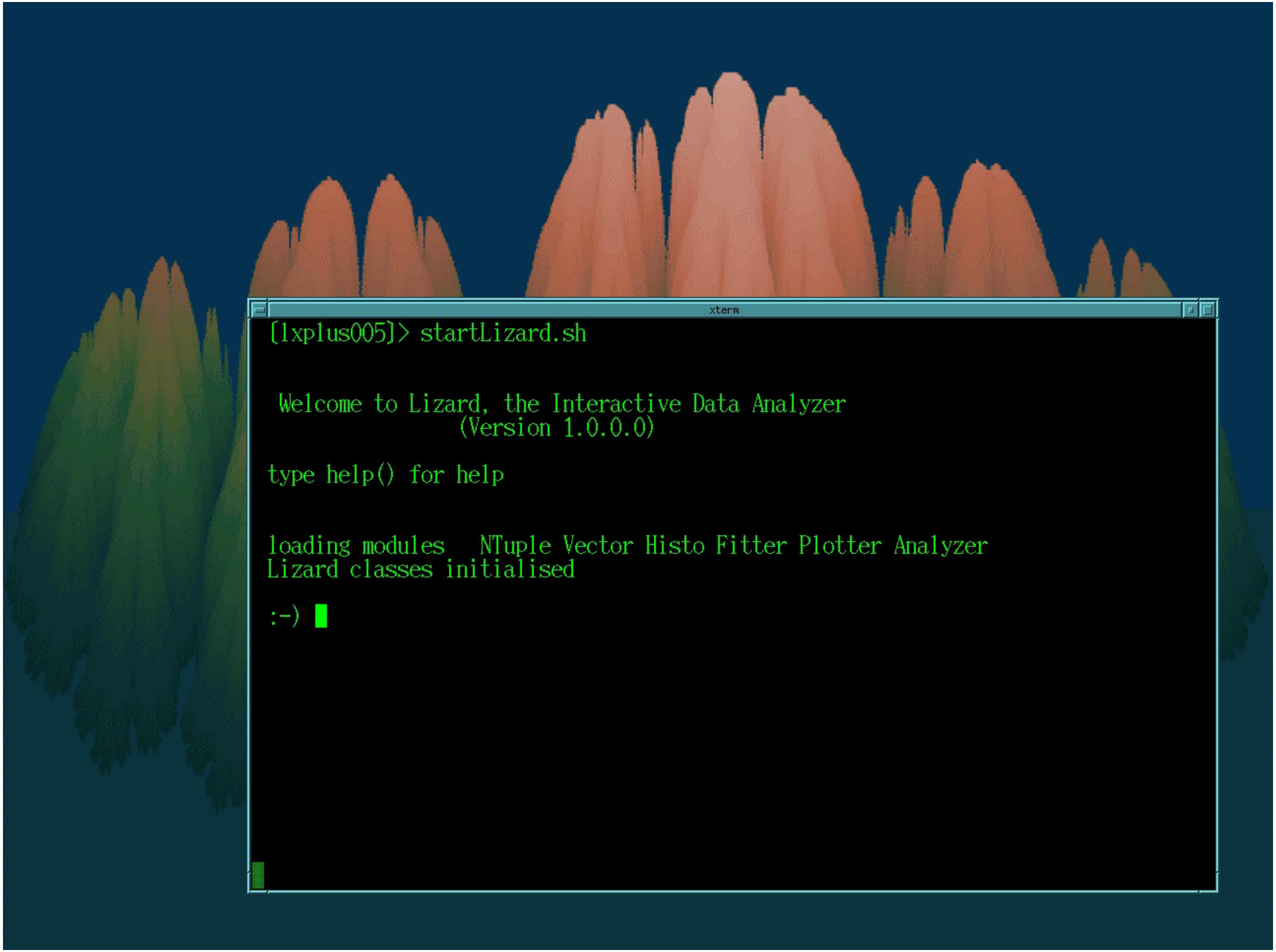

emacs: AIDA_Analysis.cpp
File Edit Apps Options Buffers Tools C++
Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail Wkshop Info News
xterm

```
#include <math.h>
#include <AIDA_Ntuple/AIDANtupleFactory.h>
#include <interfaces/INtuple.h>

int main()
{
  .....
  // Create a AID
  INtupleFact

  // Create the N
  INtuple* nt

  // List attribut
  ntuple->lis

  // Projecting ex
  ntuple->cpr


  .....
}

-----XEmacs:
Wrote /afs/ce
```

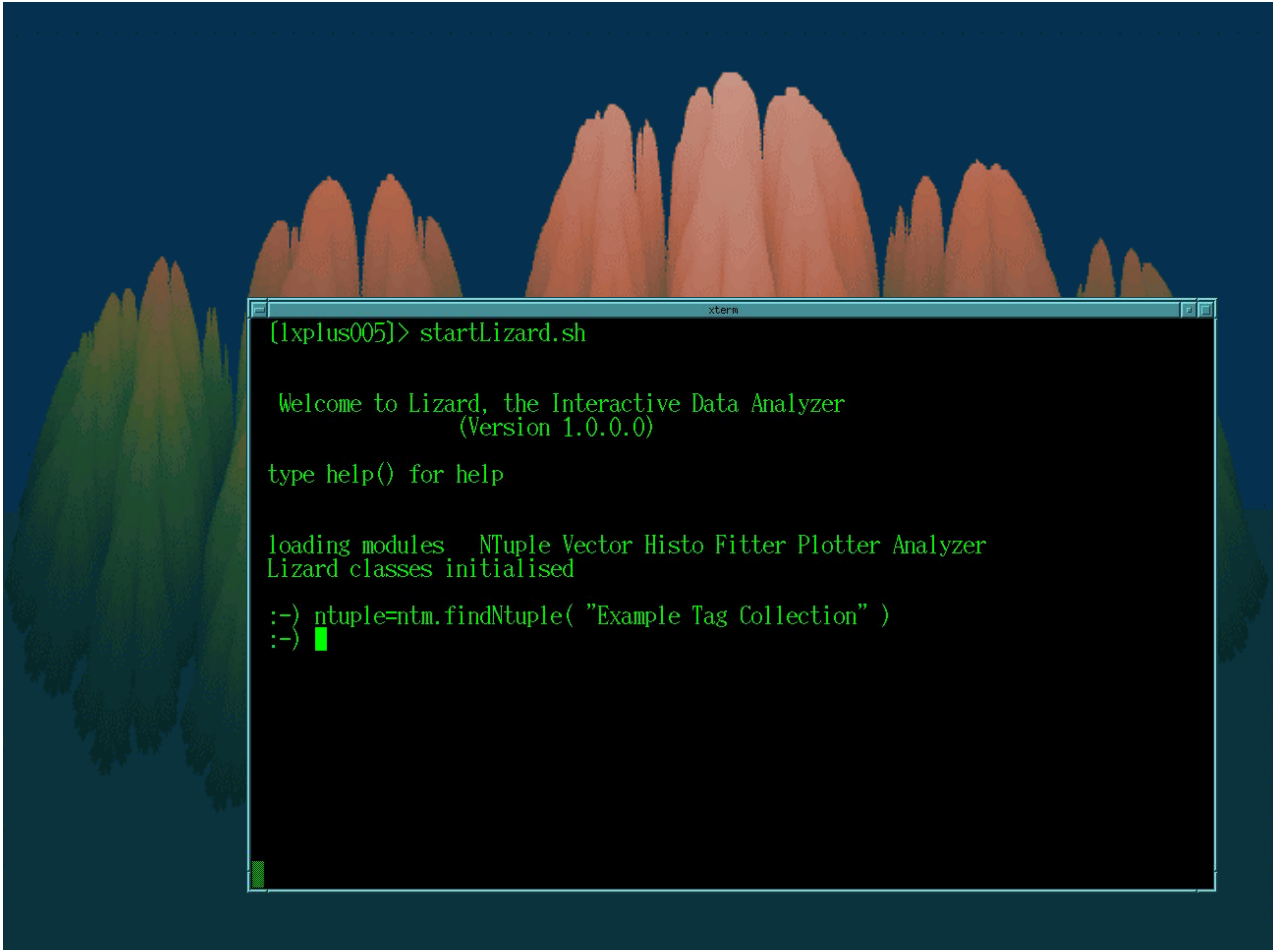
```
[lxplus005]> AIDA_Analysis

List the attributes and their types of the nTuple

eventNo        l
pt             d
phi            d
Energy         d



Projecting values of sqrt( Energy ) which match the cut:
sin( Energy ) > 0.8 || sin( Energy ) <= 0.2

mean of sqrt( Energy ): 7.97982
min  of sqrt( Energy ): 0.381013
max  of sqrt( Energy ): 52.9289

[lxplus005]>
```



