FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

Computergrafiek - Examen

Bert De Saffel

Master in de Industriële Wetenschappen: Informatica Academiejaar 2018–2019

UNIVERSITEIT
GENT

Inhoudsopgave

Hoofdstuk 1

Modelvragen eerste theorievraag

Deze vraag wordt gequoteerd op 1/4 van de totaalpunten.

1.1 Rastering

1. Bespreek de verschillende algoritmen voor de *rastering van rechte lijnen*, zonder in detail in te gaan op *multi-step* varianten. Vermeld telkens hun voor- en nadelen. (§1.2 & §1.2.1)

   - **Midpoint subdivision.** Dit algoritme maakt gebruik van een recursieve procedure waarbij het midden van een lijnstuk berekent wordt, de overeenkomstige pixel selecteert, en daarna deze procedure opnieuw aanroept om de twee helften van het segment. Dit algoritme werkt nadelig naarmate een hellingshoek van 45 graden bereikt wordt. Het effect is dan onregelmatig en dunner. Het grootste nadeel is echter dat het me reële getallen moet uitgevoerd worden.
   - **Asymmetrische DDA.** Algoritmes van de familie DDA pogen het aantal reële getallen te beperken, en voeren hun berekeningen ook incrementeel uit in plaats van recursief. Bij asymmetrische DDA wordt de x-waarde wordt telkens met 1 verhoogt terwijl de y-waarde verhoogt wordt met de richtingscoëfficiënt van de rechte. In elke stap wordt een pixel berekend en geselecteerd. Indien de helling van de rechte met de x-as groter zou zijn dan 45 graden, worden de rollen van x en y omgewisseld. Hier is echter ook het probleem dat indien de hellingshoek 45 graden nadert, dat de lijn te dun wordt getekend.
   - **Symmetrische DDA.** De symmetrische variant van DDA vermijdt het onderscheid in functie van de hellingshoek ten opzichte van de x-as, door de incrementen in beide richtingen repetitief te halveren tot wanneer deze kleiner zijn dan een pixel. Nadat de incrementen berekend zijn, wordt hetzelfde principe toegepast als asymmetrische DDA. De x en y-waarde worden nu telkens verhoogt met de eerder berekende incrementen, en selecteren de juiste pixel. Aangezien de incrementen kleiner zijn dan een pixel, zullen sommige pixels meerdere malen geselecteerd worden, wat een impact heeft op de snelheid van het algoritme. Het resultaat indien de hellingshoek 45 graden nadert, is voller dan by de asymmetrische variant, maar het patroon is minder regelmatig. Het produceert hetzelfde resultaat als midpoint subdivision.

- **Kwadrant DDA.** Dit algoritme verhoogt de incrementen met exact 1 pixel, maar nooit in beide richtingen tegelijkertijd. De verhouding van het aantal incrementen in de y-richting tot het aantal incrementen in de x-richting wordt bepaald door de richtingscoëfficiënt van de rechte. Er is een extra variabele, Δ die best geinitialiseerd wordt op $\frac{(x_2-x_1)-(y_2-y_1)}{2}$. Indien Δ foutief geinitialiseerd wordt, produceert dit algoritme slechte resultaten bij zeer vlakke of zeer steile lijnstukken. De variabele Δ bepaalt of de lijn verder getekend moet worden in de x-richting, indien $\Delta \geq 0$, of in de y-richting wanneer $\Delta < 0$.

  Een enorm voordeel van dit algoritme is dat het geen gebruik maakt van reële getallen, ook niet bij de variabele Δ aangezien het dubbele bijgehouden wordt. Bij een hellingshoek die 45 graden nadert, zijn de lijnstukken veel te dik. Dit is logisch aangezien de helft van de beslissingen het lijnstuk in de x-richting zullen selecteren, en de andere helft in de y-richting, telkens afwisselend van elkaar.

- **Octant DDA.** Dit laatste algoritme verfijnt kwadrant DDA door overlappende segmenten te vermijden. Indien de hellingshoek van het lijnstuk kleiner zou zijn dan 45 graden, zal octant DDA een pixel ofwel in de horizontale richting of in de diagonale richting selecteren, afhankelijk naargelang Δ positief of negatief is. Is de hellingshoek groter, dan zal octant DDA in de verticale of diagonale richting selecteren.

