Project Grafische Technieken 2005 : Representatie van Krommen mbv. het Lane-Riesenfeld algoritme

Benjamin De Leeuw

Departement Zuivere Wiskunde en Computer Algebra
Galglaan 2, B-9000 Gent
[Benjamin][dot][DeLeeuw][at][UGent][dot][be]
Gent University, Belgium

March 27, 2005

1 Omkadering

Dit project wordt geformuleerd in het kader van het vak Grafische Technieken, waarbij we meer specifiek het praktische gedeelte van deze cursus willen behandelen. Uit onze ervaring blijkt dat het begrijpen van de daar bescherven theorische algoritmen vlotter verloopt wanneer een eigen versie wordt gemaakt van deze algoritmen. De klemtoon ligt hierbij op de computerrepresentatie van de benodigde datastructuren (de input) en het gebruik van deze datastructuren in de functies die krommen op het computerscherm weergeven (de output).

Dit jaar beschouwen we een zeer interessant algoritme : Het Lane-Riesenfeld algoritme, een veel gebruikt algoritme in professionele CAD-paketten. Het algoritme is op zich eenvoudig, en daarom zeer interessant ter implementatie van benaderingen van krommen van arbitraire graad. Ook gesloten krommen zal men met dit algoritme kunnen benaderen.

Om overlapping met de syllabus tot het strikte minimum te beperken, verwijzen we voor de theoretische achtergrond en andere leuke weetjes naar de cursus, gegeven door J. Moreau. We beperken ons in deze projectopgave tot de beschrijving van de benodigde datastructren en functionaliteit die moet worden geïmplementeerd. Men is vrij verdere uitbreidingen te maken naar smaak en inzicht. Belangrijk hierbij is dat spectaculaire implementaties iets meer beloond worden dan de basisimplementaties. Let er wel op dat zulke spectaculaire zaken meestal veel tijd vergen, en geen absolute noodzaak zijn voor dit project.

De oplossing van dit project bedraagt een programma in Java, dat de in het onderstaande besproken functionaliteit implementeert. Dit document is vergezeld van 4 figuren, verzameld in de file $FiguurGT2005.pdf$. In de tekst wordt er naar deze figuurnummers verwezen.

2 Input

We gaan uit van een *tekenscherm* (zie later: sectie Output) met een *oorsprong*.

$$\begin{array}{l} Canvas\ tekenscherm \\ oorsprong(int\ x, int\ y, int\ z) \end{array}$$

waarbij $oorsprong(int\ x, int\ y, int\ z)$ een relatie legt tussen wat er op het scherm te zien is, en hoe het scherm behandeld zal worden als assenstelsel. We verwijzen hiervoor naar de syllabus.

De input van het programma zijn *controlepunten*. Controlepunten vormen een sequentie (lijst). De "inhoud" van een controlepunt wordt bepaald door diens aangeduide positie op het scherm en zijn volgorde in de lijst van controlepunten. Het aanduiden van de positie van een controlepunt gebeurt met de muis en een linker muisklik. De constructor zou er in een eenvoudige versie als volgt kunnen uitzien:

$$controlepunt(int\ x, int\ y, int\ z = 1)$$

We behandelen *controlepunten* in eerste instantie als liggende in het scherm-vlak.

Het tweede inputgegeven beslaat de *graad* van de kromme. De graad van de kromme bepaalt de vorm van de berekende curve. Voor meer achtergrond verwijzen we naar de syllabus.

$$int\ graad$$

Derde inputgegeven is het *aantal iteraties* ter benadering van de kromme. Dit aantal iteraties bepaalt de nauwkeurigheid van de benadering van de kromme.

$$int\ aantal_iteraties$$

3 Functionaliteit

Adhv. bovenstaande inputgegevens zijn we in staat om een kromme te tekenen volgens het Lane-Riesenfeld algoritme.

$$tekenKromme(List\ controlepunten, int\ graad, int\ aantal_iteraties)$$

waarbij *List controlepunten* een lijst is van *controlepunt*-en.

