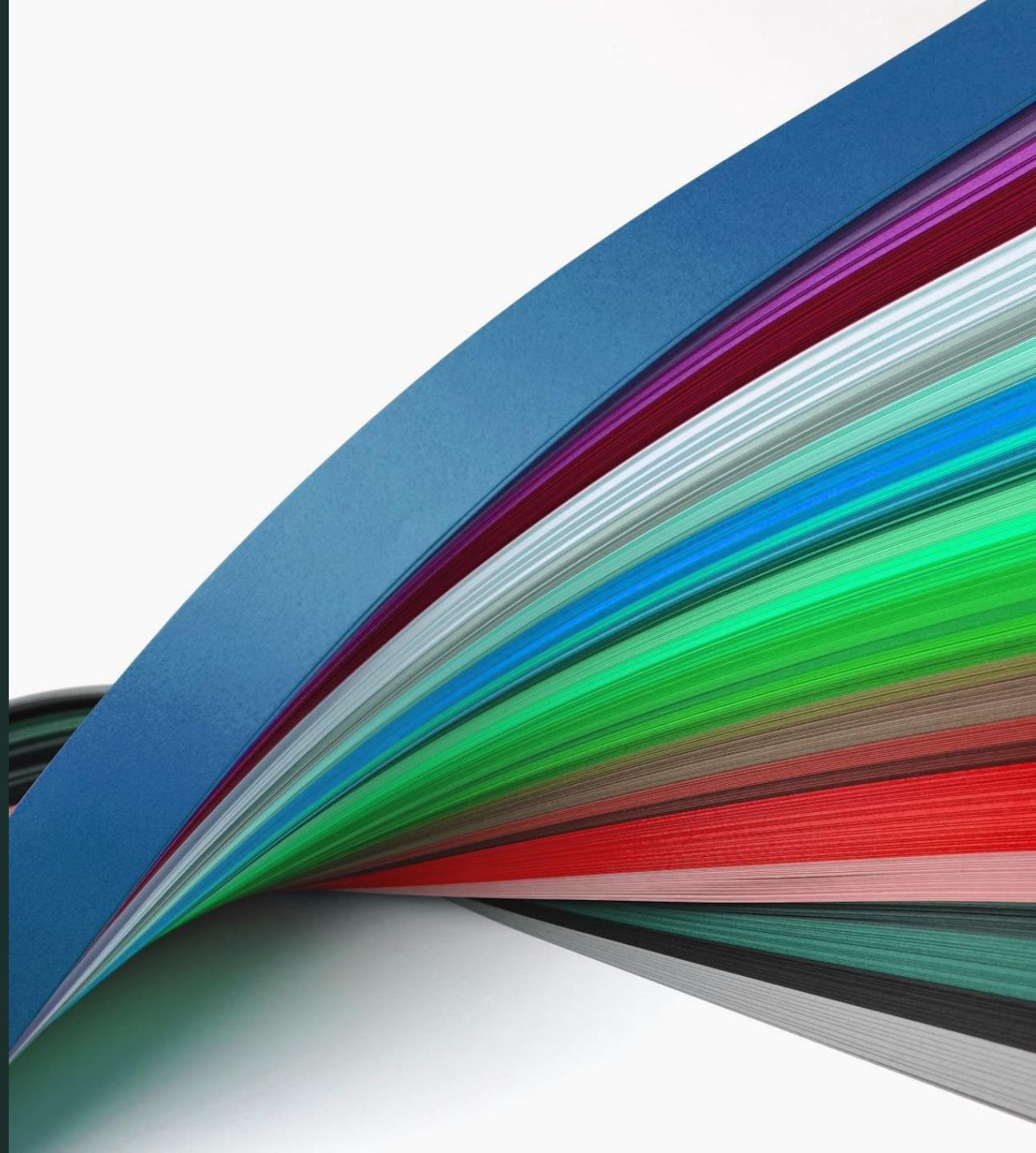


XR-WWW1

Webudvikling



Plan



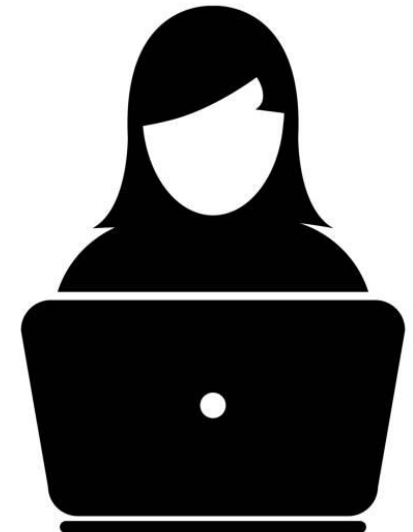
1. HTML
2. CSS
3. Responsive Design
4. Individuel aflevering
5. JavaScript: Basics
6. JavaScript: JSON & Fetch
7. JavaScript: Sessions & Cookies
8. Gruppe aflevering

9. WebGL Simpel Trekant

10. WebGL Geometri & Rotation

➡ **11. WebGL Tekstur & Belysning**

12. Afsluttende aflevering



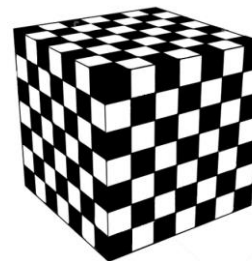
Opsummering



- Dynamisk Kode
- Definer geometri ved dimensioner
- AddTriangle() & AddQuad()
- CreateTriangle() & CreateQuad()
- Tilføj valg af geometri der skal vises
- Generering af UI til geometri
- Roterig af geometri
- Event Listener: mousemove
- Musekoordinater offsets: angle (array)
- Uniform Location: vec4
