Számítógépes Grafika

Bán Róbert
robert.ban102+cg@gmail.com

Eötvös Loránd Tudományegyetem
Informatikai Kar

2020-2021. tavaszi félév

Tartalom

Tartalom

Animáció

- Színtereink ritkán statikusak, különösen interaktív alkalmazások esetén

Animáció

- Színtereink ritkán statikusak, különösen interaktív alkalmazások esetén
- Animáció: képek egymás utáni sorozata

Animáció

- Színtereink ritkán statikusak, különösen interaktív alkalmazások esetén
- Animáció: képek egymás utáni sorozata
- Ha elég gyorsan követik egymást, akkor folytonos mozgás érzete alakul ki (ld. Critical Flicker Frequency korábban)

Hagyományos animációs módszerek

- Minden egyes képkockát külön megrajzolni

Hagyományos animációs módszerek

- Minden egyes képkockát külön megrajzolni
  - Mindent tudunk pontosan irányítani

Hagyományos animációs módszerek

- Minden egyes képkockát külön megrajzolni
  - Mindent tudunk pontosan irányítani
  - Rendkívül munkaigényes (és drága)

Hagyományos animációs módszerek

- Minden egyes képkockát külön megrajzolni
  - Mindent tudunk pontosan irányítani
  - Rendkívül munkaigényes (és drága)
- Cel animation

Hagyományos animációs módszerek

- Minden egyes képkockát külön megrajzolni
  - Mindent tudunk pontosan irányítani
  - Rendkívül munkaigényes (és drága)
- Cel animation
  - Rétegek (layerek) használata, újrahasználható fóliákon a színtér egyes részei

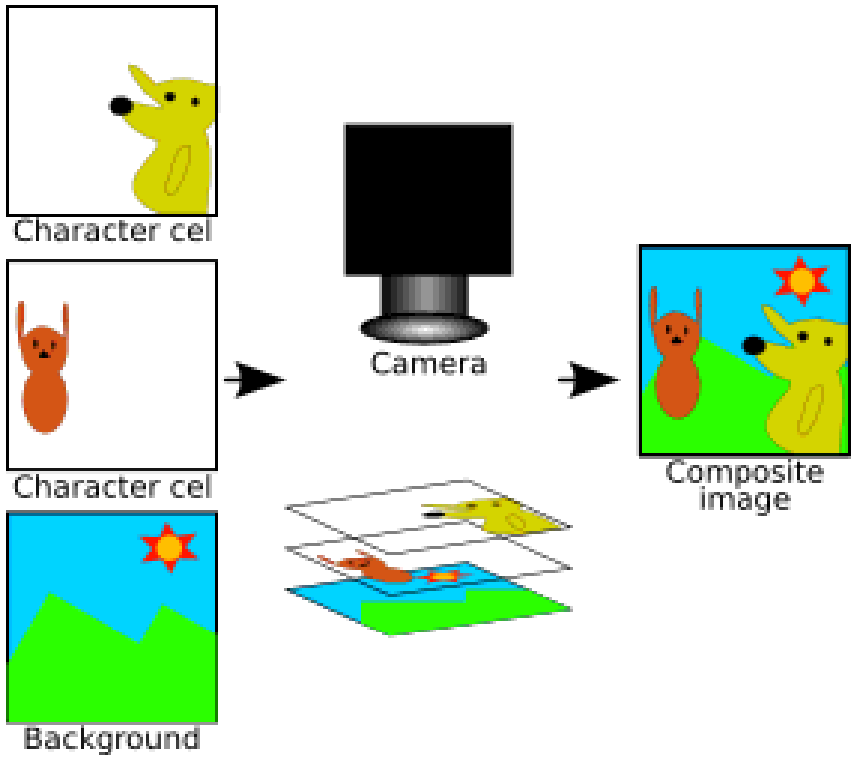
Hagyományos animációs módszerek

- Minden egyes képkockát külön megrajzolni
  - Mindent tudunk pontosan irányítani
  - Rendkívül munkaigényes (és drága)
- Cel animation
  - Rétegek (layerek) használata, újrahasználható fóliákon a színtér egyes részei
  - Kulcskereteket (keyframe-eket) rajzolják a fő rajzolók, az átmenetekért (inbetween) a „fiatalabbak” felelősek (a megadott utasításoknak megfelelően)

Hagyományos animációs módszerek

- Minden egyes képkockát külön megrajzolni
  - Mindent tudunk pontosan irányítani
  - Rendkívül munkaigényes (és drága)
- Cel animation
  - Rétegek (layerek) használata, újrahasználható fóliákon a színtér egyes részei
  - Kulcskereteket (keyframe-eket) rajzolják a fő rajzolók, az átmenetekért (inbetween) a „fiatalabbak” felelősek (a megadott utasításoknak megfelelően)
  - De: ha minden mozog a háttérben ez is elképesztően munkaigényes (és drága: pl. The Wings of Honneamise)...

Hagyományos animációs módszerek



- További érdekességek: „Principles of Traditional Animation Applied to 3D Computer”, SIGGRAPH’87, pp. 35-44.

Számítógéppel segített animáció

- Kulcskeret (keyframe) alapú animáció

Számítógéppel segített animáció

- Kulcskeret (keyframe) alapú animáció
  - A köztes animációs fázisok elkészítésének automatizálása: „csak” a kulcskeret képkockákat kell elkészíteni és megadni a kívánt átmenetek paramétereit

Számítógéppel segített animáció

- Kulcskeret (keyframe) alapú animáció
  - A köztes animációs fázisok elkészítésének automatizálása: „csak” a kulcskeret képkockákat kell elkészíteni és megadni a kívánt átmenetek paramétereit
  - Rugalmas, kevésbé munkaigényes de még mindig komoly képzettséget igényel (újakat is)

Számítógéppel segített animáció

- Kulcskeret (keyframe) alapú animáció
  - A köztes animációs fázisok elkészítésének automatizálása: „csak” a kulcskeret képkockákat kell elkészíteni és megadni a kívánt átmenetek paramétereit
  - Rugalmas, kevésbé munkaigényes de még mindig komoly képzettséget igényel (újakat is)
- Procedurális animáció

Számítógéppel segített animáció

- Kulcskeret (keyframe) alapú animáció
  - A köztes animációs fázisok elkészítésének automatizálása: „csak” a kulcskeret képkockákat kell elkészíteni és megadni a kívánt átmenetek paramétereit
  - Rugalmas, kevésbé munkaigényes de még mindig komoly képzettséget igényel (újakat is)
- Procedurális animáció
  - A mozgás algoritmikus leírása

Számítógéppel segített animáció

- Kulcskeret (keyframe) alapú animáció
  - A köztes animációs fázisok elkészítésének automatizálása: „csak” a kulcskeret képkockákat kell elkészíteni és megadni a kívánt átmenetek paramétereit
  - Rugalmas, kevésbé munkaigényes de még mindig komoly képzettséget igényel (újakat is)
- Procedurális animáció
  - A mozgás algoritmikus leírása
  - Például pattogó labda magassága az idő (t) függvényében: $m(t) = |\sin(\omega t + \theta_0)|e^{-kt}$

Számítógéppel segített animáció

- Fizikai alapú animáció

Számítógéppel segített animáció

- Fizikai alapú animáció
  - A színterünk elemeit fizikai jellemzőkkel látjuk el (tömeg, erők, rugalmasság stb.)

Számítógéppel segített animáció

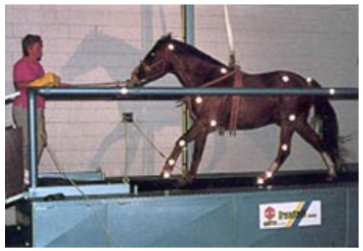
- Fizikai alapú animáció
  - A színterünk elemeit fizikai jellemzőkkel látjuk el (tömeg, erők, rugalmasság stb.)
  - A fizikai törvényei alapján felírt egyenletek megoldásaként előáll az animáció

Számítógéppel segített animáció

- Fizikai alapú animáció
  - A színterünk elemeit fizikai jellemzőkkel látjuk el (tömeg, erők, rugalmasság stb.)
  - A fizikai törvényei alapján felírt egyenletek megoldásaként előáll az animáció
  - Realisztikus (ha jól csináljuk), de nehéz irányítani

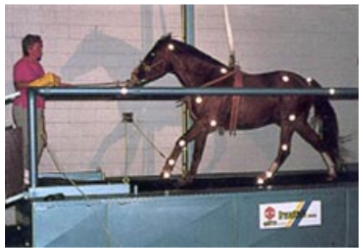
Számítógéppel segített animáció

- Fizikai alapú animáció
  - A színterünk elemeit fizikai jellemzőkkel látjuk el (tömeg, erők, rugalmasság stb.)
  - A fizikai törvényei alapján felírt egyenletek megoldásaként előáll az animáció
  - Realisztikus (ha jól csináljuk), de nehéz irányítani
- Motion capture


Számítógéppel segített animáció

- Fizikai alapú animáció
  - A színterünk elemeit fizikai jellemzőkkel látjuk el (tömeg, erők, rugalmasság stb.)
  - A fizikai törvényei alapján felírt egyenletek megoldásaként előáll az animáció
  - Realisztikus (ha jól csináljuk), de nehéz irányítani
- Motion capture
  - Műszerekkel rögzíteni amint valaki/valami elvégzi a kívánt mozgásokat majd ezt egy digitális modellre átültetni


Tartalom

Mit animálhatunk?

- Lényegében bármit, ami befolyásolja a megjelenő képet a színtérről (modellek pozíciója, orientációja, színe, reprezentációhoz kötődő tulajdonságai, BRDF stb.)

Mit animálhatunk?

- Lényegében bármit, ami befolyásolja a megjelenő képet a színtérről (modellek pozíciója, orientációja, színe, reprezentációhoz kötődő tulajdonságai, BRDF stb.)
- Elsősorban a modell és kamera transzformációkkal foglalkozunk most csak

Mit animálhatunk?

- Lényegében bármit, ami befolyásolja a megjelenő képet a színtérről (modellek pozíciója, orientációja, színe, reprezentációhoz kötődő tulajdonságai, BRDF stb.)
- Elsősorban a modell és kamera transzformációkkal foglalkozunk most csak
- A feladatunk ezeknek a paramétereknek időtől függővé tétele

Animáció szintézis

- Legyen minden o objektumra a modellezési (világ) transzformáció $M_o \in \mathbb{R}^{4\times4}$

Animáció szintézis

- Legyen minden o objektumra a modellezési (világ) transzformáció $M_o \in \mathbb{R}^{4\times4}$
- Legyen a nézeti (kamera) transzformáció: $V \in \mathbb{R}^{4\times4}$

Animáció szintézis

- Legyen minden o objektumra a modellezési (világ) transzformáció $M_o \in \mathbb{R}^{4\times4}$
- Legyen a nézeti (kamera) transzformáció: $V \in \mathbb{R}^{4\times4}$
- *Megjegyzés: ha V a View mátrix, akkor csak a kamera pozícióját, orientációját tartalmazza; ha V a View és a Projection együtt, akkor tudjuk vele a látószöget is állítani*

Animáció szintézis

- Legyen minden o objektumra a modellezési (világ) transzformáció $M_o \in \mathbb{R}^{4\times4}$
- Legyen a nézeti (kamera) transzformáció: $V \in \mathbb{R}^{4\times4}$
- *Megjegyzés: ha V a View mátrix, akkor csak a kamera pozícióját, orientációját tartalmazza; ha V a View és a Projection együtt, akkor tudjuk vele a látószöget is állítani*
- Legyen mindkettő az idő függvénye!

