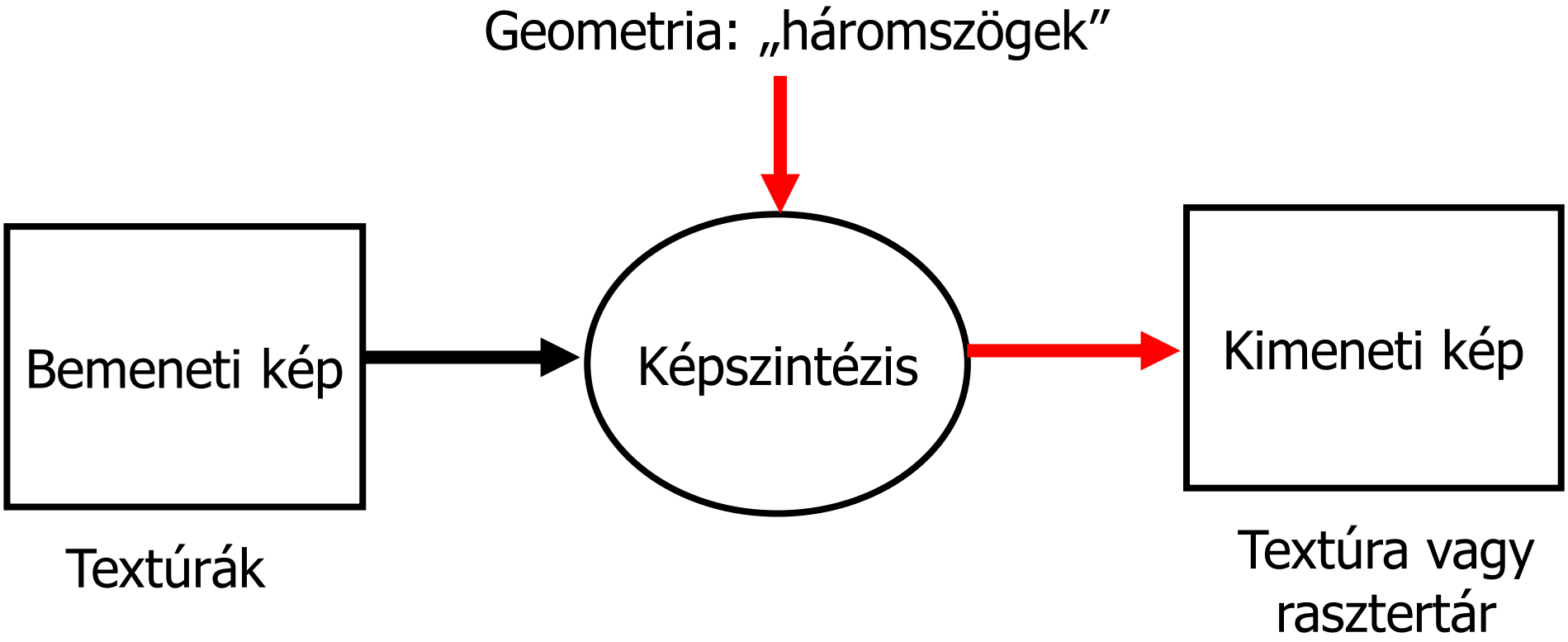
GPGPU

Szirmay-Kalos László

GPGPU Shader API-val



„Teljes képernyős" téglalap (CPU):

```
float cVtx[] = {-1,-1, 1,-1, 1,1, -1, 1};
glBufferData(GL_ARRAY_BUFFER, sizeof(cVtx),
             cVtx, GL_STATIC_DRAW);
glDrawArrays(GL_TRIANGLE_FAN, 0, 4);
```

Vertex shader

```
layout(location = 0) in vec2 cVertex;
out vec2 uv;

void main() {
   gl_Position = vec4(cVertex, 0, 1);
   uv = (cVertex + vec2(1, 1)) / 2;
}
```

Input data

Fragment shader

```
uniform sampler2D inputData;
in vec2 uv;
out vec4 result;

void main() {
   result = F(uv, bemAdat);
}
```

