Geometriai modellezés

Affin Transzformációk

* 2D forgatás (***RotationXYMatrix***)
* 3D eltolás (***TranslateMatrix***)
* 3D skálázás (***ScaleMatrix***)
* 3D forgatás (Rodrigues formula)

(d a forgatási tengely)

* Origót fixen hagyják ezek a trafók 🡪 fix (pivot) pontnál
  + pivot az origóba
  + forgatás / skálázás
  + pivot visszatolása

2D Képszintézis

* A képen szöveg, képkeret, képernyőkép látható

  Automatikusan generált leírásfényképezés
  + világ koordinátarendszeréhez képest elmozgatjuk a modell elemeit
  + veszünk egy kamera ablakot, levágjuk a kilógó részeket
  + ezt beillesztjük a viewport-ba

# Pixel vezérelt rendering

* Object alaposztály
  + szín,
  + fv: benne van-e a paraméterként átadott másik objektum
* Scene osztály (a virtuális világ tárolására)
  + objektumok lista
  + picked (kiválasztott objektum)
  + Pick(x,y) 🡪 rákattintott elem kiválasztása
  + BringToFront() 🡪 a lista elejére viszi a kiválasztott elemet
  + Render() 🡪 x-y irányban ciklus: ha az adott pixel benne van bármely objektumban (legfelső számít), akkor kiszínezi a megfelelő pixelt az ő színére
* Ez egyszerű, de NAGYON LASSÚ!

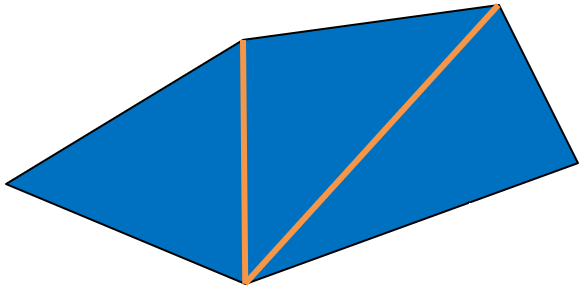
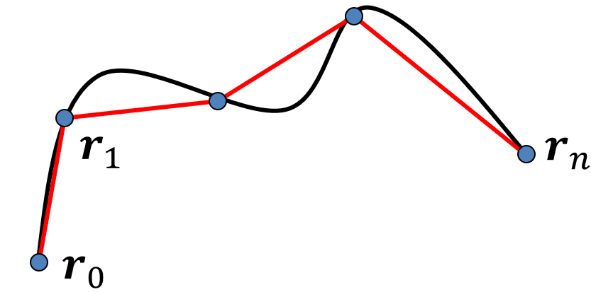
# Objektum vezérelt 2D képszintézis

* átalakításokkal (nyújtás, elmozdítás) a viewport-ba helyezzük a modellt
* GPU megjelenítési csővezeték

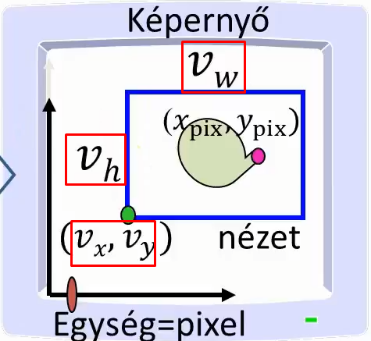
1. A képen diagram látható

   Automatikusan generált leírásreferencia helyzet alapból
2. vektorizáció
3. modell transzformáció
4. kamera transzformáció   
   (normalizált eszközkoordináták)
5. vágás + nézet transzformáció
6. raszterizáció

# Vektorizáció és háromszögesítés

* vektorizáció
  + görbék 🡪 nyílt töröttvonallá
    - minél több szakasz, annál jobban hasonlít
  + terület 🡪 zárt töröttvonallá
    - azaz poligon 🡪 háromszögek
* OpenGL 🡪 pontok, vonalak, háromszögek
* háromszögekre bontás 🡪 diagonálok mentén
* csúcs fül 🡨🡪 diagonál
  + füleket levágjuk
  + gyorsabb algoritmus

# Transzformációk

* View transzformáció 🡪 kameraablak közepét az origóba
* Projekció 🡪 kameraablakot az egységnégyzetbe
  + a szélesség és magasság lezsugorítása
* Ezeknek létezik inverz transzformációja is!
* Modellezési transzformáció 🡪 world = model\*mScale\*mRotate\*mTranslat
* Zoom 🡪 size \* s
* Pan (mozgatás) 🡪 c + t
* A saját Object osztályainknál felül kell definiálni a **GenVertexData** függvényt (vertices.push\_back)
* Új vertex és fragment shaderek
* View**port** transzformáció 🡪 normalizált dc-ból a képernyő koordinátákba
  + így kapjuk meg az pixelek x,y,z koordinátáit
  + , … (ld. diasor)

# Vágás

* ha egy pont kilóg az keretből, akkor levágjuk
  + ekkor, ha az pl. oldalt vizsgáljuk: (line-line intersection)

A képen diagram látható

Automatikusan generált leírás

* Sutherland-Hodgman poligonvágás algoritmus
* 3D-ben homogén koordinátákkal
  + (mivel )
  + mert a GPU homogén koordinátákat használ (és 3D-ben)
* vesszük a síkot 🡪 ha , akkor jó, ha nem, akkor kidobjuk
* (ahol a külső ponthoz tartozik)
  + ebből kifejezzük -t, majd visszahelyettesítjük képletébe, ez lesz a síkon lévő pont

# Raszterizáció

* Eddig:
  + modell 🡪 vektorizáció (CPU)
  + VAO, VBO 🡪 MVP transzformáció 🡪 vágás 🡪 viewport trafó (GPU)
* Most:
  + raszterizáció 🡪 pixel ops 🡪 frame buffer (Pixelek)
* Raszterizáció 🡪 pixelek kiszínezése
* Szakasz rajzolás 🡪 pontjai kielégítik az egyenletet
  + ezt kerekítjük
* Inkrementális elv 🡪 előző y-ból kiszámítható az új
* Natív háromszög kitöltése 🡪 jobb végtelenbe visszük a pontokat, megnézzük hányszor metszik a háromszöget. Ha páratlan sokszor, akkor benne vannak, ki kell színezni.
  + itt is van inkrementális elv

A képen diagram látható

Automatikusan generált leírás

* Pixel színének megh atározása
  + A képen szöveg, névjegykártya, clipart látható

    Automatikusan generált leírásuniform az egész objektum
  + csúcspont tulajdonságokból interpolált szín
  + A képen nyíl látható

    Automatikusan generált leírástextúrázás