Animáció szintézis

- Legyen minden o objektumra a modellezési (világ) transzformáció $M_o \in \mathbb{R}^{4\times4}$
- Legyen a nézeti (kamera) transzformáció: $V \in \mathbb{R}^{4\times4}$
- *Megjegyzés: ha V a View mátrix, akkor csak a kamera pozícióját, orientációját tartalmazza; ha V a View és a Projection együtt, akkor tudjuk vele a látószöget is állítani*
- Legyen mindkettő az idő függvénye!
- $M_o \in \mathbb{R} \to \mathbb{R}^{4\times4}$

Animáció szintézis

- Legyen minden o objektumra a modellezési (világ) transzformáció $M_o \in \mathbb{R}^{4\times4}$
- Legyen a nézeti (kamera) transzformáció: $V \in \mathbb{R}^{4\times4}$
- *Megjegyzés: ha V a View mátrix, akkor csak a kamera pozícióját, orientációját tartalmazza; ha V a View és a Projection együtt, akkor tudjuk vele a látószöget is állítani*
- Legyen mindkettő az idő függvénye!
- $M_o \in \mathbb{R} \to \mathbb{R}^{4\times4}$
- $V \in \mathbb{R} \to \mathbb{R}^{4\times4}$

Általános animációs program váza

```
while keep_running:
    t = get_time()
    for o in objects:
        M_o = M_o(t)
    V = V(t)
    render_scene()
```

Animáció szintézis

- Valósidejű/interaktív:

Animáció szintézis

- Valósidejű/interaktív:
  - rögtön meg is jelenítjük a képkockákat

Animáció szintézis

- Valósidejű/interaktív:
  - rögtön meg is jelenítjük a képkockákat
  - a számításnak elég gyorsnak kell lennie a folytonosság látszatához

Animáció szintézis

- Valósidejű/interaktív:
  - rögtön meg is jelenítjük a képkockákat
  - a számításnak elég gyorsnak kell lennie a folytonosság látszatához
  - a felhasználói eseményekre reagálni kell

Animáció szintézis

- Valósidejű/interaktív:
  - rögtön meg is jelenítjük a képkockákat
  - a számításnak elég gyorsnak kell lennie a folytonosság látszatához
  - a felhasználói eseményekre reagálni kell
  - $\Rightarrow$ olyan részletgazdag lehet a színtér, amit még így meg lehet jeleníteni

Animáció szintézis

- Valósidejű/interaktív:
  - rögtön meg is jelenítjük a képkockákat
  - a számításnak elég gyorsnak kell lennie a folytonosság látszatához
  - a felhasználói eseményekre reagálni kell
  - $\Rightarrow$ olyan részletgazdag lehet a színtér, amit még így meg lehet jeleníteni
  - $\Rightarrow$ inkrementális képszintézist használunk

Valósidejű animációs program váza

```
def update():
  # FrameUpdate() / UpdateScene()
  t = get_time()
  for o in objects:
    M_o = M_o(t)
  V = V(t)

def render():
  # Render() / DrawScene()
  for o in objects:
    render_object(o)
```

Animáció szintézis

- Nem valós idejű/*offline*:

Animáció szintézis

- Nem valós idejű/*offline*:
  - „Nem számít”, hogy mennyi ideig tart kiszámítani egy képkockát

Animáció szintézis

- Nem valós idejű/*offline*:
  - „Nem számít”, hogy mennyi ideig tart kiszámítani egy képkockát
  - Elkülönül a szintézis és a visszajátszás

Animáció szintézis

- Nem valós idejű/*offline*:
  - „Nem számít”, hogy mennyi ideig tart kiszámítani egy képkockát
  - Elkülönül a szintézis és a visszajátszás
    - Először elmentjük a képkockákat

Animáció szintézis

- Nem valós idejű/*offline*:
  - „Nem számít”, hogy mennyi ideig tart kiszámítani egy képkockát
  - Elkülönül a szintézis és a visszajátszás
    - Először elmentjük a képkockákat
    - Aztán majd videóként lehet visszanézni

Animáció szintézis

- Nem valós idejű/*offline*:
  - „Nem számít”, hogy mennyi ideig tart kiszámítani egy képkockát
  - Elkülönül a szintézis és a visszajátszás
    - Először elmentjük a képkockákat
    - Aztán majd videóként lehet visszanézni
  - $\Rightarrow$ a felhasználó nem tud belenyúlni az animációba

Animáció szintézis

- Nem valós idejű/*offline*:
  - „Nem számít”, hogy mennyi ideig tart kiszámítani egy képkockát
  - Elkülönül a szintézis és a visszajátszás
    - Először elmentjük a képkockákat
    - Aztán majd videóként lehet visszanézni
  - $\Rightarrow$ a felhasználó nem tud belenyúlni az animációba
  - $\Rightarrow$ olyan részletes színteret, és olyan bonyolult algoritmust használunk, amit ki tudunk várni

Animáció szintézis

- Nem valós idejű/*offline*:
  - „Nem számít”, hogy mennyi ideig tart kiszámítani egy képkockát
  - Elkülönül a szintézis és a visszajátszás
    - Először elmentjük a képkockákat
    - Aztán majd videóként lehet visszanézni
  - $\Rightarrow$ a felhasználó nem tud belenyúlni az animációba
  - $\Rightarrow$ olyan részletes színteret, és olyan bonyolult algoritmust használunk, amit ki tudunk várni
  - Azért a költségvetés keretet szab: Final Fantasy: Spirits Within (2001), egy 200 fős csapat 4 éves munkája volt (kb. 120 emberév összesítve), aminek elkészítésére 960 munkaállomást használtak. A film 141964 frame-ből állt végül (15TB), egy frame átlagosan 90 percig renderelőtött!

Offline rendering

```
Δt = 1/FPS
for (t=t_start; t<t_end; t+=Δt):
    for o in objects:
        M_o = M_o(t)
    V = V(t)
    render_scene_to_disk()

start_time = get_time()
for (t=t_start; t<t_end; t+=Δt):
    draw_frame(t)
    wait(Δ t)
```

Valósidejű animáció – rosszul

- Hogyan lehet a legkönnyebben elrontani az animáció számítást?

Valósidejű animáció – rosszul

- Hogyan lehet a legkönnyebben elrontani az animáció számítást?
- Azzal, ha nem vesszük figyelembe, hogy mennyi idő telt el két képkocka között.

Valósidejű animáció – rosszul

- Hogyan lehet a legkönnyebben elrontani az animáció számítást?
- Azzal, ha nem vesszük figyelembe, hogy mennyi idő telt el két képkocka között.
- Pl.: A középpontja körül akarjuk forgatni az objektumot állandó szögsebességgel.

```
model = rotate(phi, 0,1,0); phi += phi_step;
```

Valósidejű animáció – rosszul

- Hogyan lehet a legkönnyebben elrontani az animáció számítást?
- Azzal, ha nem vesszük figyelembe, hogy mennyi idő telt el két képkocka között.
- Pl.: A középpontja körül akarjuk forgatni az objektumot állandó szögsebességgel.

```
model = rotate(phi, 0,1,0); phi += phi_step;
```

- Az objektum olyan gyorsan fog forogni, amilyen gyakorisággal ez a kód részlet meghívódik.

Valósidejű animáció – rosszul

- Hogyan lehet a legkönnyebben elrontani az animáció számítást?
- Azzal, ha nem vesszük figyelembe, hogy mennyi idő telt el két képkocka között.
- Pl.: A középpontja körül akarjuk forgatni az objektumot állandó szögsebességgel.

```
model = rotate(phi, 0,1,0); phi += phi_step;
```

- Az objektum olyan gyorsan fog forogni, amilyen gyakorisággal ez a kód részlet meghívódik.
- Gyorsabb gépen többször, lassabb gépen kevesebbszer – és semmi sem garantálja, hogy ugyanazon a gépen két hívás között egyáltalán ugyanannyi idő telne el!

Valósidejű animáció – jól

- Hogyal lehet a legkönnyebben ezt megelőzni?

Valósidejű animáció – jól

- Hogyal lehet a legkönnyebben ezt megelőzni?
- Sose azt tároljuk, hogy mennyivel kell változtatni, hanem, hogy mi a változás *sebessége*.

Valósidejű animáció – jól

- Hogyal lehet a legkönnyebben ezt megelőzni?
- Sose azt tároljuk, hogy mennyivel kell változtatni, hanem, hogy mi a változás *sebessége*.
- Minden számítás előtt kérjük le, hogy mennyi idő telt el az előző képkocka óta, és ezzel szorozzuk a sebességet.

Valósidejű animáció – jól

- Hogyal lehet a legkönnyebben ezt megelőzni?
- Sose azt tároljuk, hogy mennyivel kell változtatni, hanem, hogy mi a változás *sebessége*.
- Minden számítás előtt kérjük le, hogy mennyi idő telt el az előző képkocka óta, és ezzel szorozzuk a sebességet.
- Pl.: `phi += phi_step;` helyett `phi += get_time_since_last_frame() * phi_speed;`

Valósidejű animáció – jól

- Hogyal lehet a legkönnyebben ezt megelőzni?
- Sose azt tároljuk, hogy mennyivel kell változtatni, hanem, hogy mi a változás *sebessége*.
- Minden számítás előtt kérjük le, hogy mennyi idő telt el az előző képkocka óta, és ezzel szorozzuk a sebességet.
- Pl.: `phi += phi_step;` helyett `phi += get_time_since_last_frame() * phi_speed;`
- Gyorsabb gépen kevesebb, lassabb gépen több idő telik el két képkocka között $\Rightarrow$ gyors gépen kisebbeket lépünk, lassabban nagyobbakat – és a „lötyögés” sem számít (hívások között eltelt idő változása)

Tartalom

Kameraanimáció

- A kamera tulajdonságai:

Kameraanimáció

- A kamera tulajdonságai:
  - szempozíció (*eye*),

Kameraanimáció

- A kamera tulajdonságai:
  - szempozíció (*eye*),
  - egy pont amire néz (*center*),

Kameraanimáció

- A kamera tulajdonságai:
  - szempozíció (*eye*),
  - egy pont amire néz (*center*),
  - felfelé irányt megadó vektor (*up*),

Kameraanimáció

- A kamera tulajdonságai:
  - szempozíció (*eye*),
  - egy pont amire néz (*center*),
  - felfelé irányt megadó vektor (*up*),
  - a képernyő/ablak oldal aránya (*aspect*),

Kameraanimáció

- A kamera tulajdonságai:
  - szempozíció (*eye*),
  - egy pont amire néz (*center*),
  - felfelé irányt megadó vektor (*up*),
  - a képernyő/ablak oldal aránya (*aspect*),
  - nyílásszög (*fovy*).

Kameraanimáció

- A kamera tulajdonságai:
  - szempozíció (*eye*),
  - egy pont amire néz (*center*),
  - felfelé irányt megadó vektor (*up*),
  - a képernyő/ablak oldal aránya (*aspect*),
  - nyílásszög (*fovy*).
- Ezek mind külön-külön változtathatók az animáció létrehozásához.

Kameraanimáció

- A kamera tulajdonságai:
  - szempozíció (*eye*),
  - egy pont amire néz (*center*),
  - felfelé irányt megadó vektor (*up*),
  - a képernyő/ablak oldal aránya (*aspect*),
  - nyílásszög (*fovy*).
- Ezek mind külön-külön változtathatók az animáció létrehozásához.
- Például a szempozíciót a korábbi órákon látott parametrikus görbékkel is megadhatjuk, ahol a paraméter az idő lesz! Pl. **a**-ból **b**-be 5 másodperc alatt a $\mathbf{p}(t) = (1 - \frac{t}{5})\mathbf{a} + \frac{t}{5}\mathbf{b}$, ha t másodpercekben adott és $t = 0$-ban indulunk **a**-ból.

Tartalom

Pozíció és orientáció

- Pozícó: *„Hol van az objektum?”*

Pozíció és orientáció

- Pozícó: „*Hol van az objektum?*”
- Orientáció: „*Hogy áll, merrefelé néz az objektum?*”

Pozíció és orientáció

- Pozícó: *„Hol van az objektum?”*
- Orientáció: *„Hogy áll, merrefelé néz az objektum?”*
- Elsősorban ezt a kettőt szeretnénk változtatni.

