PPKE ITK

A számítógépes grafika alapjai

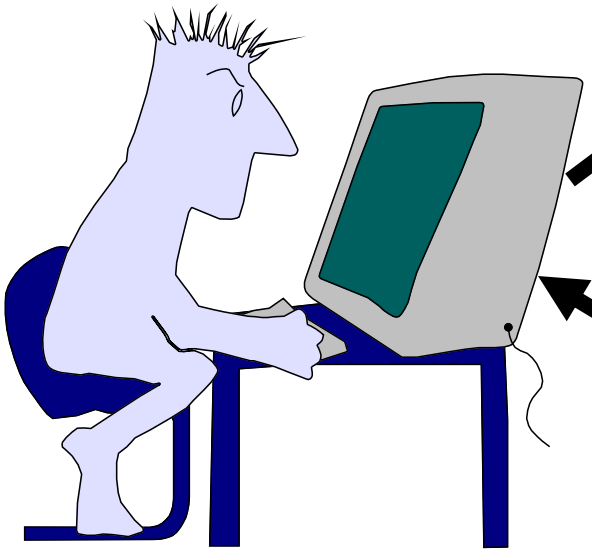
Animációk/2

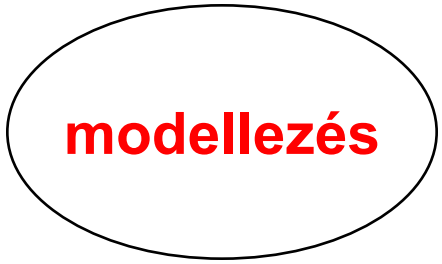
Előadó: Benedek Csaba

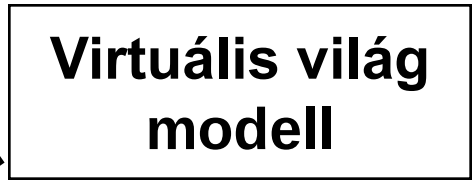
Tananyag : Szirmay-Kalos László, Benedek Csaba

Fides et Ratio

ANIMÁCIÓ (folytatás)







világ = sík

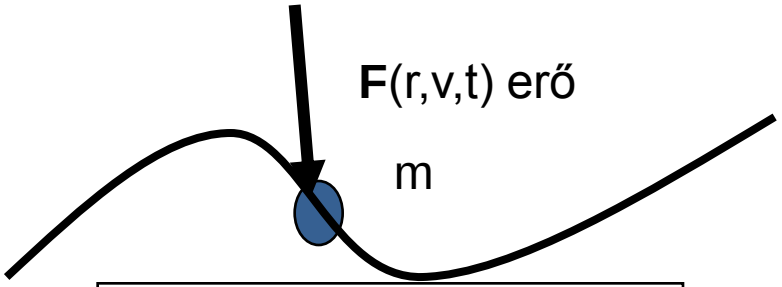
Metafórák:

- **2D rajzolás**

Fizikai animáció

- Erők (gravitáció, turbulencia stb.)
- Tömeg, tehetetlenségi nyomaték (F = ma)
- Ütközés detektálás (metszéspontszámítás)
- Ütközés válasz
  - rugók, ha közel vannak
  - impulzus megmaradás alapján

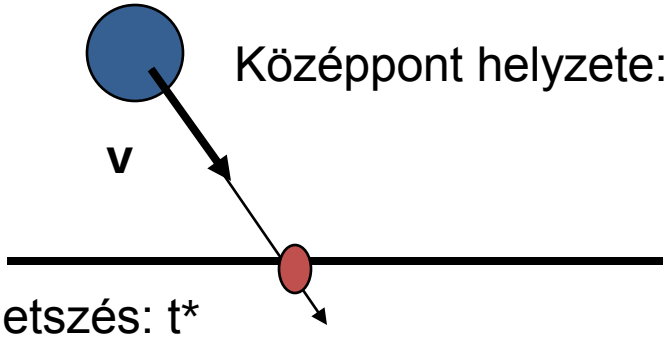
Egy kis mechanika



$$d\mathbf{r}/dt = \mathbf{v}$$
$$d\mathbf{v}/dt = \mathbf{F(r,v,t)/m}$$

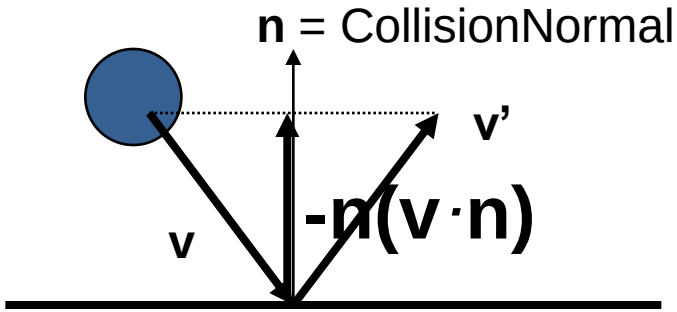
```
for ( t = 0; t < T; t += dt) {
    F = Erők számítása
    a = F/m
    v += a ·dt
    if ( ÜtközésDetektál )
        ÜtközésVálasz
    r += v ·dt
}
```

ÜtközésDetektál



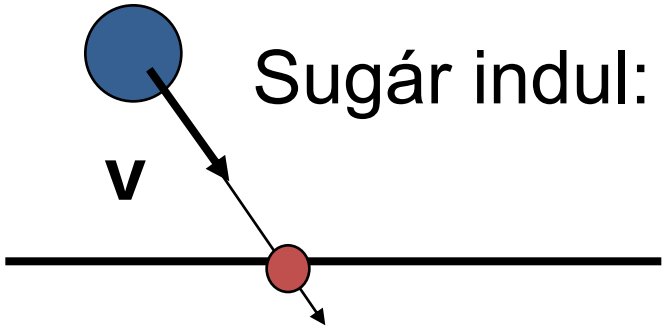
Ha t* < dt Collision

ÜtközésVálasz

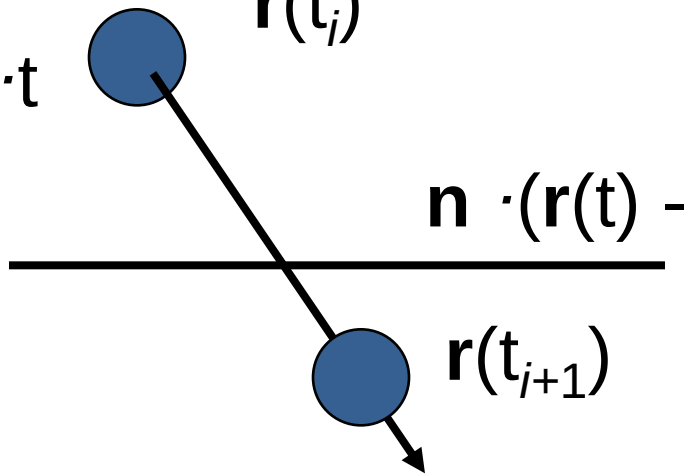


$$\mathbf{v'} = [\mathbf{v} - \mathbf{n}(\mathbf{v}\cdot\mathbf{n})] - [\mathbf{n}(\mathbf{v}\cdot\mathbf{n})\cdot bounce]$$

Folytonos-Diszkrét
ütközés detektálás pontra és féltérre



metszés: t*
Ha t* < dt Collision



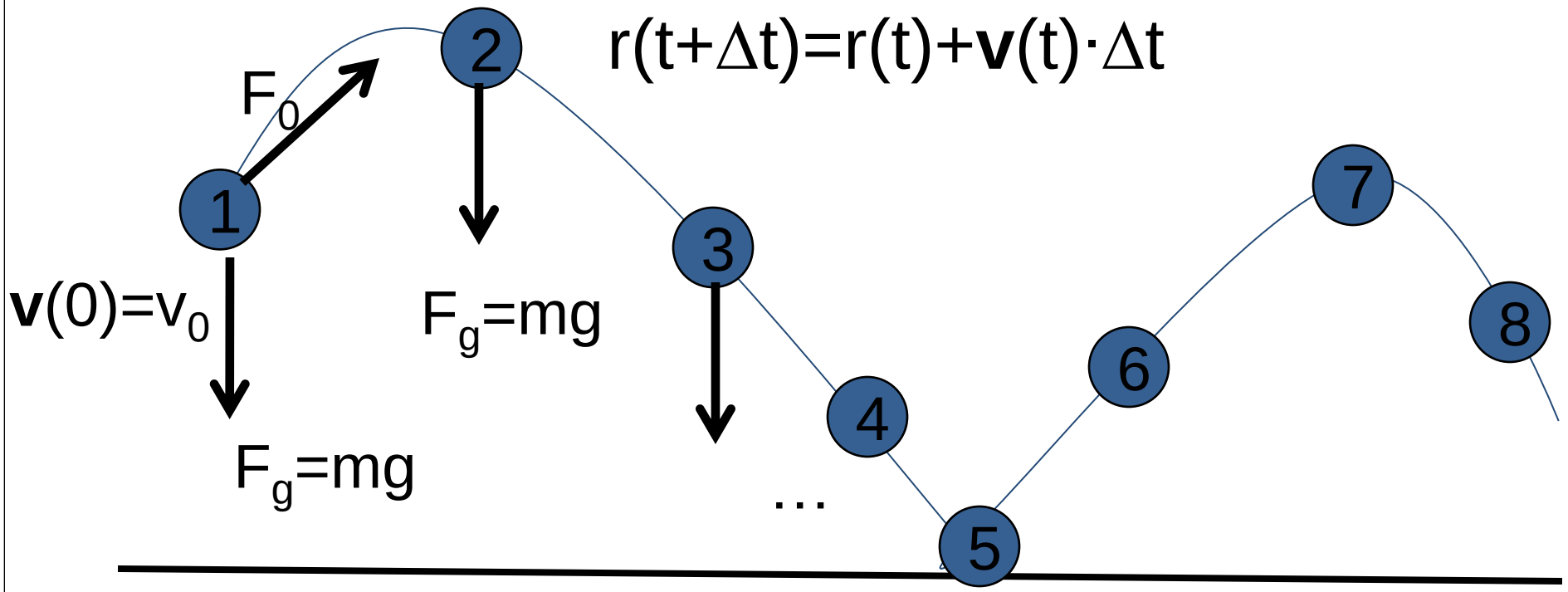
Ferde hajítás

$\Delta v = g \cdot \Delta t$

Mozgás+ gravitáció:

