MC-step intersection lib for SPDRoot (`cis-mc`)

Example of restrictive architecture on the way to NICARoot

Renat R. Dusaev
Tomsk HEP Group*

Internal SPD collaboration meeting, 25/04/018, Jinr, Dubna, RU

*) Tomsk Polytechnic University, Tomsk State University, Tomsk State Pedagogical

Outline

1. The holy cow of choosing the magnetic field configuration in SPD
2. Monte-Carlo simulations (internals)
3. Straightforward approach
4. Conclusion
5. Suggestions (software)

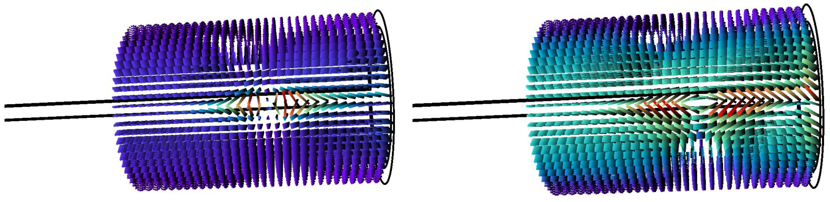
1.1 Physical Programme Selection Criteria *(SPD 02/11/017, 05/02/018, ...)*

- Drell-Yan pair produc on (muon and electron-positron ones);
