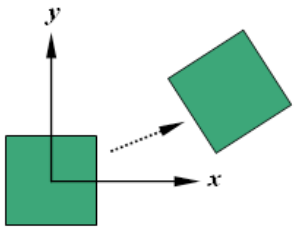


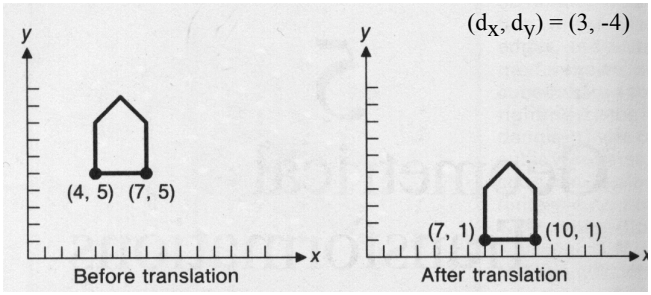
2D-grafik

Yngve Sundblad
y@kth.se
08-7907147
Rum 4621, Lindstedtsv.5, plan 6 (vid Torget)

Transformationer, Angel 4.6-4.9

- Inom datorgrafik är **transformationer** den kanske viktigaste formen av operation.
- De vanligaste transformationerna är **linjära** och kan skrivas som matriser.
- Dessa är **affina**, d.v.s. alla punkter på en linje fortsätter att ligga på linjen efter transformationen, parallella linjer bevaras, och förhållandet mellan avstånd bevaras.

Translation

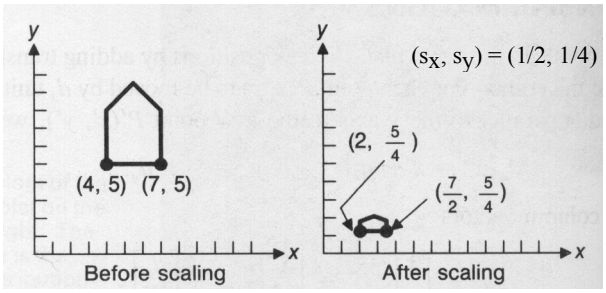


$$x' = x + d_x, y' = y + d_y$$

$$P = \begin{bmatrix} x \\ y \end{bmatrix}, P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, T = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

$$P' = P + T$$

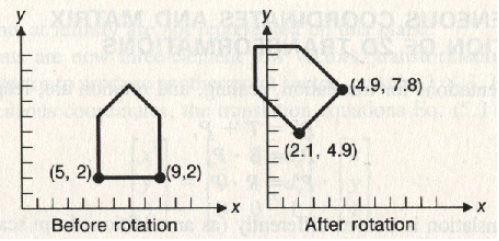
Skalning



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Skalningen sker kring origo.

Rotation



θ = π/4 (45 grader)
x' = √2/2 * 5 - √2/2 * 2 = 2.1
y' = √2/2 * 5 + √2/2 * 2 = 4.9

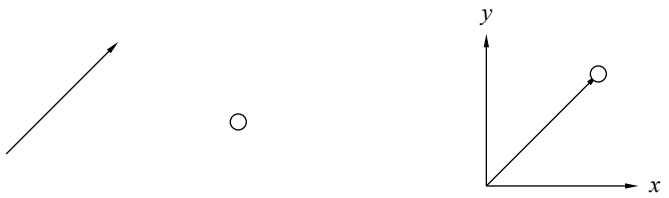
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotationen sker kring origo.

Homogena koordinater

- Det vore bra om vi kunde skriva alla dessa transformationer på ett konsekvent sätt.
- Det funkar om vi inför en tredje koordinat **w**.
- Vi säger att (x/w, y/w) är de **kartesiska koordinaterna** (cartesian coordinates).
- Definition: två punkter är lika om den enas homogena koordinater är en multipel av den andra (har samma kartesiska koordinater).
- Åtminstone en av de tre koordinaterna måste vara skild från 0.
- När w = 0 ligger punkten i oändligheten och vi definierar att (x, y) då representerar en **riktning/vektor**.

Punkter och riktningar



Riktningar (vektorer) och punkter är **olika** geometriska entiteter och existerar oberoende av referenskoordinatsystem!

Men de kan definieras i förhållande till ett sådant.
Och vi behöver ett för att kunna specificera dem med siffror.