  Dit algoritme produceert ongeveer dezelfde resultaten als asymmetrische DDA, zonder gebruik te maken van reële getallen. Naarmate de hellingshoek 45 graden nadert, wordt het lijnstuk te fijn weergegeven.

2. Hoe kunnen de methodes aangepast worden om **dikke** lijnen voor te stellen ? (§1.5)

   - **Replicatie.** Dit is een eenvoudige manier, dat bij elke iteratie van een DDA algoritme, enkele pixels boven en onder, of links en rechts van de geselecteerde pixels meeselecteren, afhankelijk van de gewenste dikte en de hellingshoek. De overgang tussen twee lijnstukken die op deze manier dikker gemaakt zijn, is niet altijd even goed. Deze techniek wordt enkel toegepast indien de lijnen relatief dun zijn.
   - **Replicatie met vorm.** Bij elke iteratie van een DDA algoritme wordt rond de huidige geselecteerde pixel, een vlak geselecteerd. Vaak gebruikte vlakken zijn vierkanten voor lijnstukken en cirkelvormige vlakken voor cirkels.
   - **Opvulling.** Een derde techniek is om tweemaal het DDA algoritme toe te passen, op lijnstukken die uit elkaar verschoven zijn. Daarna worden deze twee lijnstukken opgevuld met de techniek om veelhoeken op te vullen.

1.2 Het algoritme van Bresenham

1. Bespreek het doel van dit algoritme en geef de volledige uitwerking van het selectieproces. (§1.3)

   - Het algoritme van Bresenham is geschikt om pixels te rasteren in de vorm van een cirkel terwijl het ook enkel gehele getalllen vereist. Het algoritme berekent enkel de pixels voor de cirkelsector met beginpunt $(0, R)$ en eindpunt $(R/\sqrt{2}, R/\sqrt{2})$. De andere cirkelsectoren kunnen met behulp van symmetrie bekomen. Bij elke iteratie wordt de x-waarde van de pixel verhoogd. Het aanpassen van de y-waarde hangt af van de discriminant Δ_i. Is deze waarde strikt positief dan zal de y-waarde met 1 pixel verlaagt worden. Is de discriminant

negatief, dan blijf de y-waarde dezelfde. De discriminant Δ_i wordt gedefinieerd als het verschil van de kwadraten van de afstanden van de twee mogelijke punten tot de cirkel:

$$\begin{aligned}\Delta_i &= [(x_{i-1}+1)^2 + y_{i-1}^2 - R^2] - [R^2 - (x_{i-1}+1)^2 - (y_{i-1}-1)^2] \\ &= 2(x_{i-1}+1)^2 + y_{i-1}^2 + (y_{i-1}-1)^2 - 2R^2\end{aligned}$$

De discriminant Δ_{i+1} kan uit Δ_i afgeleidt worden.

$$\begin{aligned}\Delta_{i+1} &= 2(x_{i-1}+2)^2 + y_i^2 + (y_i-1)^2 - 2R^2 \\ &= \Delta_i + 4x_i + 6 + y_i^2 - y_{i-1}^2 + (y_i-1)^2 - (y_{i-1}-1)^2 \\ &= \begin{cases} \Delta_i + 4x_{i-1} + 6 & ,\Delta_i \leq 0 \\ \Delta_i + 4(x_{i-1} - y_{i-1}) + 10 & ,\Delta_i > 0 \end{cases}\end{aligned}$$

De beginwaarde van de discriminant, Δ_i kan bekomen worden door de substitutie $x_{i-1} = 0$ en $y_{i-1} = R$ toe te passen:

$$\Delta_1 = 3 - 2R$$

2. Hoe kunnen de methodes aangepast worden om **dikke** lijnen voor te stellen ? (§1.5)

   - **Replicatie.** Dit is een eenvoudige manier, dat bij elke iteratie van een DDA algoritme, enkele pixels boven en onder, of links en rechts van de geselecteerde pixels meeselecteren, afhankelijk van de gewenste dikte en de hellingshoek. De overgang tussen twee lijnstukken die op deze manier dikker gemaakt zijn, is niet altijd even goed. Deze techniek wordt enkel toegepast indien de lijnen relatief dun zijn.
   - **Replicatie met vorm.** Bij elke iteratie van een DDA algoritme wordt rond de huidige geselecteerde pixel, een vlak geselecteerd. Vaak gebruikte vlakken zijn vierkanten voor lijnstukken en cirkelvormige vlakken voor cirkels.
   - **Opvulling.** Een derde techniek is om tweemaal het DDA algoritme toe te passen, op lijnstukken die uit elkaar verschoven zijn. Daarna worden deze twee lijnstukken opgevuld met de techniek om veelhoeken op te vullen.

