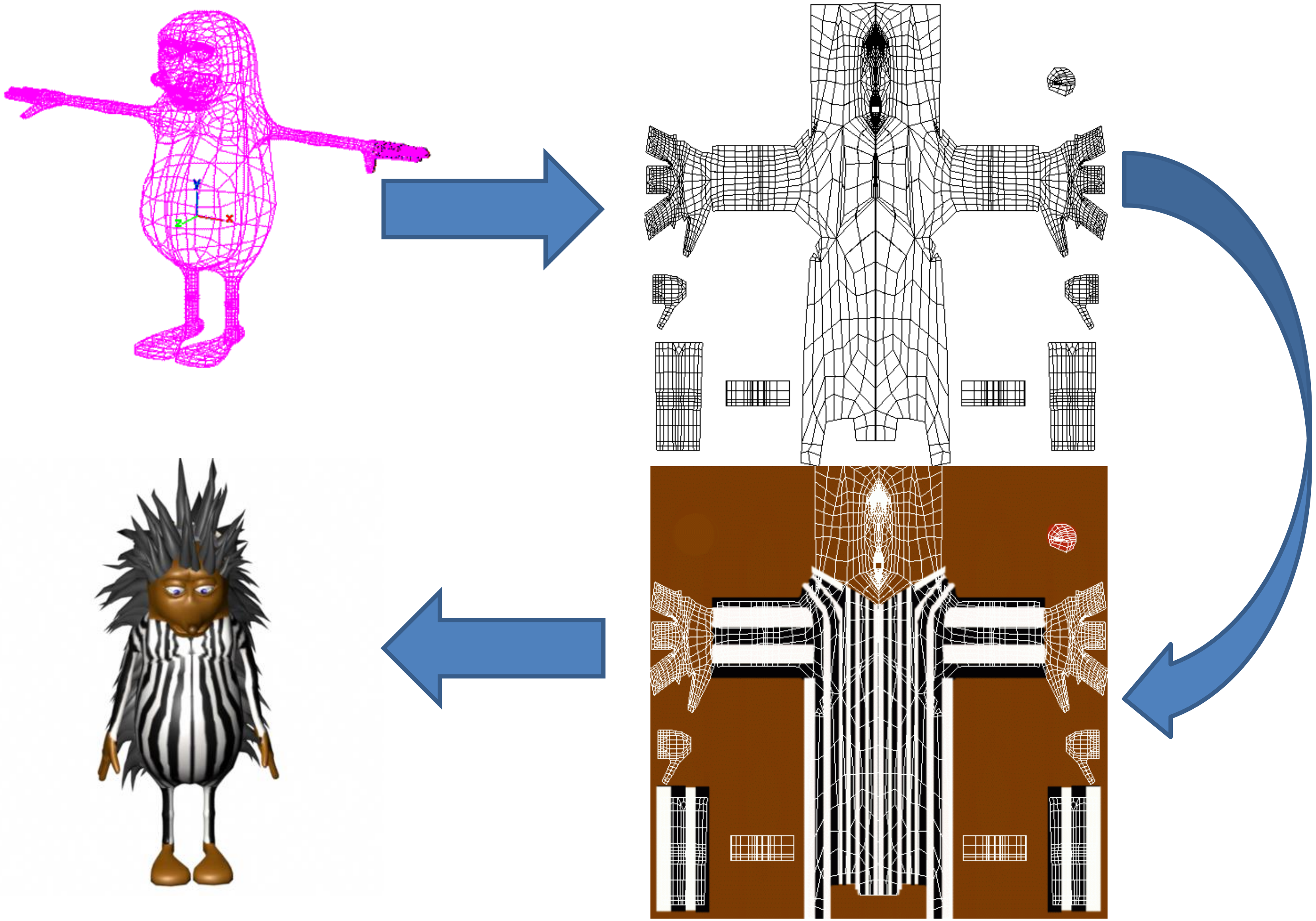
“Everything must be made as simple as possible. But not simpler.”
Albert Einstein

2D textúrázás

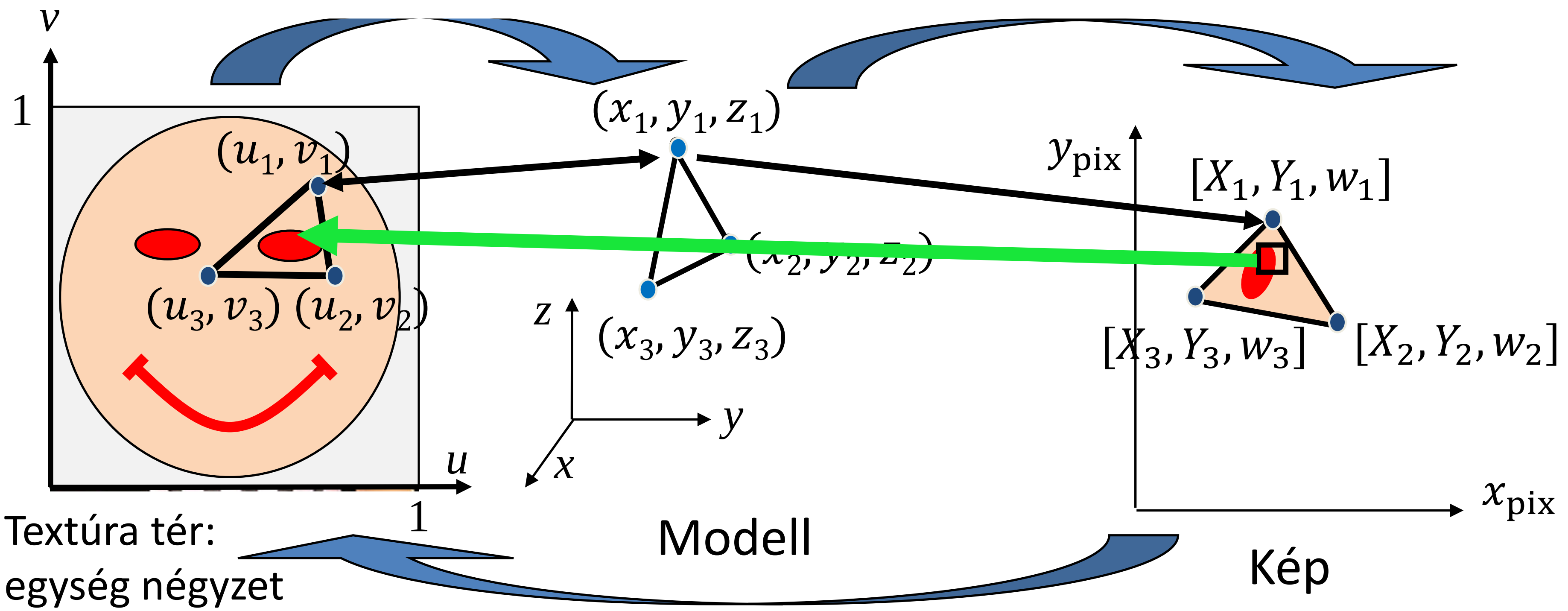
Szirmay-Kalos László

2D textúrázás



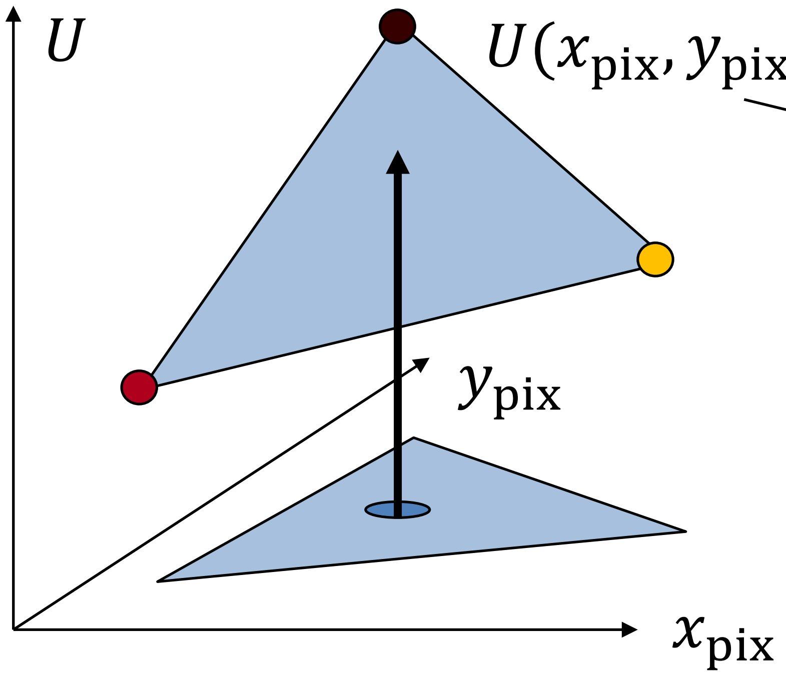
2D Textúrázás

$$[X, Y, w] = [u, v, 1] \cdot \boldsymbol{P}$$
$$(x_{\text{pix}}, y_{\text{pix}}) = [X/w, Y/w]$$

