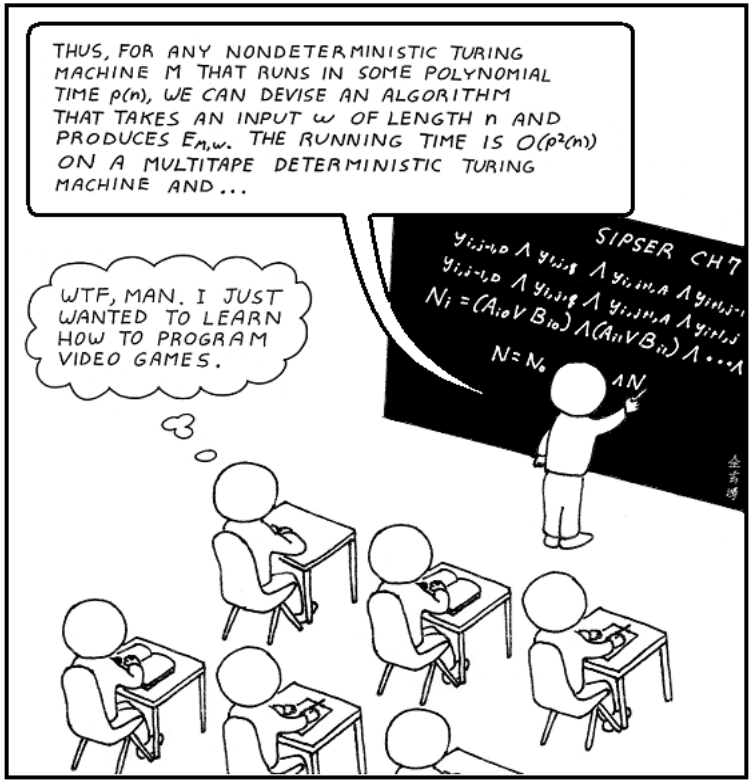
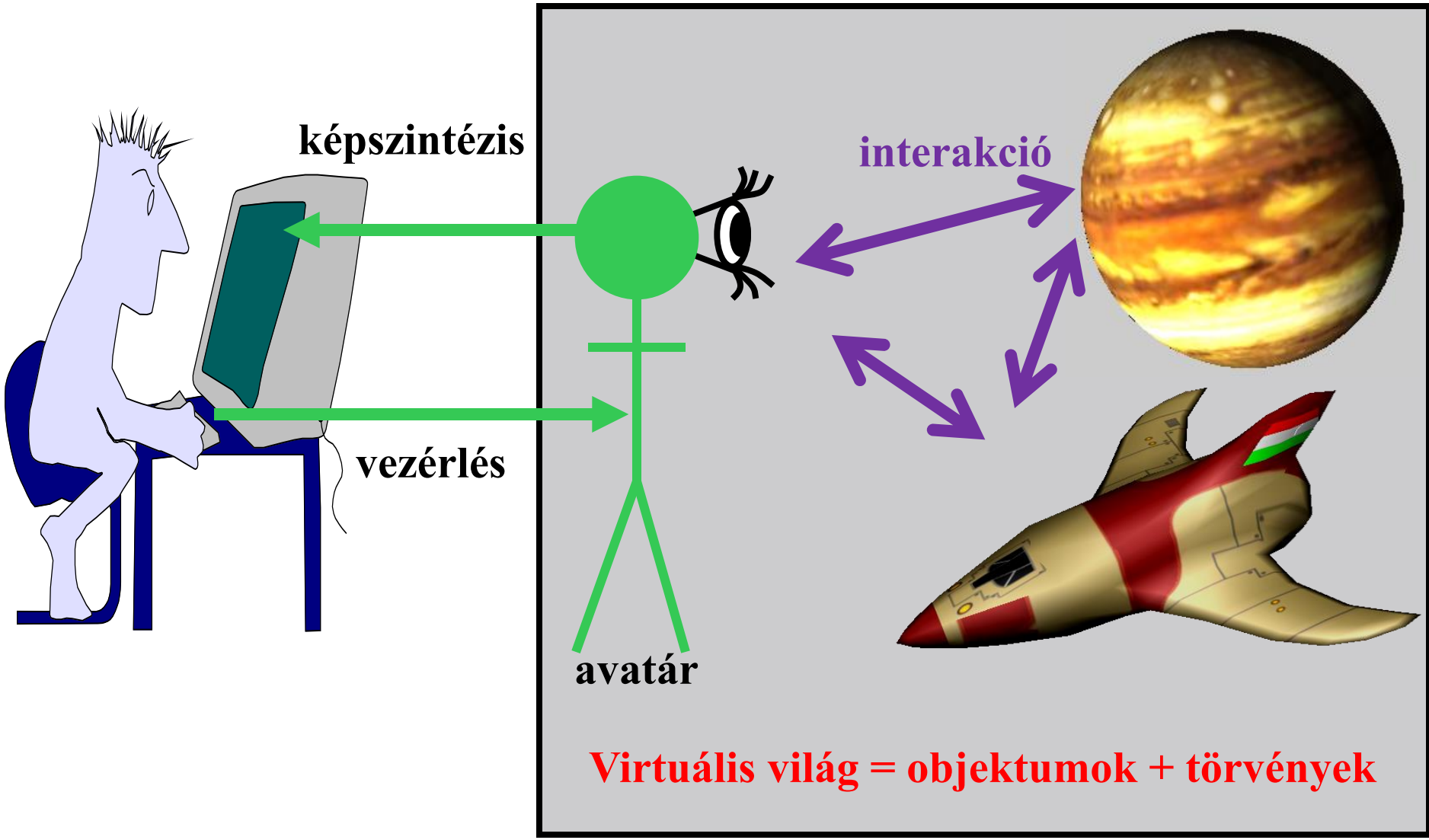
Játékfejlesztés