1.3 Rastering van veelhoeken en antialiasing

1. Hoe moeten algoritmen voor het rasteren van rechte lijnen gewijzigd worden indien men ze wil toepassen op het *opvullen van veelhoeken* ? (§1.4)

   - De veelhoek wordt afgetast met een horizontale of verticale scanlijnen. Voor elke scanlijn worden de intersectiepunten met de lijn en het veelhoek bepaalt. Het aantal intersectiepunten is altijd een even aantal. De intersecties worden (bij een horizontale scanlijn) gesorteerd op de x-waarde. Alle pixels x tussen een intersectie met een even index en een intersectie met oneven index, $x_{2i} \leq x \leq x_{2i+1}$, worden getekend. Om te voorkomen dat pixels buiten het veelvlak geselecteerd worden, beperkt men de selectie tot enkel inwendige pixels als intersectiepunten, ook als ligt een uitwendig pixel dichter bij de randlijn.

2. Geef het doel van *antialiasing*, het algemeen principe ervan, en drie algoritmen voor de praktische uitwerking (met voorbeelden). (§1.6)

- Antialiasing is het proces om de discrete eigenschap dat pixels vertonen, namelijk het gekarteld uitzicht, te minimaliseren.
  (a) **Supersampling.** Deze techniek maakt gebruik van een interne geheugenbuffer. Deze buffer biedt een fijner raster aan dan die dat hardwarematig beschikbaar zijn. Een pixel op een computerscherm kan overeenkomen met 16 pixels in het geheugen. Na het rasteringsproces, dat nu uitgevoerd wordt in het geheugen, wordt voor elke pixel nagegaan hoeveel geheugenpixels in de groep geselecteerd zijn. Als 7 pixels geselecteerd zijn met een geheugenpixelgrootte van 16 pixels, dan zal die pixel een intensiteit van 7/16 bedragen.
  (b) **Postfiltering.** Deze methode zal eerst het rasteringsproces uitvoeren in het geheugen, dat nu dezelfde resolutie heeft als de hardware. Nadien wordt de intensiteit van een pixel herberekend op basis van het al dan niet geselecteerd zijn van de naburige pixels. De groep dat in beschouwing wordt genomen heeft vaak de vorm van een rechthoek (meer specifiek, een vierkant). Het gewicht van een naburige pixel wordt berekend op basis van de afstand van die pixel tot het middenpunt van de groep.
  (c) **Prefiltering.** In tegenstelling tot de vorige twee algoritmen zal het rasterproces en antialiasingproces zich in dezelfde stap plaatsvinden. Dit algoritme heeft nood aan aangepaste DDA algoritmen. Een specifieke implementatie is het Pitteway-Watkinson algoritme. Dit algoritme maakt het mogelijk om een pixel te selecteren terwijl zijn intensiteit incrementeel berekend wordt. ToDo: ??

1.4 Transformaties

1. Welke families transformaties worden in de computer-grafiek gebruikt, en waarom ?

   - **Affiene transformaties.** Hieronder vallen de rotaties, schaaloperaties, spiegelingen en translaties. Deze transformaties behouden colliniariteit en parallellisme. Evenwijdige lijnen zullen na een affiene transformatie nog steeds evenwijdig zijn.
   - **Perspectieve projecties.** Deze transformaties maken het mogelijk om een driedimensionaal object voor te stellen op een tweedimensionaal uitvoerapparaat.
   - De combinatie van deze twee transformaties biedt de mogelijkheidt om onder andere:
     – een object vanuit een andere kijkrichting te bekijken.
     – hij construeren van een object dat uit meerdere identieke componenten bestaat.