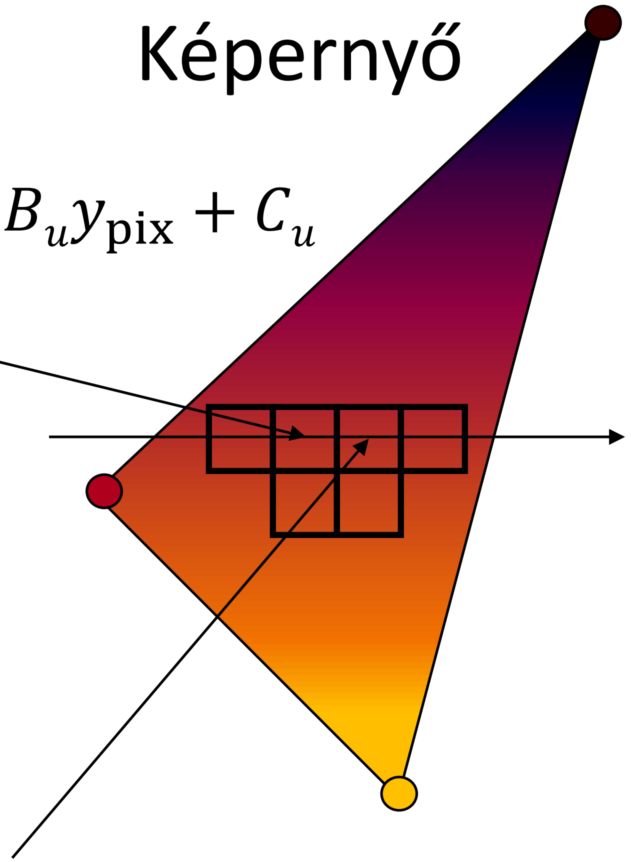


$$[u/w, v/w, 1/w] = [x_{\text{pix}}, y_{\text{pix}}, 1] \cdot \boldsymbol{P}^{-1}$$
$$[U, \quad V, \quad W] \rightarrow u = U/W, \; v = V/W$$

Lineáris interpoláció

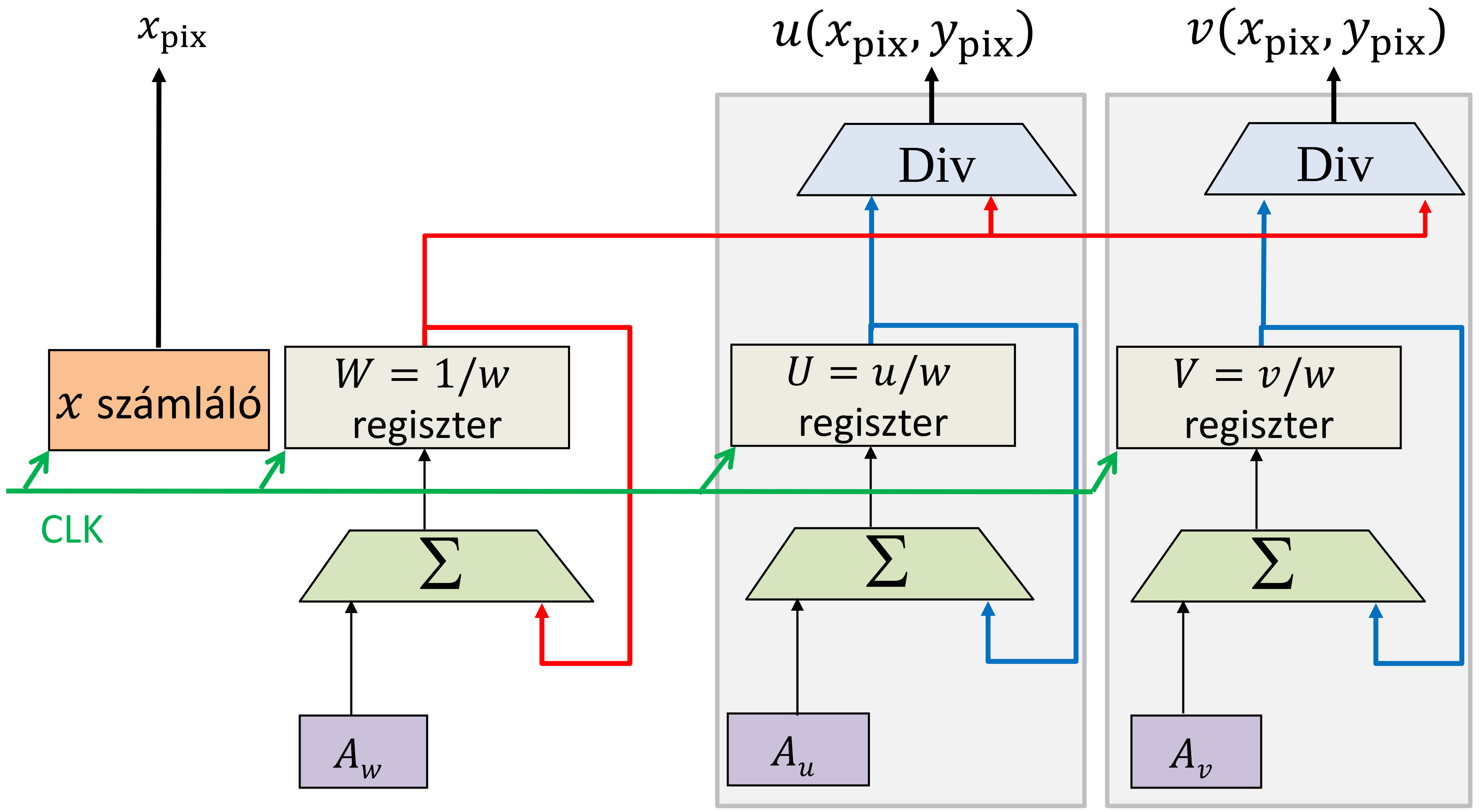


$$U(x_{\mathrm{pix}}, y_{\mathrm{pix}}) = A_u x_{\mathrm{pix}} + B_u y_{\mathrm{pix}} + C_u$$

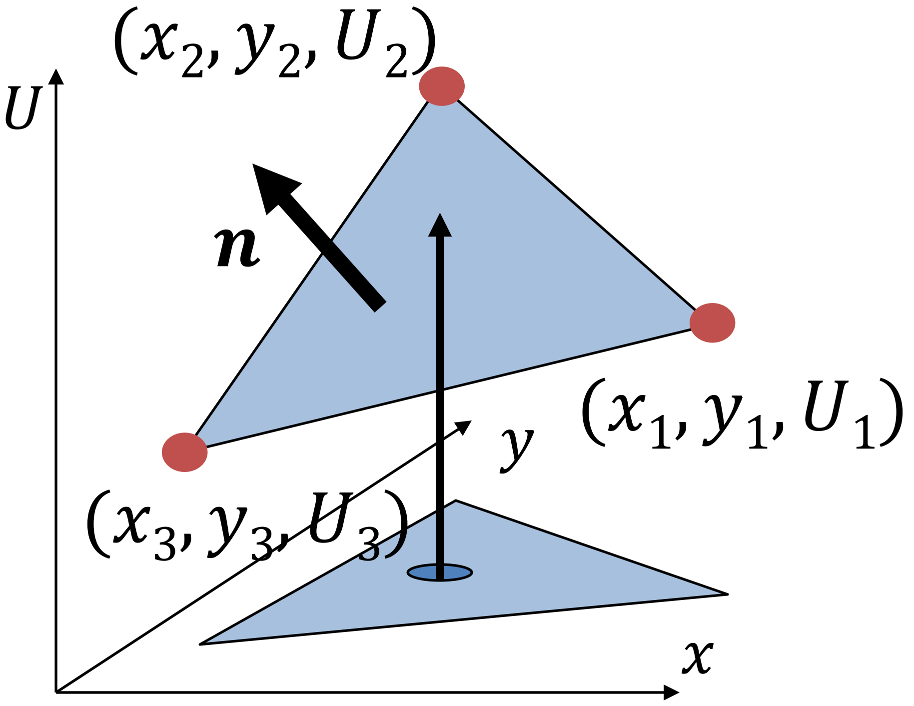


$$U(x_{\mathrm{pix}} + 1, y_{\mathrm{pix}}) = U(x_{\mathrm{pix}}, y_{\mathrm{pix}}) + A_u$$