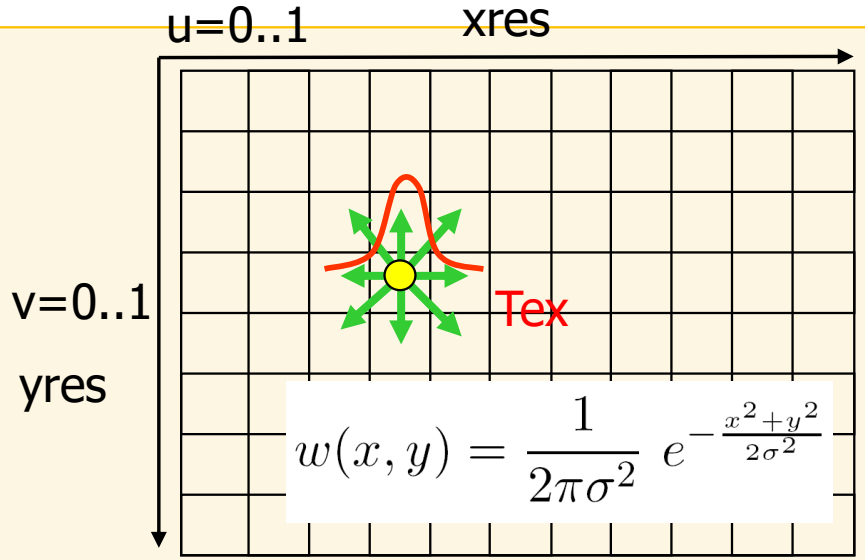
Melyik kimeneti tömbelemet számítjuk

Result array

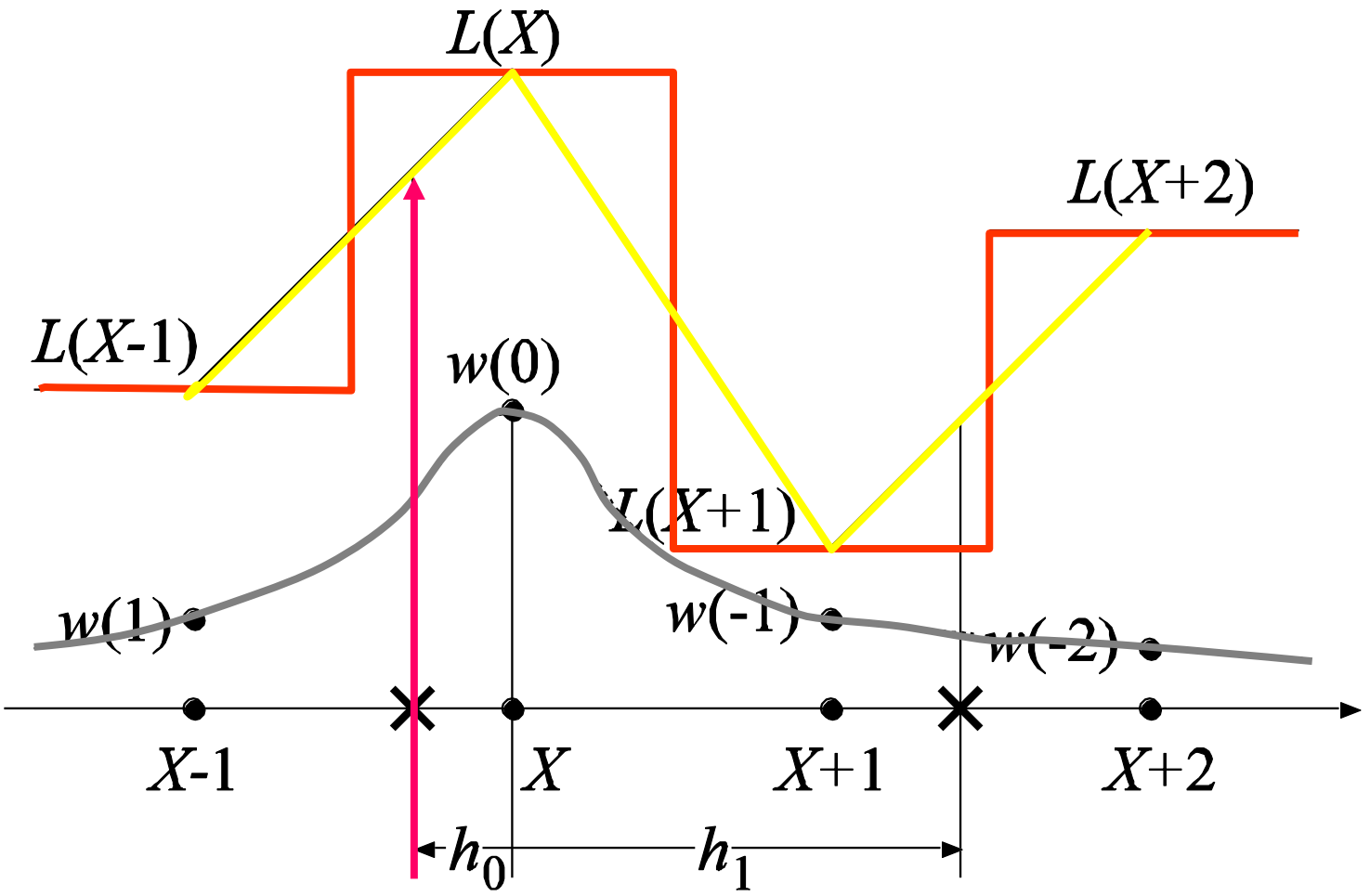
SIMD, i.e. vektorprocesszálás

Konvolúció képfeldolgozásban

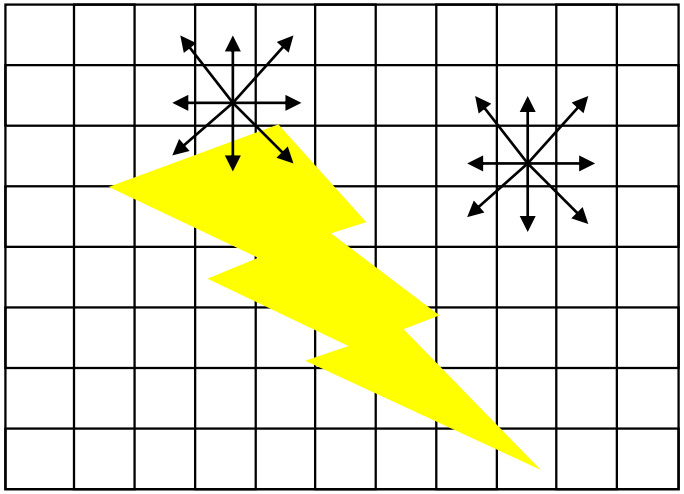
```
uniform sampler2D textureUnit;
uniform int halfFiltSize;
uniform float xres, yres;

in vec2 uv;
out vec4 outColor;

void main() {
   outColor = vec4(0, 0, 0, 0);
   for(int X = -halfFiltSize; X <= halfFiltSize; X++) {
      for(int Y = -halfFiltSize; Y <= halfFiltSize; Y++) {
         float w = FilterKernel(X, Y);
         vec2 duv = vec2(X/xres, Y/yres);
         outColor += texture(textureUnit, uv + duv) * w;
       }
   }
}
```

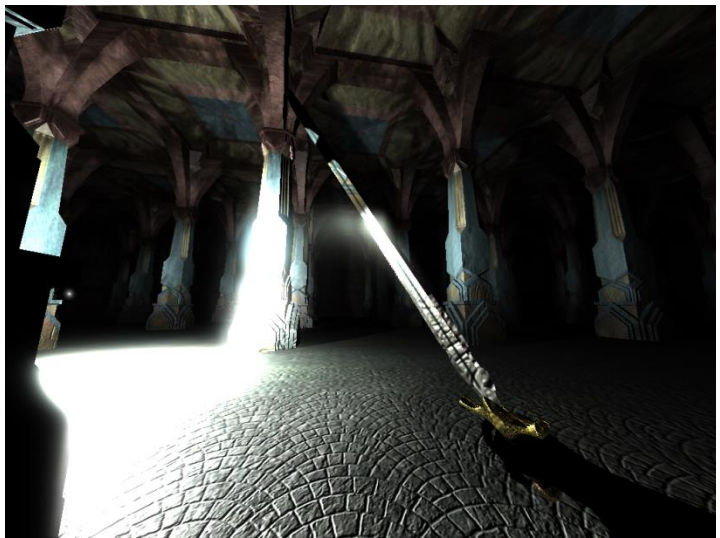


Bi-linear szűrés kiaknázása



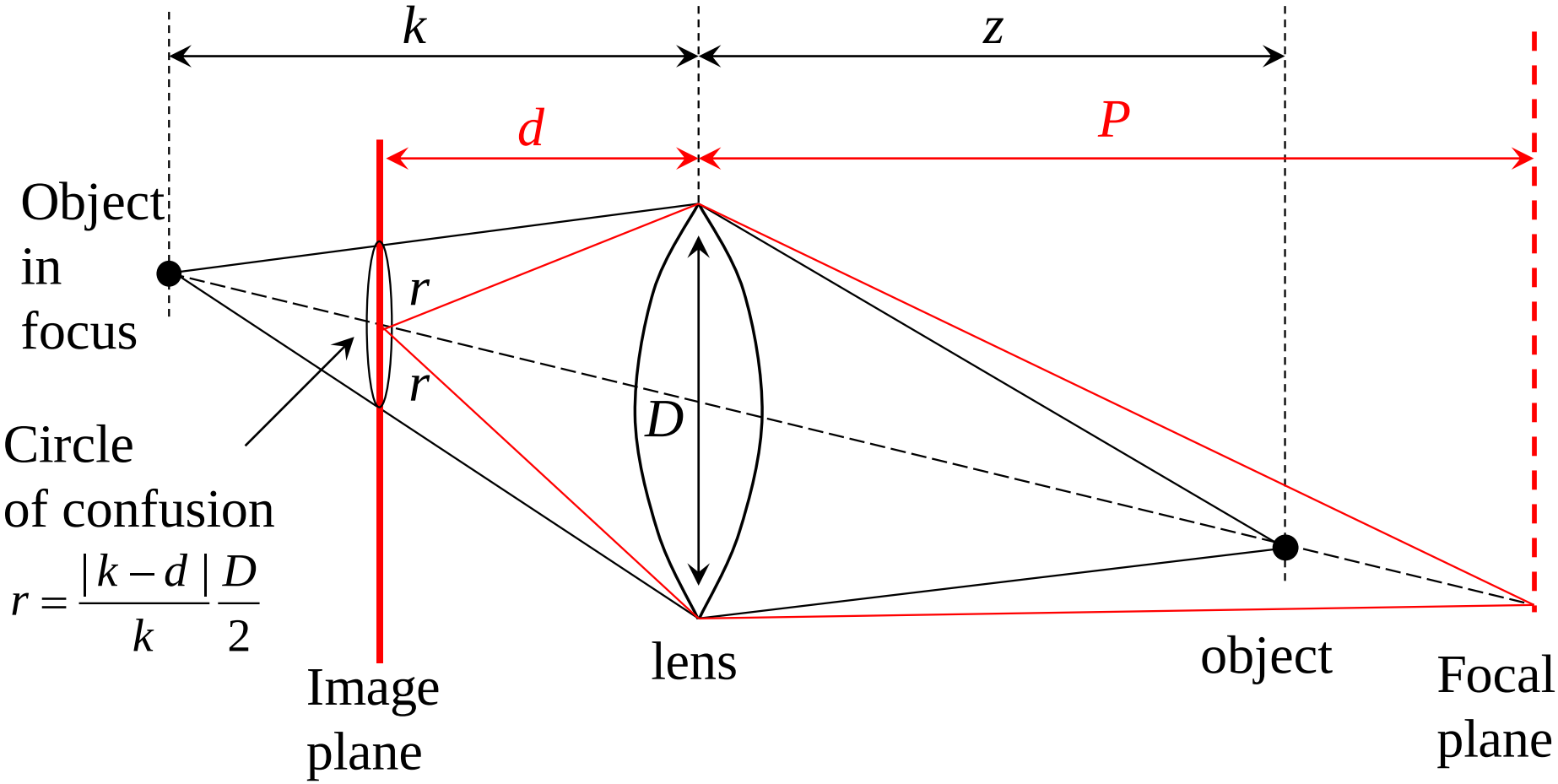
Glória
Glow/bloom





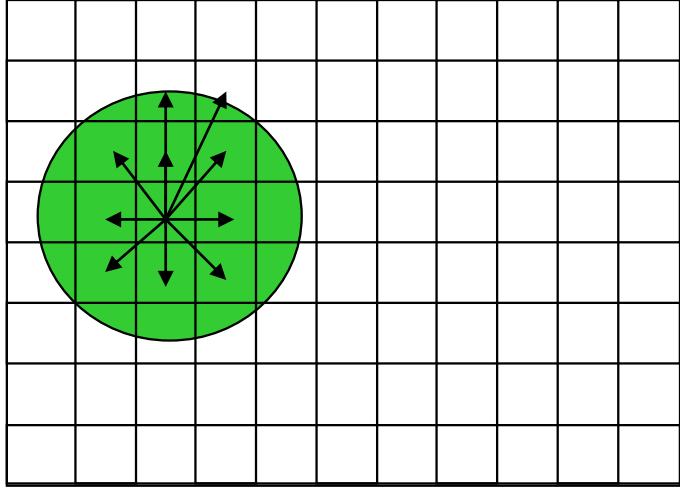
Mélységélesség
Depth of field

$$\frac{1}{f} = \frac{1}{z} + \frac{1}{k} \qquad \frac{1}{f} = \frac{1}{d} + \frac{1}{P}$$