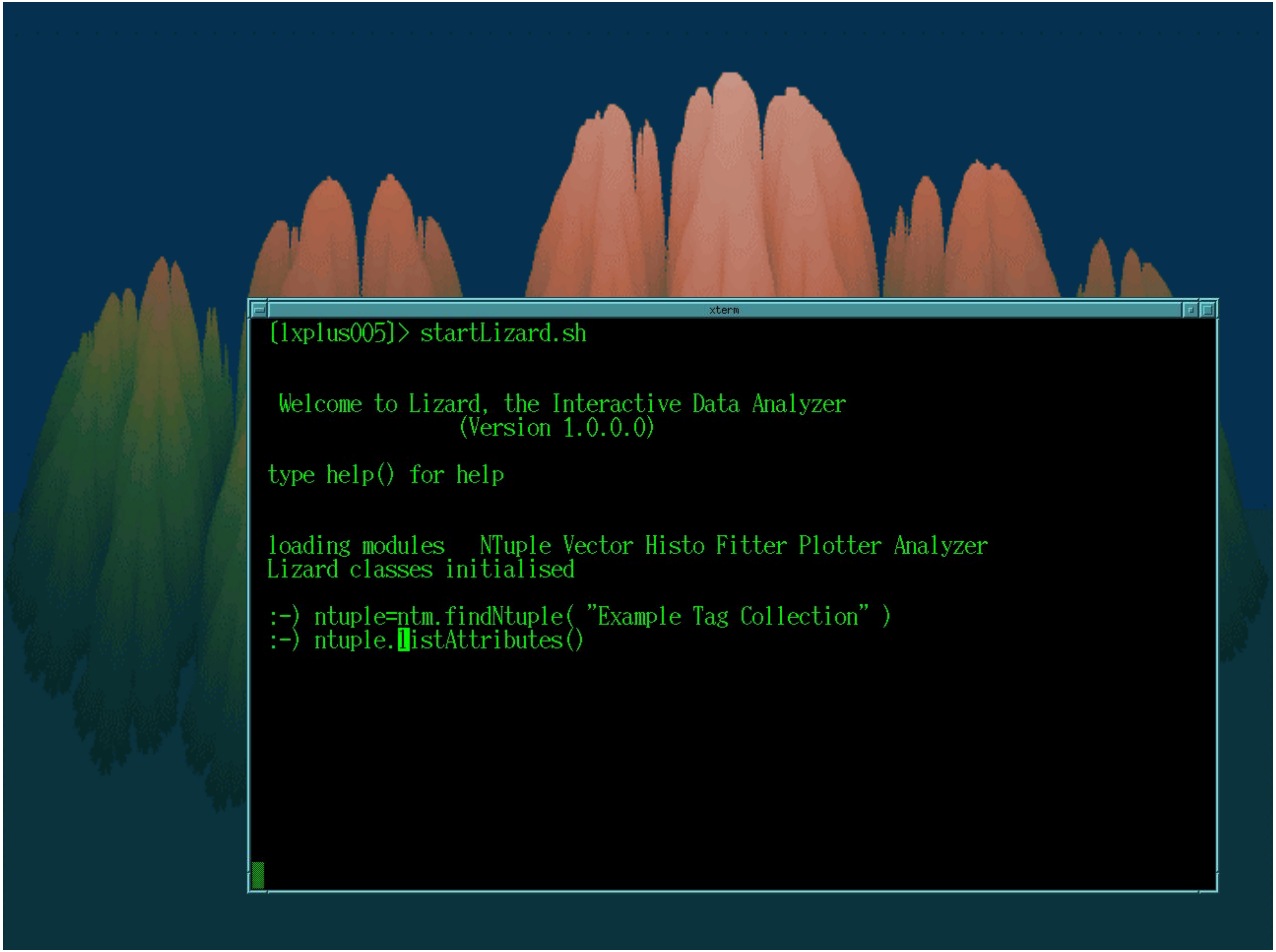
```
[lxplus005]> startLizard.sh
```



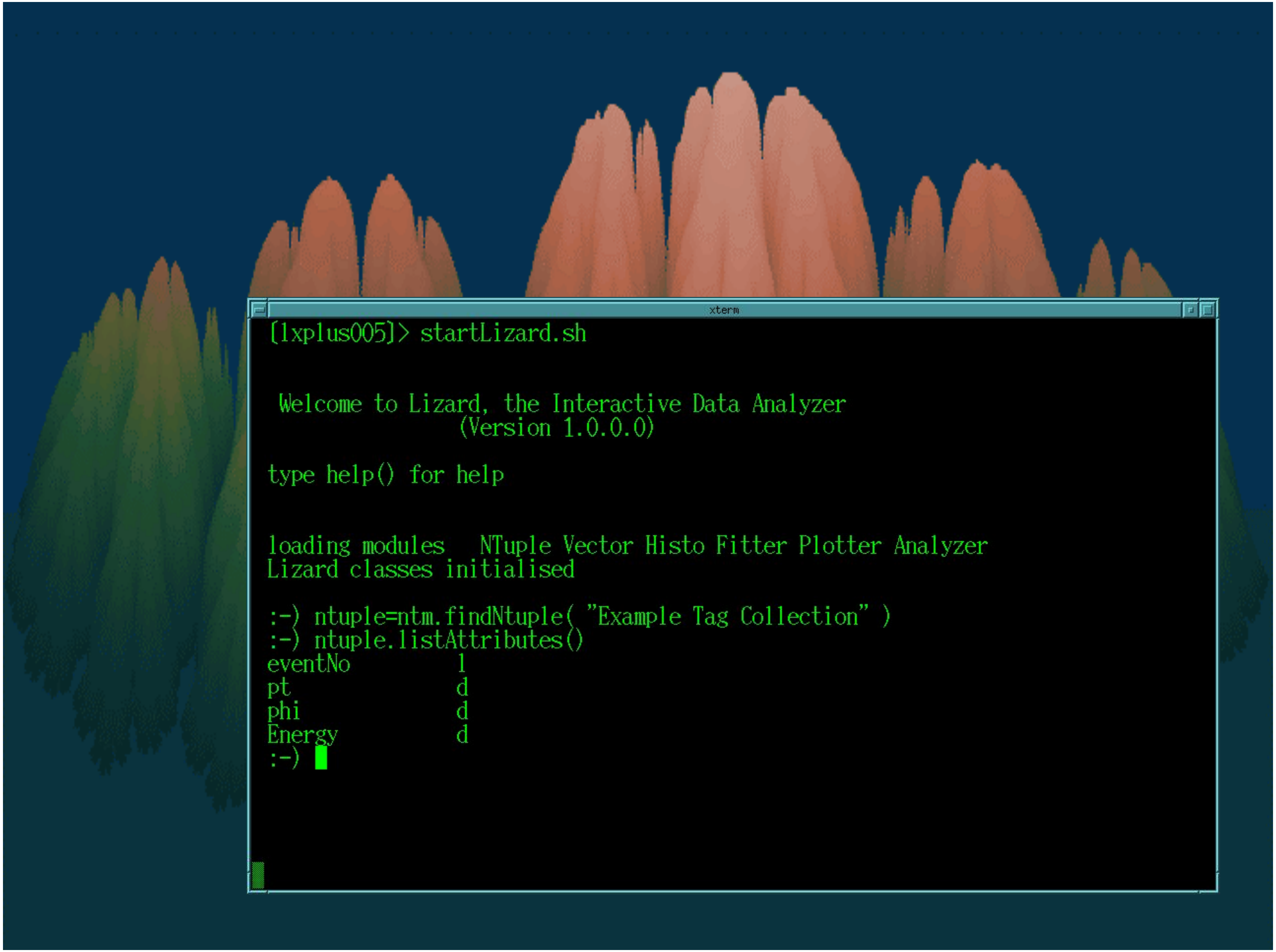
```
[lxplus005]> startLizard.sh


 Welcome to Lizard, the Interactive Data Analyzer
                  (Version 1.0.0.0)

type help() for help


loading modules   NTuple Vector Histo Fitter Plotter Analyzer
Lizard classes initialised

:-)
```