- Vertex Shader: Matrix multiplikation

Praksis:

1. Implementering af vist kode
2. Implementering af simpel 3D kasse
3. Implementering af subdivision + UI



Width	<input type="text" value="1.0"/>	Divs X	<input type="text" value="6"/>
Height	<input type="text" value="1.0"/>	Divs Y	<input type="text" value="6"/>
Depth	<input type="text" value="1.0"/>	Divs Z	<input type="text" value="6"/>

WebGL



**Hvordan sættes en tekstur
på en 3D overflade?**

WebGL



**Hvordan sættes en tekstur
på en 3D overflade?**

<https://img-new.cgtrader.com/items/3793556/954b90aece/wooden-box-3d-model-fbx-blend.jpg>

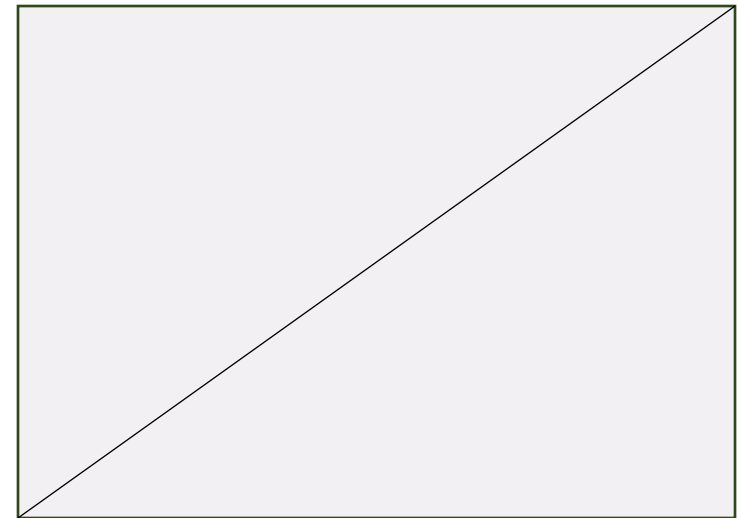
WebGL



UV Koordinater:

$(0.0, 1.0)$

$(1.0, 1.0)$



$(0.0, 0.0)$

$(1.0, 0.0)$

WebGL



En vec2 UV attribut tilføjes i vertex shaderen.

