Prof. G. Zachmann
David Mainzer

TU Clausthal
Institut für Informatik
22. Oktober 2010

Wintersemester 2010/11

Übungen zu Virtuelle Realität und Simulation - Blatt 4

Abgabe am Montag, den 1. 11. 2010, 15:00 Uhr

Aufgabe 1 (Euler-Integration (Partikelsystem), *10 Punkte*)

Ihre Aufgabe besteht darin, die Partikel des Partikelsystems zu animieren. Das Framework stellt eine Klasse Particles zur Verfügung. Sie beschreibt genau ein Partikel. In der Klasse ParticleSystem sollen Sie die Partikel animieren. Hier ist die Funktion

```
public void evolveParticle()
```

entscheidend. In ihr soll die neue Position der Partikel mittels Euler-Integration bestimmt werden. Als Kraft, welche auf die Partikel wirken, soll zunächst nur die Erdanziehungskraft berücksichtigt werden. Die Änderung der Zeit, sprich Δt, kann aus der Variable **long** runtime (in ms) bestimmt werden. Die Variable runtime gibt an, wie lange die Simulation bereits läuft.

Lesen Sie sich zusätzlich den Anhang "Informationen zum Partikelsystem" durch. Dieser enthält weitere wichtige Informationen zum Partikelsystem.

(Da diese Aufgabe benötigt wird, um Aufgabe 2 und/oder Aufgabe 3 lösen zu können, kann ich Ihnen auf Anfrage per E-Mail (dm@tu-clausthal.de) eine Lösung zukommen lassen.)

Aufgabe 2 (Optische Darstellung (Partikelsystem), *6 Punkte*)

Bei dieser Aufgabe ist Ihre Kreativität gefragt. Das Partikel soll abhängig von der Lebenszeit die Farbe und Größe ändern. Vorbild soll hier ein Feuerpartikel sein, welches mit der Zeit abkühlt und an Masse verliert. Beachten Sie, dass die Änderung der Größe über die Skalierung geschehen *muss*, da im Framework das Partikel vom Type

```
geometry Box { size 0.01 0.01 0.01 }
```

ist. Wie Sie wissen, ist das Feld size vom Typ **field** und kann somit nicht geändert werden.

Ihre Aufgabe:

- Berechnen Sie Farb- und Größenänderung in der Funktion **public void** evolveParticle () und speichern sie diese zu jedem Partikel (verwenden Sie die vorgegebenen Variablen).
- Übergeben Sie in der Funktion **public void** draw() die vorher berechneten Werte an den Scenengraphen. Hierzu können Sie sich an der Art und Weise, wie die Translation gesetzt wird, orientieren, um die Farbe und die Skalierung der Partikel an den Scenengraphen zu übergeben.

Aufgabe 3 (Operationen auf Partikeln (Partikelsystem), *8 Punkte*)

Stellen Sie sich folgendes Szenario vor. Das Partikelsystem verschießt geladene Teilchen. Diese Teilchen sollen nun von dem Pendel, welches sich schwingend durch die Teilchen bewegt, beeinflusst werden. Der Mittelpunkt der Kugel des Pendels soll als Ausgangspunkt genommen werden. Der Einfluss auf die Partikel soll dabei abhängig von der Entfernung der Partikel zu diesem Mittelpunkt sein. Der Mittelpunkt der Kugel ist in der Variablen

```
private float energyPoint [] = new float[3];
```

gespeichert. Ihre Aufgabe ist nun die Bestimmung des Abstandes und daraus die Berechnung der Anziehungs- oder Abstoßungskraft. Dabei dürfen Sie der Einfachheit halber einfach das Gravitationsgesetz anwenden. Die Funktion:

```
public void draw()
```

zeichnet die Partikel in den Szenengraph der VRML Szene.

Die von Ihnen zu bearbeitende Funktion ist:

```
public void evolveParticle()
```

diese bestimmt die neue Position für jedes Partikel. Der Einfluss des Pendels muss nun auf das Partikel angewandt werden.

Als Zusatz sei noch erwähnt, dass das Array:

```
Particles [] myParticleArray;
```

alle Partikel enthält.

Informationen zum Partikelsystem:

Ein Partikel ist wie folgt gegeben:

```
public class Particles
{
    private float lifetime;                      //!< total lifetime of the particle
    private float decay;                         //!< decay speed of the particle
    private float [] color = new float[3];       //!< color values of the particle
    private float [] position = new float[3];    //!< position of the particle
    private float [] speed = new float[3];       //!< speed of the particle

    private boolean active;                      //!< set particle to active or not

    public final float maxLifetime = 10000F;     //!< maximal lifetime of particles
    .
    .
    .
}
```

Beispiel für das Setzen einer neuen Lebenszeit des Partikels i:

```
myParticleArray[i].setLifetime(
    myParticleArray[i].getLifetime() - myParticleArray[i].getDecay()
);
```

Auf jede Variable des Partikels kann mittels *getter* und *setter* zugegriffen werden!

Das Skript zur Erzeugung der Partikel:

```
DEF PARTICLESYSTEM Script {
    field SFNode ParticlesIN USE ParticlesGroup

    # set NUM of particles
    field SFInt32 particles 100
    field SFNode Pendel USE pendel

    eventIn SFTime  cycletime
    eventIn SFVec3f bboxPendelCenter
    eventIn SFVec3f bboxPendel

  url       "ParticleSystem.class"
}
```

Hier kann über die Variable `SFInt32 particles 100` die Anzahl an Partikeln im System gegeben werden. Dieser Wert sollte nicht zu hoch gewählt werden, da sonst das Programm (auf Grund von der vielen Updates im Scenengraph) nicht mehr flüssig läuft.