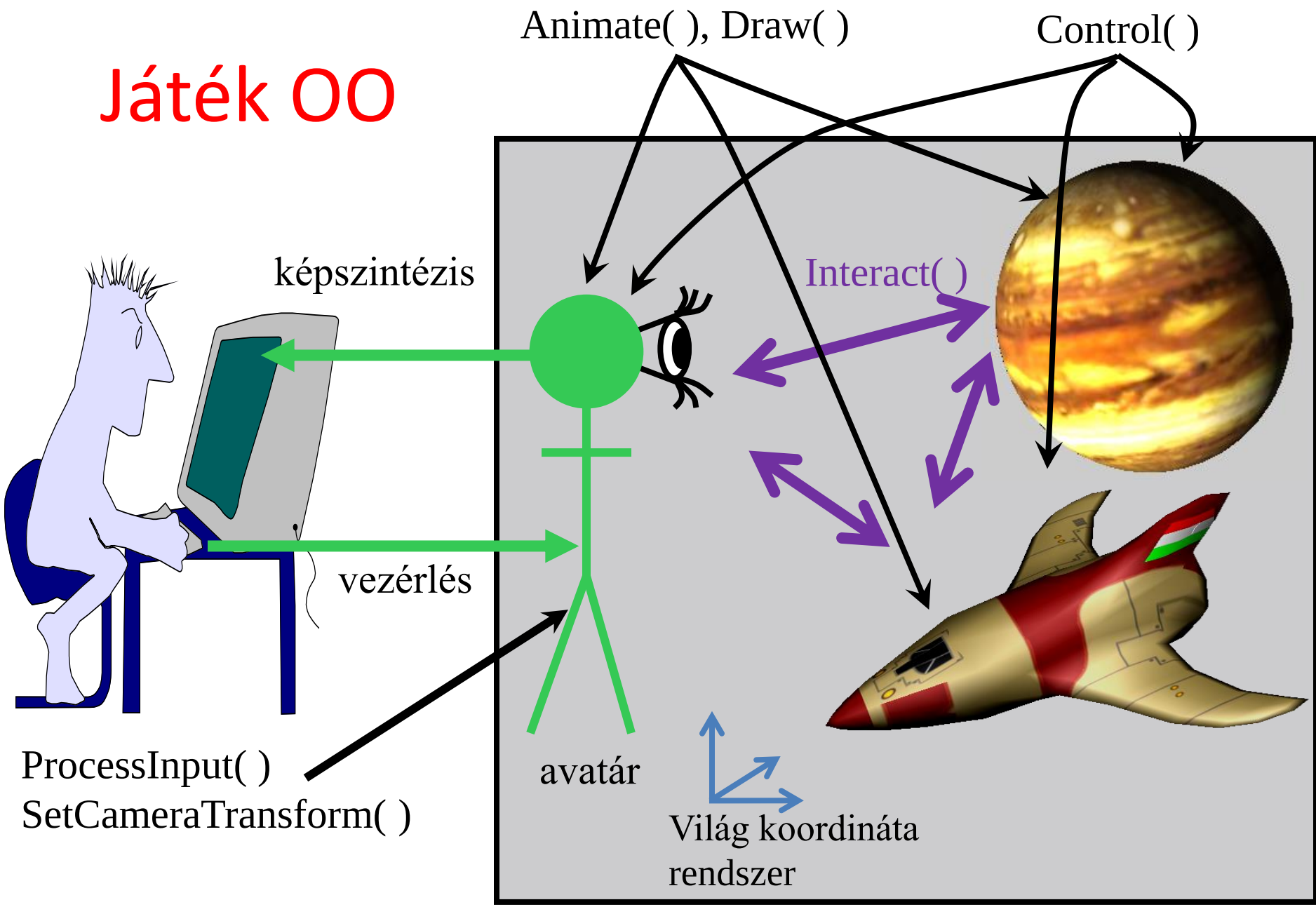


Szirmay-Kalos László

Virtuális valóság



Játék OO

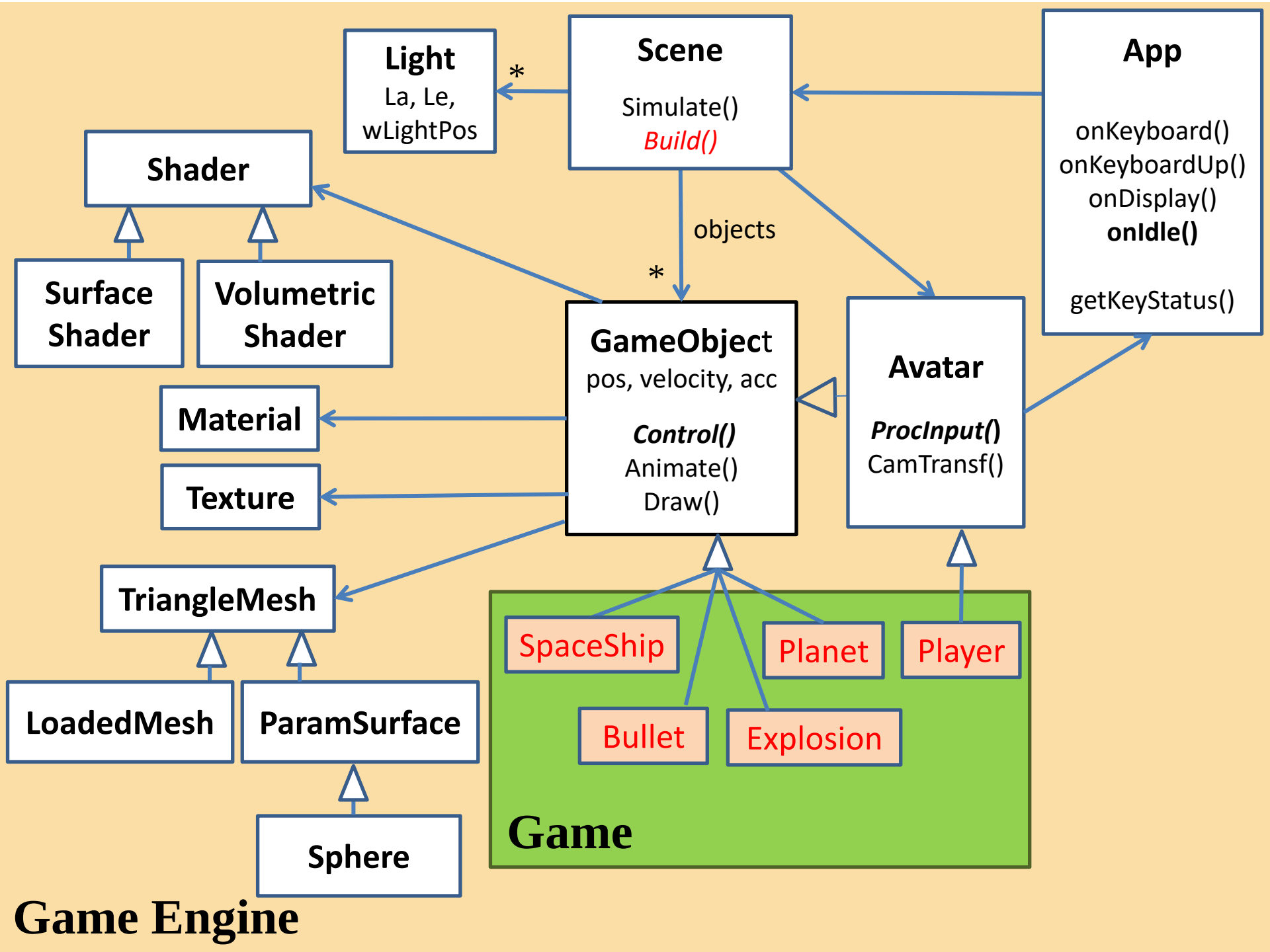


Scene: vector<GameObject *> objects;

GameObject

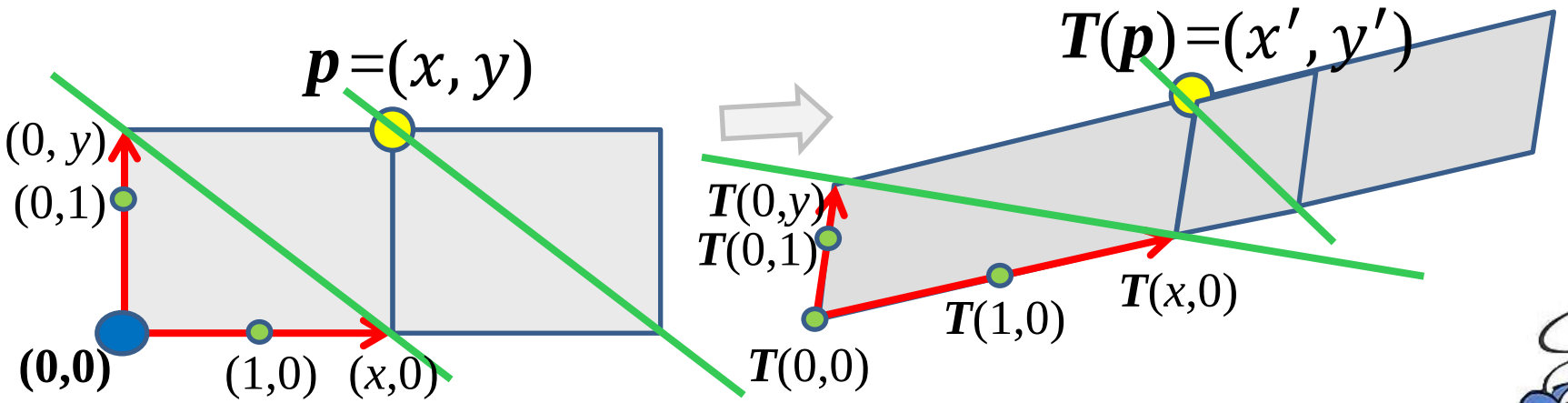
- Mely pontok tartoznak hozzá: egyenlet
  - Láthatóság: egyenlet megoldás
  - Transzformáció: egyenlet megváltoztatás

  Triangle mesh: háromszög háló
- A felületi pont hogyan változtatja meg a fényt
  - Shader: az interakció algoritmusa
  - Material: az algoritmus paraméterei
  - Textúra: a jellemzők változása az objektumon
- Műveletek:
  - Control: játékfüggő
  - Animate, Draw: általánosan is megoldható


Light
La, Le,
wLightPos
*
Scene
Simulate()
Build()
App
onKeyboard()
onKeyboardUp()
onDisplay()
onIdle()
getKeyStatus()
Shader
objects
*
Surface Shader
Volumetric Shader
GameObject
pos, velocity, acc
Control()
Animate()
Draw()
Avatar
ProcInput()
CamTransf()
Material
Texture
TriangleMesh
LoadedMesh
ParamSurface
Sphere
SpaceShip
Planet
Player
Bullet
Explosion
Game
Game Engine

Animate: Affin transzformáció

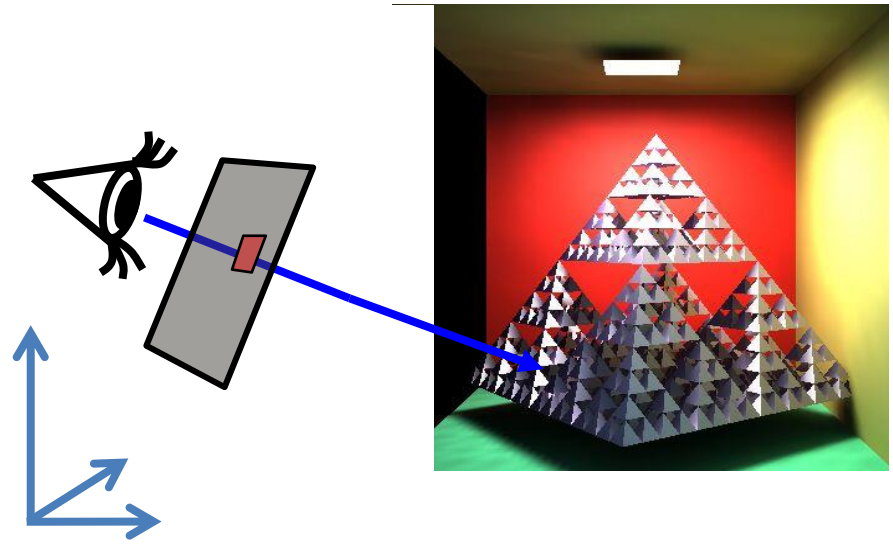
- Eltolás, elforgatás, nyújtás, nyírás, tükrözés, …
- Egyenest-egyenesbe, párhuzamosokat megtartja
  - Háromszögeket háromszögekbe képez le



$$\boldsymbol{T}(\boldsymbol{p}) = \boldsymbol{T}(0,0)+(\boldsymbol{T}(x,0)-\boldsymbol{T}(0,0))+(\boldsymbol{T}(0,y)-\boldsymbol{T}(0,0))$$
$$= \boldsymbol{T}(0,0)+x(\boldsymbol{T}(1,0)-\boldsymbol{T}(0,0))+y(\boldsymbol{T}(0,1)-\boldsymbol{T}(0,0))$$
$$= 1\,\boldsymbol{o}' \quad + \quad x\boldsymbol{i}' \quad + \quad y\boldsymbol{j}'$$

$$[x', y', 1] = [x, y, 1]\begin{bmatrix} \boldsymbol{i}'_x & \boldsymbol{i}'_y & 0 \\ \boldsymbol{j}'_x & \boldsymbol{j}'_y & 0 \\ \boldsymbol{o}'_x & \boldsymbol{o}'_y & 1 \end{bmatrix}$$

Draw: Hol látszhat/látszik



- Egy háromszög mely pixelekre vetül és azokban milyen messze van
- Minden pixelben a legközelebbit tartsuk meg.

