Graphics in C++ and ROOT

Graphics in C++

- Graphics is C++ is not easy to handle
- The first thing to do is to install the `graphics.h` library.
  - Note: we cannot do it on the computer, since you need administrator privilege.
- If you want to use graphics.h on Ubuntu platform you need to compile and install libgraph. It is the implementation of turbo c graphics API on Linux using SDL.
- It is not very powerful and suitable for production quality application, but it is simple and easy-to-use for learning purpose.

Graphics in C++

- Dowload the library from here: http://download.savannah.gnu.org/releases/libgraph/libgraph-1.0.2.tar.gz
- First install build-essential by typing

```
sudo apt-get install build-essential
```

- Install some additional packages by typing

```
sudo apt-get install libsdl-image1.2 libsdl-image1.2-dev guile-1.8 \
  guile-1.8-dev libsdl1.2debian libart-2.0-dev libaudiofile-dev \
  libesd0-dev libdirectfb-dev libdirectfb-extra libfreetype6-dev \
  libxext-dev x11proto-xext-dev libfreetype6 libaa1 libaa1-dev \
  libslang2-dev libasound2 libasound2-dev
```

- Now extract the downloaded libgraph-1.0.2.tar.gz file.
- Goto extracted folder and run following command

```
./configure
make
sudo make install
sudo cp /usr/local/lib/libgraph.* /usr/lib
```

- Now you can use `#include<graphics.h>` on ubuntu platform

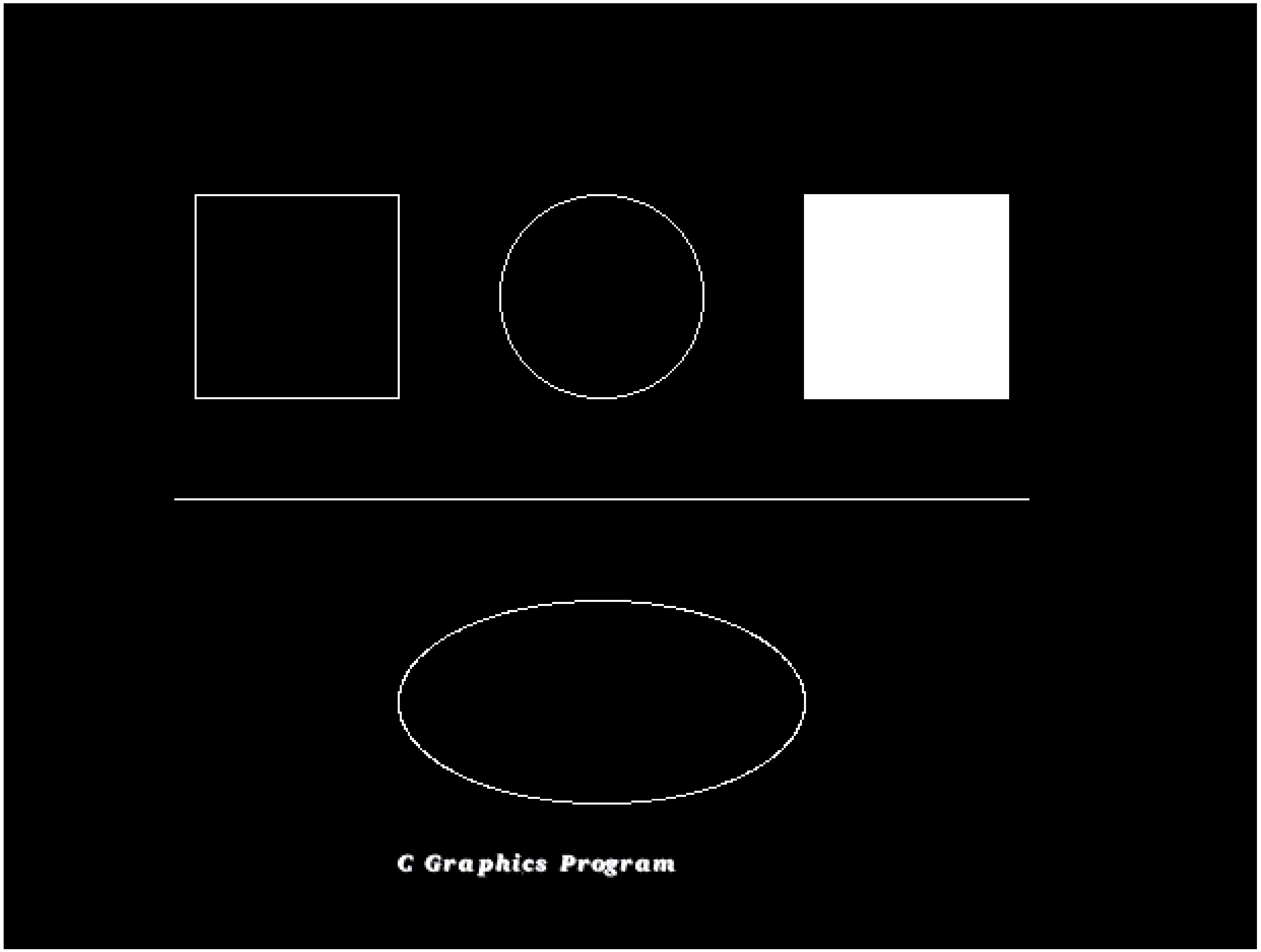
https://askubuntu.com/questions/525051/how-do-i-use-graphics-h-in-ubuntu

