Trimesteroverschrijdend Project: Curve Editor

Groep 13: Sibrand Staessens en Sibren Polders

Dinsdag 12 februari, 2008

1 Beschrijving

Het project delen we, voorlopig dan toch, op in twee secties: *basis* en *optioneel.* De *basis* stelt de functionaliteit voor die zéker in het project verwerkt zal worden. *Optioneel* daarentegen zijn functies die pas geïmplementeerd zullen worden als de basis grondig is uitgewerkt.

- **Basis**
  - een interpolatiealgoritme kiezen en daarna punten in het tekenvlak aanklikken; m.b.v deze punten wordt dan de curve berekend en uitgetekend a.h.v. dat algoritme (Bézier of Hermite)
  - nieuwe controlepunten ingeven, m.b.v. muis (op het tekenvlak klikken) of toetsenbord (coördinaten ingeven), en aan een bestaande of nieuwe curve toevoegen
  - een punt selecteren en de daaraan verbonden curve hertekenen indien dat punt versleept wordt, ook dit kan via de muis of het toetsenbord gebeuren
  - één curve selecteren en daarvan de eigenschappen laten weergeven a.h.v. een groep radio-buttons, die ook de mogelijkheid bieden om de eigenschappen van die curve te veranderen en te laten weergeven
  - een curve selecteren en daarvan de dikte en kleur veranderen
  - alle curven selecteren en daarvan de eigenschappen simultaan veranderen; selectie van een andere eigenschap leidt dan dus tot het hertekenen van alle curves
  - twee curves selecteren en verbinden, weer met behulp van één van de verschillende interpolatiealgoritmen
  - Save & Load: de mogelijkheid aanbieden om de curves in het huidige tekenvlak op te slaan en terug in te laden
- **Optioneel**
  - Tools:
    * Soundmixer: de curve, of curves voor meerdere instrumenten bijvoorbeeld, stelt dan een tijdslijn voor. De X-as wordt dan van links naar rechts doorlopen, en hoe hoger de corresponderende Y-waarde op dat moment, hoe hoger de toon die men te horen krijgt.
    * Transformer: we geven een afbeelding en laten die weergeven. De gebruiker plaatst een aantal punten in het vlak, en alnaargelang hij ze nadien versleept, vervormt de afbeelding. Ons lijkt dit het moeilijkst van al te implementeren, maar niet zó onoverkomelijk.

    - Curving: we geven een afbeelding in, en krijgen een curve-variant van die afbeelding terug. Elke lijn op de afbeelding wordt dus omgezet in één of andere curve en weergegeven.
    - ...
  - de interactiviteit met de gebruiker verder uitwerken: indien een curve geselecteerd wordt, deze laten oplichten, dikker worden, ..., indien twee curves verbonden worden, het verbindende stuk laten oplichten, ...
  - curves in een 3D-omgeving; qua ontwerp is dit zeer gelijkaardig aan wat we nu voor 2D gedaan hebben, maar qua implementatie hebben we vooralsnog geen idee
  - ...

Dit alles gaan we in Java implementeren, omdat wij ook wat ervaring wensen op te doen met grotere Java-projecten. Ook is dit een manier om de talrijke libraries wat beter te leren kennen.