Depth buffer

VideoMach unregistered

Boeing 777
337M triangles

Intel Core i7-2600
1024 x 768
3 pixels of error
Primary and shadow rays

VideoMach unregistered

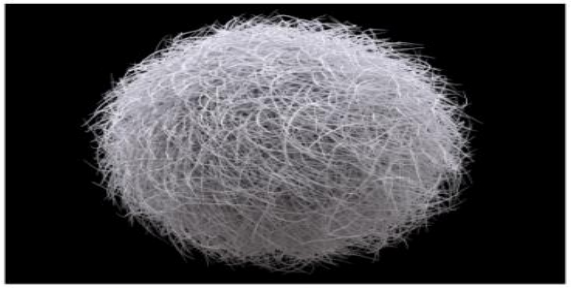
Mandelbulb model
354M triangles

Intel Core i7-620M
640 x 480
3 pixels of error
Primary rays

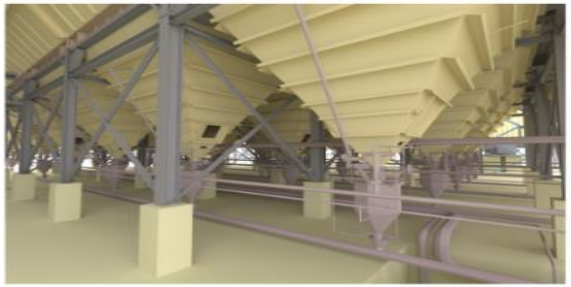
(a) CONFERENCE (282K triangles)

(b) CRYTEK SPONZA (262K triangles)

(c) FAIRY (174K triangles)



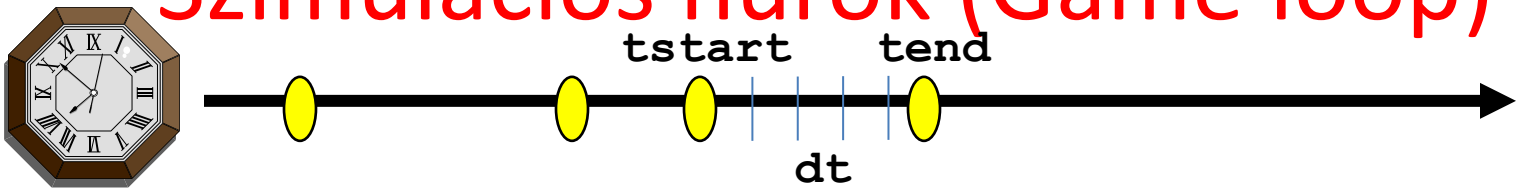
(d) HAIRBALL (2.9M triangles)

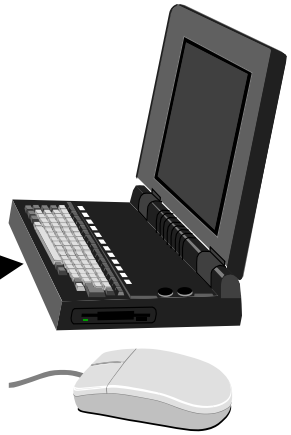


(e) POWER PLANT (12.7M triangles)

(f) SAN MIGUEL (10.5M triangles)

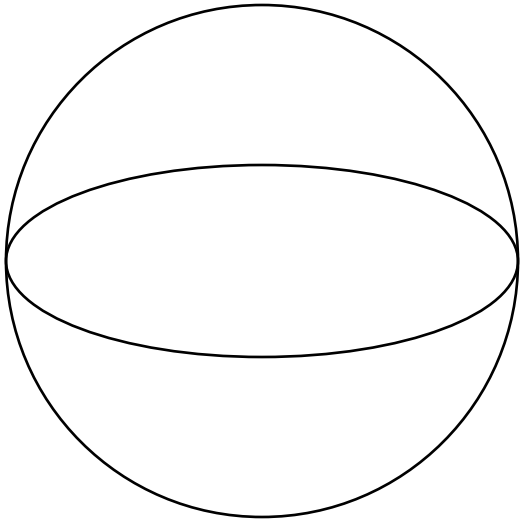
Szimulációs hurok (Game loop)



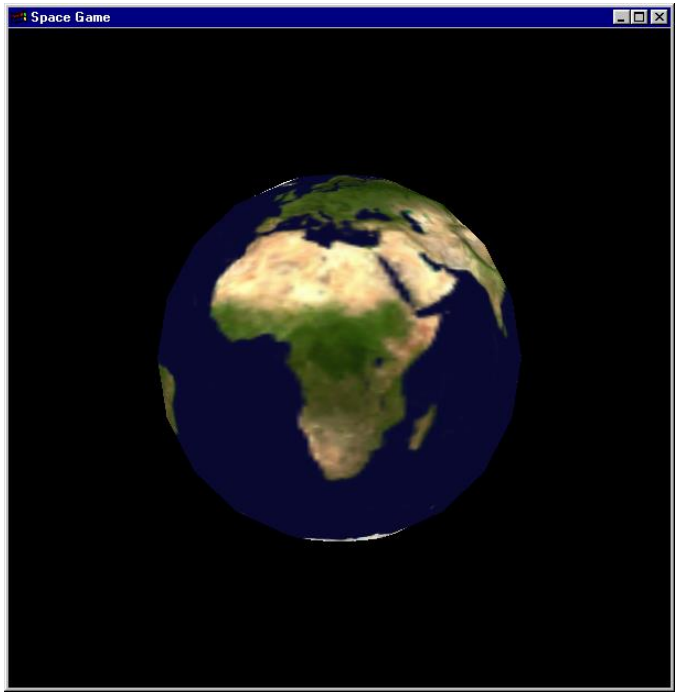


```
void Scene::Simulate ( ) { // idle call back
    static float tend = 0;
    float tstart = tend;
    tend = glutGet(GLUT_ELAPSED_TIME);
    avatar->ProcessInput( );

    // Simulation
    for(float t = tstart; t < tend; t += dt) {
        float Dt = min(dt, tend - t);
        for (GameObject * obj: objects) obj->Control(t, t+Dt);
        for (GameObject * obj: objects) obj->Animate(t, t+Dt);
    }

    // Rendering
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    avatar->SetCameraTransform(state);
    for (GameObject * obj: objects) obj->Draw(state);
    glutSwapBuffers( );
}
```

Bolygó: Planet

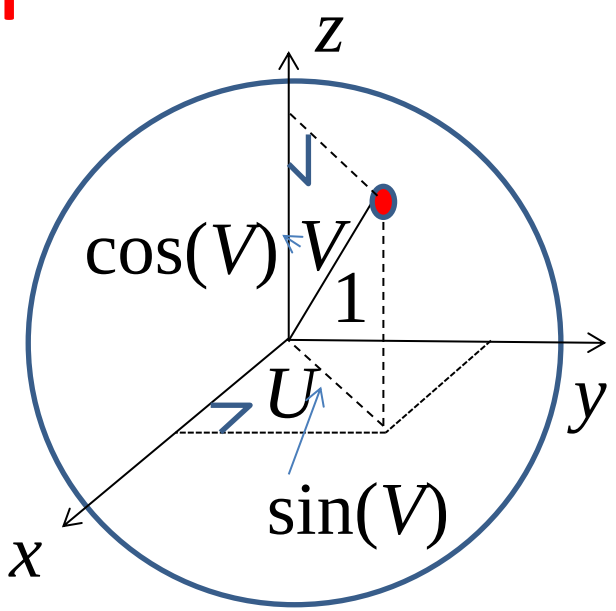


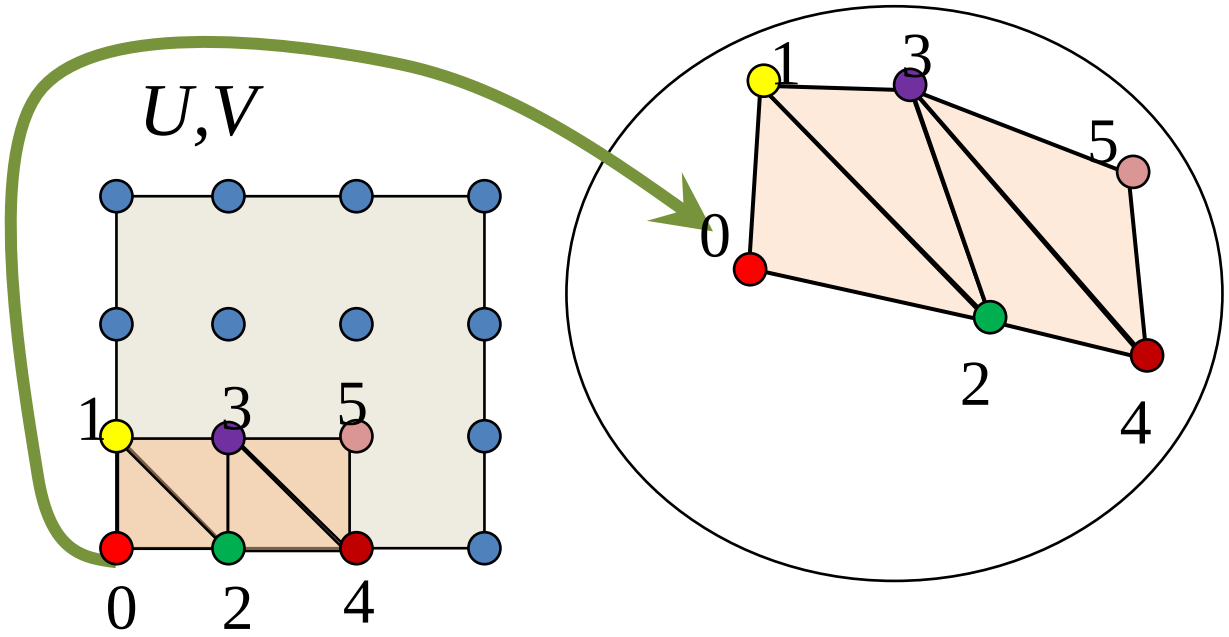
- Geometria: gömb
- Textúra: 2D térkép
- <u>Képletanimáció:</u>
  - „beégetett pálya”
  - Többiek érdektelenek
  - Nincs respektált törvény



Gömb: Triangle mesh

$$x(U,V) = c_x + r\cos(U)\sin(V)$$
$$y(U,V) = c_y + r\sin(U)\sin(V)$$
$$z(U,V) = c_z + r\cos(V)$$
$$U \in [0,2\pi], \quad V \in [0,\pi]$$





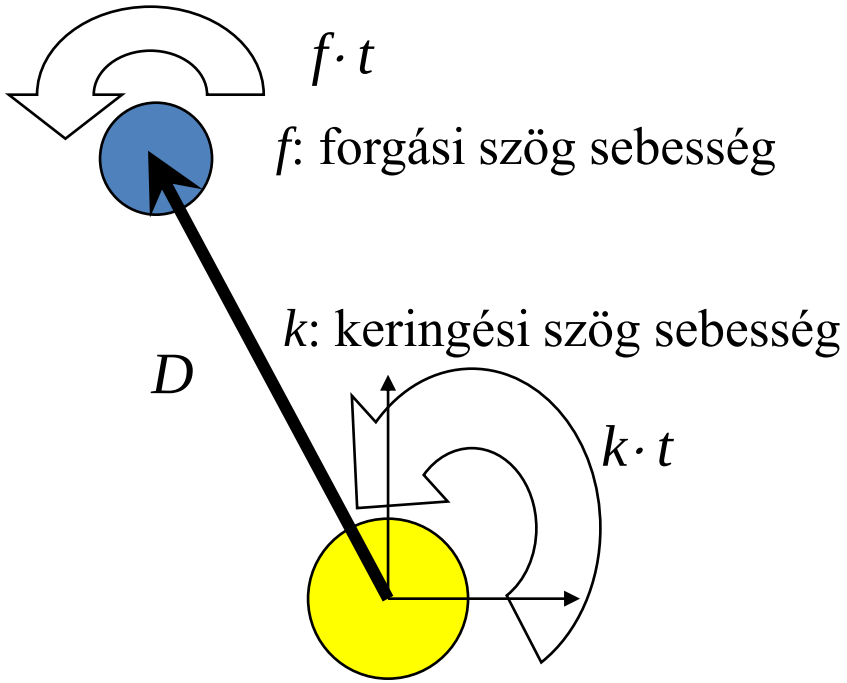
Pl. BME:

$V = 47{,}5^o$

$U = 19{,}0^o$

Animate:
A Föld forog és kering a Nap körül

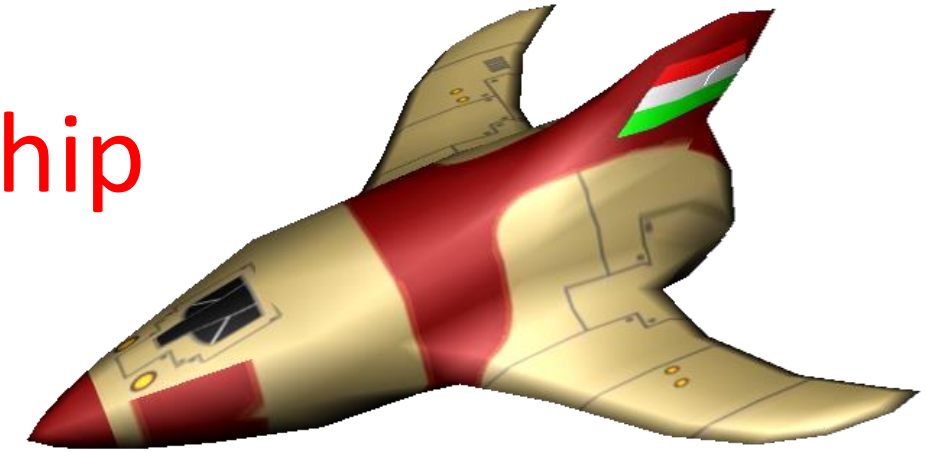
$$\mathrm{M} = \begin{bmatrix} \cos(f \cdot t) & \sin(f \cdot t) & 0 & 0 \\ -\sin(f \cdot t) & \cos(f \cdot t) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(23°) & \sin(23°) & 0 \\ 0 & -\sin(23°) & \cos(23°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

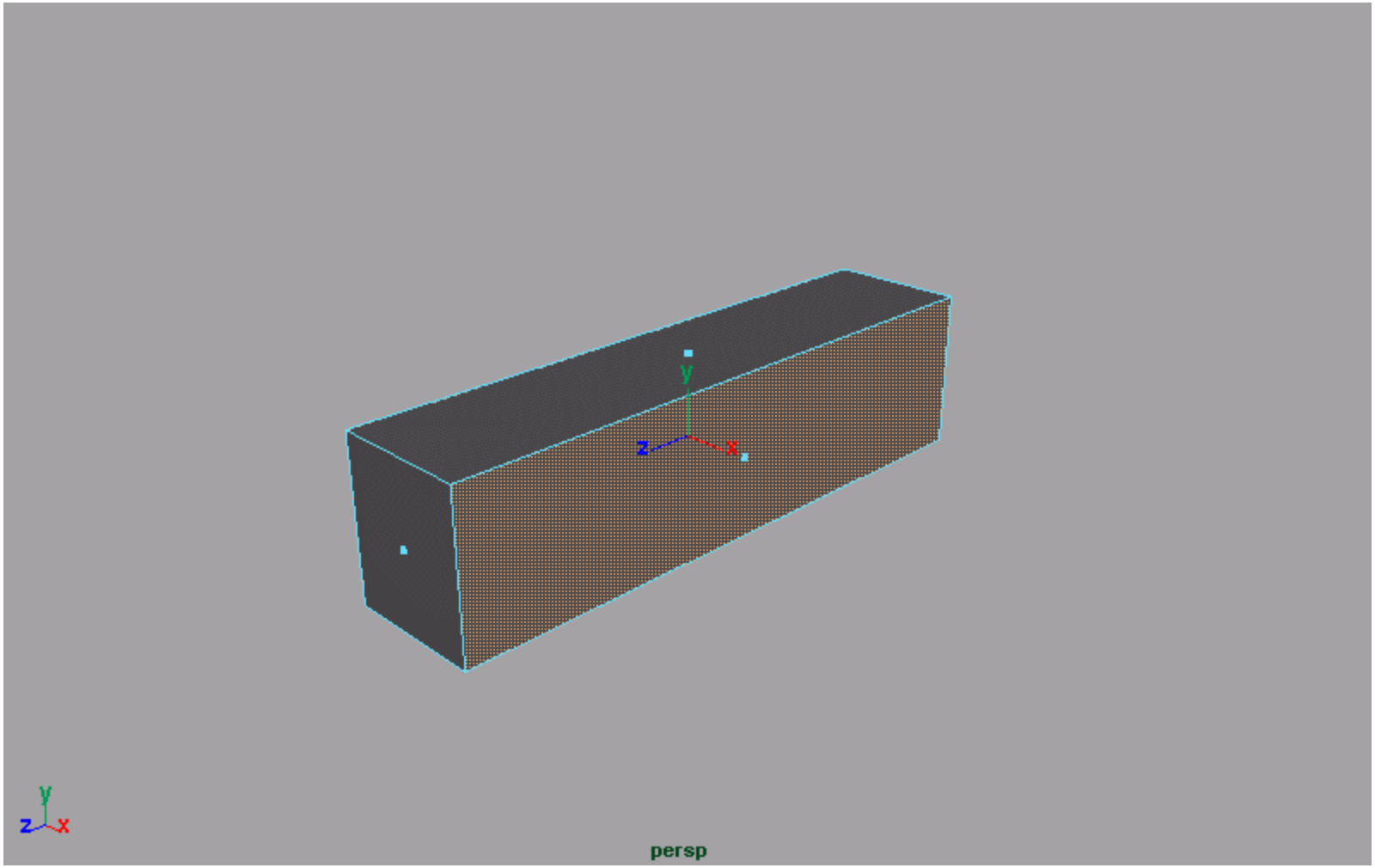
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ D\cos(k \cdot t) & D\sin(k \cdot t) & 0 & 1 \end{bmatrix}$$

Az űrhajó: SpaceShip

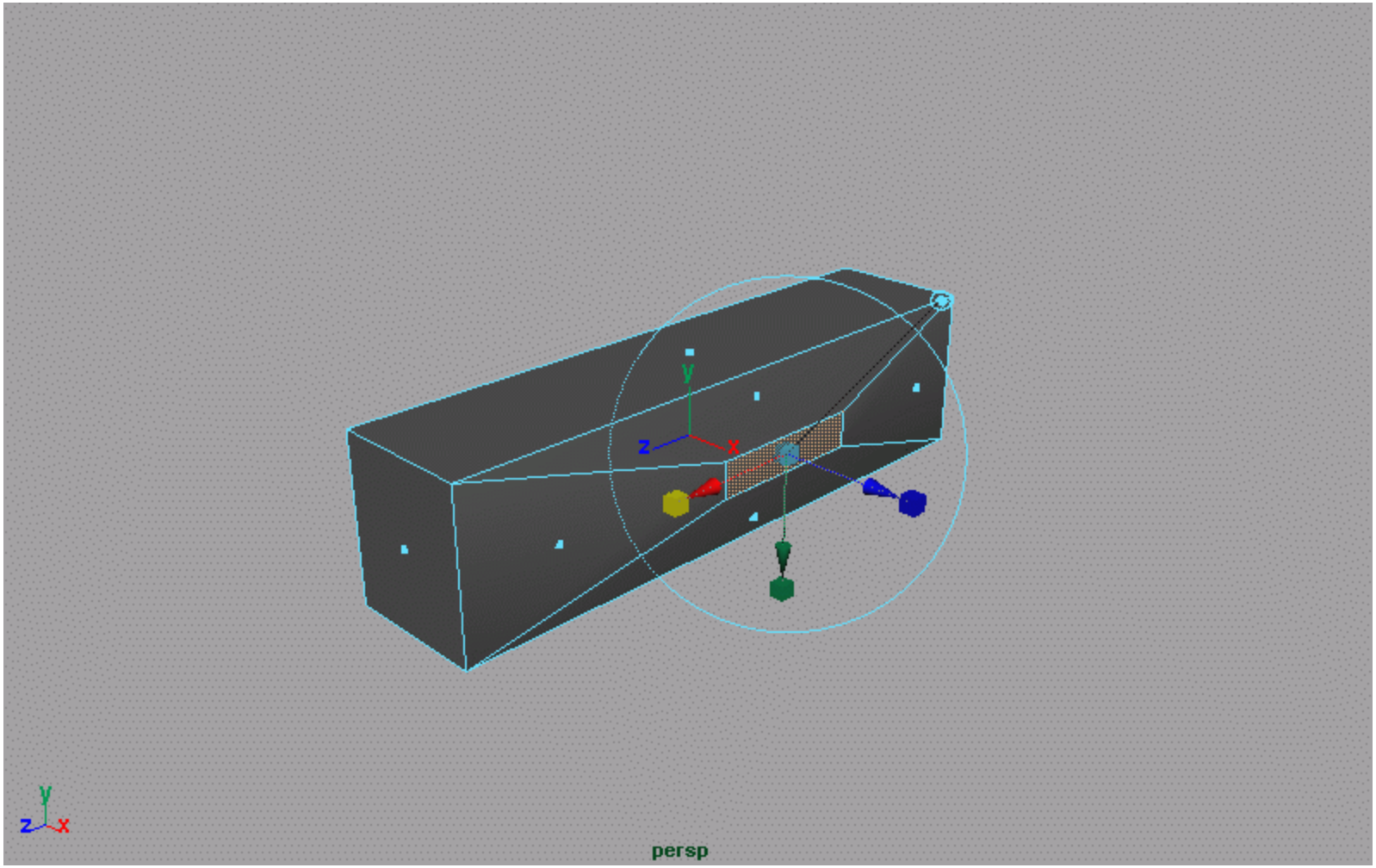


- Komplex geometria
  - négyszögháló
- Komplex textúra
- Fizikai animáció
  - erők (gravitáció, rakéták)
  - ütközések
- Viselkedés (AI)
  - A rakéták vezérlése
    - Ütközés elkerülés, avatártól menekülés, avatár üldözése