Pozíció és orientáció

- Pozícó: *„Hol van az objektum?”*
- Orientáció: *„Hogy áll, merrefelé néz az objektum?”*
- Elsősorban ezt a kettőt szeretnénk változtatni.
- $M_o(t)$ megadja mindkettőt.

Pozíció és orientáció

- Pozícó: „*Hol van az objektum?*”
- Orientáció: „*Hogy áll, merrefelé néz az objektum?*”
- Elsősorban ezt a kettőt szeretnénk változtatni.
- $M_o(t)$ megadja mindkettőt.
- Normális esetben

$$M_o(t) = \begin{bmatrix} A_{11} & A_{12} & A_{13} & p_x \\ A_{21} & A_{22} & A_{23} & p_y \\ A_{31} & A_{32} & A_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

Pozíció és orientáció

- Pozícó: „*Hol van az objektum?*”
- Orientáció: „*Hogy áll, merrefelé néz az objektum?*”
- Elsősorban ezt a kettőt szeretnénk változtatni.
- $M_o(t)$ megadja mindkettőt.
- Normális esetben

$$M_o(t) = \begin{bmatrix} A_{11} & A_{12} & A_{13} & p_x \\ A_{21} & A_{22} & A_{23} & p_y \\ A_{31} & A_{32} & A_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

- $\mathbf{p} = (p_x, p_y, p_z)$ a pozíció.

Pozíció és orientáció

- Pozícó: „*Hol van az objektum?*”
- Orientáció: „*Hogy áll, merrefelé néz az objektum?*”
- Elsősorban ezt a kettőt szeretnénk változtatni.
- $M_o(t)$ megadja mindkettőt.
- Normális esetben

$$M_o(t) = \begin{bmatrix} A_{11} & A_{12} & A_{13} & p_x \\ A_{21} & A_{22} & A_{23} & p_y \\ A_{31} & A_{32} & A_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

- $\mathbf{p} = (p_x, p_y, p_z)$ a pozíció.
- Az $\mathbf{A}$ mátrix tartalmazza az orientációt.

Orientációs paraméterek

- Tegyük **p**-t és **A**-t is időfüggővé!

Orientációs paraméterek

- Tegyük **p**-t és **A**-t is időfüggővé!
- **p** tagjait leírhatjuk külün-külön függvénnyel.

Orientációs paraméterek

- Tegyük **p**-t és **A**-t is időfüggővé!
- **p** tagjait leírhatjuk külün-külön függvénnyel.
- *Pl.: Valami esik, elég p_y-t változtatni.*

Orientációs paraméterek

- Tegyük **p**-t és **A**-t is időfüggővé!
- **p** tagjait leírhatjuk külün-külön függvénnyel.
- *Pl.: Valami esik, elég p_y-t változtatni.*
- **A** tagjai összefüggenek, és csak az orientáció érdekel belőle minket (nem érdekel: méretezés, nyírás).

Orientációs paraméterek

- Tegyük **p**-t és **A**-t is időfüggővé!
- **p** tagjait leírhatjuk külün-külön függvénnyel.
- *Pl.: Valami esik, elég p_y-t változtatni.*
- **A** tagjai összefüggenek, és csak az orientáció érdekel belőle minket (nem érdekel: méretezés, nyírás).
- Az orientáció megadható három tengely menti forgatással $\rightarrow$ három független függvénnyel.

Yaw, pitch, roll

- Egy objektum függőleges- (*yaw*), kereszt- (*pitch*) és hossztengelye (*roll*) menti elfordulásait egyszerre adjuk meg.

Yaw, pitch, roll

- Egy objektum függőleges- (*yaw*), kereszt- (*pitch*) és hossztengelye (*roll*) menti elfordulásait egyszerre adjuk meg.
- Már találkoztunk vele, 3×3-as mátrixszal megadható, három forgatás szorzata.

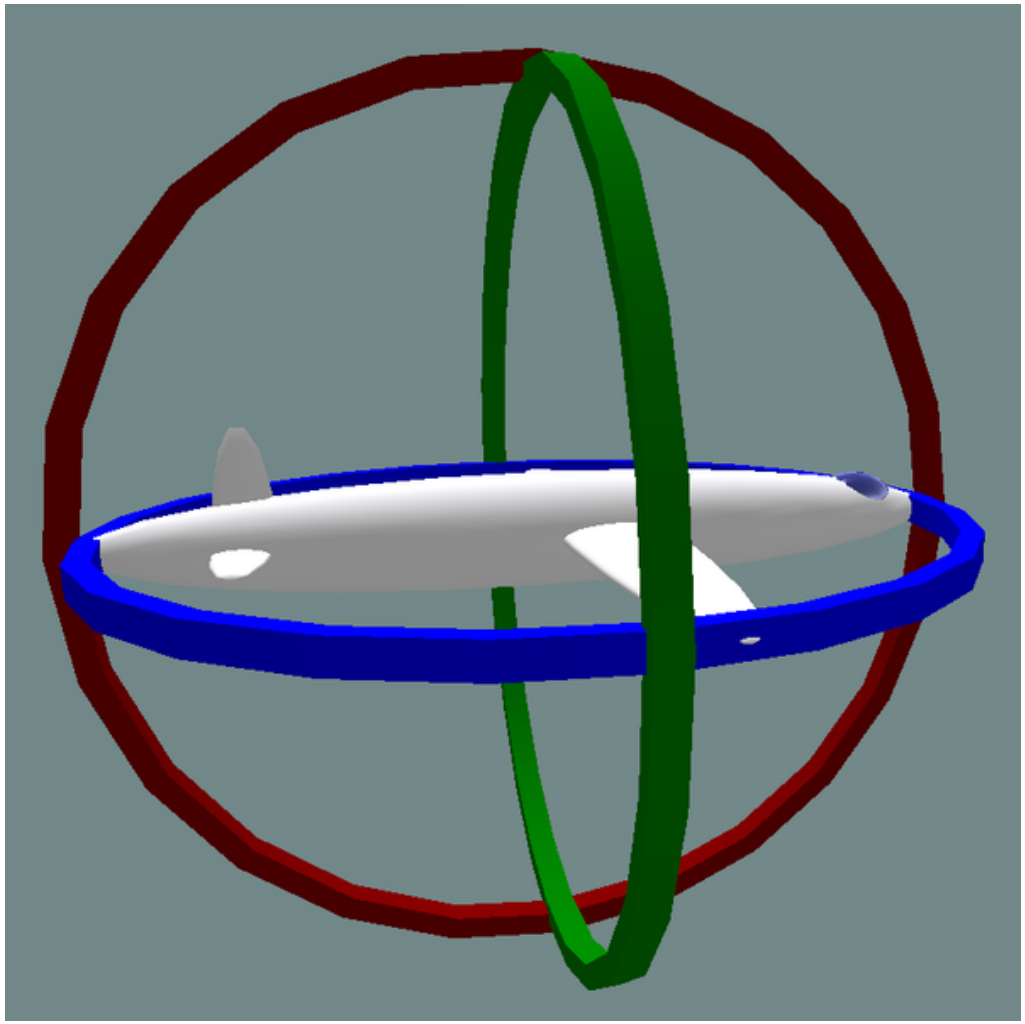
Yaw, pitch, roll

- Egy objektum függőleges- (*yaw*), kereszt- (*pitch*) és hossztengelye (*roll*) menti elfordulásait egyszerre adjuk meg.
- Már találkoztunk vele, 3×3-as mátrixszal megadható, három forgatás szorzata.
- Három tengely menti elfordulás szögével megadható $\Rightarrow$ megadja az orientációt is.

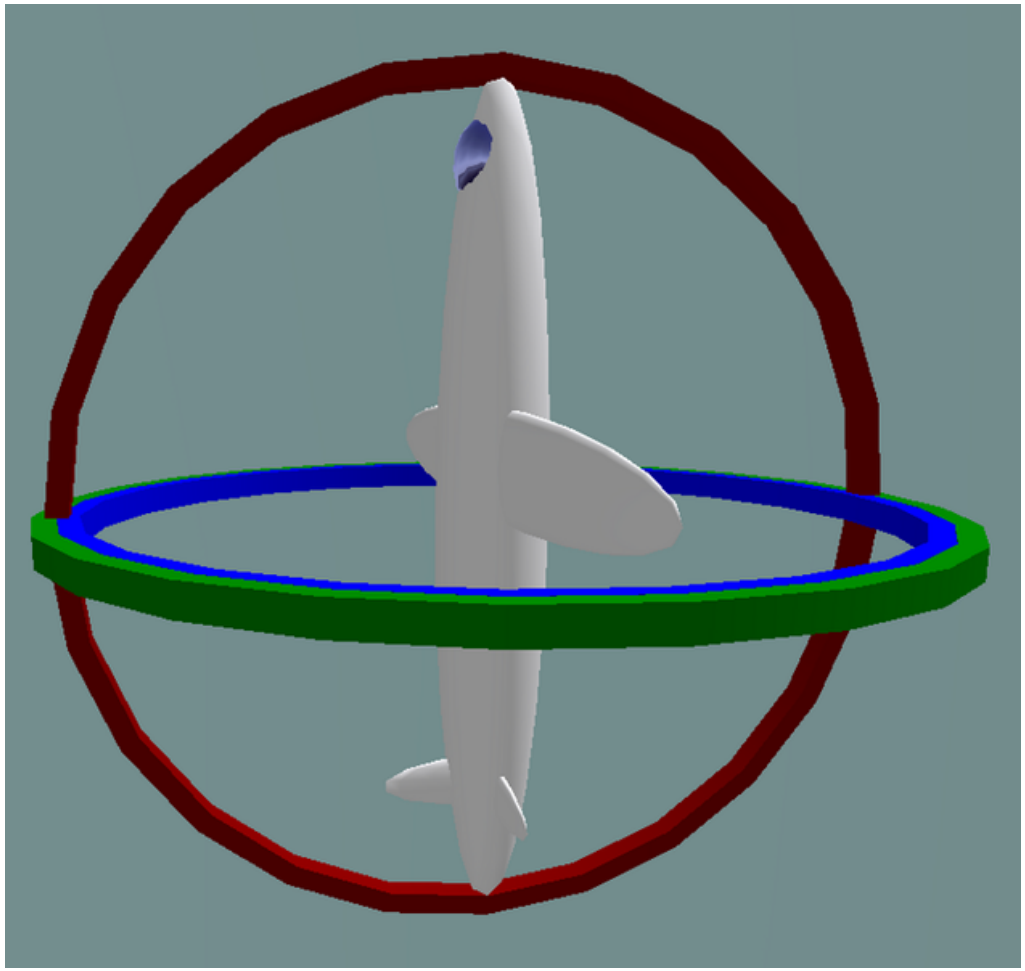
Yaw, pitch, roll

- Egy objektum függőleges- (*yaw*), kereszt- (*pitch*) és hossztengelye (*roll*) menti elfordulásait egyszerre adjuk meg.
- Már találkoztunk vele, 3×3-as mátrixszal megadható, három forgatás szorzata.
- Három tengely menti elfordulás szögével megadható $\Rightarrow$ megadja az orientációt is.
- Tetszőleges orientáció megadható vele, de animációnál vigyázzunk, mert...

„How about sending me a fourth gimbal for Christmas?” – Mike Collins, az Apollo 11 pilótája



Gimbal lock



Tartalom

Képlet animáció

- Egy adott tulajdonság változását egy megfelelő függvénnyel írjuk le.

Képlet animáció

- Egy adott tulajdonság változását egy megfelelő függvénnyel írjuk le.
- Pl: Óra mutatói

Képlet animáció

- Egy adott tulajdonság változását egy megfelelő függvénnyel írjuk le.
- Pl: Óra mutatói
  - Nagymutató: $yaw(t) = t/10$

Képlet animáció

- Egy adott tulajdonság változását egy megfelelő függvénnyel írjuk le.
- Pl: Óra mutatói
  - Nagymutató: $yaw(t) = t/10$
  - Kismutató: $yaw(t) = t/120$

Képlet animáció

- Egy adott tulajdonság változását egy megfelelő függvénnyel írjuk le.
- Pl: Óra mutatói
  - Nagymutató: $yaw(t) = t/10$
  - Kismutató: $yaw(t) = t/120$
  - Ha t-t mp-ben adjuk meg, a forgatásokat pedig fokban.

