Event Generators in PandaRoot – For Users by Users

Martin J. Galuska, Jifeng Hu,
Wolfgang Kühn
Justus Liebig Universität Gießen

Acknowledgements

All information is based on:

- EvtGen documentation by David Lange, Anders Ryd, et al.
  - robbep.home.cern.ch/robbep/EvtGen/GuideEvtGen.pdf
  - Or go to <PandaRootSourceDir>/pgenerators/EvtGen/doc and type ./do_latex (the file you want is called guide.ps)
- Presentations by
  - [1] Stefano Spataro
  - [2] Marco Destefanis
  - [3] David Lange, Anders Ryd, et al. http://indico.cern.ch/contributionDisplay.py?contribId=s2t1&confId=a031540 (click on transparencies)

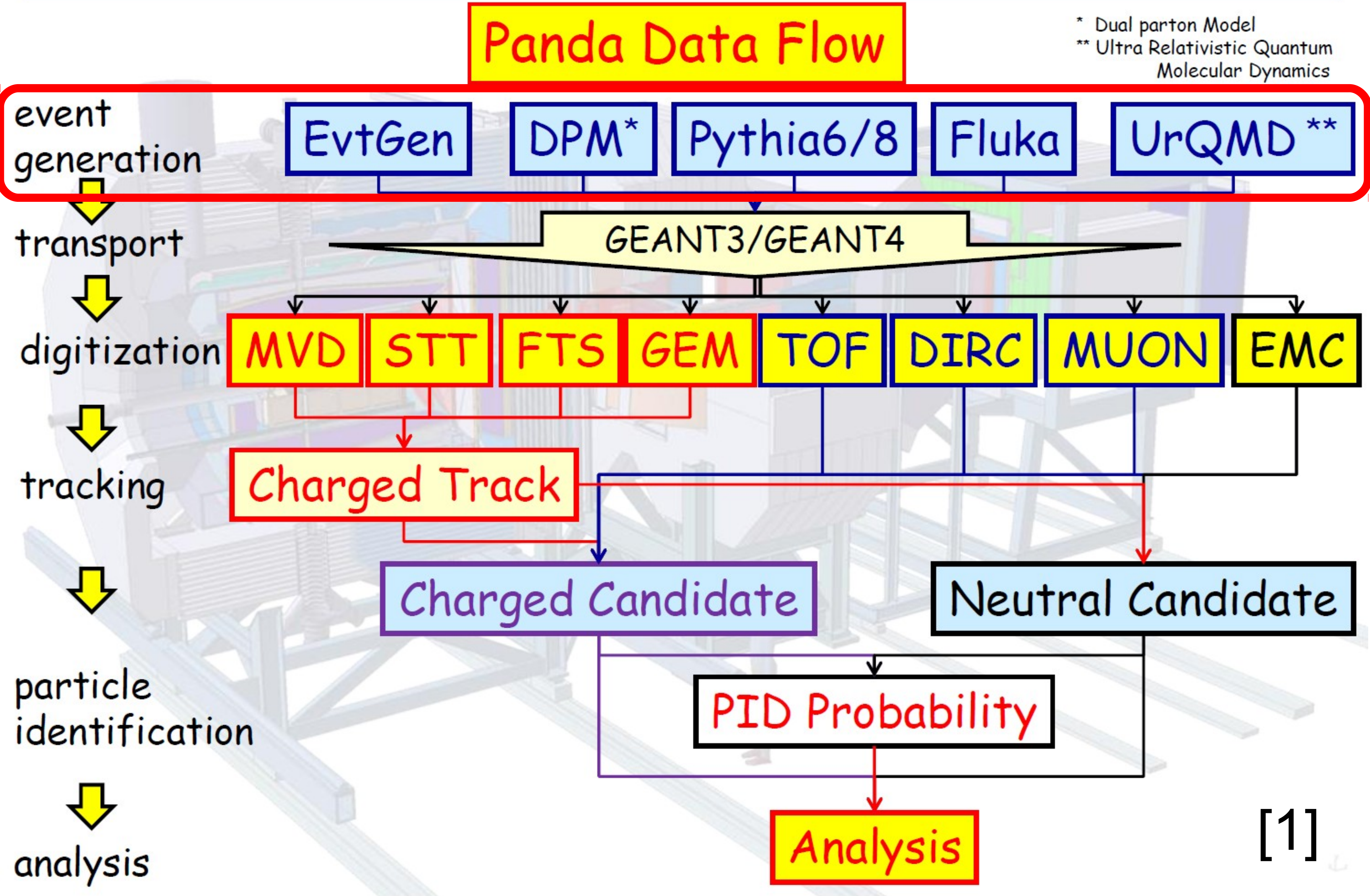
What can you expect from this talk?

- As a user
  - What event generators can I use for what?
  - How do I use them?
  - How can I write my own decay chain in EvtGen (using existing decay models)?
- As a (future) developer
  - How can I write my own decay model in EvtGen?
- As a PandaRoot code expert
  - What could be improved for the users?

Event generators

- What is an event generator?
- Which ones are available in PandaRoot?
- What can they do?
- Which one should I use for what purpose?

What is an Event generator?



