Yade: Past, Present, Future

Václav Šmilauer

CTU Prague & UJF Grenoble

12 March 2010

跷 marks advanced topics
contains clickable hyperlinks to documentation and websites
Outline

1 Past

2 Present
   - Python intro
   - Simulation structure
   - Simulation description
   - Preprocess
   - Process
   - Postprocess
   - Functionality walkthrough

3 Future

4 Researchers using Yade
   - Past projects
   - Present projects
DEM & (Pre)history

- DEM: explicit dynamics of particles
- Simple discontinuum models
  - Cundall 1979: nondeformable discs, 2d, explicit dynamics, penalty contact function
  - Frédéric Donzé: Spherical Discrete Element Code
  - Yade starts in 2004, “flexible platform” (J. Kozicki, O. Galizzi)
DEM & (Pre)history

- **DEM**: explicit dynamics of particles
- Simple discontinuum models
- Cundall 1979: nondeformable discs, 2d, explicit dynamics, penalty contact function
- Frédéric Donzé: Spherical Discrete Element Code
- Yade starts in 2004, “flexible platform” (J. Kozicki, O. Galizzi)

**Past**
- DEM & (Pre)history
- Václav Šmilauer
- Past
- Present
- Python intro
- Simulation structure
- Simulation description
- Preprocess
- Process
- Postprocess
- Functionality walkthrough

**Future**
- Researchers using Yade
- Past projects
- Present projects
DEM & (Pre)history

- DEM: explicit dynamics of particles
- Simple discontinuum models
- Cundall 1979: nondeformable discs, 2d, explicit dynamics, penalty contact function
- Frédéric Donzé: Spherical Discrete Element Code
- Yade starts in 2004, “flexible platform” (J. Kozicki, O. Galizzi)
DEM & (Pre)history

- DEM: explicit dynamics of particles
- Simple discontinuum models
- Cundall 1979: nondeformable discs, 2d, explicit dynamics, penalty contact function
- Frédéric Donzé: Spherical Discrete Element Code
- Yade starts in 2004, “flexible platform” (J. Kozicki, O. Galizzi)
Yade beginnings

- Written in c++, running on Linux/Unix
- Proof-of-concept implementations: DEM, FEM, mass-spring, lattice
- No documentation
- Sometimes functionally questionable
- Demanding on programming skills for “users”
- Object-oriented design

www.yade-dem.org
launchpad.net/yade
Sanitization period (2007-2010)

- Motivated by our development of concrete model
- Removing bad code
- Enforcement of consistent names
- Parallel computation
- Documentation
- Python scripting
Outline

1. Past
2. Present
   - Python intro
   - Simulation structure
   - Simulation description
   - Preprocess
   - Process
   - Postprocess
   - Functionality walkthrough
3. Future
4. Researchers using Yade
   - Past projects
   - Present projects
Python

- Scripting (non-compiled) object-oriented language
- Large documented standard library
- Easy to interface with fortran/c/c++
- Language of choice for many scientific projects (similar to matlab)
Python in Yade

- c++ classes mirrored in python, with full attribute access
- scripts efficient for simulation setup, postprocessing
- compatible over many internal changes
- runtime control & debugging from the command line
# Data components

## Body (particle)

<table>
<thead>
<tr>
<th>Shape</th>
<th>Sphere, Facet, ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material</td>
<td>ElastMat, FrictMat, ...</td>
</tr>
<tr>
<td>State</td>
<td>position, orientation, velocity, ...</td>
</tr>
<tr>
<td>Bound</td>
<td>for approximate collision detection (Aabb)</td>
</tr>
</tbody>
</table>

## Generalized forces

## Interaction of 2 bodies

<table>
<thead>
<tr>
<th>InteractionGeometry</th>
<th>different for Sphere+Sphere, Facet+Sphere, ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>InteractionPhysics</td>
<td>internal state of interaction (plasticity variables, damage, history)</td>
</tr>
</tbody>
</table>
Functional components

Engine

- GlobalEngine act on all bodies/interactions
- PartialEngine act on some bodies/interactions
- Dispatcher calls functions based on classes of arguments: e.g. Facet+Sphere needs different function than Sphere+Sphere collision

Functors

Callable function-like objects. Accept only certain classes and are called by Dispatchers.
Simulation structure

- **bodies**
  - Shape
  - Material
  - State
  - Bound

- **forces** (generalized)
  - other forces (gravity, BC, ...)

