Geometrisk modellering

Oskar Lappi
Kandidatavhandling
Åbo Akademi
FNT Datateknik
Handledare: Kristian Nybom
2016

Referat
Denna avhandling förklarar en del av den underliggande teknologin och metodologin som tillåter modellering och rendring. Tanken är att ge en översikt av de olika sätten tredimensionella objekt modelleras och manipuleras, samt hur de ritas på en skärm.

Modeller av tredimensionella objekt används inom t.ex. industridesign och arkitektur som basen för verkliga objekt, och inom t.ex. film- och framförallt spelindustrin som basen för grafik som visas på en skärm. Också inom vetenskapen finns behov för att representera tredimensionella datamängder grafiskt.

Inledning

Denna avhandling förklarar en del av den underliggande teknologin och metodologin som tillåter modellering och rendring. Tanken är att ge en översikt av de olika sätten tredimensionella objekt modelleras och manipuleras, samt hur de ritas på en skärm.

Modeller av tredimensionella objekt används inom t.ex. industridesign och arkitektur som basen för verkliga objekt, och inom t.ex. film- och framförallt spelindustrin som basen för grafik som visas på en skärm. Också inom vetenskapen finns behov för att representera tredimensionella datamängder grafiskt.

Avhandlingen börjar med att behandla hur man modellerar formen av objekt, går sedan kort in på vidare modellering av utseende och avslutar med ett kapitel om transformation av objekt. De två olika typerna av modeller som behandlas är polygonytor och spline-baserade objekt.

1. Grundläggande geometri
    1.1. Nät
    1.2. Polygonyta
2. Glatta former
    2.1. Bezier-kurva
        2.1.1. Algebraisk representation
        2.1.2. Komposit kurva
        2.1.3. Geometrisk kontinuitet
        2.1.4. Parametriserad kontinuitet
    2.2. Basis-spline
        2.2.1. Basfunktionen för en B-spline
        2.2.2. Kontinuitet
        2.2.3. Definitionen av en B-spline
        2.2.4. Kontinuitet
    2.3. Jämförelse
    2.4. Bezier-yta
        2.4.1. Algebraisk form
    2.5. B-spline ytor
    2.6. Glatta polygoner
        2.6.1. Catmull-Clark metoden
    2.7. Uträkning av ytnormalen
3. Materialegenskaper
    3.1. Texturer
4. Transformationer
    4.1. Att kombinera transformationsmatriser
    4.2. Skjuvning
    4.3. Translation
    4.4. Förstorning/Förminskning
    4.5. Reflektion
    4.6. Rotation
    4.7. Rymd
    4.8. Perspektivtransformation

1 Grundläggande geometri

1.1.1 Nät

Ett nät beskriver en kropp som en mängd kanter som är förenade via gemensamma punkter. Nät representeras därför naturligt av vektorgrafik. Grafiskt kan saknaden av ytor tydliggöra kroppens konstruktion för en åskådare, eftersom kanterna klart definierar var en yta slutar, och en annan börjar. Det är också mindre arbetsdrygt att enbart rita in kanterna på en skärm.

1.1.2 Polygonyta

En polygonyta beskriver en kropp som en mängd polygoner som är förenade via gemensamma punkter. Detta skiljer sig från ett nät i att elementen är polygoner och inte kanter. Modellen innehåller således information om sidoytorna i en polyeder.

Punkterna som beskriver hörnen sparas som en datamängd, en polygon definieras sedan genom att gruppera punkterna. Detta kan göras genom att indexera punkterna och sedan referera till punkternas index för varje polygon. Vanligtvis används trianglar, eftersom man alltid kan hitta ett gemensamt plan för tre godtyckliga punkter [1]. Som exempel presenteras nedan obj-formatet [2], där varje rad representerar ett element:

```
v 0 0 0
v 100 0 0
v 0 75 0
f 1 2 3
```

Triangeln $\Delta(0,0,0)(100,0,0)(0,75,0)$ representerad i obj-formatet

En punkt, v, definieras med koordinaterna x,y,z. En polygon, f, definieras med parametrarna v_1 v_2...v_n där varje v_i är referensnumret för en punkt som hör till polygonen. Numret representerar inte elementets radnummer, utan elementets ordningstal i mängden av element av den typen började från 1. **f** 1 2 3 är således en triangel med hörnen: 1:a punkten i filen, 2:a punkten i filen och 3:e punkten i filen [2].

En annan metod är att för varje polygon skiljt definiera punkterna. Ett exempel på denna metod är raw-formatet, vilket definierar en polygon (triangel) per rad:

```
0 0 0 100 0 0 0 75 0
```

$\Delta(0,0,0)(100,0,0)(0,75,0)$ representerad i raw-formatet

Hur en polygonyta manipuleras och ritas upp på en datorskärm behandlas i kapitel 3.

2 Glatta former

En polygonyta ser emellertid alltid väldigt kantig ut om den representeras grafiskt. Vad ska man då göra för att modellera ett objekt som ser runt ut? Det finns olika sätt att åstadkomma sådana grafiska resultat. Det går t.ex. att modellera 3D objekt som direkt är glatta. Glatt i detta sammanhang betyder att kurvorna har kontinuerliga derivator upp till en viss gräns. I allmänhet har kontinuiteten av derivator efter andra derivatan inte en

stor betydelse för det glatta utseendet av en kurva eller yta, men formen kan bli mer estetiskt som följd av detta .

I följande paragrafer går jag genom några glatta kurvor och ytor.

2.1 Bezier-kurva

En Bezier-kurva är en glatt kurva som styrs av kontrollpunkter. Eftersom kurvan styrs av punkter går det att implementera ett grafiskt användarsnitt för att flytta på punkterna med t.ex. en mus. Kurvan kan också intuitivt tolkas av en människa som ett rep som dras av en kraft mot kontrollpunkterna. På grund av dessa egenskaper används Bezier-kurvor flitigt i diverse formgivningsindustrier[4],[6]. Kontrollpunkterna kan vara två- eller tredimensionella. Punkterna representeras i resten av detta kapitel som positionsvektorer för att tydliggöra aritmetiska operationer, och för enkelhetens skull förkortas begreppet “punktens positionsvektor” till “punkten” när tydligheten inte lider av det.

En Bezier-kurva av grad n är en interpolerande funktion på intervallet $t \in [0,1]$ definierad av n+1 stycken kontrollpunkter $\mathbf{b}_i$. Vi definierar Bezier-kurvan rekursivt från dessa kontrollpunkter: m punkter $\mathbf{b}_i^0$ bildar m-1 sträck som sedan interpoleras vid **t** för att få n
m-1 punkter $\mathbf{b}_i^1$. När det endast finns en punkt kvar är detta värdet vid **t**.

