



Grunderna i C++

ARK 385: *Virtuella Verktyg i en Materiell värld*

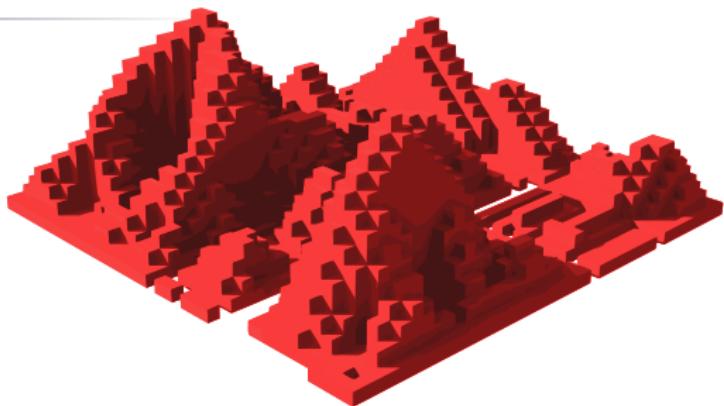
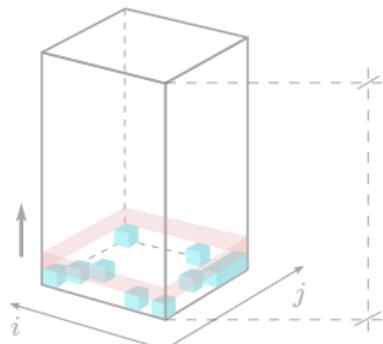
AT Arkitektur & Teknik

Chalmers Tekniska Högskola

2009 - Kursen skapades (3 förel.)

2010 - 6 förel. + 2 projekt

2011 - 8 förel. Helt omarbetade övningar



Skapad av Matz Johansson Bergström

@ matz.johansson@chalmers.se

Innehållsförteckning I

① Wrap-up

Repetition

Schema

Vad vi går igenom idag

② C++ del 4

Nästlade for-slingor

Introduktion till OpenGL

GLUT

Användning av OpenGL

DXF

Syntax

Kaos

Game of Life

L-System (extra)

Övningar

Vad lärde vi oss förra gången?

- Slumptal
- Skrivning till fil
- for loopar

Uppdaterat schema

Kursen har fått två extra tillfällen. Det nya schemat:

Tillfälle	Datum	Övningar
1	5/10	1
2	12/10	2
3	2/11	3
4	9/11	4
5	14/11	5
6	16/11	Processing/Projekt 1
7	21/11	Processing/Projekt 2
8	23/11	Buffert

Om ni har håltimmar och är överrens att vi flyttar kursen någon timma eller två så är det ok för mig.

Vad lär vi oss idag?

Idag:

- DXF formatet
- Nästlade for-loopar
- Lite om OpenGL
- Cellulära automater (Game of Life)
- L-System (extra)

Nästlade for-loopar

Vi har tidigare gått igenom While-slingor, sedan ett sätt att konvertera detta till for-slingor. Nu skall vi se exempel på hur man kan använda nästlade for-slingor, dvs. for-slinga *inuti* en annan for-slinga.

Säg att vi vill utföra en vanlig Matlab uträkning, matrix-vektor multiplikation $Ax = b$, i C++. Följande visar hur vi gör i Matlab:

```
A = [1, 2;
      3, 4;
      5, 6];
x = [2, 3]';
b = zeros(1, 3);
>> b = A*x
```

Matrisberäkning i Matlab

Matlab är speciellt utformat för att göra detta snabbt och enkelt.

Matriser med nästlade for-slingor

En matris i C++ kan skrivas som ett 2-dimensionellt fält. Notera att vi inte (behöver) transponera **b**.

```

int A[2][2] = { {1, 2},
                {4, 5} };

int x[2] = {2, 3};
int b[2] = {0, 0}; //måste initiera annars får vi skräp ...
                    här
    
```

2d fält i C++.

För själva matris-vektor multiplikationen kan vi tänka i flera steg, nästa sida visar själva idén bakom slingorna och hur man kan tänka när man skriver kod.