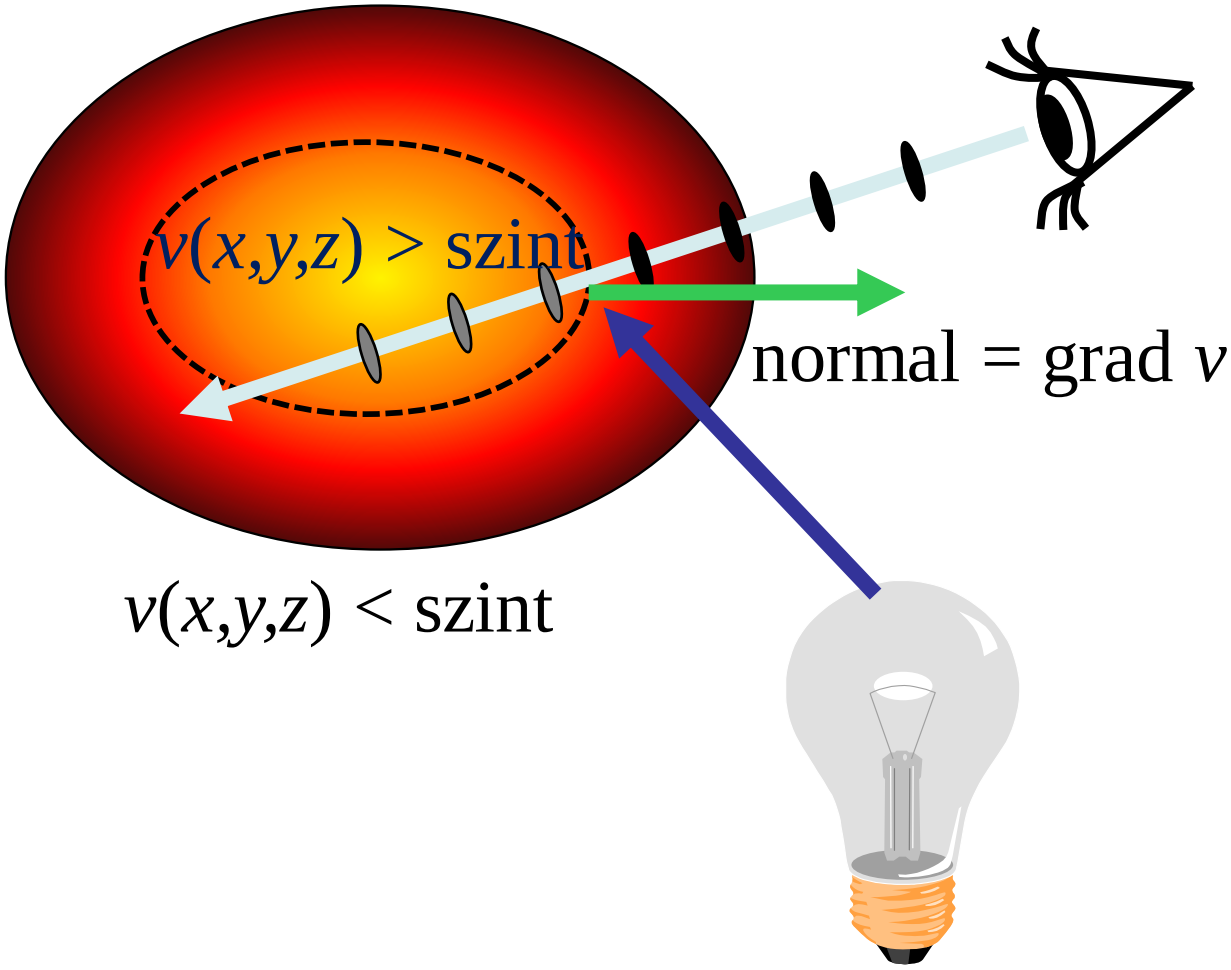


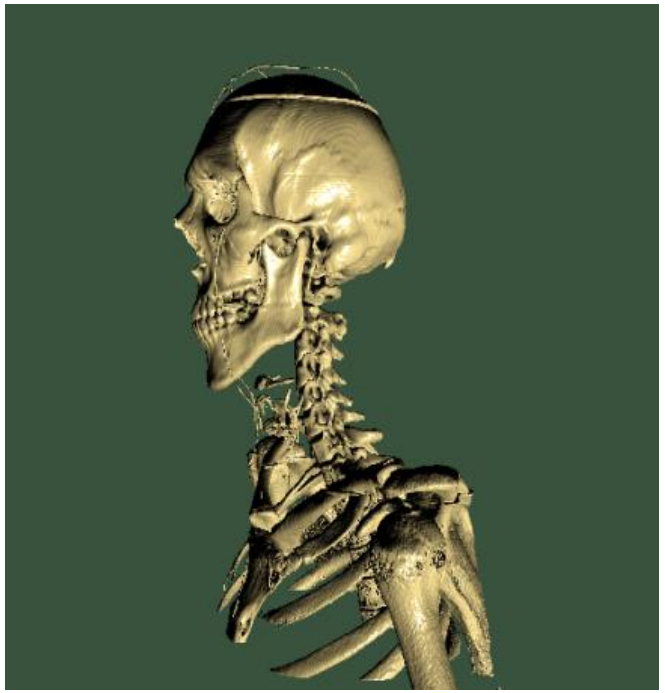
Mélységélesség

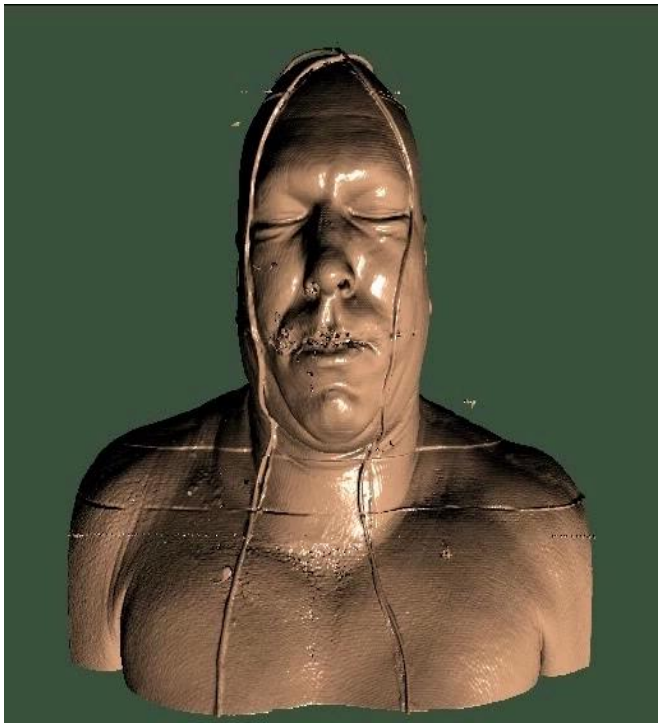
$$r = c\left|\frac{1}{z} - \frac{1}{P}\right|$$



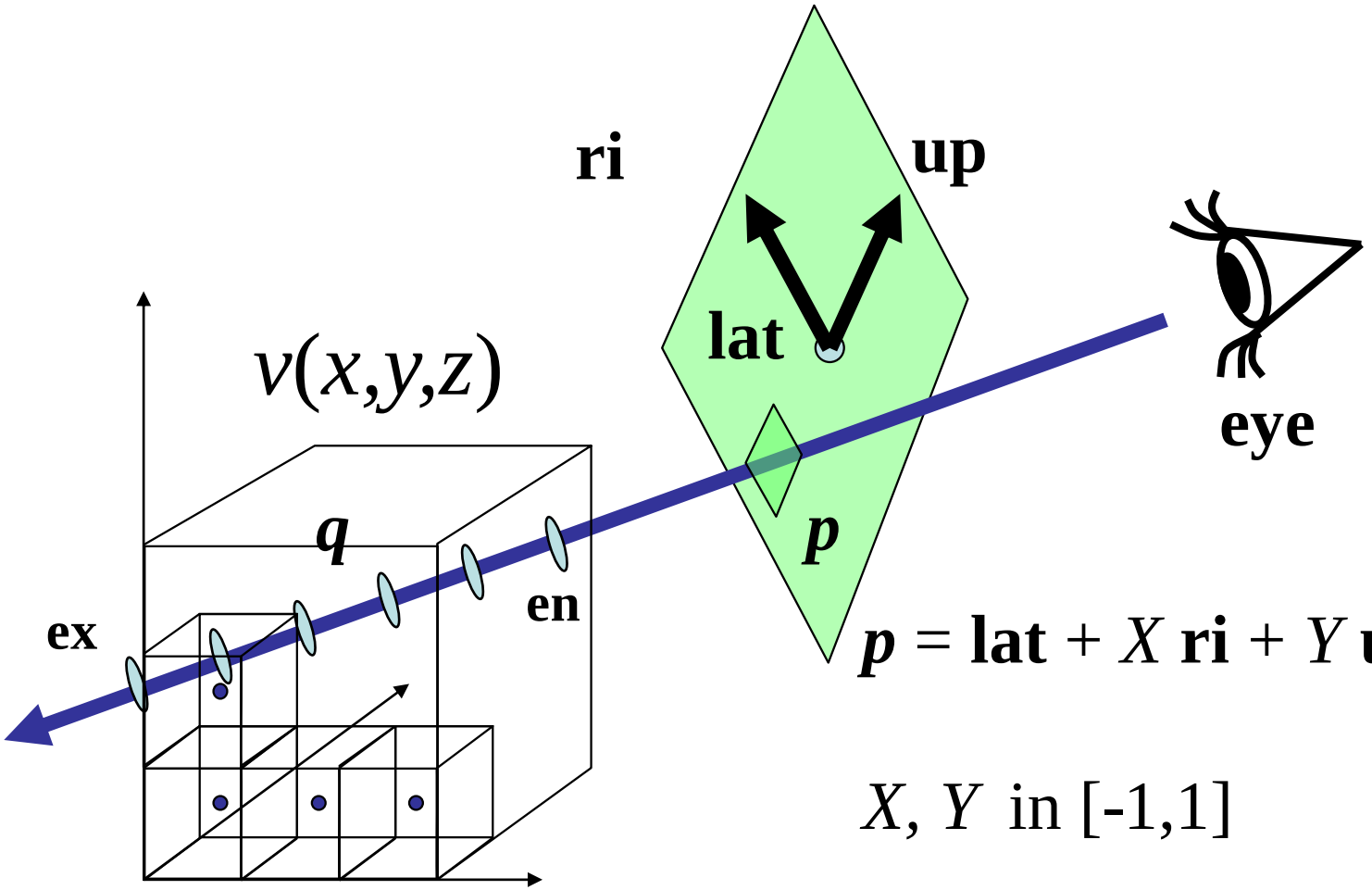
Isosurface ray casting







Isosurface ray casting

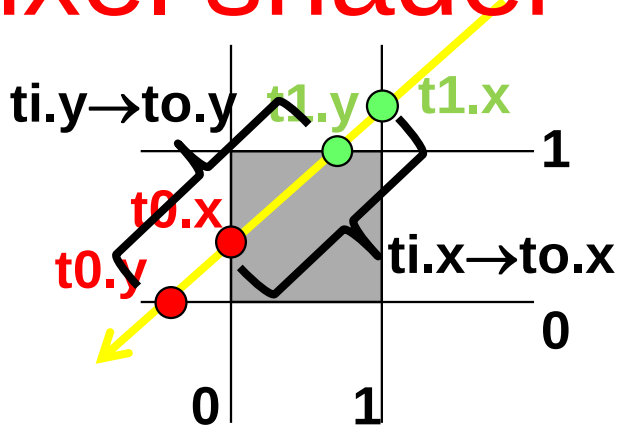


$\boldsymbol{p} = \mathbf{lat} + X\,\mathbf{ri} + Y\,\mathbf{up}$

X, Y in [-1,1]