3.1 Het Lane-Riesenfeld Algortime

Afhankelijk van het aantal gewenste iteraties gaan we in dit algoritme de opgegeven controlepunten combineren tot nieuwe punten. De techniek van combineren wordt beschreven in de syllabus. Dit algoritme heeft als parameter de gewenste graad en het aantal gewenste iteraties. We geven het voorbeeld voor $graad = 2$:

Stel een sequentie van n controlepunten ($n > graad$)
$S = \langle c_0, c_1, \ldots, c_n \rangle$

Initialisatie:
$S_i = S$

Iteratiestap: (doe *aantal_iteraties* keer)
In sequentie S_i vervang elke c_i door twee keer c_i
$S_0 = \langle c_0, c_0, c_1, c_1, \ldots, c_n, c_n \rangle$

Neem voor deze sequentie het middelpunt van alle twee opeenvolgende puntentupels
$S_1 = \langle c_0, \frac{c_0+c_1}{2}, c_1, \ldots \frac{c_{n-1}+c_n}{2}, c_n \rangle$

Doe hetzelfde voor de bekomen lijst S_1
$S_2 = \langle \frac{3c_0+c_1}{4}, \frac{c_0+3c_1}{4}, \frac{3c_1+c_2}{4}, \ldots \frac{3c_{n-1}+c_n}{4}, \frac{c_{n-1}+3c_n}{4} \rangle$

Doe hetzelfde voor alle bekomen lijsten $S_2 \ldots S_{graad-1}$ om uiteindelijk S_{graad} te construeren
$S_{graad} = S_2$

Maak klaar voor volgende Iteratie:
$S_i = S_{graad}$
Einde **Iteratiestap**

Zoals vermeld zal de lijst S_{graad} uit een iteratiestap opnieuw fungeren als input voor de volgende iteratiestap (voor $graad = 3$, zal in één iteratie dus S_3 geconstrueerd moeten worden, voor $graad = 4$, S_4, enz). Samenvattend: één *iteratie* bestaat uit: één maal verdubbelen van de puntenlijst en nadien *graad* keer de berekening van de lijst van middelpunten tussen twee opeenvolgende punten, zoals hierboven beschreven. Deze iteratiestap wordt *aantal_iteraties* keer uitgevoerd. De nieuwe punten (S_{graad} na het aantal gewenste iteraties) vormen punten van de benaderde kromme. Door deze punten te verbinden met

lijnstukken, volgens dezelfde orde als deze waarin ze verwerkt werden (de sequentie van controlepunten, zie hierboven), verkrijgen we de gewenste benadering van de kromme. Deze punten moeten op het scherm worden weergegeven, samen met de lijnstukken, als benadering van de kromme (zoals in Figuur 1). In elke onderstaande context bedoelen we met het tekenen van de kromme, het tekenen van deze berekende punten en de verbinding van deze punten met lijnstukken.

3.2 Overige Functionaliteit

Volgende extra functionaliteit wordt geïmplementeerd in het programma:

- Toevoegen van controlepunten aan het einde van de sequentie (met linker muistoets, positie van de muis bepaalt de plaats op het scherm, en dus in het vlak waarin we werken).
- Undo functie om in omgekeerde volgorde van toevoeging, controlepunten te verwijderen uit de sequentie (rechter muistoets, positie onbelangrijk).
- Verplaatsen van controlepunten (linker muistoets, waarbij de muiscursor zich op een voorheen geplaatst controlepunt bevindt).
- Verhogen en verlagen van de graad van de benaderde kromme met het toetsenbord (cursortoetsen up en down respectievelijk, zie Figuur 1).
- Verhogen en verlagen van het aantal iteraties binnen zinvolle grenzen (cursortoetsen left en right respectievelijk, zie Figuur 1).
- Openen en sluiten van een kromme: via het toetsenbord ("o"-toets) kan men de sequentie controlepunten sluiten en openen. Hiervoor wordt automatisch één controlepunt toegevoegd, en gelijkgesteld aan het eerste controlepunt in de sequentie. Bij opnieuw openen, wordt de oorspronkelijke sequentie hersteld (zie Figuur 4). Men mag ervan uitgaan dat er geen controlepunten kunnen worden toegevoegd (met de muis) indien de sequentie in gesloten formaat is.
- Grid voor de controlepunten: een grid is een raster (parallelle horizontale en verticale lijnen op gelijke afstand over heel het scherm) dat de plaatsing van controlepunten beperkt tot de snijpunten van deze grid. Voor elk vierkant in de grid, worden alle controlepunten die in dit vierhoek worden geplaatst, verplaatst naar het dichtstbijgelegen hoekpunt van het vierkant in kwestie (zie Figuur 2). Deze "snap-to-grid" functionaliteit zet men aan of uit men met de "g"-toets. Startend van een standaardgrid, kan deze ruwer of fijner gemaakt worden met de space-toets respectievelijk backspace-toets. De grid wordt in een onopvallende kleur getekend op het scherm indien deze aanstaat of verandert van grootte. Controlepunten, ingegeven vóór aanzetten van de grid, moeten aan de grid "snappen" indien de grid wordt aangezet. Deze verandering is definitief (bij uitzetten van de grid gaan de controlepunten niet terug naar hun oorspronkelijke plaats).