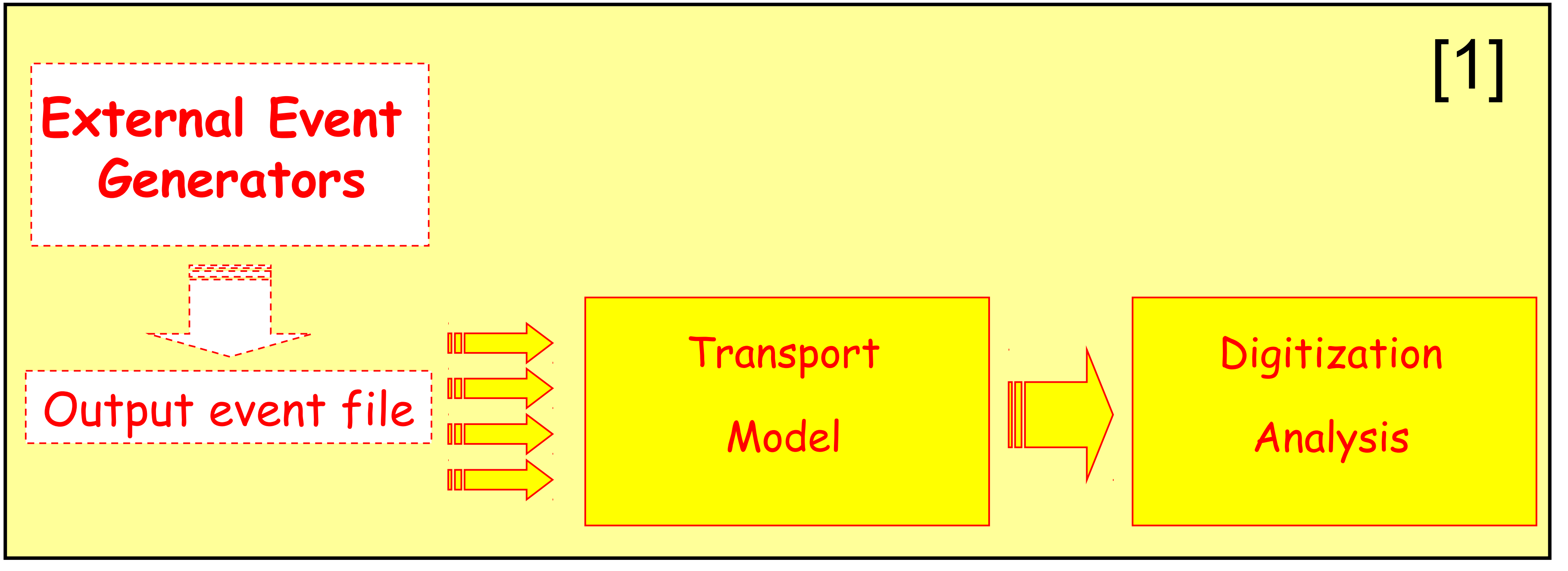
[1]

What is an event generator?

- Software which produces particles in final states for MC simulation
  - Simulates decay chains (which mother particle do you want to decay into which daughter particles?)
  - Angular distributions
- 4-momenta, initial positions, time and PID are passed to transport engine
- Transport engine simulates interaction of particles from event generator with material/detector

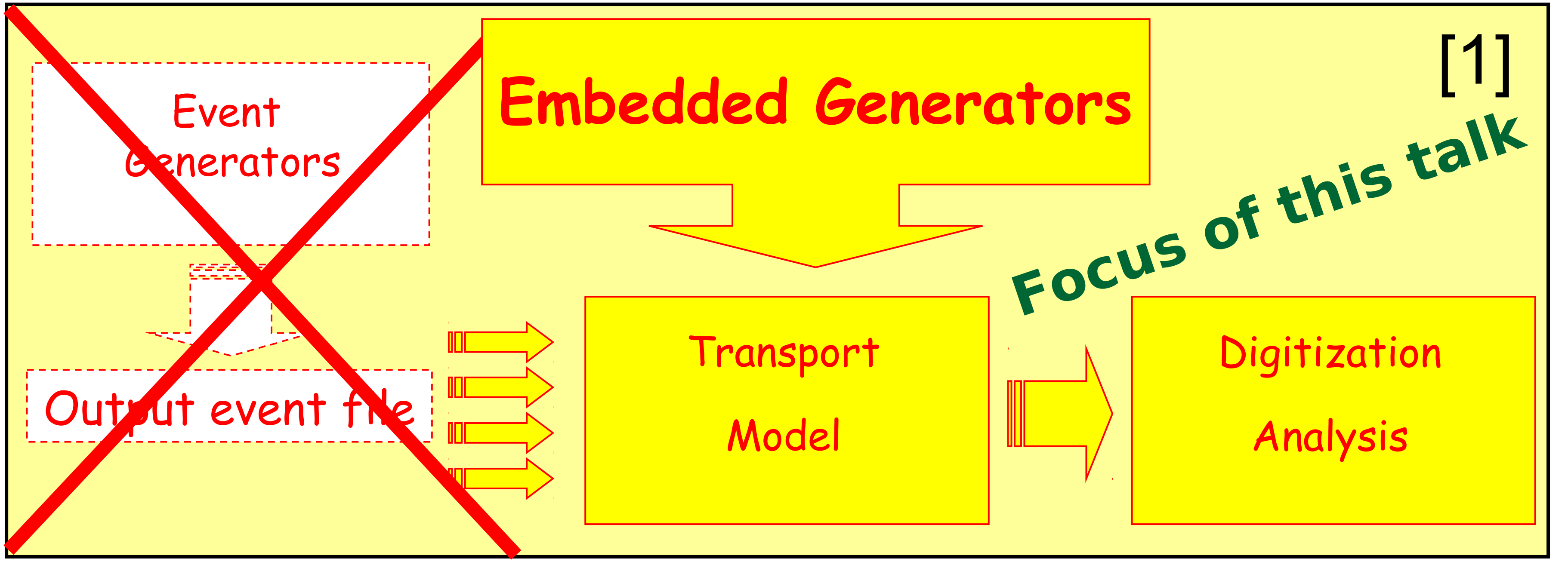
External Event Generators

- (Compile and) run external event generator “standalone”
- Produce file with particles that is read in by run_sim_... .C macro
- Generate events once, run sim, digi, reco, pid multiple times on same events (byte-wise comparison between different platforms)
- You can determine the angular distribution of the event generator without overhead