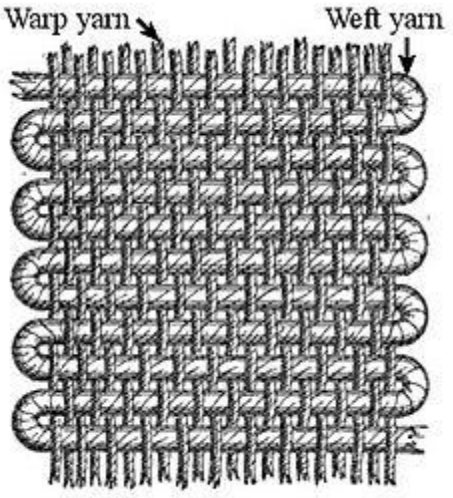
Egység kocka 3D textúrával

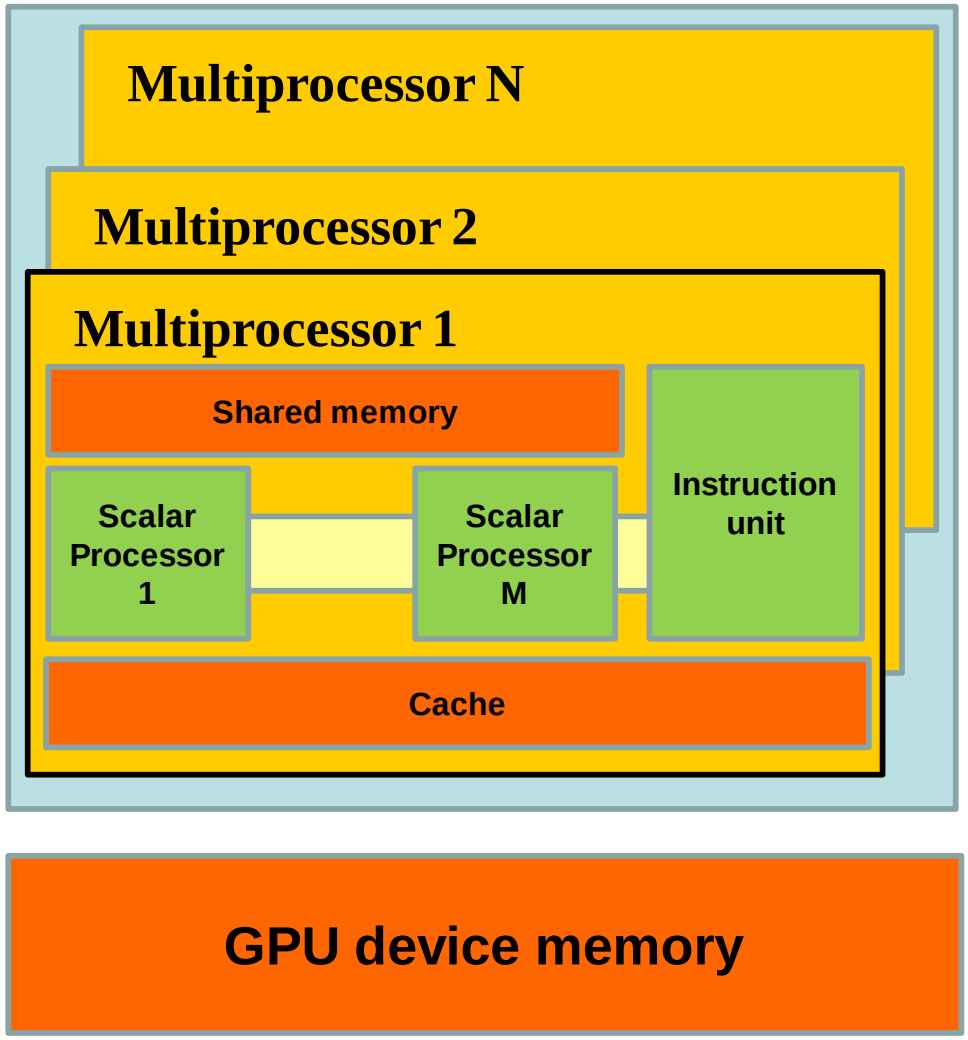
Pixel shader




```
uniform sampler3D vol;
uniform float isolevel, R, dt;
uniform vec3 eye, lat, ri, up;
uniform vec3 L, Lin, kd, background;
in vec2 uv;
out vec4 color;

void main() {
  vec3 p = lat+ri*(2*uv.x-1)+up*(2*uv.y-1);
  vec3 dir = normalize(p-eye);
  vec3 t0 = (vec3(0,0,0)-eye)/dir, t1 = (vec3(1,1,1)-eye)/dir;
  vec3 ti = min(t0, t1), to = max(t0, t1);
  float en=max(max(ti.x,ti.y),ti.z), ex=min(min(to.x,to.y),to.z);
  color = background;
  vec3 dx=vec3(1/R,0,0), dy=vec3(0,1/R,0), dz=vec3(0,0,1/R);
  for(float t = en; t < ex; t += dt) {
     vec3 q = eye + dir * t;
     if (texture(vol, q).x > isolevel) {
        vec3 N = vec3(texture(vol, q+dx) - texture(vol, q-dx),
                      texture(vol, q+dy) - texture(vol, q-dy),
                      texture(vol, q+dz) - texture(vol, q-dz));
        color = lightint * kd * max(dot(L, normalize(N)), 0);
     }
  }
}
```

CUDA (OpenCL)





GPU

Kernel program:

Threads

block, block,

Warp, Warp, …

Shared memory

SIMD execution

SIMD

Két N elemű vektor összeadása

```
#include <cuda.h>

__global__ void AddVectorGPU( float *C, float *A, float *B, int N ) {
    int i = blockDim.x * blockIdx.x + threadIdx.x; // szálazonosító
    if (i < N) C[i] = A[i] + B[i];
}

const int N = 100000;
const int Nb = N * sizeof(float);
float Ccpu[N], Acpu[N], Bcpu[N]; // adat a CPU-n

int main ( ) {  // CPU-n fut
    … // Acpu és Bcpu tömbök feltöltése
    float *Agpu, *Bgpu, *Cgpu; // GPU címtartományba mutató pointerek
    cudaMalloc(&Agpu, Nb); cudaMalloc(&Bgpu, Nb); cudaMalloc(&Cgpu, Nb);
    cudaMemcpy(Agpu, Acpu, Nb, cudaMemcpyHostToDevice);
    cudaMemcpy(Bgpu, Bcpu, Nb, cudaMemcpyHostToDevice);

    int blockDim = 256;          // #threads egy blokkban: 128, 256, 512
    int gridDim = (N + blockDim – 1) / blockDim;   // #blocks

    AddVectorGPU<<<gridDim, blockDim>>>(Cgpu, Agpu, Bgpu, N);

    cudaMemcpy(Ccpu, Cgpu, Nb, cudaMemcpyDeviceToHost);
    cudaFree(Agpu); cudaFree(Bgpu); cudaFree(Cgpu);
    … // A Ccpu használata
}
```

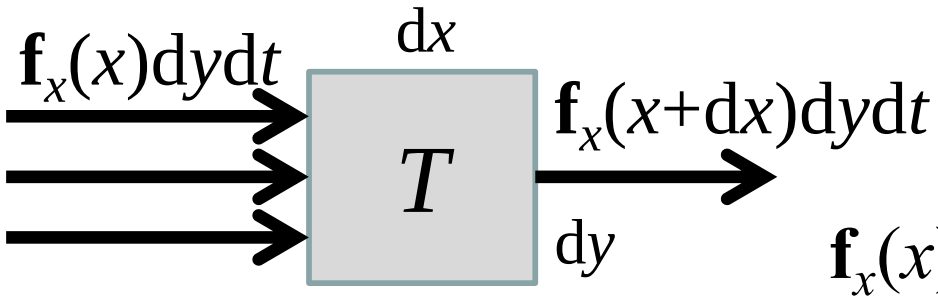
A GPU-n fut, de a CPU-ról is hívható

0 ,…, gridDim.x-1

0 ,…, blockDim.x-1

Hőáramlás, diffúzió

Energia áram



$\mathrm{d}x\mathrm{d}y$ „térfogatú” cella

Nettó energiaváltozás egységnyi térfogatban és egységnyi idő alatt:

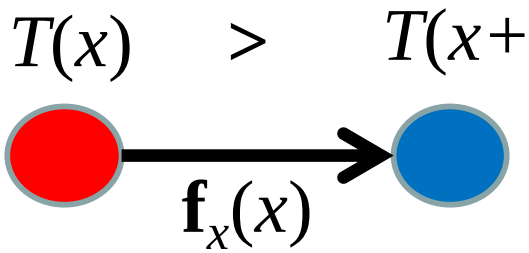
$$\frac{\mathbf{f}_x(x)\mathrm{d}y\mathrm{d}t - \mathbf{f}_x(x+\mathrm{d}x)\mathrm{d}y\mathrm{d}t}{\mathrm{d}x\mathrm{d}y\mathrm{d}t} = -\frac{\partial \mathbf{f}_x(x)}{\partial x}$$

Hőmérséklet változás sebessége:

$$\frac{\partial T}{\partial t} = -k\left(\frac{\partial \mathbf{f}_x}{\partial x} + \frac{\partial \mathbf{f}_y}{\partial y}\right) = -k\nabla\cdot\mathbf{f}$$

1/hőkapacitás

Hőáramot a hőmérséklet különbség hozza létre



hővezetés

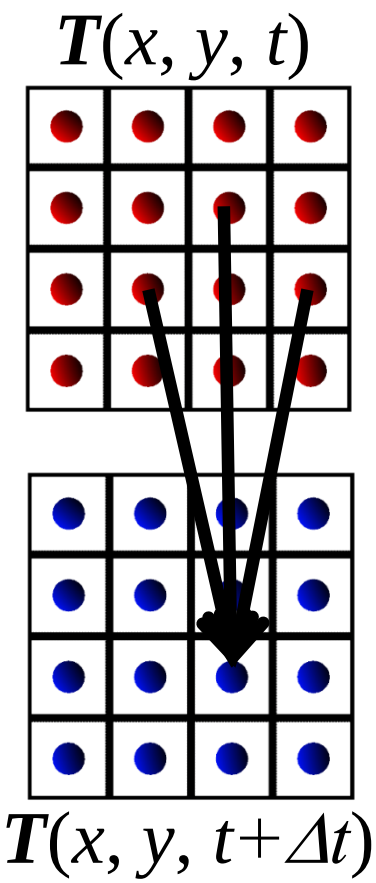
$$\mathbf{f} = (\mathbf{f}_x, \mathbf{f}_y) = -\rho\left(\frac{\partial T}{\partial x}, \frac{\partial T}{\partial y}\right) = -\rho\nabla T$$

Behelyettesítve:

$$\frac{\partial T}{\partial t} = -k\left(\frac{\partial \mathbf{f}_x}{\partial x} + \frac{\partial \mathbf{f}_y}{\partial y}\right) = -k\nabla\cdot\mathbf{f}$$

$g = k\rho$

$$\frac{\partial T}{\partial t} = g\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) = g\nabla^2 T$$



$$\frac{T'(x+\Delta x/2)-T'(x-\Delta x/2)}{\Delta x} = \frac{\frac{T(x+\Delta x)-T(x)}{\Delta x} - \frac{T(x)-T(x-\Delta x)}{\Delta x}}{\Delta x}$$

$$\frac{\partial T}{\partial t} = g\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) = g\nabla^2 T$$

$$\frac{T(x, y, t+\Delta t)-T(x, y, t)}{\Delta t} = g\,\frac{T(x+\Delta x, y, t)-2T(x, y, t)+T(x-\Delta x, y, t)}{(\Delta x)^2} + \cdots$$

Kifejezve az új időpillanatra ($T(x, y, t+\Delta t)$):

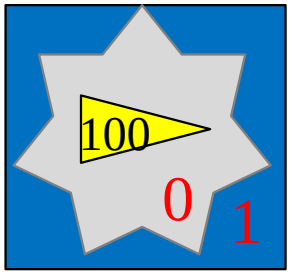
$$T(x, y, t+\Delta t) = T(x, y, t) + g\,\frac{T(x+\Delta x, y, t)-2T(x, y, t)+T(x-\Delta x, y, t)}{(\Delta x)^2}\,\Delta t + \cdots$$

$$\frac{\partial^2 T}{\partial x^2}$$

$$\Delta T = g\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)\cdot \Delta t$$

```
__global
void Diff                t* Told, float
                         t g, float dx, f
    int x =  hreadIdx.x, y = blo
    int p = /*N+x, left=p-(x>0), right=p+(x<N), up=p+(y<N) * N, down=p-(y>0) * N;
    float dTx2 = (Told[right] – 2 * Told[p] + Told[left]) / (dx*dx) ;
    float dTy2 = (Told[up] – 2 * Told[p] + Told[down])) / (dy*dy) ;
    Tnew[p] = (boundary[p]==0) ? Told[p] + g * (dTx2 + dTy2) * dt : Told[p];
}
```

```
const int N = 512, Nb = N*N*sizeof(float); // number of bytes
float Tcpu[N*N] = /*initial+boundary condition*/ , Bcpu[N*N] = /*1-0  mask*/

int main ( ) {
    float *Tgpu1, *Tgpu2, *Bgpu; // GPU címtartomány
    const float g = 3, dx=1, dy=1, dt = 0.1, tend=10;

    cudaMalloc(&Tgpu1, Nb); cudaMemcpy(Tgpu1, Tcpu, Nb, cudaMemcpyHostToDevice);
    cudaMalloc(&Tgpu2, Nb);
    cudaMalloc(&Bgpu, Nb);  cudaMemcpy(Bgpu, Bcpu, Nb, cudaMemcpyHostToDevice);

    for(float t = 0; t < tend; t += dt) {
        Diffuse<<<N, N>>>(Tgpu1, Tgpu2, Bgpu, g, dx, dy, dt, N); // Ping
        Diffuse<<<N, N>>>(Tgpu2, Tgpu1, Bgpu, g, dx, dy, dt, N); // Pong
    }

    cudaMemcpy(Tcpu, Tgpu1, nb, cudaMemcpyDeviceToHost);
    cudaFree(Tgpu1); cudaFree(Tgpu2);
}
```



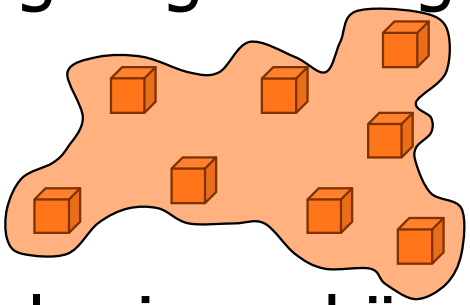
Mikor jó?

- Párhuzamos algoritmus (>10k szál):
  - A probléma elemzésével kezdődik!
- Gyűjtő típusú algoritmus (nincsenek írási memóriaütközések)
- Kevés feltételes utasítás (thread divergencia)
- Adatlokalitás
- Számításintenzív

Tudományos, mérnöki számítások

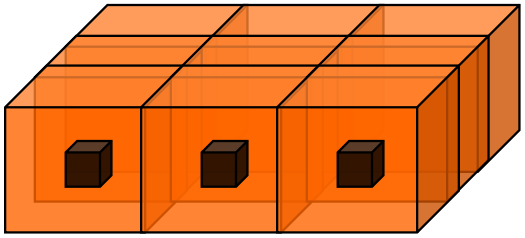
Numerikus megoldás

- Tér-idő differenciál egyenletek
- Idő és térkoordináták diszkretizálása (véges elem)
- Idő: diszkrét idő vagy diszkrét esemény
- Tér: Lagrange-i megközelítés