- processes with prompt photons in the final state;
- processes with production of large p T mesons and baryons (semi-inclusive and inclusive);
- light and heavy vector meson production.

- charged meson multiplicity
- charged baryon multiplicity
- neutral mesons’ multiplicity (besides π 0)
- … (6 more)

=> Detector prototyping implies plethora of various spatial/phase-space parameters & distributions aside from particular detector setup for certain technical proposals.

1.2 Vector Fields Visualization Techniques



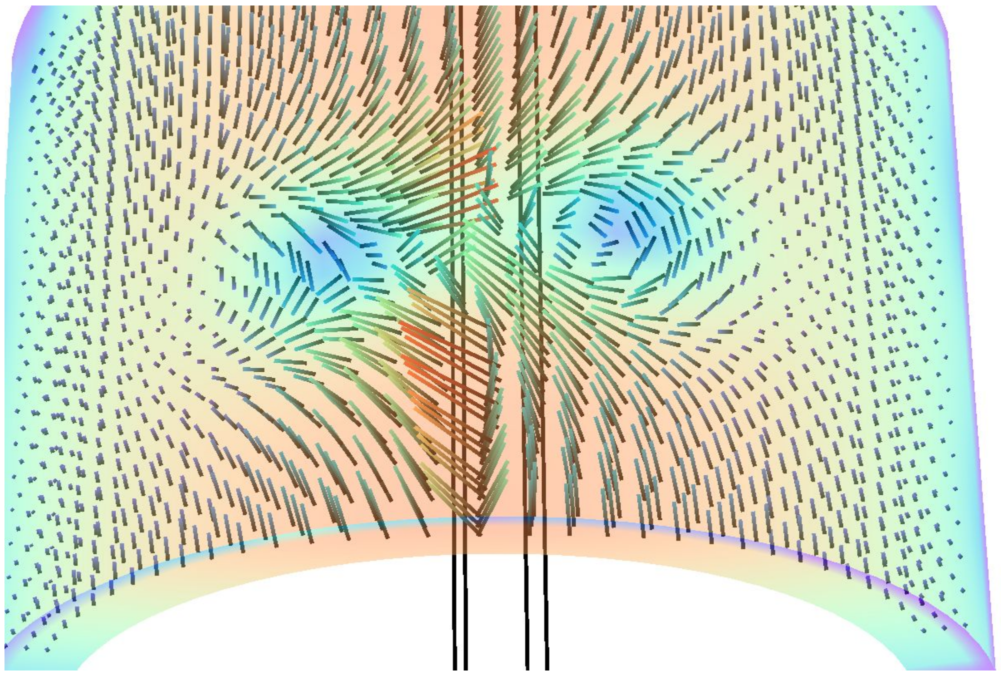
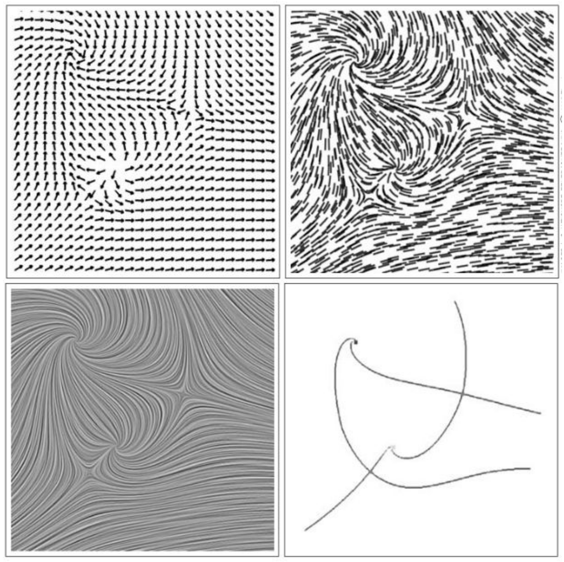
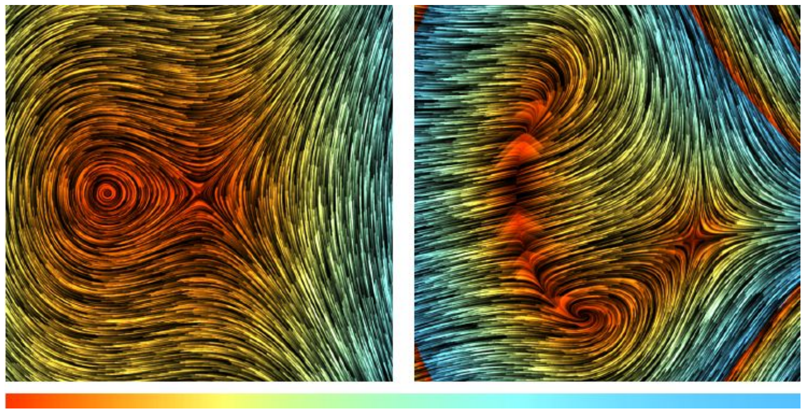


