# Cellular generator mFOAM distributed with ROOT

ROOT Users Workshop
September 2005

S.Jadach and P.Sawicki
IFJ PAN Krakow, Poland
Stanislaw.jadach@ifj.edu.pl Pawel.sawicki@ifj.edu.pl

see also
S.Jadach and P.Sawicki, physics/0506084

Supported in part by EU grant MTKD-CT-2004-510126
realized in partnership with CERN PH/TH Division

# What is FOAM ?

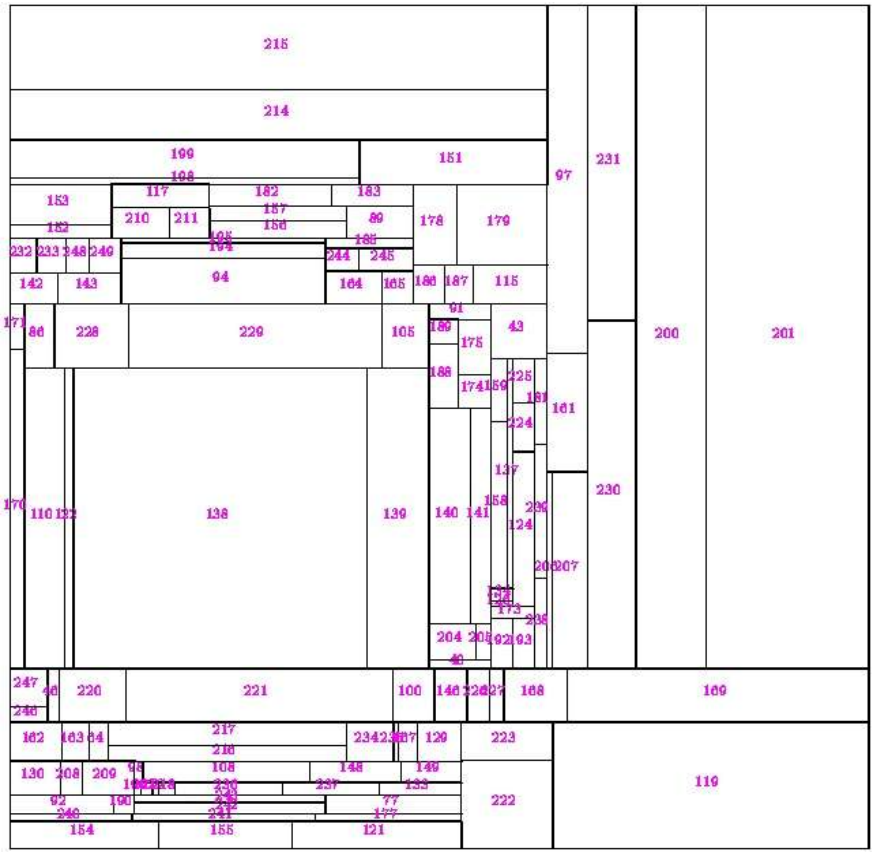
- FOAM algorithm is an universal MC event generator/integrator.
- Its primary purpose is the generation of events according to user defined arbitrarily complicated multidimensional probability distribution.
- As other MC generators FOAM can be used also for computing integrals.
- FOAM was invented primarily for use in high energy physics. However its area of applications is much wider.
- At this time FOAM has reached the level of mature MC tool and one of our goals is to make it more friendly for the average user.