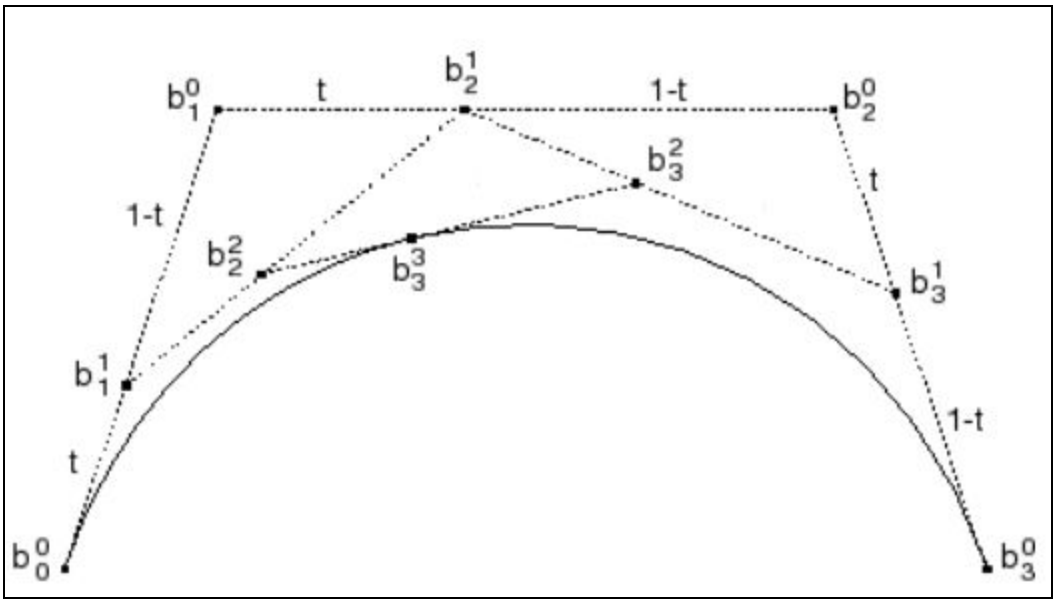

Fig. 1: En kubisk Bezier-kurva, alla sträckor är delade i förhållandet $\frac{t}{1-t}$ [6]

Detta skapar en jämnt böjd kurva som ändå följer punkternas konfiguration. Kurvan ligger alltid inom den polygon som punkterna utgör. En annan egenskap hos kurvan är att den börjar i den första kontrollpunkten och slutar i den sista. Ytterligare, eftersom kurvan är en funktion på t, behåller kurvan sin form om punkterna roteras eller reflekteras genom en linje.

2.1.2 Algebraisk representation - Bernsteinpolynom

Bezier-kurvor representeras algebraiskt av Bernsteinpolynom. Det Bernsteinska baspolynomet har formen:

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} \qquad (1)$$

Ett Bernsteinpolynom är en linjärkombination av baspolynom, koefficienterna $\mathbf{b}_i$ kallas Bernsteinkoefficienter. Ett Bernsteinpolynom har formen:

$$B(t) = \sum_{i=0}^{n} \mathbf{b}_i B_{i,n}(t) \qquad (2)$$

För en Bezier-kurva $\mathbf{r}(t)$ är koefficienterna kontrollpunkternas positionsvektorer, och som tidigare nämnts är definitionsmängden för en Bezier-kurva $t \in [0,1]$. Således representeras Bezier-kurvan algebraiskt som så:

$$\mathbf{r}(t) = \sum_{i=o}^{n} B_{i,n}(t), \quad 0 \le t \le 1 \qquad (3)$$

Från detta kan man tyda att varje kontrollpunkt påverkar hela kurvan. Undantaget är punkterna $\mathbf{r}(0)$ och $\mathbf{r}(1)$ var termerna t^i och $(1-t)^{n-i}$ endast är nollskilda vid punkterna i=0 och i=n. Detta bekräftar det tidigare påståendet om att ändpunkterna för kurvan är första och sista kontrollpunkten.

Här nedan representeras baspolynomen grafiskt. Polynomen $B_{i,n}$ bestämmer kontrollpunkternas $\mathbf{b}_i$ vikt í summan $\mathbf{r}(t)$.

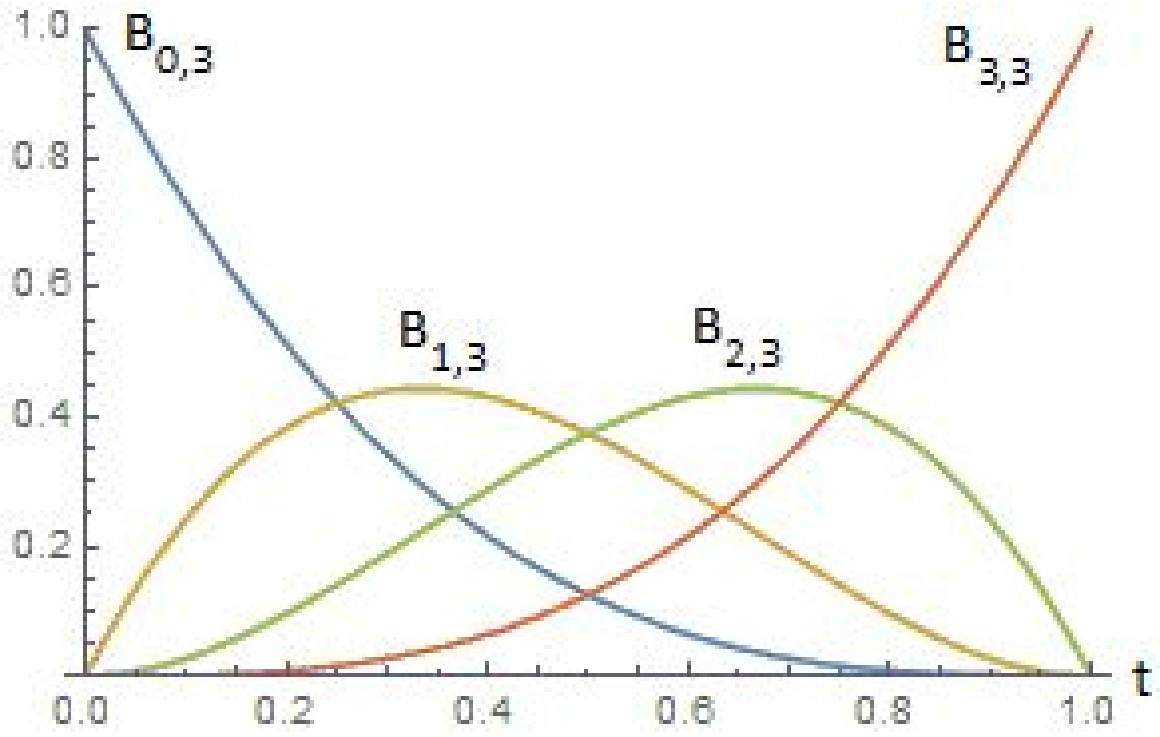

Fig. 2: Graf av Baspolynomen för en kubisk Bezier-kurva

2.1.3 Komposit kurva

En komposit Bezier-kurva består av två eller fler Bezier-kurvor sammansatta vid deras ändpunkter. På så sätt åstadkoms en mer grafiskt komplex kurva. En mer komplex kurva kan också åstadkommas genom att höja graden av en Bezier-kurva, men så kontrollpunkterna växer i antal blir de lätt otympliga att hantera, dessutom ökar komplexiteten (kostnaden) för varje kontrollpunkt. Om tillvägagångssättet då är att definiera skilda Bezier-kurvor som kopplas samman, måste vissa egenskaper av den sammansatta kurvan vid kopplingspunkterna tas i beaktan.

