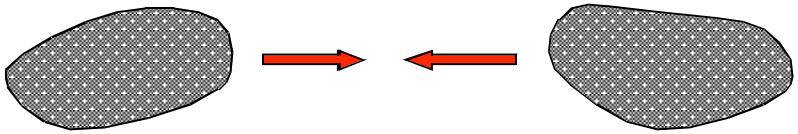
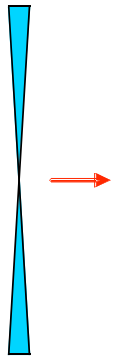
GUINEAPIG :
simulation des effets faisceau-faisceau (collisionneurs linéaires)

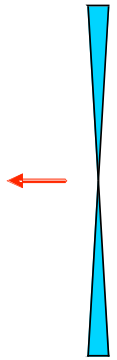
Guy Le Meur

effets faisceau-faisceau (1)

- Faisceaux ultrarelativistes
- Chaque particule (chargée) n'interagit qu'avec les particules du faisceau adverse qui se trouvent dans le même plan transverse qu'elle







Effets faisceau-faisceau (2)

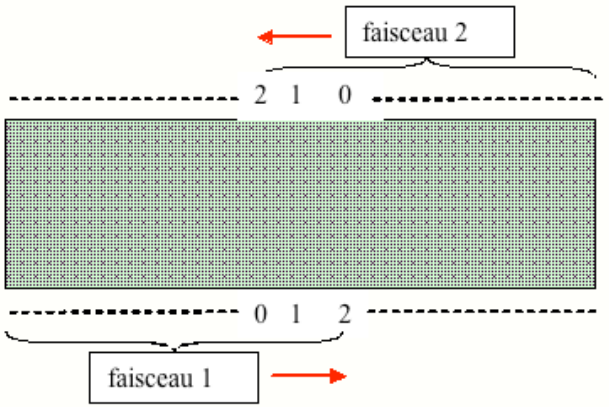
- Dynamique : *pinch effect*
- Photons
- Créations/annihilations de paires e^+/e^-
- Compton
- Rayonnement synchrotron
- Etc. etc.

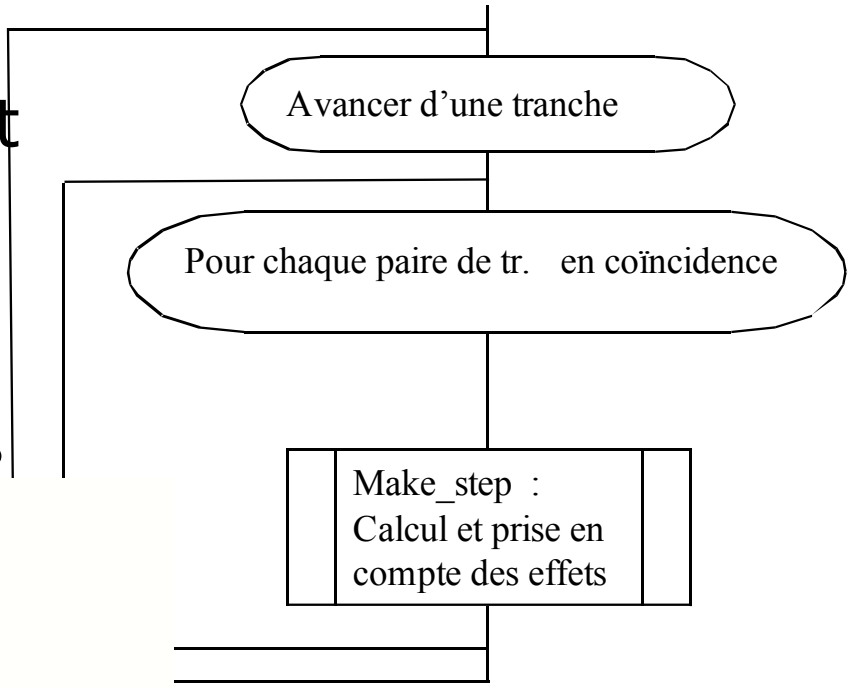
Base du projet

- Un programme écrit à l'occasion d'une thèse (Daniel Schulte, 1996) : guineapig
- Écrit en C
- Un "grand" fichier (deux ou trois "petits")
- Structures statiques globales
- Pas mal de code dupliqué
- Pas de librairie (listes chaînées programmées), sauf éventuellement *FFTW*
- "parsing" des données programmé maison
- Pas de contrôle de la compatibilité des données

Découpage en “tranches” (slices)

- Une tranche d’un faisceau n’interagit qu’avec la tranche du faisceau adverse, qui coïncide avec elles





Make_step(tranche1, tranche2,…)

- Répartir les particules aux noeuds de grille
- Calcul des champs
- Déplacer les particules en fonction des champs (avec génération éventuelle de photons)
- Si (on veut les photons) :
  - distribuer les photons
  - Si (on veut des paires) : générer des paires
  - Déplacer les photons
- Si (on trace les paires)
  - Déplacer les paires
- Quelques menues autres choses (donc des “if”)

Fichier de données

- en entrée :