- **forces → acceleration**

- **velocity update**

- **position update**

- **miscellaneous engines** (recorders, ...)

- **simulation loop**
  - increment time by $\Delta t$

- **interactions**
  - collision detection pass 1
  - collision detection pass 2
  - strain evaluation
  - physics properties of new interactions
  - compute forces from strains

- **other forces**

- **update bounds**

- **reset forces**
What it looks like in python I.

Simulation loop in code

```
O.engines=[
    ForceResetter(),
    BoundDispatcher([Bo1_Sphere_Aabb(),Bo1_Facet_Aabb()]),
    InsertionSortCollider(),
    InteractionDispatchers(
        [Ig2_Sphere_Sphere_Dem3DofGeom(),
        Ig2_Facet_Sphere_Dem3DofGeom()],
        [Ip2_FrictMat_FrictMat_FrictPhys()],
        [Law2_Dem3Dof_FrictPhys_Basic()],
    ),
    GravityEngine(gravity=(0,0,-9.81)),
    NewtonIntegrator()
]
```
Functor names explained

**Ig2 Facet Sphere Dem3DofGeom**

- *Ig2* 2-ary functor creating *InteractionGeometry*
- *Facet* accepting a *Facet* as first argument
- *Sphere* and *Sphere* as second argument
- *Dem3DofGeom* returning *Dem3DofGeom* instance
<table>
<thead>
<tr>
<th>Type of Functor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BoundFunctor</strong></td>
<td>approximate volume representation, for fast collision detection</td>
</tr>
<tr>
<td><strong>InteractionGeometryFunctor</strong></td>
<td>resolves geometry of interaction (e.g. displacement, shear), based on Shapes of bodies</td>
</tr>
<tr>
<td><strong>InteractionPhysicsFunctor</strong></td>
<td>derives properties of interaction, i.e. creates InteractionPhysics for given particles’ Materials</td>
</tr>
<tr>
<td><strong>LawFunctor</strong></td>
<td>resolves forces on particles, using InteractionGeometry and InteractionPhysics of some types, created by previous functors.</td>
</tr>
</tbody>
</table>
What it looks like in python II.

Simulation data in code

```python
O.materials.append(
    FrictMat(young=30e9, poisson=.3, density=3000, frictionAngle=.5)
)
O.bodies.append([utils.sphere((0,0,3),radius=1),
                 utils.facet([[-1,-1,0],[1,0,0],[0,1,0]])
])
O.dt=.5*utils.PWaveTimeStep()
```

Running simulation

```python
O.run(10000); O.wait()  # Basic simulation control
O.save('/tmp/a.xml')
print O.bodies[3].state.vel  # inspection of (c++) data
print O.interactions[0,2].geom.normal
print O.materials[0].young
quit()
```
“Meshing” volumes with spheres

See **horse** (surface import), **mill** (“by hand”)

### Volume representation

- **Boundary**: triangulated surface; imported (STL, GTS, gmsh) / created “by hand” (possibly parametric)
- **Volume**: constructive solid geometry, boolean composition

### Sphere packing generators (decoupled from volume)

- Import packing (text, LSMGenGeo)
- Dynamic: triax compression/decompression, gravity
- Geometric: from tetrahedron mesh (SpherePadder), from boundary specification (LSMGenGeo)
“Meshing” volumes with spheres (2)