Image Copyright © 2008 - VISGRAF-IMPA

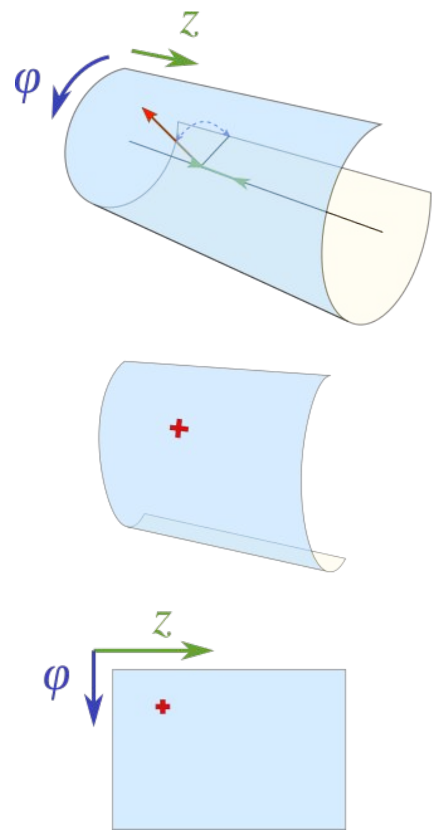


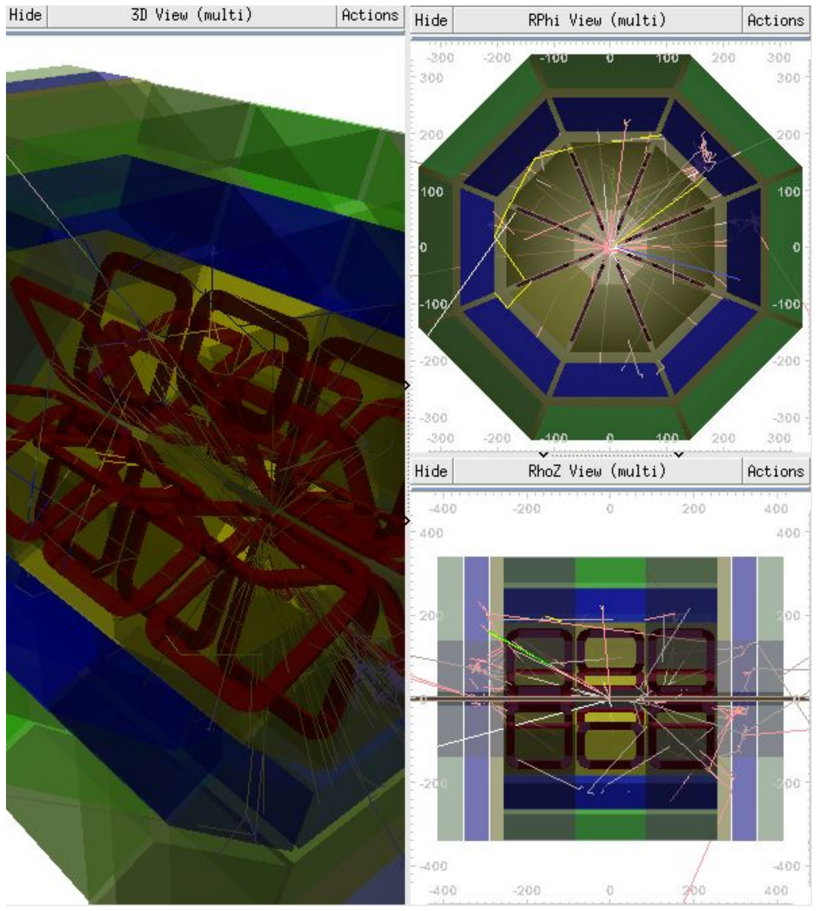
arXiv:1503.07137v1

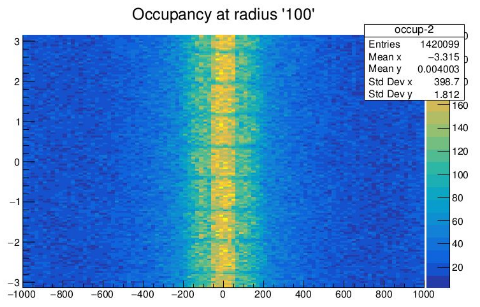


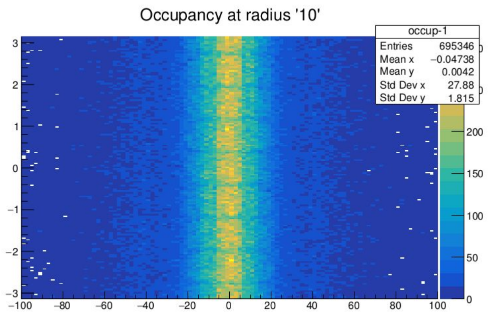
=> Albeit being imprecise, the most intuitive and straightforward way to represent spatial fields is **grids**