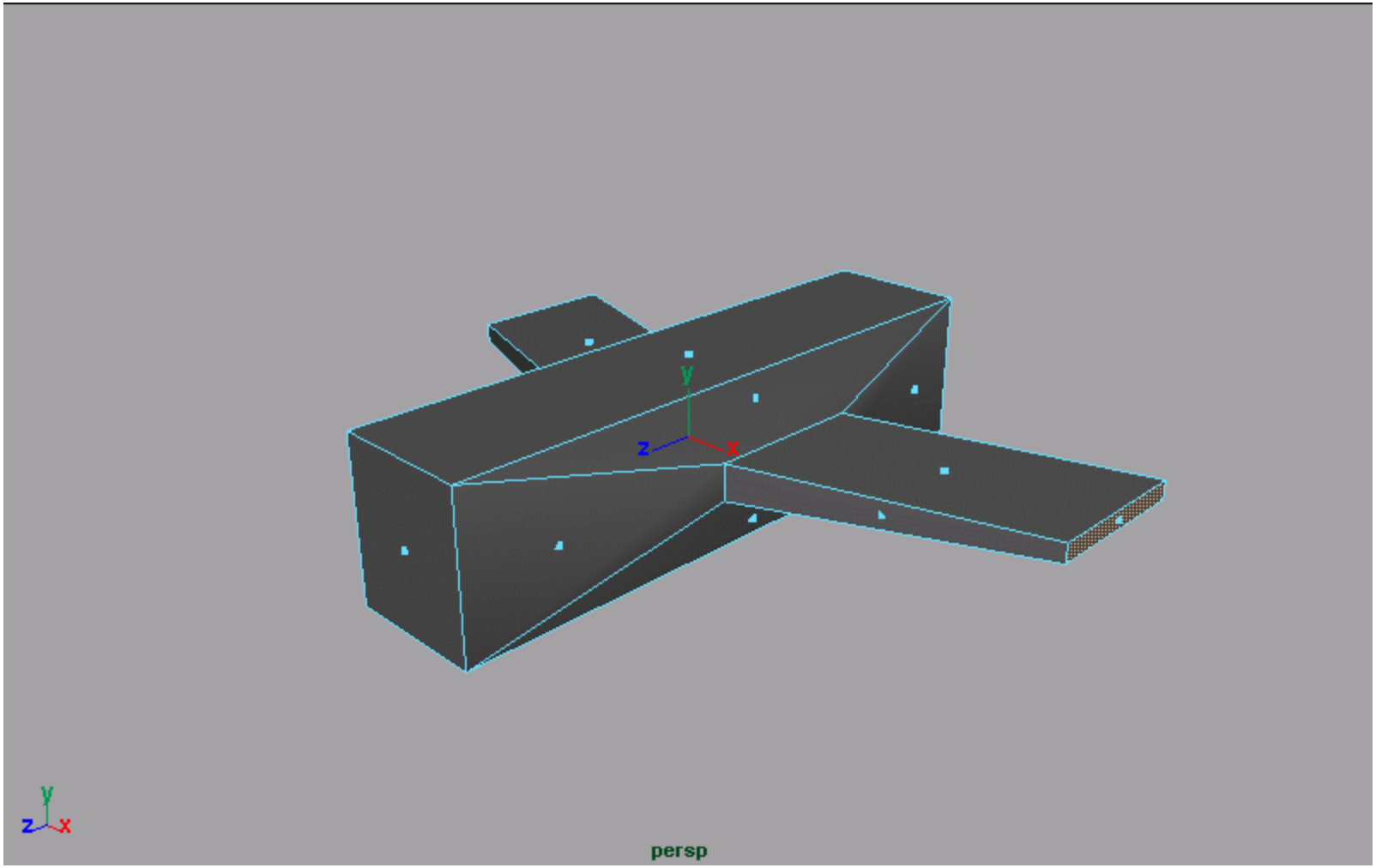
Űrhajó: Quad mesh



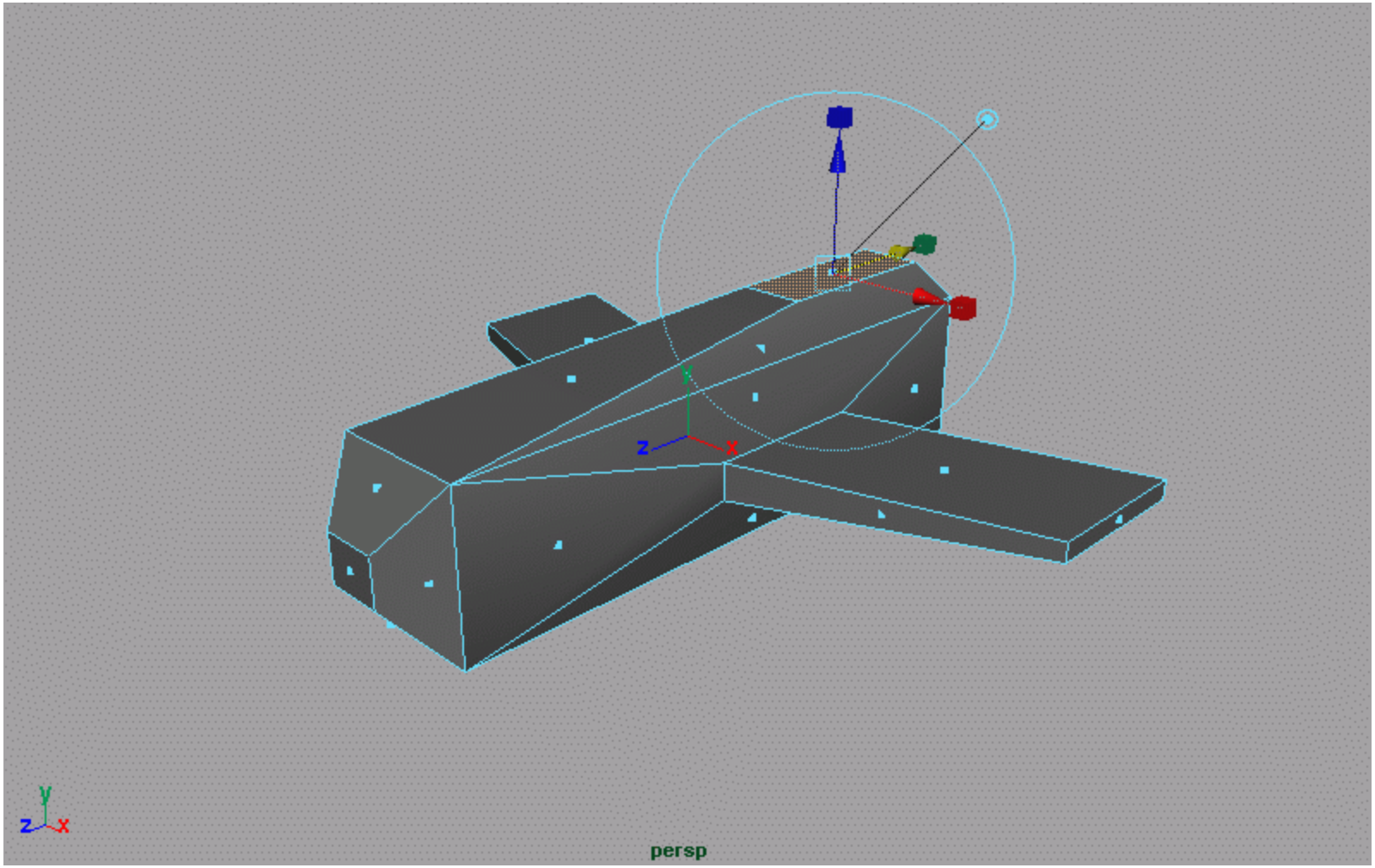
Űrhajó geometria
Euler műveletek: csúcs + lap = él + 2



