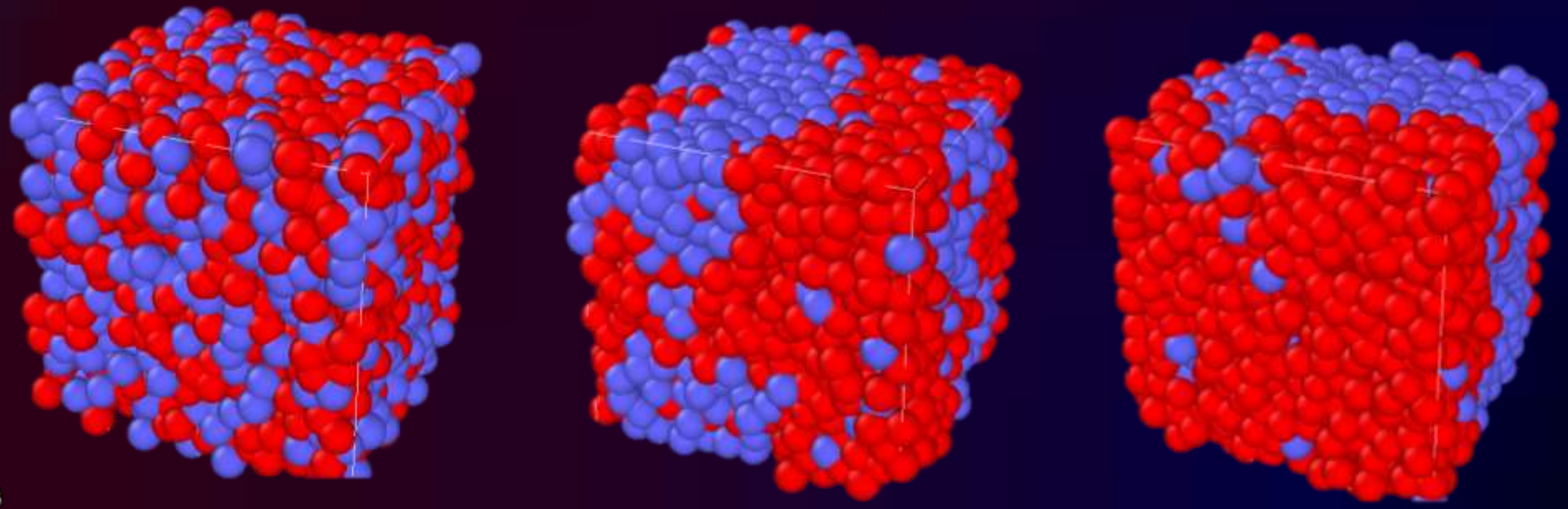
2020级C语言程序设计基础

终期汇报

吕佳潼12200404

A、B 两种原子体系的动力学模拟

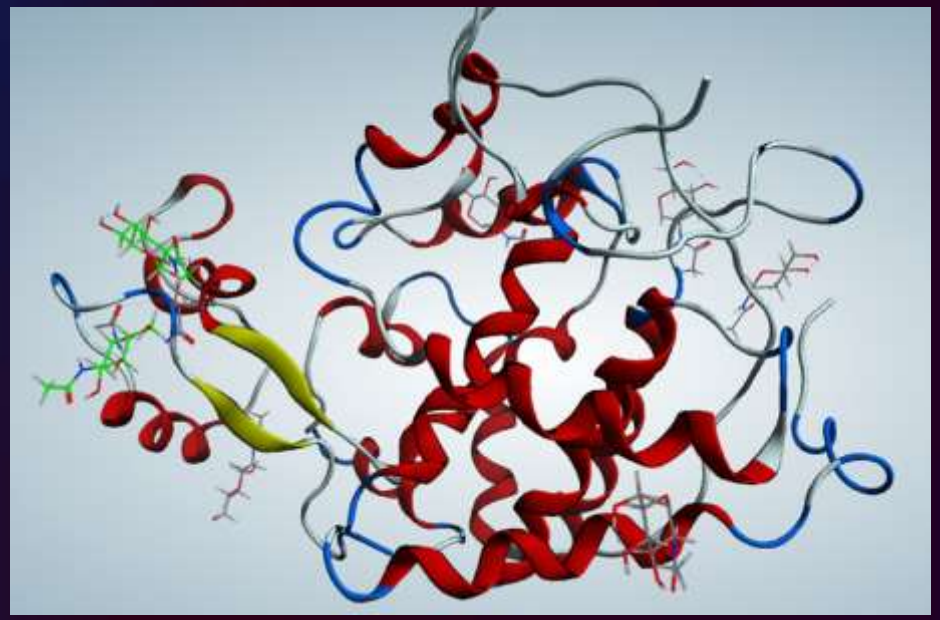


目录

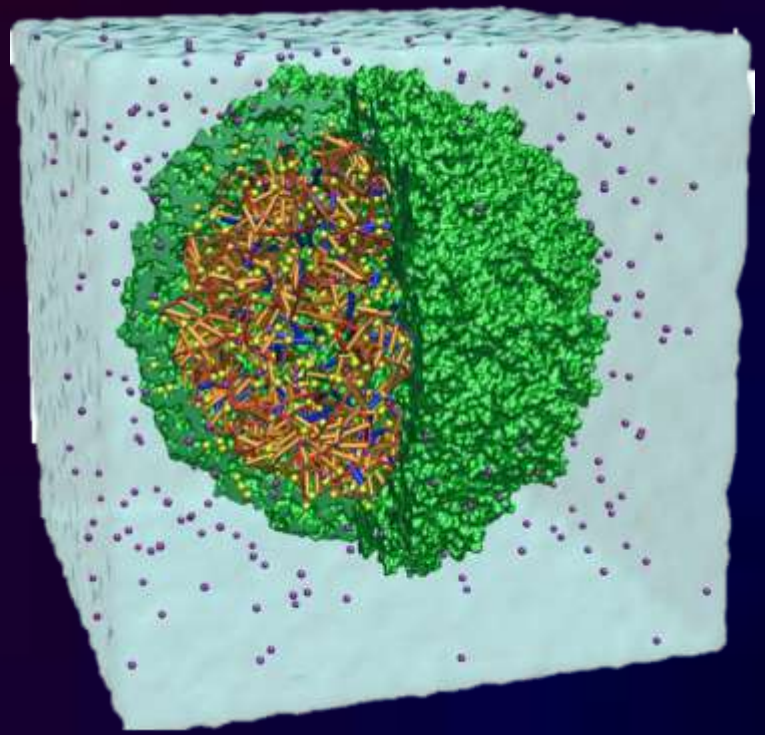