Indexing är nu **a[i][j]** där **i** avser rad och **j** kolonn, så **a[0][1]** ger elementet 2 (OBS: index från 0).

$$\underbrace{\begin{pmatrix} A[0][0] & A[0][1] \\ A[1][0] & A[1][1] \end{pmatrix}}_A$$

Så, för matris/vektor multiplikation med följande symboler får vi

$$\underbrace{\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}}_A \underbrace{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_x = \underbrace{\begin{pmatrix} \alpha x_1 + \beta x_2 \\ \gamma x_1 + \delta x_2 \end{pmatrix}}_b$$

- I hur många dimensioner måste vi "stega"? 2, kolonn och rad i matrisen A .
- Vi går igenom x :s element för varje rad i resultatet $b \rightarrow j$
- Vi går igenom kolonnerna i A , alltså $\mathbf{A[?][j]}x[j]$.
- Om vi tittar på b så ser vi $\mathbf{b[0]}=\alpha x_1 + \beta x_2$
- Vi ser att index i b beror på raden i $A \rightarrow i$.
- $\mathbf{b[i]} = \mathbf{A[i][j]}x[j]$
- Men nu får vi endast $b[0] = \beta x_1$ Vi måste summera a:s kolonnelement med x:s radelement
- $\mathbf{b[i]} += \mathbf{A[i][j]}x[j];$

```
for (int i=0; i<2; i++)
    for (int j=0; j<2; j++)
        b[i] += A[i][j]*x[j];
```

OpenGL

Open Graphics Library

En specifikation/standard som innehåller över 250 funktioner för att rita 2d och 3d primitiver samt ljussättning (ej skuggor)...

- Skapades av Silicon Graphics 1992
- Används idag i CAD, VR, Vetenskaplig visualisering, spel...
- Abstraherar bort från grafikhårdvaran
- Väldokumenterat (the red book) [WWW](#)

GLUT

För att kunna utnyttja standarden OpenGL måste vi ladda ner ett bibliotek med rutiner som stöder vår maskinvara, en av dessa heter GLUT.

Om GLUT:

...is a OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs.

GLUT är alltså en samling verktyg som gör att vi kan skriva OpenGL-kod i Windows (och andra plattformar).

Det första vi måste göra är att importera biblioteket för GLUT:

```
#include <GL/glut.h>
```

Lite om användning

Jag kommer bara förklara grunderna i OpenGL (eftersom det är så stort).

För att kunna rita nåt i fönstret måste vi definiera vad vi vill rita, detta gör man med `glBegin(mode);` och `glEnd(mode);`.

mode är en konstant som beskriver hur punkterna man ger OpenGL skall tolkas. mode = **GL_QUADS**, **GL_POINTS** eller **GL_LINES** (finns fler)

Man definierar "noder" med:

glVertex3f(x,y,z)

För att det skall se lite trevligare ut kan vi färglägga dessa "noder":

glColor3f(R,G,B) där $R, G, B \in [0, 1]$ (**Rött Grönt Blått**).

Vi skall prata mer om detta i nästa föreläsning där vi kommer ägna oss åt ett mini-projekt och göra detta i **OpenGL** alternativt rendera resultatet i Rhino.
Se även: [WWW](#).

DXF

DXF (Drawing Interchange Format) är ett format som skapades av Autodesk för att kunna importera/exportera data mellan program.

DXF används till de flesta program som finns i Workshoppen således kursens projekt. Anledningen till att vi använder DXF är att det är lätt att skriva eftersom det är vanliga bokstäver, till skillnad från andra format som kan vara binär kod. Man skulle kunna skriva en vanlig fil och döpa det till .DXF och importera i AutoCAD.