2. Geef en bespreek de matrixrepresentaties van de verschillende types transformaties en hun samenstellingen. (§2.1 behalve §2.1.4)

   - **Rotaties.**

1.5 Projecties en clipping

1. Welke soort projectie wordt in de computergrafiek gebruikt, en waarom ?

- De computergrafiek maakt gebruik van perspectieve projecties. Deze soort projecties simuleren de werking van het menselijk oog. Er wordt een oogpunt en een kijkrichting vastgelegd. De kijkrichting is een georiënteerde rechte door het oogpunt. Tussen het oogpunt en elk object wordt een rechte verbonden. Ergens op die rechte zal er een snijpunt zijn met het projectievlak. Dit snijpunt bepaalt het getransformeerde beeld van het punt.

2. Leid de algemene matrixvorm van deze projectie af. (§2.2)

   - Een perspectieve projectie kan afgeleid worden na het toepassen van een affiene transformatie. Een translatie T verplaatst het snijpunt van de kijkrichting met het projectievlak naar de oorsprong van het coördinatenstelsel. Deze transformatie wordt gevolgd door een rotatie R die het oogpunt op de negatieve z-as plaatst, in het punt met homogene coördinaten (0, 0, -d, 1). Hierdoor valt het projectievlak samen met het xy-vlak. Hierdoor is ook de z-waarde van het geprojecteerd punt nul. Op dit moment kunnen twee driehoeken beschouwd worden. De driehoek met de kijkrichting vanuit de y-as naar het zx-vlak (1) en de driehoek met de kijkrichting vanuit de x-as naar het zy vlak (2).
     (1) Hieruit kan afgeleidt worden dat $\frac{x_p}{d} = \frac{d}{z+d}$.
     (2) Hieruit kan afgeleidt worden dat $\frac{y_p}{d} = \frac{d}{z+d}$.
     In homogene coördinaten kan dit echter vereenvoudigd worden.:

$$x' \equiv w' \cdot x_p = x \qquad y' \equiv w' \cdot y_p = y \qquad \text{met} \qquad w' = \frac{z}{d} + 1$$

     In matrixvorm:

$$\begin{pmatrix} x' \\ y' \\ 0 \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

     Hieruit is het duidelijk dat een perspectieve projectie ook kan berekend worden via eenvoudige matrixvermenigvuldigingen:

$$\begin{pmatrix} x' \\ y' \\ 0 \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix} \cdot \begin{pmatrix} i_x & j_x & k_x & -\theta_x \\ i_y & j_y & k_y & -\theta_y \\ i_z & j_z & k_z & -\theta_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3. Wat is de bedoeling van clipping ? Bespreek clippen in twee en in drie dimensies. (§2.3 zonder deelparagrafen)

   - Een kijkrichting is begrensd door een bepaalde maximumwaarde. Vanuit het oogpunt kan er een pyramide beschouwd worden. Elke pixel dat zich niet in deze pyramide bevindt moet geclipt worden, aangezien deze toch niet gezien kan worden. De doorsnede van de pyramide met het projectievlak wordt het viewport genoemd. De viewport is evenwijdig met de x- en y-assen met het centrum in de oorsprong van het coördinatenstelsel waarin het geprojecteerd beeld afgebeeld. De selectie van alle pixels in perspectieve projectie die binnen de viewport liggen noemt men clipping. 3D Objecten worden eerste geclipt vooraleer ze perspectief geprojecteerd wordt. 2D objecten worden geprojecteerd en daarna pas geclipt. Het clippen van lijnstukken en opgevulde veelhoeken wordt door vrijwel alle algoritmen ondersteund. Bij het clippen van complexere veelhoeken wordt eerst de omhullende veelhoek berekent, en worden alle zijden geclipt met de viewport.

1.6 Het algoritme van Cyrus-Beck

1. Geef het doel, de toepasbaarheid, en de beperkingen van het algoritme, en de volledige uitwerking van het principe. Pas het algoritme stap-voor-stap toe op volgende viewport *(figuur wordt gegeven)* en een lijnstuk met eindpunten ... . (§2.3.2)

   - Het algoritme van Cyrus-Beck is geschikt om lijnen te clippen ten opzichte van een willekeurige convexe veelhoek. Het algoritme maakt gebruik van de geparameteriseerde voorstelling van een lijnstuk: $P(t) = P_1 + t(P_2 - P_1)$. Indien $0 \leq t \leq 1$, dan bevindt P(t) zich op het lijnstuk. Het Cyrus-Beck algoritme spoort deze t-waarden op van de intersecties met de verschillende zijden van de viewport en hieruit afleiden ofdat het lijnsegment hetzij partieel, hetzij totaal binnen de viewport ligt.

