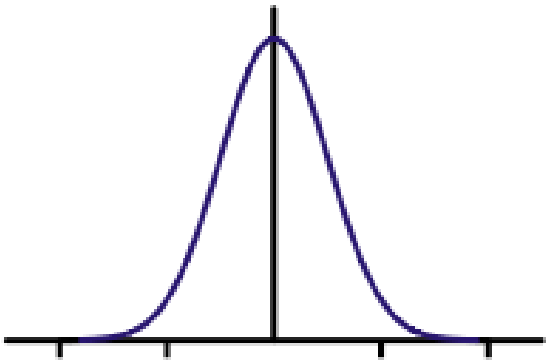
Iteratív algoritmusok

Konvolúció

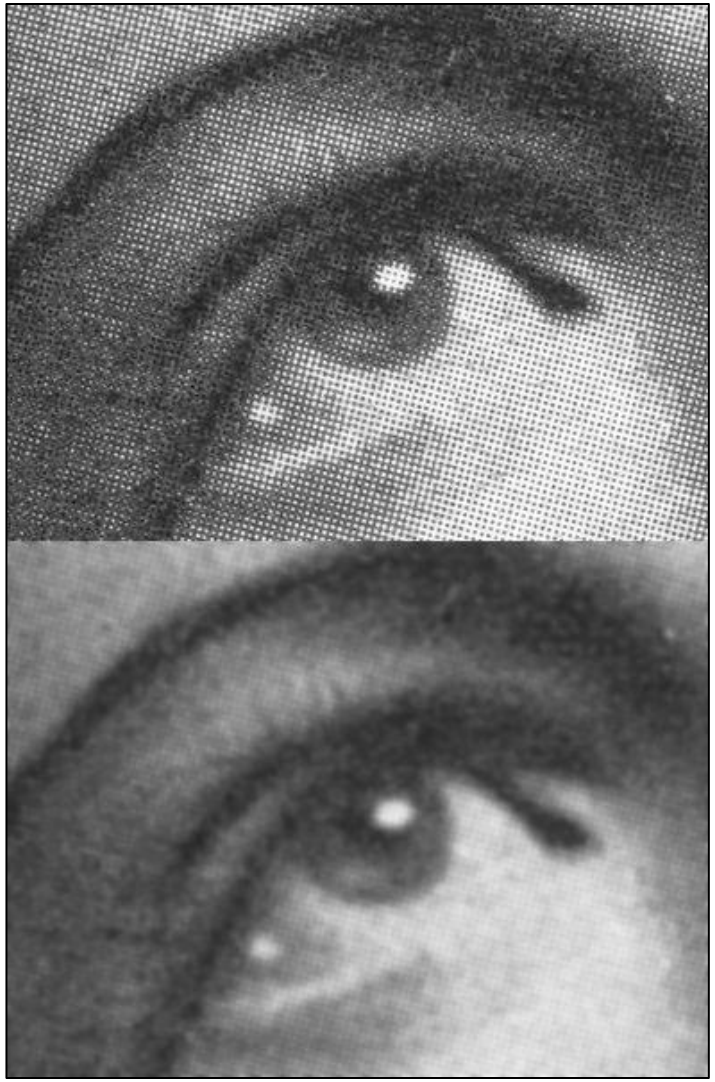
- Gauss szűrő

$$L'(X,Y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} L(X-x, Y-y)\, w(X,Y,x,y)\, dx dy$$

$$w(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} / 16$$



Konvolúció

- Gauss szűrő

```
uniform sampler2D colorMap;

const float kernel[9] = float[9]( 1.0, 2.0, 1.0,
                                  2.0, 4.0, 2.0,
                                  1.0, 2.0, 1.0);
out vec4 outColor;

void main(){
 outColor = vec4(0.0);
 for(int x=-1; x<1; ++x)
  for(int y=-1; y<1; ++y)
   outColor += texelFetch(colorMap, ivec2(gl_FragCoord) + ivec2(x,y)) *
                                                kernel[(x+1)+(y+1)*3] / 16.0;
}
```