$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + g \cdot \Delta t$

$r(t+\Delta t) = r(t) + \mathbf{v}(t) \cdot \Delta t$



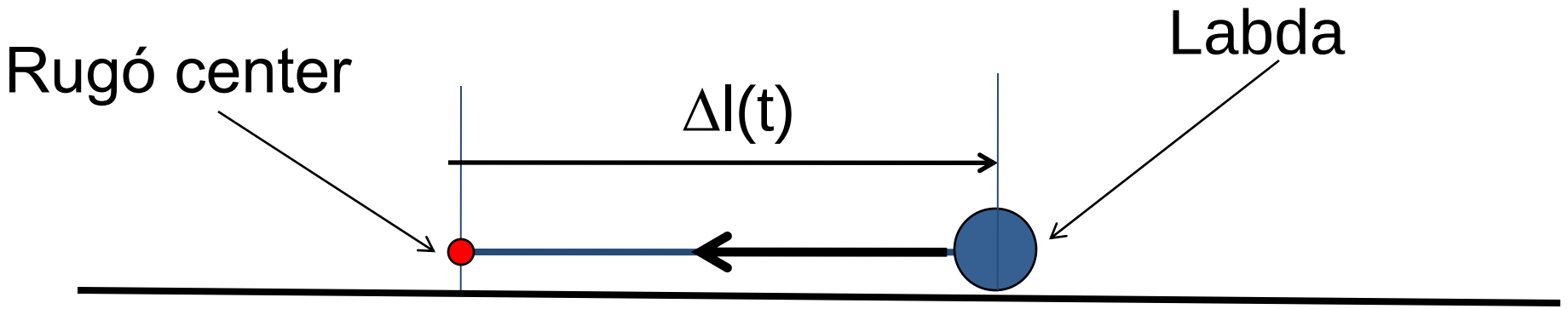
Rugalmas ütközés:
lásd előbb

Vízszintes rugón mozgó labda

$$F(t)=-\Delta l(t)\cdot D$$
$$a(t)=F(t)/m$$
$$\mathbf{v}(t+\Delta t)=\mathbf{v}(t)+a(t)\cdot\Delta t$$
$$r(t+\Delta t)=r(t)+\mathbf{v}(t)\cdot\Delta t$$

D: rugóállandó

$\Delta l(t)$: megnyúlás t-ben



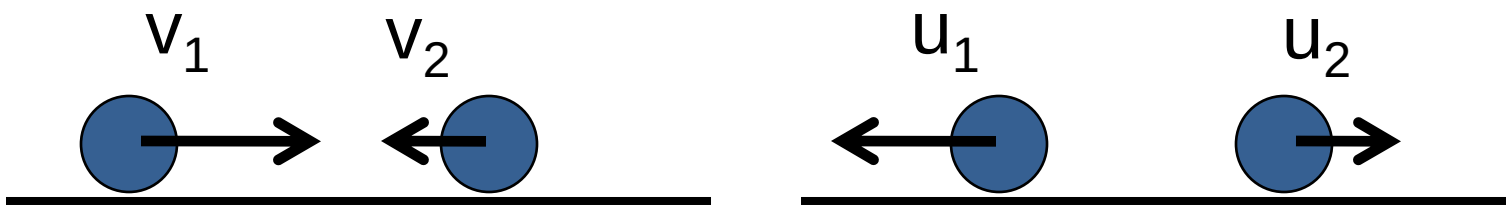
Golyók ütközése

Lendület megmaradás:

- $m_1v_1+m_2v_2=m_1u_1+m_2u_2$

Mechanikai energia megmaradás

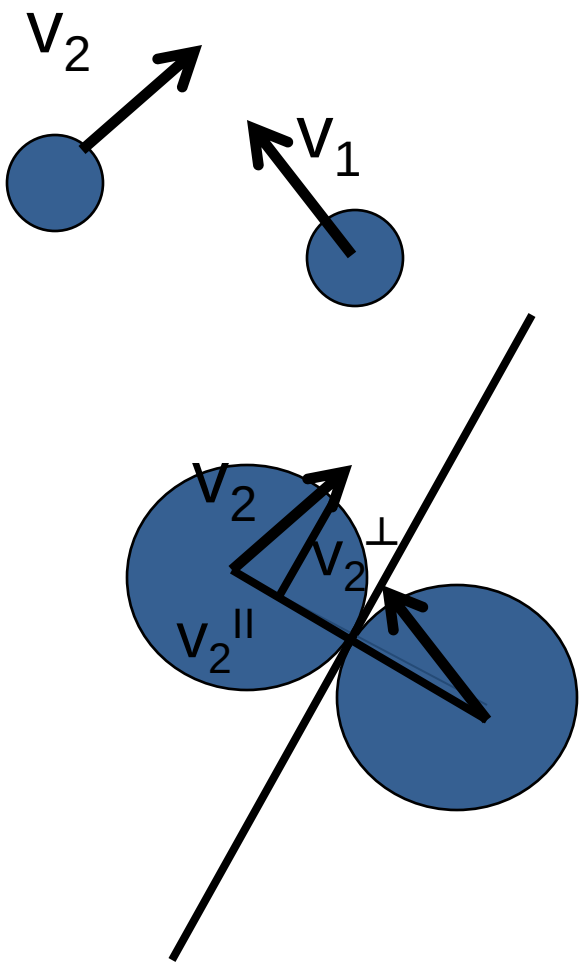
- $1/2\ m_1v_1^2+ 1/2\ m_2v_2^2= 1/2\ m_1u_1^2+ 1/2\ m_2u_2^2$



Megoldás:

$$u_1=\frac{(m_1-m_2)v_1+2m_2v_2}{m_1+m_2} \qquad u_2=\frac{(m_2-m_1)v_2+2m_1v_1}{m_1+m_2}$$

Ferde ütközés



$$v_1^{\perp} = u_1^{\perp}$$

$$v_2^{\perp} = u_2^{\perp}$$

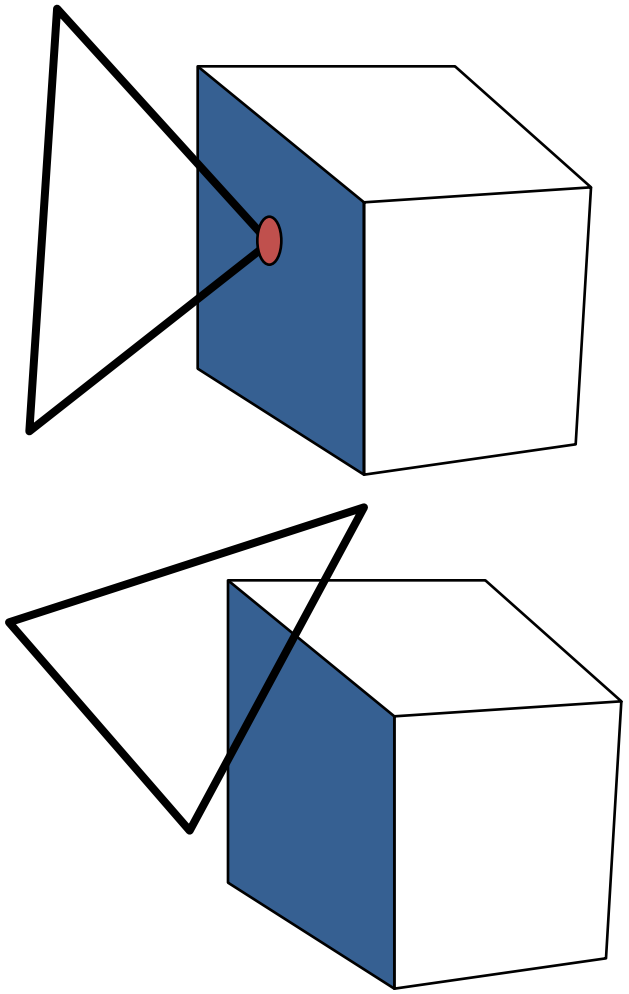
$$m_1 v_1^{II} + m_2 v_2^{II} = m_1 u_1^{II} + m_2 u_2^{II}$$

$$\tfrac{1}{2} m_1 v^2{}_1^{II} + \tfrac{1}{2} m_2 v^2{}_2^{II} =$$

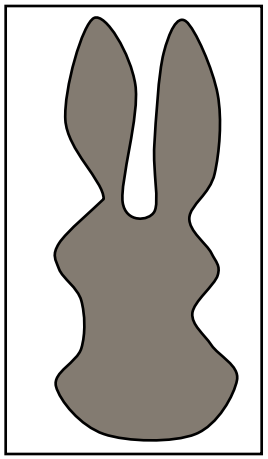
$$\tfrac{1}{2} m_1 u^2{}_1^{II} + \tfrac{1}{2} m_2 u^2{}_2^{II}$$

