GRAVITY PROJECT
N-KÖRPERPROBLEM

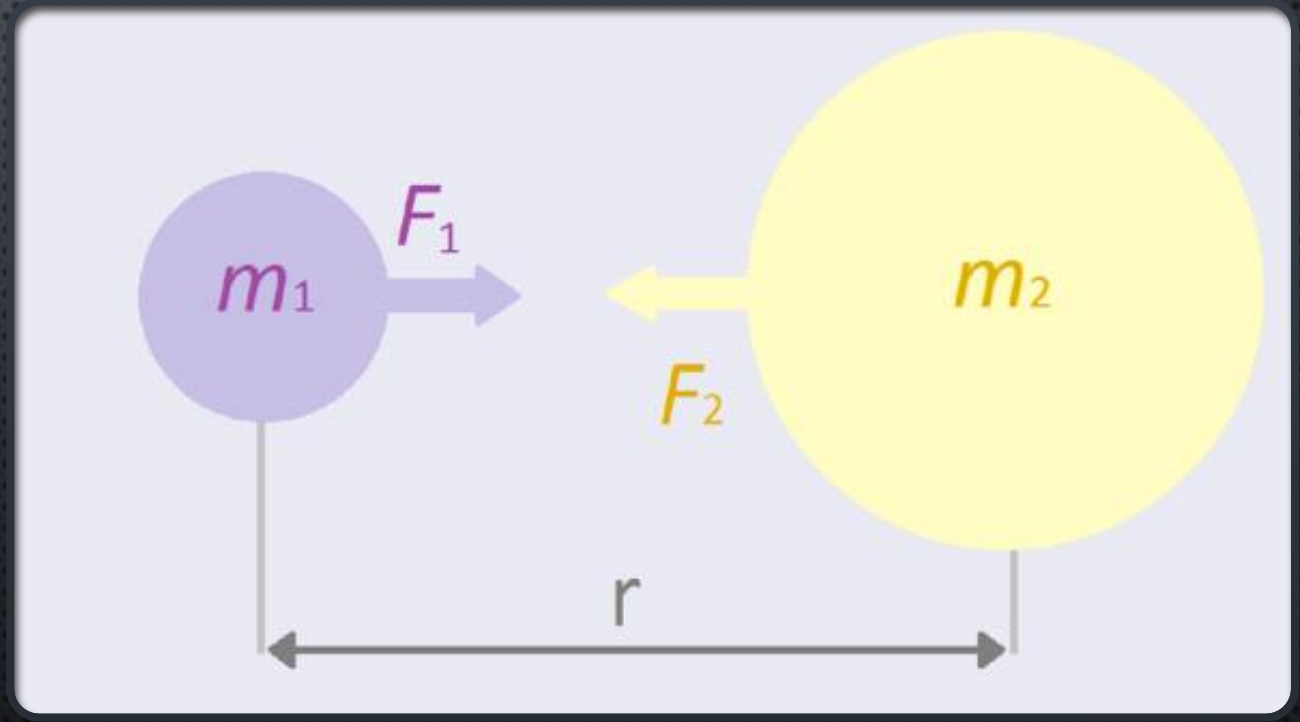
- DANIEL K. BIAO ADZA & ROSWALDAS SNARAS
- BERGISCHE UNIVERSITÄT WUPPERTAL

- 08.07.2019

Inhaltsverzeichnis

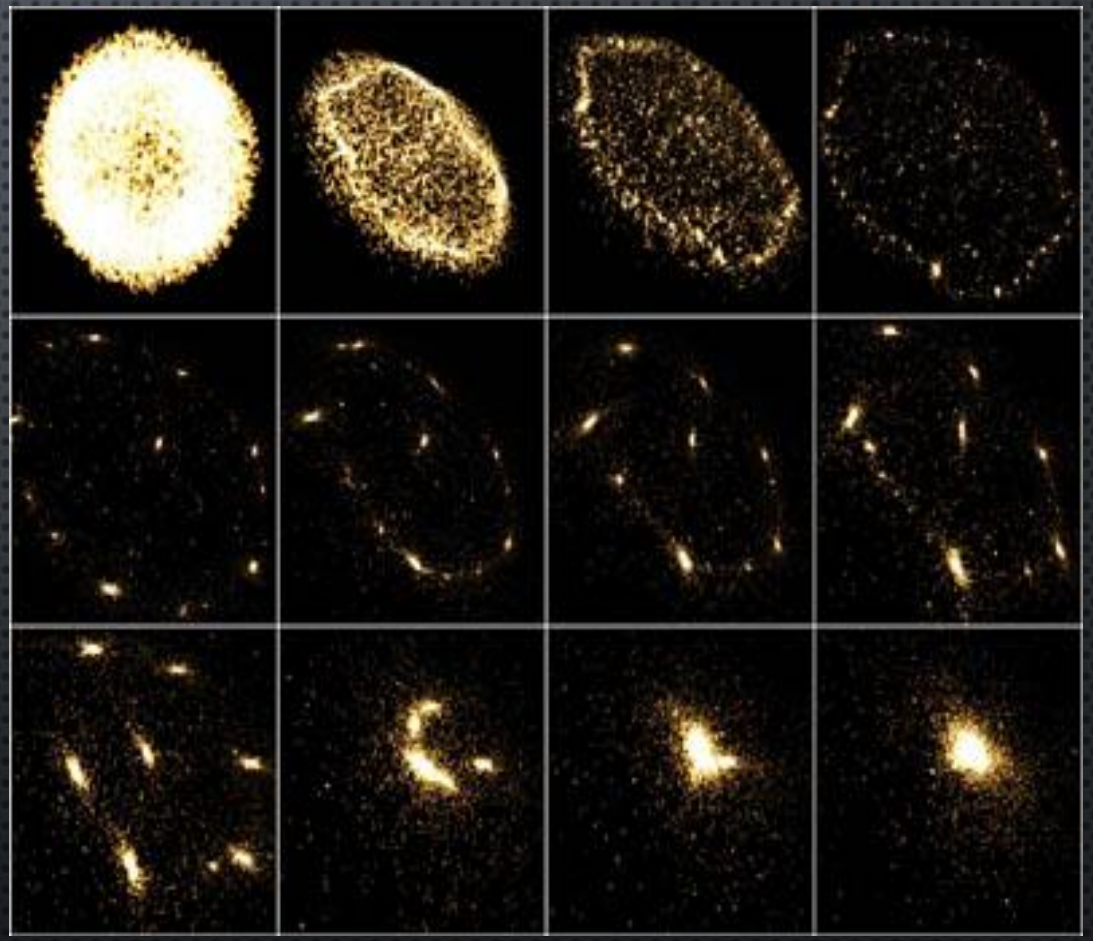
GRAVITATIONSGESETZ

- 1687 AUFGESTELLT VON ISAAC NEWTON
- $F = G \frac{m_1 m_2}{r^2} \frac{\vec{r}}{r}$



N-Körperproblem

- Ab mehr als 2 Körpern keine analytische Lösung mehr möglich
- In speziellen Fällen möglich
- N-Körperproblem ist die Verallgemeinerung des Dreikörperproblems



Algorithmus

1. Berechne Abstände
2. Berechne Kraft
3. Berechne Gesamtkraft
4. Berechne Neue Position
5. Fang von vorne an

Alles jeweils in x- , y- und z-Richtung

MASSENOBJEKTE

- ARRAY MIT MASSEN MIT "RANDOM" WERTEN WIRD ERZEUGT.

```
mass masses[n+1]; //Array mit n Massen

                    for(int i=1; i< n+1; i++){
                            masses[i].beginx = rand(-1000, 1000);
                            masses[i].beginy = rand(-1000, 1000);
                            masses[i].beginz =rand(-1000, 1000);
                            masses[i].vx = rand(-1000, 1000);
                            masses[i].vy = rand(-1000, 1000);
                            masses[i].vz =rand(-1000, 1000);
                            masses[i].m =  rand(1e10, 1e12);
                    }
```

ABSTÄNDE

- MULTIDIMENSIONALE ARRAYS

b1 – b1 = 0	b1 – b2	b1 – b3	…	b1 – bn
b1 – b1	b1 – b2 = 0	b1 – b3	…	b1 – bn
b3 – b1	b3 – b2	b3 – b3 = 0	…	b3 – bn
…	…	…	…	…
bn – b1	bn – b2	bn – b3	…	bn – bn = 0