Képlet animáció

- Egy adott tulajdonság változását egy megfelelő függvénnyel írjuk le.
- Pl: Óra mutatói
  - Nagymutató: $yaw(t) = t/10$
  - Kismutató: $yaw(t) = t/120$
  - Ha t-t mp-ben adjuk meg, a forgatásokat pedig fokban.
- Pl: Pattogó labda

Képlet animáció

- Egy adott tulajdonság változását egy megfelelő függvénnyel írjuk le.
- Pl: Óra mutatói
  - Nagymutató: $yaw(t) = t/10$
  - Kismutató: $yaw(t) = t/120$
  - Ha t-t mp-ben adjuk meg, a forgatásokat pedig fokban.
- Pl: Pattogó labda
  - $p_y(t) = |\sin(\omega t + \theta_0)| \cdot e^{-kt}$

Tartalom

Kulcskocka (*key frame*) animáció

- Egy bonyolult mozgást nehézkes lenne képlettel megadni.

Kulcskocka (*key frame*) animáció

- Egy bonyolult mozgást nehézkes lenne képlettel megadni.
- Inkább adjuk csak bizonyos időközönként, hogy *akkor* mit szeretnénk látni.

Kulcskocka (*key frame*) animáció

- Egy bonyolult mozgást nehézkes lenne képlettel megadni.
- Inkább adjuk csak bizonyos időközönként, hogy *akkor* mit szeretnénk látni.
- Ezek a kulcskockák.

Kulcskocka (*key frame*) animáció

- Egy bonyolult mozgást nehézkes lenne képlettel megadni.
- Inkább adjuk csak bizonyos időközönként, hogy *akkor* mit szeretnénk látni.
- Ezek a kulcskockák.
- Egy tulajdonságot két kulcskocka között *interpolációval* számolunk ki.

Kulcskocka (*key frame*) animáció

- Az interpolációval az objektum egyes paramétereire folytonos görbét illesztünk.

Kulcskocka (*key frame*) animáció

- Az interpolációval az objektum egyes paramétereire folytonos görbét illesztünk.
- Az animáció lejátszása/elmentése során a program minden képkockában a hozzá tartozó t értékkel kiértékeli az objektum paraméter-függvényeit.

Kulcskocka (*key frame*) animáció

- Az interpolációval az objektum egyes paramétereire folytonos görbét illesztünk.
- Az animáció lejátszása/elmentése során a program minden képkockában a hozzá tartozó t értékkel kiértékeli az objektum paraméter-függvényeit.
- Ezekből számítja a transzformációs mátrixokat.

Kulcskocka (*key frame*) animáció

- Az interpolációval az objektum egyes paramétereire folytonos görbét illesztünk.
- Az animáció lejátszása/elmentése során a program minden képkockában a hozzá tartozó t értékkel kiértékeli az objektum paraméter-függvényeit.
- Ezekből számítja a transzformációs mátrixokat.
- A mátrixok felhasználásával előállítja a képet.

Lineáris interpoláció (emlékeztető)

- Legyen a két kulcskockánk időpontja t_0 és t_1.

Lineáris interpoláció (emlékeztető)

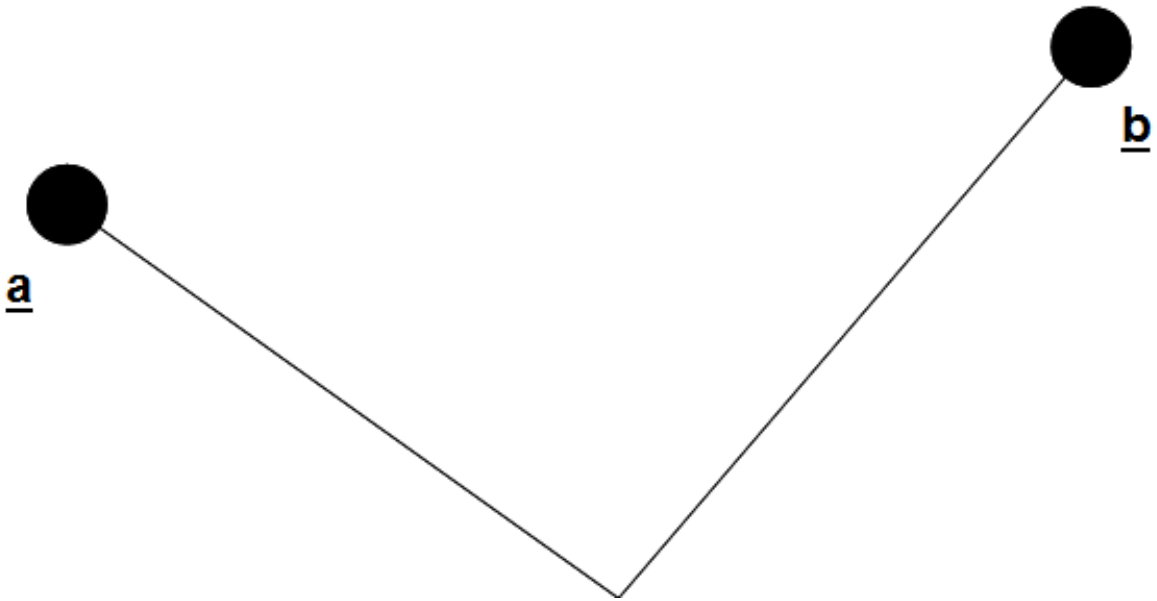
- Legyen a két kulcskockánk időpontja t_0 és t_1.
- Legyen az interpolálandó tulajdonság g.

Lineáris interpoláció (emlékeztető)

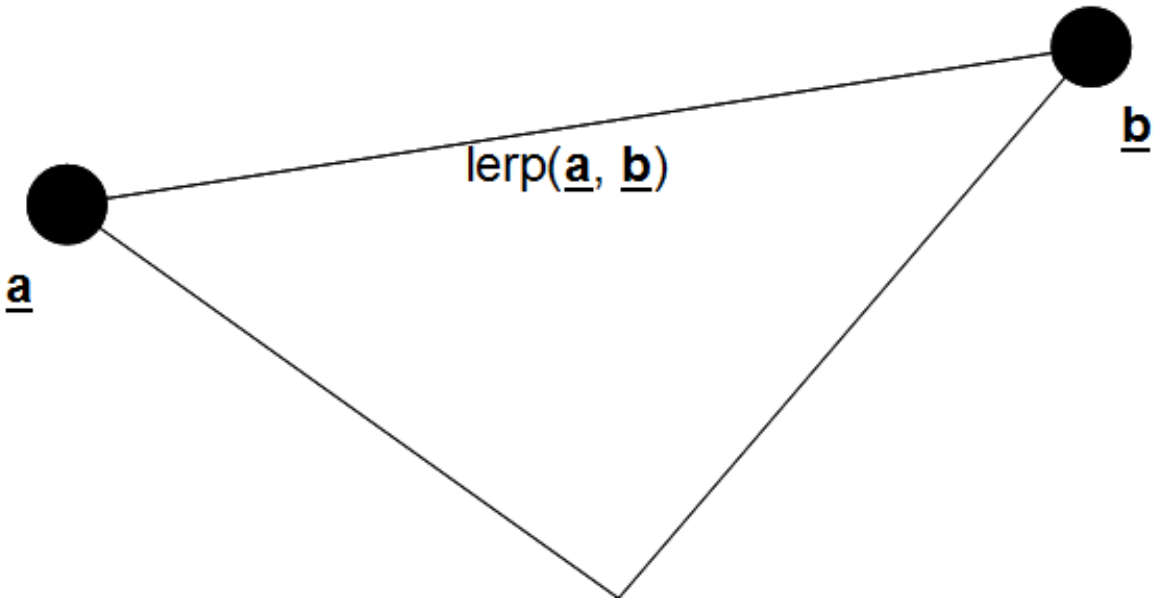
- Legyen a két kulcskockánk időpontja t_0 és t_1.
- Legyen az interpolálandó tulajdonság g.
- Lineáris interpolációval $\forall t \in [t_0, t_1]$-re kapjuk

$$g(t) = \left(1 - \frac{t - t_0}{t_1 - t_0}\right) g(t_0) + \frac{t - t_0}{t_1 - t_0} g(t_1)$$

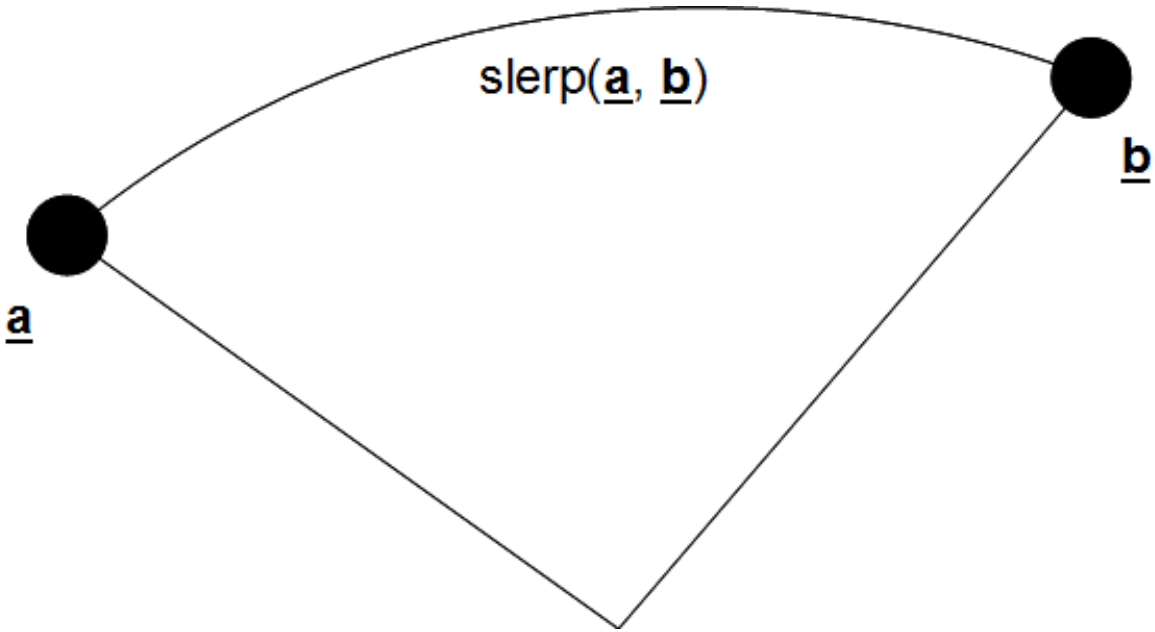
Lineáris és gömbi lineáris interpoláció



Lineáris és gömbi lineáris interpoláció



Lineáris és gömbi lineáris interpoláció



Interpoláció két kulcskocka között

K: Mi még a baj a lineáris interpolációval?

Interpoláció két kulcskocka között

K: Mi még a baj a lineáris interpolációval?

V: Ritkán néz ki természetesen. Animáció során a sebesség konstans, előtte, utána nulla.

Interpoláció két kulcskocka között

K: Mi még a baj a lineáris interpolációval?

V: Ritkán néz ki természetesen. Animáció során a sebesség konstans, előtte, utána nulla.

- Elgurított labda: folyamatosan lassul.

Interpoláció két kulcskocka között

K: Mi még a baj a lineáris interpolációval?

V: Ritkán néz ki természetesen. Animáció során a sebesség konstans, előtte, utána nulla.

- Elgurított labda: folyamatosan lassul.
- Zuhanó zongora: folyamatosan gyorsul.

Interpoláció két kulcskocka között

K: Mi még a baj a lineáris interpolációval?