Mozgó részecskék

Euler-i megközelítés

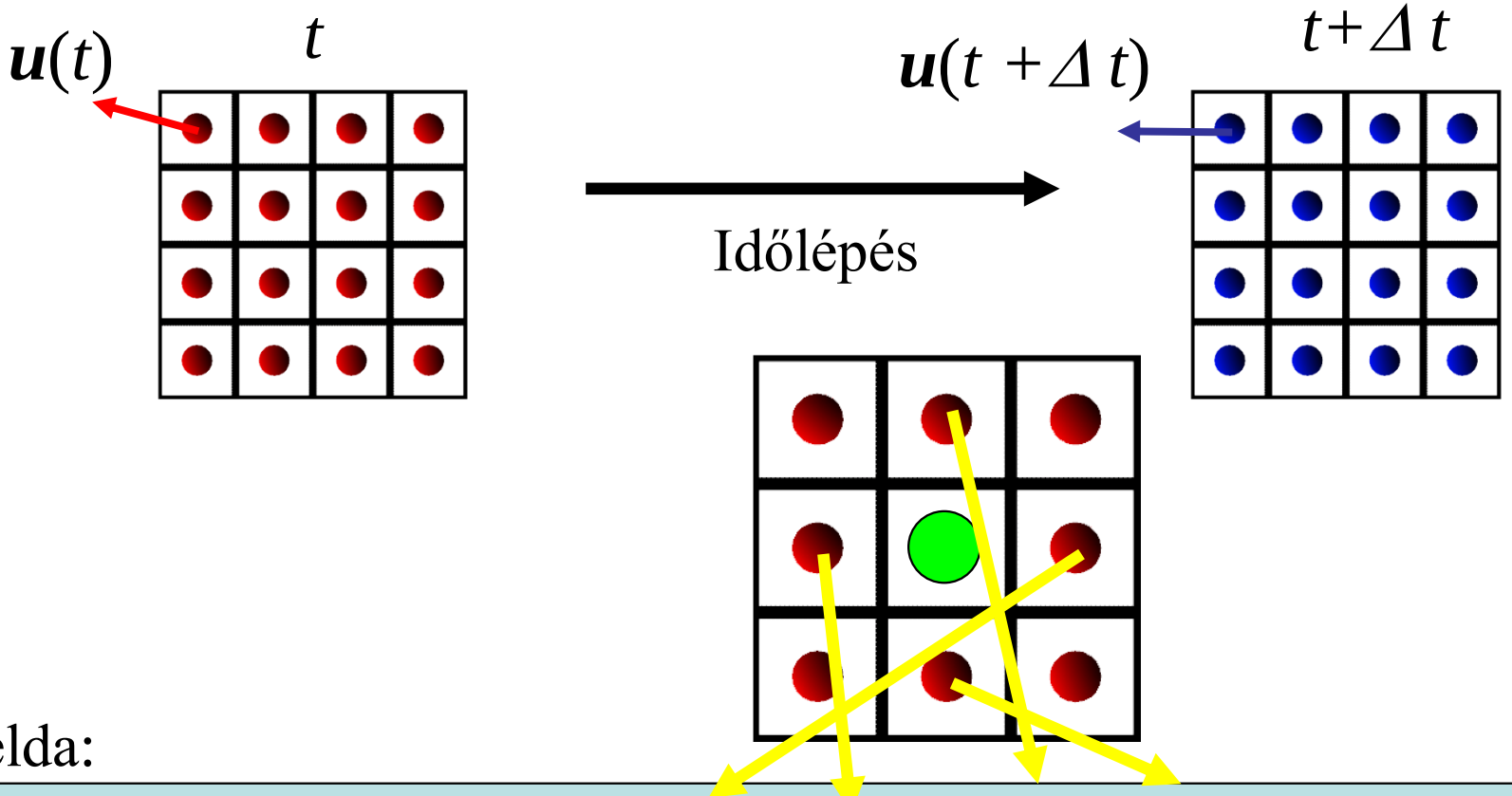


Tér mintavételezése
rögzített rácson

$$\frac{d\vec{u}}{dt} = -(\vec{u}\cdot\nabla)\vec{u} - \frac{1}{\rho}\nabla p + \nu\nabla^2\vec{u} + \vec{F}$$

$$\nabla\cdot\vec{u} = 0$$

Euler-i folyadék



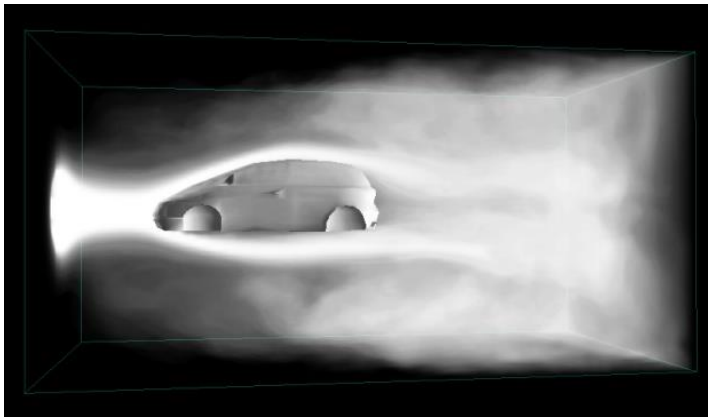
Példa:

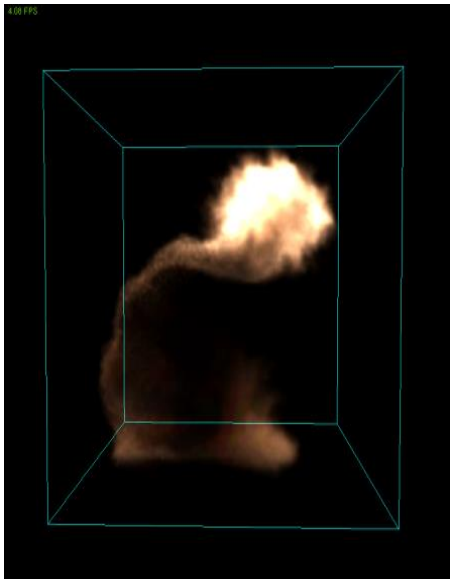
$$\nabla\cdot\vec{u} = \operatorname{div}\vec{u} = \frac{\partial\vec{u}_x}{\partial x} + \frac{\partial\vec{u}_y}{\partial y} + \frac{\partial\vec{u}_z}{\partial z} = \frac{\vec{u}_x^{\,i+1,j,k} - \vec{u}_x^{\,i-1,j,k}}{2\delta x} + \frac{\vec{u}_y^{\,i,j+1,k} - \vec{u}_y^{\,i,j-1,k}}{2\delta y} + \frac{\vec{u}_z^{\,i,j,k+1} - \vec{u}_z^{\,i,j,k-1}}{2\delta z}$$

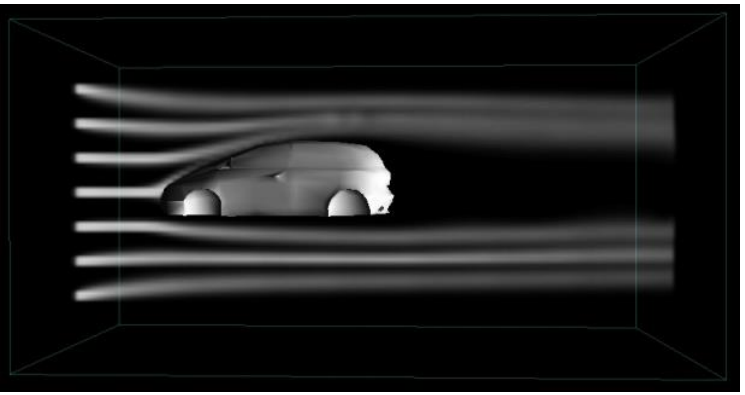
$$\frac{d\vec{u}}{dt} = -(\vec{u}\cdot\nabla)\vec{u} - \frac{1}{\rho}\nabla p + \nu\nabla^2\vec{u} + \vec{F}$$

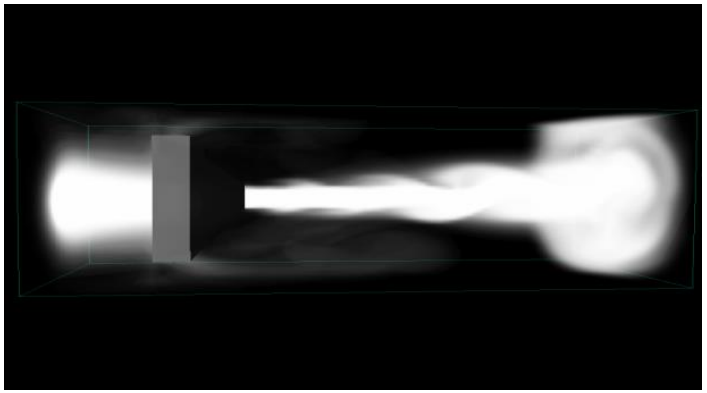
$$\nabla\cdot\vec{u} = 0$$

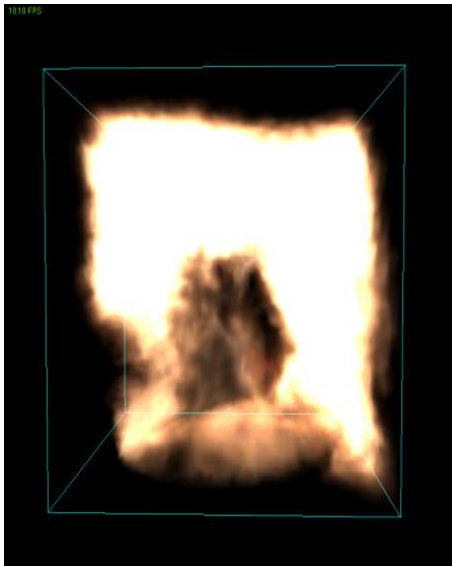
Euler-i folyadék



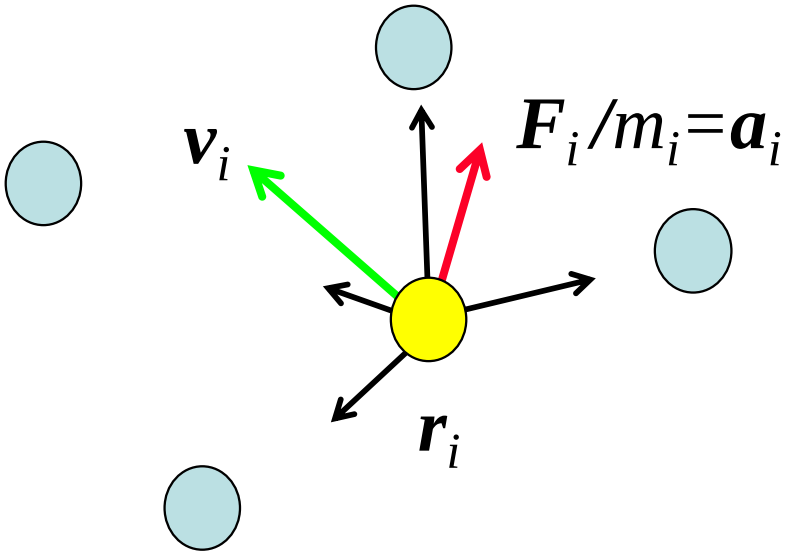








Lagrange-i módszer: Részecske rendszer, N-body



$$\boldsymbol{v}_i \mathrel{+}= \boldsymbol{a}_i \Delta t$$
$$\boldsymbol{r}_i \mathrel{+}= \boldsymbol{v}_i \Delta t$$