Ütközésdetektálás

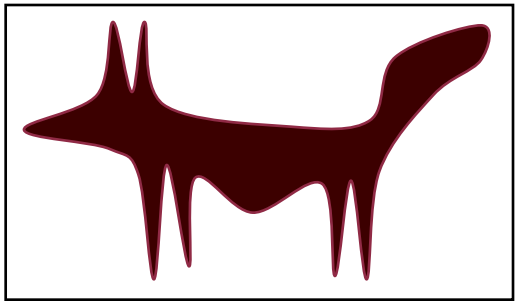
háromszög-háromszög



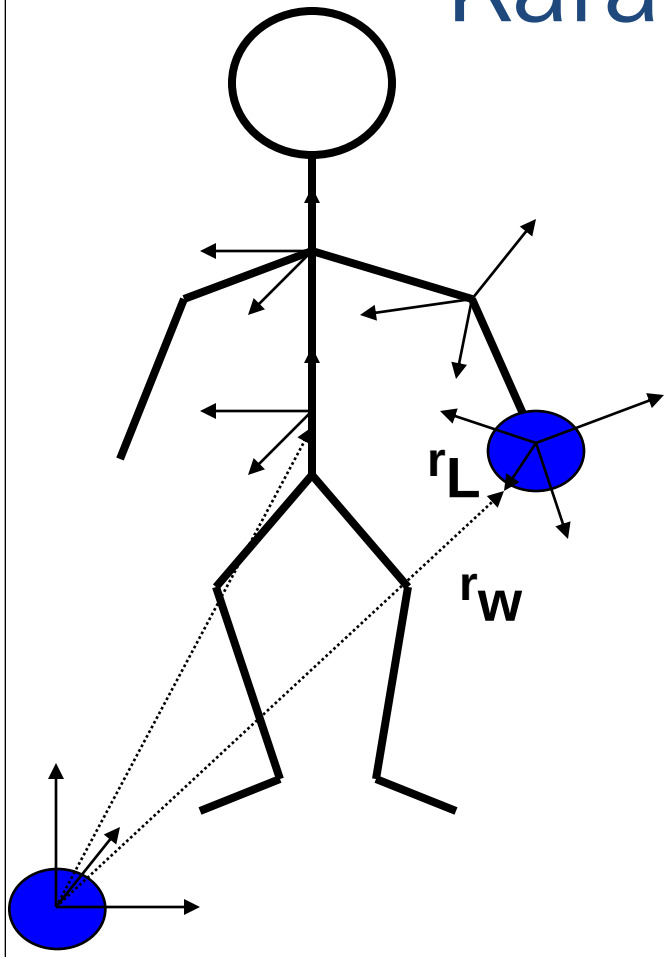
gyorsítás: befoglalók

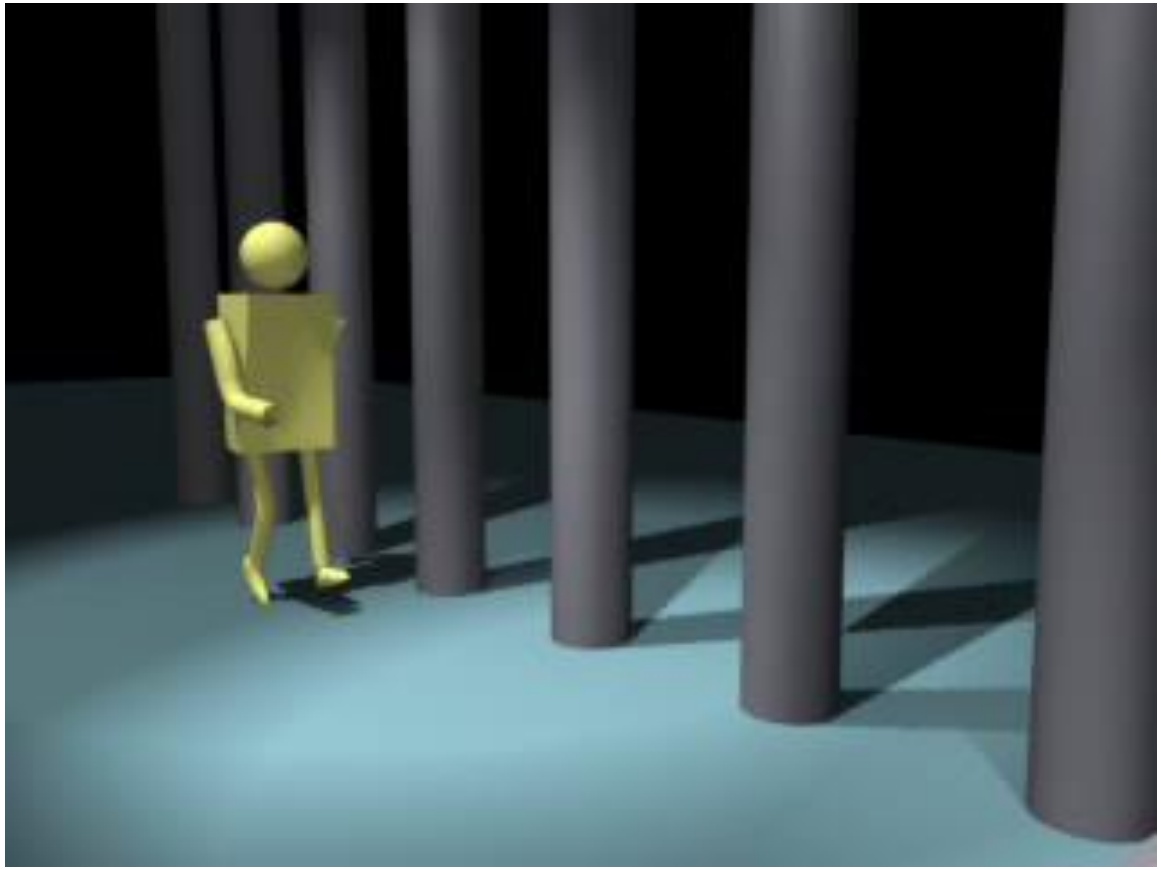


$O(n^2)$



Karakter animáció

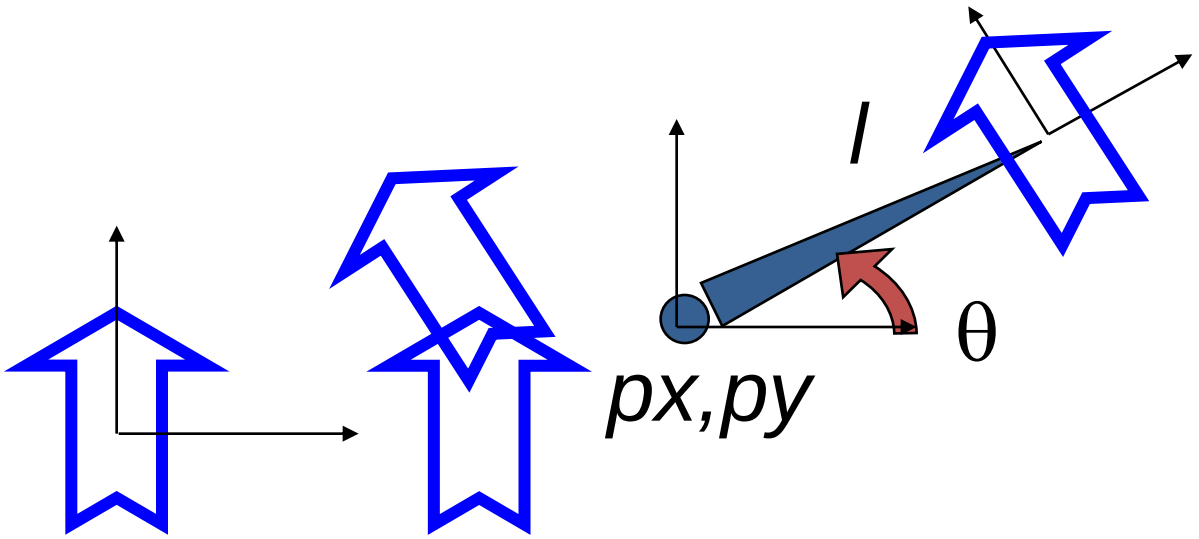




$$r_w = r_L \cdot R_{kéz} \cdot T_{alkar} \cdot R_{könyök} \cdot T_{felkar} \cdot R_{váll} \cdot T_{gerinc} \cdot T_{embe}$$

homogén koordináta 4-es

2D csont



$$\begin{bmatrix} x, y, 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & l & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ px & py & 1 \end{bmatrix}$$