V: Ritkán néz ki természetesen. Animáció során a sebesség konstans, előtte, utána nulla.

- Elgurított labda: folyamatosan lassul.
- Zuhanó zongora: folyamatosan gyorsul.
- Rakéta a Földről a Marsra: gyorsul, halad, lassít.

Interpoláció két kulcskocka között

K: Mi még a baj a lineáris interpolációval?

V: Ritkán néz ki természetesen. Animáció során a sebesség konstans, előtte, utána nulla.

- Elgurított labda: folyamatosan lassul.
- Zuhanó zongora: folyamatosan gyorsul.
- Rakéta a Földről a Marsra: gyorsul, halad, lassít.

- Ilyen interpolációra használhatunk:

Interpoláció két kulcskocka között

K: Mi még a baj a lineáris interpolációval?

V: Ritkán néz ki természetesen. Animáció során a sebesség konstans, előtte, utána nulla.

- Elgurított labda: folyamatosan lassul.
- Zuhanó zongora: folyamatosan gyorsul.
- Rakéta a Földről a Marsra: gyorsul, halad, lassít.

- Ilyen interpolációra használhatunk:
  - Gyök függvényt.

Interpoláció két kulcskocka között

K: Mi még a baj a lineáris interpolációval?

V: Ritkán néz ki természetesen. Animáció során a sebesség konstans, előtte, utána nulla.

- Elgurított labda: folyamatosan lassul.
- Zuhanó zongora: folyamatosan gyorsul.
- Rakéta a Földről a Marsra: gyorsul, halad, lassít.

- Ilyen interpolációra használhatunk:
  - Gyök függvényt.
  - Másodfokú függvényt.

Interpoláció két kulcskocka között

K: Mi még a baj a lineáris interpolációval?

V: Ritkán néz ki természetesen. Animáció során a sebesség konstans, előtte, utána nulla.

- Elgurított labda: folyamatosan lassul.
- Zuhanó zongora: folyamatosan gyorsul.
- Rakéta a Földről a Marsra: gyorsul, halad, lassít.

- Ilyen interpolációra használhatunk:
  - Gyök függvényt.
  - Másodfokú függvényt.
  - ...

Interpoláció két kulcskocka között

K: Mi még a baj a lineáris interpolációval?

V: Ritkán néz ki természetesen. Animáció során a sebesség konstans, előtte, utána nulla.

- Elgurított labda: folyamatosan lassul.
- Zuhanó zongora: folyamatosan gyorsul.
- Rakéta a Földről a Marsra: gyorsul, halad, lassít.

- Ilyen interpolációra használhatunk:
  - Gyök függvényt.
  - Másodfokú függvényt.
  - ...
- Gyakorlatban: Bézier görbe

Polinom interpoláció

- n kulcspontra fel tudunk írni $n-1$-ed fokú polinomot.

Polinom interpoláció

- n kulcspontra fel tudunk írni $n-1$-ed fokú polinomot.
- Interpolációs polinom: minden kulcskockában az előírt értéket veszi fel.

Polinom interpoláció

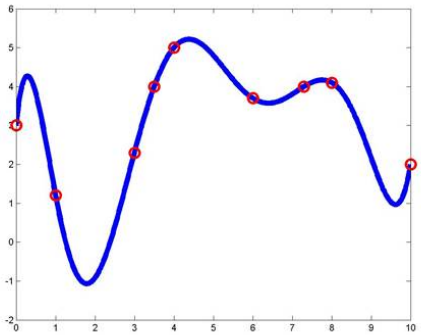
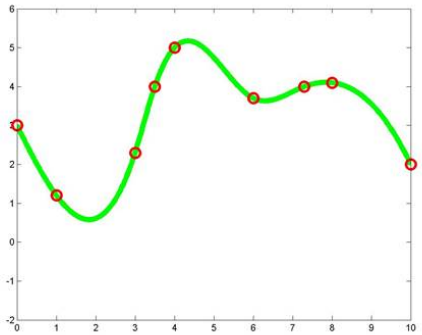
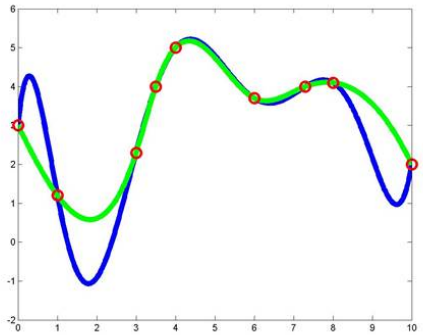
- n kulcspontra fel tudunk írni $n-1$-ed fokú polinomot.
- Interpolációs polinom: minden kulcskockában az előírt értéket veszi fel.
- Együtthatók számíthatók Lagrange interpolációval is akár (de: nagy fokszám – nagy „kilengések” adatpontok között itt is!).

Polinom interpoláció

- n kulcspontra fel tudunk írni $n-1$-ed fokú polinomot.
- Interpolációs polinom: minden kulcskockában az előírt értéket veszi fel.
- Együtthatók számíthatók Lagrange interpolációval is akár (de: nagy fokszám – nagy „kilengések” adatpontok között itt is!).
- A lineáris interpoláció a Lagrange interpoláció speciális esete $n = 2$-re.

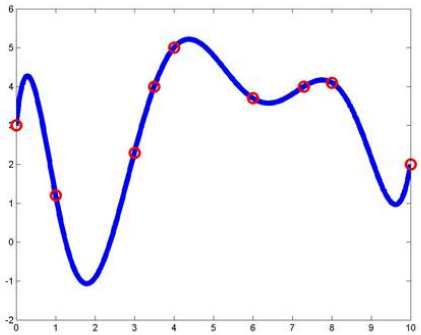
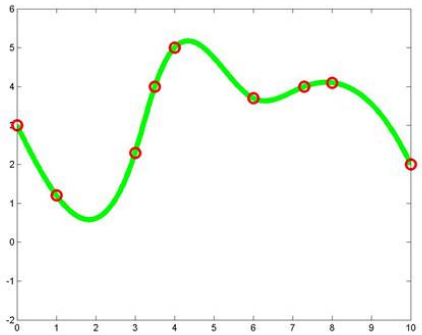
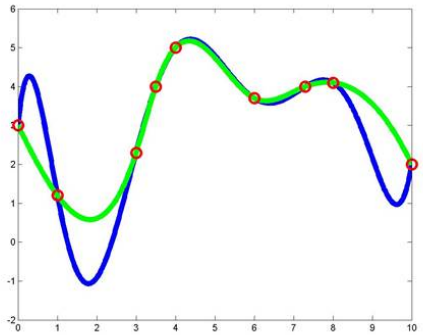
Spline interpoláció

- A polinom interpolációval kapott fv. magas fokszám esetén a szomszédos pontok között „hullámzik”, így elrontja az animációt.

Spline interpoláció

- A polinom interpolációval kapott fv. magas fokszám esetén a szomszédos pontok között „hullámzik”, így elrontja az animációt.
- Spline interpoláció: használjunk több, egymáshoz kapcsolódó, alacsony fokszámú polinomot az interpolációhoz!

Tartalom

Pályaanimáció

- Egy objektum mozgását megadhatjuk a bejárandó pálya megadásával is.

Pályaanimáció

- Egy objektum mozgását megadhatjuk a bejárandó pálya megadásával is.
- A pályát egy 3D görbével adjuk meg.

Pályaanimáció

- Egy objektum mozgását megadhatjuk a bejárandó pálya megadásával is.
- A pályát egy 3D görbével adjuk meg.
- A modell ezen a görbén halad végig.

Orientáció megadása

- Hogyan adjuk meg az objektumunk orientációját?

Orientáció megadása

- Hogyan adjuk meg az objektumunk orientációját?
- Egy *előre*, és egy *felfelé* irány egyértelműen meghatározza ezt.

Orientáció megadása

- Hogyan adjuk meg az objektumunk orientációját?
- Egy *előre*, és egy *felfelé* irány egyértelműen meghatározza ezt.
- *Megjegyzés: v.ö. kamera esetén center–eye ill. up vektorok*

Orientáció megadása

- Hogyan adjuk meg az objektumunk orientációját?
- Egy *előre*, és egy *felfelé* irány egyértelműen meghatározza ezt.
- *Megjegyzés: v.ö. kamera esetén center–eye ill. up vektorok*
- Ha a pályagörbe differenciálható, akkor az megadja a sebességvektort minden időpillanatban.

Orientáció megadása

- Hogyan adjuk meg az objektumunk orientációját?
- Egy *előre*, és egy *felfelé* irány egyértelműen meghatározza ezt.
- *Megjegyzés: v.ö. kamera esetén center–eye ill. up vektorok*
- Ha a pályagörbe differenciálható, akkor az megadja a sebességvektort minden időpillanatban.
- A sebességvektor mindig előrefelé mutat.

Orientáció megadása

- A *felfelé* irány megadására két lehetőségünk is van.

Orientáció megadása

- A *felfelé* irány megadására két lehetőségünk is van.
- Ha van egy természetes *felfelé*, akkor használjuk azt. (Mindennél, ami nem dől be a kanyarban.)

Orientáció megadása

- A *felfelé* irány megadására két lehetőségünk is van.
- Ha van egy természetes *felfelé*, akkor használjuk azt. (Mindennél, ami nem dől be a kanyarban.)
- Ha ez az irány is változik, akkor ez megegyezik a gyorsulás irányával, azaz a pályagörbe második deriváltjának irányával.

Tartalom

Hierarchikus rendszerek

- Színtérgráfoknál már találkoztunk ilyenekkel.

Hierarchikus rendszerek

- Színtérgráfoknál már találkoztunk ilyenekkel.
- Egy gyerek objektum mozgását a szülőhöz viszonyítva adjuk meg.

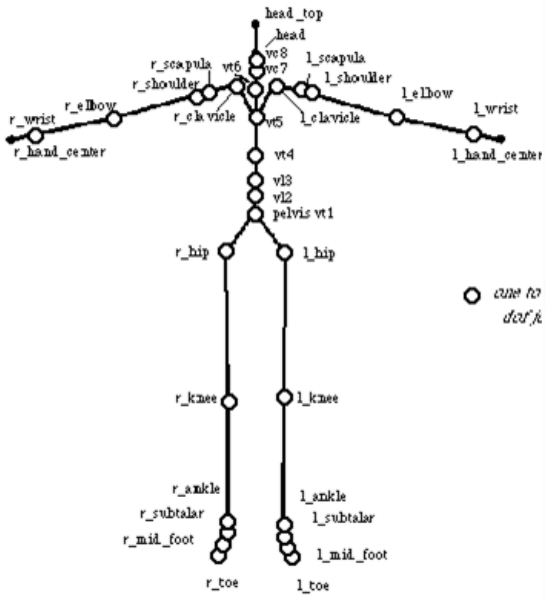
Hierarchikus rendszerek

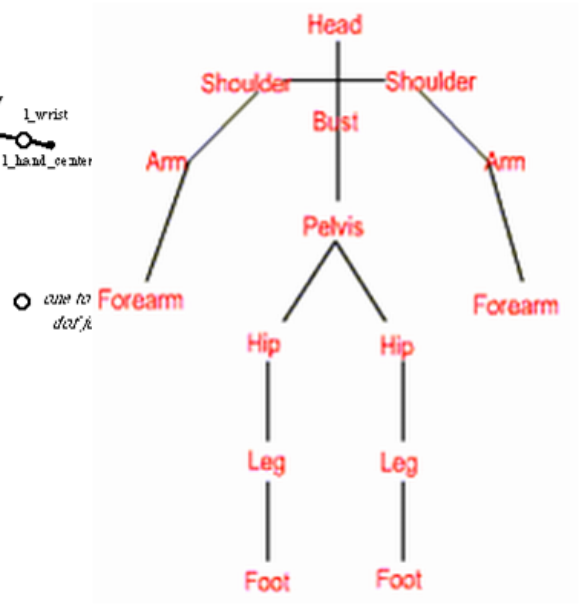
- Színtérgráfoknál már találkoztunk ilyenekkel.
- Egy gyerek objektum mozgását a szülőhöz viszonyítva adjuk meg.
- Gyerekeknek lehetnek újabb gyerekei stb.