Interpolációs hardver



Triangle setup



$$U(x, y) = Ax + By + C$$
$$n_x x + n_y y + n_u U + d = 0$$

$$U_1 = Ax_1 + By_1 + C$$
$$U_2 = Ax_2 + By_2 + C$$
$$U_3 = Ax_3 + By_3 + C$$

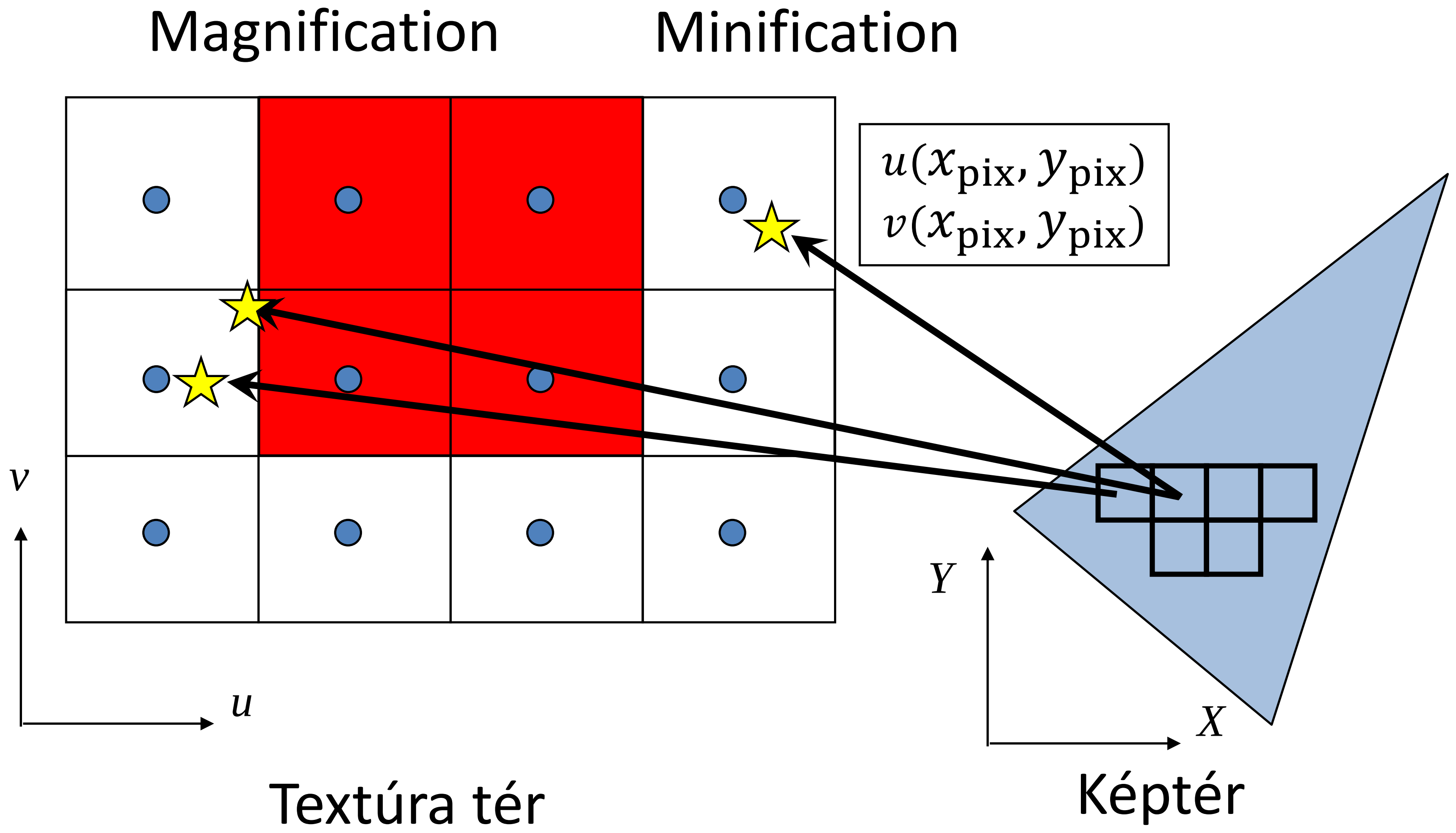
$$A = \frac{(U_3 - U_1)(Y_2 - Y_1) - (Y_3 - Y_1)(U_2 - U_1)}{(X_3 - X_1)(Y_2 - Y_1) - (Y_3 - Y_1)(X_2 - X_1)}$$

n_x

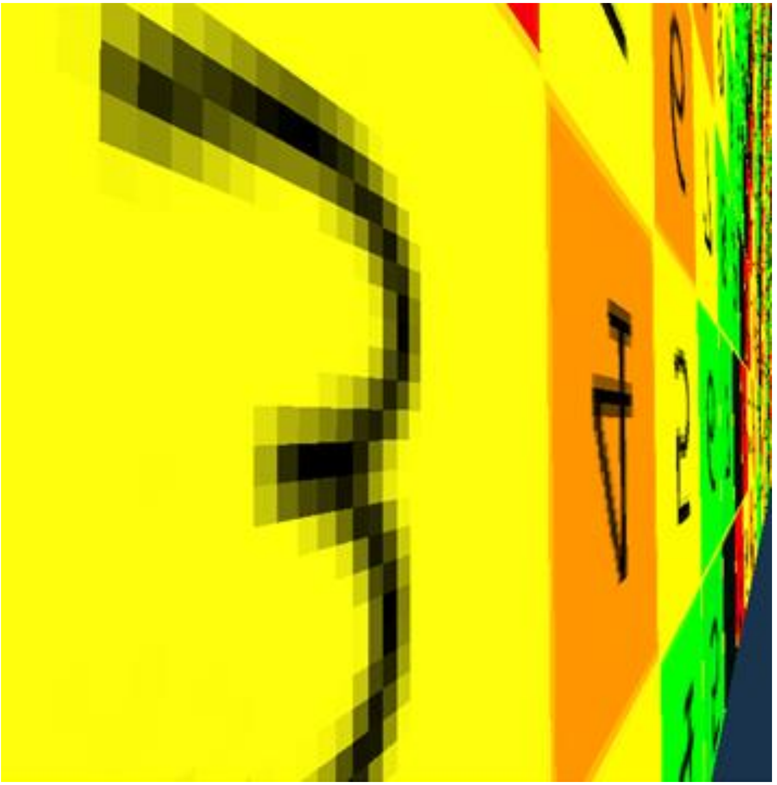
$-n_u$

$$\boldsymbol{n} = (X_2 - X_1, Y_2 - Y_1, U_2 - U_1) \times (X_3 - X_1, Y_3 - Y_1, U_3 - U_1)$$

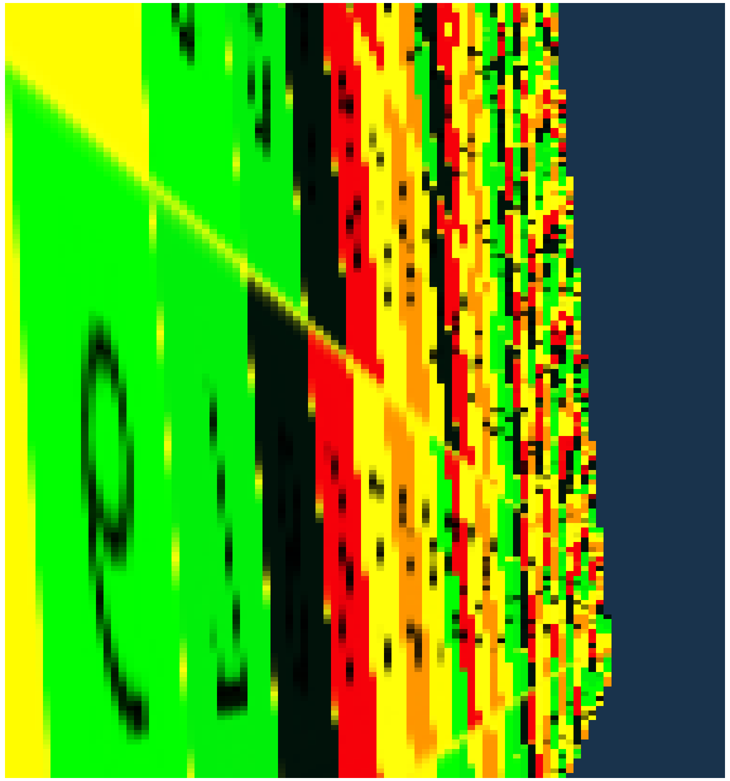
Textúra szűrés (GL_NEAREST)



