

# Integrationsmetoder

- Datorspel är tidsdiskreta!
- Explicita analytiska funktioner för hastighet och acceleration saknas

Position är integral av hastighet

Hastighet är integral av acceleration

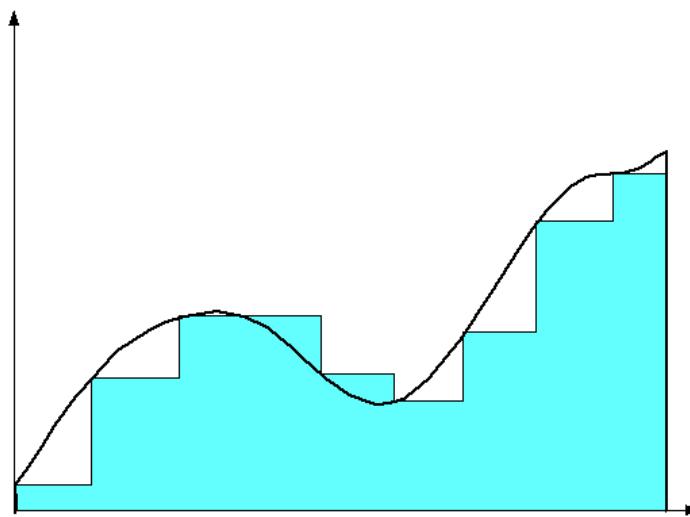
Eulerintegrering är exakt endast för konstant funktion!

Ingemar  
Ragnemalm  
[ingis@isy.liu.se](mailto:ingis@isy.liu.se)

## Eulerintegrering

Samplar funktionsvärdet och addrar

Eulerintegrering är exakt endast för konstant funktion!



Ingemar  
Ragnemalm  
[ingis@isy.liu.se](mailto:ingis@isy.liu.se)

# Eulerintegrering

$$x_{i+1} = x_i + v_i * dt$$

$$v_{i+1} = v_i + a * dt$$

Risk för instabilitet. Ju snabbare en funktion varierar, desto större risk. Fjädersystem kan komma i oscillerande tillstånd.

Felet minskar vid kortare steglängd. I praktiken innebär detta antingen högre framerate eller att man stegar mer än en gång per frame.

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Taylorexpansion

Med Taylorexpansion av funktionen kan vi analytiskt hitta bättre metoder

$$x(t+h) = x(t) + h * dx/dt + h^2/2! * d^2x/dt^2 \dots$$

Eulerintegrering är enbart de två första termerna. Ger ett fel på  $O(h^2)$ , vilket är mycket.

Andra metoder:

Verletintreggrering  
Adams-Bashforth  
Predictor-corrector  
Runge-Kutta  
Adaptiv steglängd

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Verletintegration

Hastigheten (funktionens förstaderivata) integreras inte utan beräknas från positionerna.

Tag ett steg i varera riktningen:

$$x(t+h) = x(t) + h \cdot \frac{dx}{dt} + \frac{h^2}{2} \cdot \frac{d^2x}{dt^2} + \frac{h^3}{6} \cdot \frac{d^3x}{dt^3} + O(h^4)$$

$$x(t-h) = x(t) - h \cdot \frac{dx}{dt} + \frac{h^2}{2} \cdot \frac{d^2x}{dt^2} - \frac{h^3}{6} \cdot \frac{d^3x}{dt^3} + O(h^4)$$

Addera dessa två, lös ut  $x(t)$ :

$$x(t+h) = 2 \cdot x(t) - x(t-h) + h^2 \cdot \frac{d^2x}{dt^2} + O(h^4)$$

$O(h^3)$ -termen försvann!

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Verletintegration

blir alltså

$$x(t+h) = 2x(t) - x(t-h) + a(t)$$

Kan även uttryckas med hastighet “velocity Verlet”

Att inte integrera hastigheten har också praktiska fördelar. Man slipper justera hastigheten i kollisionsupplösning.

Till synes ytterst liten skillnad, i praktiken mycket bättre.

# Adams-Bashforth

Tag med ett led till i Taylorserien.

$$x_{i+1} = x_i + \frac{1}{2} * h(3v_i - v_{i-1})$$

Kräver att vi känner hastigheten i de två tidigare tillstånden, den är inte "självstartande". Startas lämpligtvis med ett Eulersteg, första ordningens steg.