ABSTÄNDE

```
132   //Abstand
133   for(int j = 1; j<n+1; j++){
134           for(int i = 1; i < n+1; i++){
135                   xAbs[i] = masses[i].beginx - masses[j].beginx;
136                   yAbs[i] = masses[i].beginy - masses[j].beginy;
137                   zAbs[i] = masses[i].beginz - masses[j].beginz;
138                   UpxAbs[i][j] = xAbs[i];
139                   UpyAbs[i][j] = yAbs[i];
140                   UpzAbs[i][j] = zAbs[i];
141           }
142   }
```

KRAFT

- IN X-RICHTUNG $F_x = G \frac{m_1 m_2}{r^2} \frac{r_x}{r}$

```
146   //Kraft Komponenten
147   for(int j = 1; j<n+1; j++){
148           for(int i = 1; i < n+1; i++){
149                   Fx[i] =( masses[i].m * masses[j].m * G * UpxAbs[i][j] ) /
150                                   (sqrt(UpxAbs[i][j] * UpxAbs[i][j] + UpyAbs[i][j] * UpyAbs[i][j] + UpzAbs[i][j] * UpzAbs[i][j]) *
151                                    sqrt(UpxAbs[i][j] * UpxAbs[i][j] + UpyAbs[i][j] * UpyAbs[i][j] + UpzAbs[i][j] * UpzAbs[i][j]) *
152                                    sqrt(UpxAbs[i][j] * UpxAbs[i][j] + UpyAbs[i][j] * UpyAbs[i][j] + UpzAbs[i][j] * UpzAbs[i][j]));
```

GESAMTKRAFT

- DIE KRÄFTE WERDEN ADDIERT
- IF-SCHLEIFE NÖTIG, DA AUF DEN DIAGONALEINTRÄGEN 0 STEHT
  - ERGEBNIS SONST NAN

```
172   //Gesamtkraft
173   for(int j = 1; j<n+1; j++){
174           GesFx[j] = 0;
175           GesFy[j] = 0;
176           GesFz[j] = 0;
177           r = 0;
178           u = 0;
179           l = 0;
180           for(int i = 1; i<n+1; i++){
181                   if(i != j){
182                                   GesFx[j] = u + UpFx[i][j];
183                                   u = GesFx[j];
184                                   GesFy[j] = r + UpFy[i][j];
185                                   r=GesFy[j];
186                                   GesFz[j] = l + UpFz[i][j];
187                                   l=GesFz[j];
188                   }
189           }
190   }
```