Ligesom attributten Color, sendes den videre til fragment shaderen.

```
attribute vec3 Pos;
attribute vec3 Color;
attribute vec2 UV;
uniform vec4 Angle;
varying vec3 vertexColor;
varying vec2 uv;
void main() {
    vec2 a = vec2(sin(Angle.x), cos(Angle.x));
    mat4 matX = mat4(vec4(1.0, 0.0, 0.0, 0.0),
                    vec4(0.0, a.y, a.x, 0.0),
                    vec4(0.0, -a.x, a.y, 0.0),
                    vec4(0.0, 0.0, 0.0, 1.0));
    a = vec2(sin(Angle.y), sin(Angle.y));
    mat4 matY = mat4(vec4(a.y, 0.0, -a.x, 0.0),
                    vec4(0.0, 1.0, 0.0, 0.0),
                    vec4(a.x, 0.0, a.y, 0.0),
                    vec4(0.0, 0.0, 0.0, 1.0));
    gl_Position = matY * matX * vec4(Pos, 1.0);
    vertexColor = Color;
    uv = UV;
}
```



WebGL



En sampler2D uniform tilføjes i fragment shaderen. Dette er en reference til teksturen.

Yderlig tilføjes en uniform vec4 som repræsenterer Display (options).

UV koordinaterne modtages som en varying, fra vertex shaderen.

Funktionen **texture2D**, sampler en pixel fra teksturen, ved hjælp af uv koordinaterne.

```
precision mediump float;
uniform sampler2D Texture;
uniform vec4 Display;
varying vec3 vertexColor;
varying vec2 uv;
void main()
{
    float p = abs(Display.w);
    vec3 texture = texture2D(Texture, uv).rgb;
    vec3 color = vertexColor;
    gl_FragColor = vec4(mix(color, texture, p), 1.0);
}
```



WebGL



Tilføj nye variabler som globale variabler.

displayOptions = [R 0.0, G 0.0, B 0.0, 0.0];

Lyskildens farve ↑

Vis tekstur

```
// Global variable
var textureGL = 0; // Uniform Location
var display = [ 0.0, 0.0, 0.0, 0.0 ];
var displayGL = 0; // Uniform Location
```



WebGL



Denne funktion vil starte indlæsning af tekstur samt afslutte med at tilføje uniform locations som kan bruges i fragment shaderen.

```
function CreateTexture(prog, url)
{
    // Load texture to graphics card
    const texture = LoadTexture(url);
    // Flip y axis so it fits OpenGL standard
    gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, true);
    // Activate texture to texture unit 0
    gl.activeTexture(gl.TEXTURE0);
    gl.bindTexture(gl.TEXTURE_2D, texture);
    // Add uniform location to fragment shader
    textureGL = gl.getUniformLocation(prog, 'Texture');
    // Add uniform location to fragment shader
    displayGL = gl.getUniformLocation(prog, 'Display');
}
```



WebGL



Denne funktion indlæser tekstur og sender den til grafikkortet.

Men da teksten skal downloades vil det tage lidt tid før indholdet at teksten kan indlæses.

Derfor laves der en midlertidig tekstur, der i dette tilfælde blot er en 1x1 blå pixel.

Denne midlertidig tekstur vil blive vist indtil den tiltænkte tekstur er blevet downloaded færdigt.

```
function LoadTexture(url) {  
    const texture = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, texture);  
    const pixel = new Uint8Array([0, 0, 255, 255]);  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 1, 1, 0,  
gl.RGBA, gl.UNSIGNED_BYTE, pixel);  
    const image = new Image();  
    image.onload = () => {  
        gl.bindTexture(gl.TEXTURE_2D, texture);  
        gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA,  
gl.RGBA, gl.UNSIGNED_BYTE, image);  
        SetTextureFilters(image);  
    };  
    image.src = url;  
    return texture;  
}
```



WebGL



Fortæller grafikortet at vi gerne vil lave en tekstur

Men da teksten skal downloades vil det tage lidt tid før indholdet at teksten kan indlæses.