Ger ett fel på  $O(h^3)$  i stället för  $O(h^2)$

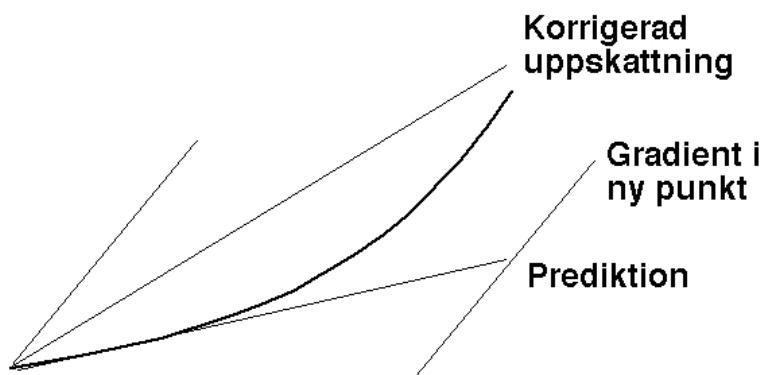
Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Predictor-corrector

Beräkna först ett preliminärt steg, prediktionen.

Beräkna gradienten i prediktionen.

Gör sedan en ny beräkning där medelvärdet mellan gradienterna i de två punkterna används.



Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Runge-Kutta

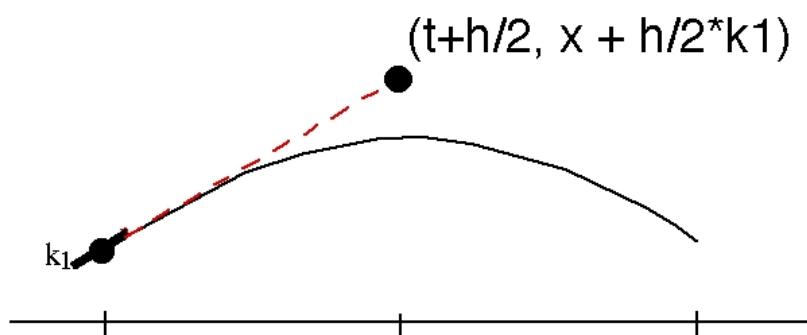
Kan ses som utvidgning av Predictor-corrector. Intervallet delas upp i flera steg, flera graderter används.

Ingemar  
Ragnemalm  
[ingis@isy.liu.se](mailto:ingis@isy.liu.se)

## Runge-Kutta, steg 1

Tag derivatan av funktionen, följ den ett halvt steg fram:

$$k_1 = x'(t, x)$$

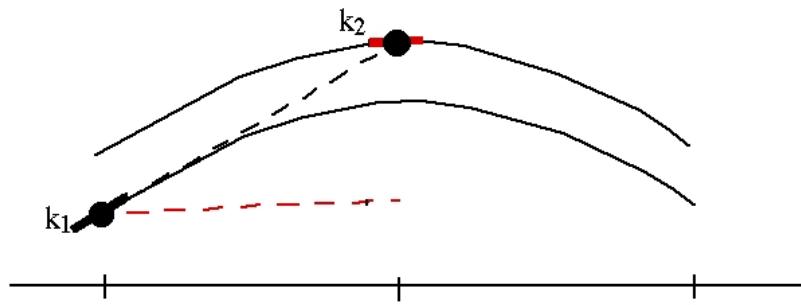


Ingemar  
Ragnemalm  
[ingis@isy.liu.se](mailto:ingis@isy.liu.se)

# Runge-Kutta, steg 2

Tag derivatan av funktionen i den nya positionen, backa, följ den ett halvt steg fram:

$$k_2 = x'(t+h/2, x + h/2*k_1)$$

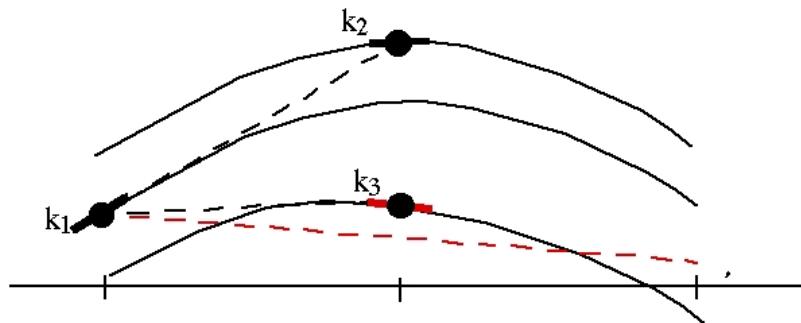


Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Runge-Kutta, steg 3

Och en gång till...

$$k_3 = x'(t+h/2, x + h/2*k_2)$$



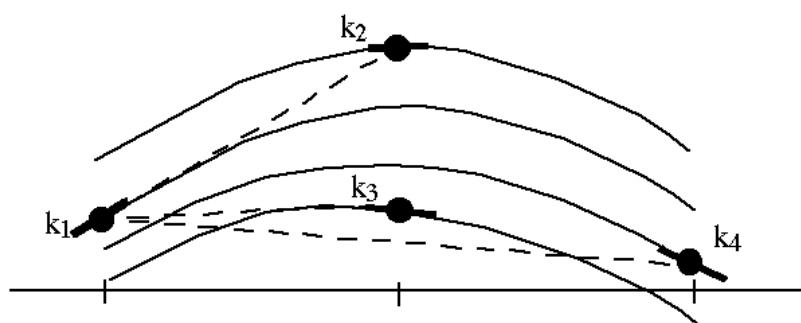
Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Runge-Kutta, steg 4