1.3 Cylindrical Surface Intersections



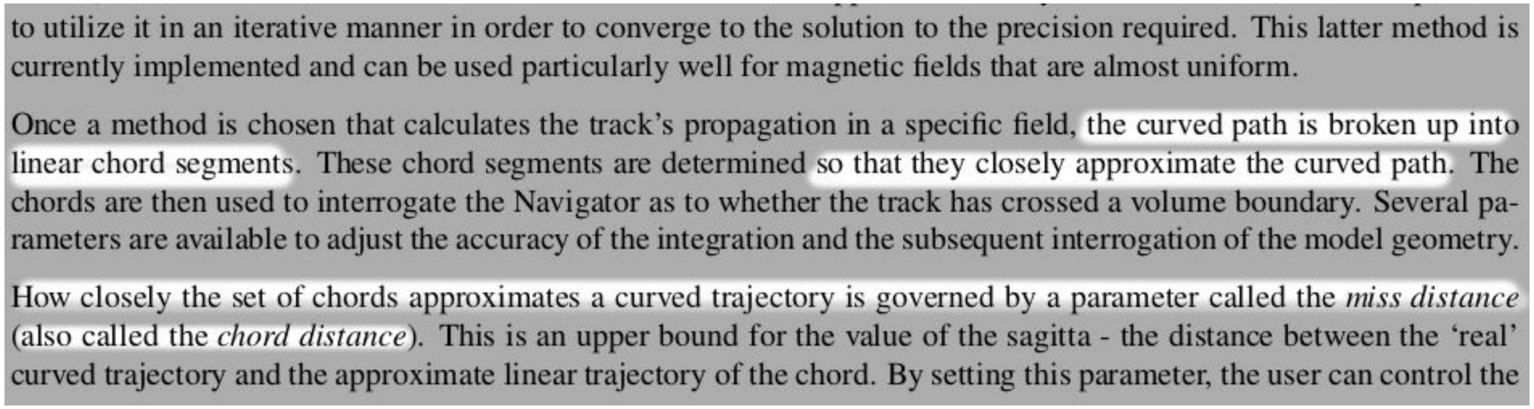






=> Unfolding cylindrical surface on plane z/φ

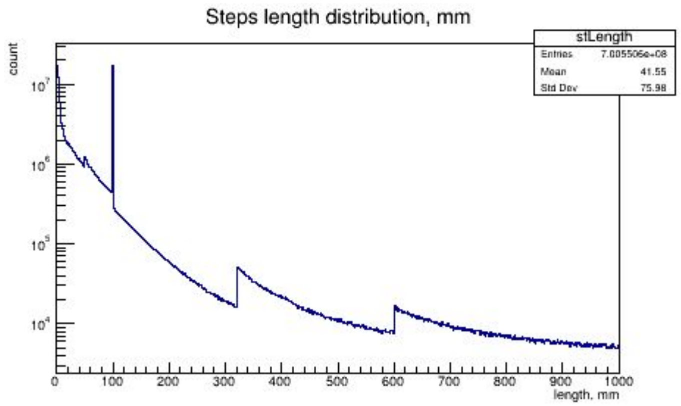
2.1 Monte-Carlo Stepping Stage

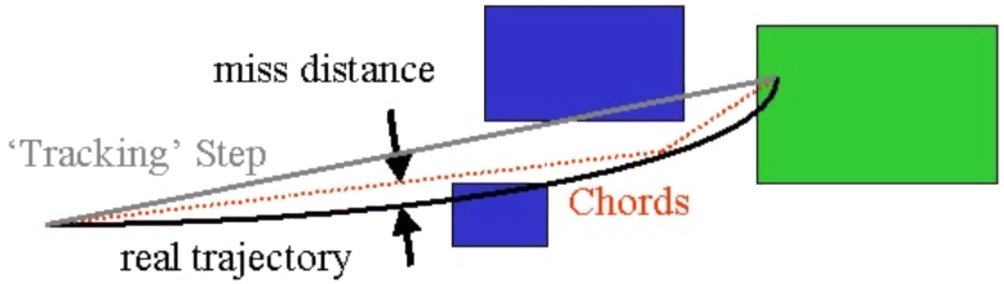

to utilize it in an iterative manner in order to converge to the solution to the precision required. This latter method is currently implemented and can be used particularly well for magnetic fields that are almost uniform.

Once a method is chosen that calculates the track's propagation in a specific field, the curved path is broken up into linear chord segments. These chord segments are determined so that they closely approximate the curved path. The chords are then used to interrogate the Navigator as to whether the track has crossed a volume boundary. Several parameters are available to adjust the accuracy of the integration and the subsequent interrogation of the model geometry.

How closely the set of chords approximates a curved trajectory is governed by a parameter called the *miss distance* (also called the *chord distance*). This is an upper bound for the value of the sagitta - the distance between the 'real' curved trajectory and the approximate linear trajectory of the chord. By setting this parameter, the user can control the

(Geant4 Book For Application Developers, rel. 10.4)





=> All the information need is already stored within the MC stack

=> We usually do not write everything because of its abundance

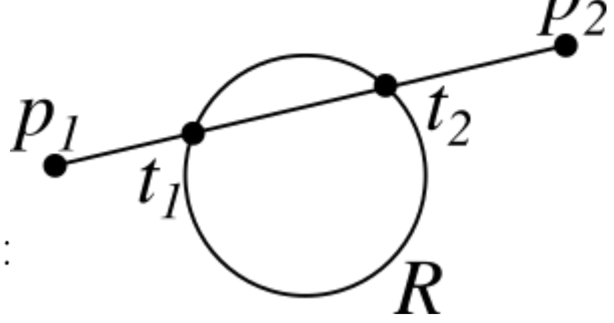
2.2 Cylindrical Surface Intersection Algo