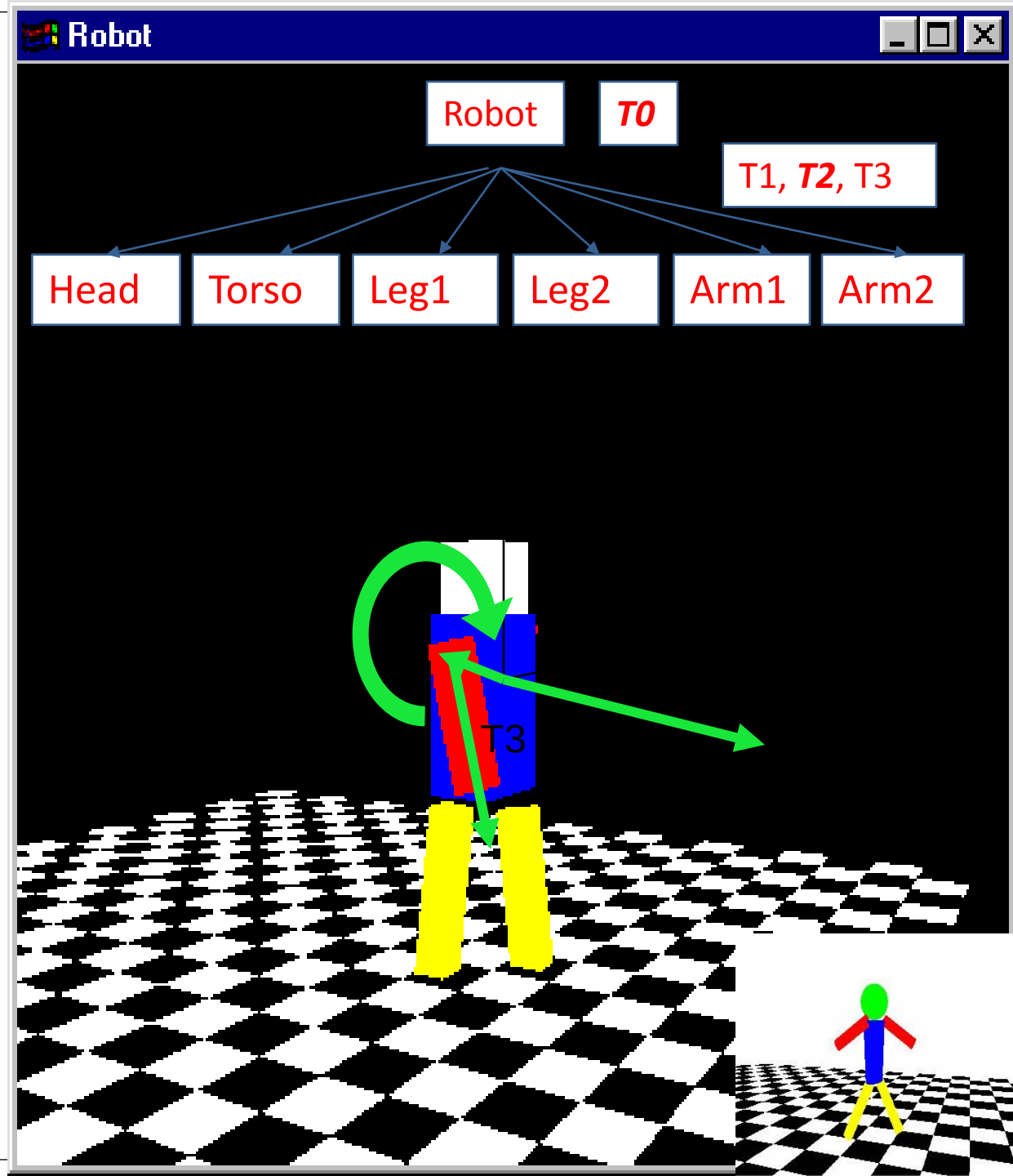
Robot példa

T0 = robot előrehalad
glTranslatef(xr, yr, zr);

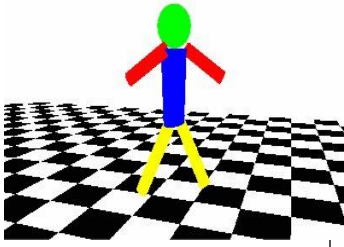
T1= kar elhelyése
glTranslatef(xv, yv, zv);

T2= forgatás
glRotatef(angle,
1.0f, 0.0f, 0.0f);

T3= skálázás
glScalef(1.0f, 4.0f, 1.0f);

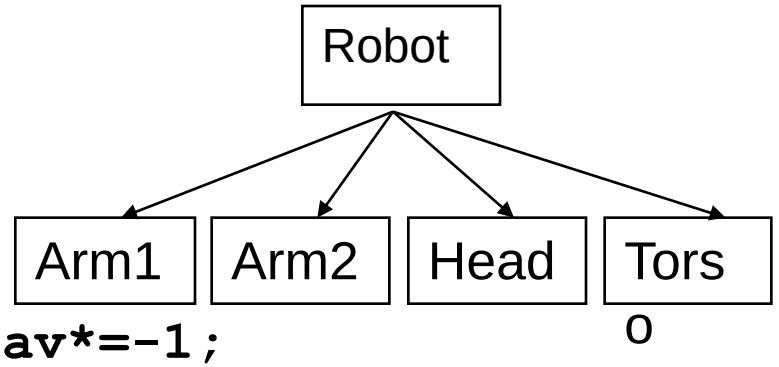


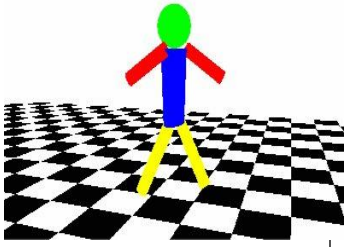
Robot rajzolás + animáció



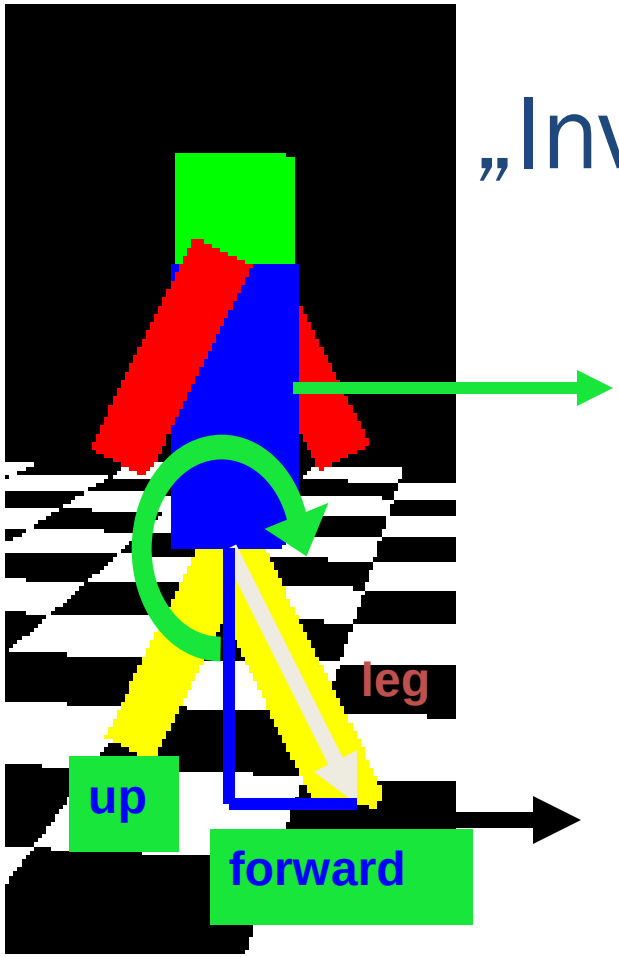
```
void DrawRobot(float dt) {
   xr += vx*dt; yr += vy*dt; zr += vz*dt;
   glPushMatrix();
      glTranslatef(xr, yr, zr);


      angle += av*dt;
      if (angle>30 || angle<-30) av*=-1;
      glColor3f(1, 0, 0);     // red
      glPushMatrix();
            glTranslatef(xv, yv, zv);
            glRotatef(angle, 1, 0, 0);
            glScalef(1, 4, 1);       // 1x4x1 cube
            DrawCube( );
      glPopMatrix();
     … Másik kéz, lábak, fej, törzs
   glPopMatrix();
}
```





„Inverz kinematika”



T0 = előrehaladás (forward, up) ???

T2 = forgatás (ang)

A láb (end effektor) földön legyen és ne csúszkáljon

```
forward += leg * fabs(sin(angNew) - sin(angOld));
up       = leg * (1 - cos(angNew));
```

Mozgáskövető animáció
(motion capture animation, MoCap):

- A létrehozott animációk valószerűsége függ:
  - a fizikai animációnál a fizikai modell és a szimuláció pontosságától,
  - a többi eddig tárgyalt esetben pedig az animátor ügyességétől
- Pontos fizikai modell felépítése csak egyszerű rendszerek esetén működik (vö: egy ember 100 csontja, ízülete, izma…)

Mozgáskövető animáció előtt…

- Rajzfilmek: a mozgás nem elégíti ki a fizikai törvényeket, mégis hihető és élvezhető

Rossz animáció

- …valódinak látszó (nem karikatúraszerű) karaktereknél zavaró!