Bastransformationsmatriser

Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Skalning

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

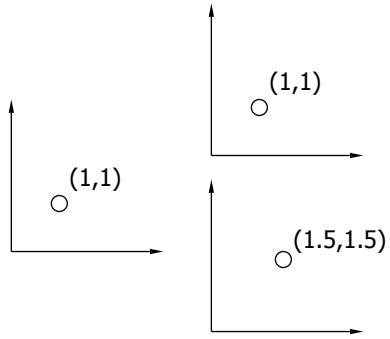
I affina transformationer ändras aldrig w-koordinaten.

Kombination av transformationer

- Vi kan nu kombinera transformationer genom att multiplicera ihop matriserna!
- Men resultatet kan bli olika om man vänder på ordningen!

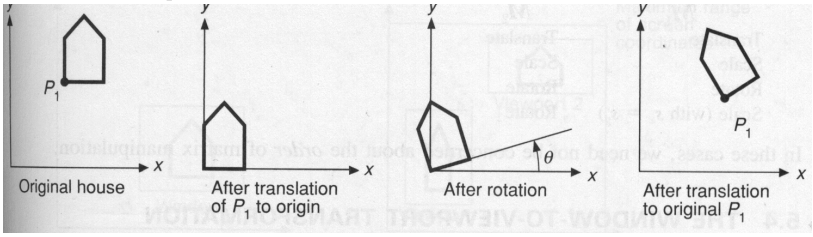
$$T = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad (S^T)P = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$S = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (T^T S)P = \begin{bmatrix} 1.5 \\ 1.5 \\ 1 \end{bmatrix}$$



Exempel

Rotera kring en annan punkt än origo:



$P_1 = (2, 5)$, $P_2 = (4,5)$, $\theta = \pi/6$ (30 grader)

Translation_1:	Rotation:	Translation_2:	Totalt: $U = T_2 R T_1$
(1 0 -2)	(√3/2 -1/2 0)	(1 0 2)	(√3/2 -1/2 9/2-√3)
(0 1 -5)	(1/2 √3/2 0)	(0 1 5)	(1/2 √3/2 4-5√3/2)
(0 0 1)	(0 0 1)	(0 0 1)	(0 0 1)

U * (2, 5, 1) = (2, 5, 1); U * (4, 5, 1) = (2+√3, 6, 1) = (2,5,1) + 2*(√3/2, 1/2, 0)

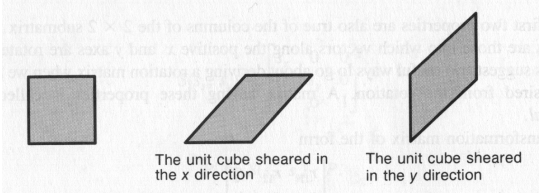
Observera att man "multiplicerar på" matriserna från vänster!

Mer om rotationsmatriser

- I en rotationsmatris är översta 2x2-delmatrisen **ortogonal,** enhetsvektorer med skalärprodukt 0.
- Riktningsvektorerna $(r_{11},r_{21},0)$ och $(r_{12},r_{22},0)$ är de två vektorer som x- resp. y-axeln är roterade till!
- Rotationsvinkeln θ, $r_{11}=r_{22}=\cos\theta$, $r_{21}=-r_{12}=\sin\theta$
- Två rotationer: θ η ger rotationen $\theta+\eta$, lätt att visa, multiplicera ihop, använd
  $\cos\eta\cos\theta - \sin\eta\sin\theta = \cos(\theta+\eta)$
  $\cos\eta\sin\theta + \sin\eta\cos\theta = \sin(\theta+\eta)$

$$R = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

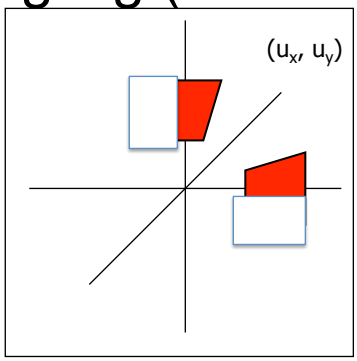
Skevning (shear)



$$SH_x = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$SH_y = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Spegling (reflection)



Där (u_x, u_y) är riktningsvektorn (med längd 1) man speglar igenom.