(for reference)

The simple numerical procedure may be derived avoiding encumbering generic CSG i/section G4 algorithms, writing the parametric equation on the line segment crossing circle in *xy* plane.

for given $R, \{x_1, y_1\}, \{x_2, y_2\}$:

$$d_x = x_2 - x_1, \quad d_y = y_2 - y_1$$
$$S = |x_2 \cdot y_1 - y_2 \cdot x_1|, \quad \delta^2 = d_x^2 + d_y^2$$
$S > r \cdot \delta$ — no intersections since $D < 0$, where D:
$$D = \sqrt{R^2\delta^2 - S^2};$$
$$t_l = y_1 \cdot d_y - x_1 \cdot d_x, \quad t_{1,2} = \mp(D \pm t_l)/\delta^2$$

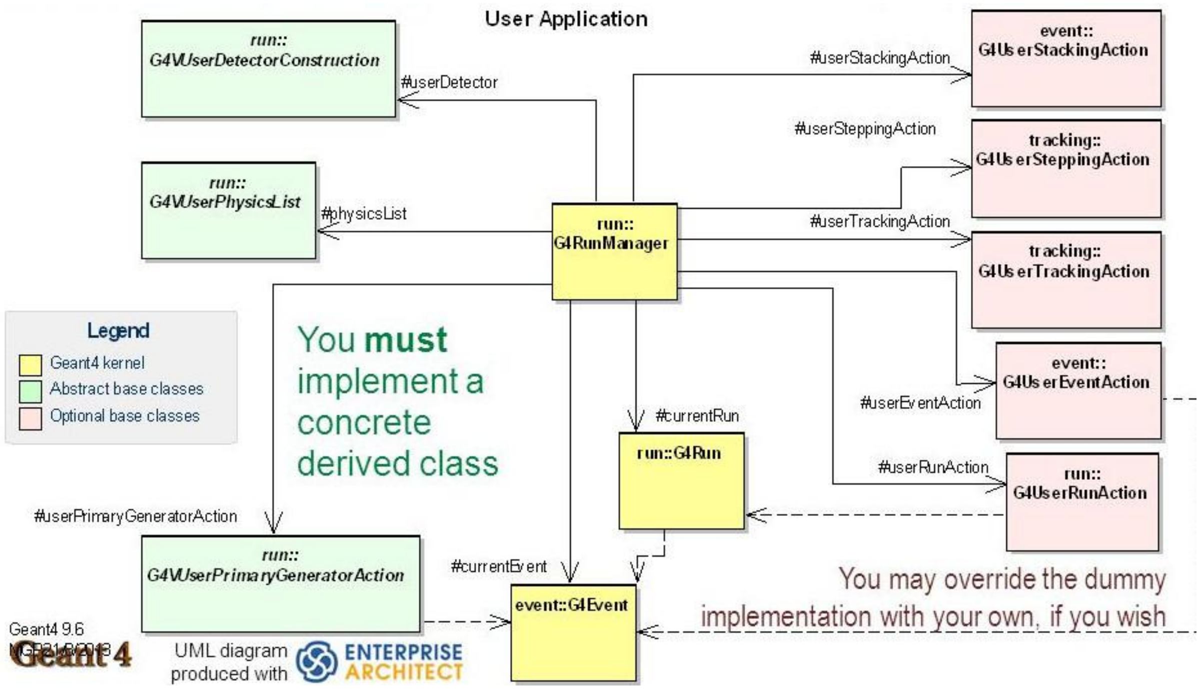


=> Numerically optimized, tested well on false positives, see scripts at `macro/cis-mc`

=> Use no voxelization, precise and cheap

2.3 Extending MC Procedure in Native G4 Terms

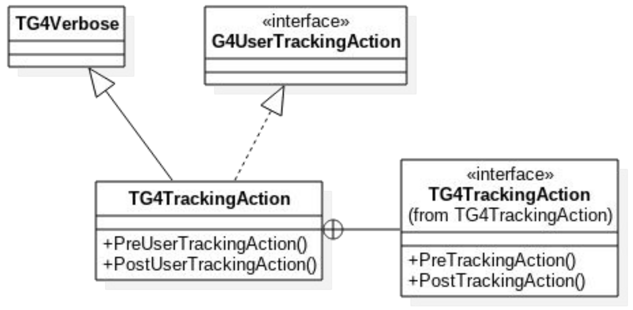
- Every interface in Geant4 has a corresponding dedicated class with clear and fixed contract.
- Data processing pipeline and MC staging have kown lifetime, have no side effects, and are well-documented.

