Trimesteroverschrijdend Project: Curve Editor

Groep 13: S. Staessens, S. Polders

dinsdag 12 februari, 2008

1 Onderwerp: Curve Editor

Het project delen we (voorlopig) op in de secties *basis en optioneel.* De basis geeft de functionaliteit weer die zeker in het project verwerkt zal worden. Optioneel daarentegen zijn functies die pas geïmplementeerd zullen worden als de basis grondig is uitgewerkt.

- **Basis**
  - Punten ingeven (m.b.v. muis of keyboard)
  - Connectie van ingegeven punten (Bezier, Hermite)
  - Verschillende curvens met elkaar connecteren
  - verplaatsen van punten (m.b.v. muis of keyboard)
  - mogelijkheid om verschillende kleuren en diktes te gebruiken
  - Save & load functies
- **Optioneel**
  - Tools:
    - Soundmixer
    - Transformer
    - Curving
    - ...
  - Oplichting aangeklikte curve/punt
  - 3d curves
  - ...

2 Algoritmes

- **Bezier:**

1. formule: $\sum_{i=0}^{n} \binom{n}{i} . P_i . (1-t)^{n-i} . t^i$ ($t \in [0,1]$)
2. uitwerking:
   zij P een vector van controlepunten (ingegeven punten).
   zij B een vector met berekende punten.
   n het aantal ingegeven punten
   N het aantal punten tussen 0 en 1 dat we willen berekenen (== t)

   initialiseer B met enkel 0-waarden.

from i = 0 to n do (1)
    from j = 1 to N do
        from k = 0 to n do
            B[i*N + j] += $\binom{n}{k}.P[k].(1-t)^{n-k}.t^k$

Het algoritme doet het volgende:

- **Hermite**

1. formule & algoritme:

De volgende punten worden door de gebruik ingegeven:
P1: Het startpunt van de curve
T1: De richting waarin het beginpunt gaat (de raaklijn van de curve in punt)
P2: Het eindpunt van de curve
T2: De richting waarin het eindpunt gaat (de raaklijn van de curve in punt)

Deze 4 punten worden dan vermenigvuldigd met de 4 basis functies van Hermite:
$h1(s) = 2s^3 - 3s^2 + 1$
$h2(s) = -2s^3 + 3s^2$
$h3(s) = s^3 - 2s^2 + s$
$h4(s) = s^3 - s^2$

Matrix S: Het interpolatie punt en zijn 3 machten
Matrix C: De parameters van de Hermite curve
Matrix h: De matrix notatie van de 4 Hermite functies

$$S = \begin{pmatrix} s^3 \\ s^2 \\ s^1 \\ 1 \end{pmatrix} \quad C = \begin{pmatrix} P1 \\ P2 \\ T1 \\ T2 \end{pmatrix} \quad h = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -2 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Om een punt op de curve te berekenen stel je de vector S op, vermenigvuldig je hem met de matrices h en C.
$P = S * h * C$

- **Curve zoeken met behulp van een gegeven punt**

1. Duid een punt aan op de curve die gezocht moet worden.
2. Doe zolang de curve niet gevonden is, voor elke volgende curve in de vector:

3. bereken voor elk punt in de curve de afstand tot dat punt
4. indien afstand < 0.01 ga naar 5 anders 2
5. indien curve gevonden, “Hoera!”, anders “:-(“

Mogelijke verfijning van het vorige algoritme:

1. Verdeel het tekengebied in N aantal gelijke rechtoeken
2. Verdeel deze rechthoeken over een goedgekoze hashmap
3. voeg een referenties van de curve toe bij de juist haskey
4. Bereken de afstand van de oorsprong tot het ingegeven punt
5. gebruik deze afstand om in de hasmap te zoeken welke curves punten hebben
   op deze afstand van de oorsprong
6. ga met deze subset van curves naar stap 2 in het vorige algoritme

Hierdoor zal het berekenen van een curve iets trager gaan (de hasmap moet in orde gebracht worden). Het zoeken daarentegen zal sneller gaan. Normaal gezien is de gebruiker meer bereid om te wachten op een berekening dan te wachten op het vinden van een curve. Dus dit zou een goed overwogen tradeoff zijn.