Particles in OpenFOAM

- dieselSpray
  - Used for liquid particles, eg fuel injection
  - Well implemented collision models
    - O'Rourke collision model
    - Trajectory collision model
- solidParticle
  - Used for solid particle, eg ash, dust..
  - NO particle interaction

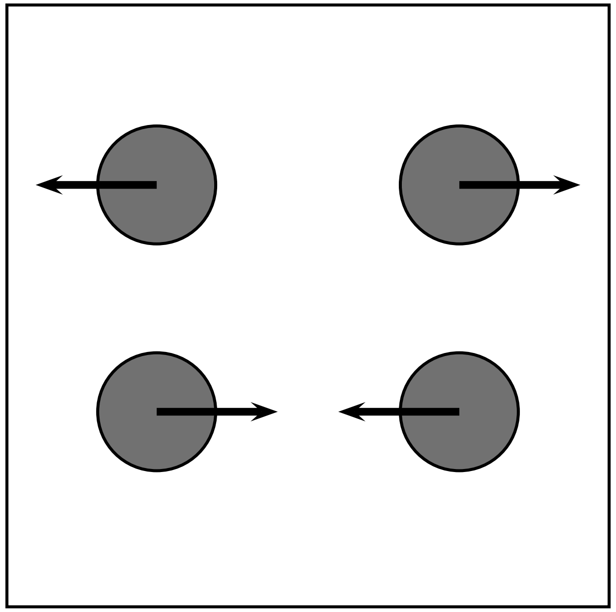
dieselSpray collision models

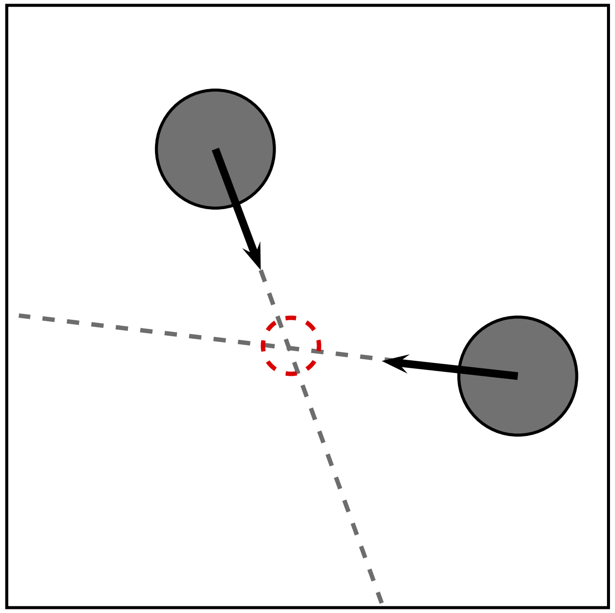
O'Rourke

- Collision if in same cell
- Disregarding particle direction

Trajectory

- Collision if in same cell
- Takes direction into account
- Checking possibilty of collision within timestep





Selecting dieselSpray collision model

Go to the case you want to run, `$FOAM_TUTORIALS/dieselFoam/aachenBomb` for example, and make the following changes in `constant/sprayProperties`

```
collisionModel  ORourke;//off;
ORourkeCoeffs
{
    coalescence     off;
}
```

or

```
collisionModel  trajectory;//off;
trajectoryCoeffs
{
    cSpace          1;
    cTime           0.3;
    coalescence     off;
}
```

`coalenscence` is used to determine if droplets can merge upon collision or not.

A model for solid particles

$$u_{p1}^* = u_{p1} + \frac{J_x}{m_{p1}}, \qquad u_{p2}^* = -\frac{J_x}{m_{p2}}$$

$$v_{p1}^* = v_{p1} + \frac{J_y}{m_{p1}}, \qquad v_{p2}^* = -\frac{J_y}{m_{p2}}$$

$$w_{p1}^* = +\frac{J_z}{m_{p1}} = 0, \quad w_{p2}^* = -\frac{J_z}{m_{p2}} = 0$$

$$J_x = -(1+e)u_{p1}\frac{m_{p1}m_{p2}}{m_{p1}+m_{p2}}$$

$$J_y = -\frac{2}{7}v_{p1}\frac{m_{p1}m_{p2}}{m_{p1}+m_{p2}}$$

$$J_z = 0$$

where e is the coefficient of restitution.

