Trimesteroverschrijdend Project: Curve Editor

Groep 13: Sibrand Staessens en Sibren Polders

Donderdag 22 mei, 2008

Inhoudsopgave

1 Voorwoord

Op welke manieren kan je een set van punten op een gladde wijze met elkaar verbinden? Dat was de vraag achter Curve Editor. Het gemakkelijkste pad dat getekend kan worden tussen de punten is natuurlijk lineair. Maar er zijn nog zoveel andere mogelijkheden, waarvan we er slechts twee hebben verwerkt in curve editor (nl. Bezier en Hermite). Deze algoritmes zullen vlakke kromme maken tussen de interpolatiepunten. De toepassingen van de algoritmes die we gebruikt hebben beperken zich niet enkel tot het tekenen van "lijntjes" tussen punten, maar ook op bijvoorbeeld camerabeweging of "AI"-beweging in games. De interpolatie tussen een gegeven set punten is een uitgebreide en interessante studie die op vele vlakken in de informatica/wiskunde zijn nut kan bewijzen. Curve editor geeft er de basistoepassing (ofwel de toegepaste wiskundige) ervan.

2 Wiskundige Voorkennis

2.1 lineair

2.1.1 Naïeve poging

Een eerste poging om lineaire interpolatie uit te rekenen bestaat er in tussen elk puntenpaar de vergelijking $y = mx + b$ uit te rekenen. Waarbij $m = (y_1 - y_0)/(x_1 - x_0)$ en b het startpunt is op de y-as. De grote nadelen van dit algoritme zijn dat er voor iedere nieuwe pixel opnieuw getekend moet worden en er behoefte is aan een floating point optelling en vermenigvuldiging (vermits m een floating point is.

2.1.2 DDA

Bij deze methode wordt nog steeds m als floating point voorgesteld, maar de berekening van de volgende y-waarde wordt als volgt vereenvoudigd: $y_{i+1} = m.x_{i+1} + b$
$= m.(x_i + \Delta x) + b$
$= m.x_i + m.\Delta x) + b$
$= y_i + m.\Delta x)$
$= y_i + m) aangezien \Delta x = 1 alswe 1 pixel verdergaan.$
Dit algorithme is duidelijk efficienter vermits we een float vermenigvuldiging hebben uitgschakeld.

2.2 Bezier

2.3 Hermite

3 Implementatie

3.1 Packages

3.1.1 Java packages

Een Java package is een mechanisme binnen Java om klassen te organiseren in namespaces. Java broncode die binnen eenzelfde categorie of functie vallen kunnen hierdoor gegroepeerd worden. Dit kan door middel van een package statement bovenaan het beginbestand om aan te geven waartoe ze behoren. Dit is om twee redenen handig omdat de sources gegroepeerd zijn onder hun categorie, wat het geheel overzichtelijker maakt. Verder kunnen er nu twee verschillende klassen eenzelfde naam krijgen en toch uniek bepaald worden door er zijn package name voor te zetten. Wat zeker handig is als de programmeur een klasse dezelfde naam heeft gegeven als een klasse uit een library die hij wil gaan gebruiken.

Voor ons project hebben we een algemeen pakket src gemaakt die verschillende sub-pakketten bevat. Wat duidelijk zichtbaar is in figuur BLAH. In wat volgt wordt een korte beschrijving gegeven van al deze pakketten. Zonder al teveel in te gaan op de technische details.

3.1.2 CurveEditor

Dit is wellicht het kleinste pakket van de reeks (figuur BLAH). Het bevat slechts een klasse, namelijk de main klasse. Deze klasse zal, zoals wellicht duidelijk is, als bootstrap dienen voor de applicatie curve editor. Er wordt ook de mogelijkheid geboden om al rechtstreeks vanuit de commandline een file mee te geven. Dit is enkel ter volledigheid vermits de gebruiker tijdens de loop van het programma zeer makkelijk bestanden kan inladen en opslaan.