2.1.4 Geometrisk kontinuitet

Geometrisk kontinuitet åstadkoms genom att låta varandra efterföljande kurvors sista respektive första kontrollpunkter vara samma punkt. Denna egenskap kallas G^0.

G^0:

$$\mathbf{v}_n = \mathbf{w}_0 \tag{4}$$

Eftersom målet var jämn böjning är en kontinuerlig tangent till kurvan också önskvärd. En kontinuerlig tangent kräver att de två kontrollpunkterna runt en kopplingspunkt, $\mathbf{v}_{n-1}$ och $\mathbf{w}_1$, och kopplingspunkten, $\mathbf{v}_n = \mathbf{w}_0$, själv alla ligger på samma linje. Denna egenskap kallas G^1.

G^1:

$$a(\mathbf{v}_n - \mathbf{v}_{n-1}) = \mathbf{w}_1 - \mathbf{w}_0 \qquad (5)$$

Var det upphöjda indexet står för segmentets index, a är förhållandet mellan vektorernas längd och $\mathbf{v}_i, \mathbf{w}_i$ är punkter som tillhör de två segmenten.
För en riktigt jämn estetisk böjning krävs även att krökningen är kontinuerlig över hela kurvan [6]. Denna egenskap kallas G^2. Bilderna nedan förklarar geometriskt de villkoren som måste uppfyllas för G^2.

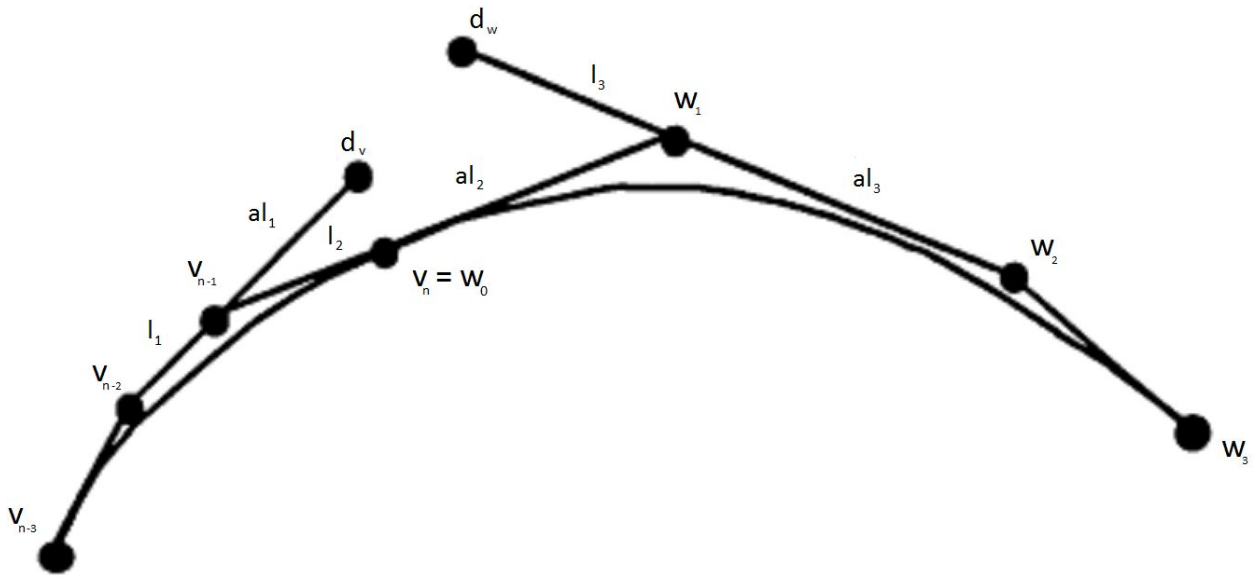

Fig. 3: En Bezier-kurva som inte uppfyller G^2

I bilden har två kubiska Bezier-kurvor fogats samman vid v_n. Två nya punkter d_v och d_w har införts, som ligger i ändan av förlängningar av vektorerna som bildas mellan de två kontrollpunkterna som ligger näst intill fogningspunkten. Punkterna är dessutom placerade så att punkterna $\mathbf{v}_{n-1}$ och $\mathbf{w}_1$ delar förlängda vektorn i samma förhållande som fogningspunkten delar vektorn från $\mathbf{v}_{n-1}$ till $\mathbf{w}_1$.För att uppfylla G^2 bör d_v och d_w vara samma punkt.

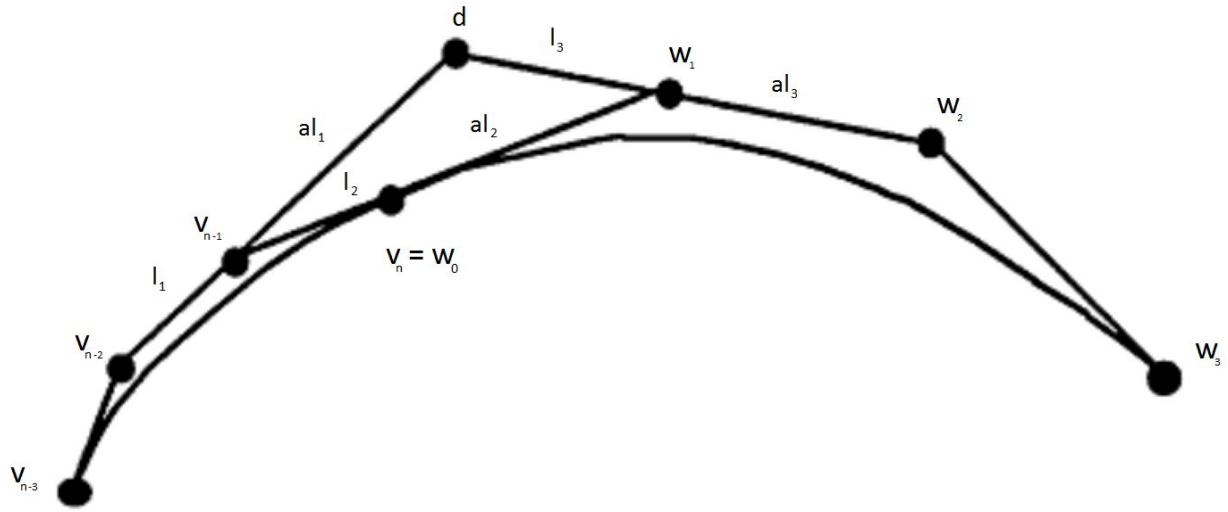

Fig. 4: En Bezier-kurva som uppfyller G^2

I denna bild ligger d_v och d_w i samma punkt. Kurvan uppfyller därmed G_2 [7]. Sambanden i bilden kan skrivas som:

$$\mathbf{w}_1-(\mathbf{w}_2-\mathbf{w}_1)\frac{v_n-v_{n-1}}{w_1-w_0} = \mathbf{v}_{n-1}+(\mathbf{v}_{n-1}-\mathbf{v}_{n-2})\frac{w_1-w_0}{v_n-v_{n-1}} \quad (6)$$

insättning av $a = \frac{w_1-w_0}{v_n-v_{n-1}}$ ger:

$$\mathbf{w}_1-(\mathbf{w}_2-\mathbf{w}_1) / a = \mathbf{v}_{n-1}+(\mathbf{v}_{n-1}-\mathbf{v}_{n-2})a$$

$$\Rightarrow \quad \mathbf{w}_2 = \mathbf{w}_1(a+1) - \mathbf{v}_{n-1}a - (\mathbf{v}_{n-1} - \mathbf{v}_{n-2})a^2 . \quad (7)$$

Vilket är villkoret för G^2.

