

c3particles
Modellierung eines Partikelsystems in C++
Praktikumsabschlusspräsentation

Rosalie Kletzander

Institut für Informatik, LMU München

6.8.2017

- Partikelsystem: Simulation (oft in der 3d Grafik) von chaotischen Systemen und natürlichen Phänomenen
- C++ als Werkzeug: mathematisch Ausdrucksstark, formale Konzepte lassen sich sauber definieren
- Ziel: Entwicklung einer Programmierabstraktion für Newton'sche Mechanik in Partikelsystemen

"Ein Körper verharrt im Zustand der Ruhe [...] sofern er nicht durch einwirkende Kräfte zur Änderung seines Zustands gezwungen wird."

"Ein Körper verharrt im Zustand der Ruhe [...] sofern er nicht durch einwirkende Kräfte zur Änderung seines Zustands gezwungen wird."

"Das Verhältnis zwischen der Masse eines Objekts, seiner Beschleunigung und der angewandten Kraft ist $F = m \cdot a$ "

"Ein Körper verharrt im Zustand der Ruhe [...] sofern er nicht durch einwirkende Kräfte zur Änderung seines Zustands gezwungen wird."

"Das Verhältnis zwischen der Masse eines Objekts, seiner Beschleunigung und der angewandten Kraft ist $F = m \cdot a$ "

→ brauchen *Kräfte*, um Beschleunigungen zu erreichen

$$a = F/m \quad (1)$$

$$a = F/m \quad (1)$$

$$\vec{v}(t) = \int (\vec{a}) dt = \vec{a} * t + C_v \quad (2)$$

$$a = F/m \quad (1)$$

$$\vec{v}(t) = \int (\vec{a}) dt = \vec{a} * t + C_v \quad (2)$$

$$\vec{s}(t) = \int (\vec{v}) dt = \int (\vec{a} * t + C_v) dt \quad (3)$$

$$= \frac{\vec{a} * t^2}{2} + C_v + C_s \quad (4)$$

Das Superpositionsprinzip:

"Wirken auf ein Objekt mehrere Kräfte, so addieren sich diese vektoriell zu einer resultierenden Kraft auf."

$$\vec{F}_{res} = \vec{F}_1 + \vec{F}_2 + \dots + \vec{F}_n \quad (5)$$

- Modellierung des Partikelsystems basiert stark auf der Physik
- besteht aus Newton'schen Objekten (Partikeln) und Kräften
→ Konzepte
- die Expressions müssen die Gesetze der Physik erfüllen können

Expressions für "Newton'sches Objekt"

- $F = m * a \rightarrow$ "apply_force(NO)", oder " $NO \ll Kraft$ "
- Berechnung der Position \rightarrow "update(NO)"

Expressions für "Kraft"

- Auslöser für eine Kraft \rightarrow "calc_force(NO, NO, ff)"
- Superpositionsprinzip \rightarrow "accumulate(f1, f2, ..)"