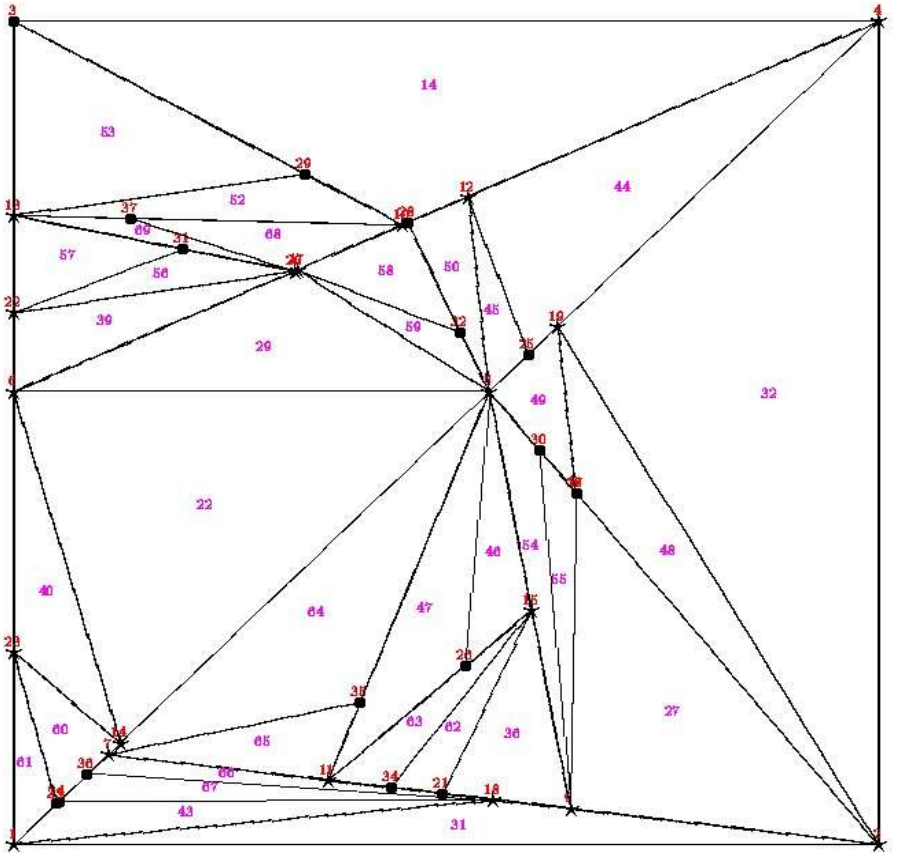
# Overview of FOAM algorithm

- FOAM is an example of self-adapting cellular generator.
- It means that integration space is divided into cells which form a system called *foam*.
- The *foam* is constructed during the *exploration* phase which is driven by the user defined probability distribution.
- The main aim of the *exploration* process is to reduce either
1) the ratio of the maximum weight to the average weight (generating unweighted events) or
2) the ratio of the variance to the average weight (computing integrals)
- After the *exploration* MC events are generated.
- For the detailed description of the algorithm see
S.Jadach, Comp. Phys. Commun. 152 (2003) 55

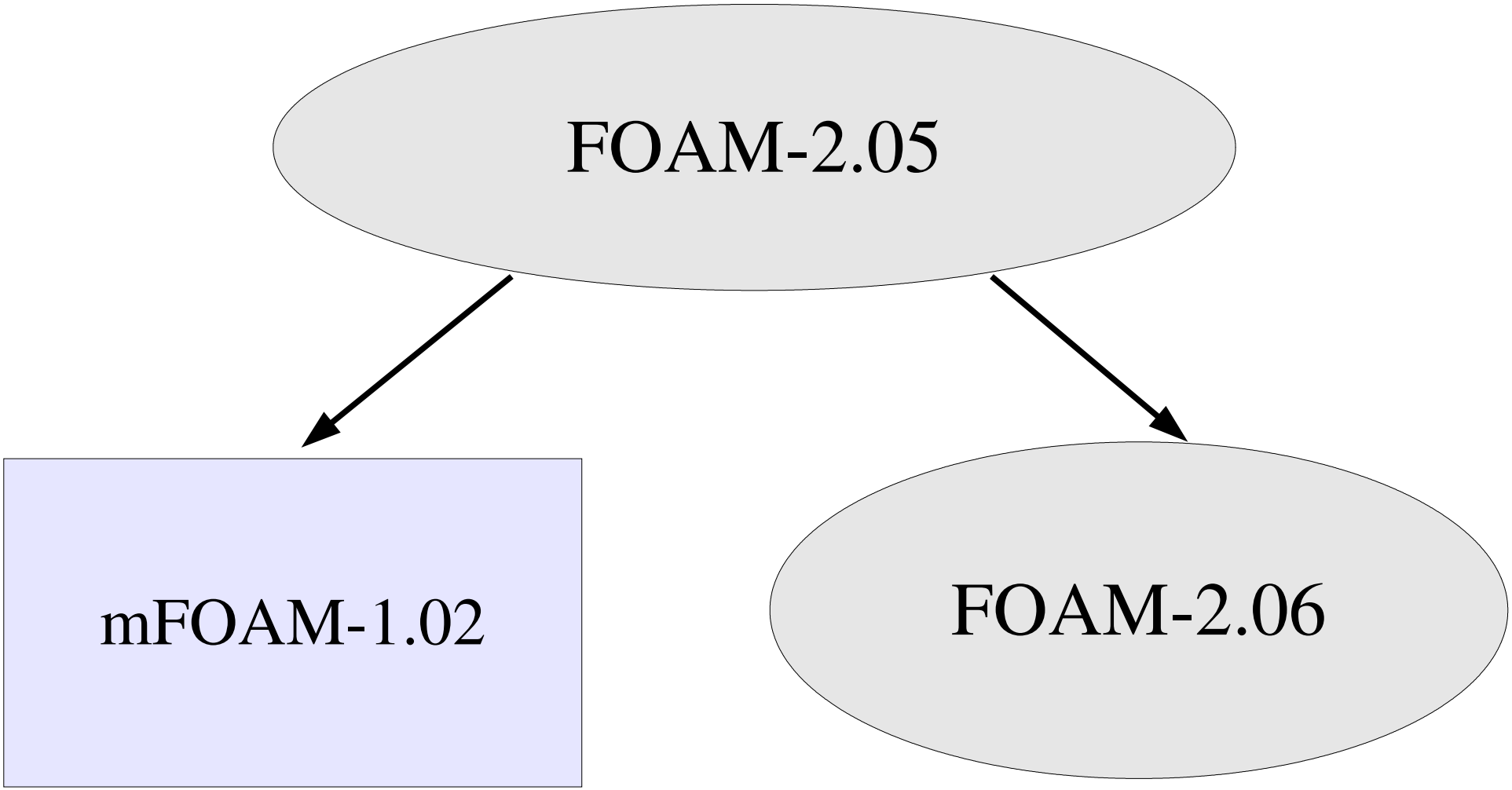
# Examples of “foam”