Szeparábilis szűrők

$$L'(X,Y) \approx \sum_{i=-N/2}^{N/2} \sum_{j=-N/2}^{N/2} L(X-i, Y-j)\, w(i,j)$$

$$w(x,y) = w_x(x) \cdot w_y(y)$$

$$L_h(X,Y) \approx \sum_{i=-N/2}^{N/2} L(X-i, y)\, w_x(i)$$

$$L'(X,Y) \approx \sum_{i=-N/2}^{N/2} L_h(X, Y-i)\, w_y(i)$$

Szeparábilis szűrők

```
uniform sampler2D colorMap;
const float kernel[3] = float[3]( 1.0, 2.0, 1.0 );
out vec4 outColor;

void main(){
 outColor = vec4(0.0);
 for(int x=-1; x<1; ++x)
  outColor += texelFetch(colorMap,
                         ivec2(gl_FragCoord) + ivec2(x,0)) * kernel[x+1] / 4.0;
}
```

```
uniform sampler2D colorMap;
const float kernel[3] = float[3]( 1.0, 2.0, 1.0 );
out vec4 outColor;

void main(){
 outColor = vec4(0.0);
 for(int y=-1; y<1; ++y)
  outColor += texelFetch(colorMap,
                         ivec2(gl_FragCoord) + ivec2(0,y)) * kernel[y+1] / 4.0;
}
```

Szeparábilis szűrők

```
uniform sampler2D colorMap;
uniform vec2 textureSize;

in vec2 fTexCoord;
out vec4 outColor;

const float ONE_PER_SQRT_TWOPI = 0.3989;

float w1(float x){
 return ONE_PER_SQRT_TWOPI / sigma / sigma * exp(-x*x/2.0/sigma/sigma);
}

void main(){
 outColor = vec4(0.0);
 for(int i= -N/2; i<N/2; ++i){
  float d = i / textureSize.x;
  outColor += texture(colorMap, fTexCoord – vec2(d,0.0)) * w1(d);
 }
}
```

Szeparábilis szűrők

- Fontosság szerinti mintavétel

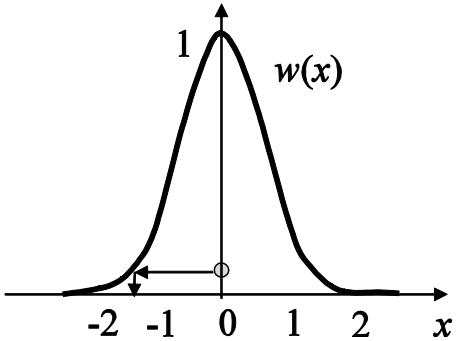
$$L'(X) = \int_{-\infty}^{\infty} L(X - x)w(x)dx$$

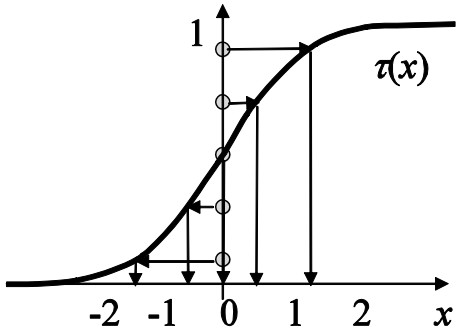
$$\frac{d\tau}{dx} = w(x) \longrightarrow \tau(x) = \int_{-\infty}^{x} w(t)dt$$

$$L'(X) = \int_{-\infty}^{\infty} L(X - x)w(x)dx = \int_{-\infty}^{\infty} L(X - x)\frac{d\tau}{dx} =$$

$$\int_{\tau(-\infty)}^{\tau(\infty)} L(X - x(\tau))d\tau = \int_{0}^{1} L(X - x(\tau))d\tau$$

$$\boxed{L'(X) = \int_{0}^{1} (X - x(\tau))d\tau \approx \frac{1}{M}\sum_{i=1}^{M} L\left(X - x\left(\frac{2i-1}{2M}\right)\right)}$$