Embedded Event Generators

- No need to compile or run standalone program (especially convenient when using batch farm)
- Embedded event generators are called directly by run_sim_... .C macro
- Output is not stored in separate file
- If you want to run over the same events again, the event generator has to reproduce them (possibly slow, same seed → same events?)



What Event Generators are Availbable in PandaRoot? How do I use them?

- Have a look at <PandaRootSourceDir>/(p)generators/ (and subdirectories) to see what is available in PandaRoot
- You can find out yourself how to use the event generator:
  - Look at comments in the *.h file, parameters of the constructor + private/protected variables
  - Look at the *.cxx file (good luck!)
  - Look at the documentation
  - Look at the code of the according standalone program
  - Look at <PandaRootSourceDir>/macro/pid/run_sim_*.C for examples on how to use some event generators
  - Try and see what happens!

Box Generator/ Particle Gun

- No decay chains!
- Generates particles with flat distribution in p, pt, phi, eta, y, theta or cos(theta)
- Fixed multiplicity per event
- Vertex can be set to point or flat distribution within rectangle
- PIDs need to be known by transport engine

- Use cases
  - Beam test detector code
  - Determine (single) particle resolutions

How do I use Box Generator?

- <PandaRootSourceDir>/macro/pid/run_sim_sttcombi_pgun.C

```
run_sim_sttcombi_pgun(Int_t nEvents=10, Int_t pid=13, Float_t p1=1.0, Float_t p2=-1, UInt_t seed=0){
   ...
// Create and Set Event Generator
//-------------------------------

FairPrimaryGenerator* primGen = new FairPrimaryGenerator();
fRun->SetGenerator(primGen);

 // Box Generator
FairBoxGenerator* boxGen = new FairBoxGenerator(pid, 1); // 13 = muon; 1 = multipl.
if (p2<0.) p2 = p1;
boxGen->SetPRange(p1,p2); // GeV/c
boxGen->SetPhiRange(0., 360.); // Azimuth angle range [degree]
boxGen->SetThetaRange(0.5, 140.); // Polar angle in lab system range [degree]
boxGen->SetXYZ(0., 0., 0.); // cm
primGen->AddGenerator(boxGen);
   ...
```

EvtGen (C++)

- Simulates entire decay chains
- Expandable Framework
  - Contains many physical decay models (as C++ modules)
  - You can write your own decay model(s)!
  - Functionality depends on decay model
    - Correct momenta + angular distributions
    - Check of conserved quantities
    - Physical effects such as CP violation, …
  - Interface to even more sophisticated event generators
- Use cases
  - Monte Carlo samples for physics processes
- BABAR version: http://www.slac.stanford.edu/~lange/EvtGen/
- CERN version: http://evtgen.warwick.ac.uk/

How do I use EvtGen?

- In your run_sim.C macro:

```
// Create and Set Event Generator
//-------------------------------

FairPrimaryGenerator* primGen = new FairPrimaryGenerator();
fRun->SetGenerator(primGen);

// Read in from file ...
// FairEvtGenGenerator* evtGen = new FairEvtGenGenerator("output.evt");
// primGen->AddGenerator(evtGen);

// ... generate your signal on the fly with given pbarMomentum
const Double_t pbarMomentum = 6.991;
PndEvtGenDirect *EvtGen = new PndEvtGenDirect("pbarpSystem","PSI2S.DEC",
                              pbarMomentum,gRandom->GetSeed(),"","X.pdl");


EvtGen->SetStoreTree(kTRUE);
primGen->AddGenerator(EvtGen);
```

Pythia8 (C++)

- Features
  - physics models for evolution from few-body hard process to complex multihadronic final state
  - initial- and final-state parton showers
  - multiple parton- parton interactions
  - beam remnants, string fragmentation and particle decays. It
  - Interfaces to external programs
- Use cases
  - high-energy pp or $\bar{p}p$ collisions
  - e+ e- or mu+ mu- annihilations
  - not e p, gamma p or gamma gamma collisions
- http://home.thep.lu.se/~torbjorn/Pythia.html
- Pythia6 (Fortran) is also available

How do I use Pythia8?