Cellular foam is constructed by binary splits of cells which can be either hyperrectangles or simplices (or even Cartesian product of both)





# Recent development in FOAM algorithm



# Short description of mFOAM-1.02 code

- mFOAM is part of ROOT distribution (version 4.04 and later).
- In mFOAM cells are hyperrectangles. The foam is constructed in the recursive process of binary splittings.
- User interface is maximally simplified. User should specify his probability distribution function (PDF) and choose random number generator from ROOT's **TRandom** class.
- There are *nine* principal configuration parameters. User can modify them easily.
- Many internal utilities are replaced by ROOT classes.
- All classes of mFOAM are fully persistent. It is possible to record easily status of the generator at disk. Later on mFOAM objects can be restored in ready to go state.

# TFoam class

- This is the main class of mFOAM.
- Each instance of TFoam class is independent MC event generator. For example the following piece of code :

  **TFoam *FoamX= new TFoam('FoamX');**

  creates an instance of mFOAM generator called **FoamX.**
- After creation of TFoam object user can modify configuration parameters with the help of appropriate setter methods.
- The most important procedure in TFoam class is the **Initialize** method which performs the foam build-up.
- Objects of TFoam class can be written to disk at any time after initialization and restored later.

# TFoamCell class

- TFoamCell class contains methods and data relevant for a single cell object.
- Most of the methods are setters and getters. There are also procedures which calculate volume of the cell and absolute coordinates of MC events.
- Cells in mFOAM are organized in the linked tree structure. Each cell contains pointers to a parent cell and to two daughter cells.
- In mFOAM instead of raw C++ pointers we employ special class **TRef** of persistent pointers to fix a certain problem in ROOT automatic streamers (cloning objects).

# User interface to PDF

•TFoamIntegrand is Abstract Base Class. User should define the **Density** method (probability distribution function).

```
class TFDISTR::public TFoamIntegrand{
public:
    TFDISTR();
    Double_t Density(Int_t, Double_t *){.....}
}
```

•The pointer to probability distribution is set to mFOAM object by setter method SetRho(TFoamIntegrand *). It can be eventually redefined later by ResetRho(TFoamIntegrand *) method.

•In applications interpreted by CINT probability distribution is defined as a global function and the setter method is SetRhoInt(void *).

# User application programs

User application programs can be run as:

1) Macro programs interpreted directly by rootcint (CINT) (preferred method for simple applications)

2) Macro programs compiled by ACLiC facility of ROOT (preferred method for medium size applications)

3) Stand-alone application program linked with ROOT libraries (large-scale programs with possibly many mFOAM objects).

In the standard ROOT distribution in the /tutorials subdirectory there are 3 demonstration programs.

# Demonstration programs

•Macro **foam_kanwa.C** can be run directly from CINT command line (scenario 1)

**root [0] .x  foam_kanwa.C**

This macro produces graphical output.