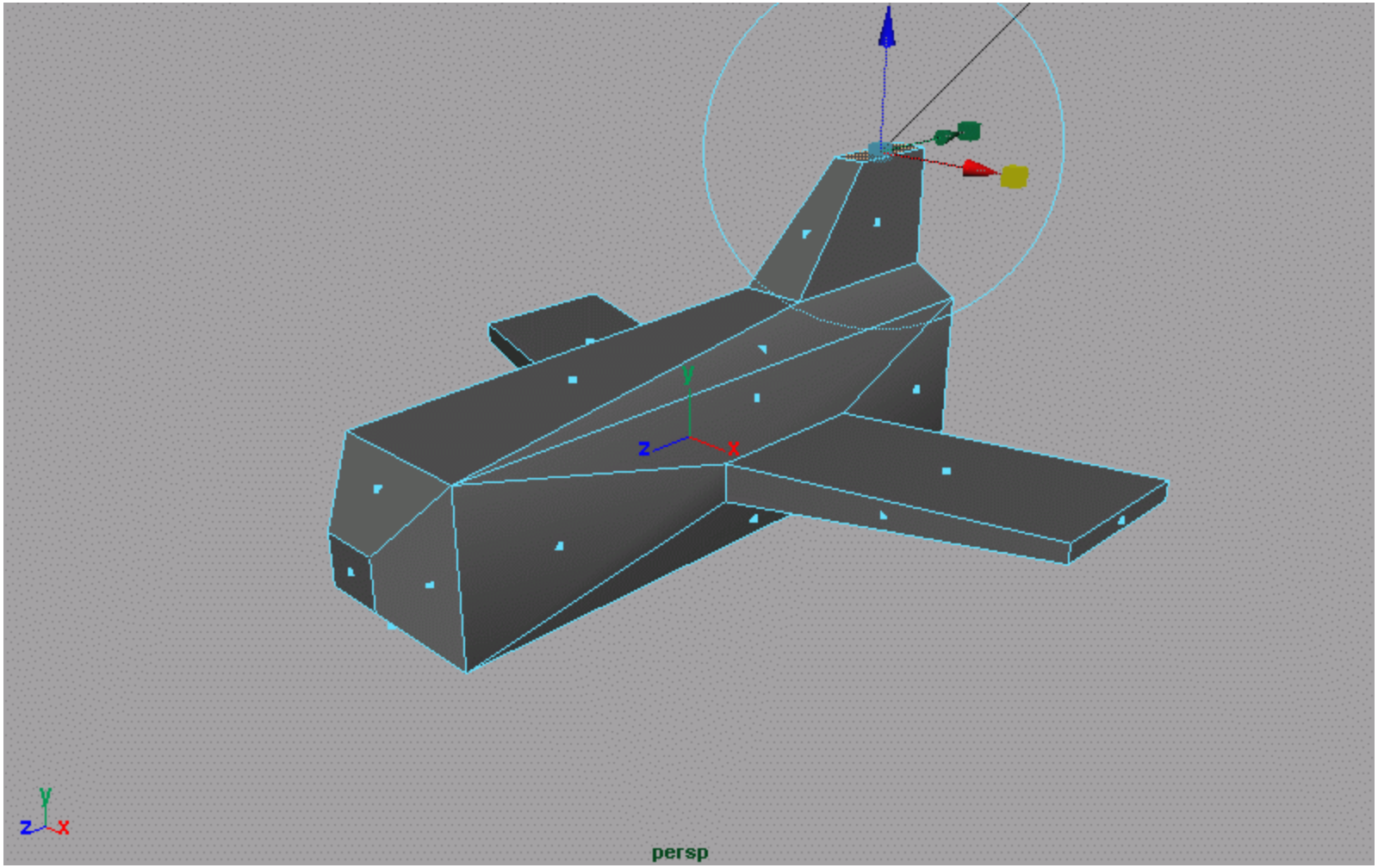
Űrhajó geometria



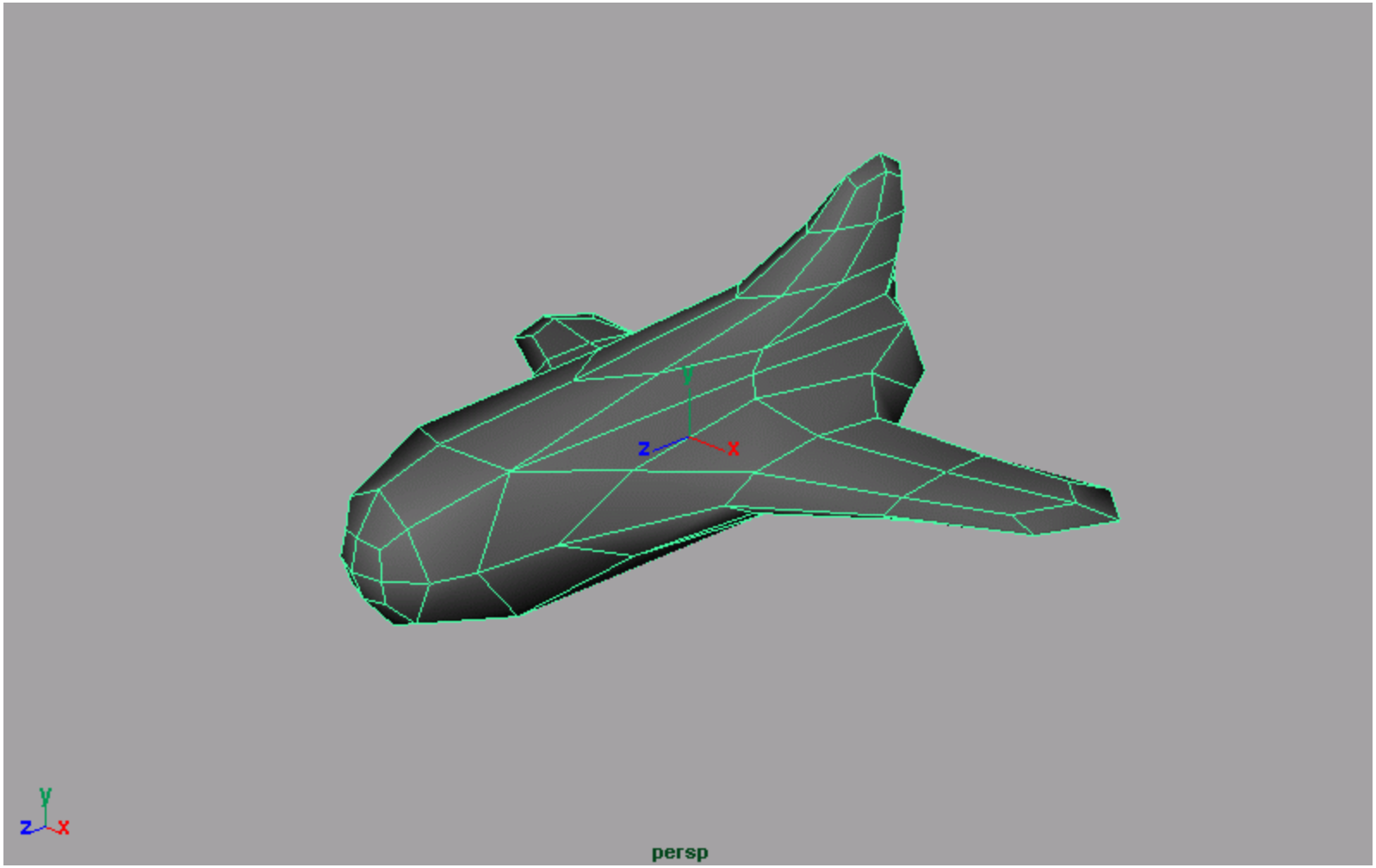
Űrhajó geometria



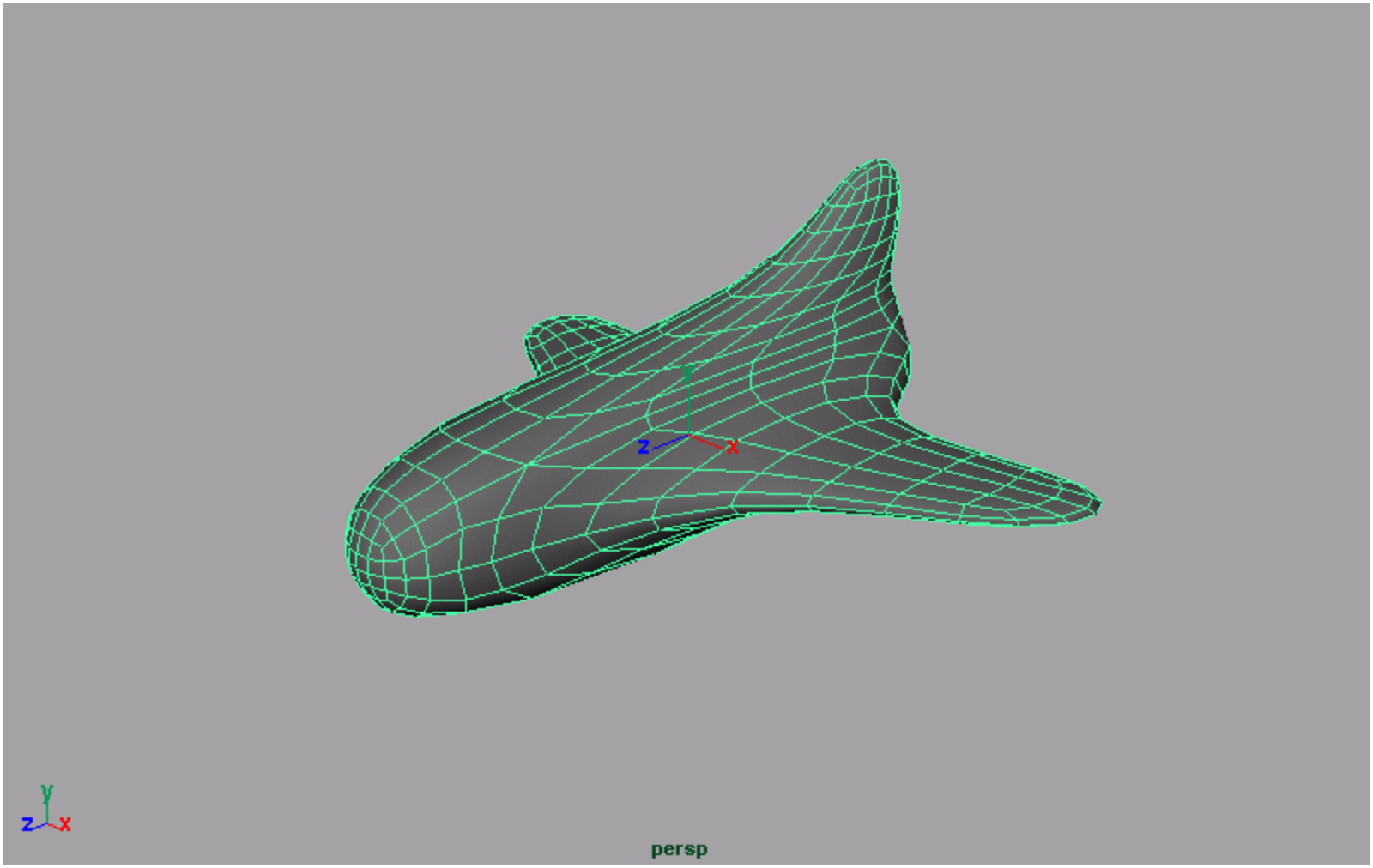
Űrhajó geometria



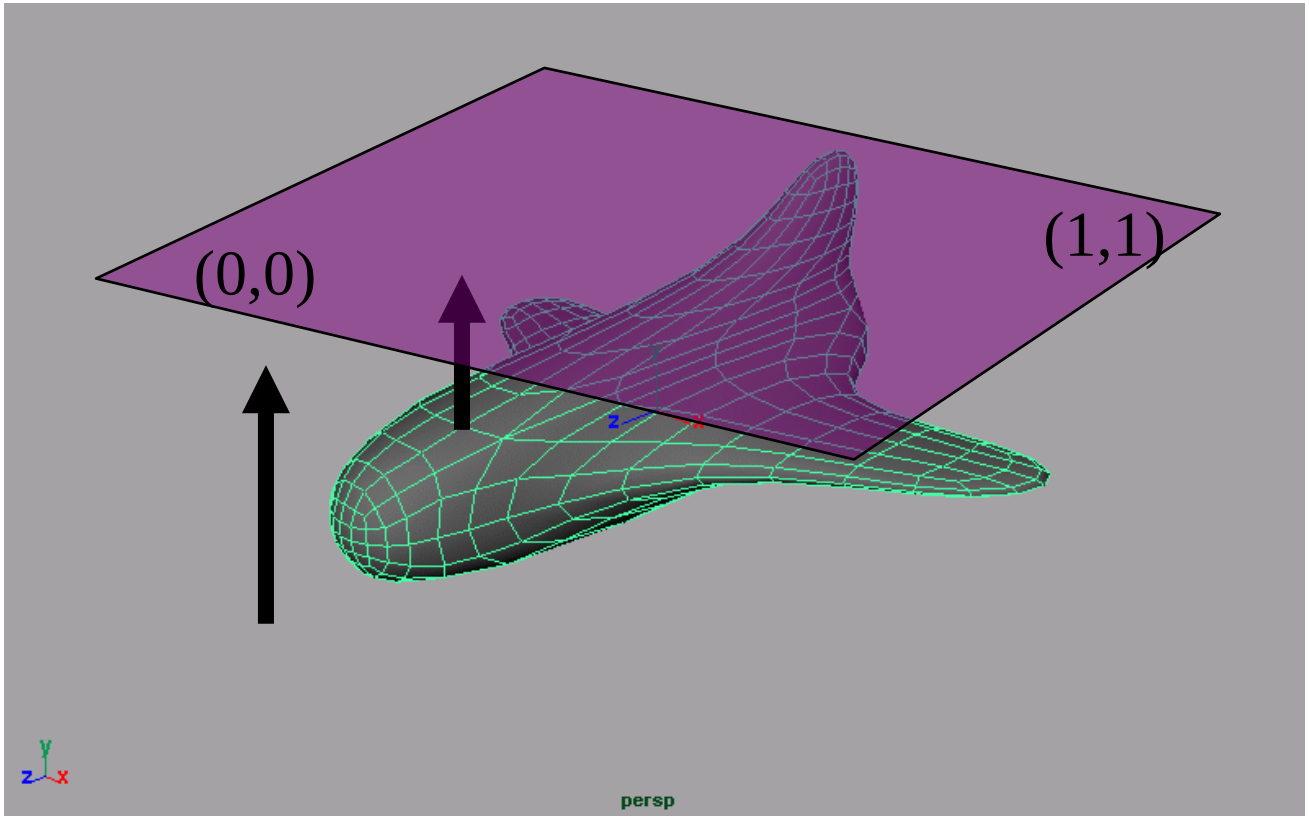
Űrhajó geometria: Subdivision surface

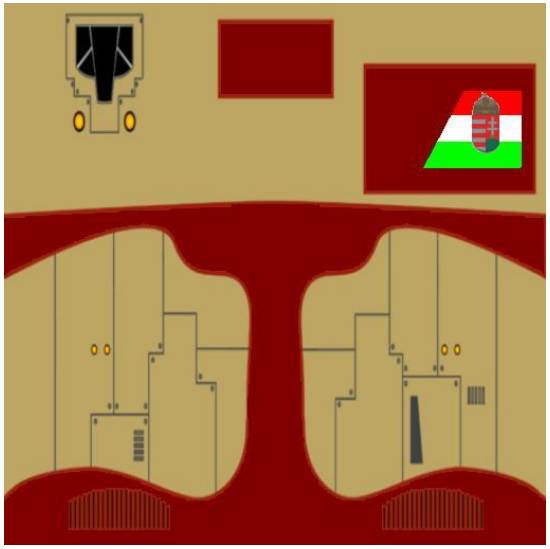


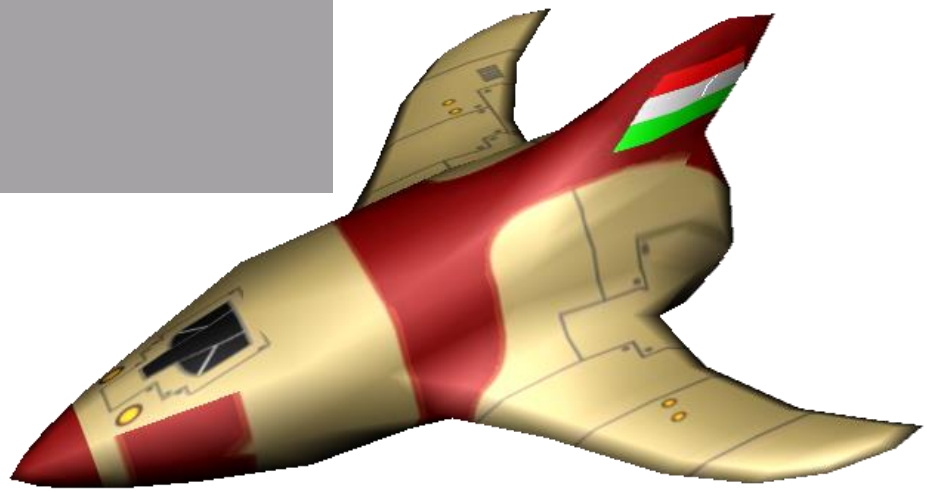
Űrhajó geometria: Subdivision surface



Textúrához paraméterezés



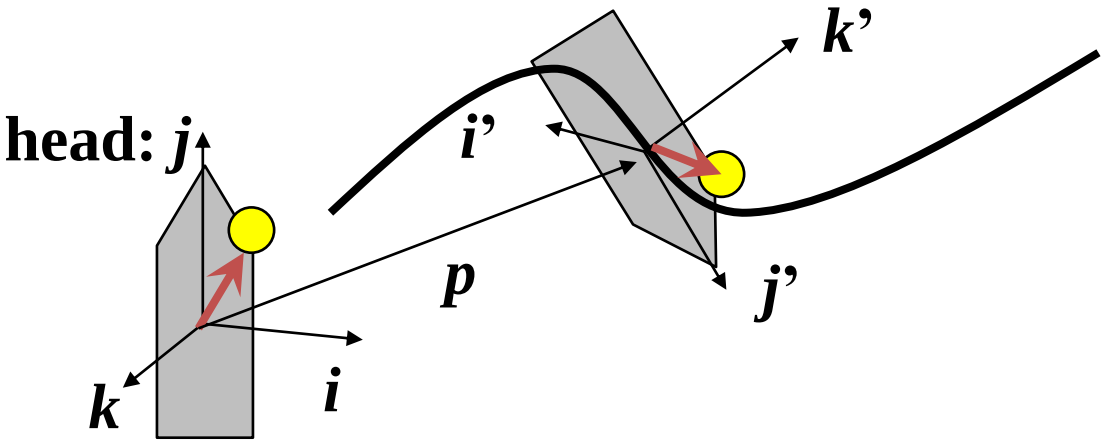




(AliasWavefront) OBJ fájlformátum

```
v -0.708698 -0.679666 2.277417
v  0.708698 -0.679666 2.277417
v -0.735419  0.754681 2.256846
...
vt 0.510655 0.078673
vt 0.509594 0.070000
vt 0.496429 0.079059
...
vn -0.843091  0.000000 0.537771
vn -0.670151 -0.543088 0.505918
vn -0.000000 -0.783747 0.621081
…
f 65/1/1 37/2/2   62/3/3   61/4/4
f 70/8/5 45/217/6 67/218/7 66/241/8
f 75/9/9 57/10/10 72/11/11 71/12/12
...
```

Animate: Newton 2 + Frenet frame