3.1.3 Algorithms

Dit pakket voorziet allerlei klassen die voor de interpolatie tussen punten zullen zorgen. Elke klasse in dit pakket implementeert de Algorithme interface. Dit is handig voor groepswerk vermits deze interface vastlegt welke functies de programmeur zal implementeren. Zodoende weet je al op voorhand welke methodes je moet aanroepen om een bepaald resultaat te verkrijgen.
De klassenamen zijn triviaal gekozen: “Lineair, Bezier, BezierC1, BezierG1, Hermite, Hermite Cardinaal, Hermite Catmull Rom”(figuur BLAH). Zoals de namen al veraden zullen deze de verschillende interpolatie methodes die besproken werden in het deel ’Wiskundige voorkennis’ implementeren. Hiervoor werd natuurlijk altijd geoogd op de meeste optimale implementatie van degene die besproken werden.

3.1.4 Core

Dit pakket bevat enkele noodzakelijk klassen(figuur BLAH).

De klasse CurveContainer zal ervoor zorgen dat ingegeven punten kunnen opgeslagen worden, samen met hun door interpolatie berekende punten.

Een eerste idee was het subdivsion principe toe te passen. In de beginsituatie is het tekenveld dan een grote rechthoek. Van zodra de gebruiker een curve begint te tekenen worden de secties waar punten geplaatst zijn onderverdeeld in steeds kleiner wordende rechthoekjes. In elk zo'n rechthoekje zat dan juist een punt van een curve. Zodat er gemakkelijk gezegd kon worden welk punt waar stond en tot welke curve het behoorde.
Dit algorithme bleek echter niet zo efficiënt te zijn wanneer we te maken hadden met een groot aantal input punten. Dit kwam voornamelijk doordat er telkens opnieuw een kleinere rechthoek moest berekend worden bij ingeven van een nieuw punt. En een grotere wanneer er punten verwijderd werden. Uiteindelijk was de applicatie meer bezig met het berekenen van rechthoekjes dan met zijn doel: voorstellen van curves.

De tweede poging leek beter te lukken. We stelden een veld op waarvan elke pixel een mogelijke houder kon zijn van een punt. De houders werden geïnitialiseerd met de waarde null zodat ze geen plaats innamen. Het toevoegen van punten is zo simpel uit te voeren door het new commando toe te passen. Verwijderen is dan gewoon de juist holder op null zetten (de garbage collection van java lost de rest op). Het zoeken gaat simpelweg door de positie van de muisklik om te zetten naar coördinaten die gebruikt kunnen worden op de vector waarin alle punten worden opgeslagen. Om daarna in een bepaalde vooraf bepaalde range te kijken of een punt houder niet op null staat, is dat zo dan zal de informatie van dat punt teruggeven worden, zoniet is er op die plaats in het veld geen punt beschikbaar.

De klasse Editor is het hart van de curve editor. Deze zal zorgen dat de data die uitgewisseld moet worden kan en ook in de juiste richting zal stromen. De uitwisseling van data zal voornamelijk bestaan uit het zoeken of verwijderen van punten uit de CurveContainer klasse. Maar ook het opvangen en afhandelen van excepties. Deze klasse zorgt er dus voor dat de verschillende andere klassen zo autonoom als mogelijk kunnen werken. Dit verhoogd natuurlijk enkel de leesbaarheid en onderhoudbaarheid van de code.

Een laatste klasse van dit pakket is de FileIO klasse, deze zal niet alleen files opslaan en inladen, maar ook zal hij de functionialiteit van undo en redo implementeren, vermits deze van dezelfde functies gebruik maakt.

3.1.5 Curve

Dit pakket bevat de de twee datatypes die doorheen het programma gebruikt worden. Point zoals de naam doet vermoeden geeft de mogelijkheid een punt op te slaan. Curve geeft dan weer de mogelijkheid om een verzameling van punten (lees een kromme of curve) op te slaan. De technische details van deze laatste klasse wordt beter uitgelegd verder in de tekst. Voorlopig is het voldoende om te weten dat Curve een vector van Point's bijhoudt en enkele basisvoorzieningen voorziet (punten opvragen, toevoeg, transleren, ...).

3.1.6 Exceptions

Een foutloos programma schrijven is al een hele opgave, vermits er altijd wel kleine bugs kunnen opduiken na langdurig gebruik. Een fool proof programma schrijven daarentegen is een onmogelijke opgave. Daarom hebben we gebruik gemaakt van exceptions om "verkeerd" gebruik van curve editor op te vangen. Onder verkeerd gebruik valt bijvoorbeeld het inladen van een onbestaande file, of een verkeerd fileformat. Het toevoegen van een punt zonder er de coördinaten van op te geven, ...