Interface to use Pythia8 for simulations

PndP8Generator* P8gen = new PndP8Generator();

P8gen->Init();

Set momentum → P8gen->SetMom(double);

Set parameters → P8gen->SetParameters(char*);

“PhaseSpace:pTHatMin = 0.001”

pythia.readstring(PhaseSpace:pTHatMin = 0.001);

Call to TRandom1 or TRandom3 classes

pytr1rng and pytr3rng

[2]

DPM – Dual Parton Model based event generator

- Simulates inelastic + elastic $\bar{p}p$ collisions
- Official background MC events producer for PANDA
- V. Uzhinsky, A. Galoyan, Cross Sections of Various Processes in Pbar P-Interactions, arXiv:hep-ph/0212369 (2002)

- Use cases
  - Background studies
  - Occupancy studies

DPM – Dual Parton Model based event generator

- <PandaRootSourceDir>/macro/pid/run_sim_sttcombi_dpm.C

```
run_sim_sttcombi_dpm(Int_t nEvents=10, Float_t mom = 5., Int_t mode =1, UInt_t seed=0){
```

...

```
// Create and Set Event Generator
//-------------------------------

FairPrimaryGenerator* primGen = new FairPrimaryGenerator();
fRun->SetGenerator(primGen);

PndDpmDirect *dpmGen = new PndDpmDirect(mom, mode, gRandom->GetSeed(), 2.);
primGen->AddGenerator(dpmGen);
```

...

- <PandaRootSourceDir>/pgenerators/PndDpmGenerator.h

```
/** Standard constructor
 *  @param Mom in GeV/C
 *  @param Mode = 0. - No elastic scattering, only inelastic
 *  @param Mode = 1. - Elastic and inelastic interactions
 *  @param Mode = 2. - Only elastic scattering, no inelastic one
 **/
PndDpmDirect(Double_t Mom, Int_t Mode, Long_t Seed = -1);
PndDpmDirect(Double_t Mom, Int_t Mode, Long_t Seed, Double_t ThtMin);
PndDpmDirect(Double_t Mom, Int_t Mode, Double_t Rsigma, TF1* DensityFunction, Long_t Seed = -1, Double_t
ThtMin=0.001);
```

How can I write my own decay chain in EvtGen?

- Important files
- What to do
- What not to do

Important files for EvtGen

- **evt.pdl**
  - Lists all particles and their properties
- **DECAY.DEC**
  - Default decays from PDG
- **User decay file**
  - Overwrites default decays
  - Name of file is arbitrary, let's call it **UserDecays.DEC**

- Files are located in <PandaRootSourceDir>/pgenerators/EvtGen

EvtGen: evt.pdl

- Every line corresponds to a particle
- Defines all properties relevant to EvtGen

Max. allowed downward deviation from mean mass [GeV]

particle name, number (stdhep numbering scheme)

c tau [mm]

```
add  p Lepton  mu-       13  0.1056584  0      0    -3  1  658654.   13
add  p Lepton  mu+      -13  0.1056584  0      0     3  1  658654.    0
add  p Meson   pi+      211  0.139570   0      0     3  0  7804.5   101
add  p Meson   pi-     -211  0.139570   0      0    -3  0  7804.5     0
add  p Meson   rho+     213  0.7685     0.151  0.4   3  2       0   121
add  p Meson   rho-    -213  0.7685     0.151  0.4  -3  2       0     0
add  p Meson   D0       421  1.86451    0      0     0  0  0.1244   105
add  p Meson   anti-D0 -421  1.86451    0      0     0  0  0.1244     0
```

Not used in EvtGen

mean mass, width [GeV]

3x charge, 2x spin

Lund-KC number (used for interfacing with other generators)

- * makes line a comment
- Note: Quantum numbers such as C or P are not included

Meaning of “Mass cutoff”

- Given decays of broad resonances to other broad resonances, we find that we must cut off the mass distribution of particles to improve the robustness of our code.

- Nominal mass range:

$$m_0\text{-}15\Gamma < m < m_0+15\Gamma$$

- If the mass cutoff in evt.pdl is not 0:

$$m_0\text{-Field\#8} < m < m_0+15\Gamma$$

- (Of course parent,daughter particle masses can also limit the mass range)

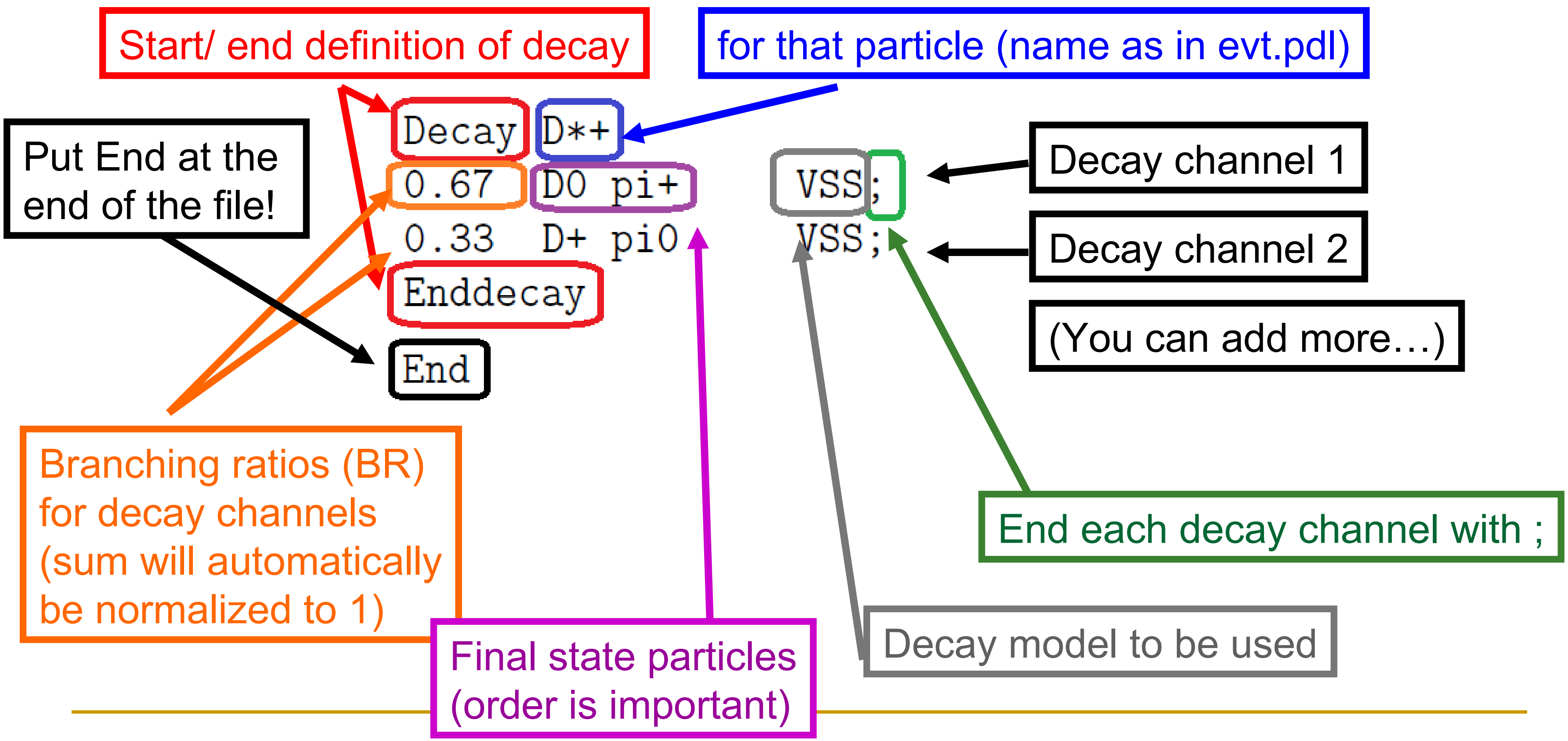
[3]