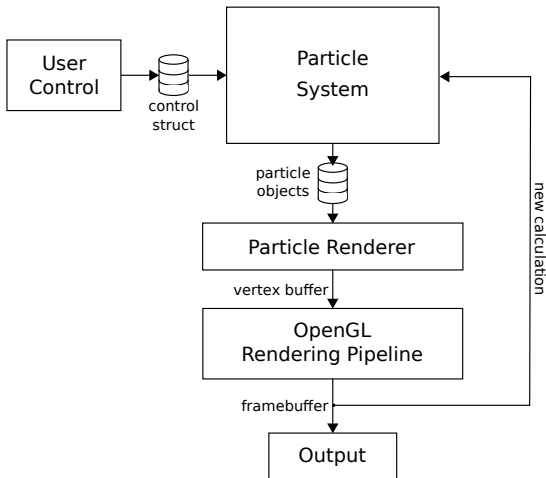


Figure: Systemdiagramm

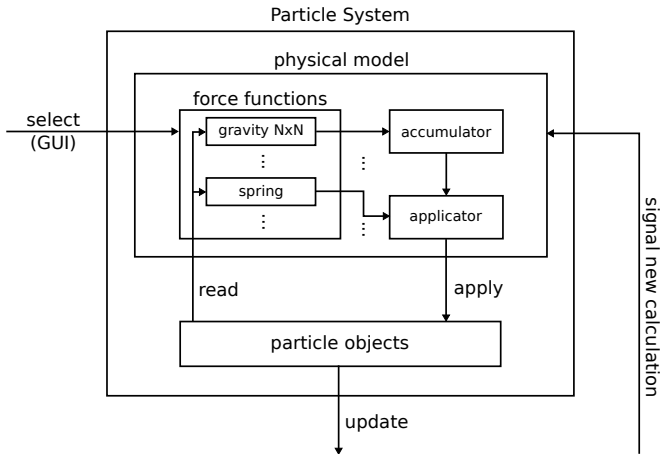


Figure: Das Partikelsystemmodell im Detail

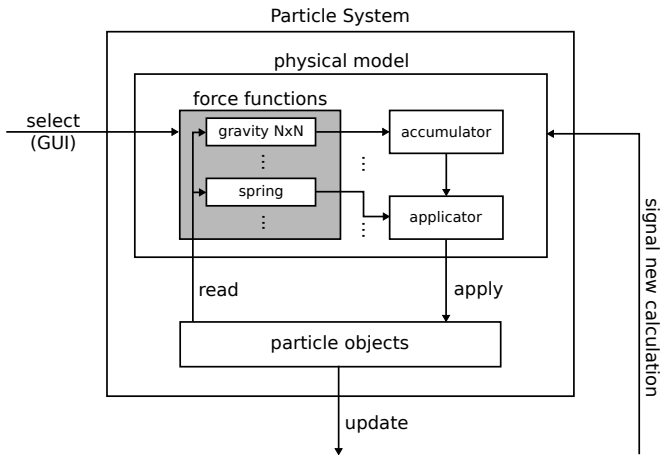


Figure: Die Kraftfunktionen (basierend auf calc_force)

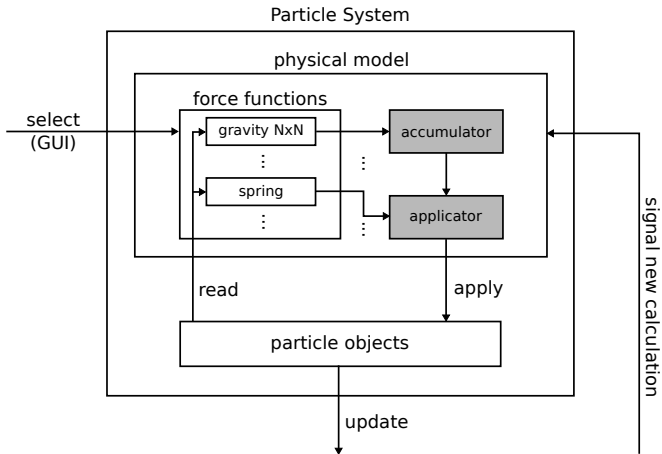


Figure: Externe vs Inter-Partikel Kräfte

Listing 1: Beispiel von Berechnung und Anwendung von Kräften

```
1 //iterate over all particles in the particle system
2 std::for_each(ps.begin(), ps.end(), [&ps](c3p::Particle&
   p)
3 {
4     // simple user-defined attraction force
5     p << calc_force(p, Particle(), [p](const Particle &,
        const Particle &)
6     {
7         glm::vec3 direction =
            glm::normalize(glm::vec3(0,0,0) - p.location());
8         return direction * 0.1;
9     });
10
11     // spring force from virtual particle at (0,0,0) to
        each particle
12     p << spring(p, Particle(0,0,0), {spring_constant,
        spring_length});
```

Listing 2: Beispiel von Berechnung und Anwendung von Kräften

```

1      ...
2
3      //gravitational forces between particles
4      p << c3p::accumulate(p,
5                           ps.particles(),
6                           {ps.g_constant()},
7                           c3p::gravity);
8  }
```

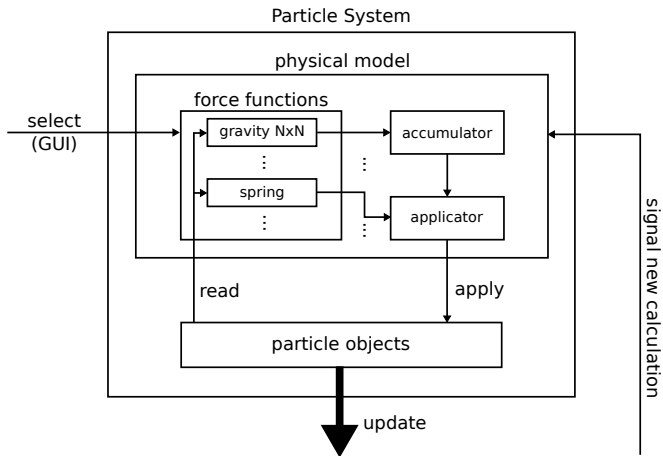


Figure: Aktualisieren der Geschwindigkeit und des Orts

Listing 3: Update Funktion

```

1 void update(Particle &p)
2 {
3     //v(t) = a*t + v(t-1)
4     p.velocity = p.acceleration * 1.0f + p.velocity;
5     //deltaT = 1.0
6
7     //s(t) = (a*t^2)/2 + v(t) + s(t-1)
8     p.location = (p.acceleration * 1.0f) / 2.0f +
9         p.velocity + p.location;
10
11     //acceleration is not accumulative, but recalculated
12     //at each time step
13     p.acceleration = {0, 0, 0};
14 }

```

- Parallelisierung der Iterationsschleifen über die Partikel

- Parallelisierung der Iterationsschleifen über die Partikel
- Offloading auf die GPU

- Parallelisierung der Iterationsschleifen über die Partikel
- Offloading auf die GPU
- Partitionierte, semi-symmetrische Kräfte matrix

Physik →

$$\vec{s}(t) = \frac{\vec{a} * t^2}{2} + C_v + C_s$$

$$F = m * a$$

$$\vec{F}_{res} = \vec{F}_1 + \vec{F}_2 + \dots \vec{F}_n$$

Konzepte →

"Newton'sches Objekt"

"NO << Kraft"

"update(NO)"

"Kraft"

"calc_force(NO, NO, ff)"

"accumulate(f1, f2, ..)"

Implementierung