NEUE POSITIONEN

- $x_{neu} = x_{alt} + (v_{alt}\ \frac{F}{a}t)t$

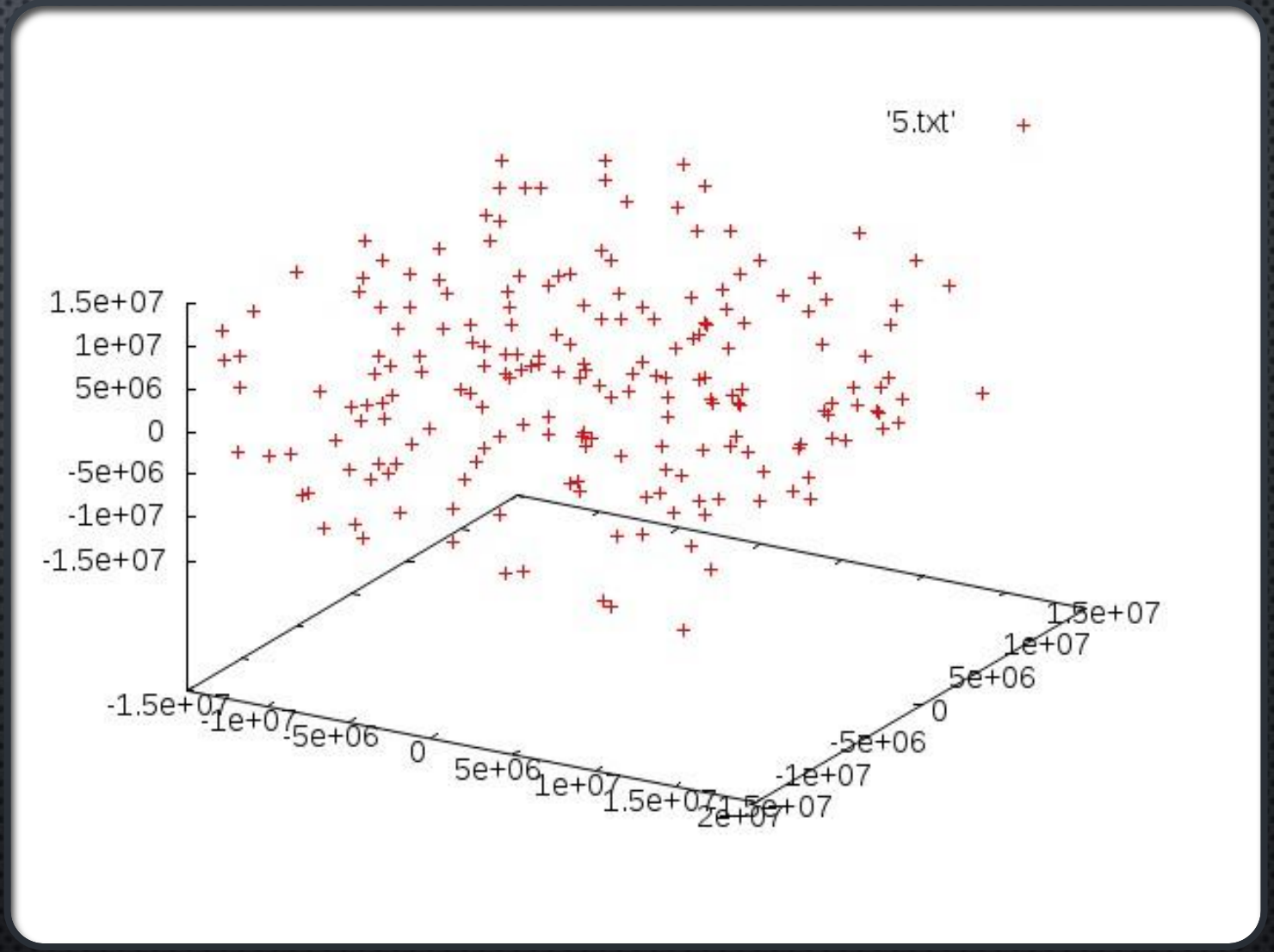
```
195  //new positions
196          for(int i = 1; i<n+1; i++){
197                  posx[i] = masses[i].beginx + (masses[i].vx + (GesFx[i] / masses[i].m)); // Zeitabstand ist Eins
198                  posy[i] = masses[i].beginy + (masses[i].vy + (GesFy[i] / masses[i].m));
199                  posz[i] = masses[i].beginz + (masses[i].vz + (GesFz[i] / masses[i].m));
200                  cout << "(" << posx[i] << "," << posy[i] << "," << posz[i] << ")" << endl;
201                  fprintf(fs,"%e %e %e\n", posx[i], posy[i], posz[i]);
202                  masses[i].beginx = posx[i];
203                  masses[i].beginy = posy[i];
204                  masses[i].beginz = posz[i];
205                  nvx[i] = masses[i].vx;
206                  nvy[i] = masses[i].vy;
207                  nvz[i] = masses[i].vz;
208                  masses[i].vx = nvx[i] + (GesFx[i] / masses[i].m);
209                  masses[i].vy = nvy[i] + (GesFy[i] / masses[i].m);
210                  masses[i].vz = nvz[i] + (GesFz[i] / masses[i].m);
```