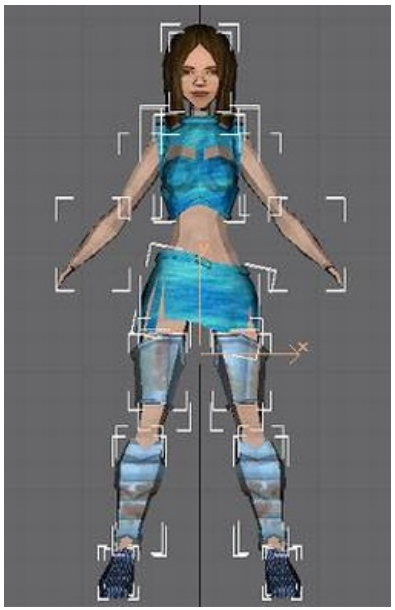
Hierarchikus rendszerek

- Színtérgráfoknál már találkoztunk ilyenekkel.
- Egy gyerek objektum mozgását a szülőhöz viszonyítva adjuk meg.
- Gyerekeknek lehetnek újabb gyerekei stb.
- Hierarchikus rendszert – fát – kapunk.

Példa: Emberi test







Kényszerek (*constraints*)

- Nem minden mozgást szeretnénk megengedni a szülőhöz képest.

Kényszerek (*constraints*)

- Nem minden mozgást szeretnénk megengedni a szülőhöz képest.
- Ezeket a megszorításokat írhatjuk le *kényszerekkel*.

Kényszerek (*constraints*)

- Nem minden mozgást szeretnénk megengedni a szülőhöz képest.
- Ezeket a megszorításokat írhatjuk le *kényszerekkel*.
- Korlátozhatjuk a szabadságfokokat: pl. könyök csak egy tengely mentén tud forogni, de a csukló kettő

Kényszerek (*constraints*)

- Nem minden mozgást szeretnénk megengedni a szülőhöz képest.
- Ezeket a megszorításokat írhatjuk le *kényszerekkel*.
- Korlátozhatjuk a szabadságfokokat: pl. könyök csak egy tengely mentén tud forogni, de a csukló kettő
- Vagy a tartományokat: kevesen bírják, ha a fejük 90°-nél többet fordul.

Tartalom

Előrehaladó kinematika

- Végállapotot határozunk meg az állapotváltozók függvényében.

Előrehaladó kinematika

- Végállapotot határozunk meg az állapotváltozók függvényében.
- Szimulációkhoz jól használható.

Előrehaladó kinematika

- Végállapotot határozunk meg az állapotváltozók függvényében.
- Szimulációkhoz jól használható.
- Minden elemre megadjuk a hozzá tartozó transzformációt.

Előrehaladó kinematika

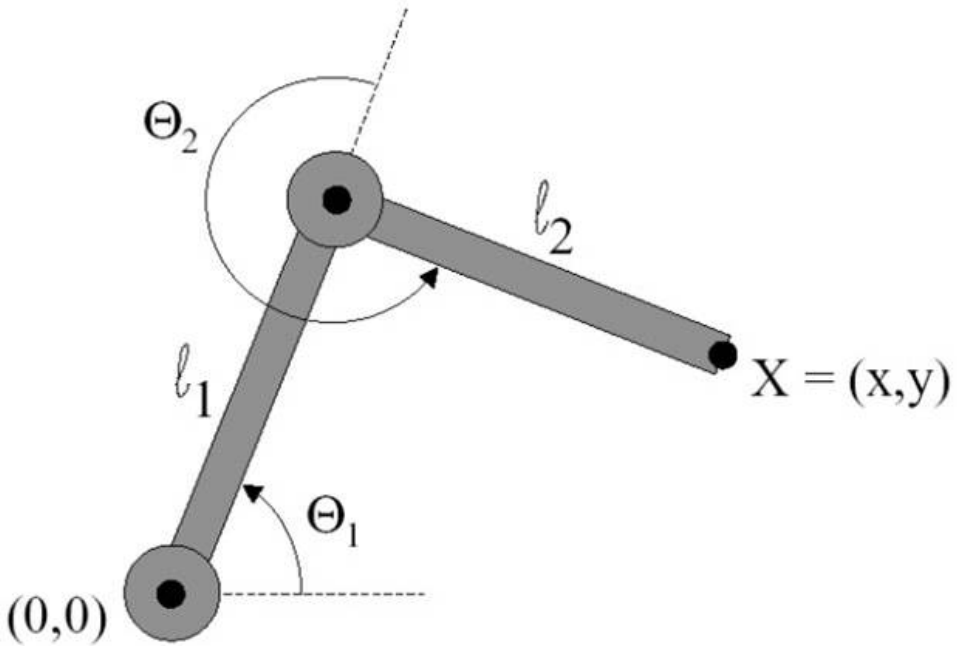
- Végállapotot határozunk meg az állapotváltozók függvényében.
- Szimulációkhoz jól használható.
- Minden elemre megadjuk a hozzá tartozó transzformációt.
- Ezeket a hierarchiában felülről lefelé haladva értékeljük ki.

Előrehaladó kinematika

- Végállapotot határozunk meg az állapotváltozók függvényében.
- Szimulációkhoz jól használható.
- Minden elemre megadjuk a hozzá tartozó transzformációt.
- Ezeket a hierarchiában felülről lefelé haladva értékeljük ki.
- Az adott elemhez tartozó transzformáció az összes ős és a saját transzformáció szorzata.

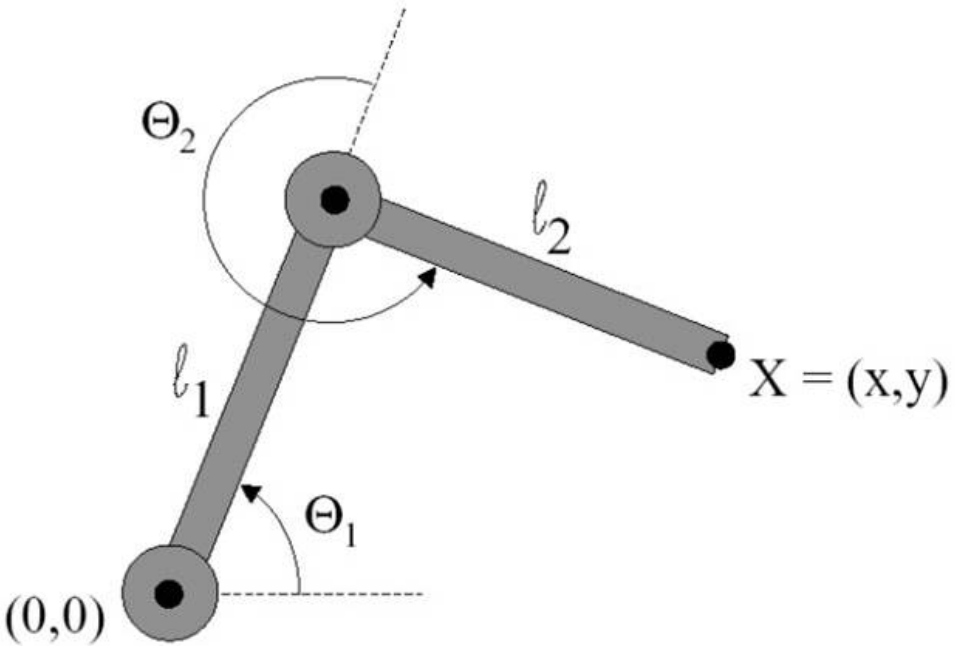
Példa

- Kétszabadságfokú, rotációs csuklókat tartalmazó rendszer.



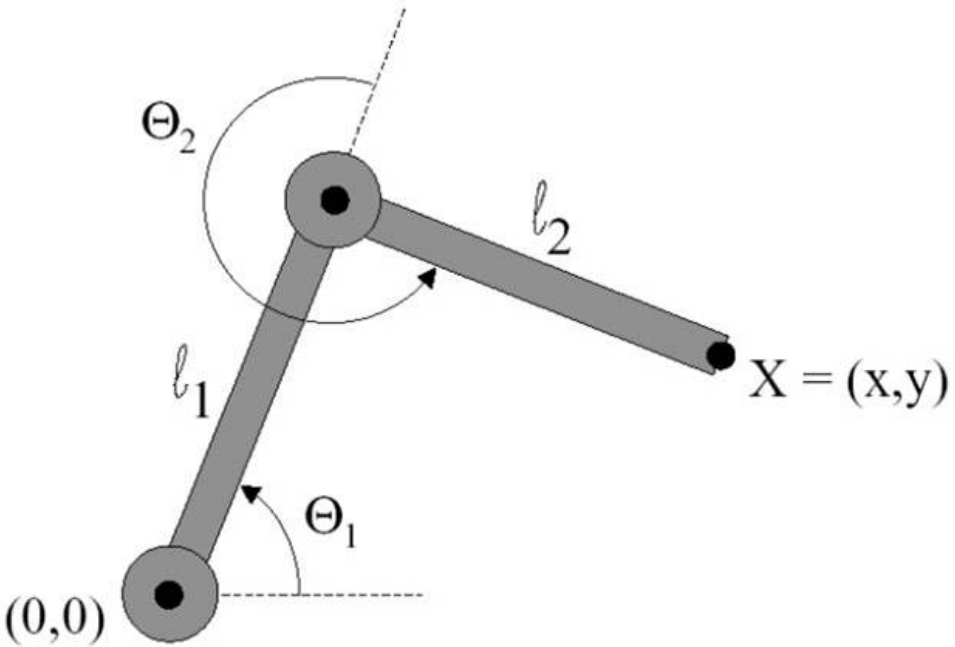
Példa

- Kétszabadságfokú, rotációs csuklókat tartalmazó rendszer.
- A csuklók csak a Z tengely körül fordulnak.



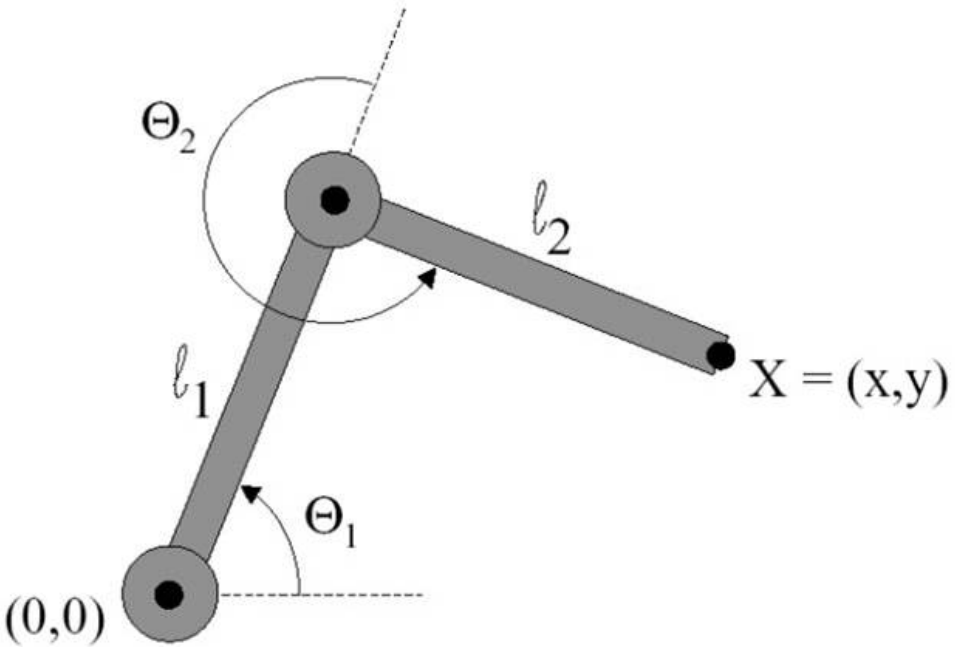
Példa – folyt.

