tent. 8.5 eindcijfer (9)
prakt. 9.5

Afdeling Wiskunde en Informatica R.U.G.

Naam: Eamon Nerbonne	Studentnummer: 1148044	Bladnr.: 1/3
Adres:	Studierichting:	Tentamen:
Postcode en		Datum:
Woonplaats:	Jaar van eerste inschrijving:	Naam docent:

8.5 pt

a) De midpoints methode loopt over een van de assen in stappen van 1, en kiest steeds of relatief van aan de vorige stap de waarde van de andere as gewijzigd moet worden. Dit impliceert ook dat bij een eenvoudige implementatie de "basis"-as met een stap kan stijgen terwijl de andere as er twee doet. Of teveel $dx = \frac{1}{dx}$ max niet 1 kan bedragen. Bij $\frac{dy}{dx} > 1$, moet het gespiegeld worden bekeken, met een loop in y en beslissing over x. (Wel zo als $\frac{dy}{dx} = -1$)

Maar:

$$4Ry = x^2$$

$$\Rightarrow \frac{dy}{dx} = \frac{1}{2R}x$$

$$\Rightarrow \max\left(\frac{dy}{dx}\right) = \frac{1}{2R}\max(x) \qquad \max(x) = R/2R$$

$$\Rightarrow \max\left(\frac{dy}{dx}\right) = 1$$

Net zo valt ook aan te tonen dat $-1 \leq \frac{dy}{dx}$

dus $-1 \leq dy/dx \leq 1$

In dit voorbeeld wordt alleen over x geïtereerd. 8

6 In een punt (x_k, y_k) op de rasterisatie van de parabool moet dus worden besloten of het volgende punt (x_k-1, y_k-1) x_{k+1}, y_{k+1} wordt of (x_k+1, y_k).

[illegible] is dus of y_{k+1} [illegible] wordt [illegible]
y_k of door $y_k + 1$. Het midpunt nemende, is duidelijk dat als de lijn erboven verloopt $y_k + 1$ goed is en y_k anders.
De parabool interpreterende als nullijn van $F(x,y)$ in de gedachte, kun je zien dat als $F(x_{..}, y_m)$ positief is, de parabool onder y_p verloopt, en als $F < 0$ anders om.

Verblijft nog F te berekenen.

2 Het eerste beslispunt is $(1, \frac{1}{2})$, waar $F = 2R - 1$. 8
Het volgende punt is telkens $(x+1, y+1)$ of $(x+1, y)$

Je moet het midpunt nemen

$F(x+1, y) = 4Ry - (x^2 + 2x + 1) = F(x, y) - 2x - 1$

$F(x+1, y+1) = 4R(y+1) - (x^2 - 2x - 1) = F(x,y) + 4R - 2x - 1$

$\Delta F_0 = -2x - 1$ als y niet opgehoogd wordt
$\Delta F_1 = 4R - 2x - 1$ als y wel opgehoogd wordt

helaas bevat ΔF nog een referentie aan x, ook deze valt weg te werken:

$\Delta\Delta F = -2.$
Al deze berekeningen zijn gewoon geheeltallig!

6 Dus in pseudocode: (erg C-achtig)

```
int x=0, y=0, F = 2*R-1, dF0 = 7, dF1 = 4R-1

putpixel(x,y)

while (x < 2*R) {
  if (F > 0) {
    F += dF0;
  } else {
    F += dF1; y += 1;
  }
  x += 1; dF0 -= 2; dF1 -= 2;
  putpixel(x, y); putpixel(-x, y);      8.
}
```

Naam: Eamon Nerbonne
Adres:
Postcode en
Woonplaats:

Studentnummer: 1148044
Studierichting:
Jaar van eerste inschrijving:

Bladnr.: 2/3
Tentamen:
Datum:
Naam docent:

Let ook dat gebruik gemaakt is van de ~~symmetrie~~ ~~symmetrie~~ symmetrie rond de y-as om zowel op -x als +x te tekenen.

1b) Het algoritme is is rekent telkens nieuwe paren (-x,y) en (x,y) punten op. ~~Hertassen~~ moet worden opgevuld. Dit elke ~~...~~ keer doen is overbodig, immers verandert y meestal niet. We gaan dus alleen vullen als y geïncrementeerd wordt.

Voeg gewoon toe dus meteen na "y+=1;" en vóór de sluit accolade:

for(i=-x; i<=x; i++) putpixel(i,y); — **nu worden de eindpunten nog 2x getekend**

i moet in c eerder als int gedeclareerd worden.

In dit ~~algo~~ algoritme is geen pixel twee keer getekent.

2) a) Voor elke pixel in ons beeld, schiet er een ray doorheen.

