

nbody.cc

```

1  #include<iostream>
2  #include<fstream>
3  #include<unistd.h>
4  #include<sys/stat.h>
5  #include<particle_simulator.hpp>
6  #ifdef ENABLE_PHANTOM_GRAPE_X86
7  #include <gp5util.h>
8  #endif
9  #include "user-defined.hpp"
10
11 void makeColdUniformSphere(const PS::F64 mass_glb,
12                            const PS::S64 n_glb,
13                            const PS::S64 n_loc,
14                            PS::F64 *amp mass,
15                            PS::F64vec *amp pos,
16                            PS::F64vec *amp vel,
17                            const PS::F64 eng = -0.25,
18                            const PS::S32 seed = 0) {
19
20     assert(eng < 0.0);
21     {
22         PS::MTS mt;
23         mt.init_genrand(0);
24         for(PS::S32 i = 0; i < n_loc; i++){
25             mass[i] = mass_glb / n_glb;
26             const PS::F64 radius = 3.0;
27             do {
28                 pos[i][0] = (2. * mt.genrand_res53() - 1.) * radius;
29                 pos[i][1] = (2. * mt.genrand_res53() - 1.) * radius;
30                 pos[i][2] = (2. * mt.genrand_res53() - 1.) * radius;
31             }while(pos[i] * pos[i] >= radius * radius);
32             vel[i][0] = 0.0;
33             vel[i][1] = 0.0;
34             vel[i][2] = 0.0;
35         }
36     }
37
38     PS::F64vec cm_pos = 0.0;
39     PS::F64vec cm_vel = 0.0;
40     PS::F64 cm_mass = 0.0;
41     for(PS::S32 i = 0; i < n_loc; i++){
42         cm_pos += mass[i] * pos[i];
43         cm_vel += mass[i] * vel[i];
44         cm_mass += mass[i];
45     }
46     cm_pos /= cm_mass;
47     cm_vel /= cm_mass;
48     for(PS::S32 i = 0; i < n_loc; i++){
49         pos[i] -= cm_pos;
50         vel[i] -= cm_vel;
51     }
52 }
53
54 template<class Tpsys>
55 void setParticlesColdUniformSphere(Tpsys & psys,
56                                    const PS::S32 n_glb,
57                                    PS::S32 & n_loc) {
58
59     n_loc = n_glb;
60     psys.setNumberOfParticleLocal(n_loc);
61
62     PS::F64 * mass = new PS::F64[n_loc];
63     PS::F64vec * pos = new PS::F64vec[n_loc];
64     PS::F64vec * vel = new PS::F64vec[n_loc];
65     const PS::F64 m_tot = 1.0;
66     const PS::F64 eng = -0.25;
67     makeColdUniformSphere(m_tot, n_glb, n_loc, mass, pos, vel, eng);

```

nbody.cc

```

68     for(PS::S32 i = 0; i < n_loc; i++){
69         psys[i].mass = mass[i];
70         psys[i].pos = pos[i];
71         psys[i].vel = vel[i];
72         psys[i].id = i;
73     }
74     delete [] mass;
75     delete [] pos;
76     delete [] vel;
77 }
78
79 template<class Tpsys>
80 void kick(Tpsys & system,
81           const PS::F64 dt) {
82     PS::S32 n = system.getNumberOfParticleLocal();
83     for(PS::S32 i = 0; i < n; i++) {
84         system[i].vel += system[i].acc * dt;
85     }
86 }
87
88 template<class Tpsys>
89 void drift(Tpsys & system,
90            const PS::F64 dt) {
91     PS::S32 n = system.getNumberOfParticleLocal();
92     for(PS::S32 i = 0; i < n; i++) {
93         system[i].pos += system[i].vel * dt;
94     }
95 }
96
97 template<class Tpsys>
98 void calcEnergy(const Tpsys & system,
99                PS::F64 & etot,
100                 PS::F64 & ekin,
101                 PS::F64 & epot,
102                 const bool clear=true){
103     if(clear){
104         etot = ekin = epot = 0.0;
105     }
106     PS::F64 etot_loc = 0.0;
107     PS::F64 ekin_loc = 0.0;
108     PS::F64 epot_loc = 0.0;
109     const PS::S32 nbody = system.getNumberOfParticleLocal();
110     for(PS::S32 i = 0; i < nbody; i++){
111         ekin_loc += system[i].mass * system[i].vel * system[i].vel;
112         epot_loc += system[i].mass * (system[i].pot + system[i].mass
113 / FPGrav::eps);
114     }
115     ekin_loc *= 0.5;
116     epot_loc *= 0.5;
117     etot_loc = ekin_loc + epot_loc;
118     #ifdef PARTICLE_SIMULATOR_MPI_PARALLEL
119     etot = PS::Comm::getSum(etot_loc);
120     epot = PS::Comm::getSum(epot_loc);
121     ekin = PS::Comm::getSum(ekin_loc);
122 #else
123     etot = etot_loc;
124     epot = epot_loc;
125     ekin = ekin_loc;
126 #endif
127 }
128
129 void printHelp() {
130     std::cerr<<"o: dir name of output (default: ./result)"<<std::endl;
131     std::cerr<<"t: theta (default: 0.5)"<<std::endl;
132     std::cerr<<"T: time_end (default: 10.0)"<<std::endl;
133     std::cerr<<"s: time_step (default: 1.0 / 128.0)"<<std::endl;