Szeparábilis szűrők

- Szeparálható Gauss szűrő

```
uniform sampler2D colorMap;
in vec2 texCoord;

const float offset[5] = float[5](-1.282, -0.524, 0, 0.524, 1.282);

void main(){
 for(int x=-2; x<2; ++x)
  outColor += I(texCoord + vec(offset[x], 0.0)) / 5.0;
}
```

```
uniform sampler2D colorMap;
in vec2 texCoord;

const float offset[5] = float[5](-1.282, -0.524, 0, 0.524, 1.282);

void main(){
 for(int y=-2; y<2; ++y)
  outColor += I(texCoord + vec(0.0, offset[y])) / 5.0;
}
```

Framebuffer

- A renderelés célpontja
  - Összefogja a rajzolási állapotokat
  - Hasonló a textúra objektumhoz

- Képernyő bufferek
- Framebuffer object
  - Color buffer
    - GL_MAX_COLOR_ATTACHMENTS
  - Depth buffer
  - Stencil buffer

Framebuffer

```
GLuint framebuffer;
glGenFramebuffers(1, &framebuffer);
glBindFramebuffer(GL_FRAMEBUFFER, framebuffer);

GLuint* colorAttachments = new GLuint[planes];
glGenTextures(planes, colorAttachments);
for(int i=0; i<planes; ++i){
 // textura beallitasok
 glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0 + i,
  GL_TEXTURE_2D, colorAttachments[i], 0);
}

GLuint depthBuffer;
glGenRenderbuffers(1, &depthBuffer);
glBindBuffer(GL_RENDERBUFFER, depthBuffer);
glRenderbufferStorage(GL_RENDERBUFFER, GL_DEPTH_COMPONENT24, width, height);
glFramebufferRenderBuffer(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT,
                                                GL_RENDERBUFFER, depthBuffer);

glBindFramebuffer(GL_FRAMEBUFFER, 0);
```

```
GLenum status = glCheckFramebufferStatus(GL_FRAMEBUFFER);
```

Framebuffer

- Használható formátumok
  - Color attachment
    - GL_RED, GL_RG, GL_RGB, GL_RGBA
  - Depth attachment
    - GL_DEPTH_COMPONENT
    - GL_DEPTH_STENCIL
  - Stencil attachment
    - GL_STENCIL_INDEX
    - GL_DEPTH_STENCIL

Framebuffer

- Framebuffer kritériumok
  - Minden attachment komplett
    - Megfelelő típusok a csatlakozási pontokon
    - Használható formátum
    - Nem nulla méret
  - Legalább egy képbuffer van
  - Implementáció függő megkötések

Framebuffer

- Beállítás a renderelés célpontjává

```
glBindFrameBuffer(GL_FRAMEBUFFER, framebufferObject);
glDrawBuffers(planes, buffers);
glViewport(0, 0, width, height);
```

- Megkötések
  - Ha nem egyforma méretűek a bufferek
    - A metszet területre rajzolunk
  - Ha nem egyforma a layerek száma
    - A metszet layerekre rajzolunk
  - A többi pixel értéke határozatlan lesz!

Framebuffer

- Több szín attachment használata

```
out vec4 outColor[3];

void main(void){
 outColor[0] = vec4(1.0, 0.0, 0.0, 1.0);
 outColor[1] = vec4(0.0, 1.0, 0.0, 1.0);
 outColor[2] = vec4(0.0, 0.0, 1.0, 1.0);
}
```