Textúratér és képtér kapcsolata



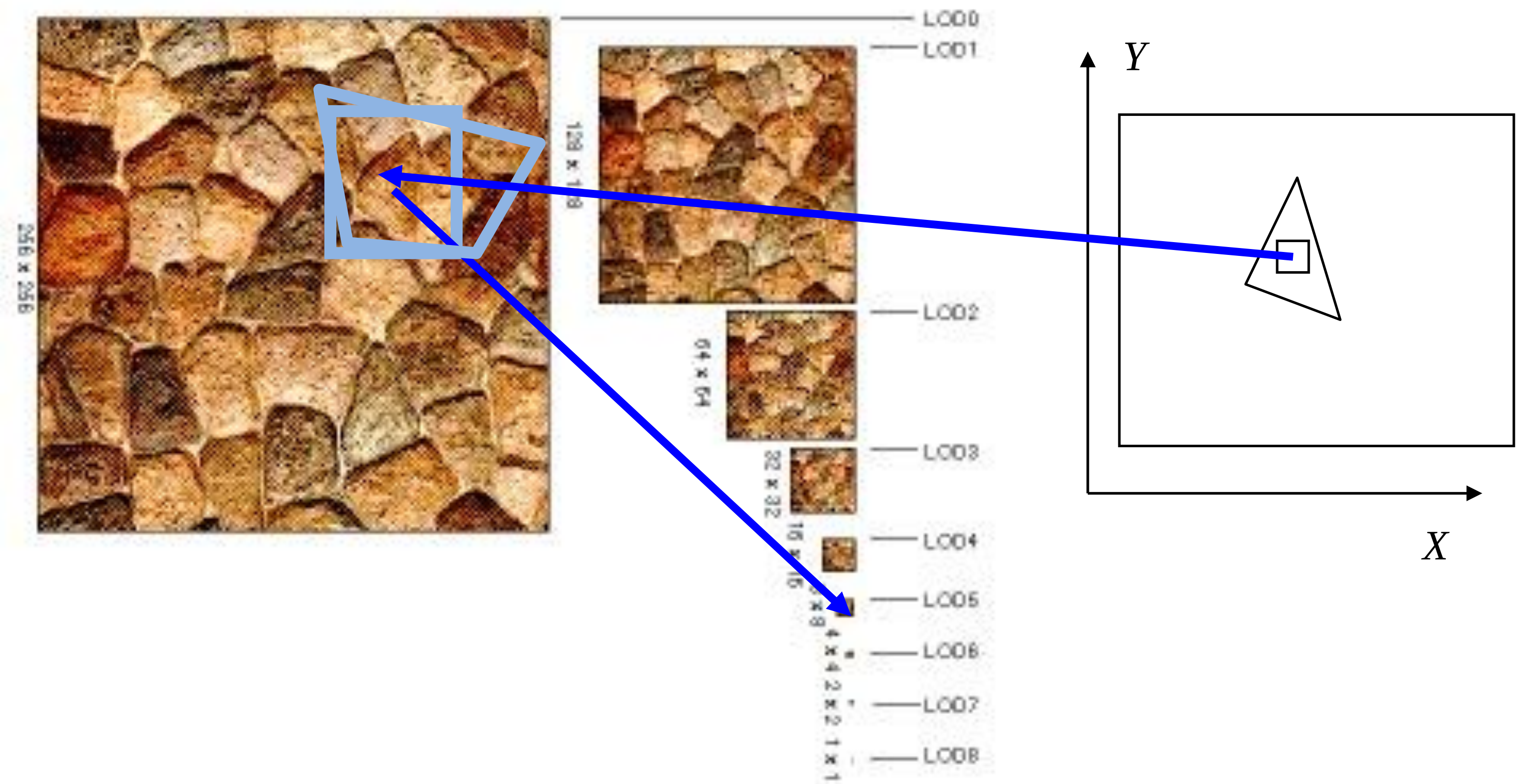
Magnification



Minification

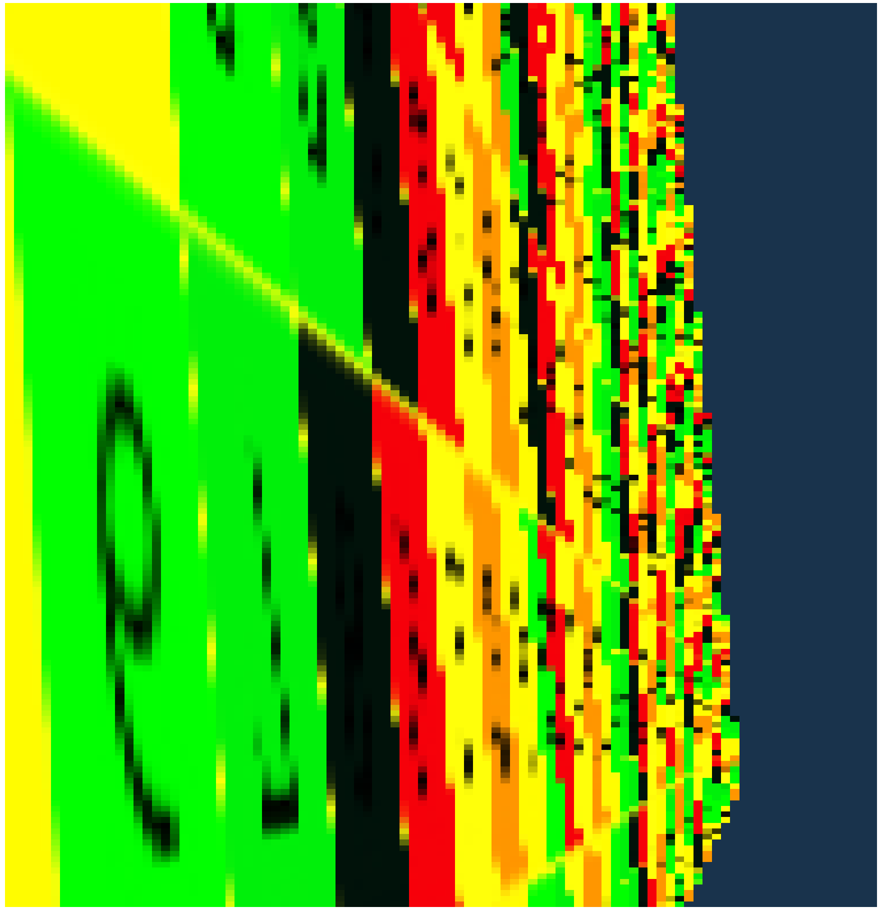
```
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
```

Mip-map (multum in parvo): Minification-ra jó

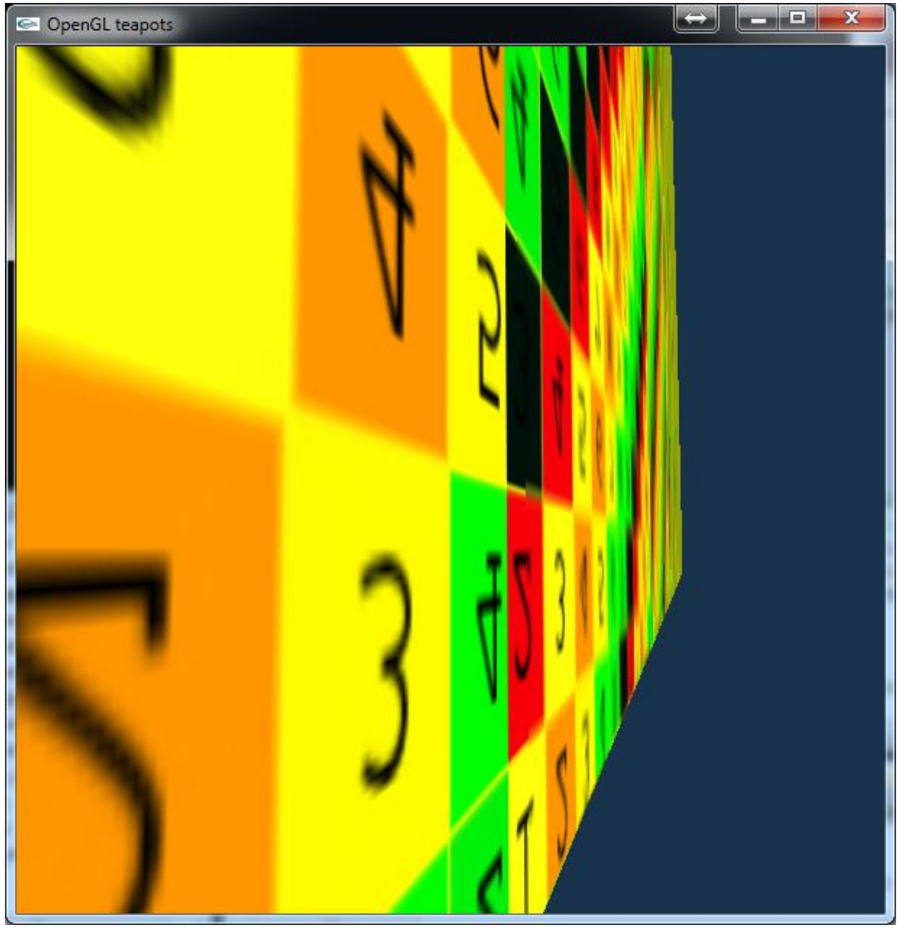



```
a) glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_NEAREST);
                                                            // Mip-mapping

b) glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
                                                         // Tri-linear filtering
```

Mip-map (GL_LINEAR_MIPMAP_...)