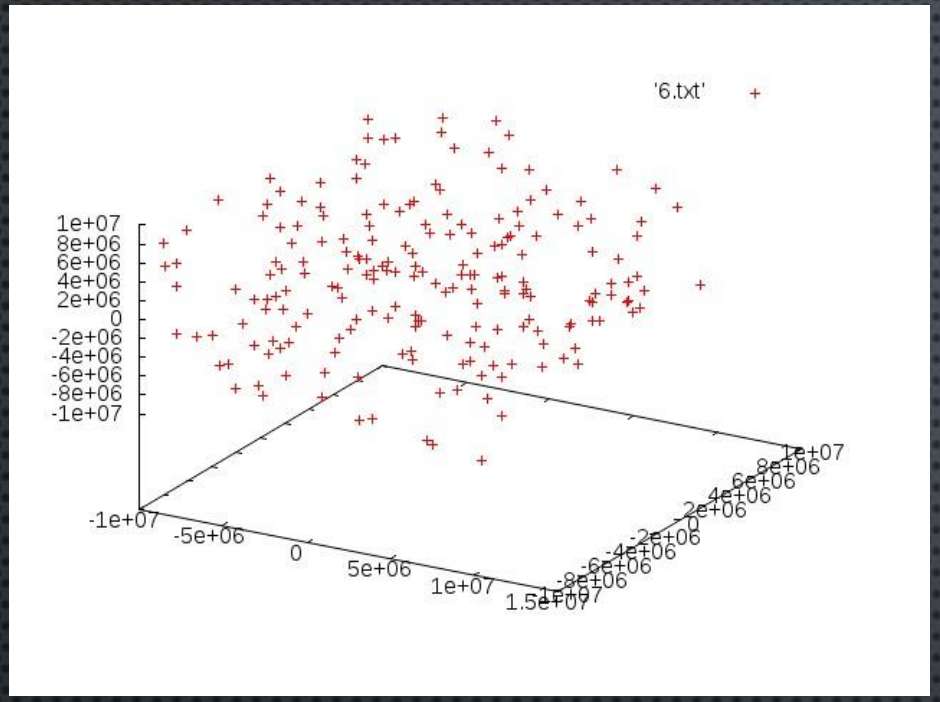
Was kann Gravity Project ?

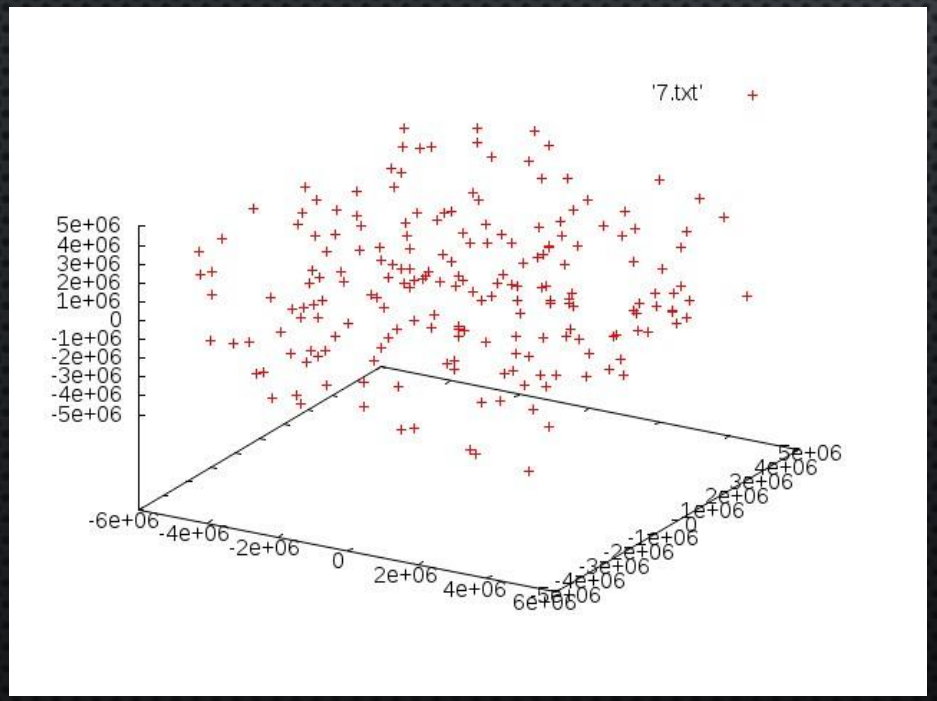
- Gravity Project kann beliebig viele Massenobjekte generieren
- Simuliert die gravitative Wirkung der Massen
- Die Zeit ist beliebig einstellbar
- Erstellt automatisch Textdateien mit den Positionen der Massen