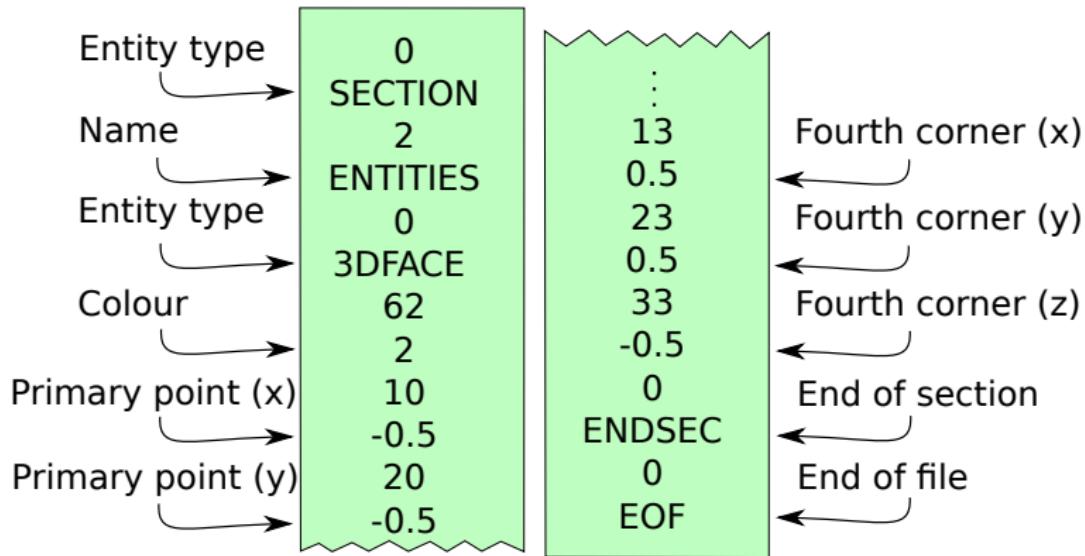
Nu använder inte vi AutoCAD utan Rhino, men den klarar formatet galant.

Vi har några enkla saker som vi vill ändra på:

- Layer (sorts gruppering)
- Color (vi kan inte rendera dock)
- Lines
- 3dfaces (polygoner)

För mer information om själva formatet gå in på [WWW](#)

Syntax



Se kapitlet "Group codes in numerical order" i manualen för betydelser. Under kapitlet "3D face" står det om koder för koordinater. Att använda färger är inte helt lätt då de är index till en färgkarta.

Cellulära Automater

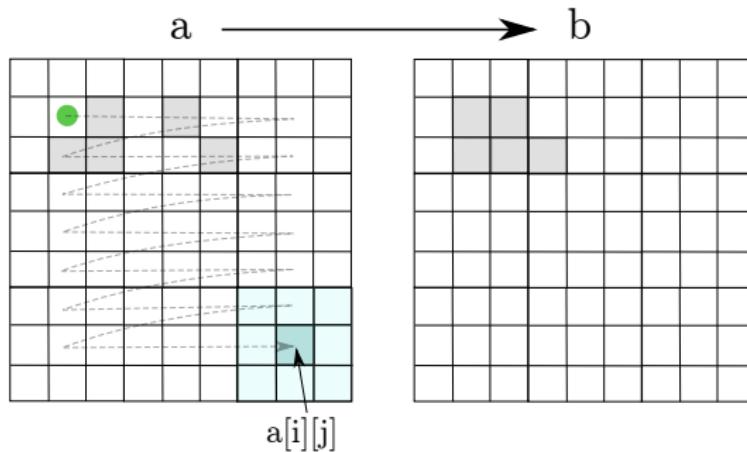
Till dagens övning introduceras ni till cellulära automater. En cellular automata (**CA**) är en matris som ofta består av binära värden (ettor och nollor). Men kan även bestå av flyttal.

Ett välkänt exempel på en binär **CA** är Conways "Game of Life" som bygger på några enkla regler där själva automaten genererar en animation som påminner om levande celler (i biologisk mening).

- ① Any live cell with fewer than two live neighbours dies, as if caused by under-population.
- ② Any live cell with more than three live neighbours dies, as if by overcrowding.
- ③ Any live cell with two or three live neighbours lives on to the next generation.
- ④ Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

Reglerna till “Game of Life”

Game of Life är ett fascinerande system, man kan ur enkla regler skapa kaotiska mönster. Illustrationen nedan visar ett steg i en generation. En generation är en frame i animationen som skapas.