```

nbody.cc

```
133     std::cerr<<"d: dt_diag (default: 1.0 / 8.0)"<<std::endl;
134     std::cerr<<"D: dt_snap (default: 1.0)"<<std::endl;
135     std::cerr<<"l: n_leaf_limit (default: 8)"<<std::endl;
136     std::cerr<<"n: n_group_limit (default: 64)"<<std::endl;
137     std::cerr<<"N: n_tot (default: 1024)"<<std::endl;
138     std::cerr<<"h: help"<<std::endl;
139 }
140
141 void makeOutputDirectory(char * dir_name) {
142     struct stat st;
143     if(stat(dir_name, &st) != 0) {
144         PS::S32 ret_loc = 0;
145         PS::S32 ret = 0;
146         if(PS::Comm::getRank() == 0)
147             ret_loc = mkdir(dir_name, 0777);
148         PS::Comm::broadcast(&ret_loc, ret);
149         if(ret == 0) {
150             if(PS::Comm::getRank() == 0)
151                 fprintf(stderr, "Directory \"%s\" is successfully ma
152 de.\n", dir_name);
153             } else {
154                 fprintf(stderr, "Directory %s fails to be made.\n", dir_
155 name);
156                 PS::Abort();
157             }
158         }
159     }
160
161 int main(int argc, char *argv[]) {
162     std::cout<<std::setprecision(15);
163     std::cerr<<std::setprecision(15);
164
165     PS::Initialize(argc, argv);
166     PS::F32 theta = 0.5;
167     PS::S32 n_leaf_limit = 8;
168     PS::S32 n_group_limit = 64;
169     PS::F32 time_end = 10.0;
170     PS::F32 dt = 1.0 / 128.0;
171     PS::F32 dt_diag = 1.0 / 8.0;
172     PS::F32 dt_snap = 1.0;
173     char dir_name[1024];
174     PS::S64 n_tot = 1024;
175     PS::S32 c;
176     sprintf(dir_name, "./result");
177     opterr = 0;
178     while((c=getopt(argc,argv,"i:o:d:D:t:T:l:n:N:hs:")) != -1){
179         switch(c){
180             case 'o':
181                 sprintf(dir_name,optarg);
182                 break;
183             case 't':
184                 theta = atof(optarg);
185                 std::cerr << "theta = " << theta << std::endl;
186                 break;
187             case 'T':
188                 time_end = atof(optarg);
189                 std::cerr << "time_end = " << time_end << std::endl;
190                 break;
191             case 's':
192                 dt = atof(optarg);
193                 std::cerr << "time_step = " << dt << std::endl;
194                 break;
195             case 'd':
196                 dt_diag = atof(optarg);
197                 std::cerr << "dt_diag = " << dt_diag << std::endl;
198                 break;
199             case 'D':
```