     De eerste stap van dit algoritme berekent alle $n_i[P(t) - f_i]$ functies en de waarden van t_i waarvoor deze functies nul worden. Indien al deze functies niet-negatieve waarden hebben voor zowel $t = 0$ als $t = 1$, dan ligt het lijnstuk volledig binnen de viewport. Is dit slechts het geval voor $t = 0$ of $t = 1$, dan ligt het lijnstuk zeker partieel in de viewport. Er moet bijgevolg extra kandidaatsnijpunten met de viewport gezocht worden. Enkel t_j waarden waarvor $0 \leq t_j \leq 1$ komen hiervoor in aanmerking. Voor elke t_j waarde wordt nagegaan of $n_i \cdot [P(t_j) - f_i] \geq 0$. De kleinste en grootste t_j leveren de gezochte snijpunten op.
   - Toegepast op een lijnstuk met punten $P_1(-1, 1)$ en $P_2(9, 3)$ in een viewport met dimensie 8 bij 4 (fig 2.23 p 27 in de cursus). Het lijnstuk wordt geparameteriseerd door

     $$P(t) = \begin{cases} x = -1 + 10t \\ y = 1 + 2t \end{cases}$$

     Neem $f_{L,B} = (0, 0)$ en $f_{T,R} = (8, 4)$. Uitrekenen van de $n_i[P(t) - f_i]$ functies wordt dan: Elke nulwaarde kan een waarde voor t zijn. Er moet echter gelden dat $n_i[P(t_j) - f_i] \geq 0, \forall i$.

zijde	$[P_1P_2]$	nulwaarde
$n_L, f_L = (0, 0)$	$(-1 + 10t - 0, 1 + 2t - 0) \cdot (1, 0) = -1 + 10t$	$1/10$
$n_R, f_R = (8, 4)$	$(-1 + 10t - 8, 1 + 2t - 4) \cdot (-1, 0) = 9 - 10t$	$9/10$
$n_B, f_B = (0, 0)$	$(-1 + 10t - 0, 1 + 2t - 0) \cdot (0, 1) = 1 + 2t$	$-1/2$
$n_T, f_T = (8, 4)$	$(-1 + 10t - 8, 1 + 2t - 4) \cdot (0, -1) = 3 - 2t$	$3/2$

     Dit komt neer op het substitueren van de nulwaarden t_i in elke andere functie en nagaan of dat het resultaat positief is voor elk van deze functies. In dit geval is dit enkel voor $t = 1/10$ en $t = 9/10$. Dit zijn dan ook de punten die snijden met de viewport.

2. Hoe clipt men meer ingewikkelde krommen en figuren ?

   - ToDo: oplossen

1.7 Clipping

1. Het algoritme van *Cohen-Sutherland*: geef het doel, de toepasbaarheid, en de beperkingen van het algoritme, en de volledige uitwerking van het principe. Pas het algoritme toe op relevante voorbeelden. Geef eveneens een variant van de techniek. (§2.3.1)

- Het algoritme van Cohen-Sutherland is geschikt voor het clippen van lijnstukken ten opzichte van een rechthoekig viewport. Dit algoritme is enkel aan te raden om 2D clipping toe te passen. Het algoritme wenst zo snel mogelijk te detecteren of lijnstukken volledig binnen of buiten het viewport liggen. Het algortime zal aan elk eindpunt van een lijnstuk een bitcode $b_3b_2b_1b_0$ toekennen. Bit b_0 wordt op 1 gezet indien het punt zich links van de viewport bevindt, en een bit b_1 identificeert een punt aan de rechterzijde van de viewport. Op dezelfde manier hebben punten onder en boven de viewport respectievelijk bits $b_2 = 1$ en $b_3 = 1$. Een lijnstuk dat volledig binnen de viewport ligt heeft dus twee eindpunten met bitcode 0000. De twee eindpunten van een lijnstuk AND'en kan twee gevallen opleveren:
  (a) Indien het resultaat verschillend van 0000 is, dan ligt het lijnstuk met zekerheid buiten de viewport.
  (b) Is het resultaat echter 0000 (zonder dat de bitcodes van beide eindpunten 0000 is) dan ligt slechts een deel van het lijnstuk in de viewport.

