Från datorgrafik till animation

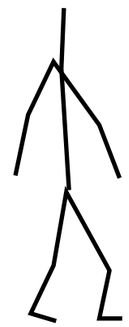
Fö 2-4: 100% grafik
Fö 5: GPUs för annat än grafik
Fö 6: Animation utan fysik

Fö 7-9: Fysik och spelfysik

Fysikbaserad animation

Character animation

- Fördefinierade poser
- Key-frame-animation
- Forward kinematics
- Inverse kinematics
- Fysikaliskt baserad animation
- Motion capture



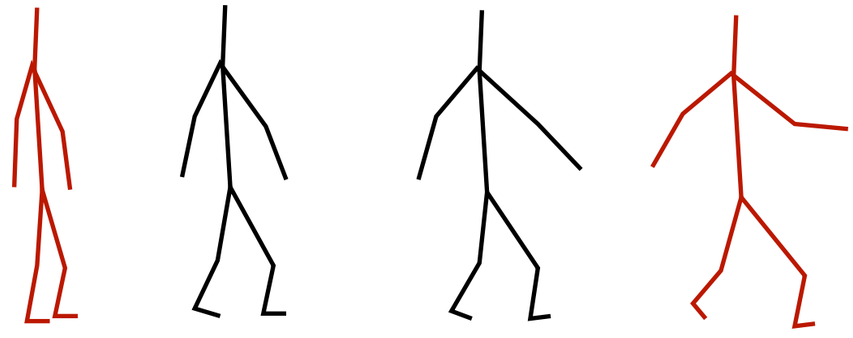
Key-frame-animation

Förrenderade animationer

Key-frames på lämpliga intervall

Frames mellan keyframes interpoleras (morphas)

Mycket vanlig metod för realtidsanimation



Kinematik

Rörelse utan krafter.

Forward kinematics: Ange vinklar, position ges av vinklarna. Lätt att implementera, svårt att använda.

Inverse kinematics: Ange position, räkna ut vinklar för att uppnå denna position.

Viktigt i robotik.

Invers-kinematik

- Analytisk beräkning
- Iterativ beräkning med inversa Jacobianen
- CCD (Cyclic coordinate descent)

Invers-kinematik analytiskt

Önskad slutposition anges

Varje led har ett antal frihetsgrader (vinklar)

Överbestämt, olinjärt system

Fungerar för små system, ohanterligt för stora.

Iterativ metod

Beräkna derivata av alla utdata m.a.p. invärden.

Ger en NxM-matris - Jacobianen.

Denna beskriver framåt-kinematiken lokalt.

$V = J\theta'$

V derivata av utdata (position), θ' derivata på indata

$J^{-1}V = \theta'$

ger oss derivatan på vinklarna.

Iterativ metod

Jacobianen normalt inte symmetrisk! Ej inverterbar!

Pseudoinversen används.

Även en variant baserad på transponat av Jacobianen finns.

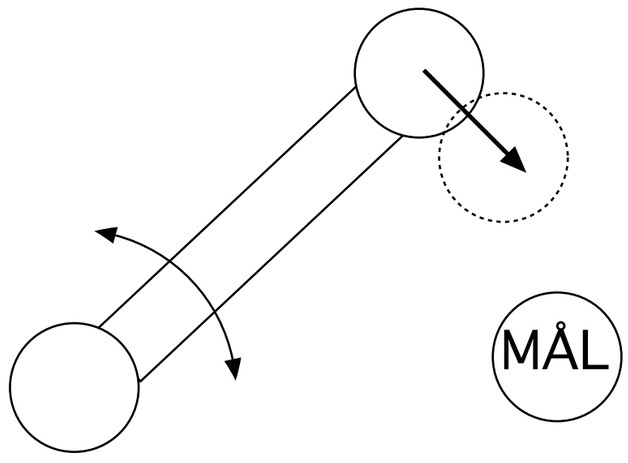
Iterativ metod (informellt)