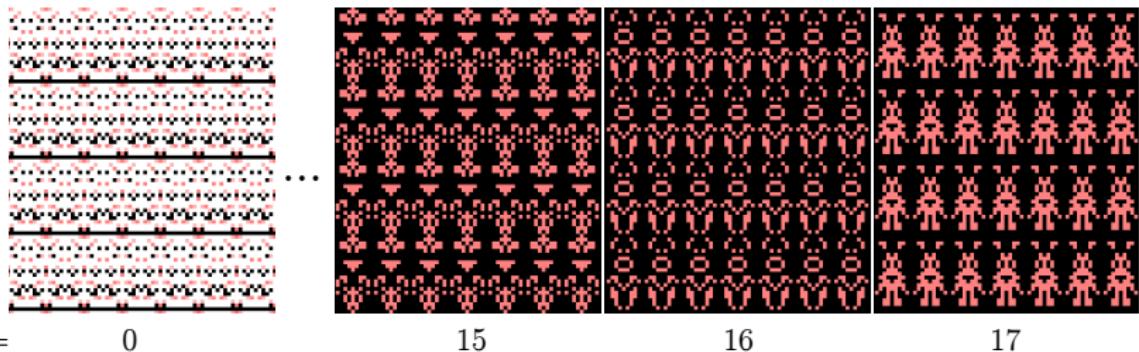
Scanna matrisen och uppdatera b med de nya värdena. Kopiera över a efter varje generation.

Vi läser **a** och skriver till **b**, sedan måste man kopiera över till **a** igen inför nästa generation, se WwW.

Game of Life

Exempel på en variation av GoL

$$a[i][j] = \sqrt{\text{mod}(i^2 + ij^2, 20)} \text{ för alla } i, j$$



$i =$

0

15

16

17

Generationer av GoL med en speciell startdistribution som genererar uppradade kaniner. En granne i detta exempel är ett element med värdet >0.9 .

Avslutande ord om funktioner

Skriver vi en funktion som inte returnerar någonting (finns ingen return i) använder vi nyckelordet **void**.

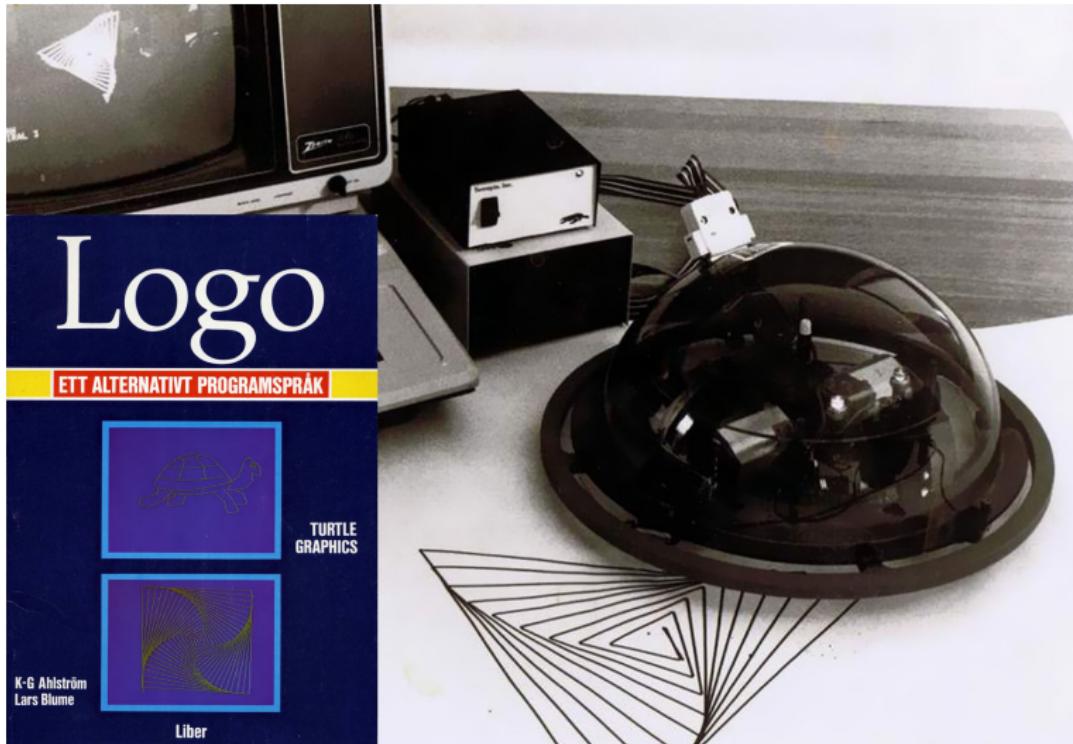
```
void A_clear_row( int i )
{
    for( int j=0; j<SIZE; j++)
        A[i][j]=0;
}
```

Exempel från övningen idag.

I detta fallet modifierar vi **A** som är ett globalt 2-dimensionellt fält.