2.1.5 Parametriserad kontinuitet

Parametriserad kontinuitet av graden n, C^n betyder att funktionen har en kontinuerlig n:te derivata. Detta skiljer sig från geometrisk kontinuitet G^n som beskrevs i förra paragrafen, vilken endast bryr sig om en kontinuerlig tangent. Eftersom det är en fråga om en funktion på t är derivatan inte densamma som tangenten. Därmed är C^n ett mer strikt villkor som kräver kontinuerlig förändringshastighet med avseende på t [6]. Om man låter a=1 för något G^n får man villkoret för C^n.

C^0:

$$\mathbf{v}_n = \mathbf{w}_0 \quad (8)$$

C^1:

$$\mathbf{v}_n - \mathbf{v}_{n-1} = \mathbf{w}_1 - \mathbf{w}_0 \quad (9)$$

C^2:

$$\mathbf{v}_n - 2\mathbf{v}_{n-1} + \mathbf{v}_{n-2} = \mathbf{w}_2 - 2\mathbf{w}_1 + \mathbf{w}_0 \quad (10)$$

Om man antar att C^0 redan är tillfredsställt förenklas C^2 till:

$$\mathbf{v}_{n-2} - 2\mathbf{v}_{n-1} = \mathbf{w}_2 - 2\mathbf{w}_1$$

För båda sortens kontinuitet begränsas valet av punkter av villkoren G^0,G^1,G^2 respektive C^0,C^1,C^2 vid varje fogningspunkt. Detta går naturligtvis att automatisera. [7] beskriver ett sätt att hitta kontinuitet av andra graden. Rapporten påstår att metoden hittar en konfiguration som uppfyller C^2, men det är frågan om G^2.

Den flexibla metoden som har blivit standard är att använda sig av styckvis definierade kubiska Bezier-kurvor. Detta för att en kubisk Bezier kan innehålla en topp och en dal eftersom den är ett tredjegrads polynom, vilket gör den mer användbar än en kvadratisk Bezier, men inte för komplicerad eller ineffektiv[4].

Ett problem med Bezier-kurvor är att de saknar vad som kallas lokal kontroll. Som jag tidigare konstaterat påverkas hela kurvan av varje punkt. Detta försvårar formgivningsprocessen eftersom man inte kan fokusera på en del av en kurva runt en kontrollpunkt utan att tänka på vad som händer med resten av formen.

2.2 Basis-spline

En Basis-spline, eller B-spline, är precis som en Bezier-kurva styrd av en mängd kontrollpunkter. Men till skillnad från Bezier begränsar en B-spline vilka kontrollpunkter som styr området mellan segment av kurvan. Segmenten skiljs åt av knutar som definieras i en knutvektor $\mathbf{T} = (t_0 \ldots, t_{n+k+1})$. Termen knutvektor är vilseledande eftersom det egentligen handlar om en mängd. En knut är ett tal t_i som har samma definitionsmängd som den oberoende variabeln t. Baspolynomet ändras beroende på vilket intervall $[t_i, t_{i+1}[$ som t ligger i.

2.2.1 Basfunktionen för en B-spline

Basfunktionen för en B-spline är $N_{i,k}(t)$ — var k är graden av funktionen och i = 0…,n — som definieras rekursivt av:

$$N_{i,0}(t) = \begin{cases} 1, \text{ då } t_i < t < t_{i+1} \\ 0, \text{ annars} \end{cases}$$

och (11)

$$N_{i,k}(t) = \frac{t-t_i}{t_{i+k}-t_i} N_{i,k-1}(t) + \frac{t_{i+k+1}-t}{t_{i+k+1}-t_{i+1}} N_{i+1,k-1}(t)$$

Detta kan se skrämmande ut men är egentligen väldigt simpelt.

$N_{i,k}$ är nollskild i intervallet $[t_i,t_{i+k}[$. $\frac{t-t_i}{t_{i+k}-t_i}$ är förhållandet mellan avståndet av t till **vänstra** sida av intervallet och hela intervallets belopp, eller till hur stor del t har gått över detta intervall. Vikten av $N_{i,k-1}$ **ökar** alltså linjärt mot högra sidan av intervallet.

$N_{i+1,k}$ är nollskild i intervallet $[t_{i+1},t_{i+k+1}[$ och $\frac{t_{i+k+1}-t}{t_{i+k+1}-t_{i+1}}$ är förhållandet mellan avståndet av t till **högra** sidan av intervallet och hela intervallets belopp, eller hur långt t har kvar att gå. Vikten av $N_{i,k-1}$ **minskar** alltså linjärt mot högra sidan av intervallet.

Varje skilt polynom $N_{i,k}$ ritar då upp en kurva i intervallet för definitionsmängden av t. Precis som för Bezier-kurvor brukar intervallet [0,1] användas. Kurvan beror på knutarna t_i-t_{i+k+1} och är alltså nollskild i detta intervall. Alternativt, om man betecknar ett segment $s_i = [t_i,t_{i+1}[$ kan man säga att $N_{i,k}$ är nollskilt i knutsegmenten s_i-s_{i+k}. Det är denna delvis nollskilda värdemängden som gör att B-splines tillåter lokal kontroll över kurvan.

2.2.2 Definitionen av en B-spline

En B-spline definieras algebraiskt av en linjärkombination av basfunktioner:

$$\mathbf{r}(t) = \sum_{i=0}^{n-1} \mathbf{p}_i N_{i,k} \quad (12)$$

Genom att definiera n knutar skapas n-k segment s_i inom vilka korresponderande basfunktioner $N_{i,k}$-$N_{i+k,k}$ är nollskilda. Detta leder till att varje segment kontrolleras av k+1 kontrollpunkter (och varje kontrollpunkt kontrollerar k+1 segment). Nu kan en formgivare flytta på en kontrollpunkt och försäkra sig om att högst k+1 segment ändrar på sig. Med andra ord existerar lokal kontroll.

Ett problem som uppstår är att en fin egenskap som existerade hos Bezier-kurvan inte återfinns, nämligen att kurvan börjar och slutar i första och sista kontrollpunkterna. Detta

åtgärdas genom att definiera k+1 stycken likvärdiga knutar i definitionsmängdens extremer: $t_0 \ldots = t_i \ldots = t_k = 0$ och $t_n \ldots = t_j \ldots = t_{n+k+1} = 1$. På så sätt är extremiteterna kontrollerade av endast en punkt. Dessa knutar kallas extrema knutar, och resten kallas interna knutar. Värt att notera är att om endast externa knutar definieras är en B-spline ekvivalent med en Bezier-kurva med samma kontrollpunkter.