Målposition m
Beräkna nuvarande position cp
Beräkna position med litet steg för varje vinkel i: cpd_i
Skillnad mellan avstånd mål-ny position och mål-nuvarande position

Möjliga nya vinklar tr_i

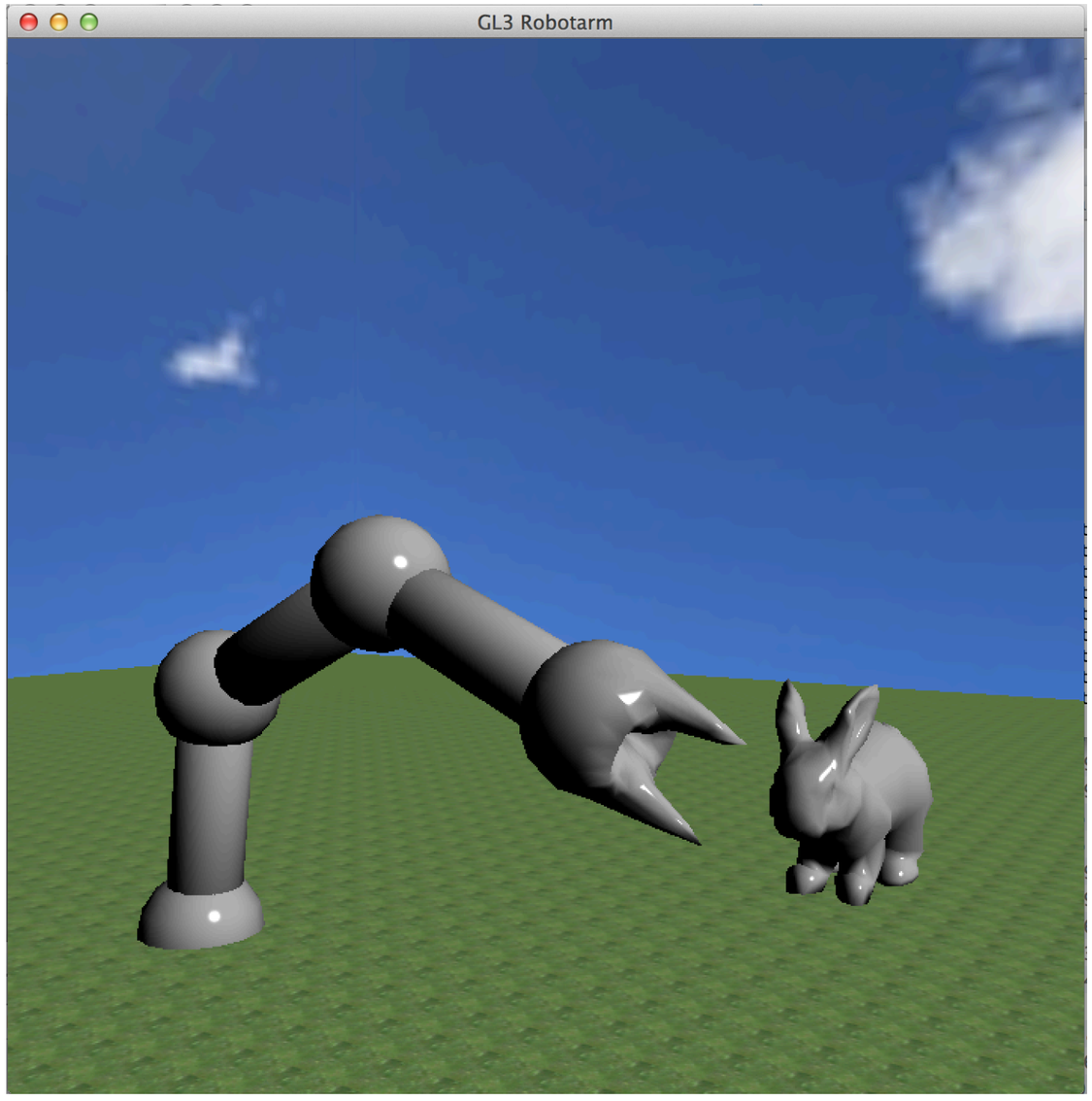
$tr_i := r_i + | m-cp | - | m - cp_i |$

Vad gjorde jag? Transponat av Jacobianen!



```
        vec3 cp = ClawPosition(r0, r1, r2, r3, r4);
        vec3 cpd0 = ClawPosition(r0+diff, r1, r2, r3, r4); // Claw position if
r0 is changed
        vec3 cpd1 = ClawPosition(r0, r1+diff, r2, r3, r4);
        vec3 cpd2 = ClawPosition(r0, r1, r2+diff, r3, r4);
        vec3 cpd3 = ClawPosition(r0, r1, r2, r3+diff, r4);