$$\mathbf{M} = \begin{pmatrix} \boldsymbol{i}' & 0 \\ \boldsymbol{j}' & 0 \\ \boldsymbol{k}' & 0 \\ \boldsymbol{p} & 1 \end{pmatrix}$$

$$\boldsymbol{F} = m\boldsymbol{a}, \qquad \boldsymbol{a} = \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t}, \qquad \boldsymbol{v} = \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t}$$

$$\boldsymbol{a} = \frac{\boldsymbol{F}}{m}, \qquad \boldsymbol{v} \mathrel{+}= \boldsymbol{a}\mathrm{d}t, \qquad \boldsymbol{p} \mathrel{+}= \boldsymbol{v}\mathrm{d}t$$

Orientáció, nem ortonormált:

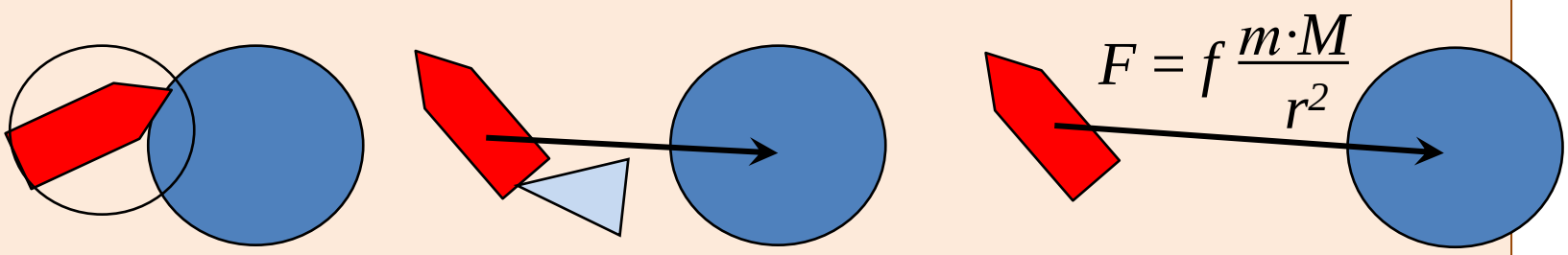
$$\boldsymbol{j}' = \boldsymbol{v},$$
$$\boldsymbol{k}^* = \boldsymbol{k}'(1-\alpha) + \boldsymbol{a}\alpha,$$
$$\boldsymbol{i}' = \boldsymbol{j}' \times \boldsymbol{k}^*$$

Gram-Schmidt ortogonalizáció:

$$\boldsymbol{j}' = \boldsymbol{v}/|\boldsymbol{v}|,$$
$$\boldsymbol{i}' = \boldsymbol{j}' \times \boldsymbol{k}^*/|\boldsymbol{j}' \times \boldsymbol{k}^*|,$$
$$\boldsymbol{k}' = \boldsymbol{i}' \times \boldsymbol{j}'$$