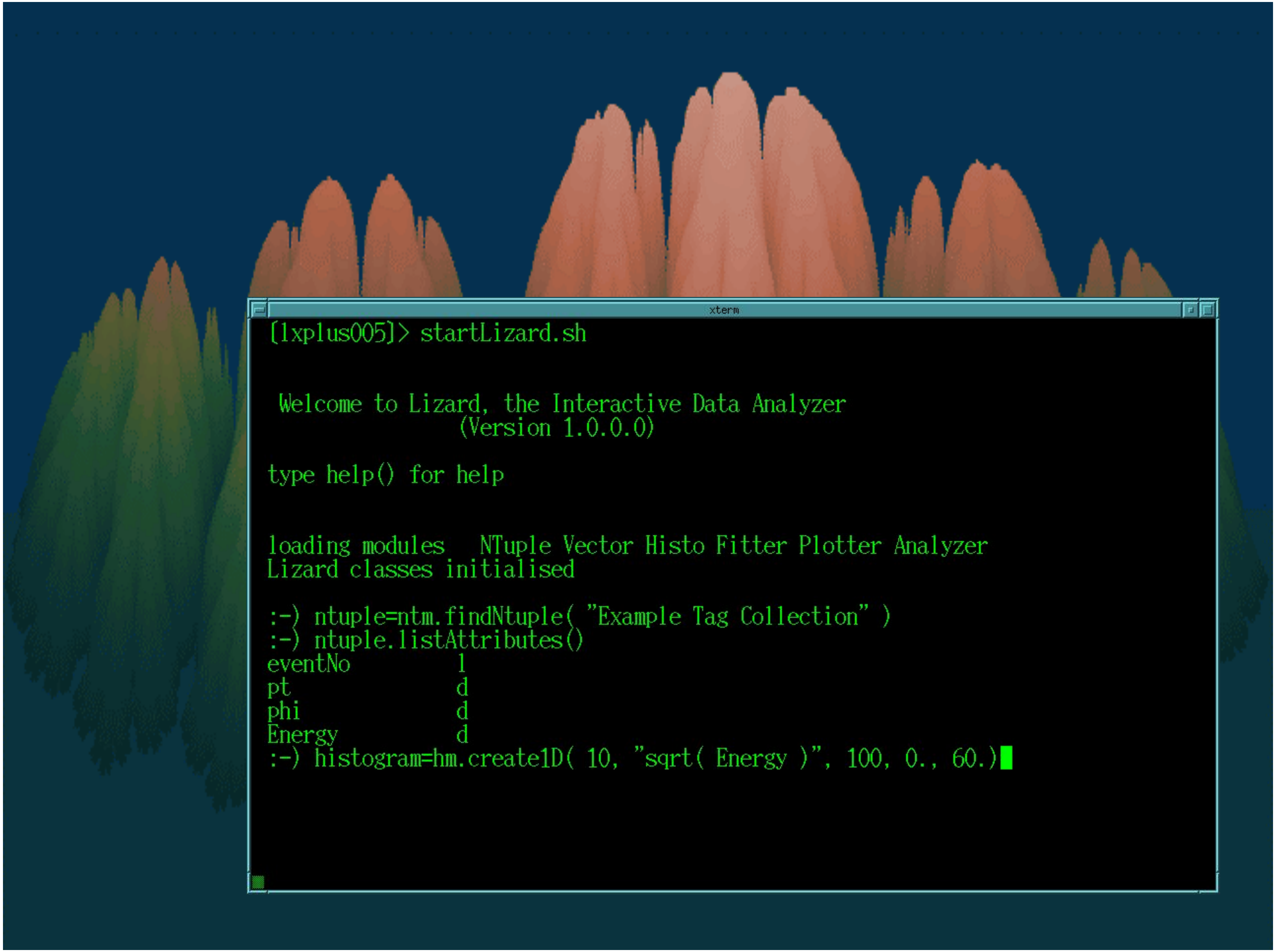
```
[lxplus005]> startLizard.sh


 Welcome to Lizard, the Interactive Data Analyzer
                (Version 1.0.0.0)

type help() for help


loading modules   NTuple Vector Histo Fitter Plotter Analyzer
Lizard classes initialised

:-) ntuple=ntm.findNtuple( "Example Tag Collection" )
:-)
```



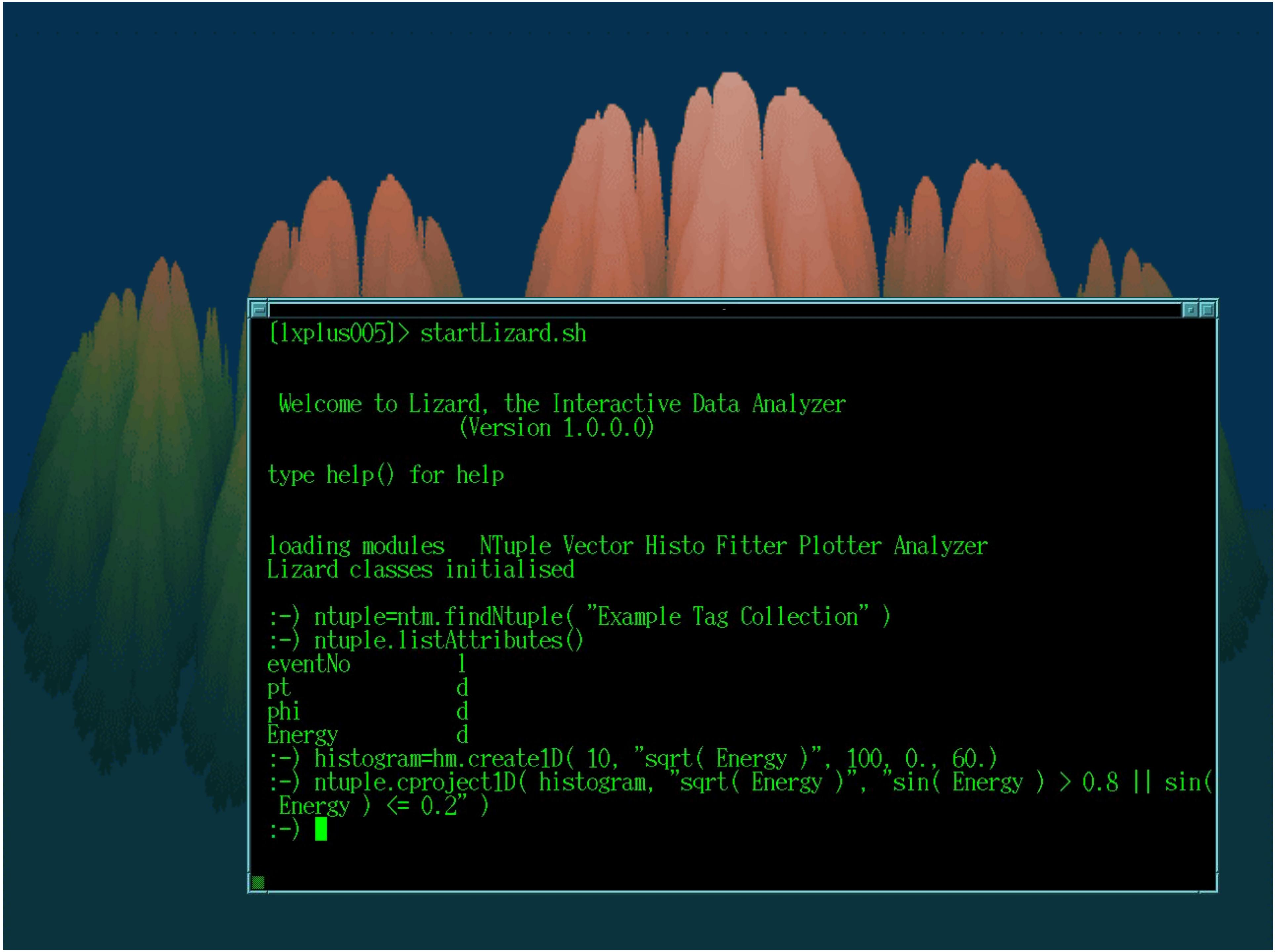
```
[lxplus005]> startLizard.sh


 Welcome to Lizard, the Interactive Data Analyzer
                  (Version 1.0.0.0)

type help() for help


loading modules   NTuple Vector Histo Fitter Plotter Analyzer
Lizard classes initialised

:-) ntuple=ntm.findNtuple( "Example Tag Collection" )
:-) ntuple.listAttributes()
```



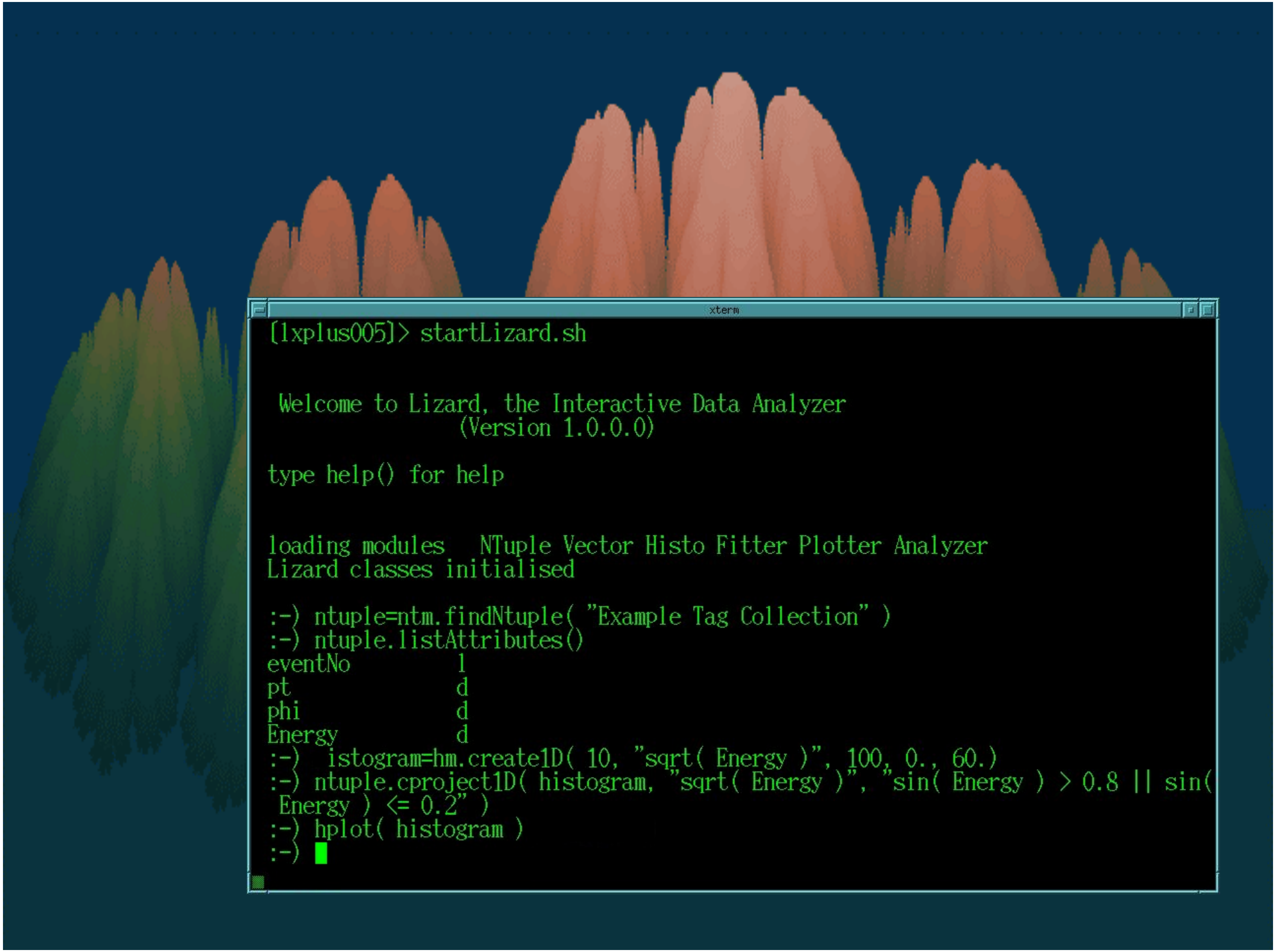
```
[lxplus005]> startLizard.sh


 Welcome to Lizard, the Interactive Data Analyzer
                (Version 1.0.0.0)

type help() for help


loading modules   NTuple Vector Histo Fitter Plotter Analyzer
Lizard classes initialised

:-) ntuple=ntm.findNtuple( "Example Tag Collection" )
:-) ntuple.listAttributes()
eventNo         l
pt              d
phi             d
Energy          d
:-) 
```