EvtGen: DECAY.DEC

- Defines the default decays for all particles in EvtGen
- Contains most decay channels (from PDG)
- Usually used for producing inclusive Monte Carlo samples
- For exclusive Monte Carlo samples, you can use a user decay file – UserDecays.DEC
  - Default decays can be overwritten
  - Same rules/syntax as for DECAY.DEC

EvtGen: UserDecays.DEC

- Each decay defined in UserDecays.DEC overwrites default decays in DECAY.DEC (decay information from PDG)
- Case-sensitive
- # makes line a comment



EvtGen: PHOTOS

- PHOTOS can be used for radiating photons of any charged particle in the final state
- Enable for all decays: yesPhotos
- Disable for all decays: noPhotos
- Control for individual decays: normalPhotos
  - For every decay channel you can define whether final state radiation should be included or not
  - Place PHOTOS before name of decay model to activate it!
- Default setting is yesPhotos

```
Decay J/psi
1.0000   e+ e-        PHOTOS VLL;
Enddecay

End
```

Activates final state radiation for this decay channel (only of meaning if normalPhotos was set)

EvtGen: pbarp-System

- Use as initial state one of the pbar*systems* below
- You can simulate operation at a certain CM-energy or with a given p-momentum
- If you use a particle instead, you will get an invariant mass resolution which is due to the particle's width and not the beam momentum spread

add	p Special	pbarpSystem	88888	2.98	0.1	0	0	0	0	88888
add	p Special	pbarpSystem0	88880	2.98	0.1	0	0	0	0	88880
add	p Special	pbarpSystem1	88881	2.98	0.1	0	0	2	0	88881
add	p Special	pbarpSystem2	88882	2.98	0.1	0	0	4	0	88882
add	p Special	pbardSystem	88889	4.32	0.1	0	0	0	0	88889
add	p Special	pbarnSystem	88887	3.07744	0.1	0	-3	0	0	88887

- Note: Using pbar*System* as initial state with PHOTOS corresponds to simulating initial state radiation. Therefore, put noPhotos or normalPhotos at the beginning of your UserDecays.DEC

Example use of decay model by A. Gillitzer

```
noPhotos
#
Decay pbardSystem
  1.0   p+  pbarnSystem                    DeuteronSpectator 1.0 1.16;
Enddecay
#
Decay pbarnSystem
  1.0   pi- phi                            PHSP;
Enddecay
#
Decay phi
  1.0   K+ K-                              VSS;
Enddecay
#
End
```

1st parameter: maximum internal momentum in the deuteron in GeV/c

2nd parameter: minimum mass of the residual pbar+n system for the specified decay (in this case m(phi)+m(pi-)

EvtGen: Constants and Parameters

You can use "constants"

Some models require parameters as input

```
Define dm    0.1
Define alpha 0.9
Decay B0
1.00   pi+ pi-    SSS_CP dm alpha 1 1.0 0.0 1.0 0.0;
Enddecay
Define alpha 1.1
Decay B0B
1.00   pi+ pi-    SSS_CP dm alpha 1 1.0 0.0 1.0 0.0;
Enddecay
End
```

dm = 0.1

alpha = 0.9

dm = 0.1

alpha = 1.1

EvtGen: Aliases

- Imagine that we want to simulate the following decay
  - anti-p- p+ → B0 anti-B0
  - B0 → J/psi K_S0,
    - J/psi → mu+ mu-
  - anti-B0 → anything

```
Decay B0
1.00      J/psi  K_S0      SVS_CP dm beta 1.0 1.0 0.0 1.0 0.0;
Endedecay

Decay   J/psi
1.00   mu+   mu-           VLL;
Enddecay
End
```