De kromme wordt steeds hertekend van zodra een zinvolle input (vanaf graad+1 controlepunten) bekend is. Dit impliceert dat bij verplaatsing van de controlepunten, de kromme direkt opnieuw berekend en getekend wordt. Idem voor verwijderen of toevoegen van controlepunten (zolang er ten minste graad+1 controlepunten zijn). We merken nog op dat graad en aantal iteraties ergens duidelijk op het scherm vermeld moeten staan, en dat deze in eerste instantie een standaardwaarde mogen aannemen. Bij verandering van graad of aantal iteraties wordt de kromme ook hertekend.

Men mag voor alle functionaliteit *eveneens* (dus naast wat hierboven staat) menu's, buttons of textfields voorzien rond het tekenscherm. Dit is echter geen vereiste (Maar: zie sectie Omkadering).

3.3 Het Rationale Lane-Riesenfeld Algoritme

Stel nu het tekenvlak voor als een vlak in de ruimte, waarbij de oorsprong niet in dit tekenvlak ligt. Beschouw de verzameling rechten door twee punten: de oorsprong (van de ruimte) en elk controlepunt. Er zijn bijgevolg evenveel van zulke rechten als er controlepunten zijn. We stellen deze rechten voor door $\langle o, c_i \rangle$. Beschouw nu de verplaatsing van de controlepunten c_i over deze rechten $\langle o, c_i \rangle$ (zie Figuur 3). We herbenoemen deze verplaatste controlepunten c'_i, voor elke c_i. Op deze c'_i kunnen we opnieuw het Lane-Riesenfeld algoritme toepassen (maar dan nu 3-dimensionaal). Door de coördinaten van de

verkregen 3-dimensionale punten van de benaderde kromme (uit het Lane-Riesenfeld algoritme) te homogeniseren verkrijgen we opnieuw punten in het schermvlak. Door deze punten te verbinden zoals voorheen, verkrijgen we een nieuwe kromme, binnen de contouren van de controlepunten.

Voorzie in het programma de benodigde functionaliteit om de effecten van dit verschuifproces te verwezenlijken. Voorzie hiervoor slidebars met exponentiële schaling die de beweging over de respectieve rechten voor controlepunten c_i voorstellen. Er zijn bijgevolg evenveel slidebars nodig als controlepunten. Deze slidebars mogen in een apart venster worden weergegeven of op gelijk welke andere manier die de duidelijkheid van het tekenscherm niet beïnvloedt. De slidebars worden aan- en uitgezet met de "e"-toets. Men *mag* hiervoor weer bijkomend een menu-optie of een button voorzien.

$$tekenKromme(List\ (controlepunt, slidebar_value), int\ graad, int\ aantal_iteraties)$$

Het is niet moeilijk om in te zien dat deze functie niet zal verschillen van de 2-dimensionale versie. We verwachten dus in de uiteindelijke oplossing slechts één versie van deze functie, die beide soorten controlepunten (2D en 3D) kan combineren tot punten van de benaderde kromme. In voorgaande versies mag men uiteraard vertrekken van het 2-dimensionale geval (uit sectie 3.1), naar eigen inzicht.

4 Output

We beschrijven (en herhalen) nu beknopt wat de output op het scherm moet zijn voor de te implementeren functionaliteit.

- Het programma start: Een leeg tekenvenster wordt weergegeven met een aanduiding van graad en aantal iteraties, ingesteld op de standaardwaarden.
- Linker muisklik: Een punt verschijnt op het scherm, op de plaats van de muisklik indien er nog geen punt stond. Indien er meer dan één punt op het scherm getekend is, worden deze verbonden met lijnstukken om de volgorde van de controlepunten aan te geven. Van zodra graad+1 of meer controlepunten op het scherm staan wordt een kromme getekend afhankelijk van graad en aantal iteraties. Indien er reeds een punt stond heeft een consecutieve drag (verplaatsing muis + linker muisknop ingedrukt) de verplaatsing van dit controlepunt en herbereking van kromme en lijnstukken tot gevolg.
- Rechter muisklik: Het laatst getekende punt verdwijnt. Indien er minder dan graad+1 controlepunten zijn, wordt er geen kromme getekend, van zodra er minder dan 2 controlepunten zijn, worden er geen lijnstukken meer getekend. Indien er geen controlepunten meer zijn gebeurt er niets.
- Cursortoetsen: Graad of aantal iteraties verandert (positief, up/right of negatief, down/left, zie Figuur 4)
- "o"-toets: De kromme verandert van/naar gesloten/open formaat
- "e"-toets: De slidebars worden aan/uitgezet. Er zijn evenveel slidebars als er controlepunten zijn. Verplaatsing van de sliders brengt een herberekening van de kromme (als deze er is) teweeg, volgens de uitbreiding zoals in sectie 3.3 beschreven, waarbij de verplaatsing van het controlepunt c_i op rechte $\langle o, c_i \rangle$ relatief en exponentieel geschaald is aan de verplaatsing van de slider. De oorspronkelijke positie van het punt c_i komt overeen met het *midden* van de respectieve slidebar.
- "g"-toets: De snap-to-grid functionaliteit wordt aan/uitgezet. De grid wordt in een onopvallende kleur weergegeven op het scherm, indien deze aanstaat.
- space/backspace-toets: De grid wordt groffer of fijner, en dit wordt aangeduid op het scherm, zelfs indien de snap-to-grid functionaliteit niet aanstaat.