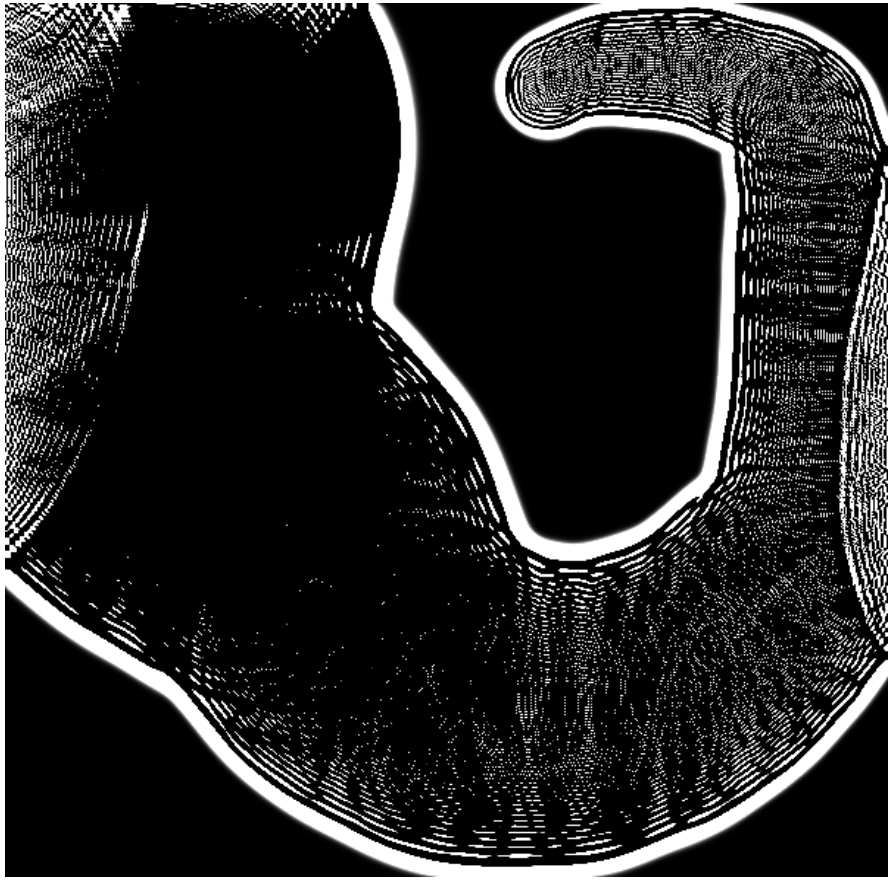
```
out vec4 outColor[3];

void main(void){
 outColor = vec4[3]( vec4(1.0, 0.0, 0.0, 1.0),
                     vec4(1.0, 0.0, 0.0, 1.0),
                     vec4(1.0, 0.0, 0.0, 1.0));
}
```

Iteratív algoritmusok

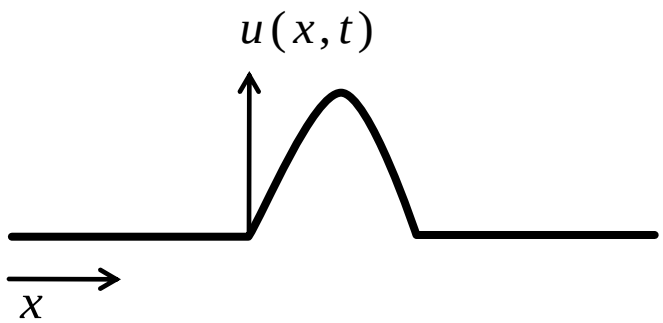
- Ping-pongozás

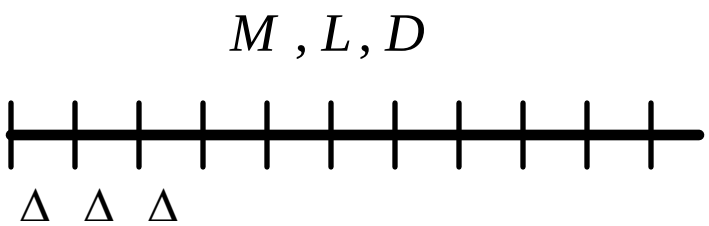
```
Framebuffer* computeBuffer[2];
int inputBuffer = 0;

// inicializálás
computeBuffer[0] = new Framebuffer(width, height, 1);
computeBuffer[1] = new Framebuffer(width, height, 1);

// iteráció
computeBuffer[(inputBuffer + 1) % 2]->setRenderTarget();
shader->enable();
shader->bindUniformTexture(„inputMap”,
                           computeBuffer[inputBuffer]->getColorBuffer(0), 0);
fullscreenQuad->render(shader);
shader->disable();
computeBuffer[(inputBuffer + 1) % 2]->disableRenderTarget();

inputBuffer = (inputBuffer + 1) % 2;
```