- Állapotváltozók: Θ_1, Θ_2



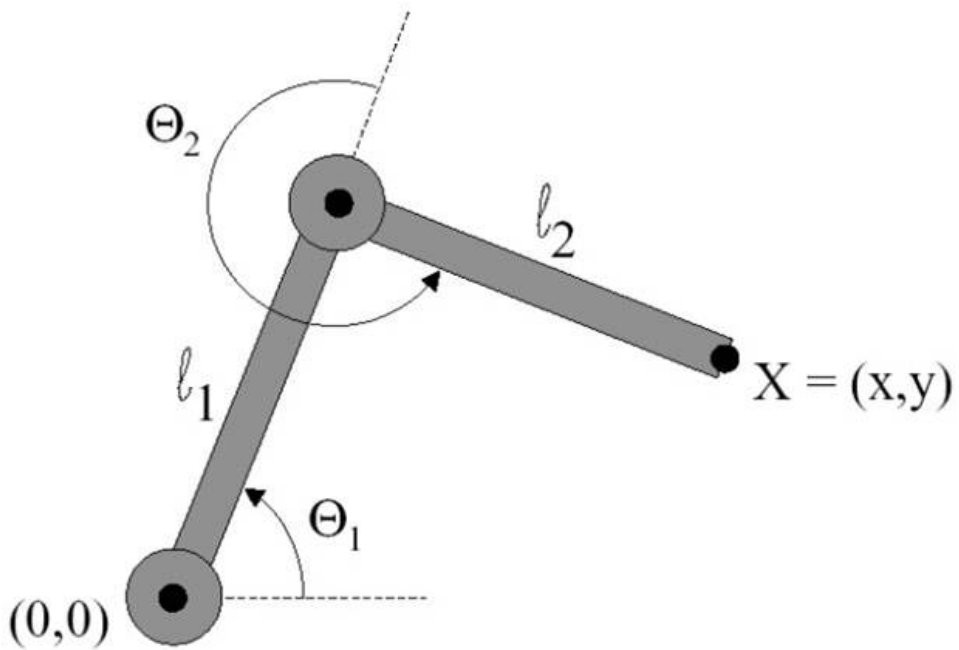
Példa – folyt.

- Állapotváltozók: Θ_1, Θ_2
- A végberendezés (X) pozícióját a gép számolja.



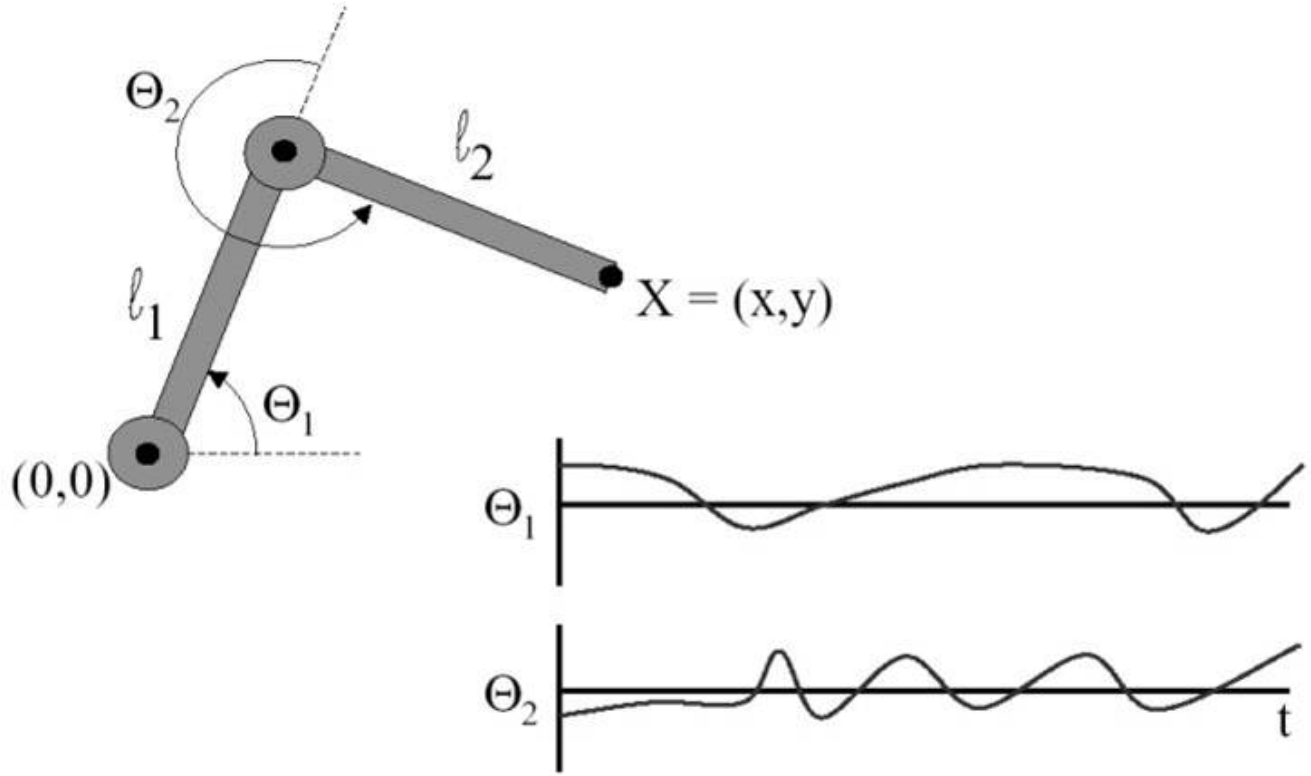
Példa – folyt.

- Állapotváltozók: Θ_1, Θ_2
- A végberendezés (X) pozícióját a gép számolja.
- $X = (l_1 \cos\Theta_1 + l_2 \cos(\Theta_1 + \Theta_2), l_1 \sin\Theta_1 + l_2 \sin(\Theta_1 + \Theta_2))$



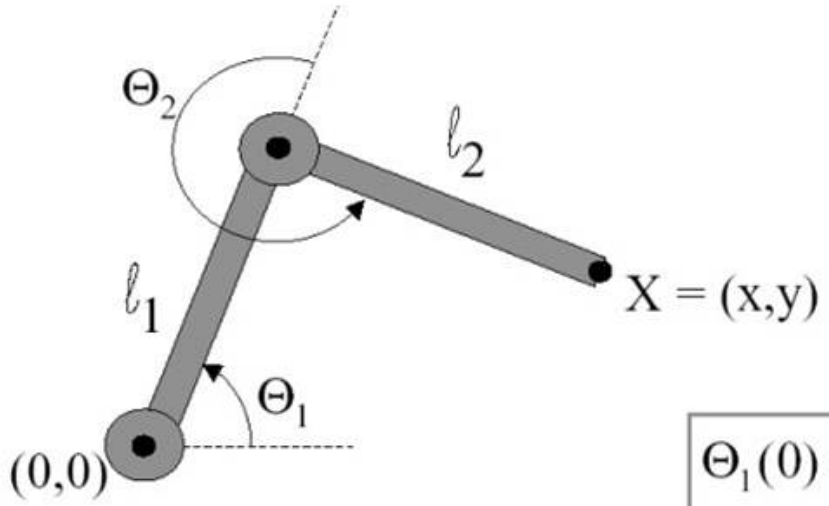
Példa – folyt.

- Az állapotváltozókat megadhatjuk (pl. spline) függvénnyel.



Példa – folyt.

- Az állapotváltozókat megadhatjuk kezdeti értékkel és sebességgel.



$$\Theta_1(0) = 60^\circ \qquad \Theta_2(0) = 250^\circ$$

$$\frac{d\Theta_1}{dt} = 1.2 \qquad \frac{d\Theta_2}{dt} = -0.1$$

Tartalom

Mit nem tud az előrehaladó kinematika?

- Az előrehaladó kinematika nem használható, ha a strukturális összefüggés erősen nem lineáris

Mit nem tud az előrehaladó kinematika?

- Az előrehaladó kinematika nem használható, ha a strukturális összefüggés erősen nem lineáris
- Hiába interpolálunk egyenletesen az állapottérben, a végberendezés vadul kalimpálhat a kulcspontok között

Mit nem tud az előrehaladó kinematika?

- Az előrehaladó kinematika nem használható, ha a strukturális összefüggés erősen nem lineáris
- Hiába interpolálunk egyenletesen az állapottérben, a végberendezés vadul kalimpálhat a kulcspontok között
- Problémás esetek:

Mit nem tud az előrehaladó kinematika?

- Az előrehaladó kinematika nem használható, ha a strukturális összefüggés erősen nem lineáris
- Hiába interpolálunk egyenletesen az állapottérben, a végberendezés vadul kalimpálhat a kulcspontok között
- Problémás esetek:
  - Láb mozgása a talajon

Mit nem tud az előrehaladó kinematika?

- Az előrehaladó kinematika nem használható, ha a strukturális összefüggés erősen nem lineáris
- Hiába interpolálunk egyenletesen az állapottérben, a végberendezés vadul kalimpálhat a kulcspontok között
- Problémás esetek:
  - Láb mozgása a talajon
  - Végállapot jó, de menet közben a berendezés részei átmehetnek egymáson.

Inverz kinematika

- Az inverz kinematika a kritikus végberendezés helyzetét interpolálja, majd az állapotváltozók értékét végberendezés interpolált helyzetéből számítja vissza.

Inverz kinematika

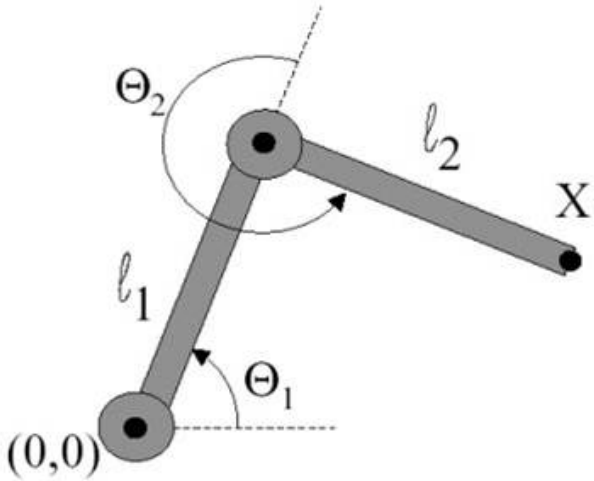
- Az inverz kinematika a kritikus végberendezés helyzetét interpolálja, majd az állapotváltozók értékét végberendezés interpolált helyzetéből számítja vissza.
- Az inverz kinematika másik neve a cél-orientált animáció.

Inverz kinematika

- Az inverz kinematika a kritikus végberendezés helyzetét interpolálja, majd az állapotváltozók értékét végberendezés interpolált helyzetéből számítja vissza.
- Az inverz kinematika másik neve a cél-orientált animáció.
- *„Ezt szeretném megfogni, hogyan forgassam az ízületeimet?”*

Példa

- A végberendezés helyzetéből visszaszámoljuk az állapotváltozók értékét.



$$\Theta_2 = \cos^{-1}\left(\frac{x^2 + x^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right)$$

$$\Theta_1 = \frac{-(l_2 \sin(\Theta_2))x + (l_1 + l_2 \cos(\Theta_2))y}{(l_2 \sin(\Theta_2))y + (l_1 + l_2 \cos(\Theta_2))x}$$

Problémák

- Nehéz „természetesnek látszó” mozgást leírni vele.