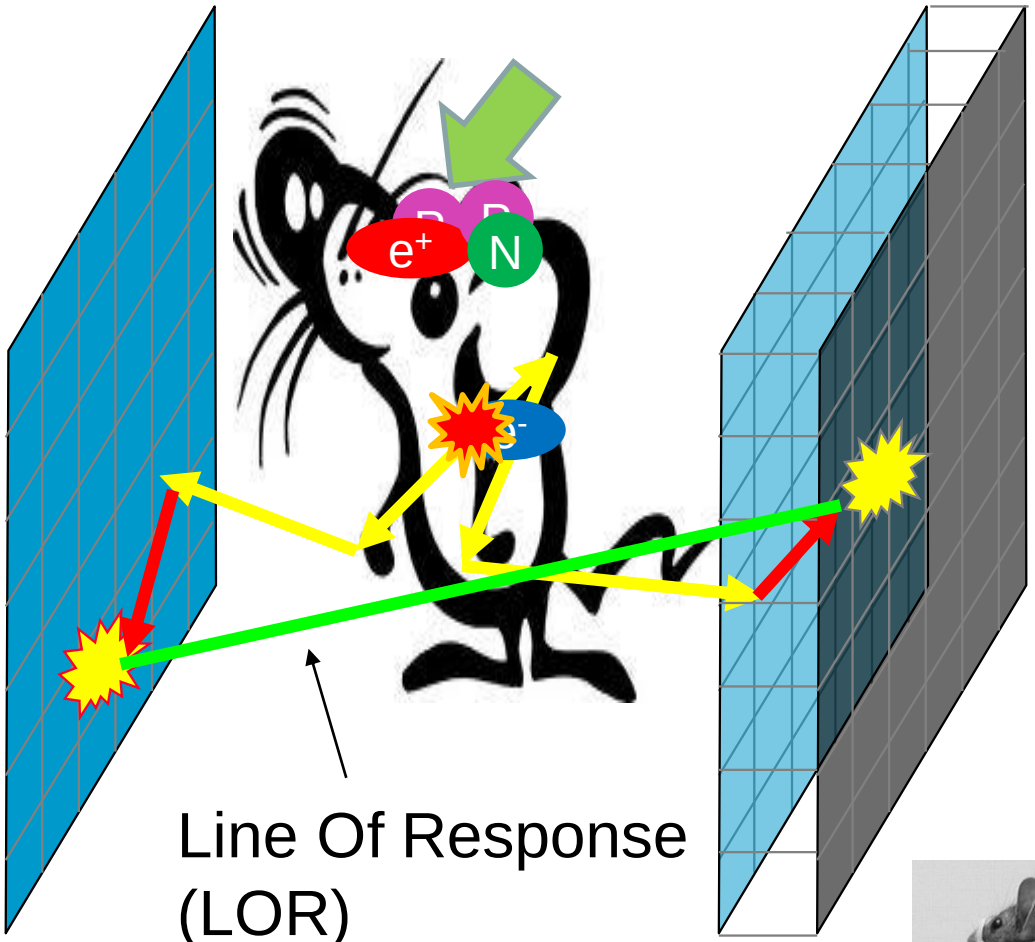
Előrelépő Euler séma.

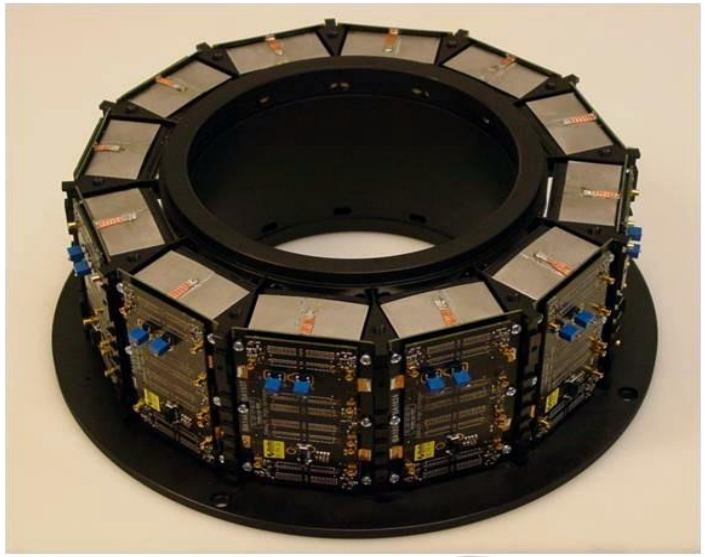
Helyette: Visszalépő Euler, Runte-Kutta, Midpoint, Verlet, …

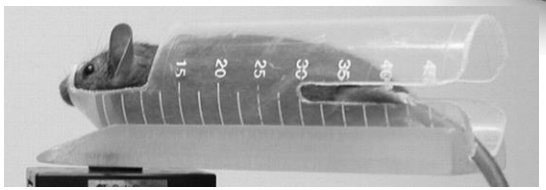
Lagrange-i módszer:
SPH: Smoothed Particle Hidrodynamics

Folyadékáramlás: Szívbillentyű tervezés

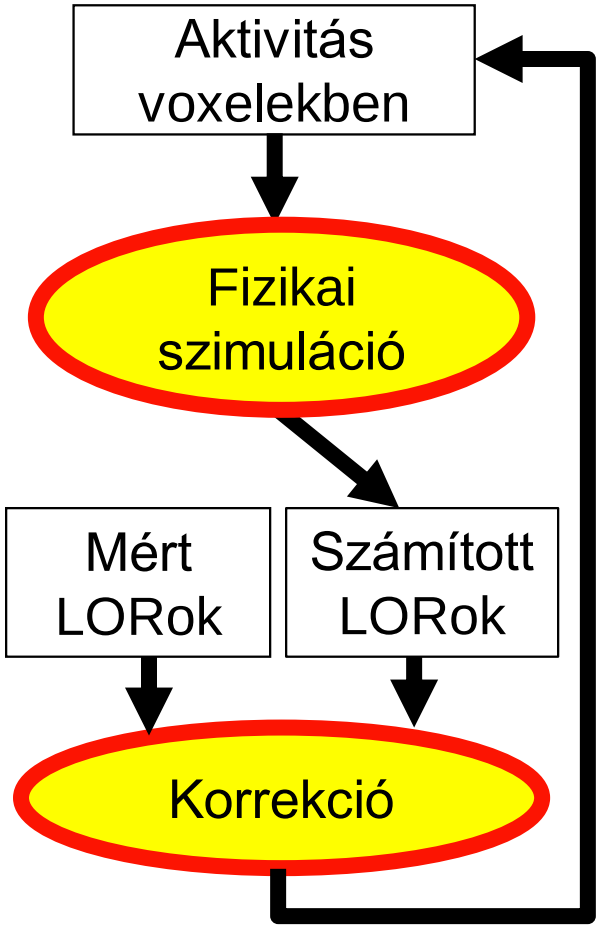
Inverz problémák:
Pozitron emissziós tomográfia



