

# unfold

production example

visual material::shockwave simulation data

## 1. original simulation data

shockwave simulation

voxel size : 400 x 400 x 400

total simulation frame : 94 frame (024-094)

file format : hdf5

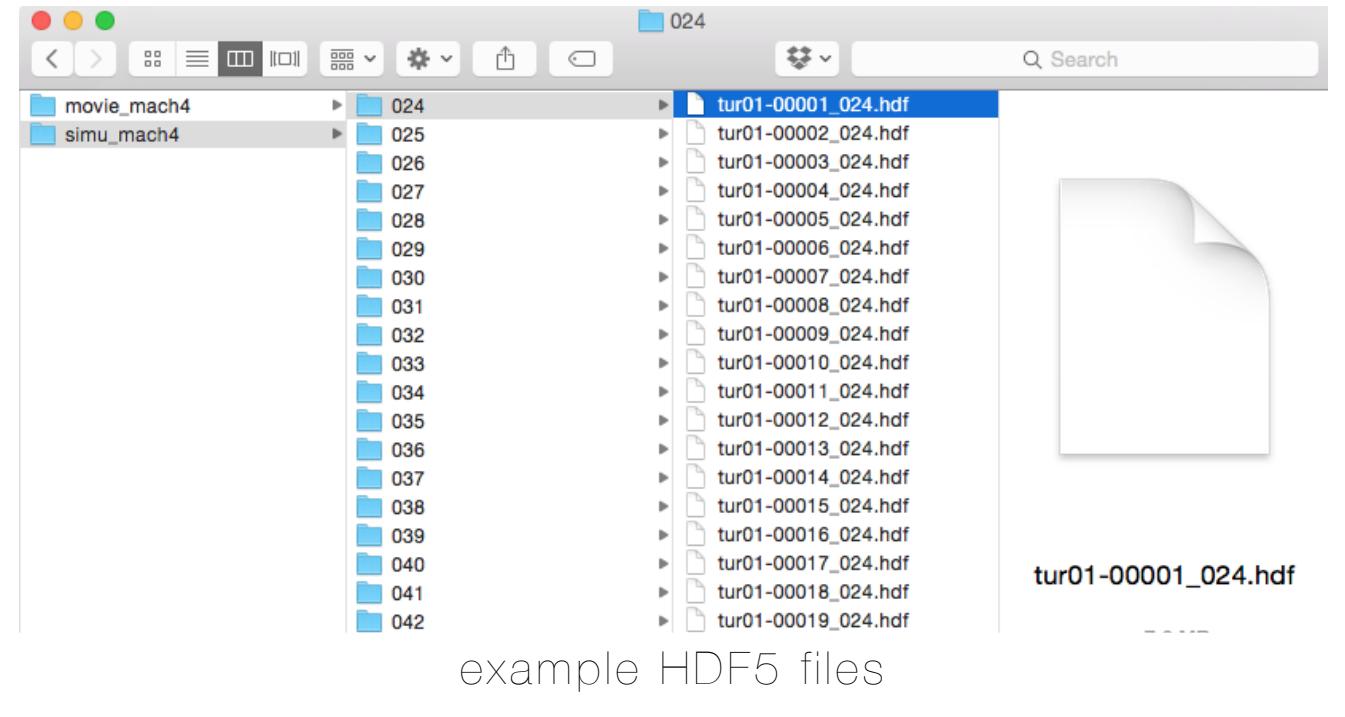
parameter : density, velocity, energy, ionization

file per voxel : 50 x 50 x 50

number of files per frame : 512 files

1 simulation frame is splitted into 512 files.

In total 94 frame x 512 = 48,128 file.



example HDF5 files

## 2. simulation data example

rho : density

u : velocity(x,y,z)

E : energy

fx : ionization

```
1 # opt[0]: rho
2 # opt[1]: u
3 # opt[2]: E
4 # opt[3]: fx
5
6 import sys
7 from pylab import *
8 import rd_heracles as rd
9
10 data=rd.HeraclesData(idump='044', form='hdf', opt=[1,1,1,1,0,0], dir='..')
11 simu_mach4'
12
13 print data.rho[0,0,0]
14 print data.u[0,0,0]
15 print data.E[0,0,0]
16 print data.fx[0,0,0]
```

python code showing parameter at [0,0,0]