        vec3 cpd4 = ClawPosition(r0, r1, r2, r3, r4+diff);

    // cpd 0-4 - bpd is the Jacobian! Describes the change for xyz for
each set of angles.
    // Change angle in the direction that improves
        float tr0 = r0 + Norm(goal - cp) - Norm(goal - cpd0);
        float tr1 = r1 + Norm(goal - cp) - Norm(goal - cpd1);
        float tr2 = r2 + Norm(goal - cp) - Norm(goal - cpd2);
        float tr3 = r3 + Norm(goal - cp) - Norm(goal - cpd3);
```



Iterativ metod, robotarmen

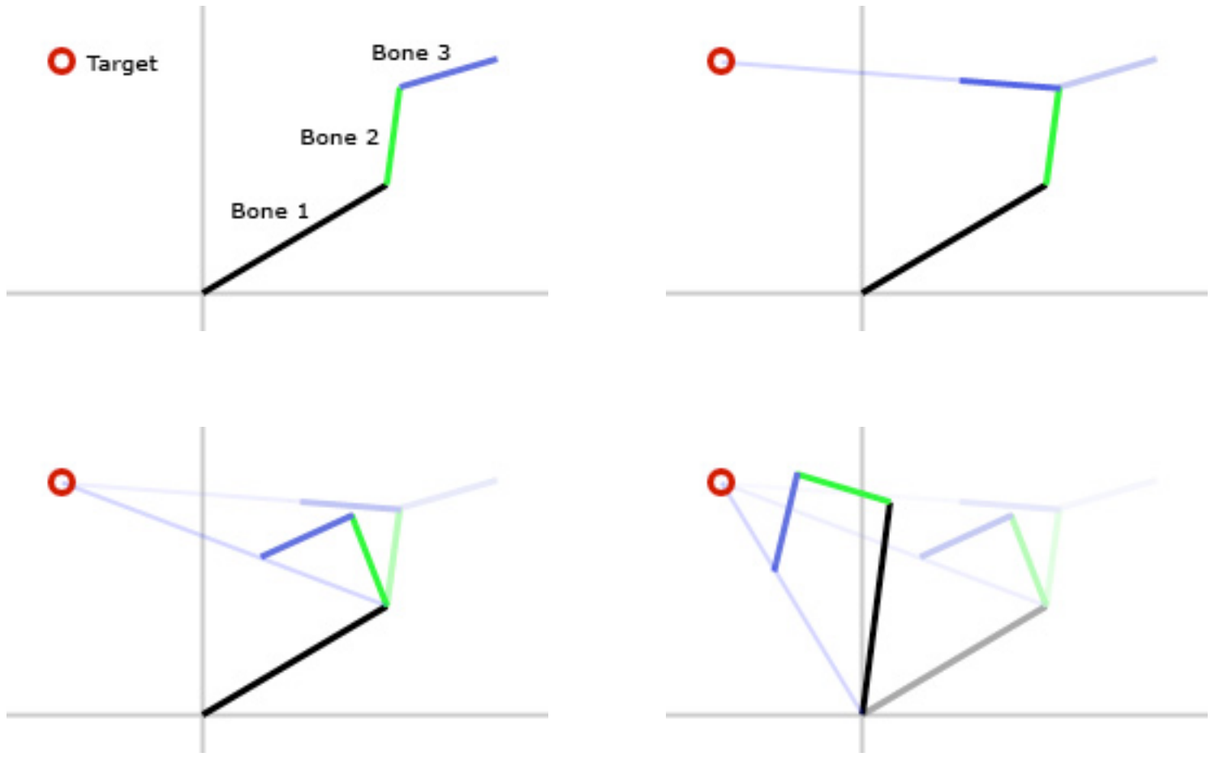
Cyclic Coordinate Descent

Iterativ metod som riktar leder explicit. En led i taget vrids åt rätt håll. Arbetar utifrån och in.