Verdere uitbreidingen zijn vrij, zolang ze het geheel niet verduisteren. De boven besproken functionaliteit en output moet duidelijk aanwezig zijn.

5 Groepen, Formaat van de Oplossing, Tijdsbestek en Forum

Het project wordt gemaakt in groepjes van minimum twee, maximum drie personen. De groepsadministratie gebeurt via *Minerva*. De indeling in groepen wordt bevroren vanaf de eerste dag van de paasvakantie (28 maart 2005). Mensen die zich dan nog niet in een groep hebben ingeschreven worden verondersteld niet deel te nemen aan het project.

De oplossing bedraagt de benodigde source-bestanden in Java om het programma te compileren en te linken onder Java SDK v1.5 (NetBeans IDE 4.0). Deze files mogen van beknopte commentaar voorzien worden. Gezipt als één bestand met naam *GT2005_groepXX.zip* wordt de "dropbox" van *Minerva* gebruikt om dit bestand naar "Benjamin De Leeuw" te zenden. De deadline voor het project is 25 april 2005, 9.00 uur. De "dropbox" wordt dan bevroren. Alle leden van alle groepen die niet voor dit tijdstip een project hebben ingediend op bovenstaande manier, worden verondersteld niet te hebben deelgenomen aan het project. Fraude onder welke vorm dan ook wordt op dezelfde manier gequoteerd.

Er is een forum "Project" voorzien op *Minerva* voor vragen, opmerkingen en discussie.

6 Quotering

Na indiening worden de projecten verbeterd. Hierbij wordt gekeken naar volgende aspecten van de programmacode:

Correct : Een programma is correct indien het Lane-Riesenfeld algoritme correct (zoals in sectie 3.1) wordt toegepast op een arbitraire sequentie controlepunten, graad en aantal iteraties, binnen de beperkingen van het algoritme zelf. Een correct programma zal de kromme, bepaald door de output van dit zelfde Lane-Riesenfeld algoritme (en op geen enkel moment een ander algoritme), correct weergeven op het scherm, zowel voor het gewone, als voor het rationale geval (met slidebars).

Volledig : Indien een correct programma alle functionaliteit implementeert, zoals in dit document beschreven, is het volledig.

Uitbreidbaar : Indien een correct en volledig programma, zo gestructureerd is, dat de code zeer doorzichtig is en voorzien van geconcentreerde commentaar, zodanig dat bvb. toevoegen van nieuwe methodes om krommen te tekenen, of integratie in een ander tekenpakket, eenvoudig is, is dit programma uitbreidbaar.

Vindingrijk : Spitsvondige redeneringen, kernachtige programmaconstructies en krachtige gui's zijn voorbeelden van de eigenschappen van deze ruimere klasse van vindingrijke programma's. Verwar dit in geen geval met een complexe of vergezochte implementatie.

Elk project krijgt op deze manier een voorlopige quotering. Later in het semester zal iedere groep mondeling zijn project moeten demonstreren. Het juiste tijdstip voor de verschillende groepen zal later worden bekendgemaakt. Hierbij kunnen de punten uit de eerste verbetering worden bijgeschaafd, binnen de beperking dat er geen nieuwere versies van de geschreven programma's gebruikt worden op deze presentatie. De lesgever zal de (voordien gezonden) demonstratiecode voorzien. Volgend op deze demonstratie zullen nog een aantal vragen worden gesteld over de achterliggende theorie, en de implementatie. Deze vragen beïnvloeden (tot 3 punten op 20, positief én negatief) de individuele score van de groepsleden. Een mondelinge verdediging zal 10 tot 20 minuten duren. Bij niet opdagen van een groepslid op de mondelinge verdediging, verliest dit groepslid 3 punten op zijn individuele score op 20.

Veel succes!