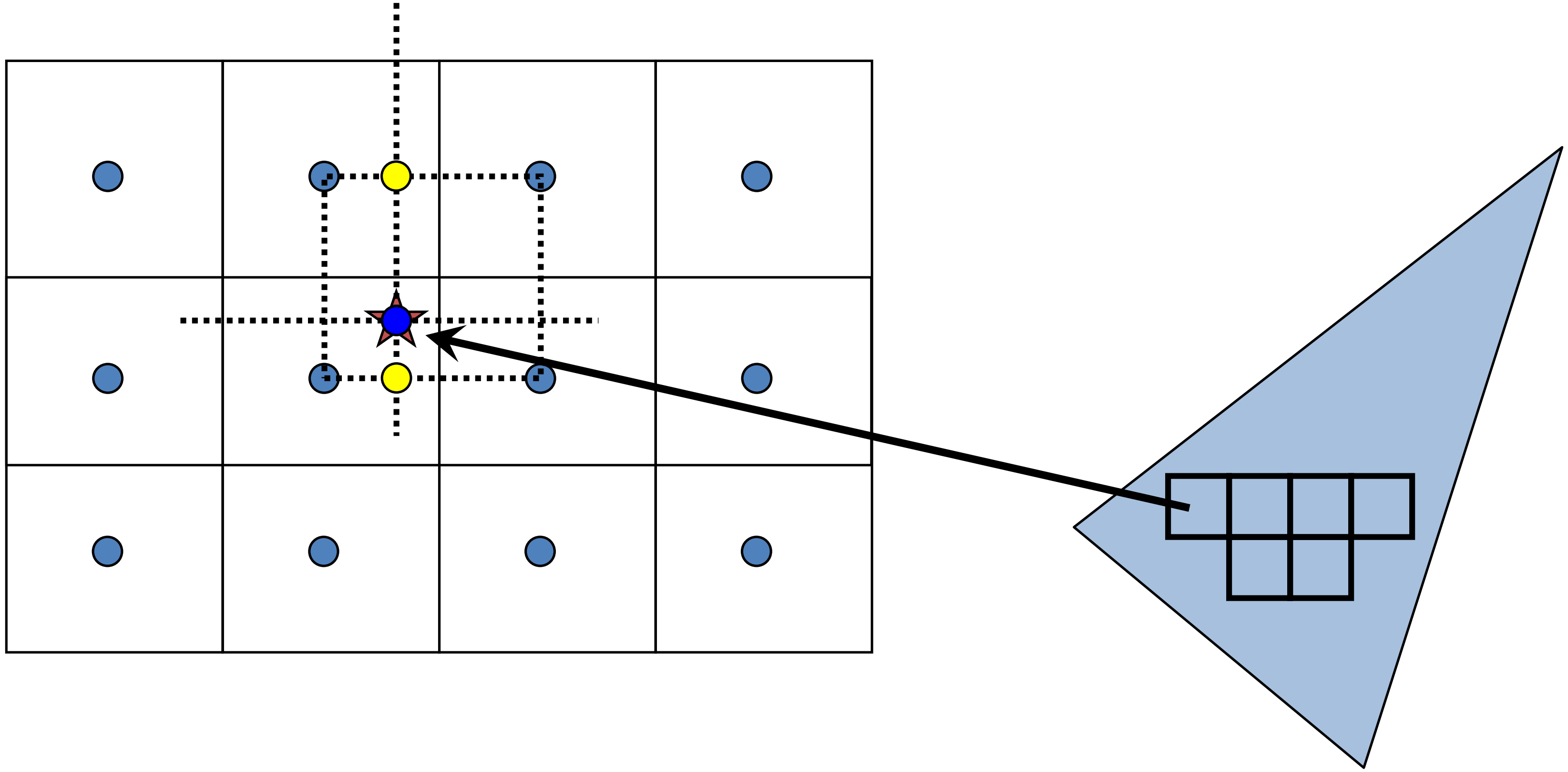


GL_NEAREST



GL_LINEAR_MIPMAP_NEAREST

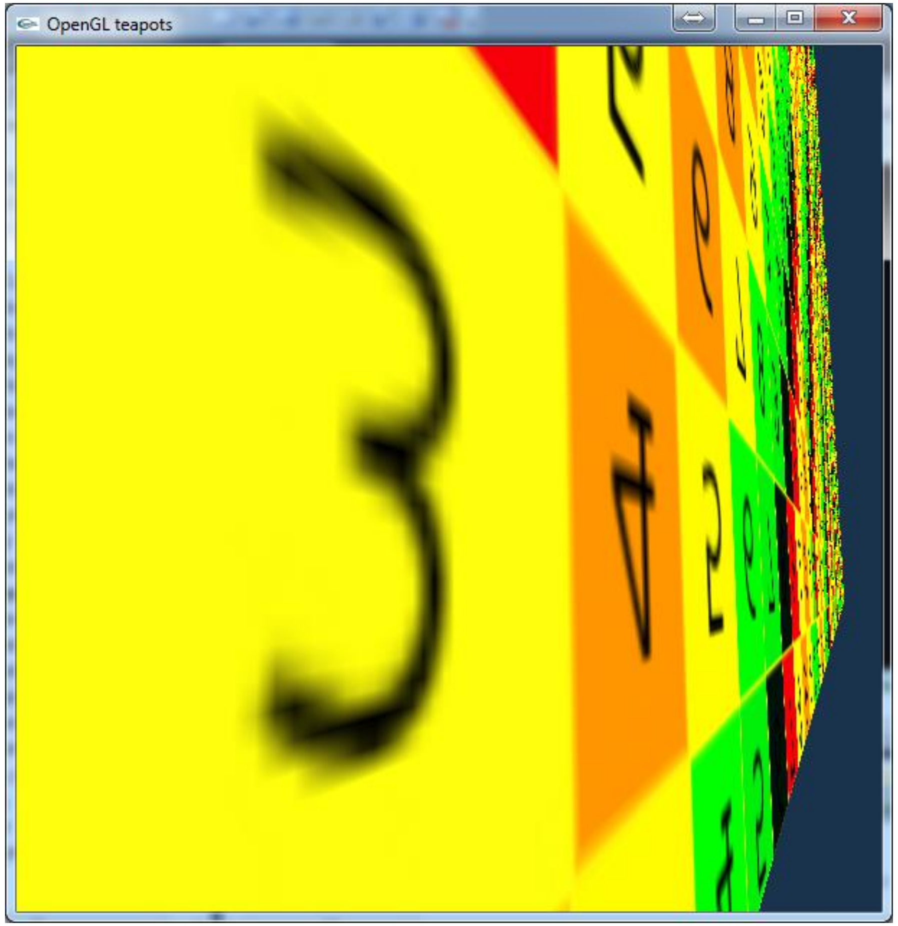
Bi-linear textúra szűrés (GL_LINEAR)
Magnification-ra igazán jó



```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

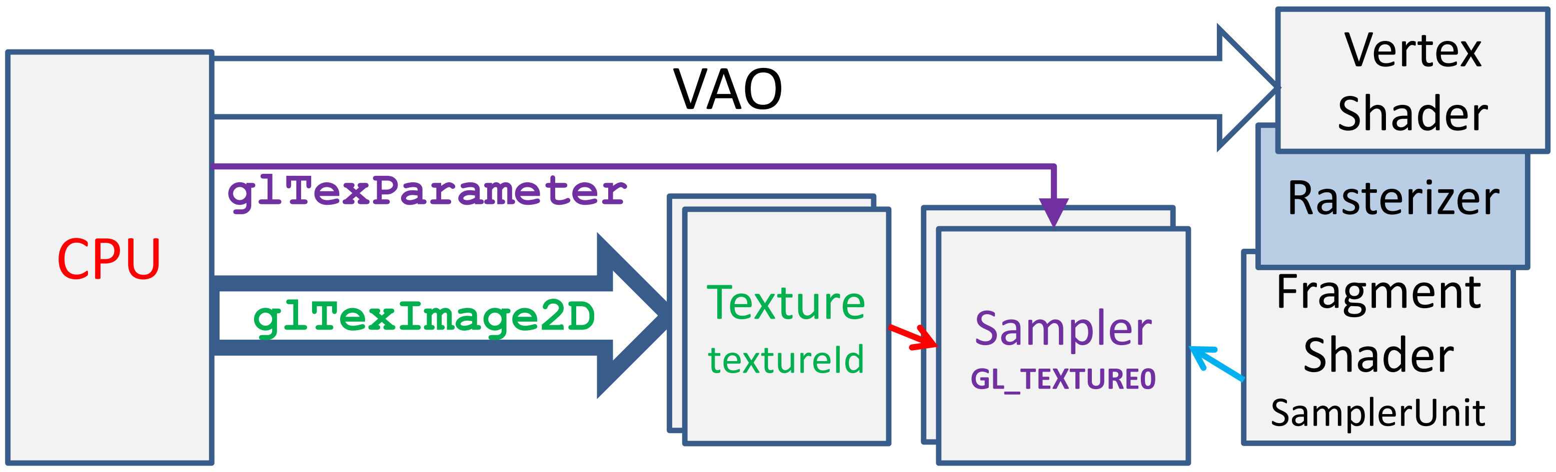
Bi-linear filtering (GL_LINEAR)

GL_NEAREST

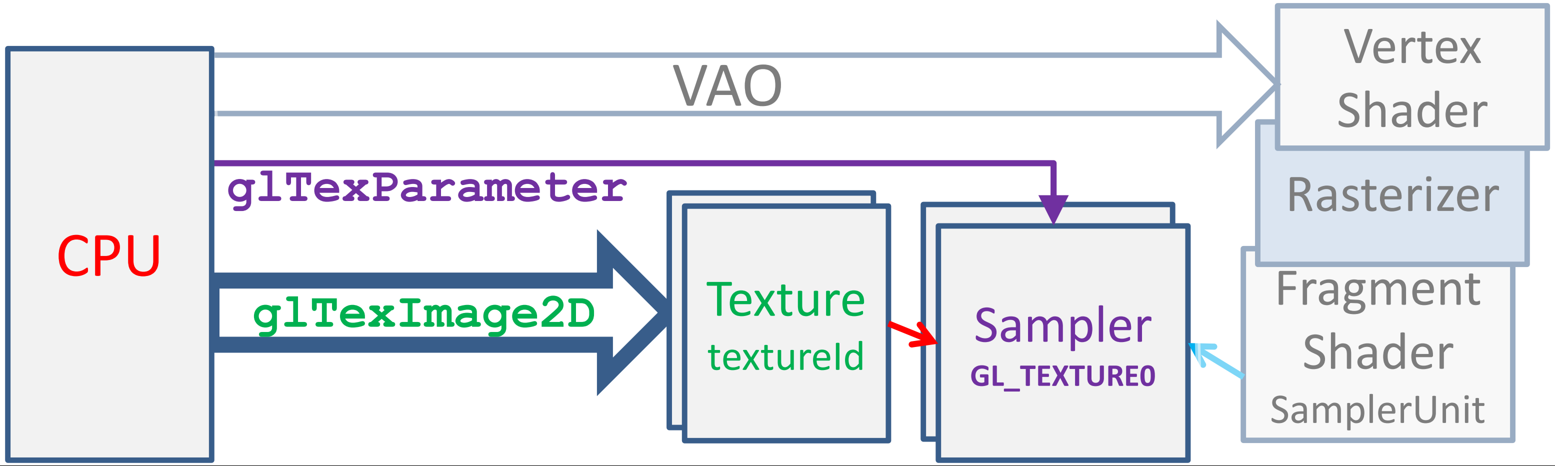


GL_LINEAR

Textúrázás a GPU-n



Textúrázás 1: Textúra GPU-ra töltése