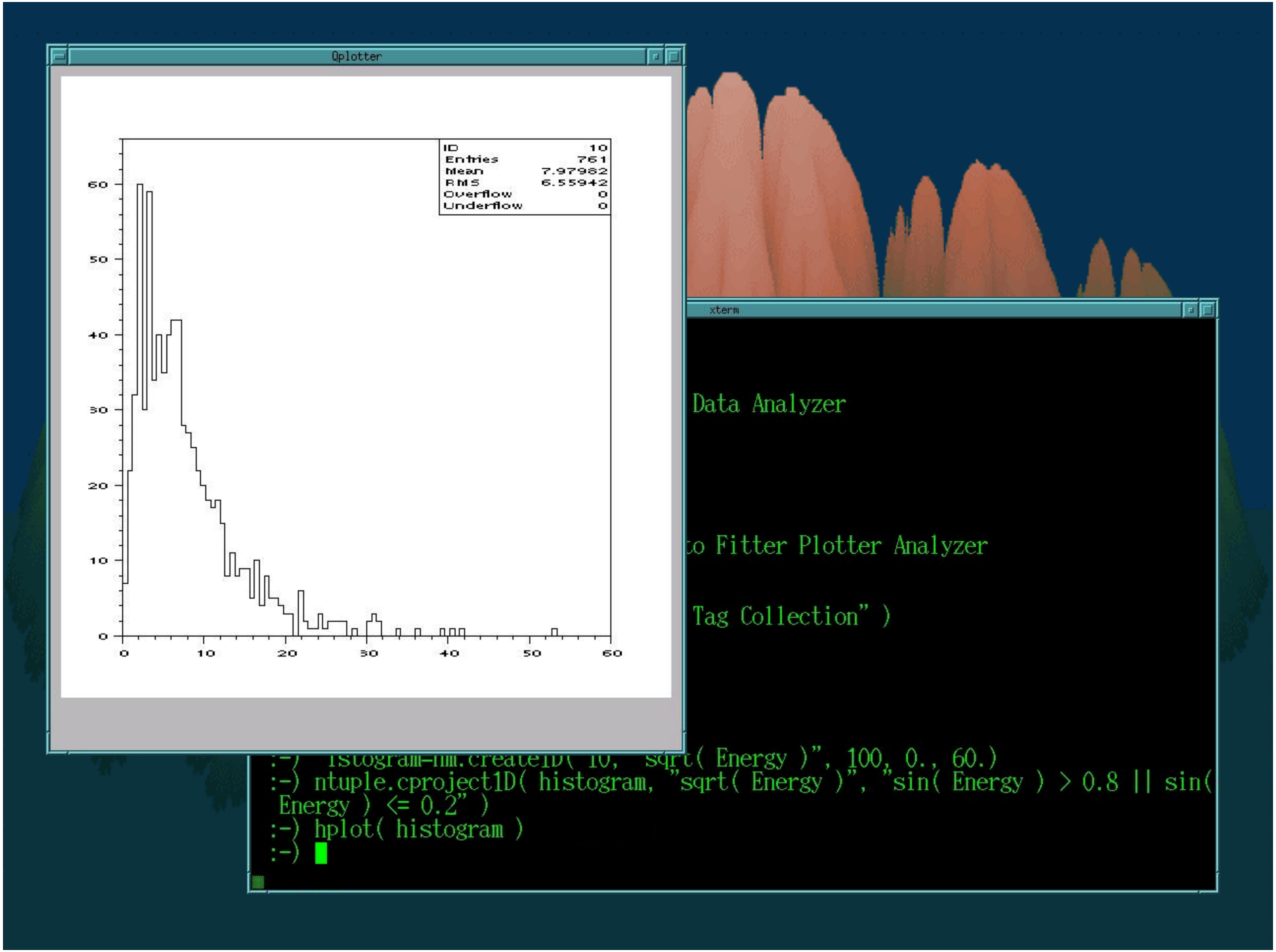
```
[lxplus005]> startLizard.sh


 Welcome to Lizard, the Interactive Data Analyzer
                (Version 1.0.0.0)

type help() for help


loading modules   NTuple Vector Histo Fitter Plotter Analyzer
Lizard classes initialised

:-) ntuple=ntm.findNtuple( "Example Tag Collection" )
:-) ntuple.listAttributes()
eventNo         l
pt              d
phi             d
Energy          d
:-) histogram=hm.create1D( 10, "sqrt( Energy )", 100, 0., 60.)
```



```
[lxplus005]> startLizard.sh


 Welcome to Lizard, the Interactive Data Analyzer
                (Version 1.0.0.0)

type help() for help


loading modules   NTuple Vector Histo Fitter Plotter Analyzer
Lizard classes initialised

:-) ntuple=ntm.findNtuple( "Example Tag Collection" )
:-) ntuple.listAttributes()
eventNo         l
pt              d
phi             d
Energy          d
:-) histogram=hm.create1D( 10, "sqrt( Energy )", 100, 0., 60.)
:-) ntuple.cproject1D( histogram, "sqrt( Energy )", "sin( Energy ) > 0.8 || sin(
 Energy ) <= 0.2" )
:-) 
```

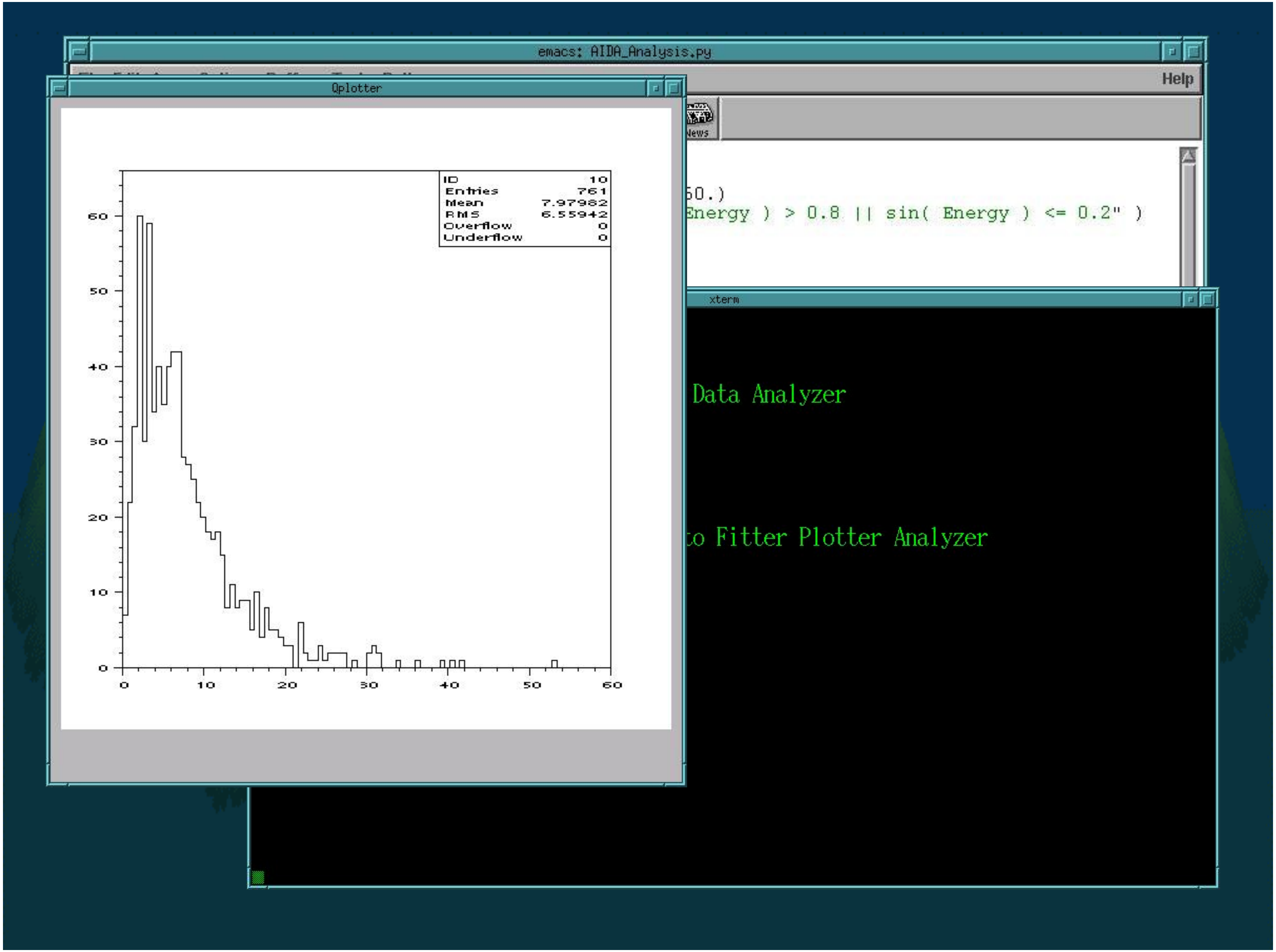


```
[lxplus005]> startLizard.sh


 Welcome to Lizard, the Interactive Data Analyzer
                (Version 1.0.0.0)

type help() for help


loading modules   NTuple Vector Histo Fitter Plotter Analyzer
Lizard classes initialised

:-) ntuple=ntm.findNtuple( "Example Tag Collection" )
:-) ntuple.listAttributes()
eventNo         l
pt              d
phi             d
Energy          d
:-)  istogram=hm.create1D( 10, "sqrt( Energy )", 100, 0., 60.)
:-) ntuple.cproject1D( histogram, "sqrt( Energy )", "sin( Energy ) > 0.8 || sin(
 Energy ) <= 0.2" )
:-) hplot( histogram )
:-) █
```



```
ID                10
Entries          761
Mean         7.97982
RMS          6.55942
Overflow           0
Underflow          0
```

```
Data Analyzer

o Fitter Plotter Analyzer

Tag Collection" )

:-) istogram=hm.create1D( 10, "sqrt( Energy )", 100, 0., 60.)
:-) ntuple.cproject1D( histogram, "sqrt( Energy )", "sin( Energy ) > 0.8 || sin(
 Energy ) <= 0.2" )
:-) hplot( histogram )
:-) █
```