Problémák

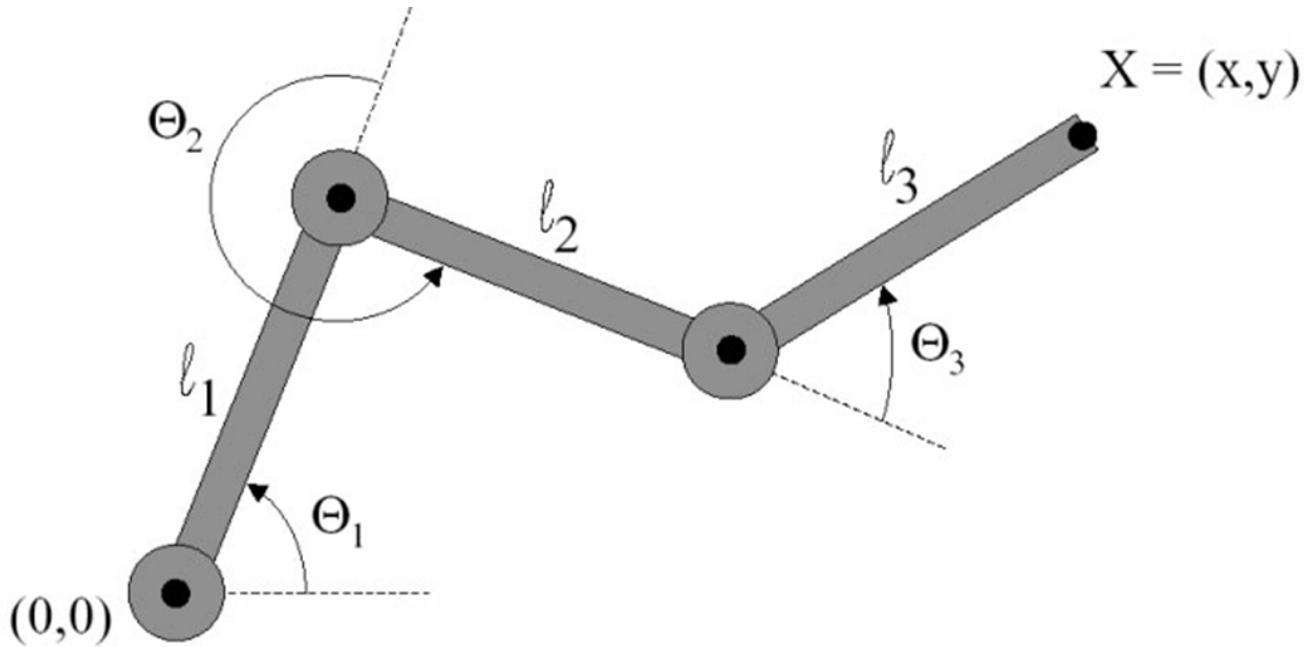
- Nehéz „természetesnek látszó” mozgást leírni vele.
- Az inverz függvény kiszámítása nem triviális,

Problémák

- Nehéz „természetesnek látszó” mozgást leírni vele.
- Az inverz függvény kiszámítása nem triviális,
- és nem is egyértelmű (redundancia).

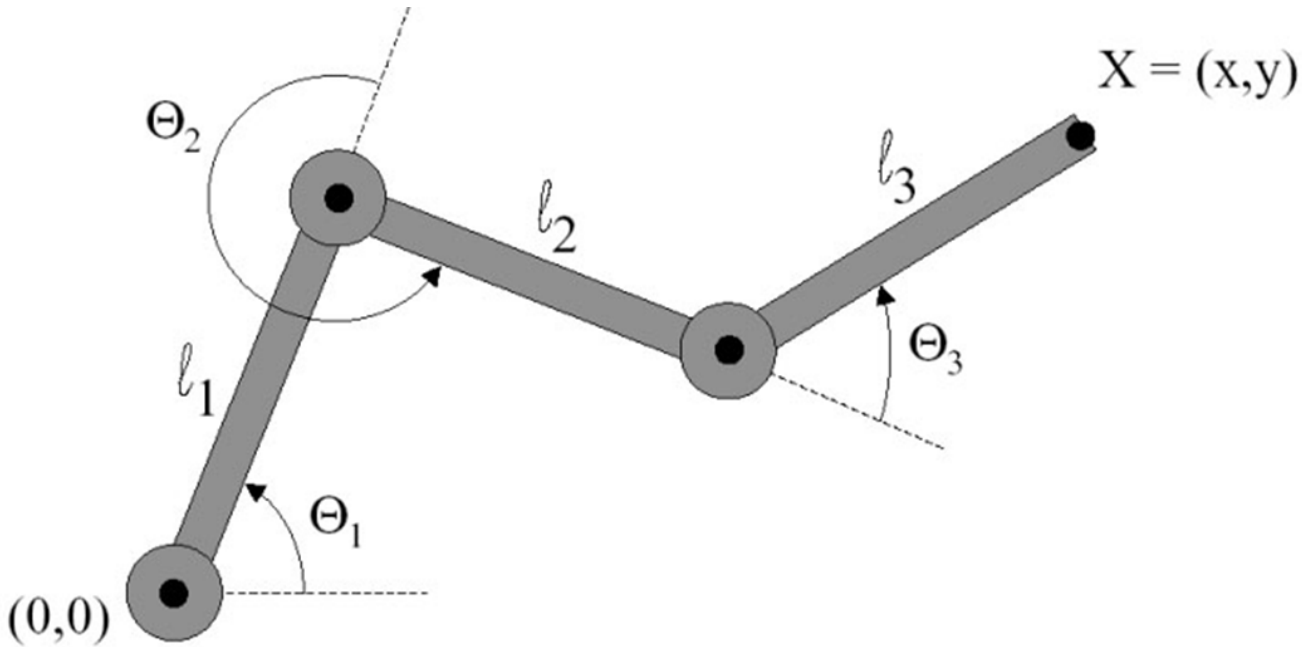
Példa

- Egyenletek száma: 2, ismeretlen változók száma: 3 $\Rightarrow$ Végtelen sok megoldás!



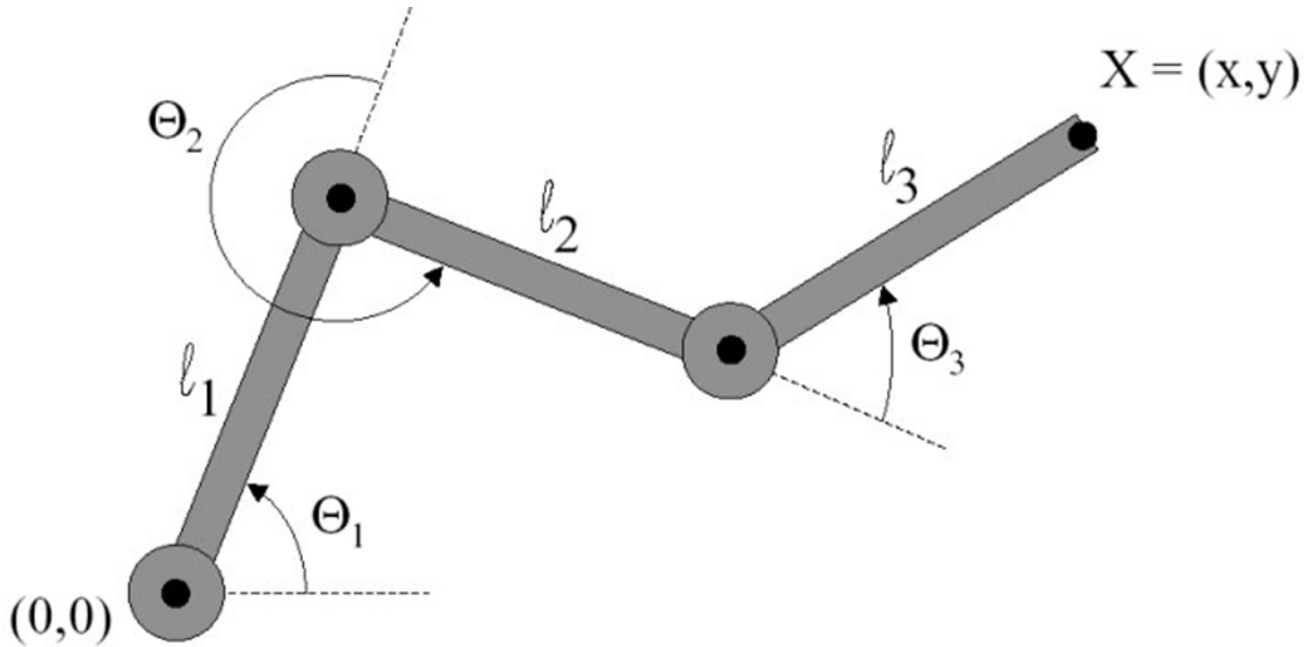
Példa

- Egyenletek száma: 2, ismeretlen változók száma: 3 $\Rightarrow$ Végtelen sok megoldás!
- Szabadságfok (degree of freedom – DOF)



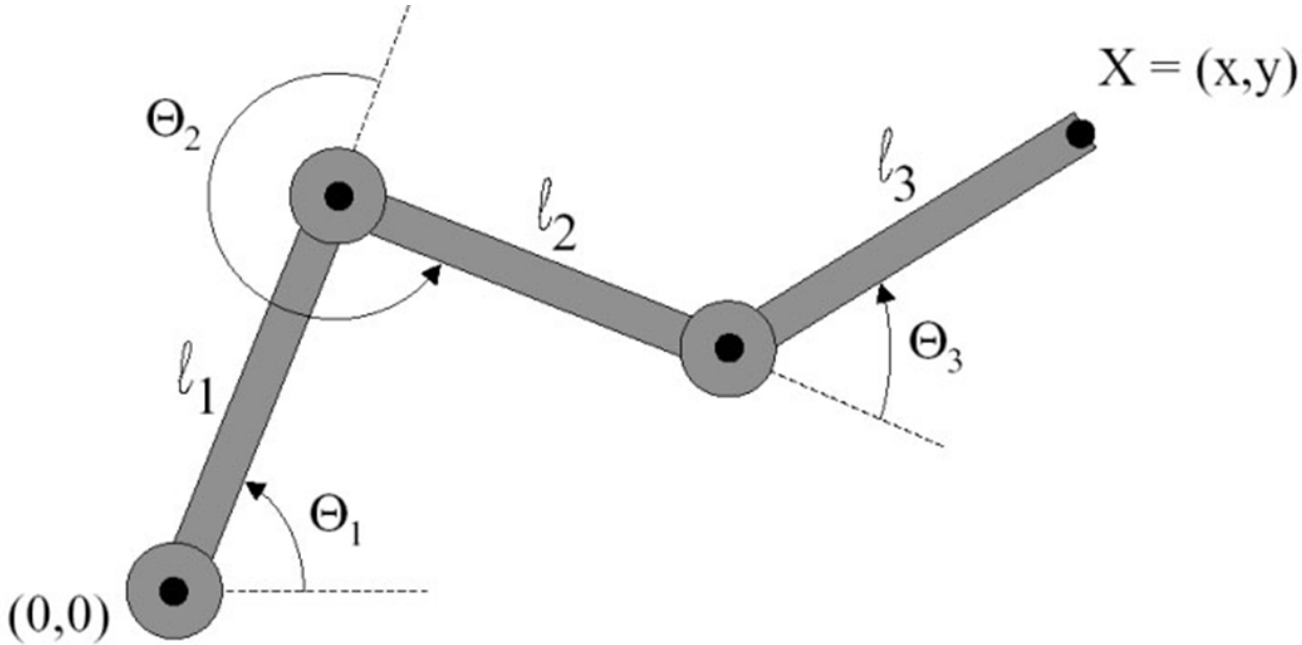
Példa

- Egyenletek száma: 2, ismeretlen változók száma: 3 $\Rightarrow$ Végtelen sok megoldás!
- Szabadságfok (degree of freedom – DOF)
- Rendszer DOF $>$ végberendezés DOF



Példa

- Egyenletek száma: 2, ismeretlen változók száma: 3 $\Rightarrow$ Végtelen sok megoldás!
- Szabadságfok (degree of freedom – DOF)
- Rendszer DOF $>$ végberendezés DOF
- Az emberi csontváz kb. 70 DOF!



Példa

Nem egyértelmű, ill. nem létező megoldás