  In dit tweede geval wordt het midden van dit lijnstuk genomen, en wordt het algoritme recursief toegepast op beide helften. Deze iteraties moeten doorgevoerd worden tot wanneer de precisie van een pixel bereikt is. Het voordeel van deze methodiek is dat het performant in de hardware kan uitgevoerd worden. Er is enkel een deling door twee vereist en kan parallel uitgevoerd worden.

2. Het algoritme van *Sutherland-Hodgman*: geef het doel, de toepasbaarheid, en de beperkingen van het algoritme, en de volledige uitwerking van het principe. Pas het algoritme stap-voor-stap toe op volgend voorbeeld: (figuur wordt gegeven) . (§2.3.3)

   - _ToDo: oplossen

Hoofdstuk 2

Modelvragen tweede theorievraag

Deze vraag wordt gequoteerd op 1/4 van de totaalpunten.

2.1 NURBS constructie van cirkels (§3.4.8, slides en les-nota's)

1. Met welke *open-uniforme NURBS* van orde drie (graad twee) kun je een *halve cirkel* (met centrum in de oorsprong en straal 1) tekenen , zonder (reële) knooppunten met meervoudige multipliciteit te moeten gebruiken ? Geef de preciese locatie van de *controlepunten* (op een figuur), hun gewichten, en de corresponderende *knopenvector*. Uit hoeveel segmenten bestaat deze NURBS ?

   - _ToDo: oplossen

2. Toon aan dat deze constructie inderdaad exact een halve cirkel oplevert.

   - _ToDo: oplossen

3. Construeer van deze NURBS de *uniforme* representatie. Vermeld de conversiestappen om tot dit resultaat te bekomen. Waarom is de constructie van de uniforme representatie belangrijk ?

   - _ToDo: oplossen

2.2 NURBS constructie van cirkels en lijnsegmenten (§3.4.8 en slides)

1. Met welke *NURBS* kun je exact een recht *lijnsegment* door twee punten tekenen ? Geef de preciese locatie van de *controlepunten* (op een figuur), hun gewichten, en de corresponderende *knopenvector*.

- _ToDo: oplossen

2. Met welke *NURBS* bestaande uit één enkel segment kun je een *halve cirkel* (met centrum in de oorsprong en straal 1) tekenen ? Geef de preciese locatie van de *controlepunten* (op een figuur), hun gewichten, en de corresponderende knopenvector. Wat is de graad van deze NURBS ?

   - _ToDo: oplossen

3. Toon aan dat deze constructie inderdaad exact een halve cirkel oplevert.

   - _ToDo: oplossen

4. Met welke *NURBS* bestaande uit één enkel segment kun je exact een *volledige cirkel* (met centrum in de oorsprong en straal 1) tekenen ? Geef de preciese locatie van de *controlepunten* (op een figuur), hun gewichten, en de corresponderende *knopenvector*. Wat is de graad van deze NURBS ?

   - _ToDo: oplossen

2.3 Reflectiemodellen (§5.2, behalve §5.2.1.2 en §5.2.1.3)

1. Waarom zijn reflectiemodellen noodzakelijk ?

   - _ToDo: oplossen

2. Omschrijf het lokale reflectiemodel (Phong-model). Geef ondermeer de

   - _ToDo: oplossen

   berekenings-voorschriften, de betekenis van de parameters, en de nadelen.

3. Geef en omschrijf (in het bijzonder de nadelen) van de drie mogelijke benaderingen voor de berekening van de *lichtintensiteit van zichtbare punten*, indien men het object beschrijft aan de hand van een verzameling *vlakke veelhoeken*.

   - _ToDo: oplossen

2.4 1D Wavelet transformaties

1. Bespreek met behulp van *Multi-Resolutie-Analyse* de algemene concepten van wavelet transformaties. (§3.5.2)

   - _ToDo: oplossen

2. Vertaal deze algemene concepten in het bijzonder geval van de *Haar-wavelet* transformatie. (§3.5.1 & §3.5.2)

   - _ToDo: oplossen

3. Bespreek de noodzaak van *spline-wavelets* (1D). Wat is het verband tussen de *Haar-wavelet* transformatie en de *spline-wavelet* transformatie ? Geef een overzicht van de relatieve voor- en nadelen.