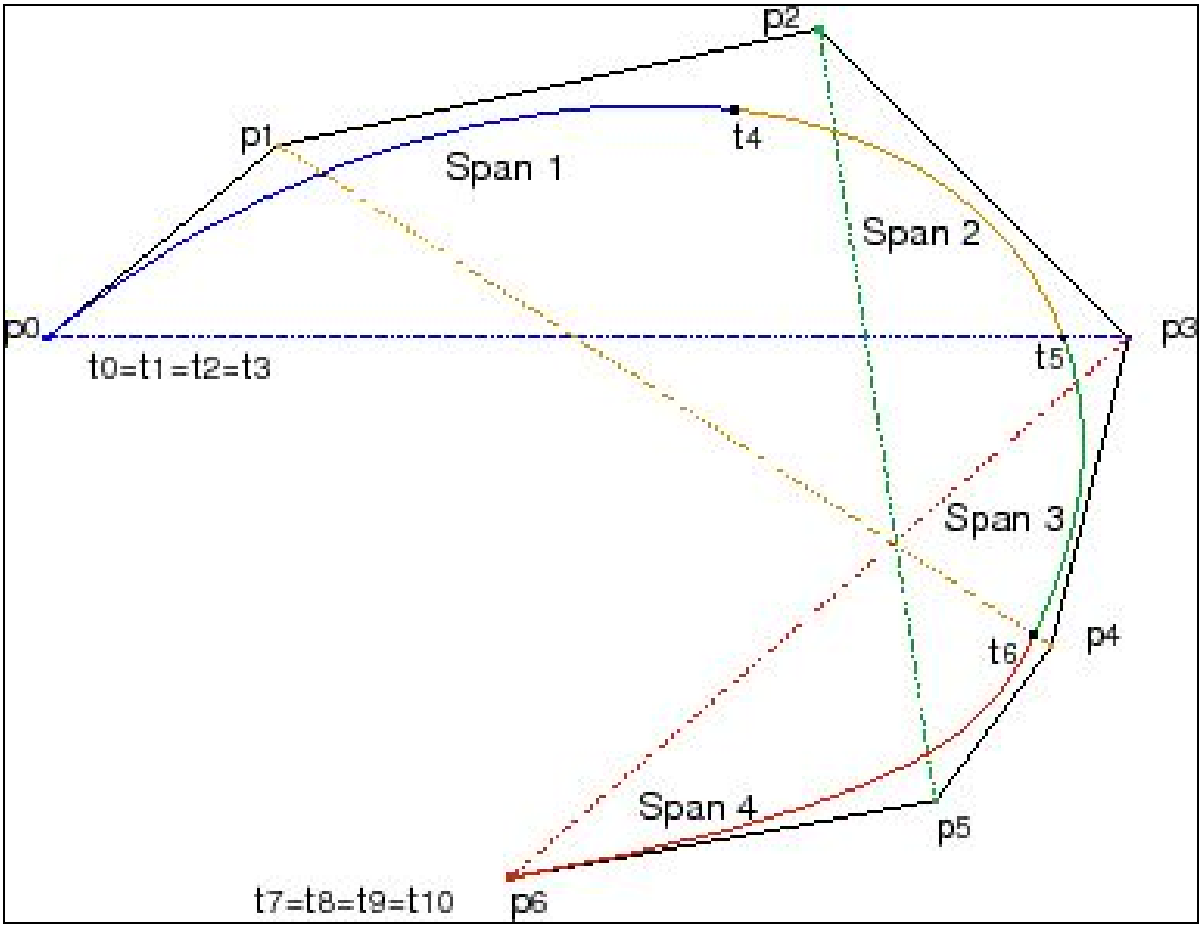

Fig. 5 En kubisk B-spline med segmenten och motsvarande kontrollpolygoner kopplade ihop med färg

På samma sätt som med Bezier-kurvan kan baspolynomen representeras grafiskt för att se till vilken grad varje kontrollpunkt styr kurvan i en viss punkt t.

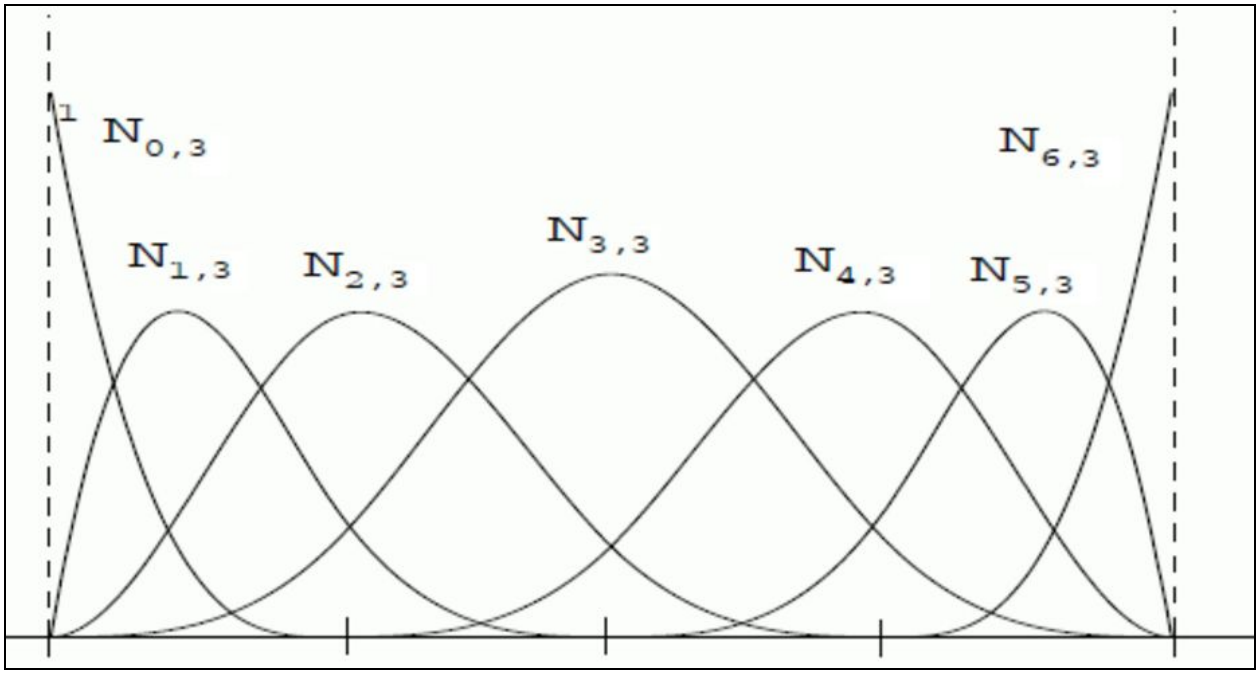

Fig. 6 basfunktionerna för en B-spline utritade.

2.2.3 Kontinuitet

En B-spline har C^{k-1} kontinuitet vid knutarna, och fullständig kontinuitet inom varje segment. Detta bryts om likvärdiga interna knutar introduceras, varje ny likvärdiga knut minskar kontinuiteten i den punkten med 1.