A simple example

```c
/*  demo.c*/

#include<graphics.h>

int main()
{
   int gd = DETECT,gm,left=100,top=100,right=200,bottom=200,x= 300,y=150,radius=50;
   initgraph(&gd,&gm,NULL);
   //   rectangle(left, top, right, bottom);

   initgraph(&gd,&gm,NULL);
   rectangle(left, top, right, bottom);
   circle(x, y, radius);
   bar(left + 300, top, right + 300, bottom);
   line(left - 10, top + 150, left + 410, top + 150);
   ellipse(x, y + 200, 0, 360, 100, 50);
   outtextxy(left + 100, top + 325, "C Graphics Program");

   delay(5000);
   closegraph();

   return 0;
}
```

```
gcc demo.c -o demo -lgraph
```

A simple example

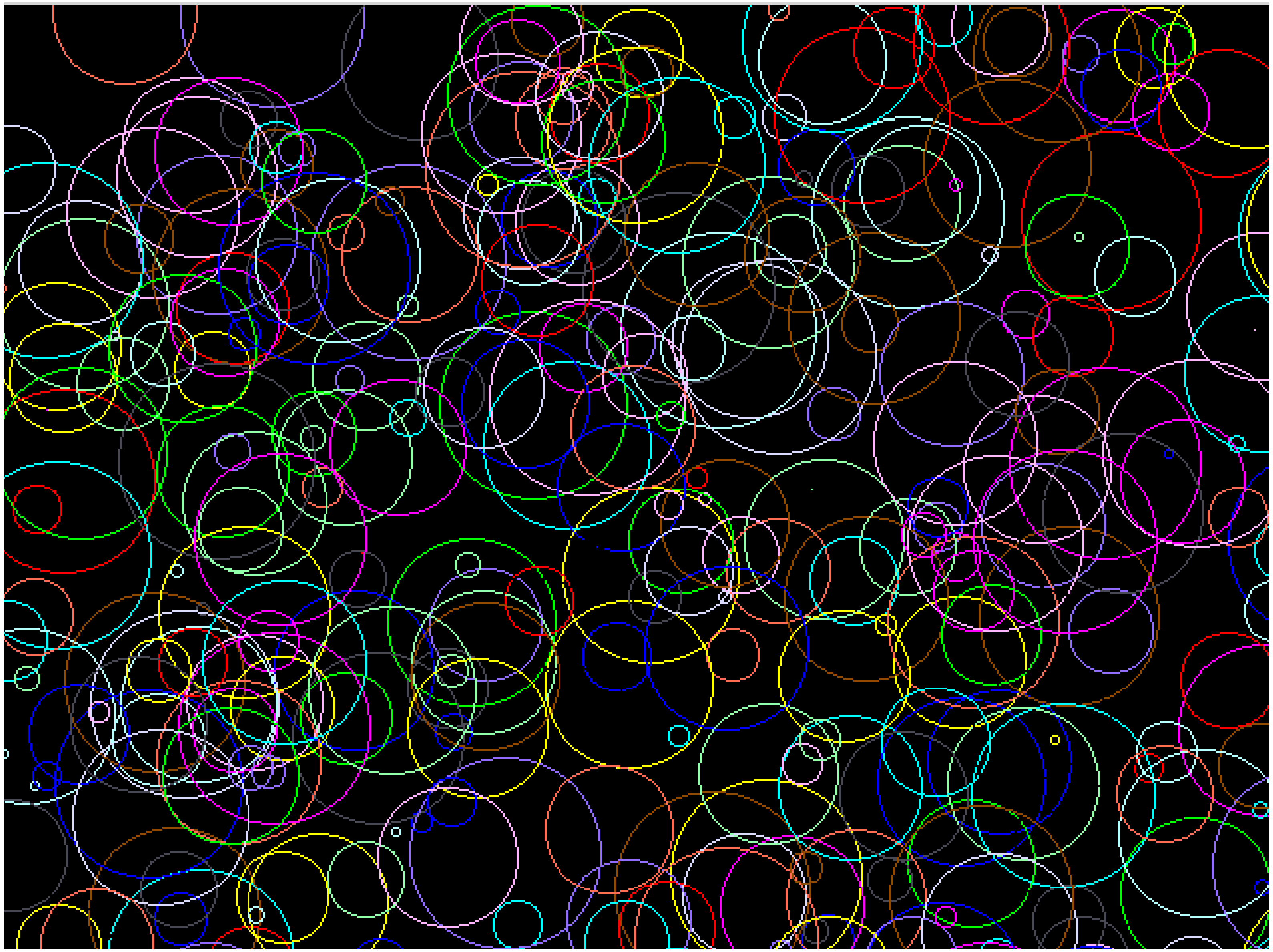


A second example

```c
/*  demo.c*/

#include<graphics.h>

int main()
{
   int gd = DETECT,gm,left=100,top=100,right=200,bottom=200,x= 300,y=150,radius=50;
   initgraph(&gd,&gm,NULL);

   double centrox, centroy, raggio;

   for(int i = 0; i < 500; ++i) {
     centrox = rand()%700;
     centroy = rand()%700;
     raggio  = rand()%radius;

     setcolor(rand()%15);
     circle(centrox, centroy, raggio);
   }

   delay(5000);
   closegraph();

   return 0;
}
```

```
gcc -std=c99 demo2.c -o demo -lgraph
```

A second example



Graphics in ROOT

- ROOT implements lots of classes devoted to graphic. If you are used to ROOT the usage of such classes is quite easy.
- Once you have ROOT installed there is no need to install external libraries.
- Some tutorials can be found in