   - ToDo: oplossen

4. Beschrijf van lage orde 1D *open-uniforme* spline-wavelet transformaties de vorm van achtereenvolgens de *schaalfuncties*, de *wavelets* en de *synthese filters*.

   - ToDo: oplossen

2.5 2D Wavelet transformaties (§4.5.1)

1. Bespreek de alternatieve methodes om 2D *schaalfuncties en wavelets* te construeren.

   - ToDo: oplossen

2. Beschrijf, aan de hand van *contourplotjes*, en voor elk van deze alternatieve methodes, de resulterende *2D Haar-schaalfuncties en Haar-wavelets* van het laagste en het op één na laagste niveau.

   - ToDo: oplossen

2.6 Toepassingen van wavelet transformaties

1. Geef de meest relevante toepassingen in de computergrafiek van 1D *Haar-wavelet* en 1D *spline-wavelet* transformaties. (§3.5.4)

   - ToDo: oplossen

2. Geef de meest relevante toepassingen in de computergrafiek van 2D *Haar-wavelet* en 1D *spline-wavelet* transformaties. (§4.5.2)

   - ToDo: oplossen

Hoofdstuk 3

Informatie derde vraag

Deze vraag is een **oefening**, gequoteerd op 1/2 van de totaalpunten, over één of enkele van volgende onderwerpen:

- (1-3) de Casteljau constructie (van een punt met specifieke parameterwaarde) van een Bézier kromme
  - Gegeven volgende inputgegevens: Stel de casteljauconstructie op voor t=0,3; t=0,6 en t=0,9 op. Teken ook de kromme.
  - Eerst algemene constructie opstellen: ToDo: lol, te moeilijk om op pc te doen, maar zoals Fig 3.24 op pagina 44 cursurs Uiteindelijke uitkomst:

Figuur 3.1: t = 0,3 Figuur 3.2: t = 0,6 Figuur 3.3: t = 0,9

- (1) verhoging van de graad van Bézier splines (in één enkele stap); voorafgaand moet het verband tussen de *oude* en de *nieuwe* controlepunten opgesteld worden (vermenigvuldiging met een specifieke matrix, cfr. theorieles)

  Eerst algemene formule (niet te kennen): Een verhoging van de graad kan eenvoudig uitgevoerd worden met een matrix, waarin de coëfficiënten van Pascal instaan. Hier uitgewerkt om een Bézier kromme van graad 4 te verhogen naar graad 7, in één enkele stap:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 3 & 3 & 1 \\ 0 & 4 & 6 & 4 \\ 0 & 0 & 10 & 10 \\ 0 & 0 & 0 & 20 \end{pmatrix} \rightarrow \begin{pmatrix} 20 & 0 & 0 & 0 \\ 10 & 10 & 0 & 0 \\ 4 & 6 & 4 & 0 \\ 1 & 3 & 3 & 1 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 20 & 0 & 0 & 0 \\ 10 & 10 & 0 & 0 \\ 4 & 12 & 4 & 0 \\ 1 & 9 & 9 & 1 \\ 0 & 4 & 12 & 4 \\ 0 & 0 & 10 & 10 \\ 0 & 0 & 0 & 20 \end{pmatrix}$$

  Elke kolom stelt één van de originele 4 punten voor. Een punt P_i kan hieruit afgeleidt worden door op de i-de rij, de som van de producten van het originele punt met de factor voor die kolom te berekenen, en deze som te delen door $\binom{m}{n}$. In dit geval is $m = 7$ en $n = 4$, zodat $\binom{7}{4} = \frac{7!}{4!3!} = 35$. De zeven punten worden dus:

  * $P_0 = \frac{1}{35}(20P_0 + 0P_1 + 0P_2 + 0P_3) = \frac{4}{7}P_0$ ToDo: klopt niet, zou gewoon P_0 moeten zijn...
  * $P_1 = \frac{1}{35}(10P_0 + 10P_1 + 0P_2 + 0P_3) = \frac{2}{7}[P_0 + P_1]$
  * $P_2 = \frac{1}{35}(4P_0 + 12P_1 + 4P_2 + 0P_3) = \frac{1}{35}(4P_0 + 12P_1 + 4P_2)$
  * $P_3 = \frac{1}{35}(1P_0 + 9P_1 + 9P_2 + 1P_3) = \frac{1}{35}(P_1 + 9P_2 + P_3)$
  * $P_4 = \frac{1}{35}(0P_0 + 4P_1 + 12P_2 + 4P_3) = \frac{1}{35}(4P_1 + 12P_2 + 4P_3)$
  * $P_5 = \frac{1}{35}(0P_0 + 0P_1 + 10P_2 + 10P_3) = \frac{2}{7}(P_2 + P_3)$
  * $P_6 = \frac{1}{35}(0P_0 + 0P_1 + 0P_2 + 20P_3) = \frac{4}{7}(P_3)$