Control: az erő számítása és AI

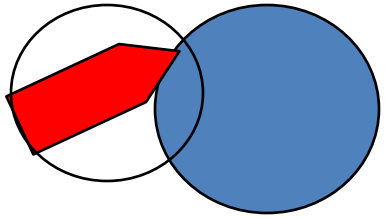
```
void Ship :: Control( float tstart, float tend ) {
    force = vec3(0, 0, 0);
    for (GameObject* obj: scene.objects) {
        if (dynamic_cast<Planet*>(obj)) {
```




```
        }
        if (dynamic_cast<Avatar*>(obj)) {
```

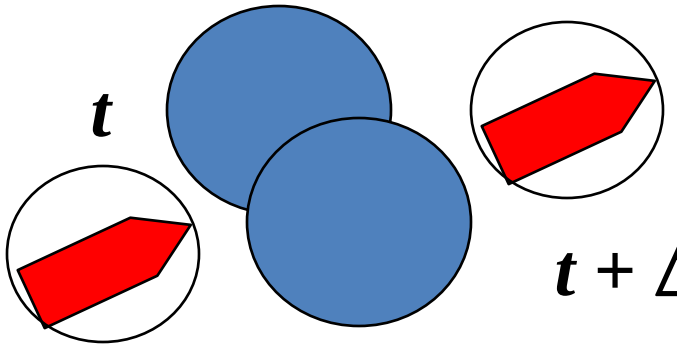

```
        }
    }
}
```

Ütközésdetektálás: lassú objektumok



adott *t*

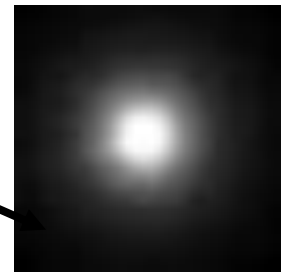
Probléma, ha az objektum gyors




```
dist = obj1.pos - obj2.pos
min = obj1.BoundingRadius() + obj2.BoundingRadius()
if (dist.Length() < min) Collision!
```

Foton torpedó: Bullet

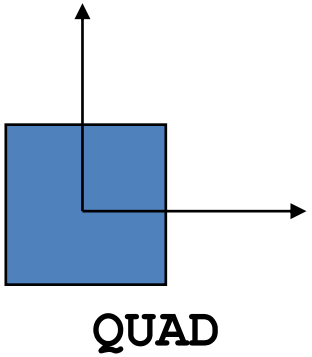
- Nagyon komplex geometria
- Hasonló kinézet minden irányból
- Könnyebb a képét használni

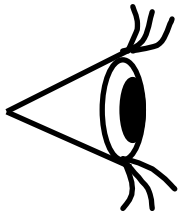


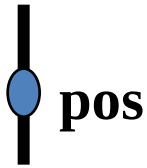
- Ütközésdetektálás = gyors mozgás

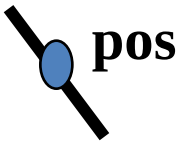
Billboard

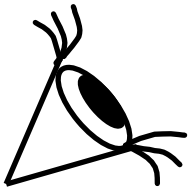
Egyetlen félig átlátszó textúra egy téglalapon









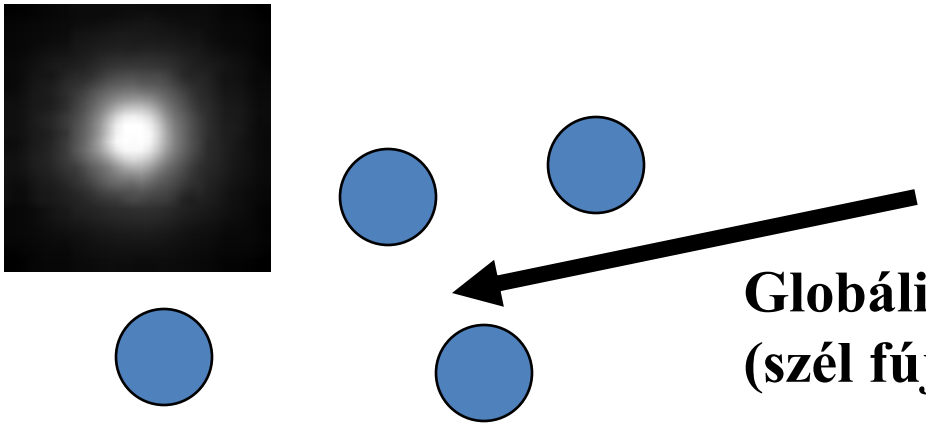


Robbanás

- Nagyon komplex geometria
- Hasonló kinézet minden irányból
- Plakátgyűjtemény

- Részecske rendszer

Részecske rendszerek

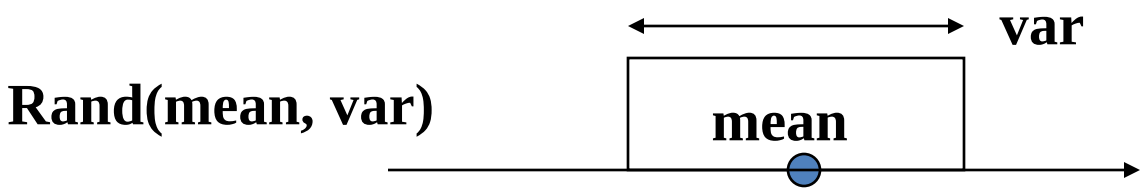


Globális erőtér
(szél fújja a füstöt)

Véletlen
Kezdeti
értékek

pos:	**pos += velocity * dt**
velocity:	**velocity += acceleration * dt**
acceleration:	**acceleration = force / weight**
lifetime	
age:	**age += dt; if (age > lifetime) Kill();**
size, dsize:	**size += dsize * dt;**
weight, dweight:	**weight += dweight * dt**
color, dcolor:	**color += dcolor * dt**

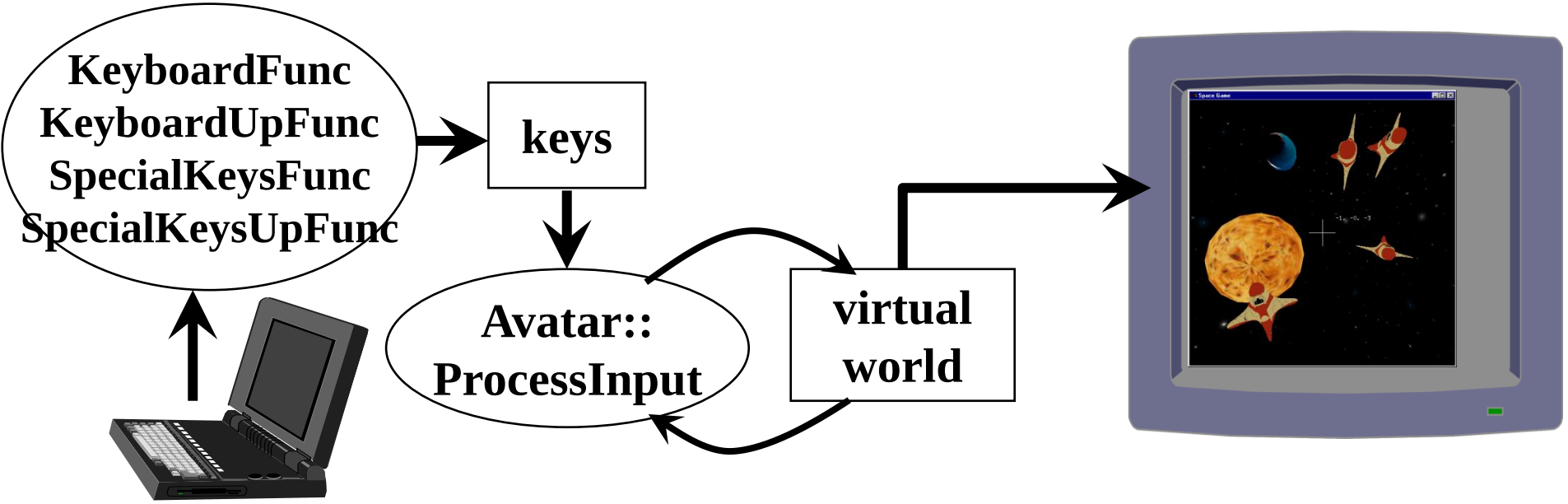
Robbanás paraméterei



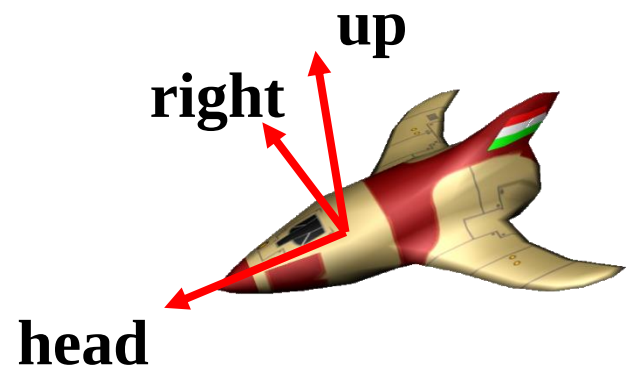

```
pos = center;                          // kezdetben fókuszált
lifetime = Rand(2, 1);

size = 0.001;                                  // kezdetben kicsi
dsize = Rand(0.5, 0.25) / lifetime;

velocity = Vector(Rand(0,0.4),Rand(0,0.4),Rand(0,0.4));
acceleration = Vector(Rand(0,1),Rand(0,1),Rand(0,1));

                       // Planck törvény: sárga átlátszatlanból vörös átlátszóba
color = Color(1, Rand(0.5, 0.25), 0, 1 );
dcolor = Color(0, -0.25, 0, -1) / lifetime;
```

Klaviatúra kezelés: Polling

```
bool keys[256];      // is pressed?
void onKeyboard(unsigned char key, int pX, int pY) {
   keys[key] = true;
}
void onKeyboardUp(unsigned char key, int pX, int pY) {
   keys[key] = false;
}
```



Avatar :: ProcessInput




```
Avatar :: ProcessInput() {
  if ( keys[' '] ) // fire!
    objects.push_back(new Bullet(pos, velocity)

      // Kormányzás: az avatár koordinátarendszerében!
  vec3 head = normalize(velocity);
  vec3 right = normalize(cross(wVup, head));
  vec3 up = cross(head, right);

  if (keys[KEY_UP])    force -= up;
  if (keys[KEY_DOWN])  force += up;
  if (keys[KEY_LEFT])  force -= right;
  if (keys[KEY_RIGHT]) force += right;
}
```