Solid representation

```python
predicate = pack.inSphere((0,0,0),1)
```

Boundary representation

```python
predicate = pack.inGtsSurface(gts.read(open('horse.coarse.gts')))
```

Boolean composition (intersection &, union |, difference -)

```python
predicate = pack.inSphere((0,0,0),1) & pack.inCylinder((.5,0,-1),(.5,0,1),.5)
```

Call packing generator with arbitrary predicate

```python
spheres = pack.randomDensePack(pack.inHyperboloid((0,0,-.1),(0,0,.1),.05,.085), spheresInCell=2000, radius=3.5e-3)
O.bodies.append(spheres)
```
Sphere falling through funnel

source script, movie
Running, controlling, collecting

Collecting data

```python
O.engines=[
    PeriodicPythonRunner(command='addPlotData()', iterPeriod=100),
    PeriodicPythonRunner(command='checkPostpeak()', realPeriod=3),
]

def addPlotData():
    plot.addData(eps=strainer.strain, sigma=strainer.avgStress)

plot.plots={'eps':('sigma',)}  # define what to plot
```

Controlling simulation from within the loop

```python
def checkPostpeak():
    maxSigma=max(maxSigma, strainer.sigma)
    if strainer.sigma<.5*maxSigma:  # check some condition
        print "Damaged, exiting. Peak stress was", maxSigma
        plot.saveGnuplot('damaged')  # save curves for postprocessing
    import sys; sys.exit(0)
```
Postprocessing

1d `yade.plot` module: `matplotlib; Gnuplot`

2d `yade.post2d` module

3d built-in OpenGL view; `VTKRecorder, with Paraview` (slices, movie export, ...)

```
-4e+07 -3.5e+07 -3e+07 -2.5e+07 -2e+07 -1.5e+07 -1e+07 -5e+06 0 5e+06
-0.003 -0.0025 -0.002 -0.0015 -0.001 -0.0005 0 0.0005
sigma eps
```
Postprocessing

1d `yade.plot` module: `matplotlib`; `Gnuplot`

2d `yade.post2d` module

3d built-in OpenGL view; `VTKRecorder`, with `Paraview` (slices, movie export, ...)

Researchers using Yade
Past projects
Present projects
Postprocessing

1d `yade.plot` module: `matplotlib`; `Gnuplot`

2d `yade.post2d` module

3d built-in OpenGL view; `VTKRecorder`, with `Paraview` (slices, movie export, . . .)
## Yade landscape

### Community

- website, wiki, bugs and specifications tracking
- responsive mailing lists for users and developers, $\approx 10$ messages/day
- used at multiple institutes, mostly research

### Code

- central code repository with history
- documented code structure (in progress)
- documentation of c++/python classes and python modules
- Linux/Unix only
Generalities

Performance

- Shared-memory parallelism using OpenMP; speedup depending on scenario & machine, \( \approx 5 \times \) on 8 cores.
- Profiling tools (\texttt{yade.timing})

Usability

- Batch scheduling and execution (parametric studies)
- Remote watching and control over http and telnet
- Debugging tools (\texttt{yade.log}), embedded debugger
- Embedded \texttt{ipython} shell
**Engines**

**Loading control**


**Applying conditions**

*GravityEngine* (constant gravity field), *ForceEngine*, *RotationEngine*, *TranslationEngine*, ...

**Algorithms**

*InsertionSortCollider* (collision detection), *NewtonIntegrator* (2nd order central-differences explicit integration scheme), *GlobalStiffnessTimeStepper* (adjust timestep based on packing stiffness)
Particles and interactions

Shapes

Sphere, Facet, Wall, Box. (Tetra, polyhedral grains, ...).

Handling collisions (InteractionGeometry)

Handling collisions of $2 \times$ Sphere, Facet+Sphere, Box+Sphere, Wall+Sphere.