Derfor laves der en midlertidig tekstur, der i dette tilfælde blot er en 1x1 blå pixel.

Denne midlertidig tekstur vil blive vist indtil den tiltænkte tekstur er blevet downloaded færdigt.

```
function LoadTexture(url) {  
  const texture = gl.createTexture();  
  gl.bindTexture(gl.TEXTURE_2D, texture);  
  const pixel = new Uint8Array([0, 0, 255, 255]);  
  gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 1, 1, 0,  
    gl.RGBA, gl.UNSIGNED_BYTE, pixel);  
  const image = new Image();  
  image.onload = () => {  
    gl.bindTexture(gl.TEXTURE_2D, texture);  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA,  
      gl.RGBA, gl.UNSIGNED_BYTE, image);  
    SetTextureFilters(image);  
  };  
  image.src = url;  
  return texture;  
}
```



WebGL



Denne funktion indlæser tekstur og sender den til

'Bind' nylavet tekstur til kommende kode

Men da teksten skal downloades vil det tage lidt tid før indholdet at teksten kan indlæses.

Derfor laves der en midlertidig tekstur, der i dette tilfælde blot er en 1x1 blå pixel.

Denne midlertidig tekstur vil blive vist indtil den tiltænkte tekstur er blevet downloaded færdigt.

```
function LoadTexture(url) {  
    const texture = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, texture);  
    const pixel = new Uint8Array([0, 0, 255, 255]);  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 1, 1, 0,  
        gl.RGBA, gl.UNSIGNED_BYTE, pixel);  
    const image = new Image();  
    image.onload = () => {  
        gl.bindTexture(gl.TEXTURE_2D, texture);  
        gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA,  
            gl.RGBA, gl.UNSIGNED_BYTE, image);  
        SetTextureFilters(image);  
    };  
    image.src = url;  
    return texture;  
}
```



WebGL



Denne funktion indlæser tekstur og sender den til grafikkortet

Blå pixel bruges som midlertidig tekstur

tid før indholdet at teksten kan indlæses.

Derfor laves der en midlertidig tekstur, der i dette tilfælde blot er en 1x1 blå pixel.

Denne midlertidig tekstur vil blive vist indtil den tiltænkte tekstur er blevet downloaded færdigt.

```
function LoadTexture(url) {  
    const texture = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, texture);  
    const pixel = new Uint8Array([0, 0, 255, 255]);  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 1, 1, 0,  
        gl.RGBA, gl.UNSIGNED_BYTE, pixel);  
    const image = new Image();  
    image.onload = () => {  
        gl.bindTexture(gl.TEXTURE_2D, texture);  
        gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA,  
            gl.RGBA, gl.UNSIGNED_BYTE, image);  
        SetTextureFilters(image);  
    };  
    image.src = url;  
    return texture;  
}
```



WebGL



Denne funktion indlæser tekstur og sender den til grafikkortet.

Fortæl grafikkort at blå pixel skal bruges som tekstur

Derfor laves der en midlertidig tekstur, der i dette tilfælde blot er en 1x1 blå pixel.

Denne midlertidig tekstur vil blive vist indtil den tiltænkte tekstur er blevet downloaded færdigt.

```
function LoadTexture(url) {  
    const texture = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, texture);  
    const pixel = new Uint8Array([0, 0, 255, 255]);  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 1, 1, 0,  
    gl.RGBA, gl.UNSIGNED_BYTE, pixel);  
    const image = new Image();  
    image.onload = () => {  
        gl.bindTexture(gl.TEXTURE_2D, texture);  
        gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA,  
        gl.RGBA, gl.UNSIGNED_BYTE, image);  
        SetTextureFilters(image);  
    };  
    image.src = url;  
    return texture;  
}
```



WebGL



Denne funktion indlæser tekstur og sender den til grafikkortet.