- Problem with that particular solution:
  - anti-B0 → J/psi something
  - J/psi from anti-B0 decays will also only decay into mu+ mu-
  - It should be allowed to decay in all channels

EvtGen: Aliases

- Solution → We use an alias for J/psi from different decays
  - Give J/psi from B0 decays another name (let's call it myJ/psi)
  - J/psi from anti-B0 can decay in all channels
  - J/psi from B0 can only decay into mu+ mu-

Define alias for J/psi called myJ/psi

Restrict decay of B0 to J/psi K_S0
(using aliased myJ/psi instead of regular J/psi)

```
Alias myJ/psi  J/psi
Decay B0
1.00    myJ/psi  K_S0     SVS_CP dm beta 1.0 1.0 0.0 1.0 0.0;
Endedecay
Decay myJ/psi
1.00    mu+   mu-          VLL;
Enddecay

End
```

Aliased myJ/psi is only allowed to decay into mu+ mu-
Regular J/psi from other decay chains are allowed to decay according to PDG

EvtGen: Cdecay

- CDecay anti-D0
  - Anti-D0 is charge conjugate of particle D0
  - CDecay anti-D0 defines decays for anti-D0 according to decays for D0 (decay channels into charged conjugated states and BRs as defined for D0)
- You can also use CDecay for aliased particles

Use mytau- as alias for tau-

```
Alias mytau- tau-
```

Define decays for mytau-

```
Decay mytau-
1.0      mu- anti-nu_mu  nu_tau    PHOTOS  TAULNUNU;
Enddecay
```

```
ChargeConj mytau- mytau+
```

Defines mytau+ as charge conjugate of mytau-

```
CDecay mytau+
```

Defines decays for mytau+ according to definition for mytau-

Implemented models in EvtGen

Many different models are implemented in EvtGen. They vary from highly specialized to rather generic. A rough grouping of these models into categories would be:

- Semileptonic decays
- CP violation
- Generic amplitudes
- Special matrix elements

Look at EvtGen documentation or
<PandaRootSourceDir>/pgenerators/EvtGen/EvtGenModels/

[3]

Semileptonic decays

- HQET - Heavy Quark Effective Theory inspired form factor param.
- ISGW, ISGW2 - Quark model based prediction, Isgur, Scora et al.
- MELIKHOV - Quark model based prediction
- SLPOLE - Generic spcification of form factors based on a lattice inspired parametrization.
- VUB - For generic b->ulnu decays, uses JetSet for fragmentation.
- GOITY_ROBERTS - Decays to non resonant D(*)pi lnu.

BABAR uses, HQET, ISGW2, VUB, and GOITY_ROBERTS in its simulation.
ISGW2 should support D, D_s and B_s decays as well as B decays.

[3]

CP violation in *B* decays

- SSD_CP - generic model for two-body decays that are common final states of the B0 and the anti-B0. Includes effects of both the mass and width differences and should apply equally well to the B_s system.
- SVV_CPLH - Model for decays with two vectors in the final state, e.g. B_s -> J/psi phi.
- BTO3PI_CP, BTO4PI_CP, BTO2PI_CP_ISO, BTOKPI_CP_ISO specialized models.

[3]

Generic amplitudes

- HELAMP, PARTWAVE - generic two-body decays specified by the helicity or partial wave amplitudes.
- SLN - Decay of scalar to lepton and neutrino.
- PHSP - N-body phase space.
- SVS, STS - Scalar decay to vector (or tensor) and scalar.
- VSS, TSS - decay of vector or tensor particle to a pair of scalars.
- VLL, SLL - Decay of vector or scalar to two leptons.
- VSP_PWAVE, vector to scalar and photon, e.g., D*->Dgamma

[3]

Special matrix elements

- BTOXSGAMMA - b->X_s gamma with JetSet fragmentation.
- BTOXSLL - b->X_sll with JetSet fragmentation.
- D_DALITZ - 3-body D-decays with substructure.
- ETA_DALITZ - eta to 3pions with measured dalitz amplitude.
- KSTARNUNU - B->K*nunubar
- LNUGAMMA - B->lnu gamma
- OMEGA_DALITZ - Dalitz structure in the omega->3-pion decay
- PHI_DALITZ - Dalitz structure in the phi->3-pion decay
- PTO3P - scalar to 3 scalars decay where you can specify intermediate resonances
- TAUHADNU - hadronic 1, 2, and 3 pion final states.
- TAULNUNU - leptonic tau decays.
- VSS_BMIX - Upsilon(4S) to BBbar, including mixing.
- VVPIPI - decay of vector to vector and two pions, e.g. psi'->psi+pi+pi.
- VECTORISR - ISR production of vector mesons: e+e- -> V+gamma

[3]

Developers, developers, developers!