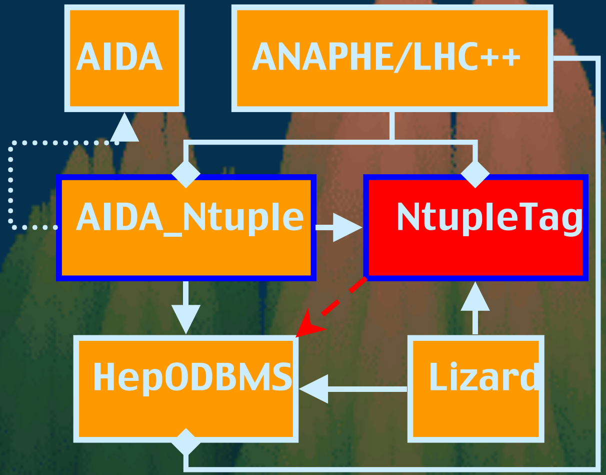


```
50.)
Energy ) > 0.8 || sin( Energy ) <= 0.2" )
```

```
ID              10
Entries        761
Mean       7.97982
RMS        6.55942
Overflow         0
Underflow        0
```

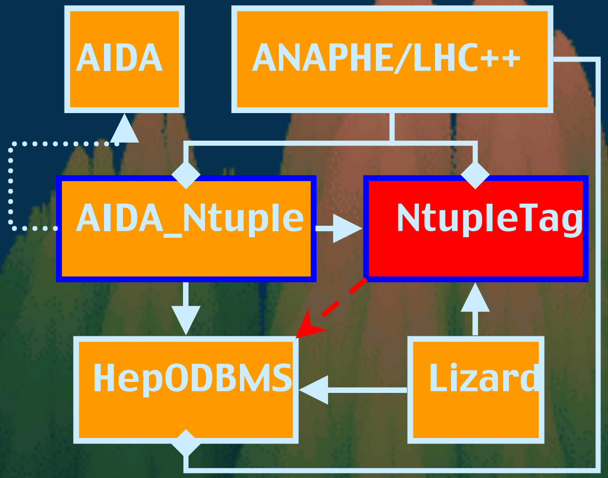
```
Data Analyzer

to Fitter Plotter Analyzer
```

NtupleTag – Overview



- Defines the ntuple as a **data type**
- **low-level** creation, updating and navigation **interface**
- **Implemented using** our **tag** model
  - remember: tag collections are viewed as ntuples
- **Safeness**, **simplicity** and **comfort** for unskilled C++ programmers
  - Physicists would like to do physics

NtupleTag – Implementation

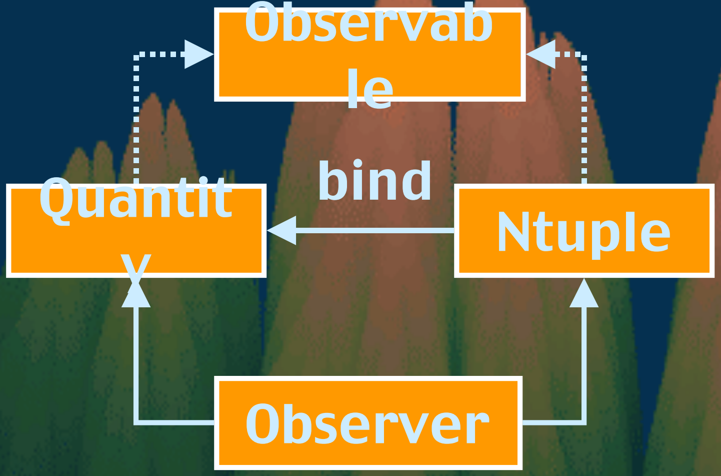


- Using new C++ language features in implementation
  - it is time to make life easier
  - “bad compilers” (ex. on Solaris 4.2) are not supported
- Open to use future data types as attributes
  - Working today with event types of tomorrow
    - just as Bill Gates would like
  - After all… why to use types from computing when analyzing physics data

NtupleTag – Navigation

- Focuses on **looping over the rows** of the ntuple.
  - like projecting, scanning, etc.
- **Navigation** interface **is simple** (but sufficient)
  - begin(), next(), skip(long int), isEnd()
- **Ntuple** has tabular format but it **is not a matrix!**
  - only the **actual row** can be seen
- Access to ntuple attributes is by **Quantities** and **binding**
  - keep things **simple**, **convenient** and **safe**

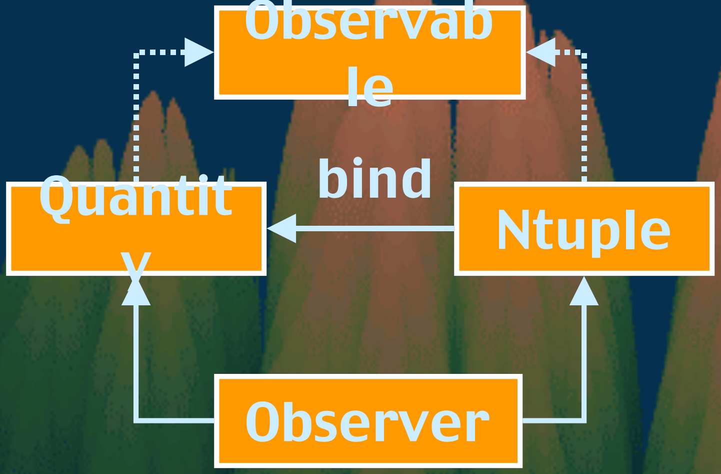
NtupleTag – Quantity and binding



- Binding is a suitable mechanism for looping
- Reflects the value of an attribute automatically

- Much work outside, little work inside a loop
- The entity which is bound to the ntuple is the Quantity

NtupleTag – Quantity and binding



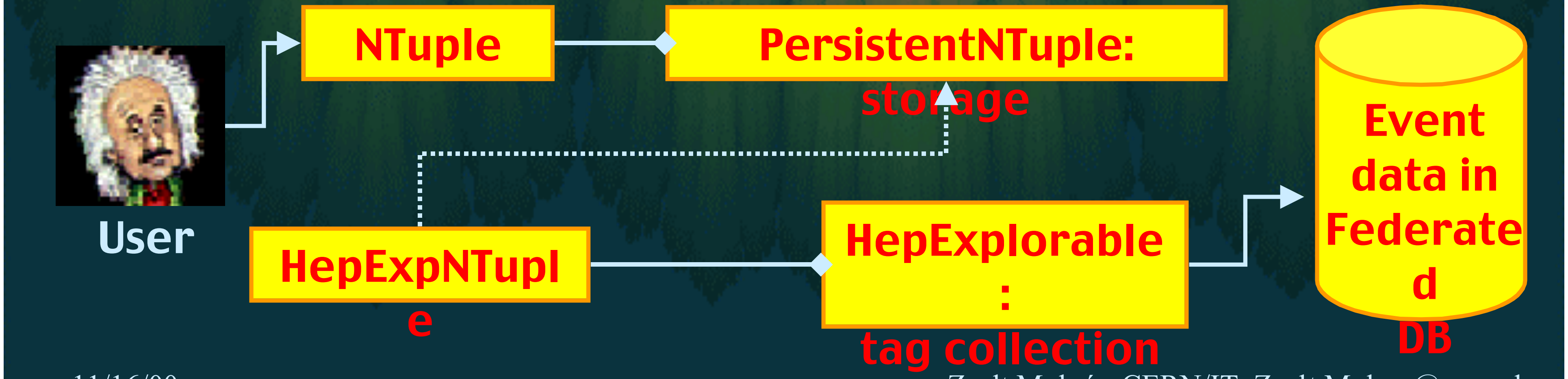
- **WARNING!** Implementation details!
- Binding builds a relationship between independent objects
  - track-keeping is necessary
- Track keeping is provided by using the Observer pattern
  - Lifetime of objects and the relationships between them are fully controlled
  - Observer can automatically delete relationships when an Observable object is deleted
  - Quantities behave as C++ variables

