Einführung in die Computergrafik

Wintersemester 2018/2019

Prof. Dr. Philipp Jenke

HAW HAMBURG | Department Informatik

Aufgabe 4

Hinweis zu den Mathematik-Klassen (`Vector`, `Matrix`, ...): Sie arbeiten hier mit Referenzen. Eine Zuweisung erzeugt also keine Kopie des Vektors. Wenn Sie eine Kopie benötigen, dann verwenden Sie den Kopier-Konstruktor (`new Vector(andererVektor)`) oder die `copy(andererVektor)`-Operation.

Monster auf Spline-Wegen

In diesem Aufgabenblatt erweitern Sie die Spielfunktionalität um Monster. Es handelt sich um fliegende Monster, die sich auf Bahnen durch das Level bewegen. Im finalen Spiel wird die Spielfigur es vermeiden müssen, mit den Monstern zu kollidieren.

Hermite-Kurve

Schreiben Sie eine Klasse `Hermite-Curve`, die die Evaluation einer Hermitekurve $q(\lambda)$ für Parameter λ aus $D = [0; 1]$ erlaubt. Die Klasse soll außerdem in der Lage sein, für jedes $\lambda \in D$ die Tangente $q'(\lambda)$ zu bestimmen. Sichern Sie die Klasse durch Unit-Tests ab.

Hermite-Spline

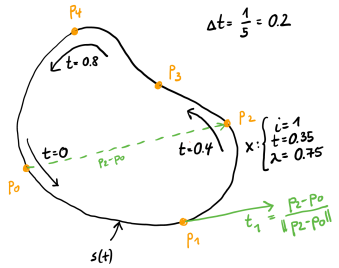

Abbildung 1: Hermite-Spline

Nun implementieren Sie auf Basis der Hermite-Kurven einen zirkulären Hermite-Spline $s(t)$. Der Spline wird durch seine Kontrollpunkte $C = p_0 \dots p_{n-1}$ festgelegt. Dem letzten Kontrollpunkt folgt virtuell wieder der erste. Die Tangente an jedem Kontrollpunkt entspricht dem normalisierten Differenzvektor aus dem Nachfolger- und dem Vorgängerpunkt. Daraus lassen sich n Hermite-Kurvensegmente ableiten. Wir benennen das Kurvensegment zwischen den Kontrollpunkten i and $i + 1$ mit Segment i.

Basierend auf dieser Definition müssen zwei Parametertypen unterschieden werden. Der globale Parameter t aus dem Definitionsbereich $D = [0; 1]$ repräsentiert den vollständigen Spline. Mit $t = 0$ and

$t = 1$ wird wegen der Zirkularität der gleiche Punkt erreicht: $s(0) = s(1)$. Zusätzlich gibt es für jedes Kurvensegment den lokalen Parameter λ, ebenfalls in $[0; 1]$ definiert (siehe Abbildung 1). Für einen Parameterwert t muss man nun

- das Kurvensegment i bestimmen und
- den lokalen Parameter λ bestimmen.

Dies geschieht mit folgender Berechnungsvorschrift:

$$\begin{aligned} \Delta t &= \frac{1}{|C|} \\ i &= \frac{t}{\Delta t} \\ \lambda &= \frac{t - i \cdot \Delta t}{\Delta t} \\ s(t) &= q(\lambda) \end{aligned}$$

Sichern Sie die Klasse durch Unit-Tests ab.

Monster

Ergänzen Sie das Spiel um eine `Monster`-Klasse, die von `GameObject` erbt. Darin müssen Sie zunächst die `fromJson` Methode überschreiben. In der JSON-Beschreibung hat ein Monster-Objekt einen Brick, der angibt, über welchem Brick sich das Monster bewegt. Außerdem verwaltet jedes Monster einen zirkulären Hermite-Spline durch den es sich kontinuierlich bewegt. Der Spline geht durch die folgenden Punkte:

- $o + (0, 0, 0)$
- $o + (s/2, s/2, 0)$
- $o + (0, s, 0)$
- $o + (-s/2, s/2, 0)$,

wobei o der Mittelpunkt Oberseite des Bricks ist. Dieser kann vom `World`-Objekt berechnet werden.

Im Rahmen des Praktikums wird über die Aufgaben hinweg ein Jump'n'Run Computerspiel (Platformer) entwickelt. In jedem Aufgabenblatt entwickelt sich das Spiel ein wenig weiter. Das Spiel ist in das Framework für die Lehrveranstaltung integriert. Eine Anleitung zum Einrichten des Frameworks finden Sie in der Dokumentation zum Framework auf der EMIL-Seite zur Veranstaltung. Innerhalb des Frameworks ist bereits Funktionalität zum Spiel vorgegeben. Eine Dokumentation dieser Funktionalität findet sich im doc-Ordner des Packages für den Platformer.
Das Spiel kann sowohl auf dem Smartphone unter Android verwendet werden als auch in einer Desktop-Anwendung. Ich empfehle, im Praktikum auf die Desktop-Entwicklung zu setzen, weil sich diese viel einfacher und schneller debuggen und kompilieren lässt. Informationen zur Einrichtung finden Sie im doc*-Ordner des Frameworks.*