```
unsigned int textureId;

void UploadTexture(int width, int height, vector<vec4>& image) {
  glGenTextures(1, &textureId);
  glBindTexture(GL_TEXTURE_2D, textureId);     // binding

  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
               GL_RGBA, GL_FLOAT, &image[0]); //Texture -> GPU

  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
}
```

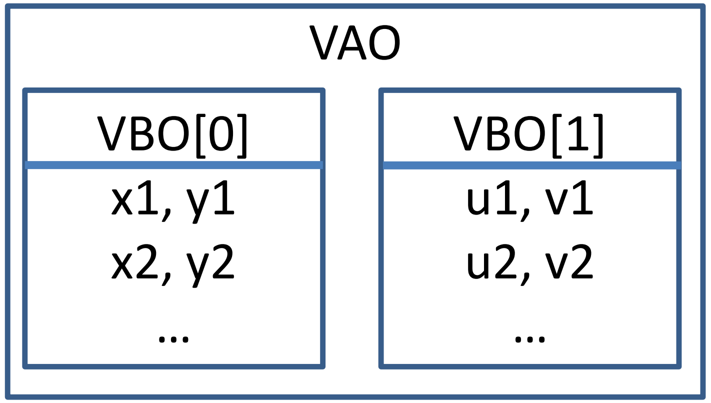
Mip-map szint (0)

célformátum (GL_RGBA)

Border (0)

forrás (GL_RGBA, GL_FLOAT, &image[0])

Textúrázás 2: Objektumok felszerelése textúra koordinátákkal




```
glGenVertexArrays(1, &vao);
glBindVertexArray(vao);

glGenBuffers(2, vbo);// Generate 2 vertex buffer objects

// vertex coordinates: vbo[0] -> Attrib Array 0 -> vertices
glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
float vtxs[] = {x1, y1, x2, y2, …};
glBufferData(GL_ARRAY_BUFFER, sizeof(vtxs),vtxs, GL_STATIC_DRAW);
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 0, NULL);

// vertex coordinates: vbo[1] -> Attrib Array 1 -> uvs
glBindBuffer(GL_ARRAY_BUFFER, vbo[1]);
float uvs[] = {u1, v1, u2, v2, …};
glBufferData(GL_ARRAY_BUFFER, sizeof(uvs), uvs, GL_STATIC_DRAW);
glEnableVertexAttribArray(1);
glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 0, NULL);
```

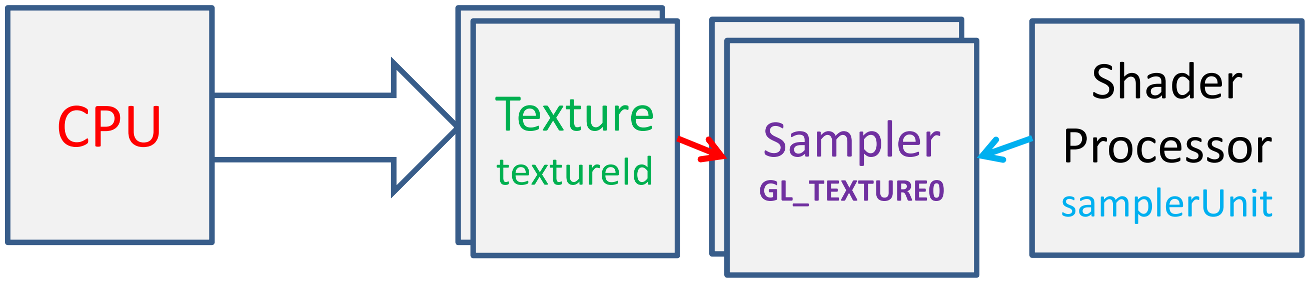
Textúrázás 3: Vertex és Pixel Shader

```
layout(location = 0) in vec2 vtxPos;
layout(location = 1) in vec2 vtxUV;

out vec2 texcoord;

void main() {
    gl_Position = vec4(vtxPos, 0, 1) * MVP;
    texcoord = vtxUV;
    …
}
```

Raszterizáció
Interpoláció

```
uniform sampler2D samplerUnit;
in vec2 texcoord;
out vec4 fragmentColor;

void main() {
   fragmentColor = texture(samplerUnit, texcoord);
}
```

Textúrázás 4: Aktív textúra és sampler




```
unsigned int textureId;

void Draw( ) {
  int sampler = 0; // which sampler unit should be used

  int location = glGetUniformLocation(shaderProg, "samplerUnit");
  glUniform1i(location, sampler);

  glActiveTexture(GL_TEXTURE0 + sampler);  // = GL_TEXTURE0
  glBindTexture(GL_TEXTURE_2D, textureId);

  glBindVertexArray(vao);
  glDrawArrays(GL_TRIANGLES, 0, nVtx);
}
```

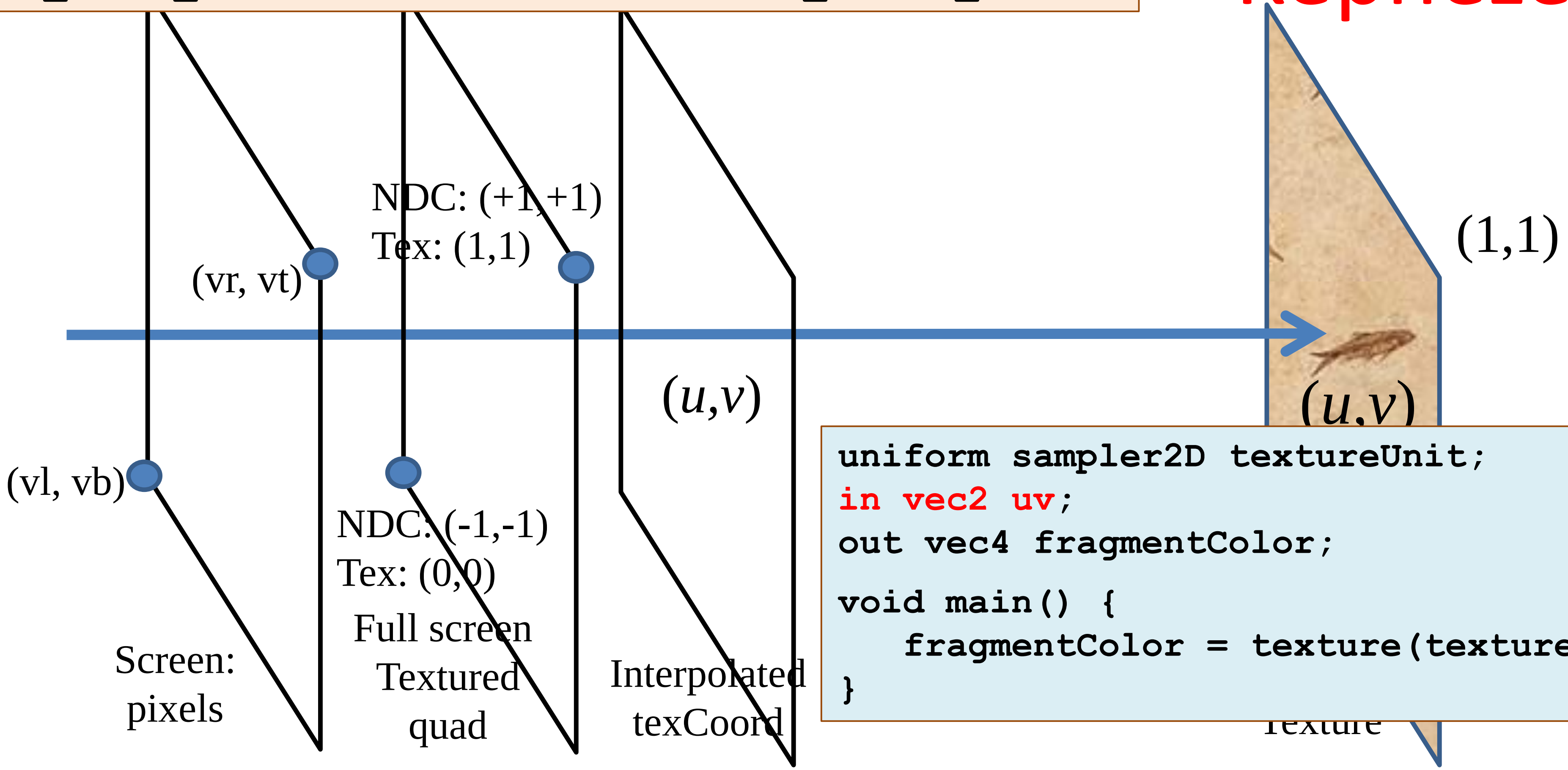
> *“The message of this lecture is that black holes ain’t as black as they are painted. So if you feel you are in a black hole, don’t give up – there’s a way out.”*
>
> *Stephen Hawking*

Képnézegető és nemlineáris 2D képeffektusok

Szirmay-Kalos László

Képnézegető