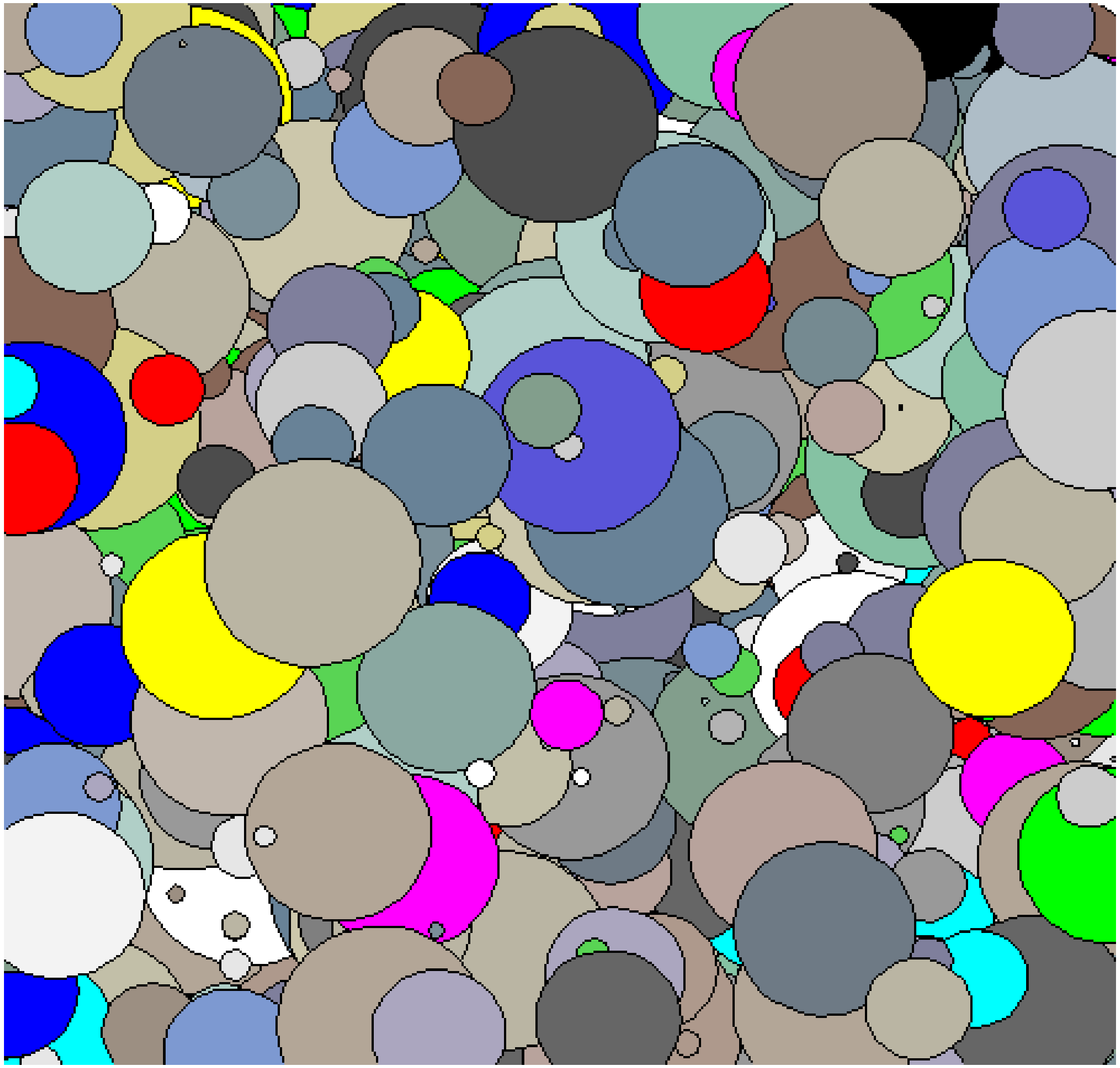
```
$ROOTSYS/tutorials/graphics
```

We will focus on few examples

Confetti generator in ROOT

```
#include <TCanvas.h>
#include <TEllipse.h>
#include <TRandom.h>

void Confetti(){

  TCanvas *c1 = new TCanvas("c1","c1",500,500);
  c1->cd();

  TEllipse *el1[1000];
  Double_t centrox,centroy,raggio;

  for(Int_t i = 0 ; i < 1000; i++){

    centrox = gRandom->Rndm();
    centroy = gRandom->Rndm();
    raggio  = gRandom->Rndm()/10.;

    el1[i] =  new TEllipse(centrox,centroy,raggio,raggio);
    el1[i]->SetFillColor(gRandom->Integer(42)%42);
    el1[i]->DrawClone();
  }

}
```