2 Analyse

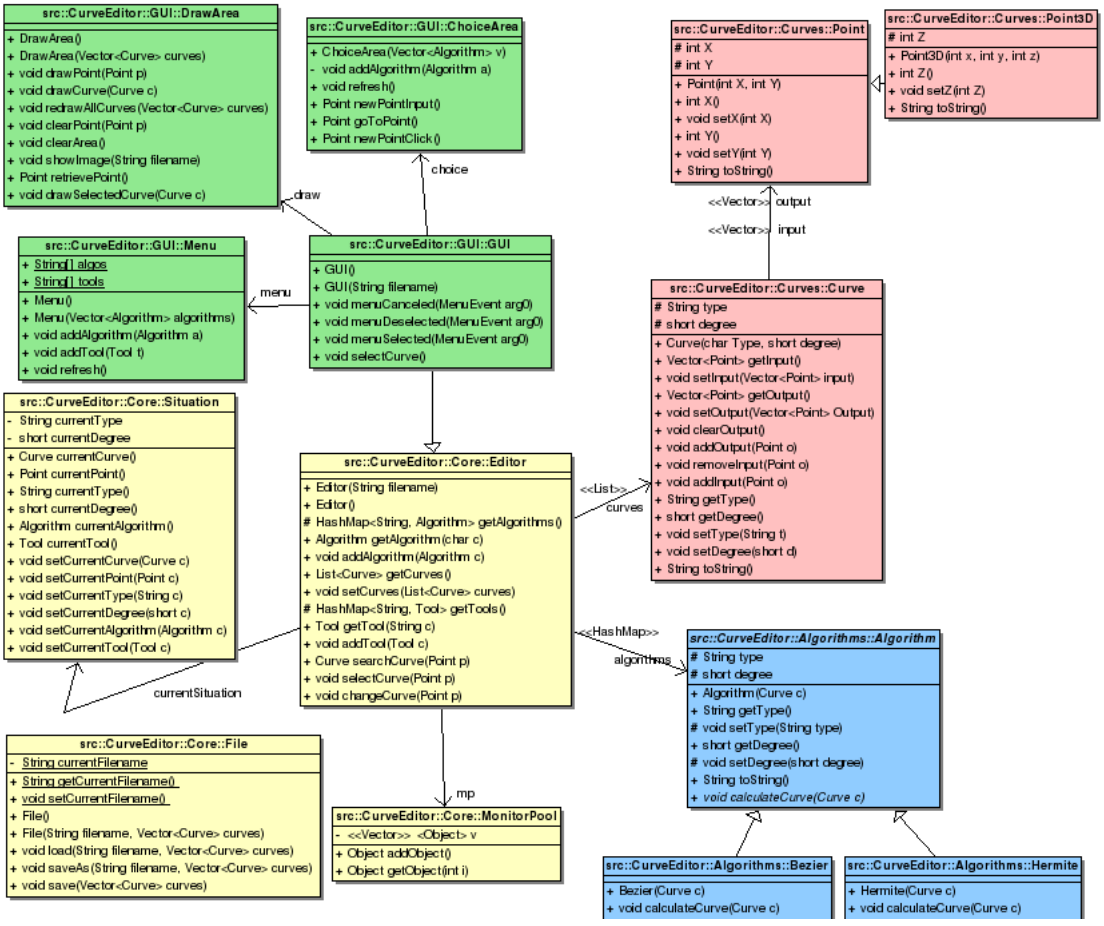
2.1 Klassediagram

Zoals in de bijgevoegde figuur te zien is, hebben we de applicatie opgedeeld in vijf delen:

- de klassen Point, Point3D en Curve; dit zijn ADT's en houden dus specifieke eigenschappen bij en voorzien ook de functies om deze te veranderen. Meer hierover kan je iets verder lezen.
- de abstracte klasse Algorithm en zijn subklassen Bezier en Hermite; deze vullen de outputvector van een meegegeven Curve-instantie m.b.v. de input-vector van die instantie. Elk soort algoritme berekent dit anders, uiteraard.
- de abstracte klasse Tool en zijn subklassen; deze geven bijvoorbeeld een vector van curves terug indien een afbeelding werd meegegeven (Curving), spelen muziektonen af als een vector van curves werd meegegeven (Soundmixer), ... Er zijn diverse mogelijkheden en gaandeweg de uitwerking van ons project zal duidelijk worden welke interactiemanier ons het meest optimaal lijkt.
- het core-gedeelte: Editor en bijhorende klassen File, Situation en MonitorPool. Editor is het centraal orgaan van de applicatie en stuurt het dataverkeer tussen de verschillende applicatieonderdelen, en bevat tevens de verzameling van reeds ingevoerde controlepunten en berekende curven. Situation is een soort van hulpklasse, die bijhoudt

welke curve momenteel actief/geselecteerd staat, welk type algoritme en welke orde-grootte momenteel in het menu aangevinkt staat, welk punt is aangeklikt geweest, ... Eén instantie van Situation wordt dus m.b.v. read- en write-locks in verschillende klassen gelezen en aangepast. MonitorPool ondersteunt het gebruik van deze locks door objecten te leveren waarop het wait-notify-systeem kan gebruikt worden. Andere oplossingen zijn uiteraard ook mogelijk, maar deze structuur leek ons volledig en veilig, doch vrij simpel.

- GUI, en DrawArea, ChoiceArea en Menu: deze klassen stellen duidelijk het grafische gedeelte van de applicatie voor. Met behulp van listeners in Editor kan de juiste functie aangeroepen worden; via Editor kan dan weer de nodige functie van DrawArea opgeroepen worden, gevolgd door bijvoorbeeld de nodige functie van een algoritme, ...



Figuur 1: Het voorlopig UML-diagram.