```
float vtx = {-1,-1, 1,-1, 1,1, -1, 1};
glBufferData(GL_ARRAY_BUFFER, sizeof(vtx), vtx, GL_STATIC_DRAW);
```

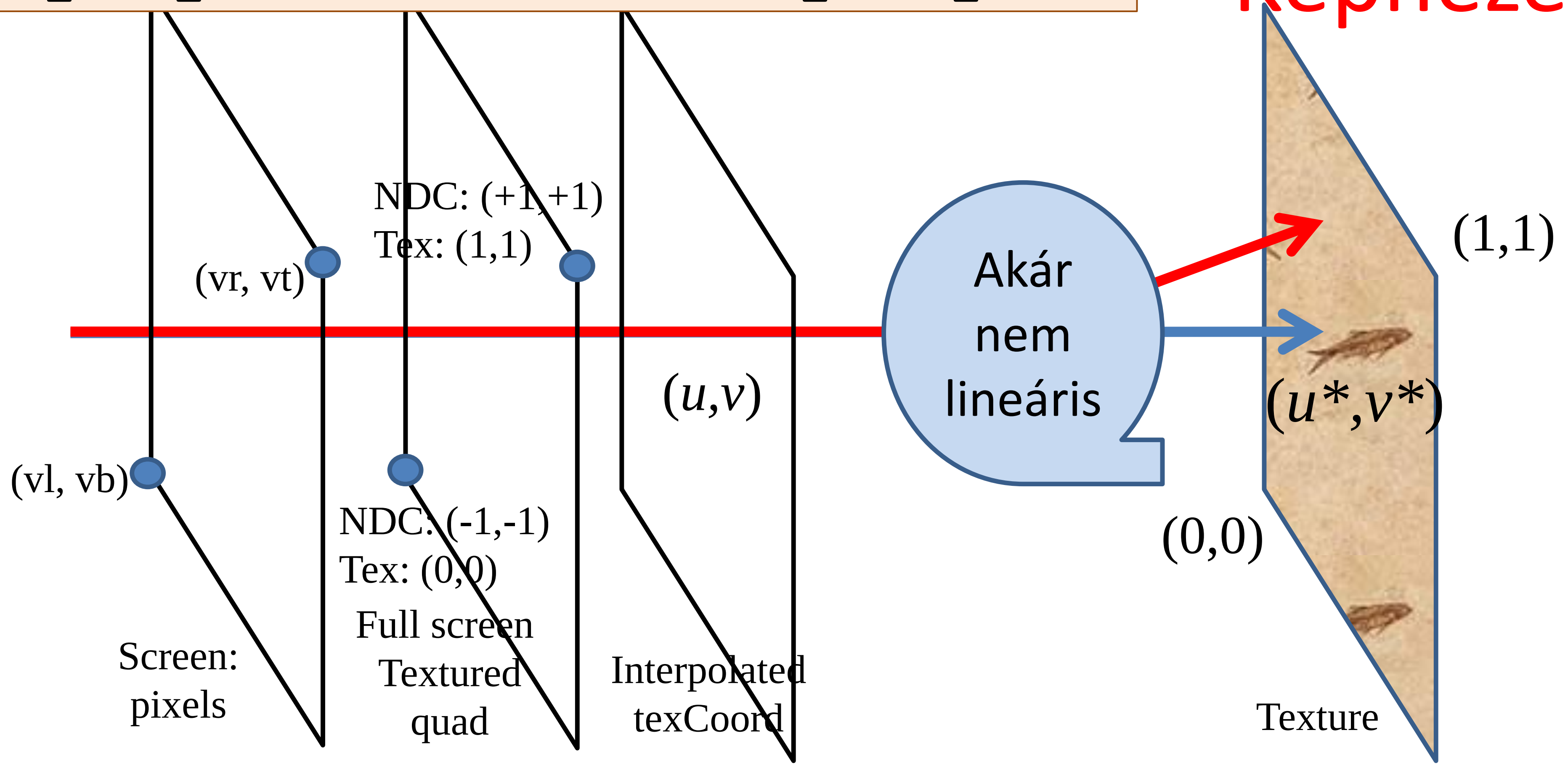



```
uniform sampler2D textureUnit;
in vec2 uv;
out vec4 fragmentColor;

void main() {
   fragmentColor = texture(textureUnit, uv);
}
```

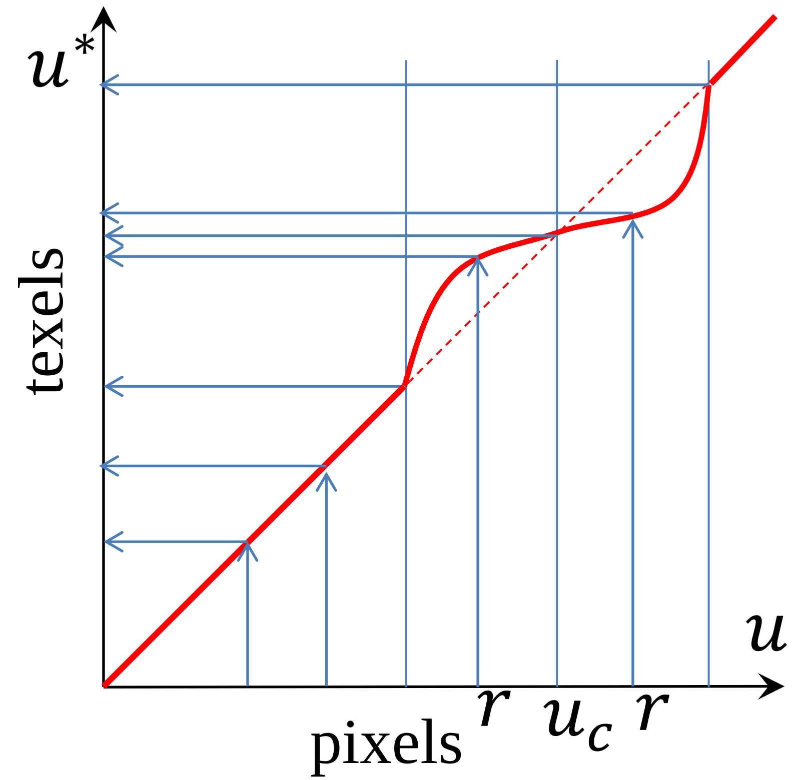
```
layout(location = 0) in vec2 vp; // Attrib Array 0
out vec2 uv;// output attribute

void main() {
   uv = (vp + vec2(1, 1)) / 2; // clipping to texture space
   gl_Position = vec4(vp.x, vp.y, 0, 1);
}
```

```
float vtx = {-1,-1, 1,-1, 1,1, -1, 1};
glBufferData(GL_ARRAY_BUFFER, sizeof(vtx), vtx, GL_STATIC_DRAW);
```

Képnézegető