2.3 Jämförelse

B-spline kurvan har exakt samma egenskaper som Bezier, och som redan snabbt berördes är en Bezier-kurva ett specialfall av en B-spline. B-splines introducerar lokal kontroll och kräver färre kontrollpunkter än Bezier-kurvor. Kontinuitet är också trivialt för B-splines, medan det krävs en betydlig mängd arbete för att försäkra C^2 eller G^2 för Bezier-kurvor, vilket dessutom äter upp frihetsgrader. En ytterligare fördel som B-splines har är att kurvans grad inte är bunden i mängden kontrollpunkter. Den enda fördelen Bezier-kurvor har är en förhållandevis enklare definition, och därmed kanske enklare implementation.

Det går att ytterligare generalisera B-splines till NURBS, som inför en vikt för varje kontrollpunkt. Detta görs med hjälp av homogena punkter, som ytligt behandlas i kapitlet om transformationer senare.

2.4 Bezier-yta

Om man vill modellera en glatt yta då?

På samma sätt som en Bezier-kurva beror på n kontrollpunkter $\mathbf{b}_i$, beror en Bezier-yta på m*n kontrollpunkter $\mathbf{b}_{ij}$ i en topologiskt rektangulär konfiguration. Man arbetar även med två Bezierpolynom $B_{i,m}(u)$ och $B_{j,n}(v)$. För ett unikt värde på både u och v hittar vi punkten på kurvan genom att för varje skiljt värde på i definiera en Bezier-kurva bestående av n kontrollpunkter $\mathbf{b}_{i0}\ldots,\mathbf{b}_{in}$. Sedan evaluerar vi denna kurva $\mathbf{r}_i(t)$ i t=v. Värdet av $\mathbf{r}_i(v)$ definierar vi som en ny kontrollpunkt $\mathbf{c}_i$. Dessa nya punkter $\mathbf{c}_i$ definierar en ny Bezier-kurva som evalueras i t = v. Denna punkt är värdet av **r**(u,v).

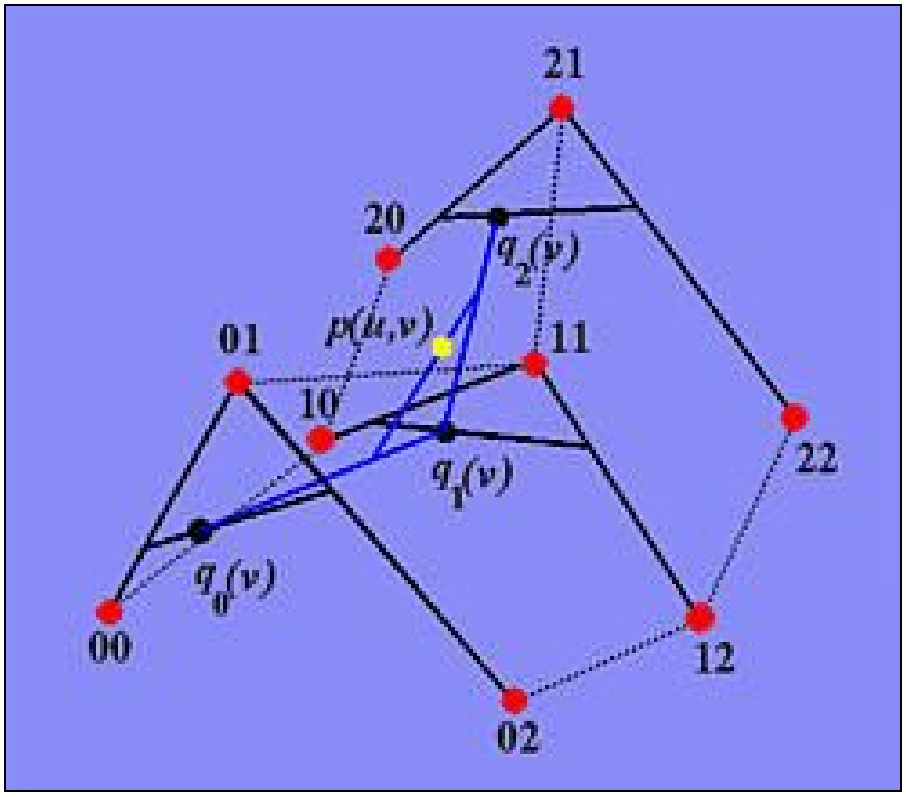

Fig. 7: De svarta Bezier-kurvorna evalueras i en punkt t=v, dessa punkter formar den blåa Bezier-kurvan som evalueras i u, vilket ger den slutgiltiga gula punkten.

2.4.1 Algebraisk form

Den algebraiska formen av en Bezier-yta är en kartesisk produkt av två mängder Bernsteinska baspolynom, enligt detta:

$$\mathbf{r}(u,v) = \sum_{i=o}^{m}\sum_{j=0}^{n} \mathbf{b}_{ij}B_{i,m}(u)B_{j,n}(v). \qquad (13)$$

Vi kan bryta ut $\mathbf{B}_{i,m}(u)$ från den inre summan och få:

$$\mathbf{r}(u,v) = \sum_{i=o}^{m} B_{i,m}(u)\left(\sum_{j=0}^{n} \mathbf{b}_{ij} B_{j,n}(v)\right). \quad (14)$$

Vilket överensstämmer med den intuitiva förklaringen, eftersom uttrycket $\sum_{j=0}^{n} \mathbf{b}_{ij} B_{j,n}(v)$ bildar en ny mängd punkter, punkterna $\mathbf{c}_i$ i förra stycket [6].

2.5 B-spline ytor

Analogt till Bezier-ytor beror en B-spline yta på m*n kontrollpunkter $\mathbf{p}_{ij}$ i en topologiskt rektangulär konfiguration. Det behövs även två knutvektorer $\mathbf{U} = (u_0,...,u_{m+k+1})$ och $\mathbf{V} = (v_0,...,v_{n+l+1})$. B-spline ytan fås av:

$$\mathbf{r}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{p}_{ij} N_{i,k}(u) N_{j,l}(v). \quad (15)$$

Samma strategi för att reda ut en punkt i $u=u_0$ och $v=v_0$ återanvänds. En ny mängd punkter $\mathbf{q}_j$ genereras av $\mathbf{q}_j = \sum_{i=0}^{m} \mathbf{p}_{ij} N_{i,m}(u_0)$, som fungerar som kontrollpunkter för en B-spline kurva. Varje sådan B-spline kurva evalueras i $v=v_0$ för att generera ytan.

2.6 Glatta polygonytor

Det är även möjligt att få en polygonyta att verka glatt. Nedan presenteras en metod i korthet som går ut på att dela upp polygonytan i mindre delar som närmar sig en glatt yta.

2.6.1 Catmull-Clark metoden

Catmull-Clark metoden ([3], [5]) skapar nya, mindre polygoner baserat på polygonen och närvarande polygoner. För varje yta, definiera en yt-punkt y som medelvärdet av alla ytans punkter. För varje kant, definiera en kant-punkt k som medelvärdet av kantens ändpunkter och punkterna som definierades i förra steget för de två ytor som kanten hör till. Flytta varje punkt p som fanns i början av steget till:

$$\frac{Y+2K+(n-3)p}{n}\ . \quad (16)$$

Där Y är medelvärdet av yt-punkterna som hör till ytor som p tillhör, K är medelvärdet av kant-punkterna som hör till kanter som p tillhör och n är antalet ytor/kanter som p tillhör [5]. Steget upprepas med de nya punkterna tills polygonerna är så pass små att fortsatt uppdelning inte ändrar slutresultatet. Metoden har slipats av och an, den har till exempel gjorts icke-rekursiv [9] och minneseffektivare [10] med mera. Frekvensen av

vetenskapliga artiklar som ännu hänvisar till denna metod visar att den ännu är relevant mer än 40 år efter sin uppkomst.