Er zijn ook twee HandleException klassen voorzien. Eentje in dit pakket, deze zal gewoon het exceptie bericht in de console uitprinten. In het GUI pakket is een HandleException klasse voorzien die in een dialog scherm de exceptie zal uitprinten.

3.1.7 GUI

De GUI is vrij grondig opgesplitst(zie figuur BLAH). Elk groot deel heeft zo zijn eigen klasse. Zo heb je een menubalk klasse, een snelknopbalk klasse, een tekengebied klasse en een keuzegebied klasse.
De GUI klasse zal op zijn beurt het centrale orgaan spelen die al deze klassen connecteerd. Het is daarom ook niet te verwonderen dat deze is afgeleid van de klasse Editor, die centrale klasse was van de core van de applicatie.
Het opdelen van de GUI in verschillende klassen heeft enorme voordelen als we spreken over leesbaarheid en onderhoudbaarheid van code. Het event-handle systeem van java is zelf optimaal voor dit soort afscheiding. In java worden de events opgevangen door een klasse af te leiden van een van de Listener interface's (ActionListener, ItemListener, ...). Deze klasse zal dan de nodige member functies impementeren om de juiste evenafhandeling te kunnen doen. Het is dus voldoende om deze event handle klassen in GUI aan te maken en door te geven aan de juiste componenten (menubalk, tekengebied, ...). Meer uitleg hierover vind je verder in de tekst.

3.1.8 Tools

Dit bevat een curve simulator. In principe is het een “sjieke” naam voor een bolletje dat over het pad loopt. Dit kan gebruikt worden door de gebruiker om de stijging en daling van de curve te bestuderen. Zodoende kan hij de getekende curve visueel evalueren.

3.2 Uitwerking van de GUI

3.2.1 Overzicht

Zoals in screenshot BLAH te zien is kan de vormgeving van de GUI in 4 grote stukken opgedeeld worden.

1. Menubalk (fig BLAH) Hierin kan de gebruiker elke actie terugvinden die door het programma kan uitgevoerd worden. Bijna elke actie is ook te bereiken met keyboard sneltoetsen (Alt-F, Ctrl-O, ...) zodat elke gebruiker op zijn lievelingsmethode kan navigeren door de verschillende uitvoerbare taken. Er is telkens op gelet dat de nesting van de menu-items niet te diep is noch te breed, verder is elke naam zo triviaal mogelijk gekozen en meestal ook voorzien van een icoon. Dit alles om het gebruiksgemak te verhogen.

2. Snelknopbalk (fig BLAH) Hierin vind je enkele veel gebruikte taken terug (nieuw bestand, een bestand openen, nieuwe curve, ...). Dit opdat de gebruiker deze taken snel en en dus efficiënter kan afhandelen.

3. Keuzegebied (fig BLAH) In dit gebied zal de gebruiker snel kunnen kiezen welk soort curve hij wil tekenen (bezier, hermite, ...) en kan hij ten alle tijden een punt toevoegen door zijn coördinaten in te geven. Verder kan de gebruiker hier ook altijd kiezen of hij de coördinaten, tangens en puntnummers wil laten zien bij het tekenen.
   Het Edit-veld van dit gebied hangt af van de Modus waarin je zit. Er zijn voor de gebruiker twee grote modi zichtbaar. De ene is curve modus, hier bewerk je curves (selecteren, deselectere, nieuwe curve, ...) en de punt modus (punt verslepen, selecteren, toevoegen). Naargelang de modus waarin je zit zulle andere mogelijkheden zichtbaar worden in het edit veld. Dit leidt tot een groter gebruiksgemak vermits de taken die verschijnen meestal ook gebruikt worden waneer je in de modus werkt.

4. Tekengebied (fig BLAH) Dit gebied is waarschijnlijk wel het interessantste voor de gebruiker. Met behulp van muis interactie kan de gebruiker punten aan dit veld toevoegen en deze punten laten connecteren door een interpolatie methode. De gebruiker kan tevens in dit veld punten en curves selecteren, verwijderen en zelf verslepen. Het

tekengebied maakt gebruik van double buffering om beeldflikkering te vermijden en ook van clipping zodat alleen getekend wordt wat ook echt op het scherm verschijnt.