Sista gången tar vi ett helt steg fram.

$$k_4 = x'(t+h, x + h^*k_3)$$

$$x(t+h) = x(t) + h^*(k_1 + 2^*k_2 + 2^*k_3 + k_4)/6$$



Ingemar  
Ragnemalm  
[ingis@isy.liu.se](mailto:ingis@isy.liu.se)

## Adaptiv steglängd

Steglängden  $h$  påverkar direkt felet, men felet är också lågt när funktionens högre derivator varierar långsamt.

Om vi varierar steglängden efter behov så kan felet hållas nere. Något felmått måste beräknas för att styra detta.

Ingemar  
Ragnemalm  
[ingis@isy.liu.se](mailto:ingis@isy.liu.se)

# **Slutsatser om integration**

**Viktig del av speciellt spelfysik. Tidsdiskret fysik -> integration av acceleration/hastighet i diskreta steg!**

- Euler lätt, men ger stora fel
- Verlet enkel metod för mycket bättre resultat
- Runge-Kutta böligare, mer beräkningstung, men ger fina resultat.

## **Ännu mer bump mapping**

**Supplement 1 (om bump mapping, normal mapping och parallax mapping) kommer upp på kurshemsidan i kväll!**

- Mer om koordinatsystemen
- Bump mapping i olika koordinatsystem
  - Normal mapping
  - Parallax mapping

# Grafiklabb på fredag

med dugga på grafikdelen

Labben handlar om

- High dynamic range
- Bump mapping

Labbmaterialet inte klart än

Ingemar  
Ragnemalm  
[ingis@isy.liu.se](mailto:ingis@isy.liu.se)

## Basvektorerna för bump mapping

Normalvektorn kommer från OpenGL

Transformeras till vykoordinater:

$$\mathbf{n} = \text{gl_NormalMatrix} * \text{gl_Normal}$$

Men de andra två...?

$\mathbf{V}_s, \mathbf{V}_t$  basvektorer för texturen, i  
modellkoordinater

Ingemar  
Ragnemalm  
[ingis@isy.liu.se](mailto:ingis@isy.liu.se)

## Olika sätt att beräkna $V_s$ , $V_t$

- Enkel demopolygon: Ges direkt av geometrin
- Egen generering av texturkoordinater: Kommer närmast på köpet
- Fulhack för demobruk: kryssa  $n$  med vad som helst!

Snygg lösning: Beräkna differentiale baserat på vertexdata. (Dietrichs metod, fö 4)

$$V_s = (dx/ds, dy/ds, dz/ds)$$
$$V_t = (dx/dt, dy/dt, dz/dt)$$

Ingemar  
Ragnemalm  
ingis@isy.liu.se

## Transformera $V_s$ , $V_t$

$$P_s = \text{gl_NormalMatrix} * V_s$$
$$P_t = \text{gl_NormalMatrix} * V_t$$

så vi är i vykoordinater!

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Koordinatsystem

- Modellkoordinater
  - Vykoordinater
  - Texturkoordinater (tangentkoordinater)

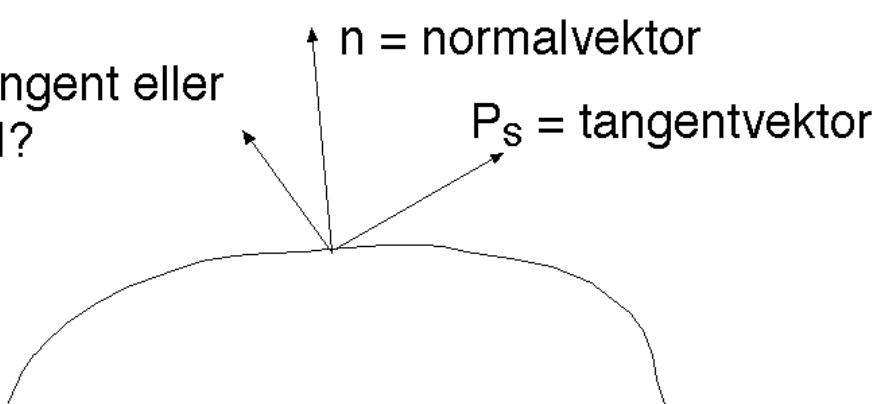
**gl\_NormalMatrix** går från modellkoordinater till vykoordinater.

$$M_{vt} = \begin{bmatrix} \mathbf{P}_s \\ \mathbf{P}_t \\ \mathbf{n} \end{bmatrix} = \begin{bmatrix} P_{sx} & P_{sy} & P_{sz} \\ P_{tx} & P_{ty} & P_{tz} \\ n_x & n_y & n_z \end{bmatrix}$$

transformerar från vykoordinater till texturkoordinater!