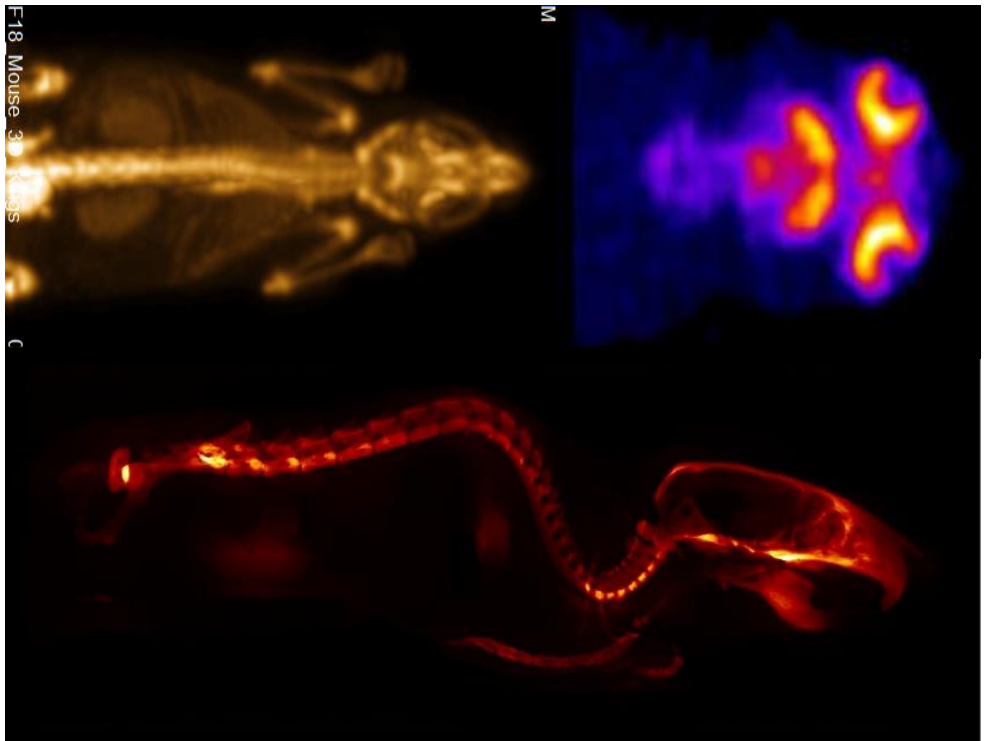




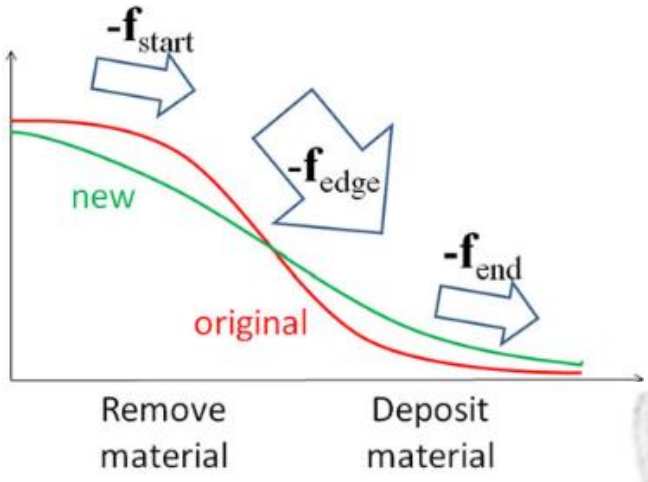
PET rekonstrukció



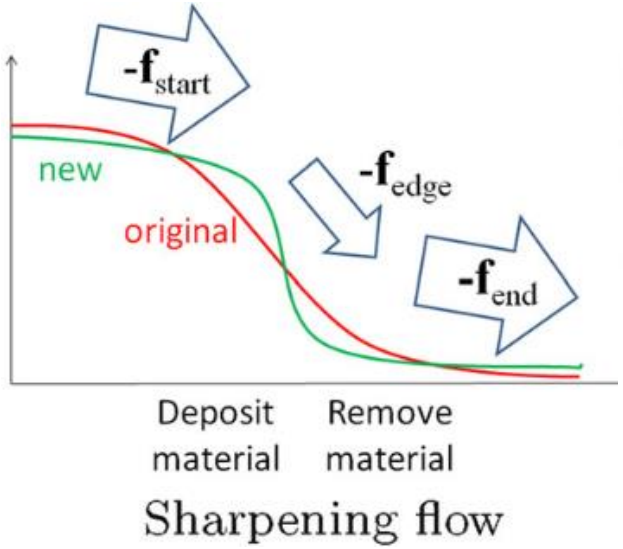
Voxel, LOR szám: több százmillió



Anizotróp diffúzió

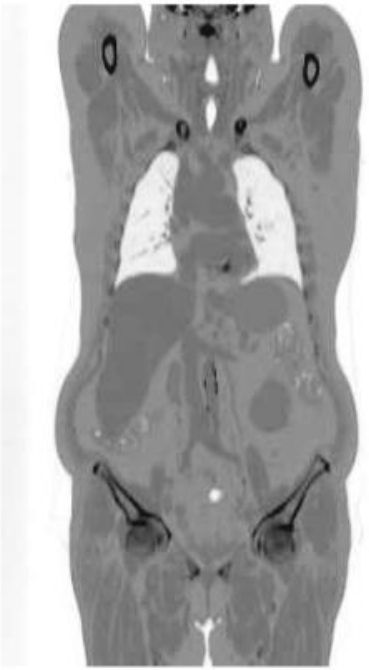


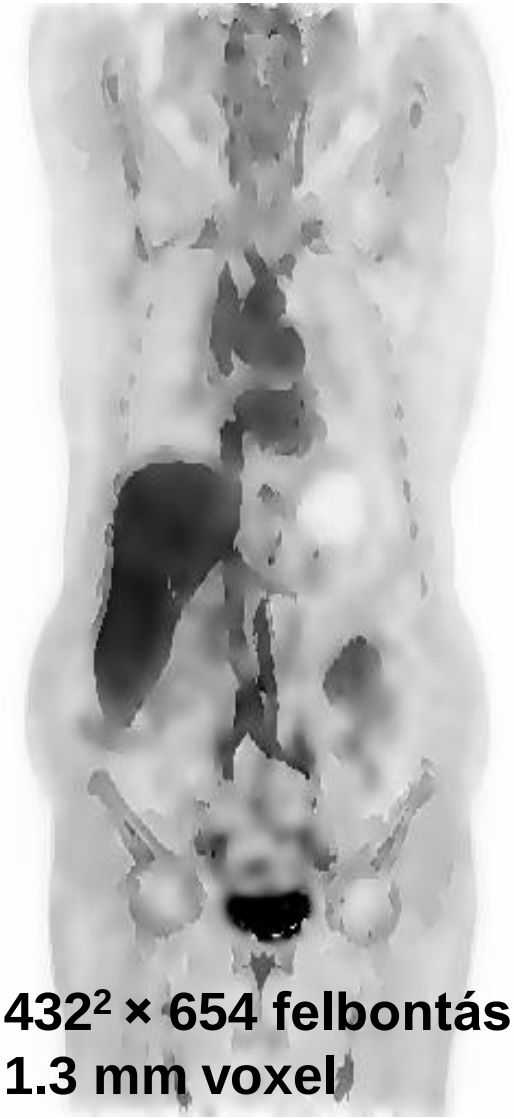
Smoothing flow



Sharpening flow

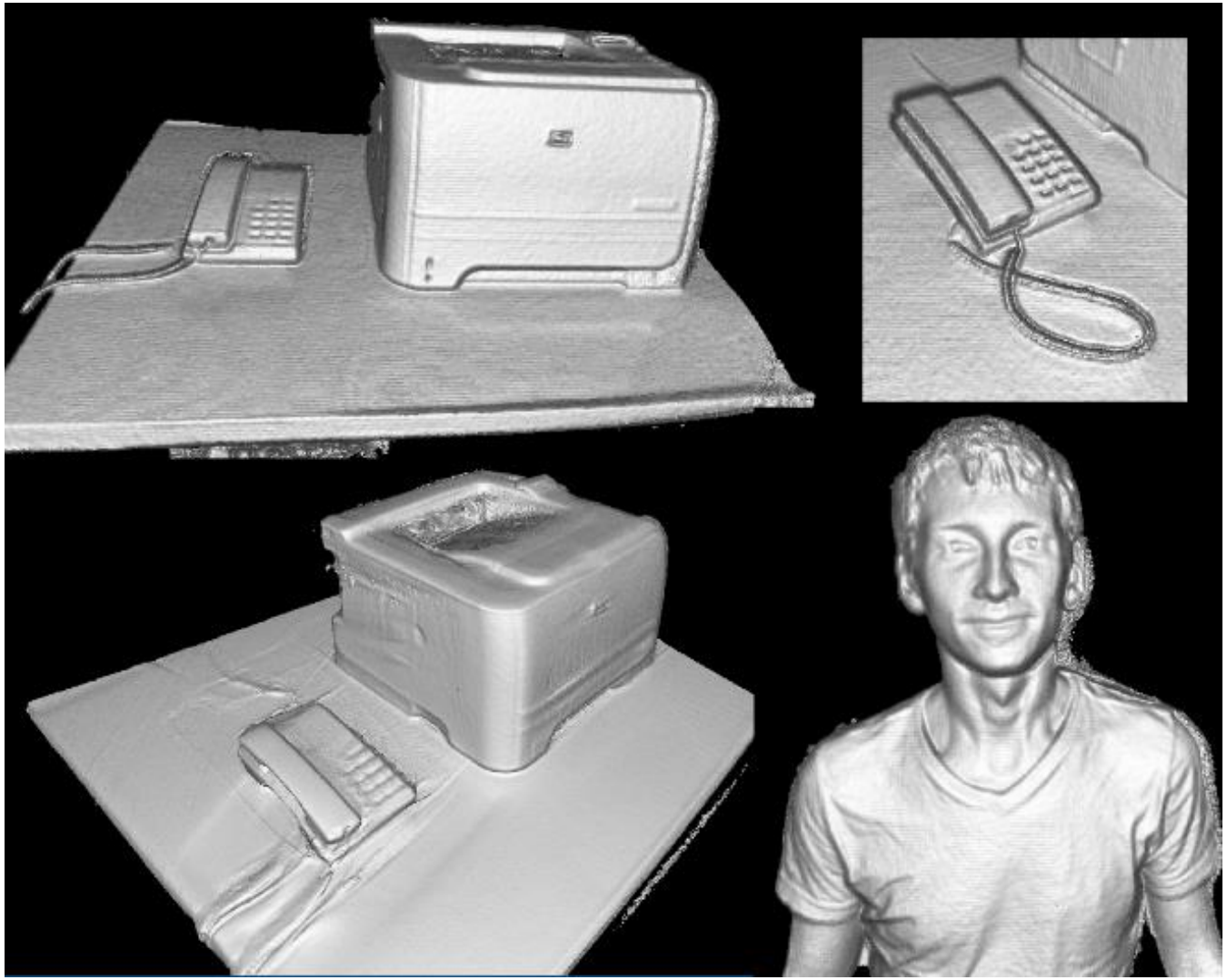

Original PET


Resampled CT



$432^2 \times 654$ felbontás
1.3 mm voxel

Geometria rekonstrukciója mélységképekből (Kinect)



Mélységkép javítás

Free viewpoint video