3.3 Java Listeners

Java heeft een aparte stijl om events op te vangen (signalen gestuurd door buttons, menu items, ...). De taal bezit hiervoor zogenaamde event listener interfaces. De programmeur moet een klasse aanmaken die een dergelijke interface zal implementeren. Deze klasse wordt dan m.b.v. een simpel commando geconnecteerd met een bepaald component (bijvoorbeeld een button). In de interface is er een eventhandle functie gedefiniëerd die dan door de programmeur zal geïmplementeerd worden zodat, wanneer het component een signaal uitvoert, de opgevangen event afhandeld zal worden.

Dit systeem is pas echt voordelig als je een event moet afhandelen in een geneste klasse structuur. Stel bijvoorbeeld je hebt in een klasse A een instantie van een Klasse B zitten die op zijn beurt weer een instantie heeft van klasse C en je wilt dan klasse A een handeling doet als klasse C een event uitstuurt(figuure BLAH). Met callbacks is de properste manier (om klasse onafhankelijkheid te bewaren), een callback te connecteren van klasse A naar klasse B, daarna een callback van klasse B naar klasse C. Bij java kan je gewoon een event handle klasse vanuit klasse A meegeven aan B die het op zijn beurt meegeeft aan klasse C. Wat de code natuurlijk wat overzichtelijker maakt.

In deze implementatie is ervoor gekozen om deze eventhandle klassen in de klasse GUI te definiëren. Het voordeel hiervan is dat de enkel de klasse GUI de eventhandle klassen kan instantiëren en deze instanties zulle alle member variabele en functies van GUI kunnen gebruiken. Wat ervoor zorgt dat er geen extra connectie moet gelegd worden tussen de eventhandle klassen en GUI (Een soort van friendly class die niet geïnstantieerd kan worden dus).

3.4 Datastructuren

3.4.1 Point

Deze datastructuur heeft niet zoveel uitleg nodig, een Point bestaat uit zijn x- en y-waarde en de nodige get() en set() functies. Het dient, zoals de naam al doet vermoeden, om een punt in een vlak voor te stellen.

3.4.2 Curve

Deze structuur heeft al iets meer uitleg nodig. Een curve is op het laagste niveau niets anders dan een container van punten.
Elke curve Houdt drie essentiele dingen bij

1. Type (bezier, hermite, ...)
2. een vector van input punten
3. een vector van output punten

Hierbij worden type en de input punten gegeven door de gebruiker. De vector van output punten wordt intern berekend door de verschillende interpolatie algoritmes. Daarna wordt de vector gebruikt om te kunnen tekenen op het tekengebied.

We hebben voor een vector gekozen om de input punten op te slaan omdat de verschillende interpolatietechnieken gemakkelijk moeten kunnen springen tussen de gegeven input punten om hun algoritme te kunnen uitvoeren. Voor de output punten hebben we dan gekozen voor een enkelvoudig gelinkte lijst vermits de punten maar in een richting dienen toegevoegd en gelezen te worden.
Deze klasse bevat ook nog functies om punten te verwijderen, toe te voegen, om na te gaan of een gegeven punt al dan niet tot de curve, ...

3.5 Extra's

TODO

3.6 Interessante problemen

TODO

4 Planning

5 Taakverdeling

1. Sibrand Staessens: Bezier algorithmes, CurveContainer, Editor, DrawArea, GUI
2. Sibren Polders Hermite algorithmes, FileIO, Menu, ToolBar, ChoiceArea, GUI, PathSimulation

6 Appendix

A Handleiding

Had gij dacht ik?

B Screenshot

Gij ook? Alleja ik ga kijken welke gij al hebt en dan bijmaken wat ik nodig heb. Zodoende gaan we geen dubbel werk leveren .... "Stel niet uit tot morgen wat je vandaag door een ander kan laten doen"

C Referenties

Prof. dr. F. VAN REETH, Prof. dr. P. BEKAERT, 2007-2008, Computer Graphics