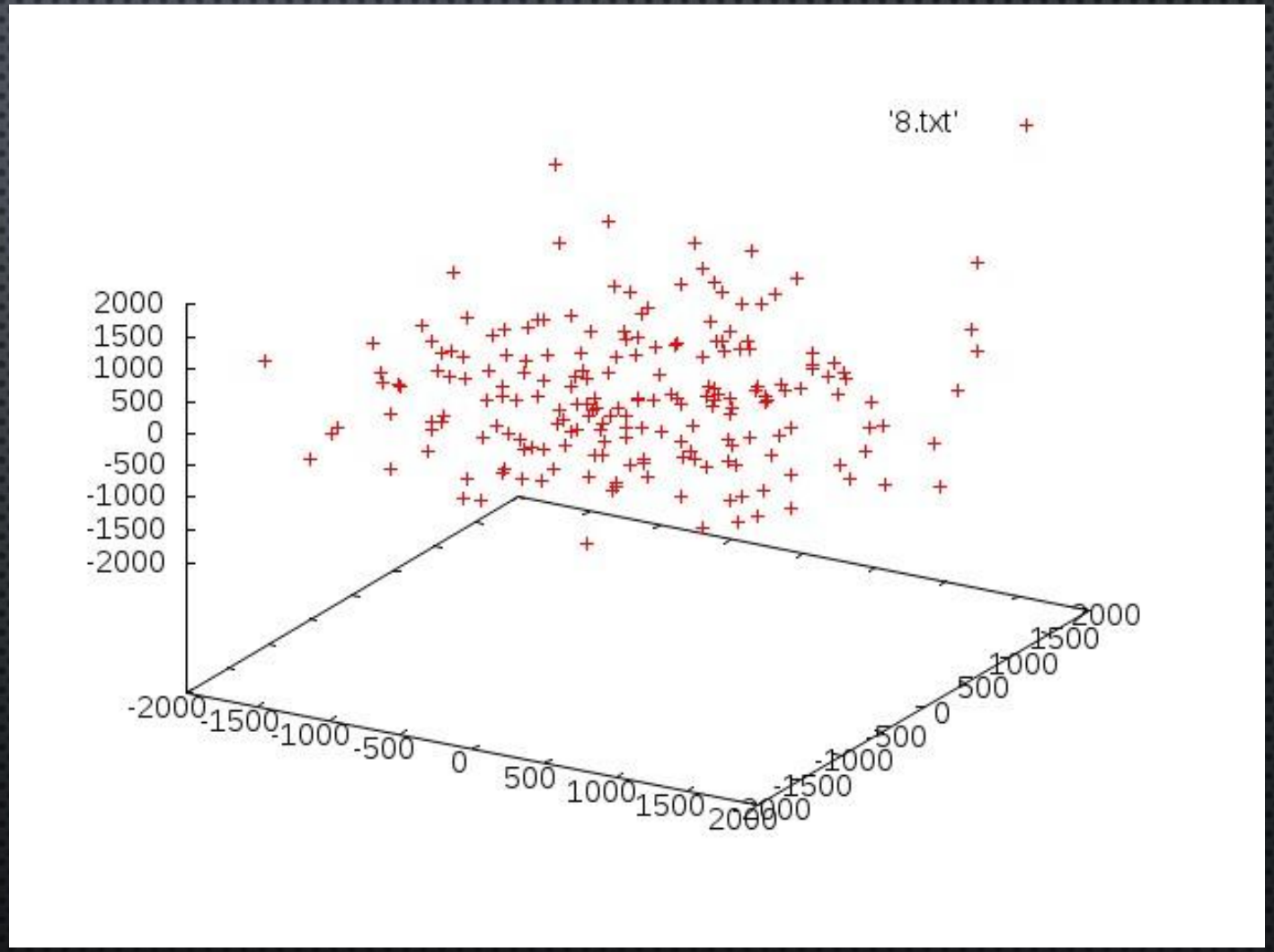
SIMULATION

- 200 OBJEKTE
- 16000 SEKUNDEN
- ERSTELLTE DATEIEN BEI 0, 5000, 10000, 15000 SEKUNDEN




'6.txt'
1e+07
8e+06
6e+06
4e+06
2e+06
0
-2e+06
-4e+06
-6e+06
-8e+06
-1e+07
-1e+07
-5e+06
0
5e+06
1e+07
1.5e+07


'7.txt'
5e+06
4e+06
3e+06
2e+06
1e+06
0
-1e+06
-2e+06
-3e+06
-4e+06
-5e+06
-6e+06
-4e+06
-2e+06
0
2e+06
4e+06
6e+06


'8.txt'
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2000
-1500
-1000
-500
0
500
1000
1500
2000

DANKE FÜR DIE AUFMERKSAMKEIT