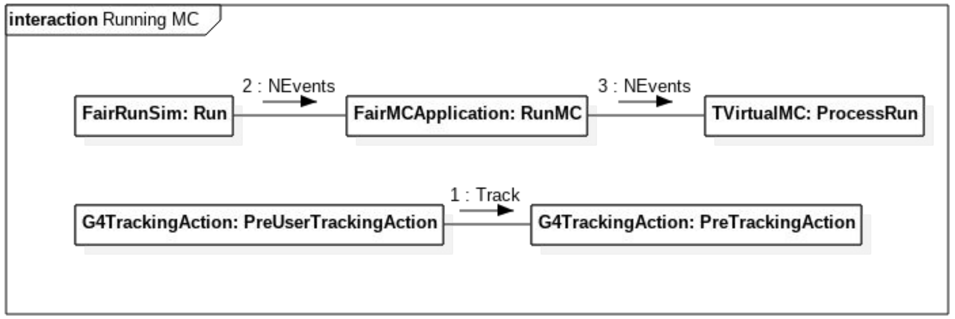


credits: http://slideplayer.com/slide/6374717/

=> Interfaces are provided by concrete base classes with dummy implementation.
So far, so clear...

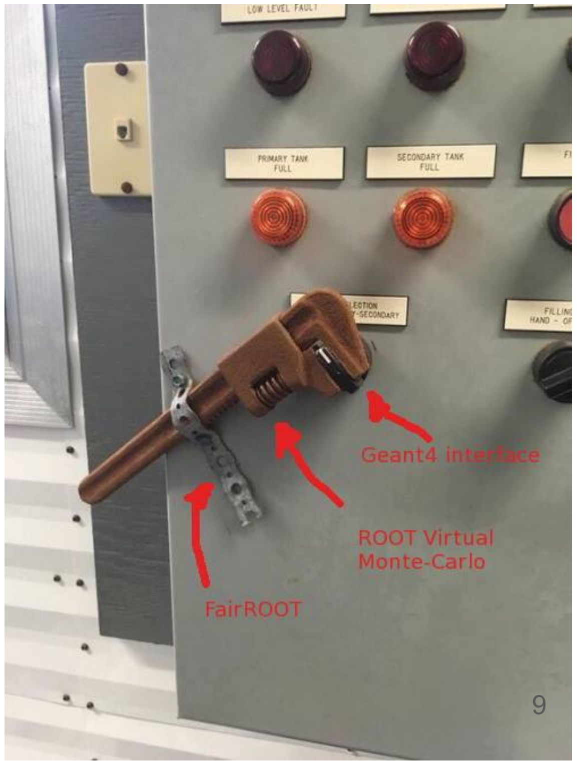
2.4 VMC + FairROOT Extension Interfaces





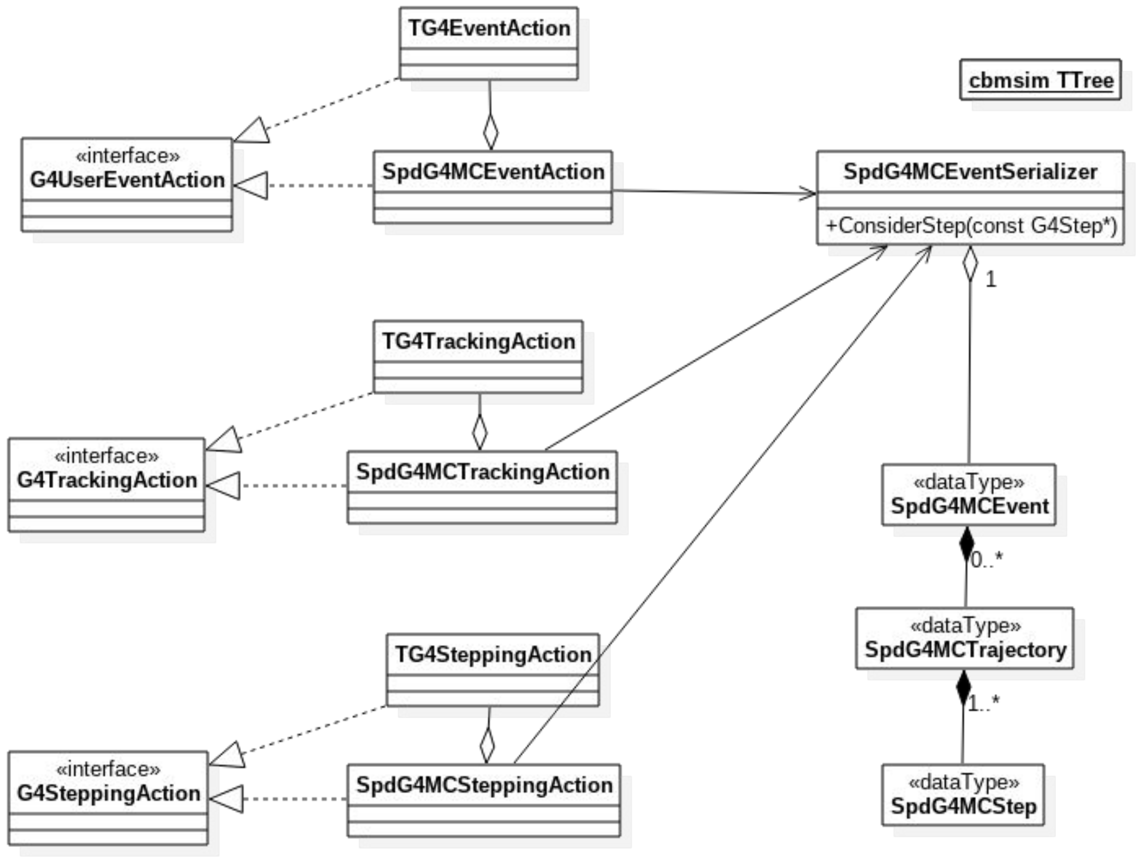
`FairMCApplication::Stepping()` has ~80 lines of code while the instance itself is created within the **`FairRunSim::Init()`** method of ~100 lines of code.

Overriding it via legitimate way (inside VMC interfaces) will require injection of >3 additional classes with copy/pasting semantically saturated code.



=> No way to unobtrusively extend FairROOT’s interface consumers (reason ‘coz MPD/BMN went the other way?)

3.1 The Aggregation Kludge