Ickelinjära transformationer

- Förekommer inte lika ofta som linjära, men är inte alls ovanliga.
- Exempel: Environment mapping.

$$s = \frac{R_x}{2\sqrt{R_x^2 + R_y^2 + (R_z + 1)^2}} + \frac{1}{2}$$

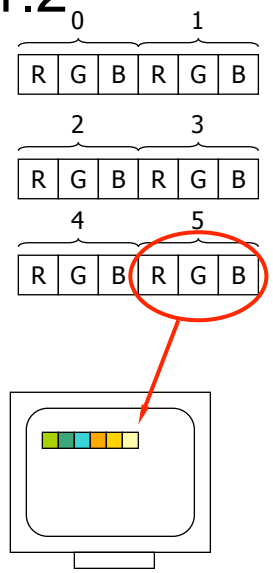
$$t = \frac{R_y}{2\sqrt{R_x^2 + R_y^2 + (R_z + 1)^2}} + \frac{1}{2}$$

Transformationer av hörn

- Det vore i praktiken omöjligt i grafik att transformera varje pixel för sig.
- Om vi använder linjära transformationer räcker det dock att transformera hörnen och binda samman dem med linjer!
- Man interpolerar sedan data definierad i hörnen linjärt över primitiverna.
- Om det är färg man interpolerar brukar det kallas **Goraud shading**.

Framebuffer, Angel 1.2

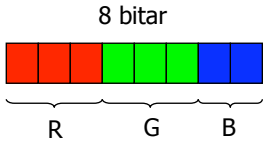
- Datorminne som lagrar information för pixlarna som ska visas på skärmen
- Grafikkortet hämtar värdena för mängden rött, grönt, och blått för varje pixel
- Styrkan hos elektronkanonen anpassas efter värdena



Pixlar

- Med **pixelvärde** menar man oftast RGB-värdena i framebuffern
- Antalet bitar för R, G resp. B-värdena bestämmer hur många färger som kan representeras

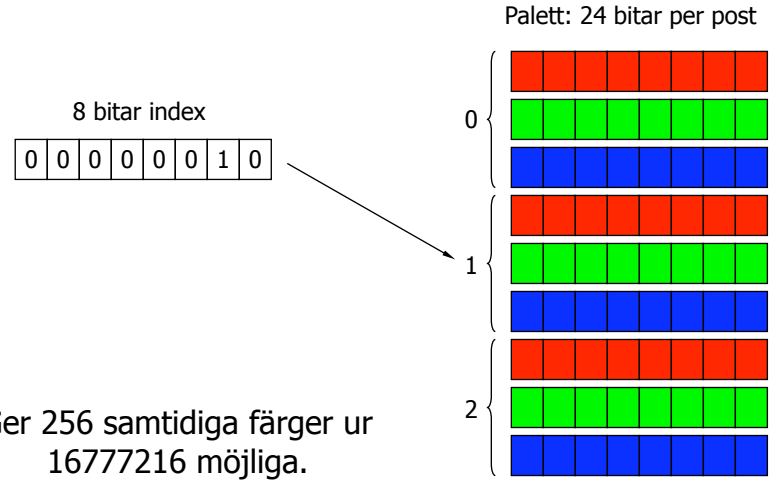
Pixlar: 8 bitar



Man tilldelar färre bitar till den
blå kanalen eftersom människan har
svårare att se skillnader mellan blå nyanser.

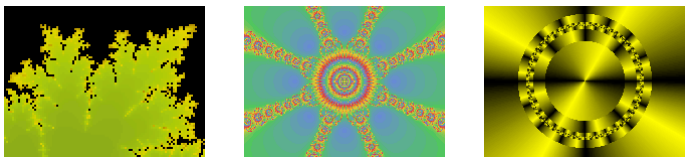
Ger $2^3+2^3+2^2 = 16$ färgnyanser.
Detta är en mycket ovanlig färgkodning idag.

Pixlar: 8 bitar palett



Ger 256 samtidiga färger ur 16777216 möjliga.

Pixlar: Color cycling

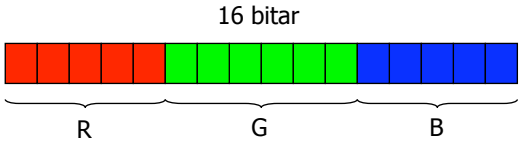


http://www2.vo.lu/homepages/phahn/fractals/cycling.htm

Shifta färgerna i paletten
(oftast med hjälp av ett offsetvärde)

Klassisk 80-talseffekt i spel!