How does EvtGen work?

$$\tau \to \pi\nu$$
$$B \to D^*\tau\bar{\nu}$$
$$D^* \to D\pi$$

decay amplitude

$$A = \sum_{\lambda_{D^*}\lambda_\tau} A^{B\to D^*\tau\nu}_{\lambda_{D^*}\lambda_\tau} \times A^{D^*\to D\pi}_{\lambda_{D^*}} \times A^{\tau\to\pi\nu}_{\lambda_\tau}$$

states of spin degrees of freedom

Is replaced with an algorithm that allows each decay to be simulated independently

1 Kinematics are generated according to phase space probability for $B \to D^*\tau\bar{\nu}$ $\quad P_B = \sum_{\lambda_{D^*}\lambda_\tau} |A^{B\to D^*\tau\nu}_{\lambda_{D^*}\lambda_\tau}|^2$

kinematics are regenerated until the event passes an accept-reject algorithm

spin density matrix $\quad \rho^{D^*}_{\lambda_{D^*}\lambda'_{D^*}} = \sum_{\lambda_\tau} A^{B\to D^*\tau\nu}_{\lambda_{D^*}\lambda_\tau} [A^{B\to D^*\tau\nu}_{\lambda'_{D^*}\lambda_\tau}]^*$

describes a D^* from the $B \to D^*\tau\nu$ decay after summing over the degrees of freedom for the τ.

How does EvtGen work?

$$B \to D^*\tau\bar{\nu}, \quad \tau \to \pi\nu, \quad D^* \to D\pi$$

decay amplitude

$$A = \sum_{\lambda_{D^*}\lambda_\tau} A^{B\to D^*\tau\nu}_{\lambda_{D^*}\lambda_\tau} \times A^{D^*\to D\pi}_{\lambda_{D^*}} \times A^{\tau\to\pi\nu}_{\lambda_\tau}$$

states of spin degrees of freedom

Is replaced with an algorithm that allows each decay to be simulated independently

2

Kinematics are generated according to phase space probability for $D^* \to D\pi$

$$P_{D^*} = \frac{1}{\mathrm{Tr}\,\rho^{D^*}} \sum_{\lambda_{D^*}\lambda'_{D^*}} \rho^{D^*}_{\lambda_{D^*}\lambda'_{D^*}} A^{D^*\to D\pi}_{\lambda_{D^*}} [A^{D^*\to D\pi}_{\lambda'_{D^*}}]^*$$

does not affect the angular distributions
makes the maximum decay probability of each sub-decay independent of the full decay chain

kinematics are regenerated until the event passes an accept-reject algorithm

information about the D^* decay $\tilde{\rho}^{D^*}_{\lambda_{D^*}\lambda'_{D^*}} = A^{D^*\to D\pi}_{\lambda_{D^*}} [A^{D^*\to D\pi}_{\lambda'_{D^*}}]^*$

How does EvtGen work?

$$\begin{array}{l} \qquad \tau \to \pi\nu \\ B \to D^* \tau \bar{\nu} \\ \qquad D^* \to D\pi \end{array}$$

decay amplitude

$$A = \sum_{\lambda_{D^*}\lambda_\tau} A^{B\to D^*\tau\nu}_{\lambda_{D^*}\lambda_\tau} \times A^{D^*\to D\pi}_{\lambda_{D^*}} \times A^{\tau\to\pi\nu}_{\lambda_\tau}$$

states of spin degrees of freedom

Is replaced with an algorithm that allows each decay to be simulated independently

③ Kinematics are generated according to phase space probability for $\tau \to \pi\nu$

$$P_\tau = \frac{1}{\mathrm{Tr}\,\rho^{D^*}} \sum_{\lambda_\tau \lambda'_\tau} \rho^{D^*}_{\lambda_\tau\,\lambda'_\tau} A^{\tau\to\pi\nu}_{\lambda_\tau} [A^{\tau\to\pi\nu}_{\lambda'_\tau}]^*$$

does not affect the angular distributions
makes the maximum decay probability of each sub-decay independent of the full decay chain

kinematics are regenerated until the event passes an accept-reject algorithm

spin density matrix of the τ

$$\rho^\tau_{\lambda_\tau\lambda'_\tau} = \sum_{\lambda_{D^*}\lambda'_{D^*}} \tilde{\rho}^{D^*}_{\lambda_{D^*}\lambda'_{D^*}} A^{B\to D^*\tau\nu}_{\lambda_{D^*}\lambda_\tau} [A^{B\to D^*\tau\nu}_{\lambda'_{D^*}\lambda'_\tau}]^*$$

How can I write my own decay model in EvtGen?

You can add decay models to EvtGen as C++ modules in
<PandaRootSourceDir>/pgenerators/EvtGen/EvtGenModels/