MoCap

- MoCAP = „lopunk a természettől”
- Egy valódi szereplőt, aki lehet ember, állat, tárgy stb. rábírunk arra, hogy végezze el a kívánt mozgást, amit kamerákkal rögzítünk.
- Az elkészült filmekből kinyerjük a számunkra fontos mozgásadatokat, majd a modellünket ezekkel az adatokkal vezéreljük

Markeres MoCap:

- speciális ruha és szenzorok

Markeres MoCap:

- Arcon megjelenő pöttyök követése

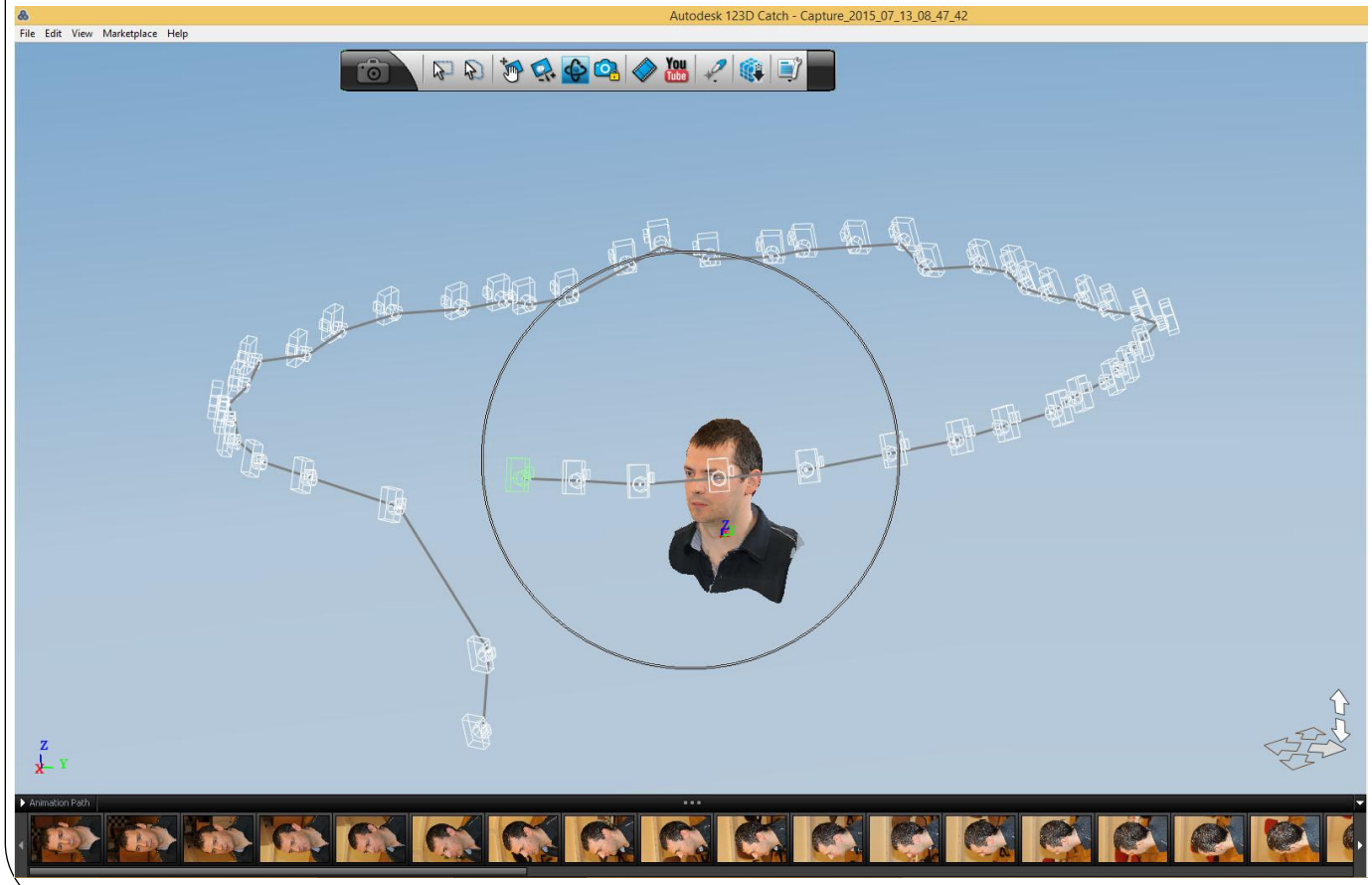
Mozgó avatarok felvétele markerek nélkül– 4D Studio

- Speciális zöld stúdió – kalibrált videokamerákkal
- Célok:
  - Ugyanazt az alakzatot egyszerre több nézőpontból rögzítjük
  - Dinamikus 3D modelleket készítünk mozgó alakzatokról

Panorámakép az MTA SZTAKI-ban található 4D stúdió belsejéről
http://vision.sztaki.hu/4Dstudio/

Statikus alakzatok felvétele

- Egy-két kameranézet elegendő, de több kamera → nagyobb pontosság és részletezettség
- A kamerák (vagy a szkenner) az objektum körül mozognak (körbejárják, vagy forognak)



Jankó Zsolt
A SZTAKI 4D Stúdió vezető fejlesztője

AUTODESK® 123D® CATCH

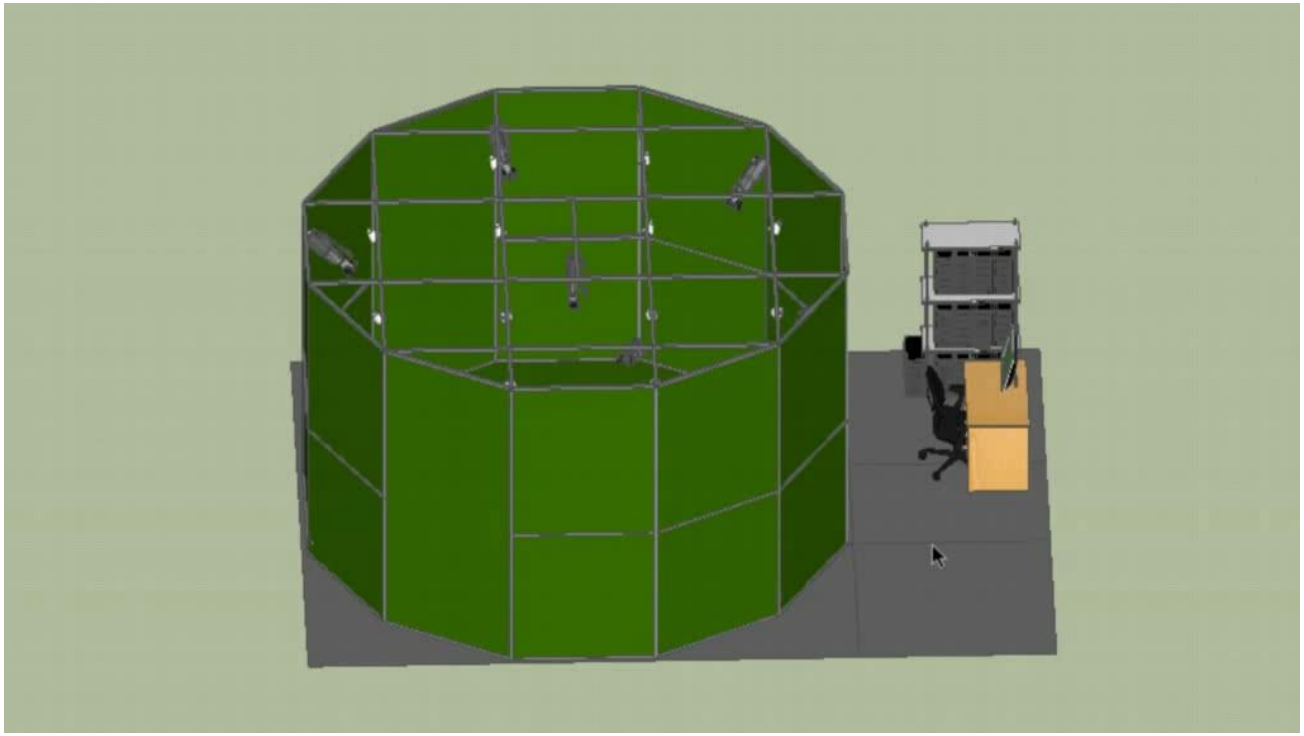
Dinamikus alakzatok felvétele

- Dinamikus (mozgó) alakzatok
  - Szimultán képkészítés több nézőpontból → többkamerás rendszer
  - Fix kamerapozíciók
  - Kevesebb nézet → rosszabb minőség
  - Redundáns (nagyon átlapolódó) kameranézetek → nagyobb pontosság

Kapcsolódó videók

Hardver komponensek

- Zöld stúdió
  - Dodekagon alapú henger
  - Masszív, merev acélváz
  - 12 kamera egyenletesen elosztva a helyszín körül + 1 felső kamera
  - Zöld függöny és szőnyeg → homogén háttér