目录

项目背景

模拟蛋白质



模拟病毒



目录

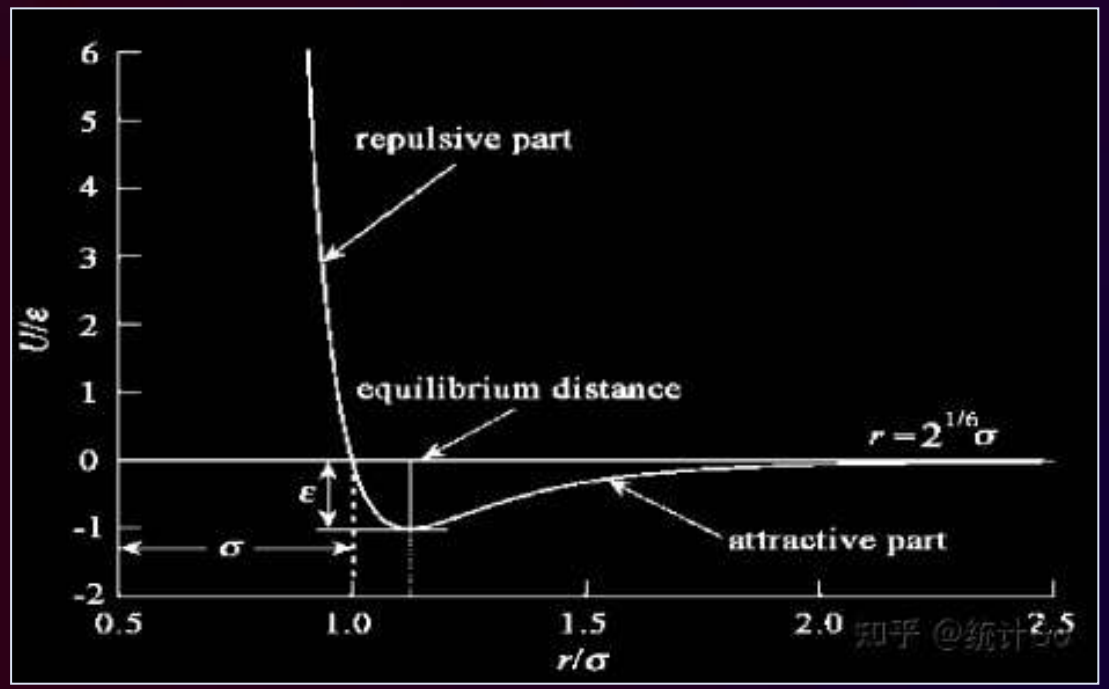
相关知识

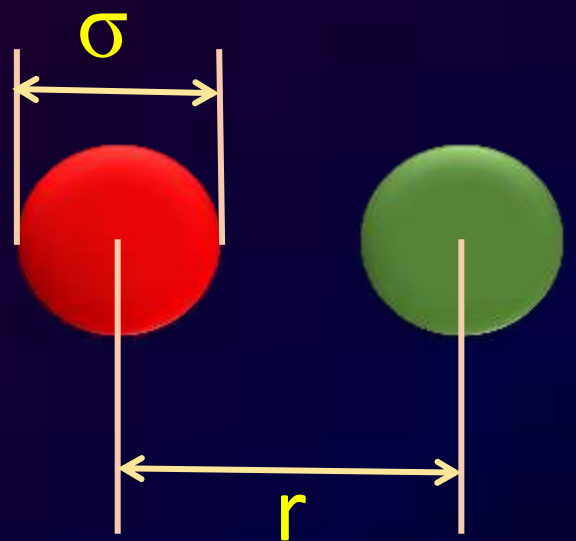
- A、B两种粒子间存在不同的相互作用
- 粒子间的相互作用取决于粒子的类型和之间的距离
- 单个粒子在Δt内受到的合力决定了它Δt内的运动方式
- 经过Δt后粒子的相对位置稍有改变，影响了它的受力和势能，而新的受力和势能会进一步影响它的运动
- 切割极短的Δt，分步计算每个粒子的运动，再通过循环Δt近似计算出一段时间内体系中粒子的运动轨迹，最后在Ovito中实现轨迹的可视化，形象模拟出粒子的动力学行为