NtupleTag – Storage

- Ntuple is stored in some kind of storage
  - ex. text file; federated database; etc.
- Ntuple is totally free from the type of storage
- Different ntuples == different types of storage
- Storage access goes through PersistentNTuple
  - It is also an abstract class
- Persistent ntuple contains the full data of ntuple.
  - Originally it meant an ntuple stored in persistent store, ex. in a database in a hard disk.
- Architecture supports selective looking of ntuple attributes

NtupleTag – Storage

- Tag collection is a **type of storage**
  - based on HepODBMS
  - this is the current implementation
- By properly implementing PersistentNTuple one can
  - handle HBOOK ntuples or ntuples stored by other DB systems, etc.
  - **run the analysis on all kind of ntuples** if their type structure allows it



User

NTuple

PersistentNTuple: storage

HepExpNTuple

HepExplorable: tag collection

Event data in Federated DB

NtupleTag – RTTI

- RTTI: **Run-Time Type Info**
- RTTI provides the **description of ntuple** independently of implementation
  - with the help of template members an RTTI of C++
- Attributes are identified by their **names**
- **User can choose a type** for an attribute
  - and PersistentNTuple can accept or refuse it
  - ex. Vector to float is invalid, float to float is valid, double to float may be valid
- Only **small overhead** when using RTTI

NtupleTag – References

- Aim at accessing the **whole data/event** from which ntuple is extracted
- A **link** can be set up which is called **Reference**
- A Reference:
  - has the same **Observable** properties like Quantity has
  - acts exactly like **"The Event"**
- Using Reference requires the **full definition of data**
  - standalone program can simply include and link it
  - an interactive analysis environment needs a **plugin-like construct**
    - a simple plugin mechanism is provided by **ExpressionProcessor**

NtupleTag – Expressions

- Expressions are computed by ExpressionProcessors (EP)
  - They also can be some kind of plugins
- ExpressionProcessor can
  - attach an expression to a given ntuple
  - attach external (user) objects necessary for actual analysis
  - compute actual value of an expression
- Expressions could be arranged into “libraries” in a session
  - optimization an reusability in a session

NtupleTag – EP by Compilation

- Compiles and loads expressions on the fly using a C++ compiler
- Provides the execution speed of compiled and optimized C++ code
  - even inside an GUI-controlled environment
- Handles plugins/external (user) objects
- Speed of compilation and linking is very fast
- Example:

```
MASS * reference<MyLorentzVector>->norm() >= 100.0
```

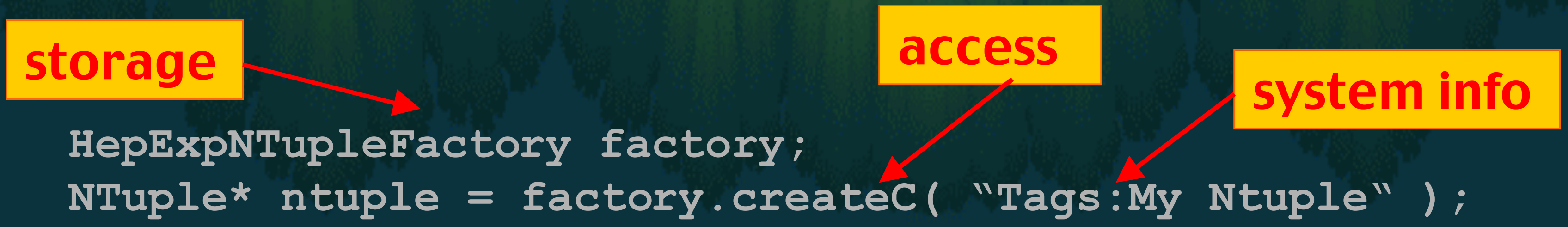
  - detecting the word “reference” starts the plugin system

NtupleTag – Creating/Writing

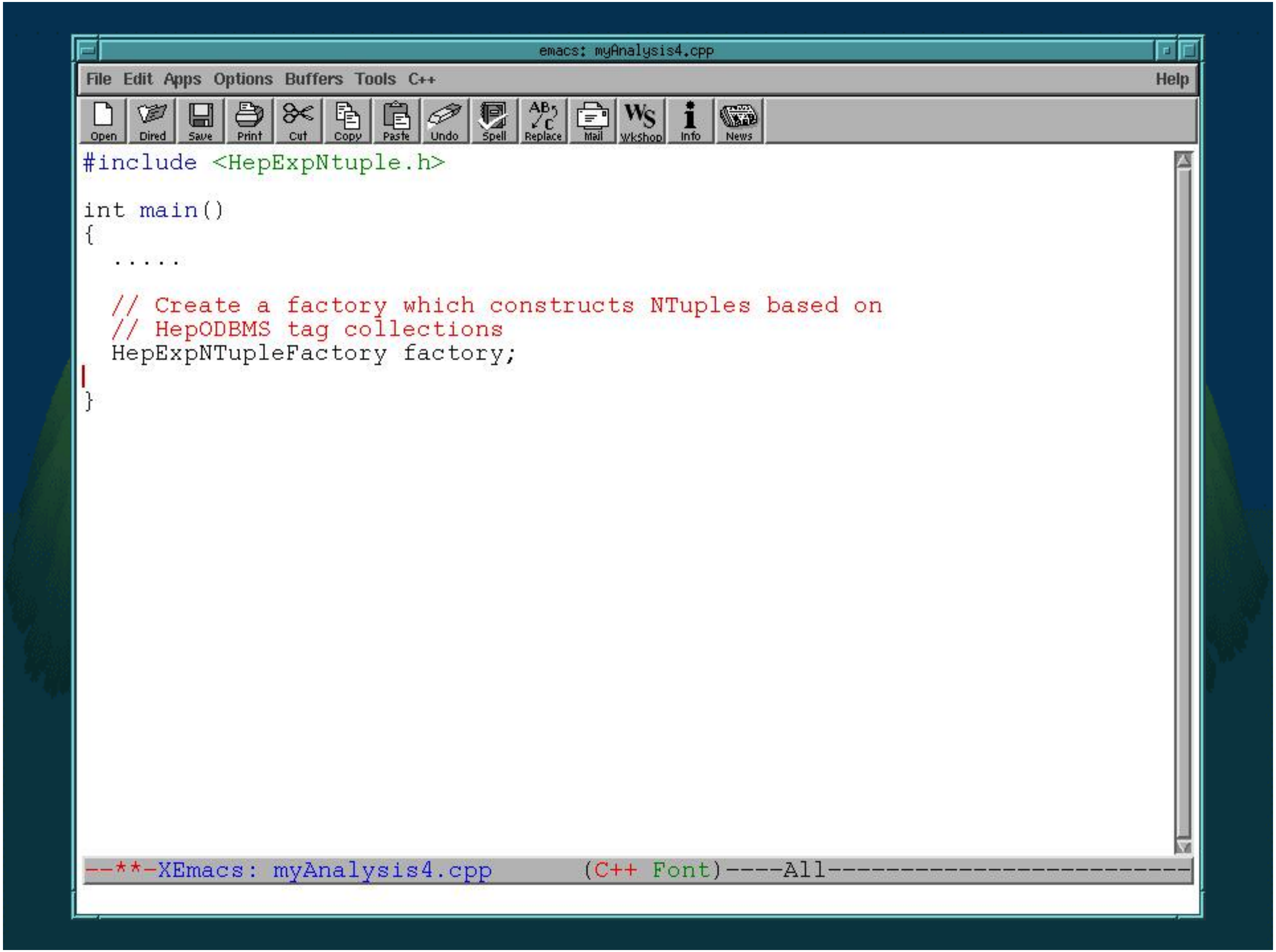
- **Ntuples must be generated before reading**
  - NtupleTag includes the interface performing this task
- **Has the same paradigm as reading has**
  - via the Quantity – Reference – binding – type handling mechanism
  - actual values of bound Quantities are mirrored into the ntuple
- **One navigation system for multiple contexts**
- **Updating and extending ntuples are also possible**
  - add new attributes and/or rows
  - modify existing values

NtupleTag – Factory

- System is informed about storage type and access type by choosing and using Factories
  - storage type <-> factory type
  - access type <-> constructing method
  - Factory is also a design pattern
- The Factory
  - handles and processes the system-dependent information
  - properly creates an ntuple
  - hides the details of system and storage background
  - the resulting ntuple can be analyzed in general way