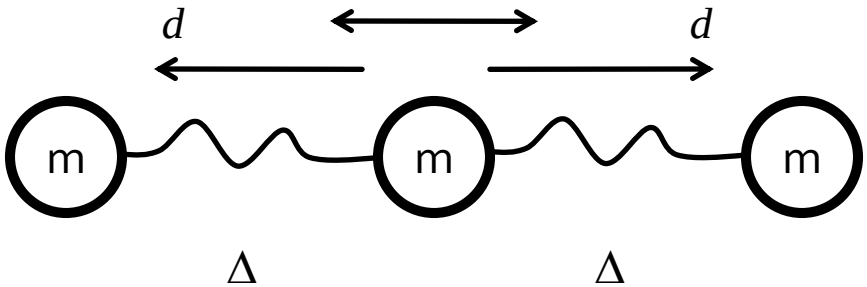
Hullám egyenlet



Hullám egyenlet





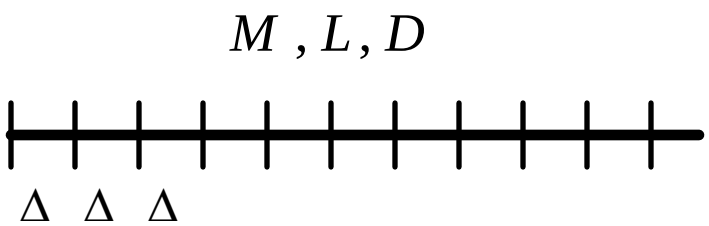


$u(x-\Delta,t)$ $\quad$ $u(x,t)$ $\quad$ $u(x+\Delta,t)$

Hullám egyenlet

- Newton

$$F = m \cdot a = m \frac{\partial^2 u}{\partial t^2}$$



- Hooke

$$F = \Delta\, d \cdot \left(\frac{u(x+\Delta,t) - u(x,t)}{\Delta} - \frac{u(x,t) - u(x-\Delta,t)}{\Delta} \right)$$

$$F = \Delta^2 d \cdot \left(\frac{\frac{u(x+\Delta,t) - u(x,t)}{\Delta} - \frac{u(x,t) - u(x-\Delta,t)}{\Delta}}{\Delta} \right)$$

$$\frac{\left.\frac{\partial u}{\partial x}\right|_x - \left.\frac{\partial u}{\partial x}\right|_{x-\Delta}}{\Delta} = \frac{\partial u^2}{\partial x^2}$$

Hullám egyenlet

$$m \cdot \frac{\partial^2 u}{\partial t^2} = \Delta^2 d \frac{\partial^2 u}{\partial x^2}$$

$$\frac{\partial^2 u}{\partial t^2} = \frac{\Delta^2 d}{m} \cdot \frac{\partial^2 u}{\partial x^2}$$

$$\left.\begin{aligned} m &= M \frac{\Delta}{L} \\ d &= D \frac{L}{\Delta} \end{aligned}\right\} \frac{d}{m} = \frac{DL^2}{M\Delta^2}$$

$$\frac{\partial^2 u}{\partial t^2} = \underbrace{\frac{DL^2}{M}}_{c^2} \frac{\partial^2 u}{\partial x^2} \quad \Rightarrow \quad \boxed{\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}}$$

Hullám egyenlet

$$t \rightarrow t + \Delta t$$

$$u(t) = \frac{u(t+\Delta t) - v(t)}{\Delta t}$$

$$v(t) \cong \frac{u(t+\Delta t) - u(t)}{\Delta t} \quad \Rightarrow \quad v(t) = \frac{\partial u}{\partial t}$$

$$\Downarrow$$

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

Hullám egyenlet

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

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial v}{\partial t}$$

$$\frac{\partial^2 u}{\partial x^2} \cong \frac{\left.\frac{\partial u}{\partial x}\right|_x - \left.\frac{\partial u}{\partial x}\right|_{x-\Delta}}{\Delta} = \frac{u(x+\Delta, y) - 2u(x,y) + u(x-\Delta, y)}{\Delta^2}$$

$$\frac{\partial^2 u}{\partial y^2} \cong \frac{\left.\frac{\partial u}{\partial y}\right|_y - \left.\frac{\partial u}{\partial y}\right|_{y-\Delta}}{\Delta} = \frac{u(x, y+\Delta) - 2u(x,y) + u(x, y-\Delta)}{\Delta^2}$$