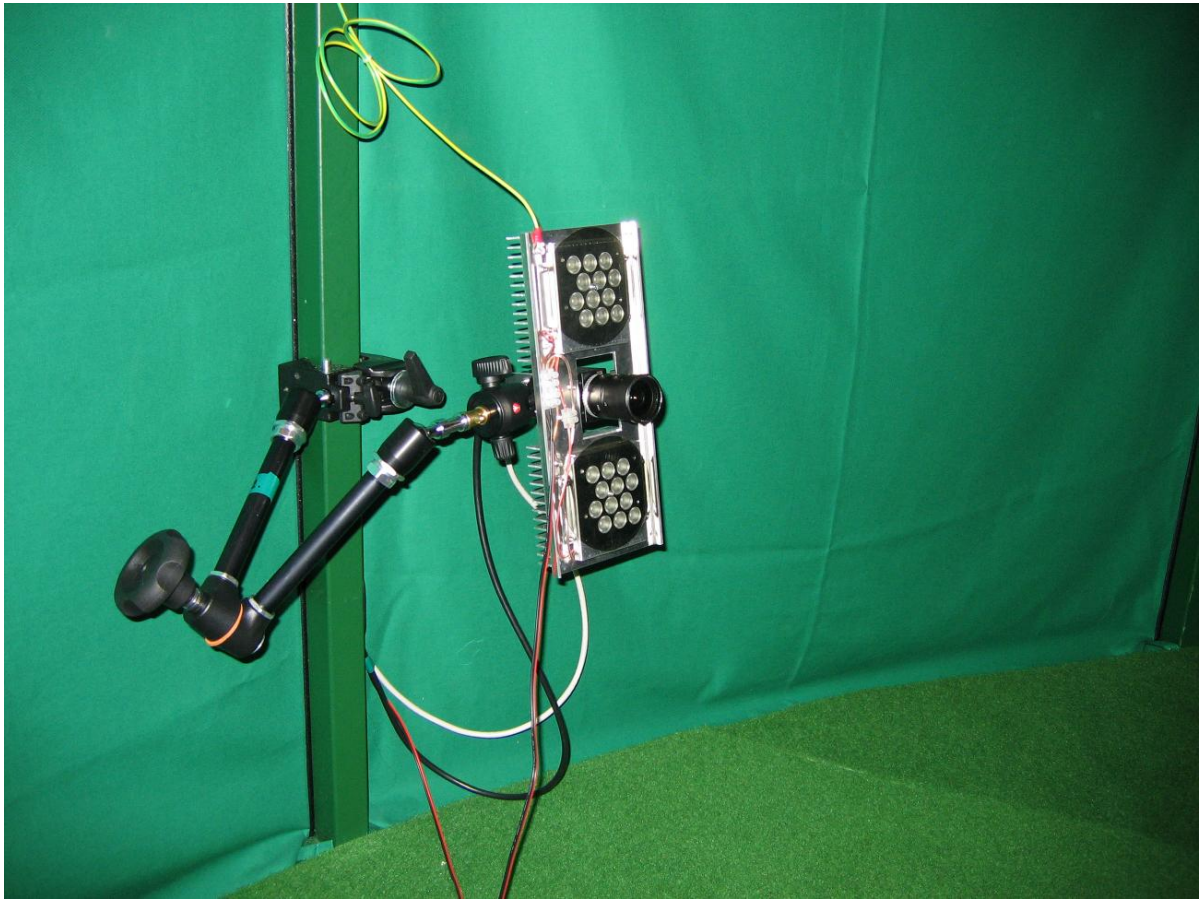


A video online elérhető az SZTAKI 4D Stúdió honlapjáról

Hardver komponensek

- Kamerák
  - Széles látószögű lencsék
  - 1624 x1236 pixel
  - 25 fps, GigE (Gigabit Ethernet)
- Megvilágítás
  - light-emitting dióda (LED) csoport minden kamera körül
  - Nagy frekvenciával lehet ki/be kapcsolni
- Mikro-kontroller
  - Szinkronizálja a kamerákat és a fényeket
    - Amikor egy kamera képet készít, a szemközti fényt ki kell kapcsolni
    - Lehetővé teszi a kamerák helyzetének teljesen rugalmas konfigurációját
- Számítási kapacitás
  - 7 hagyományos PC → 2 kamerára jut 1 PC

Hardver komponensek



Állítható platform a vázhoz rögzített kamerával és LEDcsoporttal

Stúdió szoftver - két fő szoftver blokk

- **`Studio`**
  - Képszegmentációs szoftver

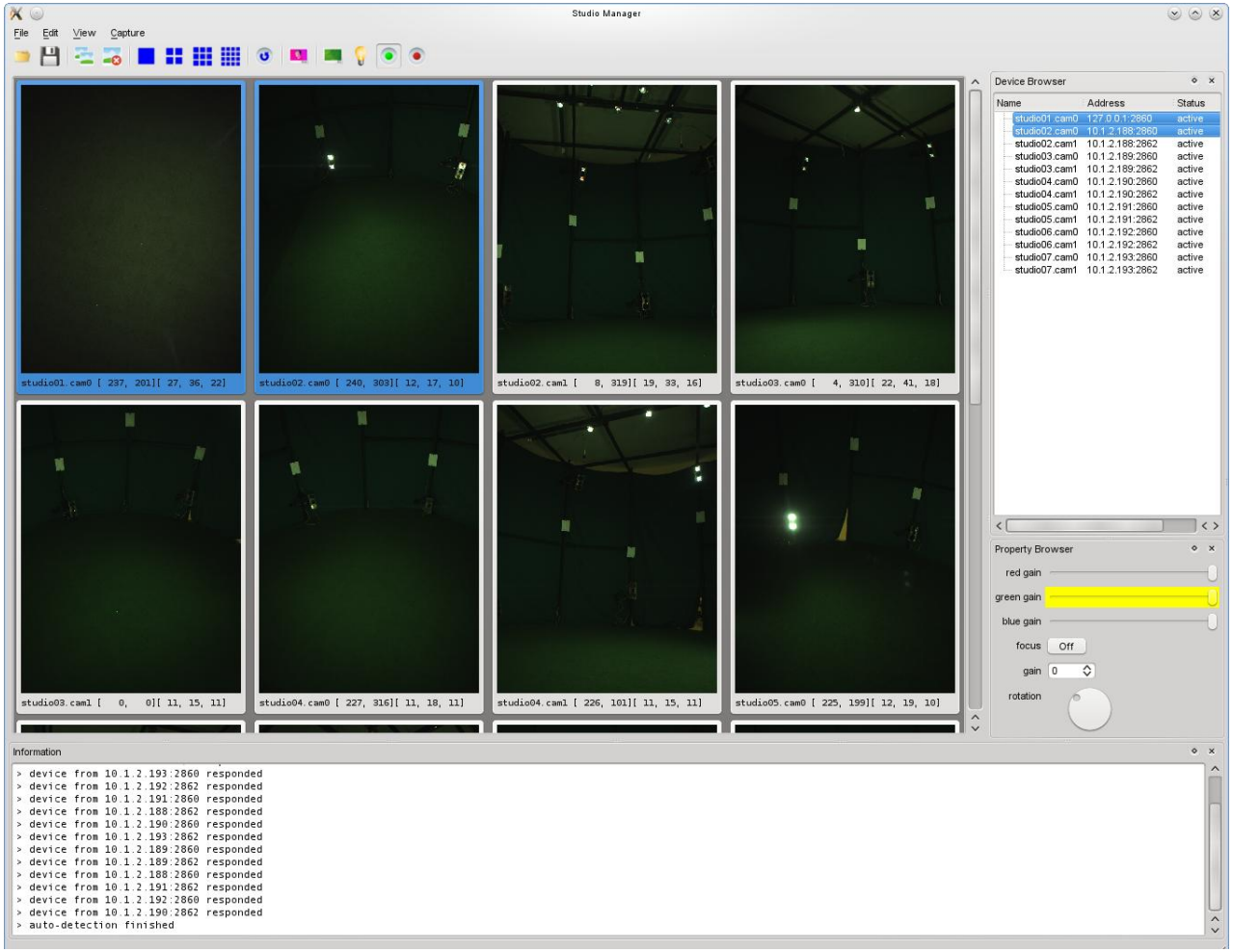
    → videofelvétel
- **`ModelMaker`**
  - 3D rekonstrukciós szoftver

    → dinamikus 3D modellek készítése
- A teljes szoftverrendszert a SZTAKI-ban fejlesztették
  - OpenCV funkciókat használ

Szoftver komponensek

- Kiválasztja az adatgyűjtés során használandó kamerát
  - Használhatjuk a teljes kamerahalmaz egy részhalmazát
- Konfigurálja a kamerákat
  - Fókusz, erősítés, fehéregyensúly stb.
- A kamerarendszert kalibrálja (sztereo rekonstrukcióhoz kell)
  - Intrinsic (belső) paraméterek (fókusz, lencse torzítása)
  - Pozíció és orientáció a közös koordináta rendszerben → extrinsic (külső) paraméterek
- Szinkronizálja a kamerákat és a megvilágítás vezérlőjét
- Szinkronizált videoszekvenciákat vesz fel

Az adatgyűjtő szoftver inerfésze



A kamerarendszer kalibrációja

- OpenCV-s rutinokon alapul (Z.Zhang módszere)
- Az operátor mozgat és a minden kamerának megmutat egy 7×6 sima kalibrációs sakktáblamintát
  - Képek készülnek különböző a kalibrációs minta különböző orientációival
  - A detektált sarkok megfeleltetése egyértelmű intrinsic parameters of each camera

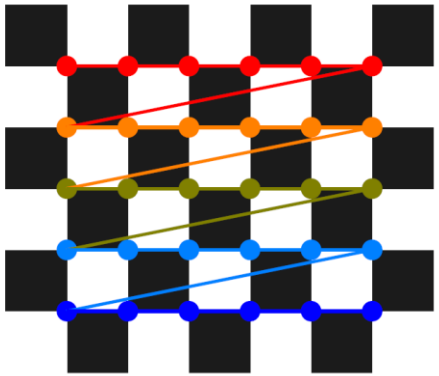
  →az egyes kamerák lencséjének torzító paraméterei

  → relatív pozíció és orientáció a szomszédos kamerák között
- Sakktábla mintát asztalra tesznek és a felső sor kameráinak megmutatják

  → a felső sor kameráinak extrinsic paraméterei
  - Az alsó sorban lévő kamerák relatív pozíciói már ismertek

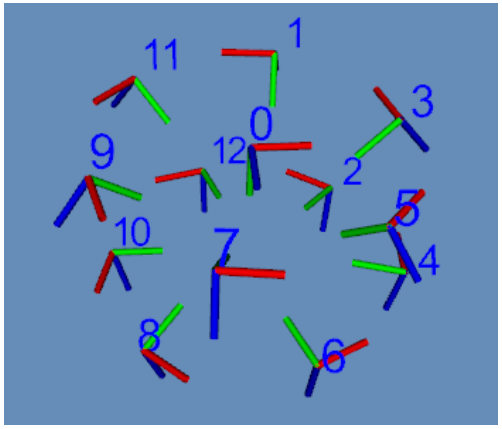
  → valamennyi kamera extrinsic paraméterei ismertek