nbody.cc

```
200         dt_snap = atof(optarg);
201         std::cerr << "dt_snap = " << dt_snap << std::endl;
202         break;
203     case 'l':
204         n_leaf_limit = atoi(optarg);
205         std::cerr << "n_leaf_limit = " << n_leaf_limit << std::e
206 ndl;
207         break;
208     case 'n':
209         n_group_limit = atoi(optarg);
210         std::cerr << "n_group_limit = " << n_group_limit << std:
211 :endl;
212         break;
213     case 'N':
214         n_tot = atoi(optarg);
215         std::cerr << "n_tot = " << n_tot << std::endl;
216         break;
217     case 'h':
218         if(PS::Comm::getRank() == 0) {
219             printHelp();
220         }
221         PS::Finalize();
222         return 0;
223     default:
224         if(PS::Comm::getRank() == 0) {
225             std::cerr<<"No such option! Available options are he
226 re."<<std::endl;
227             printHelp();
228         }
229         PS::Abort();
230     }
231 }
232
233 makeOutputDirectory(dir_name);
234
235     std::ofstream fout_eng;
236     char sout_de[1024];
237     sprintf(sout_de, "%s/t-de.dat", dir_name);
238     std::cerr << sout_de << std::endl;
239     fout_eng.open(sout_de);
240
241     if(PS::Comm::getRank() == 0) {
242         fprintf(stderr, "Number of processes: %d\n", PS::Comm::getNu
243 mberOfProc());
244         fprintf(stderr, "Number of threads per process: %d\n", PS::C
245 omm::getNumberOfThread());
246     }
247
248     PS::ParticleSystem<FPGrav> system_grav;
249     system_grav.initialize();
250     PS::S32 n_loc = 0;
251     PS::F32 time_sys = 0.0;
252     if(PS::Comm::getRank() == 0) {
253         setParticlesColdUniformSphere(system_grav, n_tot, n_loc);
254     } else {
255         system_grav.setNumberOfParticleLocal(n_loc);
256     }
257
258     const PS::F32 coef_ema = 0.3;
259     PS::DomainInfo dinfo;
260     dinfo.initialize(coef_ema);
261     dinfo.collectSampleParticle(system_grav);
262     dinfo.decomposeDomain();
263     system_grav.exchangeParticle(dinfo);
264     n_loc = system_grav.getNumberOfParticleLocal();
265
266 #ifdef ENABLE_PHANTOM_GRAPE_X86
```

nbody.cc

```
260     g5_open();
261     g5_set_eps_to_all(FPGrav::eps);
262 #endif
263
264     PS::TreeForForceLong<FPGrav, FPGrav, FPGrav>::Monopole tree_grav
265 ;
266     tree_grav.initialize(n_tot, theta, n_leaf_limit, n_group_limit);
267     tree_grav.calcForceAllAndWriteBack(CalcGravity<FPGrav>,
268                                       CalcGravity<PS::SPJMonopole>,
269                                       system_grav,
270                                       dinfo);
271     PS::F64 Epot0, Ekin0, Etot0, Epot1, Ekin1, Etot1;
272     calcEnergy(system_grav, Etot0, Ekin0, Epot0);
273     PS::F64 time_diag = 0.0;
274     PS::F64 time_snap = 0.0;
275     PS::S64 n_loop = 0;
276     PS::S32 id_snap = 0;
277     while(time_sys < time_end){
278         if( (time_sys >= time_snap) || ( (time_sys + dt) - time_snap
279 ) > (time_snap - time_sys) ){
280             char filename[256];
281             sprintf(filename, "%s/%04d.dat", dir_name, id_snap++);
282             FileHeader header;
283             header.time = time_sys;
284             header.n_body = system_grav.getNumberOfParticleGlobal();
285             system_grav.writeParticleAscii(filename, header);
286             time_snap += dt_snap;
287         }
288
289         calcEnergy(system_grav, Etot1, Ekin1, Epot1);
290
291         if(PS::Comm::getRank() == 0){
292             if( (time_sys >= time_diag) || ( (time_sys + dt) - time_
293 diag ) > (time_diag - time_sys) ){
294                 fout_eng << time_sys << " " << (Etot1 - Etot0) / E
295 tot0 << std::endl;
296                 fprintf(stderr, "time: %10.7f energy error: %e\n",
297                             time_sys, (Etot1 - Etot0) / Etot0);
298                 time_diag += dt_diag;
299             }
300         }
301
302         kick(system_grav, dt * 0.5);
303
304         time_sys += dt;
305         drift(system_grav, dt);
306
307         if(n_loop % 4 == 0){
308             dinfo.decomposeDomainAll(system_grav);
309         }
310
311         system_grav.exchangeParticle(dinfo);
312
313         tree_grav.calcForceAllAndWriteBack(CalcGravity<FPGrav>,
314                                           CalcGravity<PS::SPJMonopo
315 le>,
316                                           system_grav,
317                                           dinfo);
318
319         kick(system_grav, dt * 0.5);
320
321         n_loop++;
322     }
323 #ifdef ENABLE_PHANTOM_GRAPE_X86
324     g5_close();
325 #endif
```

nbody.cc

```
322 #endif
323
324     PS::Finalize();
325     return 0;
326 }
```