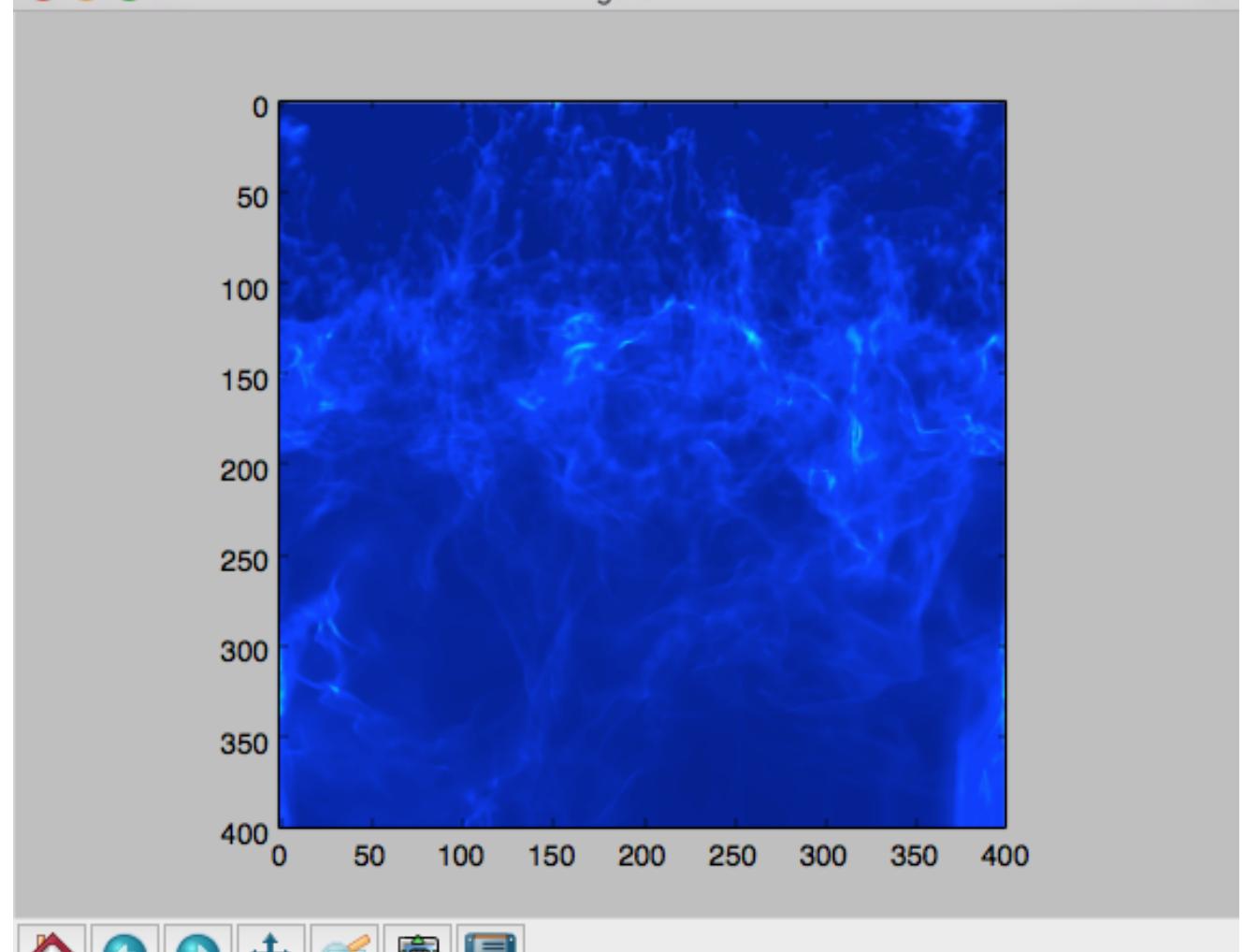
```
Last login: Fri Dec 11 01:45:23 on ttys001
mtb-iMac:~ mtb$ cd /Volumes/StudioRK_unfold/unfold/simulation_data/heracles_simulation
mtb-iMac:heracles_simulation mtb$ cd code/
mtb-iMac:code mtb$ python viewer.py
Reading Heracles hdf5 outputs
Reading 512 files
7.18444209686e-23
[-8010.77501493 187662.18391449 -1100220.60105105]
1.42183950486e-10
[ 0.99873033]
```