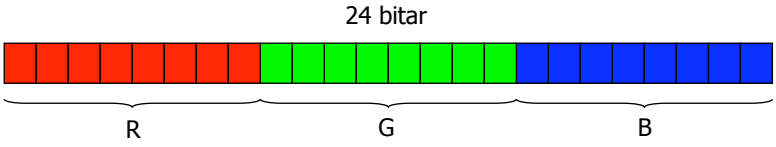
Pixlar: 16 bitar



5 bitar till R och B, 6 bitar till G eftersom
människan uppfattar skillnader i gröna nyanser bäst.

Ger $2^5+2^6+2^5 = 65536$ olika färgnyanser.
Kallas ibland **hi-color**.

Pixlar: 24 bitar



1 byte (8 bitar) till R, G, och B.

Ger $2^8+2^8+2^8 = 16777216$ olika färgnyanser.
Kallas ibland **true-color**, eftersom
människan kan uppfatta ungefär så många nyanskillnader.

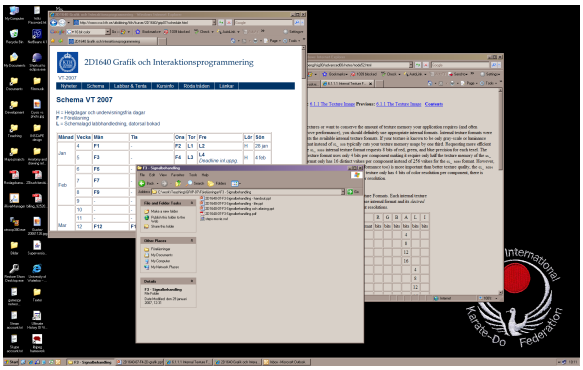
Pixlar: 32 bitar

- 8 röd + 8 grön + 8 blå + 8 alpha
- 16777216 färgnyanser +
  256 genomskinlighetsnivåer

Pixlar: mer än 8 bitar per kanal

- Konverterar du en 24-bitars bild till svartvitt är du tillbaka i 256 färgnyanser!
- Manipulation av bilder kan minska antalet signifikanta siffror eller ge avrundningsfel så att du tappar information
- Idag används ofta 12 eller 16 bitar per kanal (16x4 = 64 bit)
- Ibland t.o.m. 32 bitar per kanal (128 bit)

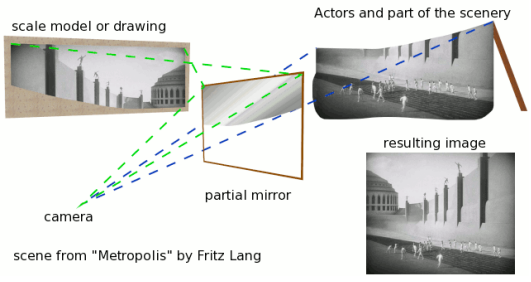
Fönstersystem



För att göra det enklare för programmeraren har oftast varje fönster en egen "framebuffer".

Dessa kombineras sedan ihop i den "riktiga" framebuffern.

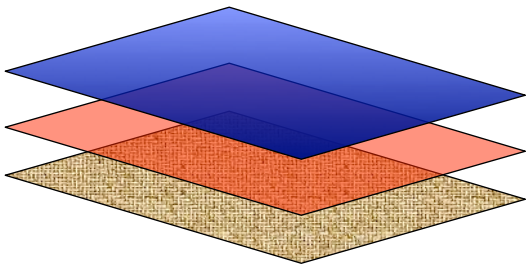
Compositing



Compositing

Georges Méliès, 1908

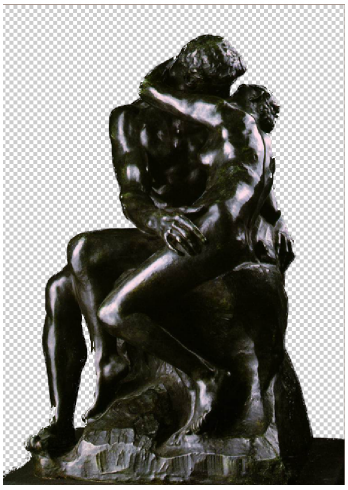
Lager



Bildbuffrar som kombineras ihop pixel för pixel nedifrån och upp.
För varje nytt lager definierar man en funktion som beskriver hur informationen från föregående lager ska kombineras ihop med aktuellt lager.