Vrid en led i taget så att man kommer så nära målet som möjligt. Starta med den yttersta, jobba mot roten.

Upprepa tills målet nås eller avståndet är litet nog.

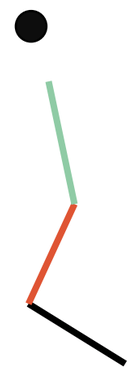
CCD, exempel



(http://www.ryanjuckett.com)

Problemfall

Ofta ger CCD bra resultat med snabb konvergens. Dock, det finns vissa situationer som ger långsam konvergens, och extremfall som inte konvergerar alls (lokal extrempunkt).



Konvergerar långsamt

Konvergerar inte alls

Cyclic Coordinate Descent

Åter har en heuristisk metod, en enkel approximation, visat sig minst lika bra som en analytisk!

CCD kan ni implementera när som helst. Enkel, intuitiv, rättfram, ger bra resultat!

Motion capture

Mycket vanligt i film!

- Tracking markers
- Aktiva sensorer på kroppen
- Markör/sensorfritt, analys enbart från naturlig bild.

Perfekt för att skapa förgenererade animationer.

Ansiktsanimation

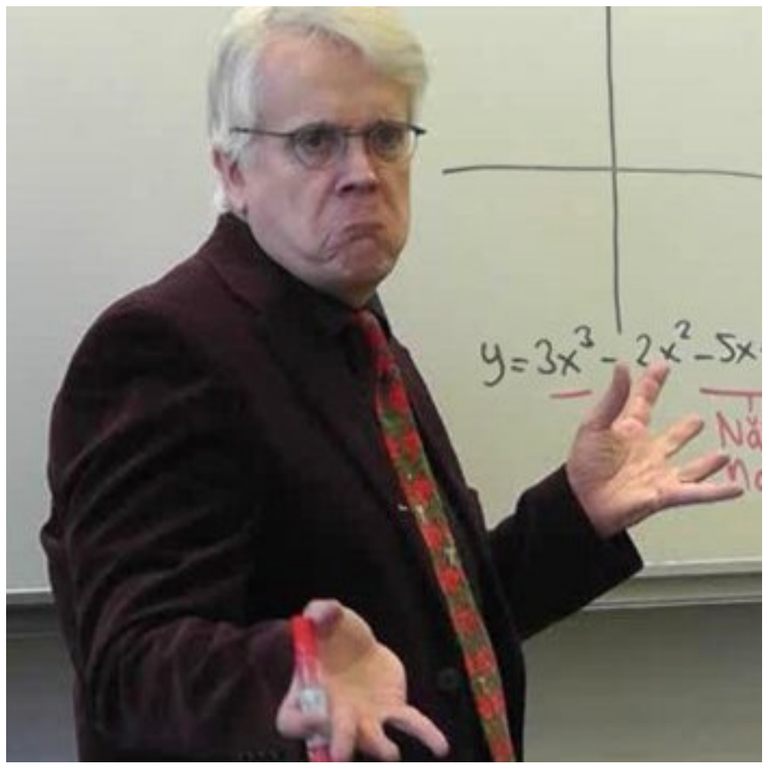
Svårt problem; vi är mycket känsliga för fel!

Animation med "action units" (≈muskler) eller "face animation parameters" (hög detalj)

FAPs ingår i MPEG-4.

The Candide model

Candide, ansiktsmodell från ISY



Skapad av
Mikael Rydfalk

Används i
Kinect!