Problemet med denna metod är att information om närliggande polygoner behövs. Metoder som endast använder sig av information om kanterna som tillhör en polygon existerar, till exempel [8]. Dessa kräver dock nån information av ytans normal i hörnen. Dessa kan antingen definieras av en formgivare, eller räknas ut som ett medeltal av normalerna för varje yta som punkten tillhör.

2.7 Uträkning av ytnormalen

Om vi har två vektorer **u** och **v** hittar vi en normal **n** till planet de spänner upp genom att använda kryssprodukten **u** x **v,** som är ortogonal till både **u** och **v**. Varje plan har två antiparallella ytnormaler. Kryssprodukten hittar den högerhänta normalen. Det vill säga om vi tänker oss att tummen är **u**, pekfingret **v**, så pekar långfingret i kryssproduktens riktning om vi böjer in den mot handflatan. För att hitta ytnormalen till en triangel väljer vi en punkt och definierar **u** och **v** som vektorer från den punkten till de övriga två punkterna. Det är här som valet av riktning kommer in. Om vektorerna **u** och **v** är mellan första och andra samt första och tredje punkten, ger kryssprodukten för medurs-ordnade punkter en vektor som pekar bort från betraktaren.

Riktningen av ytnormalen används för att definiera den synliga sidan av polygonen. Den sida av polygonen som är “bakom” vektorn ritas alltså inte. För att underlätta uträkningar av skalärprodukter dividerar vi ännu ytnormalen med dess längd för att få en enhetsvektor [4].

Som redan tidigare nämndes kan ytnormalen vid ett hörn räknas ut som ett medeltal av alla närliggande ytors normaler. Denna hörnnormal (eng. *vertex normal*) brukar också representeras som en enhetsvektor. I sådant fall kallas den en normaliserad hörnvektor (eng. *normalised vertex normal*).

3. Materialegenskaper

Fysiska objekts utseende baserar sig på ett antal materialegenskaper. En kopparskiva är kopparfärgad och glansig, men en tegelvägg är röd och matt. Tegelstenar är ganska släta, men de har vanligtvis inte en helt enhetlig färg, och om vi tittar noga kan vi se

porer, förhöjningar och skuggor skapade av dessa. Hur ska detta modelleras? Följande är en beskrivning av ett urval olika metoder för att modellera material.

3.1 Texturer

Ofta vill man inte endast modellera ett objekts geometriska form, utan också dess utseende. Ett sätt att göra detta är med texturer. Texturer modellerar det helhetsintryck man får av en ytas material i verkligheten när det kommer till färg och mönster. I praktiken betyder detta att man spänner en bild som ett skinn på en yta. Bilden som används kan vara av olika format men för enkelhetens skull presenteras här fallet med en bitmap.

En bitmap är en matris av färginformation. För att spänna en bitmap på en yta definierar man korresponderande punkter (matrisindex) på bilden till ytans hörn. Man fäster en punkt på bitmappen till varje hörn i polygonen och interpolerar sedan mellan punkterna på båda ytorna och använder för var punkt inom polygonen den motsvarande punktens färg i bitmappen. Metoden kallas texture mapping [1].

Exempel:
I obj-formatet kan en textur kopplas till en yta med hjälp av texturpunkter (texture vertices):

```
...
v 0 0 0
v 100 0 0
v 0 75 0
vt 0 0
vt 0 1
vt 1 0
f 1/1 2/3 3/2
...
```

Oftast motsvarar inte punkten på polygonen exakt en punkt på bitmappen. Detta kan lösas t.ex genom att välja färgen av den närmaste punkten (nearest neighbour) eller att interpolera mellan de fyra punkter på bitmappen punkten ligger mellan.

4. Transformationer

När man väl definierat ett geometriskt objekt, vill man antagligen betrakta objektet. Men från vilken vinkel? I vilken storlek? Var på skärmen? Det som behövs är alltså en avbildning av punkterna i modellen till punkter på skärmen. En sådan avbildning kallas för en transformation. De olika sätten en punkt kan transformeras är: translation, rotation, reflektion, skjuvning och förminskning/förstorning. För en tydligare intuitiv förklaring betraktas problemet i två dimensioner.

En punkt (x,y) ska förändras till en ny punkt (x',y')

$$\begin{cases} x' = ax + by \\ y' = cx + dy \end{cases} \tag{17}$$

Kutymen är att representera transformationer i matrisform, eftersom de enkelt går att kombinera genom att multiplicera matriserna, och är effektivare för datorer.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{18}$$

Emellertid går det inte att utföra en translation på detta sätt. En translation är en transformation av typen

$$\begin{cases} x' = x + t_x \\ y' = y + t_y \end{cases} \tag{19}$$

För att kunna uttrycka alla transformationer borde den allmänna formen därför se ut som så:

$$\begin{cases} x' = ax + by + t_x \\ y' = cx + dy + t_y \end{cases} \tag{20}$$

Lösningen är att arbeta med en homogen punkt. För varje punkt (x,y) finns ett oändligt antal homogena punkter (x',y',t) = (xt,yt,t). Dessa punkter ligger på en linje som går genom origo och (x,y,1) i ett xyt-system. Genom att välja samma $t = t_0$ för varje punkt

ligger alla punkter i ett plan som är parallellt med x- och y-axlarna vid djupet t_0. För enkelhetens skull väljs $t_0 = 1$.

Homogena punkter kan ritas upp på ett xy-koordinatsystem som en ortogonal projektion på xy-planet. t's geometriska betydelse i detta fall är en skalningsfaktor som skalar avståndet till origo, eftersom punkten (x,y) förvandlas till (x',y') = (xt,yt).

Resulterande punkten (x',y',t') ska ligga vid samma djup, det vill säga t'=t=1. För att åstadkomma detta är sista raden i transformationsmatrisen [0 0 1] eftersom
$0 \cdot x + 0 \cdot y + 1 \cdot t = t$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{21}$$

Vilket är ekvivalent med (20). För att utföra transformationer på punkter i tre dimensioner används en tredimensionell homogen punkt och en 4x4 transformationsmatris

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{22}$$

4.1 Att kombinera transformationsmatriser

Säg att två transformationsmatriser T_1 och T_2 måste appliceras på en punkt P i följd. Den första tranformationen ger ett mellansvar $P'=T_1P$ och efter en till transformation blir den nya punkten $P_{ny}=T_2P'$. Eller, uttryckt i ett enda steg: $P_{ny}=T_2T_1P$. Vi inser nu att den fullständiga transformationen har formen $T=T_2T_1$. En serie transformationsmatriser kan alltså kombineras till en matris genom att vänstermultipliceras i stigande ordning.

Transformationerna förklaras algebraiskt utan matriser nedan. Resultaten härleds inte utan presenteras endast.