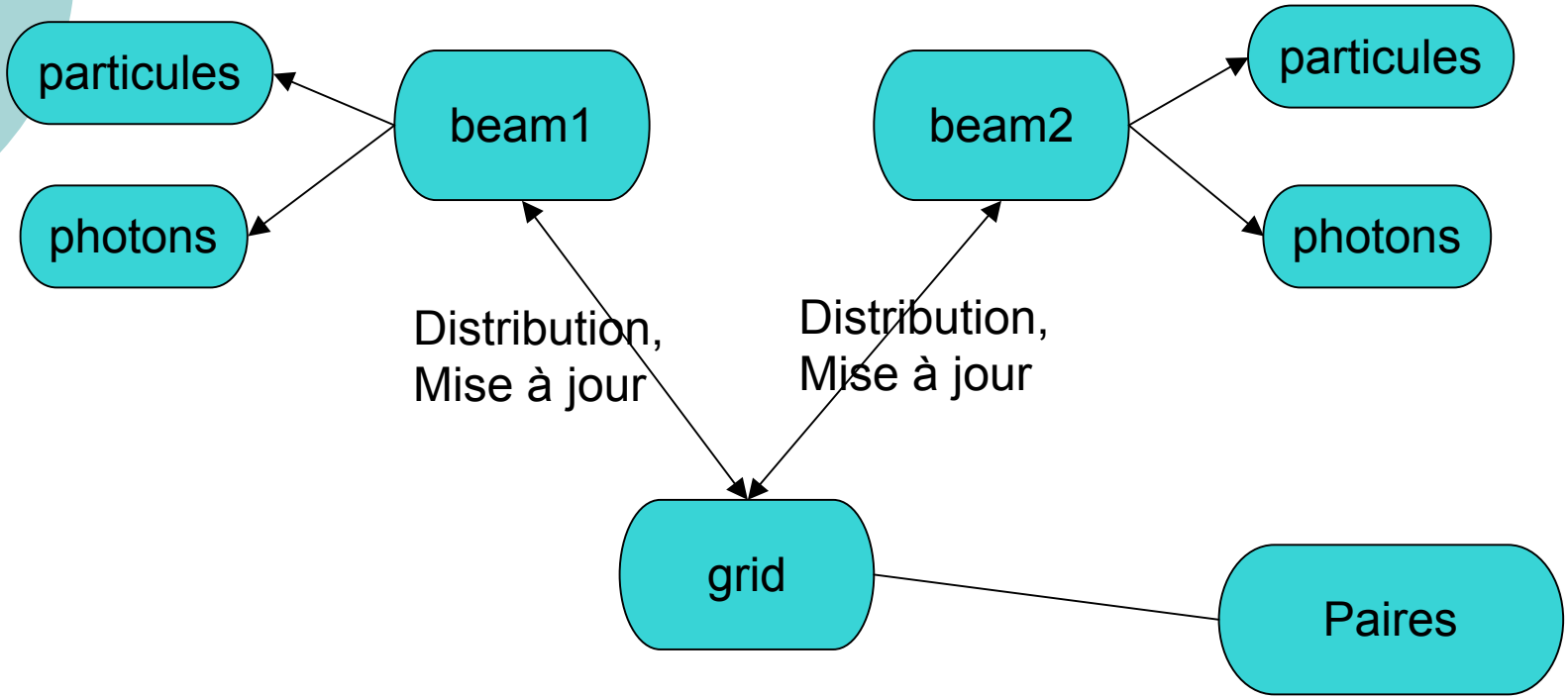
- En sortie : fichiers ascii

```
$ACCELERATOR:: nominal
{
energy=250.0;
particles=2;
beta_x=21.0;
beta_y=0.4;
offset_x=0.;
offset_y=0.;
emitt_x=10.0;
emitt_y=0.04;
sigma_z=300.0;
charge_sign= -1;
}

$PARAMETERS:: pairsprod
{
n_x=32 ;
n_y=64 ;
n_z=32 ;
n_t=5 ;
cut_x=3.0*sigma_x.1 ;
cut_y=6.0*sigma_y.1 ;
cut_z=3.0*sigma_z.1 ;
n_m=10000 ;
force_symmetric=0;
integration_method=2 ;
do_eloss = 1 ;
do_isr = 0;
store_beam=1 ;
electron_ratio=1. ;
do_photons=1 ;
photon_ratio=1. ;
store_photons=1 ;
store_pairs=2;
do_pairs=1 ;
track_pairs=1;
grids=7 ;
pair_ratio = 1.;
pair_ecut = 0.005;
pair_q2=2;
beam_size=1;
ext_field=0;
do_compt_phot = 0;
do_hadrons=0 ;
store_hadrons = 0 ;
do_jets=0 ;
store_jets=0 ;
rndm_seed=0;
rndm_load=0;
rndm_save = 0;

}
```

Les données


particules
photons
beam1
beam2
particules
photons
Distribution,
Mise à jour
Distribution,
Mise à jour
grid
Paires

Objectifs du projet

- Créer un outil de simulation fiable et évolutif
- Ajouter de nouvelles fonctionalités :
  - Bhabhas
  - Dépolarisation
  - …
- Améliorer les performances en vue de simulations massives pour ILC: grille? Parallélisme?

Passage en C++

- Premier temps :
  - Désintriquer en créant des classes : modularité
  - calquer les classes (plus ou moins) sur les structures existantes :
    - Grid, Beam, Particle_beam, Photon, Pair_particle, etc. etc.
  - Utilisation de STL
- Deuxième temps : “restructurer”

Modernisation

- Interface utilisateur graphique : commodité et contrôle de la cohérence des données
- Analyse graphique des résultats : histogrammes!

Utilisation d'outils

- Gestion de configuration : CMT?
- Svn?
- Documentation : doxygen
- Trac?