- (2) verhoging van de graad van Bézier splines (*stapsgewijs*: één graad verhogen per stap)

  – Slechts één graad verhogen per stap is heel eenvoudig:

$$Q_k = \frac{k}{n+1}P_{k-1} + \frac{n+1-k}{n+1}P_k$$

  Stel een bézierkromme van de 3de graad met punten $P_{000}, P_{001}, P_{011}$ en P_{111}.
  De nieuwe punten van de bézierkromme van de 4de graad worden dan:

  * $Q_{0000} = \frac{0}{3+1}P_{0-1} + \frac{3+1-0}{3+1}P_1 = P_{000}$
  * $Q_{0001} = \frac{1}{4}P_0 + \frac{4-1}{4}P_1 = \frac{1}{4}P_{000} + \frac{3}{4}P_{001}$
  * $Q_{0011} = \frac{1}{2}P_{001} + \frac{1}{2}P_{011}$
  * $Q_{0111} = \frac{3}{4}P_{011} + \frac{1}{4}P_{111}$
  * $Q_{1111} = P_{111}$

  Het punt Q_{0001} ligt bijvoorbeeld op $t = 0.75$ voor het lijnstuk $[P_{000}P_{001}]$.

- (3,4) segmentering (subdivisie) van Bézier krommen (eventueel meerdere segmenten in één enkele stap)

  - _ToDo: oplossen

- (9-10,14-16) constructie van de *kromtecirkel* in een punt van een Bézier kromme
  - _ToDo: oplossen

- (5-7,9-11) constructie van de *Bézier representatie* van een (polynomiale) NURBS
  - _ToDo: oplossen

- (6-10) constructie van controlepunten na toevoeging van één of meerdere *reële* knopen in de knopenvector van een (polynomiale) NURBS (zonder over te gaan op de Bézier representatie)
  - _ToDo: oplossen

- (10) constructie van controlepunten na toevoeging van één of meerdere *virtuele* knopen in de knopenvector van een (polynomiale) NURBS (zonder over te gaan op de Bézier representatie)
  - _ToDo: oplossen

- (11) berekening en constructie van de controlepunten van de *open-uniforme* representatie van een (polynomiale) NURBS met een *uniforme* knopenvector
  - _ToDo: oplossen

- (12,19) berekening en constructie van de controlepunten van de *uniforme* representatie van een polynomiale of rationale NURBS met een *open-uniforme* knopenvector
  - _ToDo: oplossen

- (12) de Boor constructie (van een punt met specifieke parameterwaarde) van een (polynomiale) NURBS
  - _ToDo: oplossen

- (13) constructie van de *hodograaf* van een Bézier kromme of spline
  - _ToDo: oplossen

- (13-16) vaststellen van de continuïteit in de knooppunten van Bézier splines (*stelling van Stärk*)
  - _ToDo: oplossen

- (17) constructie van de controlepunten van de *uniforme* Lagrange representatie van een Lagrange geïnterpoleerde kromme met *niet-uniforme* knopenvector; schematisch aantonen hoe de berekening van de Bézier representatie van deze kromme zou kunnen uitgevoerd worden (ondermeer opstellen van de *inverse* van de Bézier basismatrix).
  - _ToDo: oplossen

- (18,19) constructie van een *benadering door lijnstukken* van een uniforme NURBS door toepassing van het algoritme van Lane & Riesenfeld

  - _ToDo: oplossen

- (20) constructie van een *triangulair schema* met behulp van het veralgemeend algoritme van Neville (voor een specifieke configuratie van inputgegevens), en berekening hieruit van de *gewichtsfuncties* en de *matrixrepresentatie*

  - _ToDo: oplossen