相关知识

粒子间势能公式：

$$U = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$





相关知识

通过势能求出作用力：

$$F_i = -\nabla U_i \qquad U_i = \sum_j u_{ij}$$

牛顿第二定律：

$$F_i = m_i a_i$$

运动学方程：

$$r_i\,(\Delta t) = r_i(0) + v_i(0)\Delta t + \frac{1}{2} a_i(0)\Delta t^2$$

$$v_i\,(\Delta t) = v_i(0) + \frac{a_i(0) + a_i(\Delta t)}{2}\Delta t$$

相关知识

运动学方程推导为：

$$r_i(\Delta t) = r_i(0) + v_i(0)\Delta t + \frac{1}{2}a_i(0)\Delta t^2$$

$$r_i(\Delta t) = r_i(0) + \Delta t[v_i(0) + \frac{1}{2}a_i(0)\Delta t]$$

$$r_i(\Delta t) = r_i(0) + \Delta t[v_i(\Delta t/2)]$$

$$v_i(\Delta t) = v_i(0) + \frac{a_i(0) + a_i(\Delta t)}{2}\Delta t$$

$$v_i(\Delta t/2) = v_i(0) + a_i(0)\frac{\Delta t}{2}$$

&

$$v_i(\Delta t) = v_i(\Delta t/2) + a_i(\Delta t)\frac{\Delta t}{2}$$

相关知识

推进$\frac{1}{2}\Delta t$，计算速度和距离：

$$v_i(\Delta t/2) \leftarrow v_i(0) + \frac{1}{2}\frac{f_i(0)}{m_i}\Delta t$$

$$r_i\ (\Delta t) \leftarrow r_i(0) + v_i(\Delta t/2)\Delta t$$

通过距离计算相互作用和势能：

$$f_i(\Delta t) \leftarrow\ r_i\ (\Delta t)$$

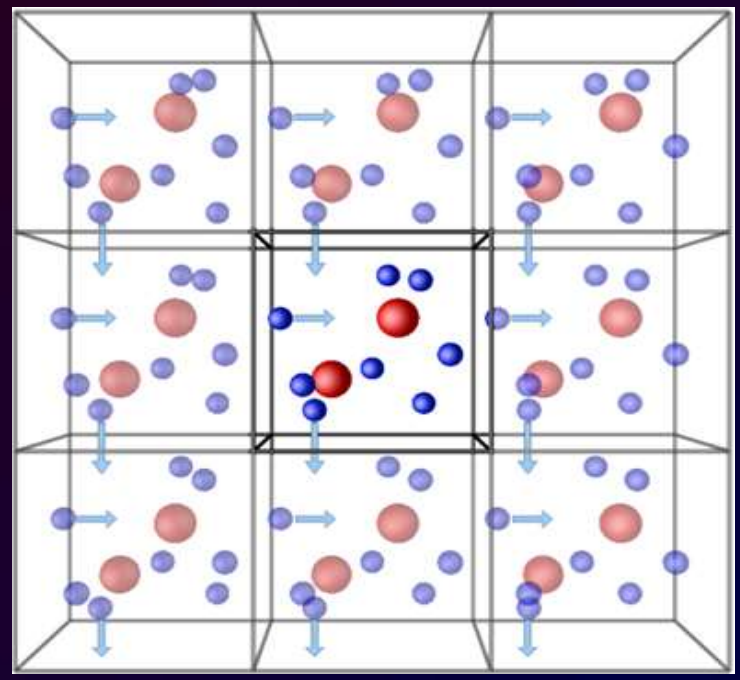
继续推进$\frac{1}{2}\Delta t$：

$$v_i(\Delta t) \leftarrow v_i(\Delta t/2) + \frac{1}{2}\frac{f_i(\Delta t)}{m_i}\Delta t$$

相关知识

特殊注意：

周期性边界条件（periodic boundary conditions, PBC）的作用是，在无限空间里复制有限大小的盒子，尽可能体现真实研究体系。



类比晶胞的无隙并置、无限循环，每一个盒子内的原子数保持不变（从右侧出去一个原子的同时从左侧进来同一个原子）

目录

设计思路

（一）用数组存储坐标和能量四个变量

```
10
11  struct float3   // structure to hold information
12      {
13      float3() : x(0.0), y(0.0), z(0.0){}
14      float3(float x0, float y0, float z0) : x(x0), y(y0), z(z0){}
15
16      float x;
17      float y;
18      float z;
19      };
20
21  struct float4 // structure to hold information
22      {
23      float4() : x(0.0), y(0.0), z(0.0), w(0.0){}
24      float4(float x0, float y0, float z0, float w0) : x(x0), y(y0), z(z0), w(w0){}
25
26      float x;
27      float y;
28      float z;
29      float w;
30      };
31
32  float pbc(float x, float box_len)  // implement periodic bondary condition
33      {
34      float box_half = box_len * 0.5;
35      if (x >  box_half) x -= box_len;
36      else if (x < -box_half) x += box_len;
37      return x;
38      }
39
```

设计思路

（二）随机生成坐标并筛选合理值

```
// Random number generator [0.0-1.0)
float R2S()
    {
    int ran = rand();
    float fran = (float)ran/(float)RAND_MAX;
    return fran;
    }
// initially generate the position and mass of particles
void init(unsigned int np, float4* r, float4* v, float3 box, float min_dis)
    {
    for (unsigned int i=0; i<np; i++)
        {
        bool find_pos = false;
        float4 ri;
        while(!find_pos)
            {
            ri.x = ( R2S() - 0.5 ) * box.x;
            ri.y = ( R2S() - 0.5 ) * box.y;
            ri.z = ( R2S() - 0.5 ) * box.z;
            find_pos = true;

            for(unsigned int j=0; j<i; j++)
                {
                float dx = pbc(ri.x - r[j].x, box.x);
                float dy = pbc(ri.y - r[j].y, box.y);
                float dz = pbc(ri.z - r[j].z, box.z);
                float r = sqrt(dx*dx + dy*dy + dz*dz);
                if(r<min_dis)                    // a minimum safe distance to avoid the overlap of LJ particles
                    {
                    find_pos = false;
                    break;
                    }
                }
            }
        if(R2S()>0.5) // randomly generate the type of particle, 1.0 represent type A and 2.0 represent type B
            ri.w =1.0;
        else
            ri.w =2.0;

        r[i] = ri;
        v[i].w = 1.0;
        }
```