```
layout(location = 0) in vec2 vp; // Attrib Array 0
out vec2 uv;// output attribute
void main() {
   uv = (vp + vec2(1, 1)) / 2; // clipping to texture space
   gl_Position = vec4(vp.x, vp.y, 0, 1);
}
```



Mágikus lencse

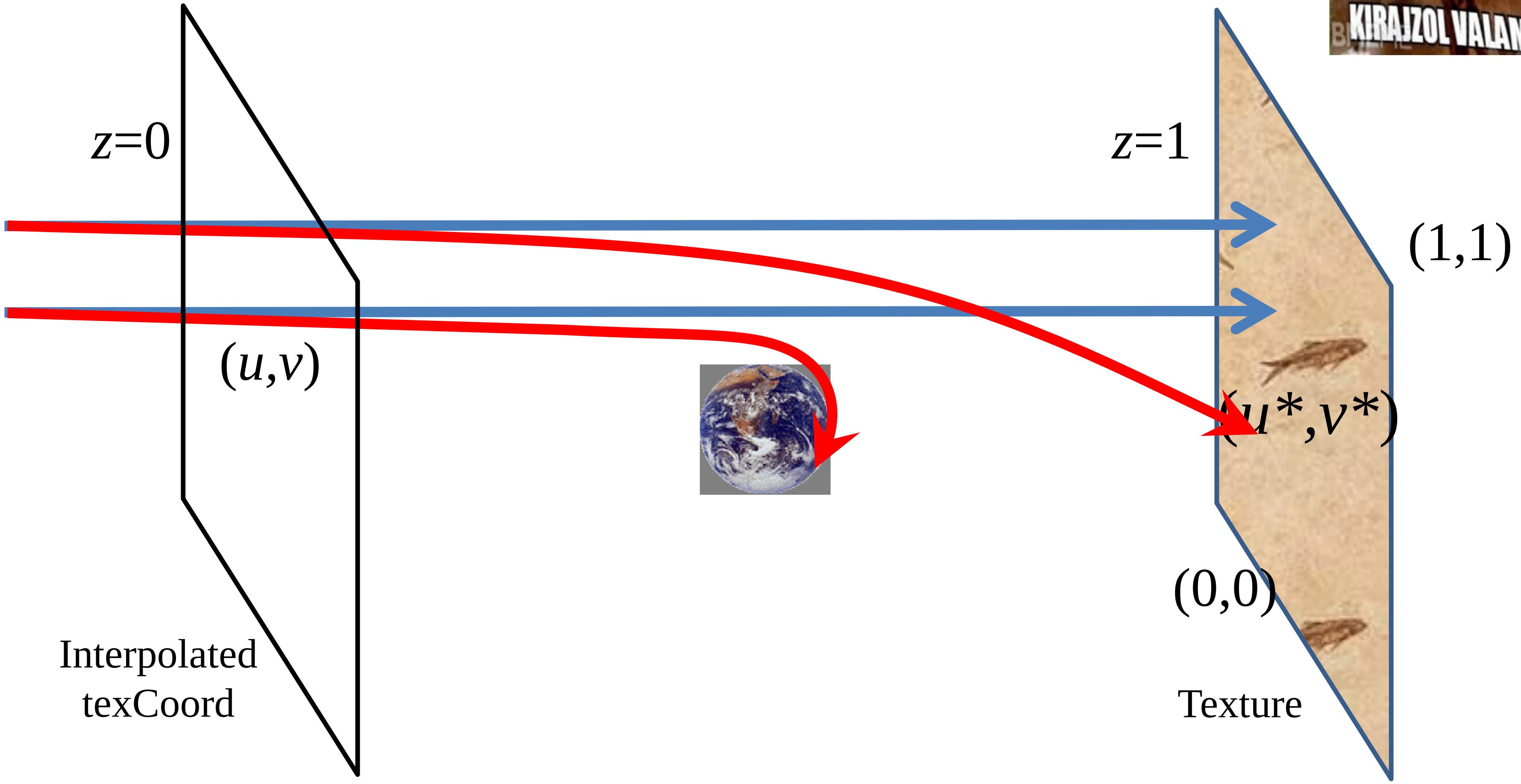
$$u^* = \frac{(u-u_c)^3}{r^2} + u_c \quad \text{if } |u-u_c| < r$$

```
uniform sampler2D textureUnit;
uniform vec2 uvc; // cursor position in texture space

in vec2 uv;      // interpolated texture coordinates
out vec4 fragmentColor;

void main() {
   const float r2 = 0.05f;
   float d2 = dot(uv - uvc, uv - uvc);
   vec2 tuv = (d2 < r2) ? (uv - uvc) * d2 / r2 + uvc : uv;
   fragmentColor = texture(textureUnit, tuv);
}
```

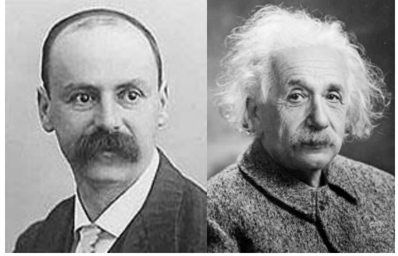
Örvény: Swirl

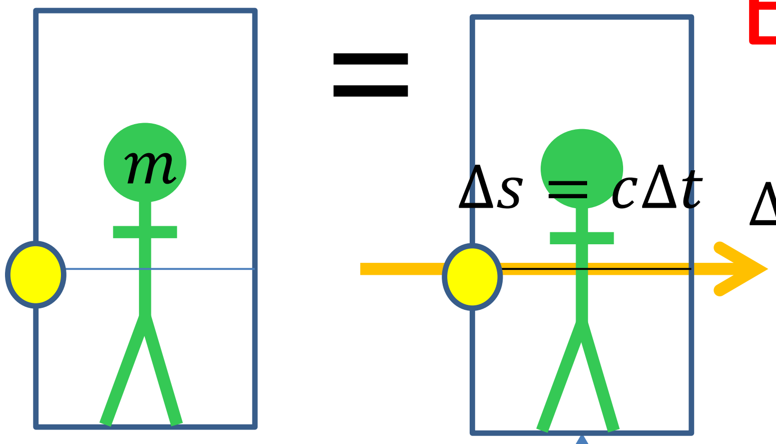

```
uniform sampler2D textureUnit;
uniform vec2 uvc; // cursor position in texture space

in vec2 uv;     // interpolated texture coordinates
out vec4 fragmentColor;

void main() {
   const float a = 8, alpha = 15;
   float ang = a * exp( -alpha * length(uv - uvc) );

   mat2 rotMat = mat2( cos(ang), sin(ang),
                      -sin(ang), cos(ang) );

   vec2 tuv = (uv - uvc) * rotMat + uvc;
   fragmentColor = texture(textureUnit, tuv);
}
```

Gravitáció (fekete lyukak)



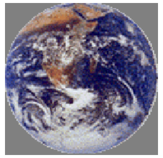
Ekvivalencia elv





$$\Delta d = \frac{g}{2}(\Delta t)^2 = \frac{fM}{2r^2c^2}(\Delta s)^2 = \frac{r_0}{4r^2}(\Delta s)^2$$

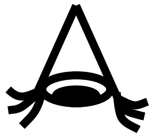
$\frac{r_0}{2}$: Schwarzschild rádiusz

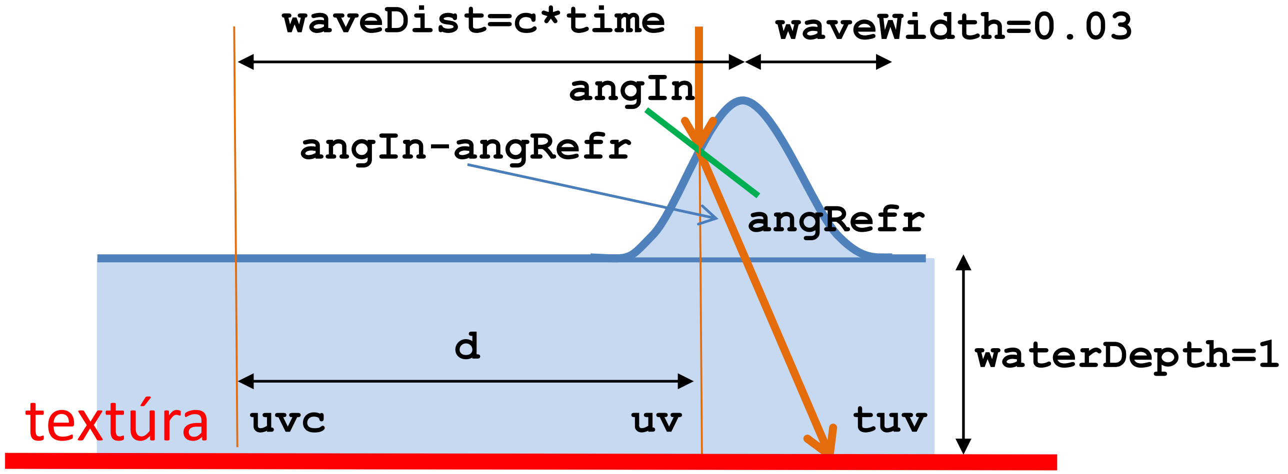


$$g = \frac{fM}{r^2}$$


```
void main() {
   const float r0 = 0.09f, ds = 0.001f;
   vec3 p = vec3(uv,0), dir = vec3(0,0,1), blackhole = vec3(uvc,0.5f);
   float r2 = dot(blackhole - p, blackhole - p);
   while (p.z < 1 && r2 > r0 * r0) {
      p += dir * ds;
      r2 = dot(blackhole - p, blackhole - p);
      vec3 gDir = (blackhole - p)/sqrt(r2); // gravity direction
      dir = normalize(dir * ds + gDir * r0 / r2 / 4 * ds * ds);
   }
   if (p.z >= 1) fragmentColor = texture(textureUnit,vec2(p.x,p.y));
   else          fragmentColor = vec4(0, 0, 0, 1);
}
```

Hullám: Wave






```
uniform float time;
const float PI = 3.14159265, n = 1.33, c = 0.1, aMax = 0.1;

void main() {
   float d = length(uv - uvc), waveDist = c * time;
   if (abs(d - waveDist) < waveWidth) {
      float angIn = aMax/waveDist * sin((waveDist-d)/waveWidth*PI);
      float angRefr = asin(sin(angIn)/n);
      vec2 dir = (uv - uvc)/d;
      vec2 tuv = uv + dir * tan(angIn - angRefr) * waterDepth;
      fragmentColor = texture(textureUnit, tuv);
   } else {
      fragmentColor = texture(textureUnit, uv);
   }
}
```