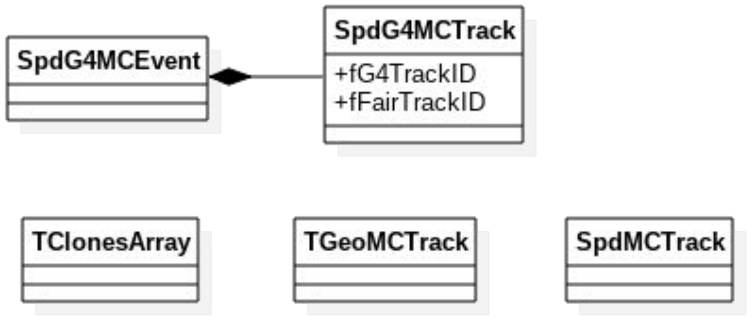
Instead of extension by inheritance, the VMC's actions are immersed within supporting implementations via aggregation.

Possible drawback: type downcasting chain broken.

SPDRoot must implement own MCSim/MCRun classes to inject customized logic within MC stack staging process.

=> Doing things right now.

3.2 Identifying the Tracks



- Geant4 uses uniq IDs to identify the tracks built
- MC stack itself is deterministic
- VMC writes MC stack formulated with `TGeoTrack’`s kept by FairStack (!)
- FairROOT applies cuts before writing SpdMCTrack’s into TTree instance defined by its IO manager (thus, breaking deterministic order)
- Three instances referring same entity arranged in non-matching sequential arrays!