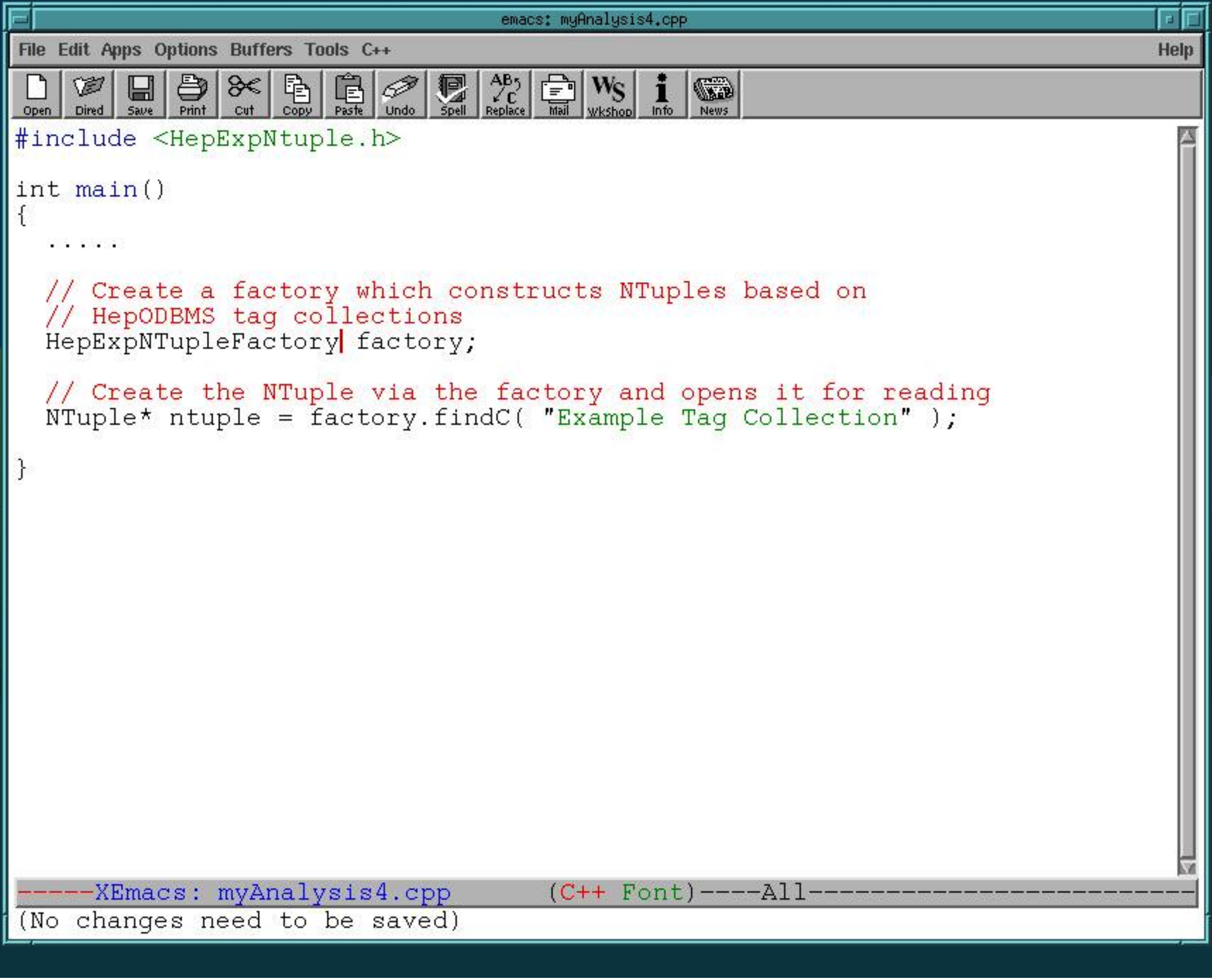
```
HepExpNTupleFactory factory;
NTuple* ntuple = factory.createC( “Tags:My Ntuple“ );
```



emacs: myAnalysis4.cpp
File Edit Apps Options Buffers Tools C++ Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail WkShop Info News

```
#include <HepExpNtuple.h>

int main()
{
  .....

  // Create a factory which constructs NTuples based on
  // HepODBMS tag collections
  HepExpNTupleFactory factory;

}
```

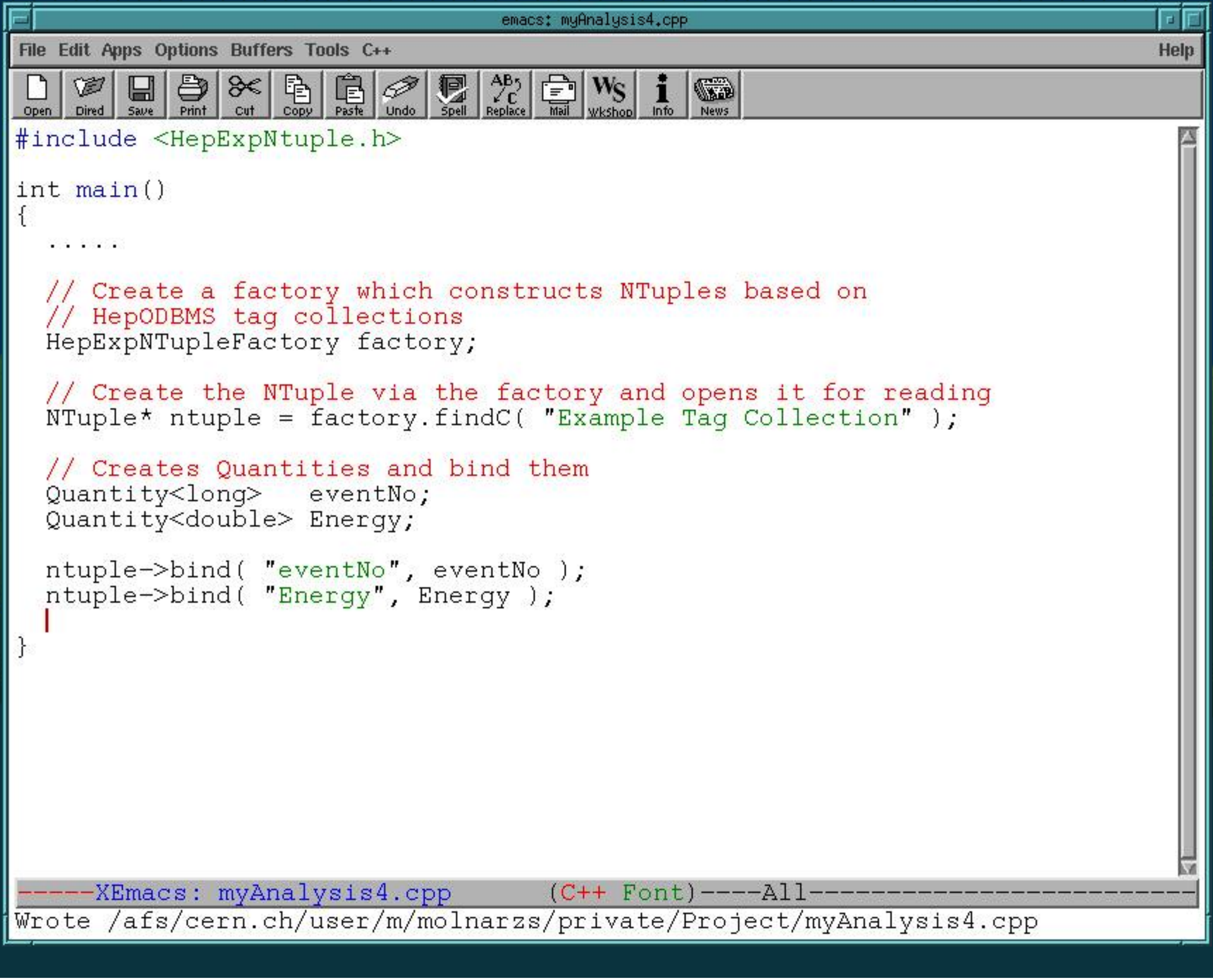
--**-XEmacs: myAnalysis4.cpp   (C++ Font)----All------------------------



emacs: myAnalysis4.cpp
File Edit Apps Options Buffers Tools C++ Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail WkShop Info News

```
#include <HepExpNtuple.h>

int main()
{
  .....

  // Create a factory which constructs NTuples based on
  // HepODBMS tag collections
  HepExpNTupleFactory factory;

  // Create the NTuple via the factory and opens it for reading
  NTuple* ntuple = factory.findC( "Example Tag Collection" );

}
```