Constitutive laws

Dry friction (classical DEM), Mindlin’s contact, Plassiard’s formulation, Cohesive-frictional model, rock model, concrete model, capillary effects between grains. (more outside source tree or undocumented)

Coupling

OpenFOAM, Comsol, fluids.
What a constitutive law looks like

```cpp
void Law2_Dem3Dof_Elastic_Elastic::go(shared_ptr<InteractionGeometry>& ig, shared_ptr<InteractionPhysics>& ip,
Interaction* I, Scene* scene){
    // init
    Dem3DofGeom* geom=static_cast<Dem3DofGeom*>(ig.get());
    ElasticContactInteraction* phys=static_cast<ElasticContactInteraction*>(ip.get());

    // compute normal displacement
    Real displN=geom->displacementN();

    // delete interaction with positive deformation
    if(displN>0){rootBody->interactions->requestErase(I->getId1(),I->getId2()); return; }

    // compute normal force
    phys->normalForce=phys->kn*displN*geom->normal;

    // compute maximum and trial shear force
    Real maxFsSq=phys->normalForce.SquaredLength()*pow(phys->tangensOfFrictionAngle,2);
    Vector3r trialFs=phys->ks*geom->displacementT();

    // plastic slip if necessary
    if(trialFs.SquaredLength()>maxFsSq){
        geom->slipToDisplacementTMax(sqrt(maxFsSq));
        trialFs*=maxFsSq/(trialFs.SquaredLength());
    }

    // apply forces
    applyForceAtContactPoint(
        phys->normalForce+trialFs,geom->contactPoint,
        I->getId1(),geom->se31.position,
        I->getId2(),geom->se32.position,scene
    );

    // finito
}
```
Outline

1. Past
   - Python intro
   - Simulation structure
   - Simulation description
   - Preprocess
   - Process
   - Postprocess
   - Functionality walkthrough

2. Present
   - Python intro
   - Simulation structure
   - Simulation description
   - Preprocess
   - Process
   - Postprocess
   - Functionality walkthrough

3. Future

4. Researchers using Yade
   - Past projects
   - Present projects
Future

Continue maintenance

- documentation
- code cleanup
- Improve performance

Becoming reference platform for discrete models

- Reusable common functionality (e.g. deformation computation, collision detection, integrator, ...)
- Encourage cooperation via python (numpy).
- Integrate couplings with external software (OpenFOAM, ...).
Outline

1. Past
   - Python intro
   - Simulation structure
   - Simulation description
   - Preprocess
   - Process
   - Postprocess
   - Functionality walkthrough

2. Present
   - Present projects

3. Future

4. Researchers using Yade
   - Past projects
   - Present projects
3d lattice model of tensile concrete fracture.
Behavior of granular media with capillary effects between grains.
Wenjie Shiu, Grenoble

Missile impact on concrete structures.
Wenjie Shiu, Grenoble

Missile impact on concrete structures.
A finite volumes-DEM coupled formulation for fluid-solid interactions in granular media.
Benoît Charlas, Grenoble

Hydride metal powders in hydrogen storage tanks — swelling & shrinking due to chemical reactions with hydrogen, creating mechanical effects.
Coupling Computational Flow Dynamics (CFD) and DEM — OpenFOAM and Yade.
Anton Gladky, Freiberg

Mineral processing — analyzing rock destruction in the machine.
Mineral processing — analyzing rock destruction in the machine.
Modeling snow grains based on CT scans, as polyhedra which can deform along crystallographic planes.
Interaction between DEM-modeled solid and Lattice Boltzmann Method (LBM) modeled fluid. (Started by Luc Scholtès)
Fractured rock mass with smooth contact discontinuities; discontinuities can be imported from Discrete Fracture Network Modelers.
Fractured rock mass with smooth contact discontinuities; discontinuities can be imported from Discrete Fracture Network Modelers.
Particle model of concrete, based on continuous formulation (plasticity, rate-dependence, damage).
Thanks for attention

Got questions

Ask them at
yade-users@lists.launchpad.net