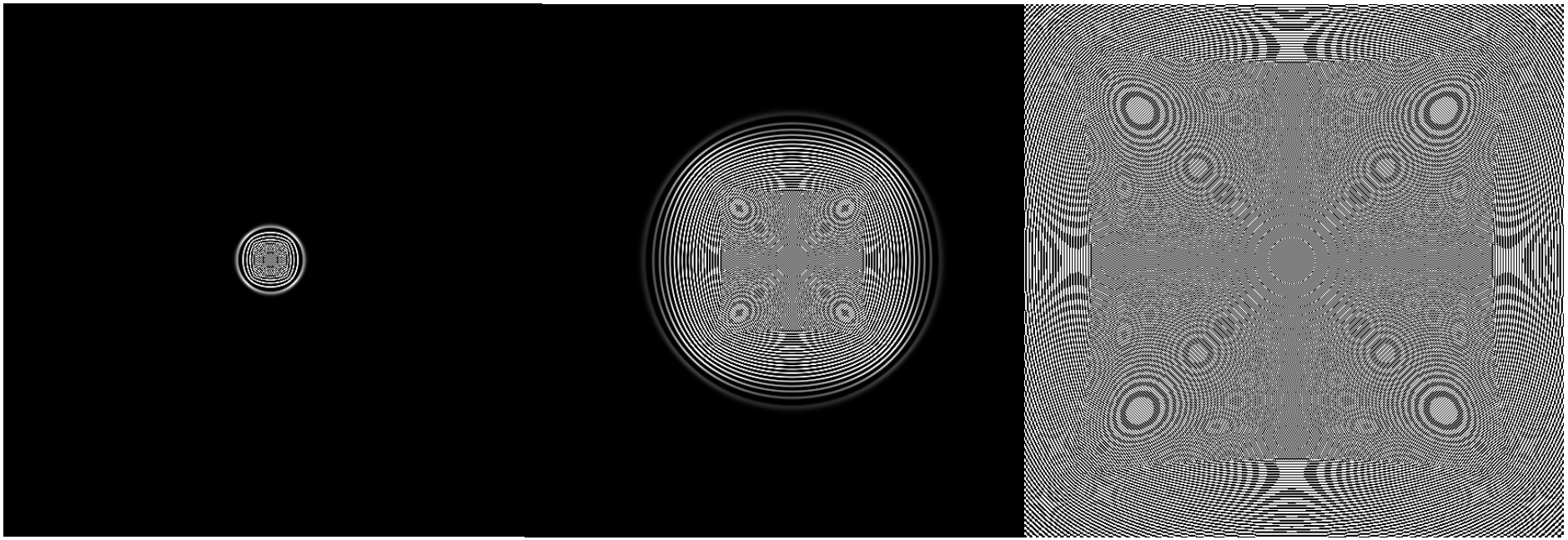
$$\boxed{u(t+\Delta t) = u(t) + v(t)\cdot\Delta t}$$

$$\boxed{v(t+\Delta t) = v(t) + c^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)\cdot\Delta t}$$

Hullám egyenlet

```
uniform sampler2D inputMap;
out vec4 outColor;

const float deltat = 0.0001;
const float deltax = 0.01;
const float sqrc = 20.0;

void main(void){
 vec4 data = texelFetch(inputMap, ivec2(gl_FragCoord), 0);

 float u = data.x;
 float v = data.y;

 float d2u = texelFetch(inputMap, ivec2(gl_FragCoord) + ivec2(-1,  0),0).x +
             texelFetch(inputMap, ivec2(gl_FragCoord) + ivec2( 1,  0),0).x +
             texelFetch(inputMap, ivec2(gl_FragCoord) + ivec2( 0, -1),0).x +
             texelFetch(inputMap, ivec2(gl_FragCoord) + ivec2( 0,  1),0).x -
             4 * u;

 u = u + v * deltat;
 v = v + sqrc * d2u * deltat;

 outColor = vec4(u, v, 0.0, 0.0);
}
```