2.2 ADT's

Point en Point3D spreken voor zich: zij representeren punten in het vlak of in de ruimte voor door middel van de coördinaten.
Curve bevat enerzijds een Vector van Points, die de verzameling van de initiële controlepunten voorstelt. Wij hebben voor een Vector gekozen, omdat de volgorde van die punten heel belangrijk is. Anderzijds bevat Curve een tweede Vector van Points, die de verzameling van de interpolerende punten voorstelt; deze Vector wordt gevuld met behulp een functie van een Algorithm-instantie die de input-vector als parameter meekrijgt.
Algorithm is een abstracte klasse, daar de methode *calculate* in de subklassen dient geïmplementeerd te worden. Algorithms kunnen beschouwd worden als ADT's die curves creëren aan de hand van een verzameling meegegeven punten.
In de centrale klasse Editor maken we gebruik van HashMaps voor het bijhouden van alle algoritmen en tools. Op deze manier kunnen we makkelijk en efficiënt het nodige object vinden aan de hand van een String die de naam van de tool/algoritme voorstelt. Ook kunnen we met behulp van deze HashMaps eenvoudig de menu's aanmaken: gewoon voor elke key een menu-item aanmaken en dat item aan de overeenkomstige value koppelen.

2.3 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.
   Zij n het aantal ingegeven punten.
   Zij N het aantal punten tussen 0 en 1 dat we willen berekenen (N is het aantal t-waarden waarvan we een interpolerend punt willen berekenen).

   Initialiseer B met enkel 0-waarden.

   from i = 0 to n do
   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 vult dus de vector van punten B.

- **Hermite**

  1. Formule & algoritme:

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

     De vier punten worden dan vermenigvuldigd met de vier Hermite-basisfuncties:
     $h1(s) = 2s^3 - 3s^2 + 1$
     $h2(s) = -2s^3 + 3s^2$
     $h3(s) = s^3 - 2s^2 + s$
     $h4(s) = s^3 - s^2$

     Matrix S: het interpolatiepunt en zijn machten
     Matrix C: de parameters van de Hermite-curve
     Matrix h: de matrix-notatie van de vier 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 en vermenigvuldig je die met de matrices h en C.
     $P = S * h * C$

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

  1. Duid een punt aan op of in de buurt van de curve die gezocht moet worden.
  2. Doe, zolang de curve niet gevonden is, voor elke volgende curve in de verzamelende Vector:
  3. Bereken voor elk punt in de curve de afstand tot dat punt.
  4. Indien de afstand < 0.01, ga naar 5. Anders, ga naar 2.
  5. Indien de curve is gevonden, “Hoera !”. Anders, “:-(“.

Een mogelijke verfijning van het vorige algoritme:

1. Verdeel het tekengebied in N aantal gelijke rechthoeken.
2. Verdeel deze rechthoeken over een goedgekozen HashMap.
3. Voeg een referentie van elke curve toe bij de juiste Hash-key.
4. Bereken de afstand van de oorsprong tot het ingegeven punt.
5. Gebruik deze afstand om in de HashMap te zoeken welke curves punten hebben op deze afstand van de oorsprong.
6. Ga met deze deelverzameling van alle curves naar stap 2 in het vorige algoritme.

Hierdoor zal het berekenen van een curve iets trager verlopen (de HashMap moet immers steeds in orde gebracht worden). Het zoeken daarentegen zal wél sneller gaan. Een doorsnee gebruiker is eerder bereid om meer tijd te geven aan het berekenen van een curve dan aan het zoeken van een curve. Dit zou dus een goed overwogen trade-off zijn.

2.4 Bestandstructuur

Hier kunnen we kort over zijn. Meer dan de op het moment van opslaan weergegeven curves moet er eigenlijk niet opgeslaan worden, daar hieruit alles weer kan gereconstrueerd worden; deze curves kunnen we binair wegschrijven, of tekstueel om ze daarna terug te gaan parsen naar de juiste curve. Beide aanpakken vergen niet meer dan alle elementen van de Vector curves in de klasse Editor wegschrijven en terug inladen.

3 Taakverdeling en planning

Planning:

- week 7, 12/02: inleveren verslag en planning
- week 7: opzoekwerk *Java*: Swing, EventListeners, wait/notify
- week 8: feedback afwachten + samenbrengen van de resultaten van het opzoekwerk
- week 9, 26/02: feedback + ev. veranderingen aanbrengen, starten met implementeren: GUI + algoritmen
- week 10-12: examens
- week 13-14: algoritmen + GUI af, Save en Load idem
- week 1-2: grondig testen van code + implementeren van de dan voor de hand liggende functies

- week 3-4: basisfuncties compleet af + kijken naar de implementatie-mogelijkheden van de extra's + bijsturen doelstellingen
- week 5-6: implementeren van de gekozen extra's
- week 7-8: afronden + testen
- week 9-10: verder testen + presentatie voorbereiden + verslag schrijven

Taakverdeling: ieder één algoritme, de GUI (één iemand de DrawArea, de andere de rest) + de extra tools nog onder elkaar te verdelen (elk één tool bijvoorbeeld, indien de tijd dat toelaat).

4 SVN-repository

De meest recente bronbestanden zijn steeds terug te vinden op:
http://code.google.com/p/curveeditor/