分别用1和2代表A原子和B原子的能量，方便后面计算A和B的运动轨迹。（二者的和可以用来表示其间的作用力，由于不同原子间的相互作用不同）

设计思路

（三）分别定义分步计算速度、位移、力与能量的函数

```
83  // first step integration of velocity verlet algorit
84  void first_integration(unsigned int np, float dt, fl
85      {
86      for (unsigned int i=0; i<np; i++)
87          {
88          float4 ri = r[i];
89          float mass = v[i].w;
90
91          v[i].x += 0.5 * dt * f[i].x / mass;
92          v[i].y += 0.5 * dt * f[i].y / mass;
93          v[i].z += 0.5 * dt * f[i].z / mass;
94
95          ri.x += dt * v[i].x;
96          ri.y += dt * v[i].y;
97          ri.z += dt * v[i].z;
98
99          r[i].x = pbc(ri.x, box.x);
100         r[i].y = pbc(ri.y, box.y);
101         r[i].z = pbc(ri.z, box.z);
102         }
103     }
```

```
104 // non-bonded force calculation
105 void force_calculation(unsigned int np, float3 box, float3 epsilon, float3 sigma, float4* r, float4* f, float rcut)
106     {
107     for(unsigned int i=0; i<np; i++)
108         {
109         float4 force = float4(0.0, 0.0, 0.0, 0.0);
110         for(unsigned int j=0; j<np; j++)
111             {
112             /* particles have no interactions with themselves */
113             if (i==j) continue;
114
115             /* calculated the shortest distance between particle i and j */
116
117             float dx = pbc(r[i].x - r[j].x, box.x);
118             float dy = pbc(r[i].y - r[j].y, box.y);
119             float dz = pbc(r[i].z - r[j].z, box.z);
120             float type = r[i].w + r[j].w;
121
122             float r = sqrt(dx*dx + dy*dy + dz*dz);
123
124             /* compute force and energy if within cutoff */
125             if (r < rcut)
126                 {
127                 float epsilonij, sigmaij;
128                 if(type==2.0)               // i=1.0, j=1.0
129                     {
130                     epsilonij = epsilon.x;
131                     sigmaij = sigma.x;
132                     }
133                 else if(type==3.0)          // i=1.0, j=2.0; or i=2.0, j=1.0
134                     {
135                     epsilonij = epsilon.y;
136                     sigmaij = sigma.y;
137                     }
138                 else if(type==4.0)          // i=2.0, j=2.0
139                     {
140                     epsilonij = epsilon.z;
141                     sigmaij = sigma.z;
142                     }
143
144                 float ffac = -4.0*epsilonij*(-12.0*pow(sigmaij/r,12.0)/r + 6.0*pow(sigmaij/r,6.0)/r);  // force between particle i and j
145                 float epot = 0.5*4.0*epsilonij*(pow(sigmaij/r,12.0) - pow(sigmaij/r,6.0));             // potential between particle i and j
146
147                 force.x += ffac*dx/r;
148                 force.y += ffac*dy/r;
149                 force.z += ffac*dz/r;
150                 force.w += epot;
151                 }
152             }
153         f[i] = force;
154         }
155     }
```

设计思路

以及输出每改变Δt原子位置坐标的改变

```
156  // second step integration of velocity verlet algorithm
157  void second_integration(unsigned int np, float dt, float4* v, float4* f)
158      {
159      for (unsigned int i=0; i<np; i++)
160          {
161          float mass = v[i].w;
162          v[i].x += 0.5 * dt * f[i].x / mass;
163          v[i].y += 0.5 * dt * f[i].y / mass;
164          v[i].z += 0.5 * dt * f[i].z / mass;
165          }
166      }
167  // system information collection for temperature, kinetic energy, potential and total energy
168  void compute_infc(unsigned int np, float4* v, float4* f, float* infc)
169      {
170      float ekin=0.0;
171      float potential;
172      for (unsigned int i = 0; i < np; i++)
173          {
174          float4 vi = v[i];
175          float mass = vi.w;
176          ekin += 0.5*mass*(vi.x*vi.x + vi.y*vi.y + vi.z*vi.z);
177          potential += f[i].w;
178          }
179      unsigned int nfreedom = 3 * np - 3;
180      float temp = 2.0*ekin/float(nfreedom);
181      float energy = ekin + potential;
182
183      infc[0] = temp;
184      infc[1] = potential;
185      infc[2] = energy;
186      }
187  // output system information and frame in XYZ formation which can be read by VMD
188  void output(FILE *traj, unsigned int step, float* infc, float4* r, unsigned int np)
189      {
190      float temp = infc[0];
191      float potential = infc[1];
192      float energy = infc[2];
193
194      fprintf(traj,"%d\n  step=%d  temp=%20.8f  pot=%20.8f  ener=%20.8f\n", np, step, temp, potential, energy);
195      for (unsigned int i=0; i<np; i++)
196          {
197          float4 ri = r[i];
198          if (ri.w == 1.0)
199              fprintf(traj, "A  %20.8f %20.8f %20.8f\n", ri.x, ri.y, ri.z);
200          else if (ri.w == 2.0)
201              fprintf(traj, "B  %20.8f %20.8f %20.8f\n", ri.x, ri.y, ri.z);
202          }
203
204      }
205
```