Collision if two particles occypy the same cell and the distance between them is less than their diameter.

Create collidingSolidParticleFoam

```
cd $FOAM_RUN
svn checkout http://openfoam-extend.svn.sourceforge.net/svnroot/\
openfoam-extend/trunk/Breeder_1.5/solvers/other/solidParticleFoam/
cd solidParticleFoam/
```

Rename the directory for the new class and copy the needed solidParticle files into it.

```
mv solidParticleFoam collidingSolidParticleFoam
cd collidingSolidParticleFoam
wclean
cp $FOAM_SRC/lagrangian/solidParticle/solidParticle* .
cp -r $FOAM_SRC/lagrangian/solidParticle/lnInclude .
```

Do a word replacement from solidParticle to collidingSolidParticle in all the files.

```
sed -i s/solidParticle/collidingSolidParticle/g solidParticle.C \
solidParticleCloud.C solidParticleCloud.H solidParticleCloudI.H \
solidParticleFoam.C solidParticle.H solidParticleI.H solidParticleIO.C
```

Create collidingSolidParticleFoam

Rename the files

```
mv solidParticle.C collidingSolidParticle.C
mv solidParticleCloud.C collidingSolidParticleCloud.C
mv solidParticleCloud.H collidingSolidParticleCloud.H
mv solidParticleCloudI.H collidingSolidParticleCloudI.H
mv solidParticleFoam.C collidingSolidParticleFoam.C
mv solidParticle.H collidingSolidParticle.H
mv solidParticleI.H collidingSolidParticleI.H
mv solidParticleIO.C collidingSolidParticleIO.C
```

Edit Make/files

```
collidingSolidParticleFoam.C
collidingSolidParticle.C
collidingSolidParticleIO.C
collidingSolidParticleCloud.C
EXE = $(FOAM_USER_APPBIN)/collidingSolidParticleFoam
```

Create collidingSolidParticleFoam

and Make/options

```
EXE_INC = \
-I$(LIB_SRC)/finiteVolume/lnInclude \
-I$(LIB_SRC)/lagrangian/basic/lnInclude
EXE_LIBS = \
-lfiniteVolume \
-llagrangian
```

You can try compiling and running the case. Nothing should have changed from the original

```
wmake
cd ../box
blockMesh
collidingSolidParticleFoam >log

foamToVTK
paraview
```

Load by `File>Load State>baseState.pvsm`

collidingSolidParticleCloud

In `collidingSolidParticleCloud.C`, add

```
void Foam::collidingSolidParticleCloud::checkCell()
{
  List<label> lcell((*this).size());
  List<scalar> ld((*this).size());
  List<vector> lU((*this).size());
  List<vector> lposition((*this).size());
  bool collision;
  label i=0;
  forAllConstIter(Cloud<collidingSolidParticle>,*this,iter)
  {
      const collidingSolidParticle& p=iter();
      lcell[i]=p.cell();
    //Info <<"Particle " <<i<< " is in cell "<<lcell[i]<<endl;
      lU[i]=p.U();
      ld[i]=p.d();
      lposition[i]=p.position();
      i++;
  }
```

```
      //Info <<"p0-p1 = "<<mag(lposition[0]-lposition[1])<<endl;
      //Info <<"Diameter = "<<(ld[0]+ld[1])/2<<endl;

        //Only works for two particles.
if (lcell[0]==lcell[1] && mag(lposition[0]-lposition[1])<=(ld[0]+ld[1])/2)
        {
            collision=true;
        }
        else
        {
            collision=false;
        }

        collision_=collision;

    U0_=lU[0];
  }
```

Also add `#include "vector.H"` in the header of the file.

collidingSolidParticleCloud

In `collidingSolidParticleCloud.H`, add

```
bool collision_;
vector U0_;
```

to the private member data, and

```
void checkCell();
bool collision(){return collision_;};
inline vector  U0(){return U0_;};
```

to public member functions.