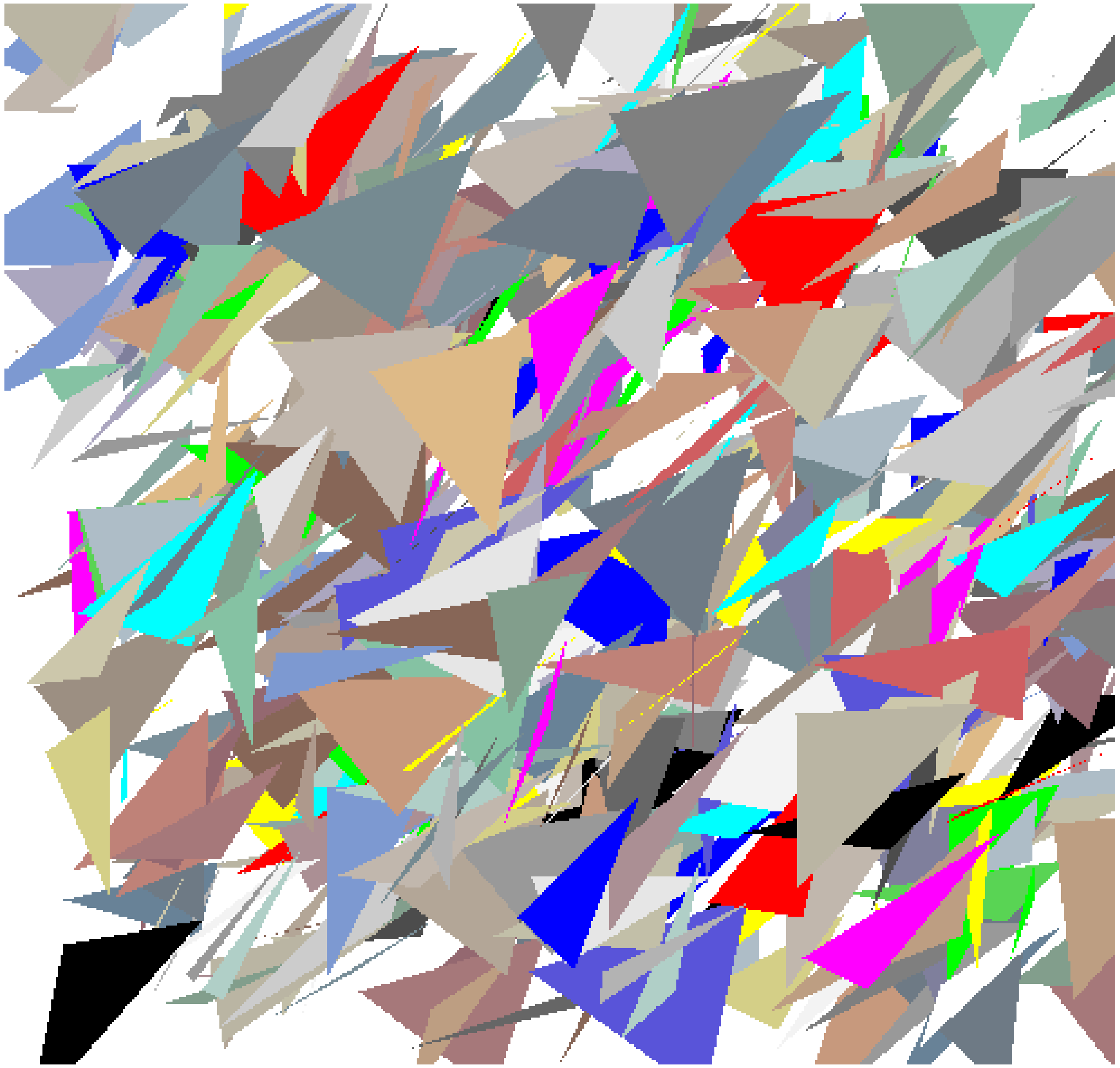
Confetti generator in ROOT



Triangles in ROOT

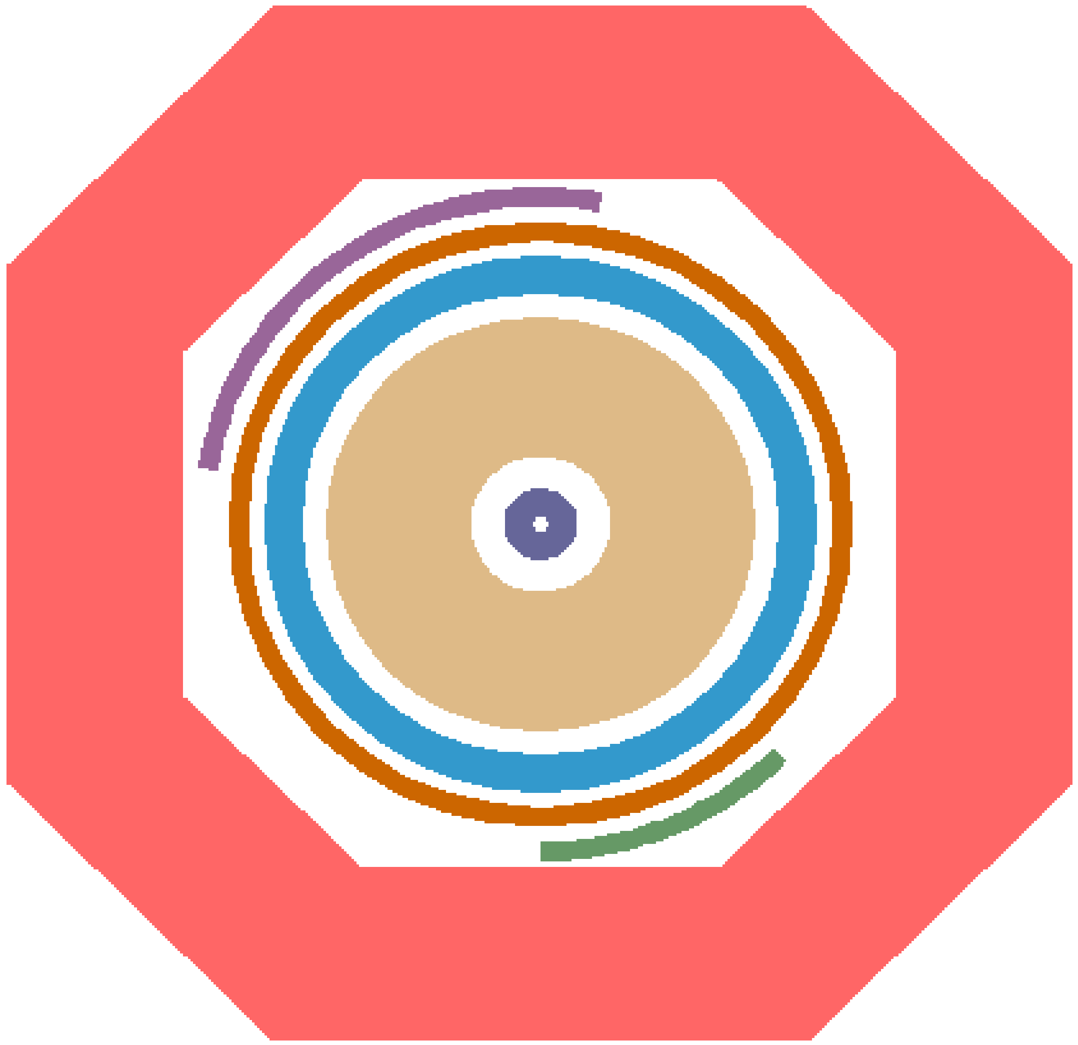
```
//generate small triangles randomly in the canvas.
#include <TCanvas.h>
#include <TRandom3.h>
#include <TPolyLine.h>
#include <TStyle.h>
#include <TColor.h>
#include <TROOT.h>

void Triangles(Int_t ntriangles = 500){

  TCanvas *c1 = new TCanvas("c1","triangles",500,500);
  TRandom3 r;
  Double_t dx = 0.2; Double_t dy = 0.2;
  Int_t ncolors = gStyle->GetNumberOfColors();
  Double_t x[4],y[4];
  TColor *c;
  Int_t ci;

  for (Int_t i=0;i<ntriangles;i++) {

    x[0] = r.Uniform(.05,.95); y[0] = r.Uniform(.05,.95);
    x[1] = x[0] + dx*r.Rndm(); y[1] = y[0] + dy*r.Rndm();
    x[2] = x[1] - dx*r.Rndm(); y[2] = y[1] - dy*r.Rndm();
    x[3] = x[0];               y[3] = y[0];

    TPolyLine *pl = new TPolyLine(4,x,y);
    ci = ncolors*r.Rndm();
    c  = gROOT->GetColor(ci);
    c->SetAlpha(r.Rndm());
    pl->SetFillColor(ci);
    pl->Draw("f");
  }
}
```

Triangles in ROOT



Exercise

Try to reproduce the following experimental setup



Tip:
Use **`TPolyLine`** and **`TCrown`** classes