**Institutionen för teknisk databehandling**
Bo Nordin (bosse@cb.uu.se, 183470) 950114

# Lösningsförslag till
# Tentamen i datorgrafik MN1, 950114

Dess lösningar är ej fullständiga, en del uppgifter kräver figurer och i några fall (uppgift 4 och 8b) krävs något utförligare svar för full poäng (se referensen till läroboken).

---

1a) Homogena koordinater har en extra dimension jämfört med vanliga koordinater. Man skriver t.ex. (x,y,z,1) i.st.f. (x,y,z). Om homogena koordinater används kan också translation och perspektivprojektion uttryckas m.h.a. matrismultiplikation. Exemplifiera t.ex. med translation. Med hjälp av homogena koordinater kan alla enkla transformationer uttryckas med en enda transformationsmatris (*se vidare sidan 171 i Foley et al*).
1b) I grafiska paket kan man ofta specificera figurer i världskoordinater (WC) som kan anges i helt valfria, godtyckliga enheter. På en grafisk skärm anges alla punkter i skärmkoordinater (SC). Vid översättning från WC till SC används ett fönster (window) i WC och en viewport i SC. En punkt inom fönstret mappas till en punkt i tittgluggen. Oftast utförs klippning i samband med mappningen (*se vidare sidorna 177-179 i Foley et al*).

---

2. Lösning 1: Tre steg

a) translation till (0,0)

Matris T(-2,-2):

$$\begin{matrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{matrix}$$

b) rotation 180 grader

Matris R(180):

$$\begin{matrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

c) translation till (5,-2)

Matris T(5,-2):

$$\begin{matrix} 1 & 0 & 5 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{matrix}$$

Sammanlagd matris T(6,2)R(180)T(-2,-2):

$$\begin{matrix} -1 & 0 & 7 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Lösning 2: Två steg:

a) translation till (1,2)

Matris T(-1,0):

$$\begin{matrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

b) skalning för spegling i x-axeln

Matris S(1,-1):

$$\begin{matrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Sammanlagd matris S(1,-1)

$$\begin{matrix} 1 & 0 & -1 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

OBS: Förutsätter postmultiplikation av kolumnvektorer. Om man i stället vill premultiplicera radvektorer blir matriserna transponerade.

---

3a. Begreppen request, sample och event mode används vid grafisk input (från någon enhet, t.ex. mus eller tangentbord).

Request - applikationen frågar systemet (det grafiska t.ex.) om ett värde (i detta fall har programmet initiativet).

Sample - applikationen läser värden från en enhet kontinuerligt utan triggning

Event - systemet (det grafiska t.ex.) informerar applikationen om att någon enhet har triggats (i detta fall har användaren - som triggar enheter - initiativet)

3b. En X-Windows applikation befinner sig nästan alltid i en s.k. "event loop":

```
while (1) { GetNextEvent(); DoSomething; }
```

De två typerna av requests (som alltså kommer från applikationen) är "round-trip" (som kräver ett svar) och sådana som inte kräver något svar. Anledningen till att man skall skilja på dessa är att många round-trip requests slöar ner programmen.

---

4. Rastrering i detta sammanhang innebär att linjer, cirklar, etc, översätts från en beskrivning i matematiska termer (t.ex. linje från $P_1$ till $P_2$ eller cirkel med centrum i $P_1$ och radien r) till pixlar. Rastrering måste göras eftersom (de flesta) skärmar och hardcopy-enheter baseras på rasterteknik medan de grafiska primitiverna vanligtvis anges i matematiska termer

Problemen med att använda cirkelns ekvation är i första hand två: dels är det ineffektivt (beroende på roturdragning m.m.), dels uppstår glapp där lutningen är stor. Ett sätt att vara litet smartare är att utnyttja det faktum att en cirkel är symmetrisk: man behöver rastrera endast en åttondel av cirkeln, resten fås genom speglingar.

**Midpoint circle algorithm**: Beskriver här för den andra oktanten. Använder integeraritmetik för att från nuvarande punkt välja endera av punkten E (österut) eller SE (sydost). Man kontrollerar på vilken sida av mittpukten (M) mellan E och SE som linjen går. Metoden använder sig av inkrementella skillnader och använder en beslutsvariabel d som uppdateras olika beroende på om S eller SE väljs (*se vidare sidorna 80-84 i Foley et al*).