Ingemar  
Ragnemalm  
ingis@isy.liu.se

$P_t$  = bitangent eller binormal?



# Texturkoordinater/tangentkoordinater?

Nästan samma sak.

Texturkoordinater inte nödvändigtvis  
ortogonala

Ingemar  
Ragnemalm  
[ingis@isy.liu.se](mailto:ingis@isy.liu.se)

## Modifiering av normalvektor i vykordinater

Togs upp kort i PFNP

$$\begin{aligned} \mathbf{bs} &= \mathbf{db}/ds = \mathbf{b}[s+1, t] - \mathbf{b}[s, t] \\ \mathbf{bt} &= \mathbf{db}/dt = \mathbf{b}[s, t+1] - \mathbf{b}[s, t] \end{aligned}$$

$$\mathbf{n}' = \mathbf{n} + \mathbf{bs} \cdot \mathbf{Ps} + \mathbf{bt} \cdot \mathbf{Pt}$$

eller

$$\mathbf{n}' = \mathbf{n} - \mathbf{bs} \cdot \mathbf{Ps} - \mathbf{bt} \cdot \mathbf{Pt}$$

Ingemar  
Ragnemalm  
[ingis@isy.liu.se](mailto:ingis@isy.liu.se)

# Modifiering av normalvektor i texturkordinater

Samma sak men mycket enklare!

$\pm bs$   
 $\pm bt$   
1

MEN vi måste också transformera ljusriktning och betraktningsriktning till texturkoordinater!

Ingemar  
Ragnemalm  
ingis@isy.liu.se

## Normalmappning

Utförs i texturkoordinater!

Förberäkning:

$(bs, bt, 1)^t$  förberäknas till textur!

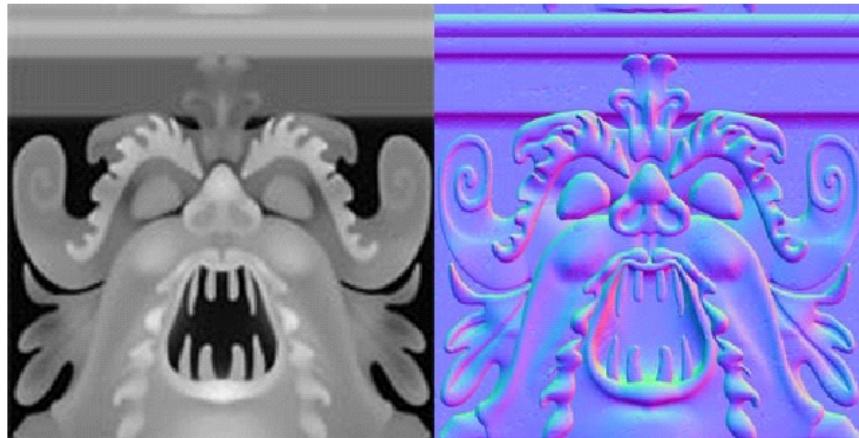
$$\begin{aligned}-bs &= b[s, t] - b[s+1, t] \\-bt &= b[s, t] - b[s, t+1]\end{aligned}$$

Normeras till bilddata [0..1]:

$$\begin{aligned}R &= (x+1)/2 \\G &= (y+1)/2 \\B &= (z+1)/2\end{aligned}$$

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Resulterande förberäkning:



bump map

normal map

Ingemar  
Ragnemalm  
ingis@isy.liu.se

## Användning

Uppackning:

$$\begin{aligned}x &= 2R - 1 \\y &= 2G - 1 \\z &= 2B - 1\end{aligned}$$

$$n_t = (x, y, z).$$

Transformera ljusriktning och  
betraktningsriktning till texturekoordinater:

$$L_t = M_{vt} \cdot L$$

och vi har allt som behövs för ljussättning!

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Parallaxmappning

Transformera betraktningsriktning till texturkoordinater:

$$V = M_{vt} \cdot V_v$$

Dela upp i  $V_{xy}$  och  $V_z$

Parallaxmappning får med

$$T_n = T_0 \pm b(i, j) \cdot V_{xy} / V_z$$

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Parallaxmappning

Variant: “offset limiting”

$$T_n = T_0 \pm b(i, j) \cdot V_{xy}$$

Tar helt enkelt bort nämnaren!

Ser plattare ut i branta vinklar.

Varför är det bättre?

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# **Supplement 1**

## **ombump mapping/normal mapping**

### **kommer på kurssidan i kväll!**

Ingemar  
Ragnemalm  
[ingis@isy.liu.se](mailto:ingis@isy.liu.se)