L-System

...med turtle graphics



Used for educational purposes with permission of Denning Branch International.

L-System

Ett rekursivt system skapat av Biologen *Aristid Lindenmayer* (1968) som bygger på en *parallel omskrivning av instruktioner*. Idén var först att ge en formell modell för att beskriva strukturen hos algerⁱ.

L-System bygger på följande "regler":

Start - Begynnelsesträng.

Regler - Mönster för omskrivning av variabler.

Variabler - Ersättande tecken.

Konstanter - De tecken som inte kan ersättas (i Regler).

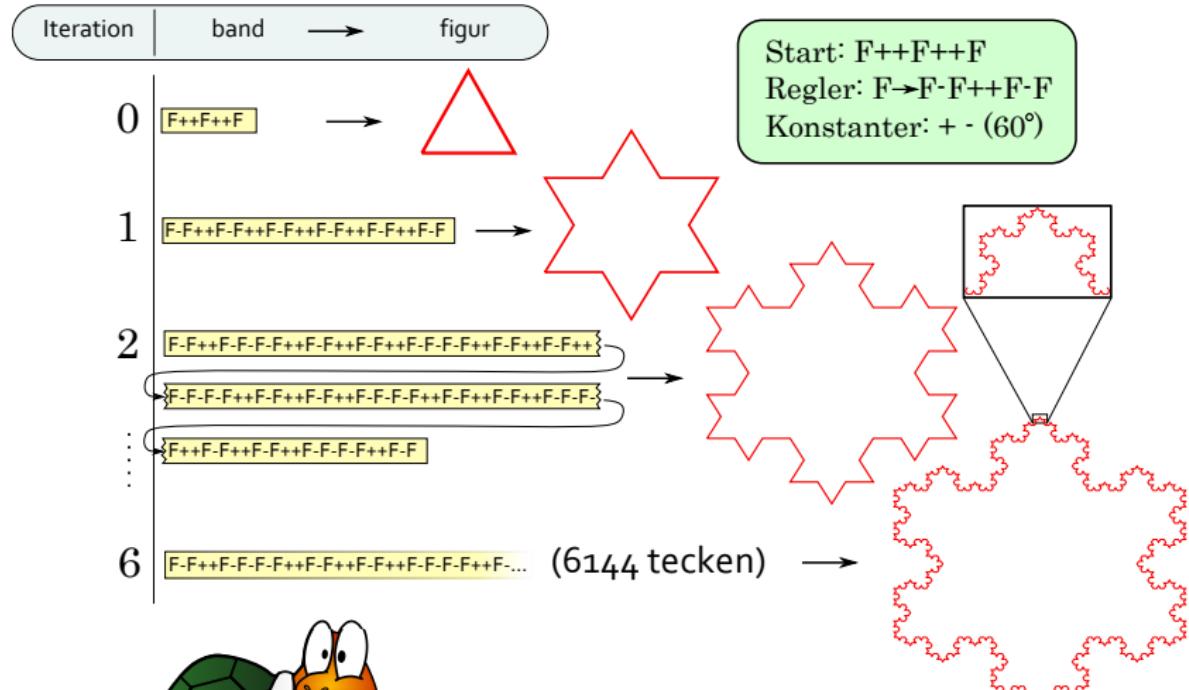
För följande system – som för övrigt är en fraktal – använder vi:

- Start: F++F++F
- Regel: $F \rightarrow F-F++F-F$
- Variabler: F ("rita ett steg framåt")
- Konstanter: +,- ("rotera höger resp. vänster")
- Vinkel: 60°

ⁱKallas för "bracketed L-Systems" vilket vi inte kommer gå igenom.

L-System

Exempel



Övningar

- Ni behöver inte logga in för att hämta ner från kurshemsidan [WWW!\[\]\(37c9b59beadc0efda7b2e0764c2d2e4f_img.jpg\)](#)
- Öppna **Instruktioner till Övning 4.pdf** (under Övningar/4/)
- Öppna **Referensblad.pdf** (under Extra/)
- Om ni undrar något så kolla i referensbladet, där står allt ni behöver veta.
- Är jag upptagen så kolla på ledtrådarna, annars kan ni skriva upp er på tavlan så tar jag er i tur och ordning. Ni som är snabba får gärna hjälpa övriga.