Exempel

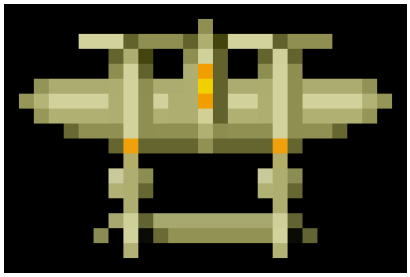
Exempel



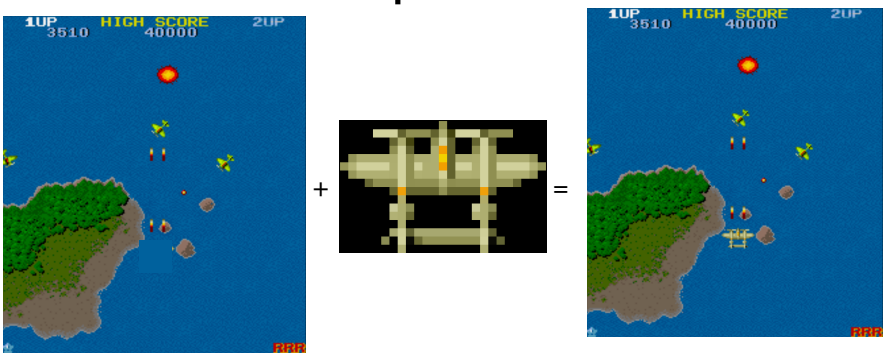
I ett spel som "1942" används inte linjer.
Vilka element består grafiken av? Hur funkar det?

Sprites

- **Sprite** är en term från 80-talet
- I färgpaletten för en 8-bitarsbild bestämmer man att en av färgerna ska vara genomskinlig
- Idag använder man alpha-kanalen istället



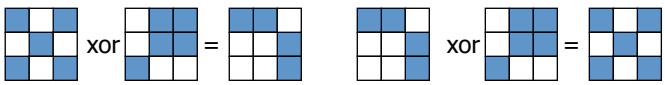
Sprites



Bilden byggs upp i lager, nerifrån och upp.
I många äldre datorer fanns en operation för att snabbt "sprite"-kopiera från minne till framebuffer, s.k. **bitblt**.
Vissa maskiner kunde skala och rotera sprites vid utritningen.
Ibland fanns stöd för kollisionsdetektion sprites emellan.

BitBlt och RasterOp

- För att kunna flytta objekt i realtid på tidiga rastergrafiska arbetsstationer, t.ex. Xerox Alto (1974) resp CMU Perq (1982) hade processorn snabba rasteroperationer, Bit Block Transfer resp RasterOp. Finns även i Java/Swing.
- De kunde flytta en rektangel i framebufferten till annat läge i logisk samverkan med det som låg där:
  AND, OR, XOR, REPLACE
  XOR upprepat användbar vid flyttningar, t.ex. markör
  z = x XOR y; z XOR x ->y



Sprites

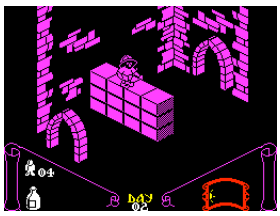
http://www.nes-snes-sprites.com/DonkeyKongCountryNew.html

Det blir **MÅNGA** sprites, ibland en för varje animationsruta!
(Donkey Kong Country för SNES hade över tusen, t.ex.)
Duktiga spritetecknare återanvände bildrutor.

Om bitblt-operationen kunde spegla bilderna
sparade man mycket minne.

Isometriska spel

Q*Bert (Arkad)



Knight Lore (ZX Spectrum)

SimCity 2000 (PC)

2.5D

Doom

En **billboard** är en polygon som
alltid vänder "framsidan" åt kameran.