Kamerarendszer kalibrációja



Sarokpontok sorrendje

sakktábla mutatása



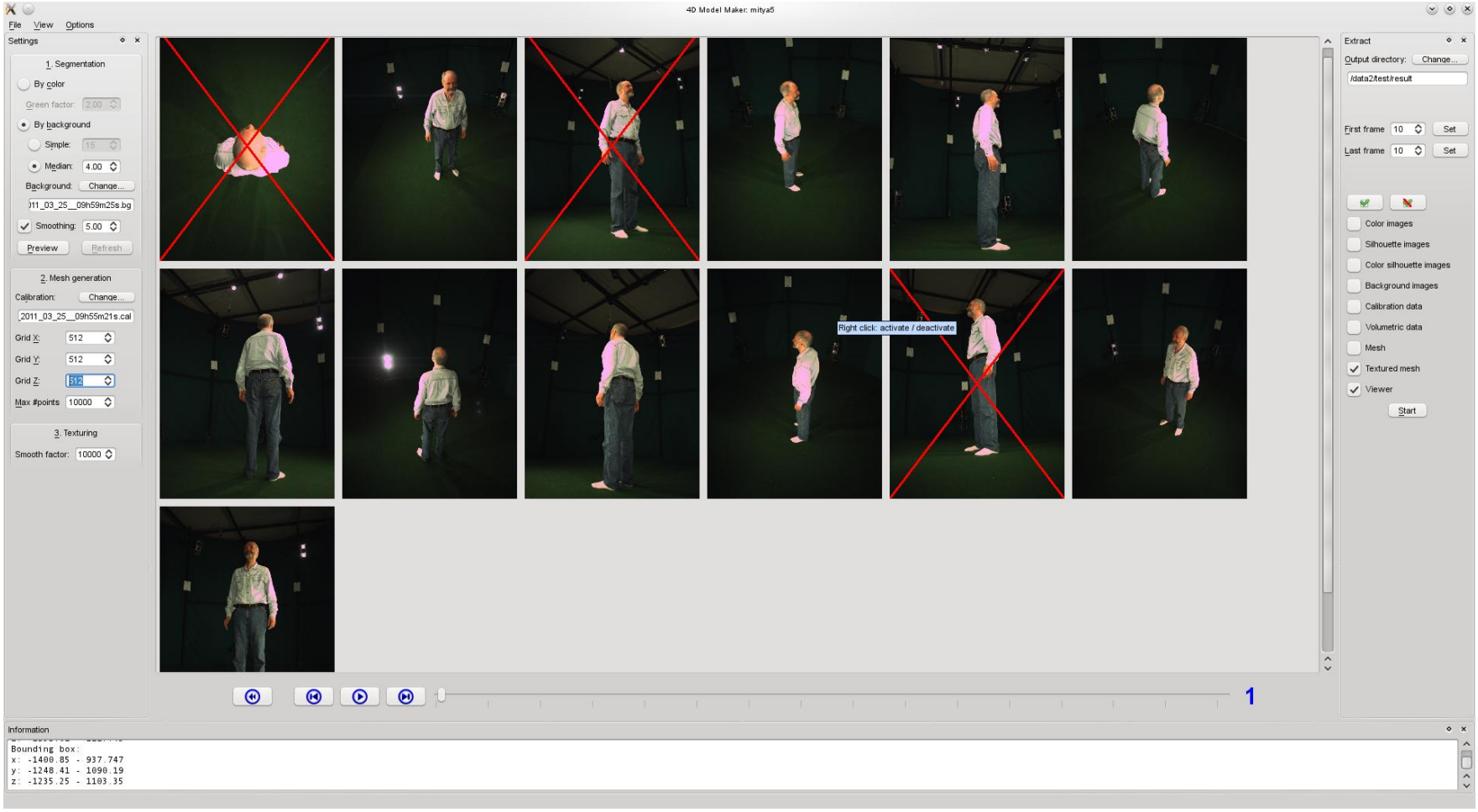
extrinsic (külső) kalibráció

- Aszimmetrikus mintát használunk a sarokpontok egyértelmű azonosítására a forgatott minta esetén
- A kalibrációs mintát forgatjuk hogy az orientációja változzon
- A külső paramétereket a közös koordinátarendszerben definiáljuk

Rekonstrukciós szoftver – lépésről lépésre

1. Nyerjünk ki színes képeket a nyers adatokból
2. Szegmentáljuk a színes képeket előtér és háttér osztályokba
3. Készítsünk volumetrikus modellt a Visual hull algoritmussal
4. Készítsünk háromszögelt hálót a volumetrikus modellből
5. Adjunk textúrát a modellhez

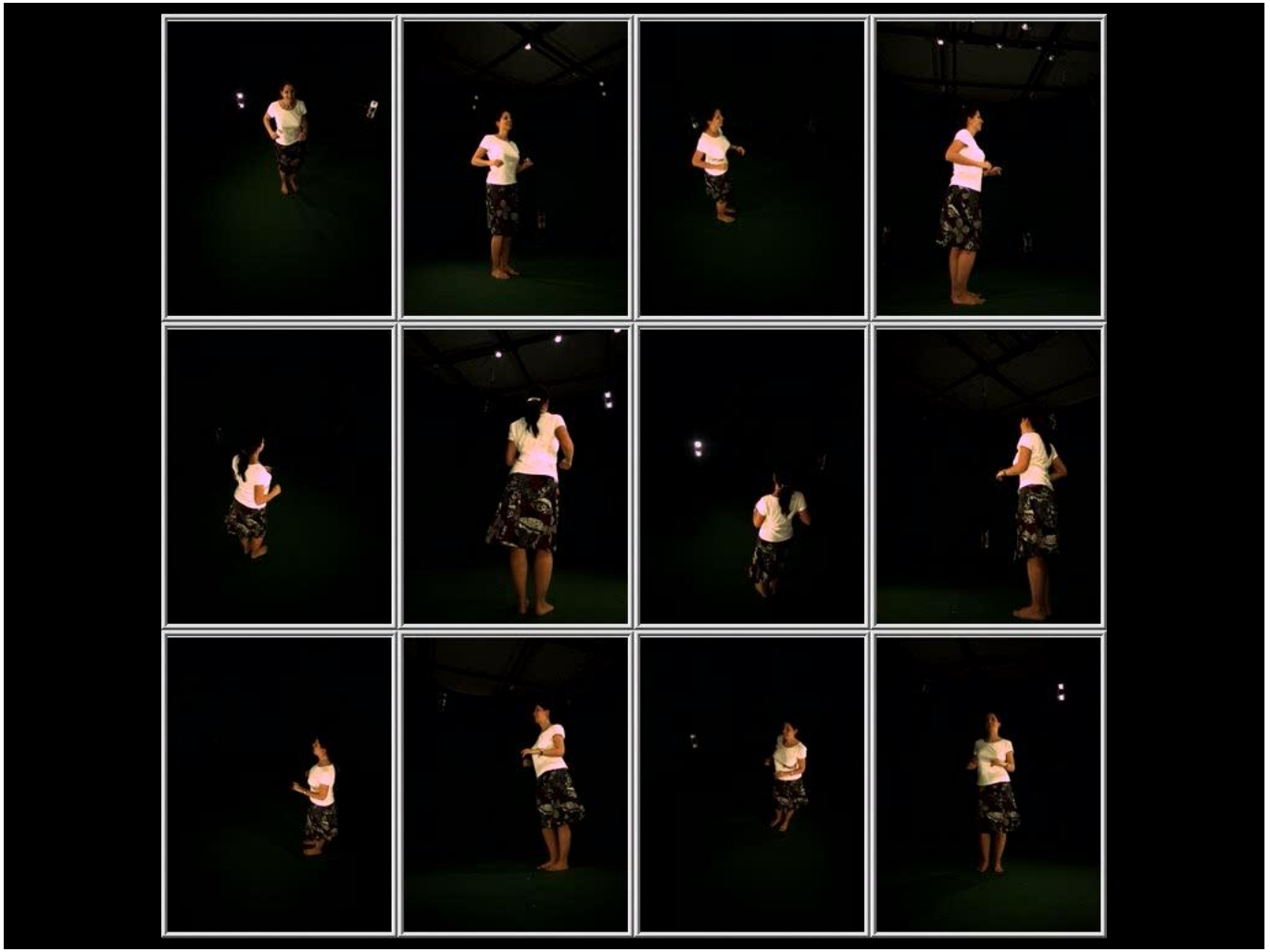
A rekonstrukciós szoftver interfésze



Rekonstrukciós szoftver – lépésről lépésre

- Alap konfiguráció: a videófolyamok képkockáit külön-külön dolgozzuk fel
  - Tovább fejlesztés: időbeli koherencia kihasználása
- A színes képek binarizálása
  - Kódolás: 0 a háttér 255 az objektum
  - **Objektumok sziluettjei a** bináris képeken
- A szegmentálás alapjai
  - Feltesszük, hogy a háttér nagyobb mint az objektum
  - Referencia háttérkép eltárolása az objektum megjelenése előtt
  - A rögzített RGB képet gömbi koordinátarendszerbe konvertáljuk (pl. CIE L*a*b* színtér) → növeljük a robusztusságot a megvilágítás változásaira
  - Az aktuális képkocka és a referencia háttérkép közötti különbségkép számítása
  - Az objektumok „outlier” értékekként jelennek meg, amihez robusztus outlier detekciós eljárás szükséges