设计思路

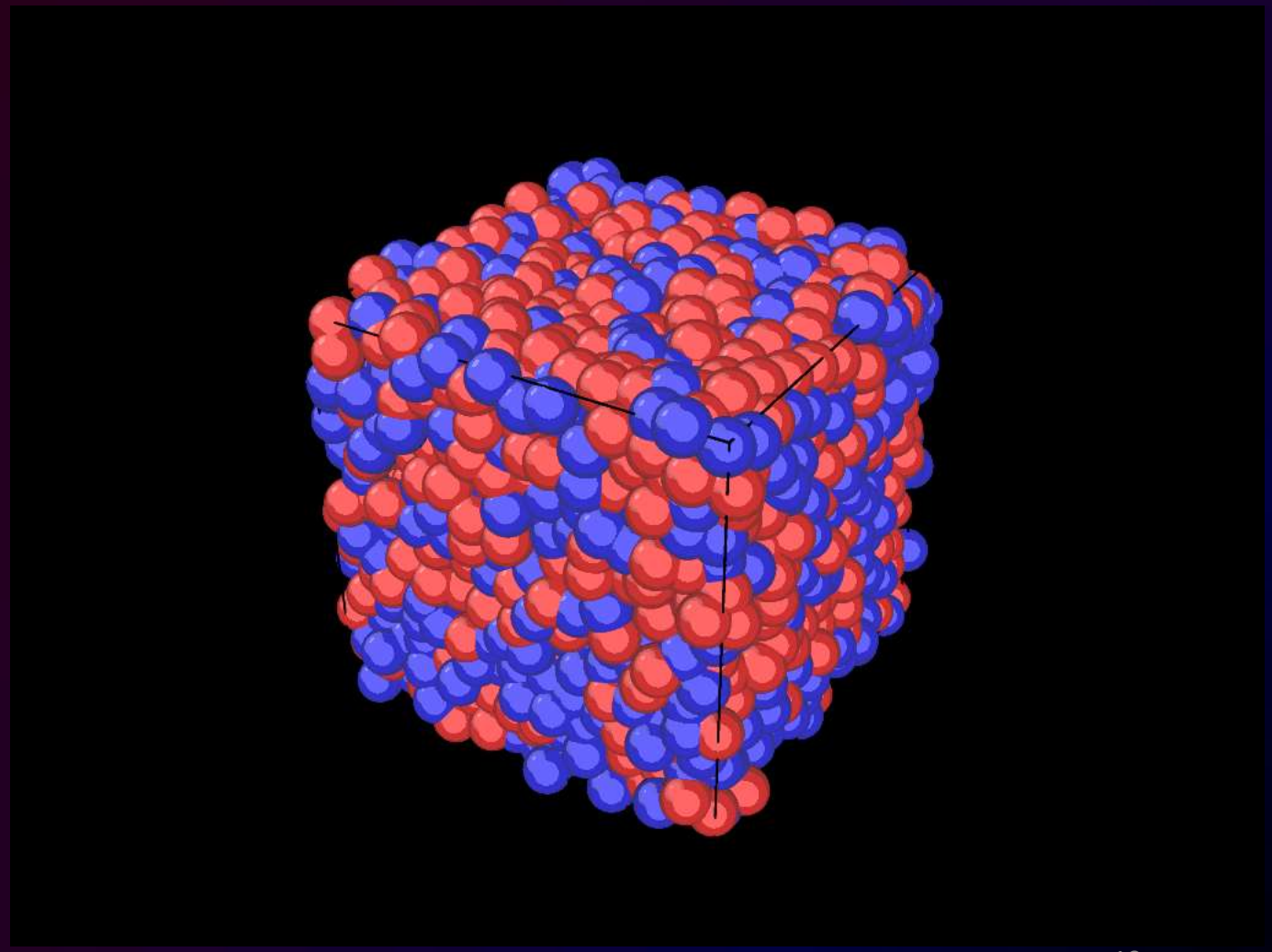
（四）运行主函数

```
// main function
int main(int argc, char **argv)
    {
    //running parameters
    unsigned int np = 2700;                 // the number of particles
    unsigned int nsteps = 500;              // the number of time steps
    float dt = 0.001;                       // integration time step
    float rcut = 3.0;                       // the cutoff radius of interactions
//  float temperature = 1.0;                // target temperature
    unsigned int nprint = 100;              //  period for data output

    timeval start;                          // start time
    timeval end;                            // end time

    float3 box =float3(15.0, 15.0, 15.0);     // box size in x, y, and z directions
    float3 epsilon = float3(1.0, 0.5, 1.0);   // epsilon.x for type 1.0 and 1.0; epsilon.y for type 1.0 and 2.0; epsilon.z for type 1.0 and 2.0
    float3 sigma = float3(1.0, 1.0, 1.0);     // sigma.x for type 1.0 and 1.0; sigma.y for type 1.0 and 2.0; sigma.z for type 1.0 and 2.0
    float min_dis = sigma.x*0.8;              // the minimum distance between particles for system generation

    //memory allocation
    float4* r = (float4 *)malloc(np*sizeof(float4));  // rx, ry, rz, type(0, 1, 2 ...)
    float4* v = (float4 *)malloc(np*sizeof(float4));  // vx, vy, vz, mass
    float4* f = (float4 *)malloc(np*sizeof(float4));  // fx, fy, fz, potential
    float* info = (float *)malloc(10*sizeof(float));  // temperature, potential, energy ...

    FILE *traj=fopen("traj.xyz","w");                 // trajectory file in XYZ format that can be open by VMD

        /* generate system information */

    printf("Starting simulation with %d atoms for %d steps.\n", np, nsteps);
    printf("Generating system.\n", np, nsteps);
    init(np, r, v, box, min_dis);

    gettimeofday(&start,NULL);                        //get start time
    /* main MD loop */
    printf("Running simulation.\n", np, nsteps);
    for(unsigned int step =0; step <= nsteps; step++) //running simulation loop
        {
        /* first integration for velverlet */
        first_integration(np, dt, box, r, v, f);

        /* force calculation */
        force_calculation(np, box, epsilon, sigma, r, f, rcut);

        /* compute temperature and potential */
        compute_info(np, v, f, info);

        /* second integration for velverlet */
        second_integration(np, dt, v, f);

        /* write output frames and system information, if requested */
        if ((step % nprint) == 0)
            {
            output(traj, step, info, r, np);
            printf("time step %d \n", step);
            }
        }
```