---

5. **Scan-line fill**

Algoritmen opererar på "spans" mellan höger- och vänsterkanten på polygonen. Spannens extremvärden uppdateras inkrementellt för varje rad. I princip:

1/ Hitta skärningspunkter mellan aktuell scan-linje och alla polygonens kanter

2/ Sortera skärningspunkterna efter ökande x

3/ Fyll i alla pixlar mellan par av skärningspunkter, används "odd-parity"-regeln för att avgöra om en punkt är inuti polygonen. Parity är jämn från början och alla skärningar inverterar parity-biten.

Horisontella kanter tas om hand genom att man helt bortser från dem.

I implementationen av algoritmen används en aktiv-kant-lista (AET) som hela tiden uppdateras från en kant-lista (ET) som initialt innehåller alla kanter sorterade i grupper (buckets)

efter y-värdet. Varje kant-entry innehåller information om ymax (min finns ju implicit genom var i ET kanten lagras) och lutning (för effektiv uppdatering)
(*se vidare sidorna 87-93 i Foley et al*)
För att fylla med ett mönster används enklast en binär matris med mönstret som läggs in m.h.a. moduloberäkningar (*se vidare sidorna 94-96 i Foley et al*).
Alternativt kan man tänka sig att beskriva **SeedFill**.

---

6. Normalt används ekvationer för att beskriva objekt. Fraktaler däremot beräknas med (oftast rekursiva och oändliga) procedurer.
Fraktal dimension: om man skalar ned ett objekt i N delar ser varje del ut som originalet skalat ned M ggr. Dimensionen D fås då ur sambandet $N^{1/D} = M$. Ex: för en linje (1D) är både N och M (och därmed D) 1. För en kvadrat (2D) är (t.ex.) N 4 och M 2 (eller N 9 och M 3). I båda fallen blir D 2.
Kochs snöflinga finns på sidan 358 i Foley et al. Dess fraktala dimension är log(4) / log(3) (~ 1.26). Fås ur $4^{1/D} = 3$.
För att erhålla naturliga objekt införs någon form av slumpmässighet, t.ex. Brownian motion eller random midpoint-displacement.
(*se vidare sidorna 358-359 i Foley et al samt den utdelat material ur Hearn-Baker*)

---

7. I alla skuggningsmodellerna vi diskuterar här används polygoner.
Konstant skuggning: hela polygonen får samma färg (beräknad m.h.a. normalen till polygonen)
Interpolerad skuggning: skuggningsinformationen interpoleras över varje polyogon beroende på kanternas normaler.
Konstant skuggning ger korrekta bilder om ljuskällan och ögat befinner sig oändligt långt från objektet och om polygonerna inte är en approximation utan beskriver objektet exakt.
Gouraud interpolerar intensiteten över polygonytan medan Phong interpolerar normalerna. Phong shading kräver mer beräkningar men ger också bättre resultat.
(*se vidare sidorna 492-496 i Foley et al*)

---

8a. Modelling Coordinates — modelling transformation ->
World Coordinates — view orientation matrix ->
View Reference Coordinates — view mapping matrix ->
Normalized Projection Coordinates — window-to-viewport transformation ->
2D device Coordinates
(*se vidare sidorna 234-235 i Foley et al*)
8b. Ray tracing ger en mycket enklare pipeline, helt enkelt därför att Ray tracingen själv gör vy-transformationen, skymda-ytor-beräkningar och ljussättning
(*se vidare sidorna 521-524 i Foley et al*)

---

9. Vanliga operationer: AND, XOR, REPLACE, OR.
REPLACE: ersätt det gamla värdet med det nya
XOR: ger tillbaka originalet om man skriver två gånger (bra för dynamik)
OR: kan användas för highlights, t.ex.
AND: kan användas till att nollställa pixlar selektivt
(*se vidare sidorna 54-56 i Foley et al*)

---

10. CRT - cathode ray tube, skickar elektronstrålar på en fosforiserande skärm, måste refreshas
    - energikrävande, tung
    + stora skärmar, hög upplösning

    LCD - liquid crystal display - flytande kristaller, bygger på att polarisera ljus m.h.a elektriska spänningar.
    - liten tittvinkel, lägre upplösning
    + platt, lätt

(*se vidare sidorna 135-141 i Foley et al*)

---