Remember

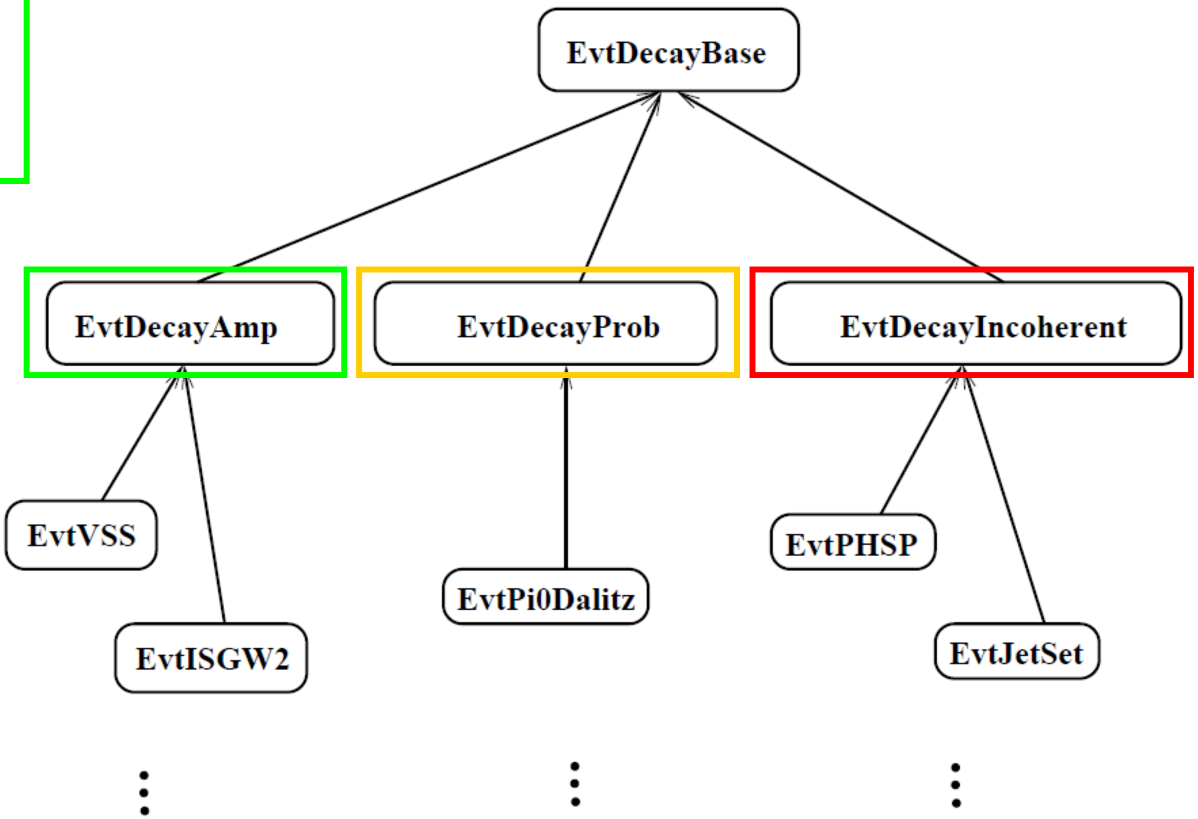
1. Check what exists!
2. Read documentation!
3. Look at the code! (Is the documentation still up to date?)
4. Disclaimer:
   - I am not an expert
   - I have not developed a decay model (ever)
   - My information is based on documentation and may already be outdated
   - I am not responsible for your actions ;)

Derive your decay model from one of the 3 classes below

- Provide amplitude
  - Complete simulation of angular distribution

- Provide probability
  - No polarization
  - No correlation between daughter particles

- Provide 4-vectors
  - Accept all 4-vectors
  - Interface to other generators (Pythia)



What do I need to implement in my decay model (at least)?

- virtual void getName(EvtString& name)=0
  - Returns name of model
- virtual EvtDecayBase* clone()=0
  - Returns a new instance of the decay model class
  - For each entry in the decay table an instance of the decay model class is created
- void initProbMax()
  - Sets maximum probability for decay via setProbMax(double)
  - Used by accept-reject algorithm
  - If value is too low: Distribution of daughter particles will not be correct
  - If value is too high: Decay model will work inefficiently
- virtual void Decay(EvtParticle *p)=0
  - Calculate amplitudes/probabilities/4-vectors for process

What do I need to implement in my decay model (at least)?

- Your decay model has to be registered with the EvtGen framework in order to be used
- In <PandaRootSourceDir>/pgenerators/EvtGen/EvtGenModels/EvtModelReg.cc type
  - modelist.Register(new EvtModelName);
- Add the class in <PandaRootSourceDir>/pgenerators/EvtGen/CMakeLists.txt
  - To set(EVTGEN_SRCS …) add EvtGenModels/EvtModelName.cc
- Model arguments can be accessed using
  - getNArg() (number of arguments)
  - getArg(i) (returns i^{th} argument)

Some useful information

States in EvtGen

- EvtGen works with amplitudes. The amplitudes are specified as amplitudes between the initial and final state in a set of basis vector provided by EvtGen.
- EvtGen uses the following representation for the lower spin states:

Class name	Rep.	J	States	Example
EvtScalarParticle	1	0	1	π, B^0
EvtDiracParticle	u_α	1/2	2	e, τ
EvtNeutrinoParticle	u_α	1/2	1	ν_e
EvtVectorParticle	ϵ^μ	1	3	ρ, J/Ψ
EvtPhotonParticle	ϵ^μ	1	2	γ
EvtTensorParticle	$T^{\mu\nu}$	2	5	D_2^*, f_2

- Also J=3/2 EvtRaritaSchwinger 4 states
- Higher spin states are represented by a generic helicity state basis

[3]

Some useful information

EvtGen support classes for states

- EvtGen provides implementations of several classes that are used to represent the states:
- EvtComplex - implementation of complex numbers
- EvtVector3R, EvtVector3C - real and complex 3-vectors
- EvtVector4R, EvtVector4C - real and complex 4-vectors
- EvtTensor3C, EvtTensor4C - complex second rank tensors
- EvtDiracSpinor - 4-component Dirac spinor
- EvtRaritaSchwinger - Rarita-Schwinger spinor (for spin 3/2 particles)
- EvtGammaMatrix - Dirac gamma matrix implementations

[3]

Naming conventions in EvtGen

- General
  - Do_not_use_underscores or s p a c e s
  - Instead: LeaveOutSpacesAndCapitalizeTheFirstLetterOfEachWord
- Classes and global symbols
  - Start name with “Evt”
  - EvtDecayTable
- Member functions
  - Start with lower case letter
  - getName
- Member data (unlike in PandaRoot)
  - Start name with _underscore
  - _Data

Thank you!