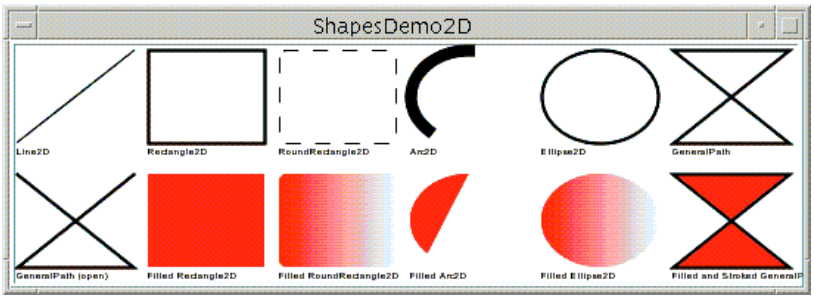
2D-grafik i Java

AWT - Abstract Windowing Toolkit - with Sun's Java 1995
Swing - Model-View-Controller GUI Tool released by
Netscape 1996, Sun 1997

```
import java.awt.*;
import javax.swing.*;
```

ger tillgång till klasserna
`Graphics` med metoder för att rita geometriska objekt med attribut som färg, XOR mode mm
`Graphics2D` med metoder för att göra affin transformation, klippa, komponera, mm mm

2D-grafik i Java



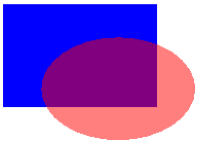
Transformationer i Java2D

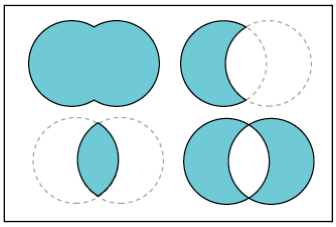
```
GeneralPath p = new GeneralPath();
p.moveTo(50, 50);
p.lineTo(150, 50);
p.lineTo(20, 100);
p.closePath();
g2.draw(p);

AffineTransform t =
AffineTransform.getTranslateInstance(100, 150);
t.rotate(-Math.PI/3.0);
AffineTransform old = g2.getTransform();
g2.transform(t);
g2.draw(p);
g2.setTransform(old);
```

Mer om Java2D

- Det finns möjlighet att använda lager och maskning i Java2D.
- Man kan också kombinera ihop primitiver automatiskt med boolska operationer.
- Stöd för att ladda och rita bitmaps/bilder finns också.





Registrering

- `res checkin grip09`
- `course join grip09`

Inlämningsuppgift 2

- 2D-grafik med transformationer
- Flertalet uppgifter matematiskt betonade kring transformationer

Finns på kurswebben under fliken Inl.uppg., till y@kth.se senast 6 februari

Individuell inlämningsuppgift 2 till GrIP-kursen vt 2009

2-D-grafik med transformationer

Skicka denna blankett ifylld senast fred.6 februari kl. 23.59 till y@kth.se

För resultatet 1 (godkänd) krävs totalt minst 8 poäng på talen, för 2 (väl godkänd) minst 12 poäng, för 3 (mycket väl godkänd) minst 16 poäng.

Resultatet på inlämningsuppgiften vägs ihop med resultatet på uppgift 3 (3D transformationer mm) & 4 (projektet) till slutbetyget A-F på kursen.

Ditt namn:

Personnummer:

Tal 1 (4 poäng)
a. Hur ser homogena koordinaterna för punkten (17, -3) ut?
b. Vad vinner man på att använda tredimensionella homogena koordinater istället för tvådimensionella punkter vid transformationer? Ge exempel.

Tal 2 (6 poäng)
Utred, med matematisk motivering, under vilka förutsättningar bastransformationerna kommuterar, dvs när ger
a. en translation och en rotation samma resultat oberoende av i vilken ordning de görs
b. en rotation och en skalning samma resultat oberoende av i vilken ordning de görs
c. en skalning och en translation samma resultat oberoende av i vilken ordning de görs
Tänk på att inte bara ange resultat som formler utan också att ”tolka” dem.

Tal 3 (4 poäng)
Visa med transformationsmatriserna att spegling kan åstadkommas med hjälp av successiv användning av vissa av bastransformationerna translation, rotation och skalning.
Tips: Vad betyder negativ skalning?

Tal 4 (6 poäng)
Ange deltransformationsmatriserna som åstadkommer följande totala transformation av hörnpunkterna i den vänstra rektangeln till den högra. Hur ser den totala transformationsmatrisen ut?
p = 1 + sista siffran i din födelsedag
q = p + 1 + sista siffran i din födelsemånad
Nedre vänstra hörnet i högra (resultat-)figuren har x-koordinaten p-1.