collidingSolidParticle

In `collidingSolidParticle.C` replace

```
U_ = (U_ + dt*(Dc*Uc + (1.0 - rhoc/rhop)*td.g()))/(1.0 + dt*Dc);
```

with

```
        scalar e = td.spc().e();                         //Restitution coefficient
        scalar m = rhop*d_*d_*d_*mathematicalConstant::pi*4.0/3.0; //Mass
        bool checkcoll=td.spc().collision();
        //if(checkcoll){Info<<"Particles collide!"<<endl;}
        vector V0=td.spc().U0(); //OLD velocity of particle 1

        scalar Jx = -(1.0+e)*V0.x()*m*m/(2.0*m);//Impulsive force x-comp
        scalar Jy = -2.0/7.0*V0.y()*m*m/(2.0*m);//Impulsive force y-comp

        if(ID_==0 && checkcoll) //Collision for particle 1
        {
            U_.x()=V0.x()+Jx/m;
            U_.y()=V0.y()+Jy/m;
            U_.z()=0.0;
        }
```

collidingSolidParticle

```
if(ID_==1 && checkcoll) //Collision for particle 2
{
    U_.x() = -Jx/m;
    U_.y() = -Jy/m;
    U_.z() = 0.0;
}

//If no collision
U_ = (U_ + dt*(Dc*Uc + (1.0 - rhoc/rhop)*td.g()))/(1.0 + dt*Dc);
```

Add

```
//- Particle ID
scalar ID_;
```

to the private member data in collidingSolidParticle.H.

collidingSolidParticle

In `collidingSolidParticleIO.C` add

```
IOField<scalar> ID(c.fieldIOobject("ID"));
c.checkFieldIOobject(c, ID);
```

and

```
p.ID_ = ID[i];
```

to the readFields function.

Add

```
IOField<scalar> ID(c.fieldIOobject("ID"), np);
```

and

```
ID[i] = p.ID_;
```

and

```
ID.write();
```

to the writeFields function.

collidingSolidParticleFoam

In `collidingSolidParticleFoam.C` before `particles.move(g);`
add

```
particles.checkCell();
```

Compile again

```
wmake
```

Now set up a case..

Modify box

In `../box/constant/polyMesh/blockMeshDict` change the blocking to

```
hex (0 1 2 3 4 5 6 7) (3 3 3) simpleGrading (1 1 1)
```

Run `blockMesh` to get a new mesh.

In `0/lagrangian/defaultCloud/`

```
cp d ID
```

Change context of ID to

```
FoamFile
{
  version     2.0;
  format      ascii;
  class       scalarField;
  location    "0";
  object      ID;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
2(0 1)
// ***************************************************************** //
```

Modify box

In the `U` file, change the velocities to

```
2((1.0 0 0) (-1.0 0 0))
```

In `positions`, set the coordinates to

```
2((1e-2 9e-2 0.05) 15 (7e-2 9e-2 0.05) 16)
```

Run the case from `box` directory

```
collidingSolidParticleFoam >log\
```

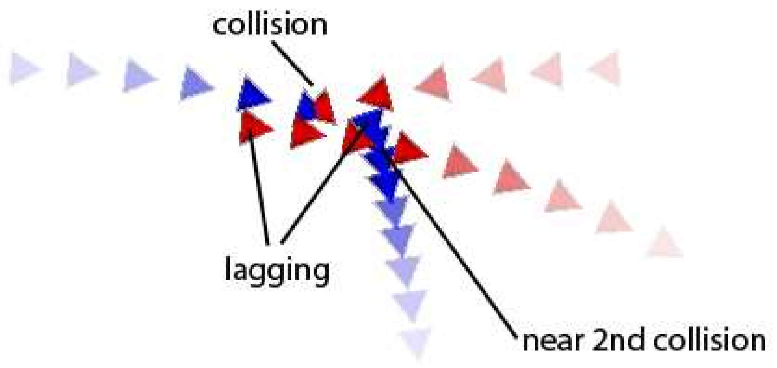
To postprocess, type

```
foamToVTK
paraview
```

Load by clicking `File>Load State>baseState.pvsm`.

Future work

- Switch velocity at collision



- Generalize for more particles
- Improve collision model