项目实现

运行结果及可视化：

A	-3.92580080	7.05951166	6.03312111
A	-3.50001359	0.59640527	-1.87189531
B	0.18803090	2.51585674	0.47409654
A	-0.93543601	6.47752571	6.46214676
B	-3.23559880	3.57801485	2.09968233
A	2.81792068	-5.01038742	-0.89843225
A	4.93801641	-2.54494286	-4.06547737
A	-2.24459720	2.80004835	6.84702349
A	2.35956073	5.38014460	-0.90660143
A	-1.52345002	4.72150326	2.76327801
A	-0.26264012	-4.26262569	6.75378513
B	-5.28509998	5.71593237	2.11620927
B	1.79394722	-3.28410888	4.29003143
B	-0.79449624	-4.10840034	-4.68700314
A	0.84665626	-1.25248075	-4.95589352
A	-5.95243216	-5.60887003	-0.06833911
B	7.27127457	6.52505970	2.76667547
B	3.74656320	-1.97004676	-3.08759451
A	1.26732767	-3.83380890	-5.21415329
A	-5.61787653	4.40205574	-5.03847122
A	-6.38205290	6.75156069	-6.71206093
A	-4.85684013	-3.89906454	4.46697044
A	2.34845448	7.01107693	2.09187531
B	-6.32651806	-6.45140409	-4.43017387
	4.79515934	1.09977901	3.83371258
	-5.13289309	7.49990368	-4.43507099
	-5.61797285	7.46698475	-6.68913651
	-6.41506815	-7.43757582	6.34603691



Danke Sehr!