Voor elke ray, bereken de dicht-bij zijnde snijpunt met de scene. De licht waarde van de ray is de emissie van dit object in de scene plus de som van alle zichtbare licht-bronnen in diffuse belichting (dwz ook spec. maar afhankelijk van je belichtings model), plus de licht-termen van alle subrays. De hele optelling is gewogen afhankelijk van je belichtings model, wat te maken heeft met het materiaal van het geraakte object. Er kunnen ook termen in zitten voor allerlei andere uitbreidingen; bijv zou je een distance attenuation kunnen doen van alle rays om fog te simuleren. Subrays kunnen weer voor van alles zijn zoals reflecties en

foreach pixel:
    calc ray through eye and pixel, find closest intersection
    pixel = ray.shoot();

7

ray.shoot(): don't process if recursed "too far"
    color = weighted color model average of:
        - surface emission
        - diffuse reflection (foreach light source)
        - ambient lighting
        - sub-ray colors for: (so calc sub-ray directions + intersections)
            - reflection (possibly multiple)
            - diffraction
    return color; (maybe attenuated by distance/atmospheric term)

de recursie is niet evident in deze pseudocode

Er is veel flexibiliteit voor uitbreiding in dit recursief model. Je zou bv voorbeeld van elke subray voor je em zelf evalueerd, zijn coefficient uit rekenen en op basis daarvan en niet recursie diepte beslissen wel of niet verder te rekenen.

b) Arbitraire algebraïsche oppervlakken kunnen niet zonder meer in een gesloten expressie (dus zonder trial and error zoektocht) worden opgelost. Dit is wel mogelijk als de graad laag is. Substitutie van y en z bijv. levert dan een oplossingen set. Omdat lijnen (uiteraard) lineair zijn, is substitutie altijd mogelijk en blijft de graad van het resultaat ook gelijk. ?

5

Normalen zijn berekenbaar dmv. partiele differentiatie. Er zijn formules om lage graads polynomen op te lossen, deze zijn gewoon opzoekbaar.
Als je bijv. de part. dif met resp. tot x en tot y uit rekent, krijg je twee vectoren die allebij parallel aan de oppervlakte zijn maar onderling orthogonaal. Met een kruisproduct kun je dan de normaal vinden. Het is niet de bedoeling om in de code part. diff. enzo uit te rekenen, dit kunsje van te voren zodat al deze stappen gewoon reduceren tot het berekenen van een polynoom.

$$\underline{n} = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$$

Naam: Eamon Nerbonne
Adres:
Postcode en Woonplaats:

Studentnummer: 1148044
Studierichting:
Jaar van eerste inschrijving:

Bladnr.: 3/3
Tentamen:
Datum:
Naam docent:

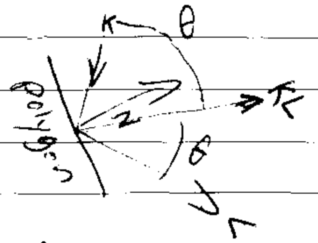
c) ~~b)~~ Bounding boxes: bereken alleen die snijpunten die ~~[illegible]~~ überhaupt in de buurt komen van je object, en depth-sorting: als je object volledig achter een ~~[illegible]~~ ander object zit, ~~[illegible]~~ hoef je de snijpunt ook niet uit te rekenen – terwijl normaliter je alle ~~[illegible]~~ snijpunten vindt en dan de meest dichtbijzijnde kiest. **Werkt niet bij transp. objecten**

3

d) Radiosity: Deze methode splitst de ~~scene~~ scene oppervlaktes in regios, en berekent hoe goed elke oppervlak elke andere kan "zien". Met deze informatie, en een vrij simpel belichtings model met ~~[illegible]~~ absorptie en emmissie en een "zien"-gebaseerde diffuse reflectie, kun je een heleboel ~~parallel~~ simultane vergelijkingen maken. Deze worden meestal gewoon iteratief opgelost, en je hebt per oppervlak een diffuse licht waarde. **1e deel vraag niet beantwoord.**

2

3. a) Gouraud en Phong verschillen subtiel. Beiden proberen de gereflecteerde lichtintensiteit te berekenen ~~[illegible]~~ door de normaal van de oppervlakte van het polygoon ~~[illegible]~~ en het gemiddelde tussen kijk en licht te nemen en hiervan een inproduct te nemen en ~~[illegible]~~ dit tot een macht nemen. dus: $(KL \cdot N)^P$ met KL en N genormaliseerd.



θ hoort gelijk θ maar het diagram is scheef.

4

Maar ~~[illegible]~~ Gouraud maakt deze ~~[illegible]~~ berekening per vertex en interpoleert de pixelkleurwaardes dan, en Phong interpoleert de vectors en berekent dan pas het resultaat. ?

b) Als de highlite in het inwendige van het polygoon valt zullen alle vertexes ~~[illegible]~~ donkerder dan de echte highlite. Maar ~~Gouraud~~ Gouraud-shading interpoleert kleuren en zal dan ook een (vaak veel) te donker resultaat opleveren op het highlite. Bij animaties bijzonder storend

6

(maar als highlight zich vaak weleens tussen vertex-
en centrum van een polygoon beweegt en je dan (vooral bij een hoge macht p) een irritant flitsen ziet (flicker).

c) Om de berekening goed uit te voeren heb je de licht vector en de kijk vector nodig. Bij gouraud-shading maakt het dus niet echt uit wanneer je dit
60 doet (maar intern moet je wel al een camera positie en licht positie hebben), maar Phong is per pixel en moet dus perse tijdens de rasterisatie, en dus na de kijktransformatie.