=> Significant effort to establish the one-to-one relation within the ROOT file

3.3 Domain Specific Language for Filtering Conditions

```
([[:alpha:]][[:alnum:]]*):(([=><]{1,2})([^\|\&]+))((\||\&)(([=><]{1,2})([^\|\&]+)))*
```

Currently filtering conditions are hardcoded or passed by CINT’s callbacks.

```
class SpdPickInitialParticlesTask : public FairTask {
public:
    SpdPickInitialParticlesTask( std::set<Int_t> *
setPtr );
    virtual InitStatus  Init();
    virtual void Exec(Option_t* opt);
    void Collect( Int_t code );
private:
    std::set<Int_t> fPDGCodes;
    std::set<Int_t> * fChoosenOnes;
    TClonesArray * fMCTracks;
    ClassDef(SpdPickInitialParticlesTask,  0);
};
```

Examples with DSL:

- *particleType*:!gamma
- *particleType*:*’
- *eKin*>=100MeV&eKin<5GeV;*particleType*:=mu+|mu
–

=> Expressive solutions for every-day needs.

Grammar:

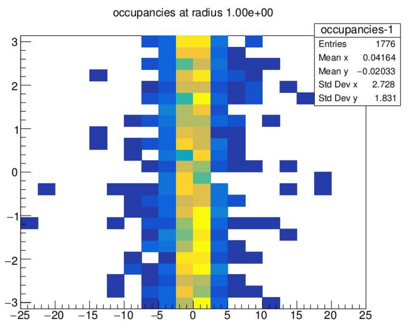
```
expessions ::= expr
             | expr ';' expressions
             ;

      expr  := label ':' filters
             ;

   filters ::= filter
             | '(' filters ')'
             | filter '&' filters
             | filter '|' filters
             ;

    filter ::= predicate
STRING_LEXEME
             ;

 predicate ::= '='
             | '!'
             | '>'
             | ">="
             | '<'
             | "<="
             ;

     label ::= ALNUM
             ;
```

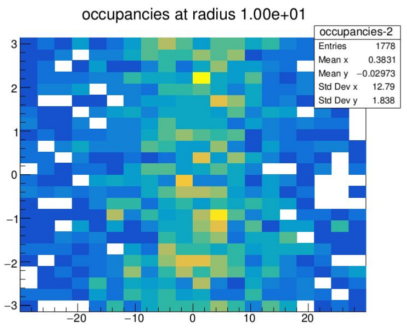
4. Conclusion

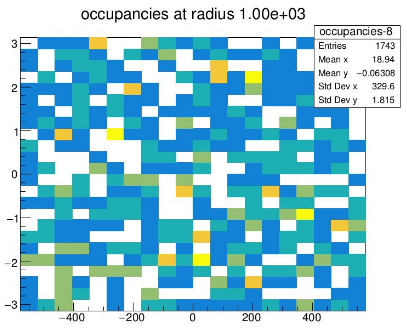
- SPDRoot shall either define overriding classes on FairROOT or stay straight within the existing functionality regulated by forthcoming 3-rd party releases (ROOT7, ALFA?)
- Elusive sacred knowledge: (almost) no docs, no dev pipeline, no self-documenting, C++ 94

5. Results

- `Dockerfiles` ready for recent and last but one releases of FairSoft/FairROOT bundles
- Doxygen cfg introduced (where to deploy?)
- `cis-mc` lib getting ready for practical use (TODO: DSL, for convenience), promising preliminary experience







(primary DY mu+/mu- for 4e3 events)

5. Suggestions

Coding

1. Virtual factories: event data hierarchy, application classes, RPC/IPS messaging
2. Fixed signatures & contracts for treatment utils
3. Configuration files (YAML/JSON/etc.)
4. Involve generic programming (C++ templates), C++11, `boost phoenix`
5. Logging facility (sinks)
6. Unit testing, system testing
7. Issue tracking, ticket assignment, activity logging, etc

System

1. Integration with CI/CD (keeping `master` branch protected!)
2. Deterministic builds (HybriLIT fits!):
   a. No more dangling shell scripts with volatile environment (no explicit usage of `config.[c]sh`, `thisroot.[c]sh`, etc)
   b. Full reproducible OS assembly (clamped repo, binary compatibility)
   c. Supersede FairSoft's shell scripting (~2.5 straightforward scripts doing labile fetches)
   d. Binary packaging during scheduled (nightly) builds
3. https://readthedocs.org/
4. **NICARoot**