4.2 Skjuvning

En skjuvning är en transformation som förskjuter punkter i riktningen av en axel proportionellt till deras position på de övriga axlarna. Exempelvis x'=x+ay+bz.

4.3 Translation

En translation är en transformation som förskjuter punkter med samma vektor oberoende av var de ligger. Om vi noggrant granskar hur translation representeras i transformationsmatrisen ser vi att $x'=x+t_x*t$, förskjutningen är alltså beroende av t (även om t=1). Den geometriska betydelsen av translationsmatrisen är alltså en skjuvning som förskjuter det 3-dimensionella tvärsnittet vid t=1 av x,y,z,t-rummet (Detta tvärsnitt är detsamma som det rum vi arbetar i). Eftersom t=1 är förskjutningen lika stor som skjuvningsfaktorn.

4.4 Förstoring/förminskning

En förstoring eller förminskning är en transformation som ändrar på längdskalorna i en av riktningarna, exempelvis x'=a*x. Förstoringen är centrerad i origo, vilket innebär att en kropp med punkter endast utanför origo kommer att röra sig bort från origo. Men, eftersom transformationer går att kombinera är det möjligt att centrera förstoringen i vilken punkt som helst genom att först applicera en translation på punkterna till valfri position relativt till origo.

4.5 Reflektion

Reflektion är en transformation som förskjuter punkter genom en linje så att avståndet till linjen bevaras.

4.6 Rotation

En rotation runt en kardinal-axel med vinkeln α, ges av

$$\begin{cases} (x'|y'|z') = (x|y|z) * cos(\alpha) - (y|z|x) * sin(\alpha) \\ (y'|z'|x') = (x|y|z) * sin(\alpha) + (y|z|x) * cos(\alpha) \\ (z'|x'|y') = (z|x|y) \end{cases}.$$

(23)

För att rotera en punkt runt en godtycklig axel kan vi först utföra två rotationer runt y- och z-axlarna som orienterar x-axeln i rotationsaxelns riktning. Punkten roteras nu runt

x-axeln med den valda vinkeln och sedan tillbaka med motsatt vinkel runt z- och y-axlarna i den ordningen.

Det är förstås möjligt att applicera en translation först om rotationsaxeln inte går genom origo, därvid motsatta translationen ännu utförs som sista operation.

En annan strategi är att använda kvaternioner.

4.7 Rymd

Det finns etablerad terminologi som beskriver de olika referenssystem som används baserat på var de är centrerade.

-Objektrymd är ett referenssystem lokalt till objektet punkterna tillhör. Origo för detta system representerar objektets rotationscentrum. Punkterna i en fil som beskriver objektet existerar i detta rum.

-Globalrymd är det system som beskriver varje objekt som tillhör en scen. Scen är begreppet som används för att beskriva en mängd objekt i en viss rymdkonfiguration.

-Lokalrymd för något specifikt objekt är globalrymden centrerat runt ett objekts rotationscentrum.

-Kamerarymd är ett lokalt rum centrerat på kameran i kamerans riktning.

-Bildrymd är det system som beskriver var punkter ligger på en skärm.

Med hjälp av transformationerna som beskrivits kan objekt föras från objektrymd till globalrymd, och från globalrymd till kamerarymd. Men för att översätta kamerarymden till bildrymd behövs något som ännu inte behandlats.

4.8 Perspektivtransformation

För att kunna representera vår modell, som existerar i ett 3-dimensionellt utrymme, på vår datorskärm måste vi projicera den i ett plan.

En ortogonal projektion ger dock inte ett resultat som liknar hur en människa ser 3d-objekt, vi har en känsla för djup. För att representera detta i vår grafik måste vi utföra en perspektivtransformation på alla punkter i vår modell.

5. Avslutning

Det finns många olika metoder att modellera geometriska objekt, av vilka endast en bråkdel har presenterats i denna avhandling. Alla har den gemensamma egenskapen av att vara representerade av punkter. Samma teori kan därmed användas på alla för att transformera dem, en sådan teori presenterades i kapitel 4. Något som inte behandlats är den geometriska algebran, som utgör ett annat tillvägagångssätt till den linjär-algebraiska matrisrepresentationen av transformationer som presenterats i avhandlingen.

Avhandlingen har behandlat hur olika geometriska kroppar och deras utseende representeras av data, och hur detta förhåller sig till den slutgiltiga bilden på en skärm. Dessa olika typers kroppar - kantade och glatta - och deras egenskaper har presenterats. Den information om dessa objekt som presenteras i avhandlingen ger goda förutsättningar för att lära sig gränssnitt som använder sig av dessa objekt (exempelvis OpenGL) och förstå deras dokumentation. Om inte annat är avhandlingen en god intuitiv förklaring till matematiska koncept som annars kan vara delvis ogenomträngliga för en mindre teknisk person.

Geometrisk modellering är ett verktyg för konstnärer och formgivare att skapa exakta former som inte varit åtkomlig till allmänheten förrän de senaste årtionden. Författaren har under senaste åren sätt en ökad mängd konst cirkulera på webben skapad med hjälp av statiska eller tredimensionella objekt. Jag hoppas att utvecklingen fortsätter åt det hållet, och modelleringsverktyg görs ännu mer lätttillgängliga åt kreativa individer.

Källor:

[1] P. Bajcsy, K. McHenry, *An Overview of 3D Data Content, File Formats and Viewers. National Center for Supercomputing Applications (NCSA)*, University of Illinois at Urbana-Champaign, Urbana, IL. Technical Report. October 2008.
[2] Wavefront, *Obj 3.0 format-specifikation*, [Online]. Tillgänglig vid:http://www.cs.utah.edu/~boulos/cs3505/obj_spec.pdf
[3] E. Catmull, *A Subdivision Algorithm for Computer Display of Curved Surfaces,* Doktorsavhandling, University of Utah, 1974
[4] J. Vince, *Mathematics for Computer Graphics 4th* ed, Springer, 2014.
[5] E.Catmull J.Clark,"Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes", *Computer Aided Design,* vol 10,nr 6, 1978
[6] N. Patrikalakis, T. Maekawa, W. Cho, (December 2009) *Shape Interrogation for Computer Aided Design And Manufacturing* ,(Hyperbook ed.) [Online].Tillgänglig vid:http://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/
[7]. R. Roach; J. Forrest, *Curvature Continuity of Cubic Bezier Curves in the Solid Modeling Aerospace Research Tools Design Software ,*Interim Report: NASA-CR-186583, NAS 1.26:186583, Januari 1990
[8] C. Van Overveld ; B. Wyvill *An algorithm for polygon subdivision based on vertex normals,* Computer Graphics International, 1997
[9] J. Stam *Exact Evaluation Of Catmull-Clark Subdivision Surfaces At Arbitrary Parameter Values,* Siggraph '98
[10] W. Zhu ;J. Zheng ; H. Zhou ; L. Shen *A Novel Adaptive Algorithm of Catmull-Clark Subdivision Surfaces,* FSKD '09.
[BILDKÄLLA Fig. 7] [Online] Tillgänglig vid: http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/surface/bezier-de-casteljau.html
[BILDKÄLLA Fig. 2][Online] (Modifierad) Tillgänglig vid: http://computergraphics.stackexchange.com/questions/335/non-uniform-rational-b-spline-nurbs-basics/343