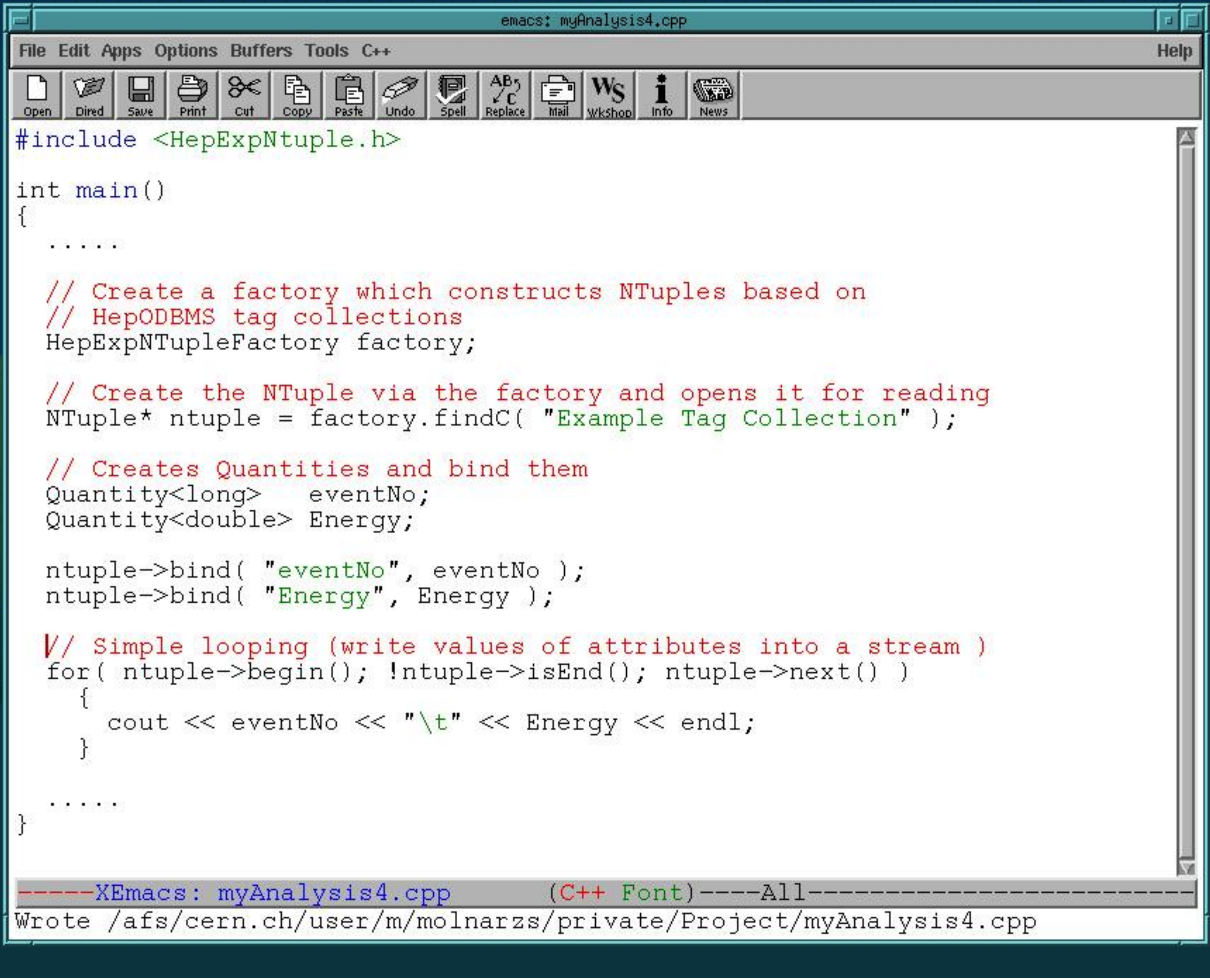
```
-----XEmacs: myAnalysis4.cpp       (C++ Font)----All-------------------------
(No changes need to be saved)
```



emacs: myAnalysis4.cpp
File Edit Apps Options Buffers Tools C++ Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail WkShop Info News

```
#include <HepExpNtuple.h>

int main()
{
  .....

  // Create a factory which constructs NTuples based on
  // HepODBMS tag collections
  HepExpNTupleFactory factory;

  // Create the NTuple via the factory and opens it for reading
  NTuple* ntuple = factory.findC( "Example Tag Collection" );

  // Creates Quantities and bind them
  Quantity<long>   eventNo;
  Quantity<double> Energy;

  ntuple->bind( "eventNo", eventNo );
  ntuple->bind( "Energy", Energy );

}
```

```
-----XEmacs: myAnalysis4.cpp      (C++ Font)----All-----------------------
Wrote /afs/cern.ch/user/m/molnarzs/private/Project/myAnalysis4.cpp
```



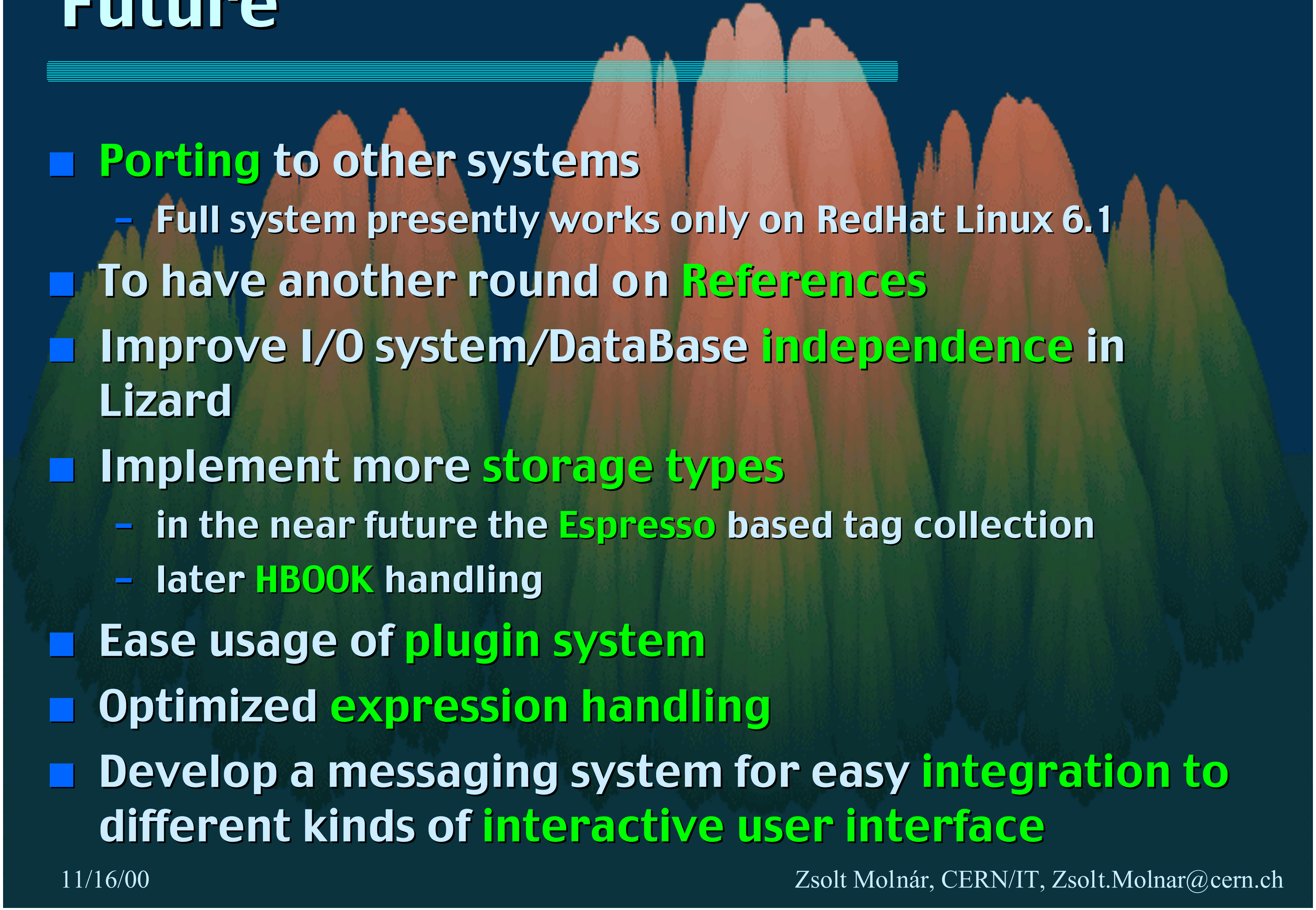
```
#include <HepExpNtuple.h>

int main()
{
  .....

  // Create a factory which constructs NTuples based on
  // HepODBMS tag collections
  HepExpNTupleFactory factory;

  // Create the NTuple via the factory and opens it for reading
  NTuple* ntuple = factory.findC( "Example Tag Collection" );

  // Creates Quantities and bind them
  Quantity<long>   eventNo;
  Quantity<double> Energy;

  ntuple->bind( "eventNo", eventNo );
  ntuple->bind( "Energy", Energy );

  // Simple looping (write values of attributes into a stream )
  for( ntuple->begin(); !ntuple->isEnd(); ntuple->next() )
    {
      cout << eventNo << "\t" << Energy << endl;
    }

  .....
}
```

```
-----XEmacs: myAnalysis4.cpp      (C++ Font)----All------------------------
Wrote /afs/cern.ch/user/m/molnarzs/private/Project/myAnalysis4.cpp
```

Summary

- Use of standard solutions and new software technologies
- General, AIDA-compliant ntuple solution
- Two interface layers in one system
  - for physicist and application programmers
- Hide system and implementation details
- Tag collection is a type of ntuple implementation
- Aims: scalability, good performance, simple usage
- Re-configuration of the whole system on the fly
- Utilities to help application programming
- Optimization for analysis tasks

Future



- **Porting** to other systems
  - Full system presently works only on RedHat Linux 6.1
- To have another round on **References**
- Improve I/O system/DataBase **independence** in Lizard
- Implement more **storage types**
  - in the near future the **Espresso** based tag collection
  - later **HBOOK** handling
- Ease usage of **plugin system**
- Optimized **expression handling**
- Develop a messaging system for easy **integration to different kinds of interactive user interface**

Info

AIDA -- Abstract Interfaces for Data Analysis

http://wwwinfo.cern.ch/asd/lhc++/AIDA/

The Lizard project: an AIDA compliant Interactive Analysis Environment

http://wwwinfo.cern.ch/asd/lhc++/Lizard/

Analysis for Physics Experiments – Anaphe

http://wwwinfo.cern.ch/asd/lhc++/lhcppguide/

HepODBMS User Guide

http://wwwinfo.cern.ch/asd/lhc++/HepODBMS/user-guide/