result

## 3. visualization for scientific usage

```
1 # opt[0]: rho
2 # opt[1]: u
3 # opt[2]: E
4 # opt[3]: fx
5
6 import sys
7 from pylab import *
8 import rd_heracles as rd
9
10 ion()
11 data=rd.HeraclesData(idump='044', form='hdf', opt=[1,1,1,1,0,0], dir='..')
12 simu_mach4'
13
14 imshow(sum(data.rho[:, :, :], 2))
```

python code for visualization



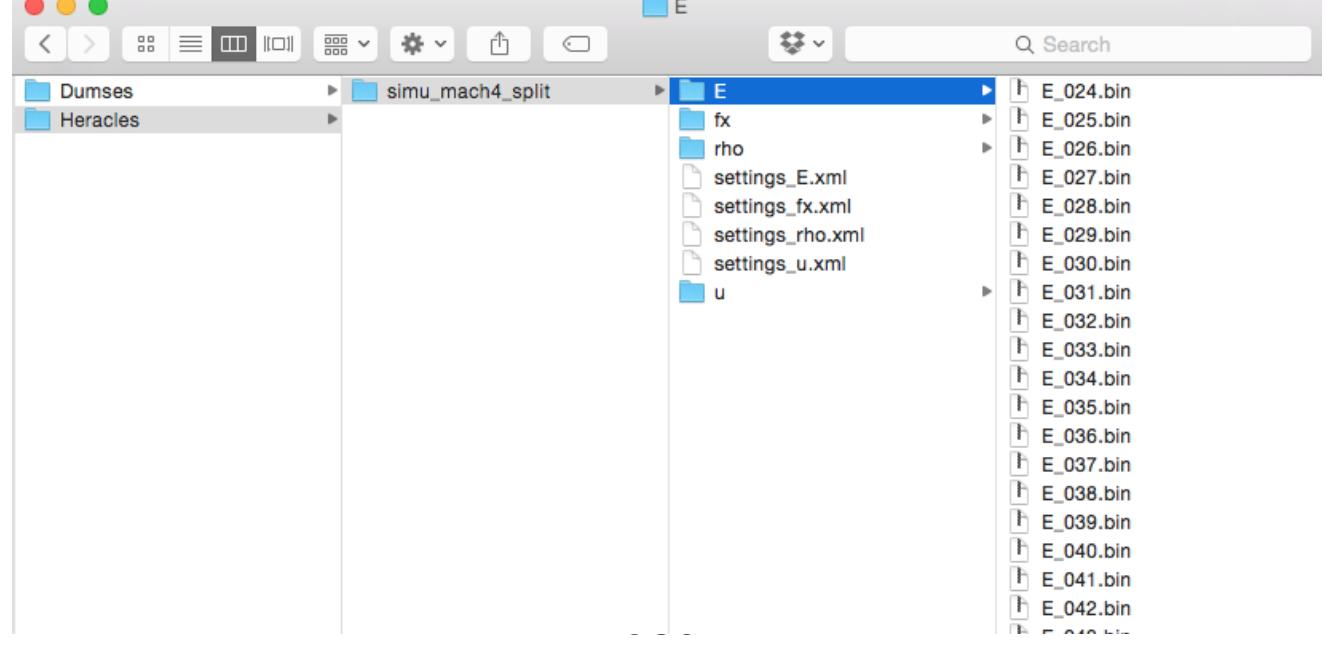
2D visualization of density

4. split/convert to binary file

split parameter and convert hdf5 to binary for C++ app.

```
1 import sys
2 from pylab import *
3 import rd_heracles as rd
4 from multiprocessing import Pool
5
6 def split_param( idump ) :
7     dirName = '0' + str(idump)
8     print( "start : " + dirName )
9     data=rd.HeraclesData(idump=dirName, form='hdf', opt=[1,1,1,1,0,0,0], dir='../simu_mach4')
10
11    file = open( '../split/rho/rho_' + dirName + '.bin', 'w' )
12    file.write(data.rho)
13    file.close()
14
15    file = open( '../split/u/u_' + dirName + '.bin', 'w' )
16    file.write(data.u)
17    file.close()
18
19    file = open( '../split/E/E_' + dirName + '.bin', 'w' )
20    file.write(data.E)
21    file.close()
22
23    file = open( '../split/fx/fx_' + dirName + '.bin', 'w' )
24    file.write(data.fx)
25    file.close()
26    print( "finish : " + dirName )
27    print("\n\n")
28
29 if __name__ == "__main__":
30
31     range_st = int(sys.argv[1])
32     range_end = int(sys.argv[2])
33     rng = range( range_st, range_end )
34
35     processor = 4
36     p = Pool( processor )
37     p.map( split_param, rng )
38     print( "\n\nFINISH ALL :" + str(range_st) + "-" + str(range_end) + "\n" )
39
```

python code for split and convert



## 5. load to C++ framework

## Example visualization process

1. Load density data
  2. apply log calculation
  3. separate to 7 groups based on parameter range
  4. add pseudo color for each groups
  5. add line between groups

The screenshot shows the Xcode interface with the project 'loadsim\_2' selected. The left sidebar displays the project structure, including targets ('debug.xcconfig', 'release.xcconfig'), info.plist, source files ('cApp.cpp', 'DataGroup.h'), and header files ('Blocks', 'libmtb'). The main editor window shows the 'cApp.cpp' file with the following content:

```
#define RENDER

#include "cinder/app/AppNative.h"