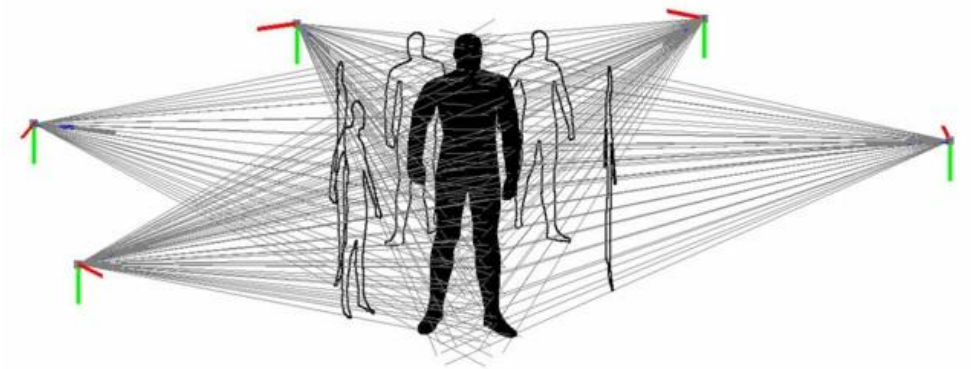
A szegmentálás video-illusztrációja

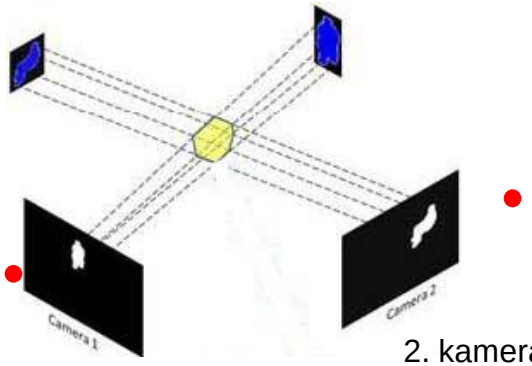


Volumetrikus modell készítése

- Alkalmazzunk „Alak a sziluettekből” (shape-from-silhouettes) rekonstrukciót a vizuális törzs (visual hull) elkészítéséhez
  - Maximális térfogatrész kinyerése, ami konzisztens a sziluettekkel
- A sziluettképek visszaprojektálása a 3D térbe
- Általánosított kúpok metszetének számítsa
  - A 3D objektum befoglaló térfogatát nyerjük kit

  → egyes konkáv részletek elveszhetnek
  - Több kamera → jobb geometria



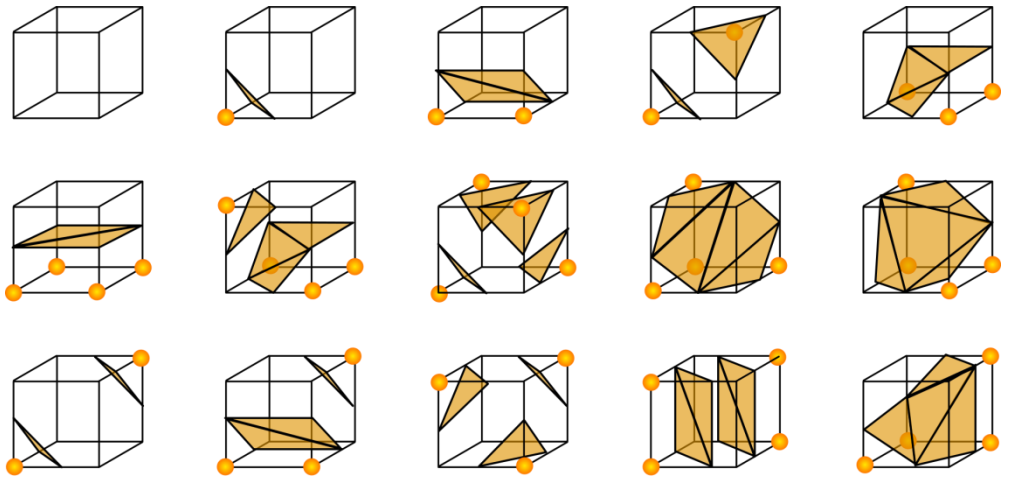


1. kamera középpontja

2. kamera középpontja

Háromszögháló generálása a volumetrikus adatból

- A standard marching cubes algoritmus használata
- Minden voxelnél tekintsük a 8 szomszédságot → kocka
- Határozzuk meg az(oka)t a poligon(oka)t, amelyek szükségesek a felület adott kockán áthaladó részének a reprezentációjához
  - Szabályok halmaza arra, hogy milyen háromszögeket kell behúzni az egyes konfigurációkban
- Utófeldolgozás: szűrés, decimálás



A pontok jelölik az 1-es értékű csúcspontokat a volumetrikus modellben

A háromszögháló textúrázása

- Minden háromszöghöz számítsuk ki a láthatóság mértékét
  - A háromszög látható a kamerából
  - A háromszög normálvektora a kamera felé mutat
- Írjunk fel egy költségfüggvényt a láthatóság és regularizációs tagok alapján
  - A regularizáció csökkenti az erős textúraéleket szomszédos háromszögek között

  → egyensúly láthatósági és a simasági kényszerek között
- Minimalizáljuk a költségfüggvényt gráfvágásokkal

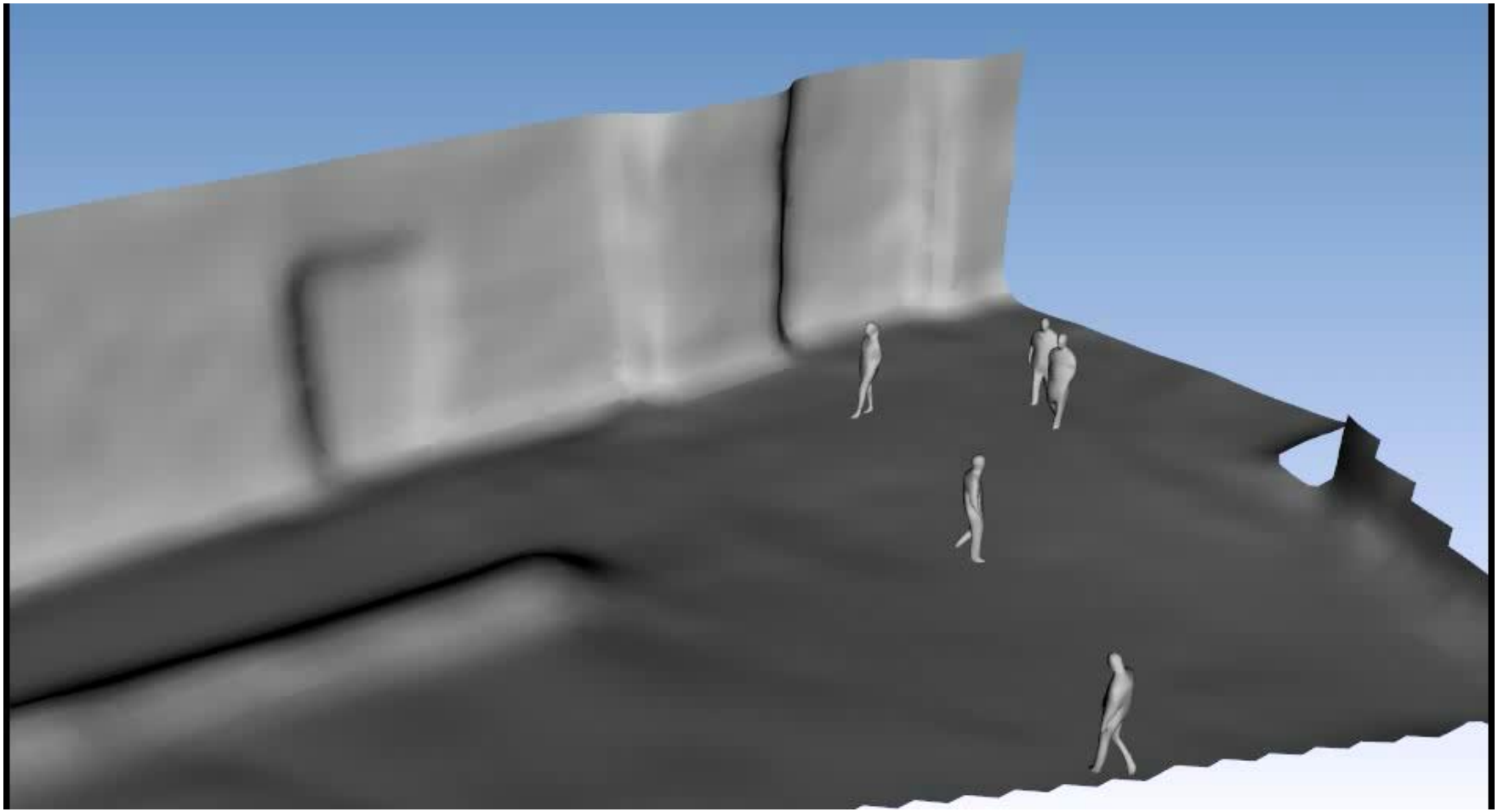
  → keressük meg a legjobb képet a háromszög textúrázásához
- A textúrázás minősége a felületi geometria precizitásától függ
  - A Visual hull és Marching cube algoritmusok okozhatnak pontatlan normálvektorokat

  → a textúra részletei elveszhetnek, vagy torzulhatnak

Mozgó virtuális szereplők- 4D stúdió

A video online elérhető az i4D Projekt weblapjáról

Integrált virtuális valóság modell kimenete



A video online elérhető az i4D Projekt weblapjáról