Hullám egyenlet



Hullám egyenlet

- Hiba

$$\frac{dx}{dt} = f(x,t) \qquad x(t+\Delta t) = x(t) + \frac{dx}{dt}\Delta t = x(t) + f(x,t)\Delta t$$

Iteráció	Hiba
x(0)	0
dt	h
2*dt	2*h
...	...
m*dt	m*h

$$h = o(\Delta t^2)$$

Hullám egyenlet

- Stabilizálás
  - Visszamenő Euler
  - Crank-Nickolson módszer
  - Másodrendű Runge-Kutta
  - Verlet

- Lineáris egyenletrendszer

$$u(t+\Delta t) = u(t) + v(t+\Delta t)\cdot\Delta t$$

$$v(t+\Delta t) = v(t) + c^2\nabla^2 u(t+\Delta t)\cdot\Delta t$$

Hullám egyenlet

Midpoint közelítés

$$\frac{dx}{dt} = f(x,t)$$

$$x_e(t+\Delta t) = x(t) + f(x,t)\Delta t$$

$$x_m = \frac{x(t) + x_e(t+\Delta t)}{2}$$

$$\boxed{x(t+\Delta t) = x(t) + f\left(\frac{x(t) + x_e(t+\Delta t)}{2}, t + \frac{\Delta t}{2}\right)\Delta t}$$

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u$$

$$\left.\begin{aligned} \frac{\partial v}{\partial t} &= c^2 \nabla^2 u \\ \frac{\partial u}{\partial t} &= v \end{aligned}\right\} \frac{\partial [u,v]}{\partial t} = \left[c^2 \nabla^2 u, v\right]$$

Hullám egyenlet

```
uniform sampler2D inputMap;
out vec4 outColor;

const float deltat = 0.0001;
const float deltax = 0.01;
const float sqrc = 20.0;

void main(void){
 vec4 data = texelFetch(inputMap, ivec2(gl_FragCoord), 0);

 float u = data.x;
 float v = data.y;

 float d2u = texelFetch(inputMap, ivec2(gl_FragCoord) + ivec2(-1, 0),0).x +
             texelFetch(inputMap, ivec2(gl_FragCoord) + ivec2( 1, 0),0).x +
             texelFetch(inputMap, ivec2(gl_FragCoord) + ivec2( 0, -1),0).x +
             texelFetch(inputMap, ivec2(gl_FragCoord) + ivec2( 0,  1),0).x -
             4 * u;

 float um = u + v * deltat / 2.0;
 float vm = v + sqrc * d2u * deltat / 2.0;

 outColor = vec4(u, v, um, vm);
}
```

Hullám egyenlet

```
uniform sampler2D inputMap;
out vec4 outColor;

const float deltat = 0.0001;
const float deltax = 0.01;
const float sqrc = 20.0;

void main(void){
 vec4 data = texelFetch(inputMap, ivec2(gl_FragCoord), 0);

 float u = data.x;  float v = data.y;
 float um = data.z; float vm = data.w;

 float d2u = texelFetch(inputMap, ivec2(gl_FragCoord) + ivec2(-1,  0),0).z +
             texelFetch(inputMap, ivec2(gl_FragCoord) + ivec2( 1,  0),0).z +
             texelFetch(inputMap, ivec2(gl_FragCoord) + ivec2( 0, -1),0).z +
             texelFetch(inputMap, ivec2(gl_FragCoord) + ivec2( 0,  1),0).z -
             4 * u;

 u = u + vm * deltat;
 v = v + sqrc * d2u * deltat;

 outColor = vec4(u, v, 0.0, 0.0);
}
```

Hullám egyenlet