#include "cinder/Rand.h"
#include "cinder/Utilities.h"
#include "cinder/gl/gl.h"
#include "cinder/gl/Fbo.h"
#include "cinder/gl/Texture.h"
#include "cinder/Camera.h"
#include "cinder/MayaCamUI.h"
#include "cinder/Perlin.h"
#include "cinder/params/Params.h"
#include "CinderOpenCv.h"

#include "mtUtil.h"
#include "ConsoleColor.h"
#include "Exporter.h"
#include "DataGroup.h"

#include <iostream>
#include <fstream>

using namespace ci;
using namespace ci::app;
using namespace std;

class cApp : public AppNative {

public:
    void setup();
    void update();
    void draw();
    void mouseDown( MouseEvent event );
    void mouseDrag( MouseEvent event );
    void keyDown( KeyEvent event );
    void resize();
    void loadSimulationData( int idump );

    int boxelx, boxely, boxelz;
    MayaCamUI camUi;
    Perlin mPln;

    Exporter mExp;
    vector<DataGroup> mDataGroup;

    gl::VboMesh bridge;
    unsigned int idump = 24;
};

void cApp::setup(){
    setWindowPos( 0, 0 );
    setWindowSize( 1080*3*0.5, 1920*0.5 );
    mExp.setup( 1080*3, 1920, 100, GL_RGB, mt::getRenderPath(), 0 );

    CameraPersp cam(1080*3, 1920, 54.4f, 1, 1000000 );
    cam.lookAt( Vec3f(0,0,600), Vec3f(0,0,0) );
    cam.setCenterOfInterestPoint( Vec3f(0,0,0) );
    //cam.setPerspective( 54.4f, getWindowAspectRatio(), 1, 100000 );      // 35mm
    camUi.setCurrentCam( cam );

    mPln.setSeed(123);
    mPln.setOctaves(4);

    boxelx = boxely = boxelz = 400;

#ifdef RENDER
    mExp.startRender();
#endif
}

void cApp::update(){

    for( auto dg : mDataGroup )
        dg.clear();
    mDataGroup.clear();

    idump++;
    loadSimulationData( idump );
}

void cApp::loadSimulationData( int idump){

    string fileName = "sim/Heracles/simu_mach4_split/rho/rho_0" + to_string(idump) + ".bin";

    fs::path assetPath = mt::getAssetPath();
    string path = ( assetPath / fileName ).string();
    cout << "loading binary file : " << path << endl;
    std::ifstream is( path, std::ios::binary );
    if(is){
        cout::b("load OK binary file");
    }else{
        cout::r("load ERROR bin file");
        quit();
    }

    // get length of file:
    is.seekg ( 0, is.end );
    int fileSize = is.tellg();
    cout << "length : " << fileSize << " byte" << endl;
    is.seekg ( 0, is.beg );
}
```

6. rendering as sequence of PNG file