•Macro **foam_demo.C** can be compiled by ACLiC

**root [1] .x foam_demo.C+**

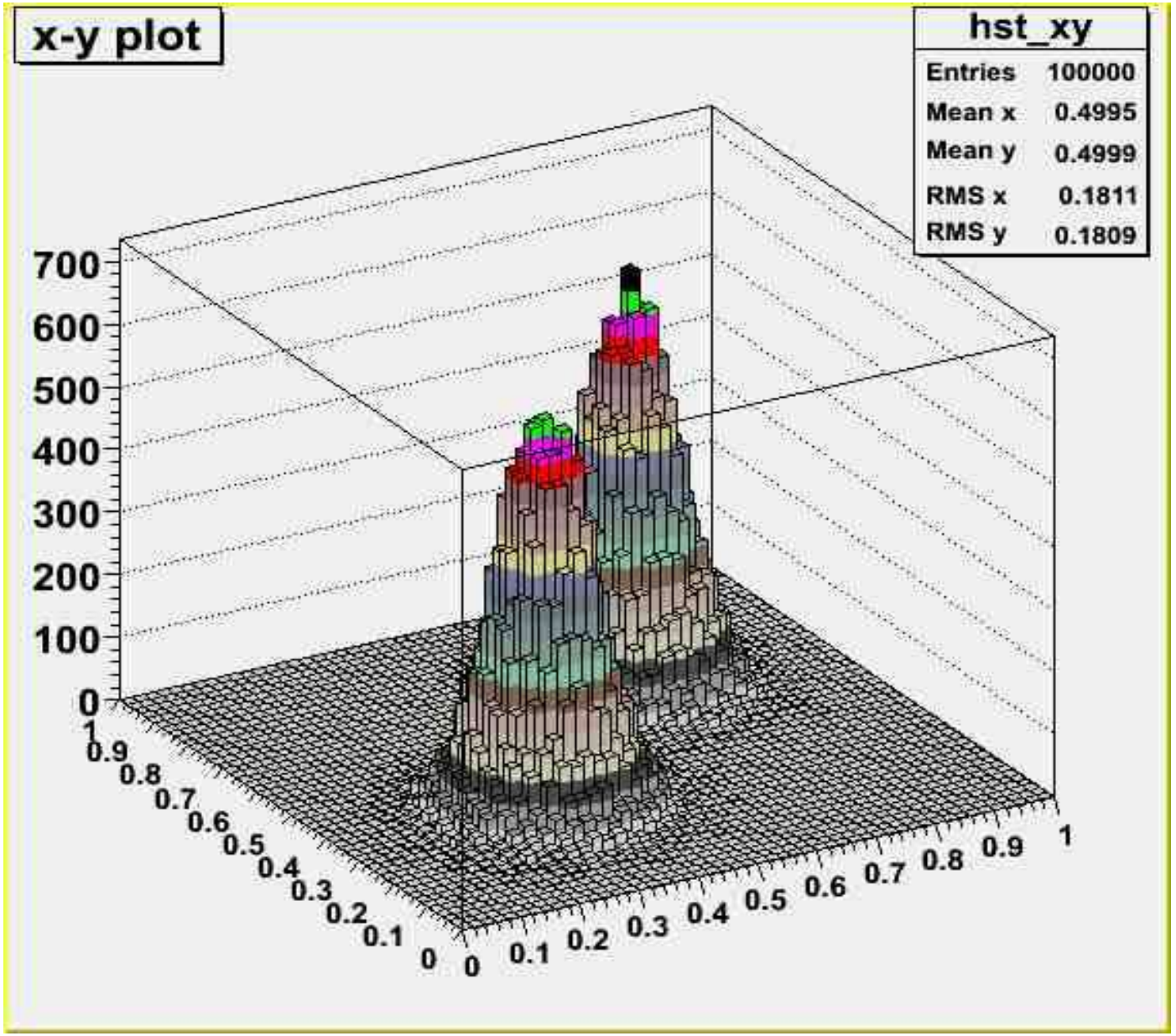
The shared library is created and just after initialization mFOAM object is written to disk.

•We can now test persistency with the  macro **foam_demopers.C**

**root [2] .x foam_demopers.C**

The output from  **foam_demo.C** and **foam_demopers.C** is identical.

# Example graphical output



# More advanced use of mFOAM

- The package mFoam-examples-1.2.tar contains stand-alone demonstration programs (It is available from authors web page).
- To compile and install the above examples **automake** tools are required (see README file).
- These programs demonstrate advanced use of mFOAM with two mFOAM objects and one central random number generator serving both.
- There are many methods to organize relations between random number generators and user defined probability distributions objects in the implementation of persistency.

# Conclusions

- FOAM has reached the level of well tested MC tool.
- mFOAM was designed to provide solutions of many every-day problems in the MC simulations with less effort.
- In the present project we payed special attention to make it more user friendly. We achieved this goal by integration of FOAM with ROOT.
- We expect the feedback from its current and future users