Men da teksten skal downloades vil det tage lidt

Lav et billede til download af tekstur

Derfor laves der en midlertidig tekstur, der i dette tilfælde blot er en 1x1 blå pixel.

Denne midlertidig tekstur vil blive vist indtil den tiltænkte tekstur er blevet downloaded færdigt.

```
function LoadTexture(url) {  
    const texture = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, texture);  
    const pixel = new Uint8Array([0, 0, 255, 255]);  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 1, 1, 0,  
gl.RGBA, gl.UNSIGNED_BYTE, pixel);  
    const image = new Image();  
    image.onload = () => {  
        gl.bindTexture(gl.TEXTURE_2D, texture);  
        gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA,  
gl.RGBA, gl.UNSIGNED_BYTE, image);  
        SetTextureFilters(image);  
    };  
    image.src = url;  
    return texture;  
}
```



WebGL



Denne funktion indlæser tekstur og sender den til grafikortet.

Men da teksten skal downloades vil det tage lidt tid før indholdet af teksten kan indlæses

Når billede er downloaded så opdater tekstur

tilfælde blot er en 1x1 blå pixel.

Denne midlertidig tekstur vil blive vist indtil den tiltænkte tekstur er blevet downloaded færdigt.

```
function LoadTexture(url) {  
    const texture = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, texture);  
    const pixel = new Uint8Array([0, 0, 255, 255]);  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 1, 1, 0,  
    gl.RGBA, gl.UNSIGNED_BYTE, pixel);  
    const image = new Image();  
    image.onload = () => {  
        gl.bindTexture(gl.TEXTURE_2D, texture);  
        gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA,  
        gl.RGBA, gl.UNSIGNED_BYTE, image);  
        SetTextureFilters(image);  
    };  
    image.src = url;  
    return texture;  
}
```



WebGL



Denne funktion indlæser tekstur og sender den til grafikkortet.

Men da teksturen skal downloades vil det tage lidt tid før indholdet at teksturen kan indlæses.

Derfor laves der en midlertidig tekstur, der i dette tilfælde blot er en 1x1 blå pixel.

Sæt tekstur filter (vi har snakket om dette i DAP)

tiltænkte tekstur er blevet downloaded færdigt.

```
function LoadTexture(url) {  
    const texture = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, texture);  
    const pixel = new Uint8Array([0, 0, 255, 255]);  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 1, 1, 0,  
gl.RGBA, gl.UNSIGNED_BYTE, pixel);  
    const image = new Image();  
    image.onload = () => {  
        gl.bindTexture(gl.TEXTURE_2D, texture);  
        gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA,  
gl.RGBA, gl.UNSIGNED_BYTE, image);  
        SetTextureFilters(image);  
    };  
    image.src = url;  
    return texture;  
}
```



WebGL




Denne funktion indlæser tekstur og sender den til grafikkortet.

Men da teksten skal downloades vil det tage lidt tid før indholdet at teksten kan indlæses.

Derfor laves der en midlertidig tekstur, der i dette tilfælde blot er en 1x1 blå pixel.

Start download af billede



```
function LoadTexture(url) {  
    const texture = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, texture);  
    const pixel = new Uint8Array([0, 0, 255, 255]);  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 1, 1, 0,  
gl.RGBA, gl.UNSIGNED_BYTE, pixel);  
    const image = new Image();  
    image.onload = () => {  
        gl.bindTexture(gl.TEXTURE_2D, texture);  
        gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA,  
gl.RGBA, gl.UNSIGNED_BYTE, image);  
        SetTextureFilters(image);  
    };  
    image.src = url;  
    return texture;  
}
```

WebGL




Denne funktion indlæser tekstur og sender den til grafikkortet.

Men da teksten skal downloades vil det tage lidt tid før indholdet at teksten kan indlæses.

Derfor laves der en midlertidig tekstur, der i dette tilfælde blot er en 1x1 blå pixel.

Denne midlertidig tekstur vil blive vist indtil den

Returner handle af tekstur gemt på grafikkortet

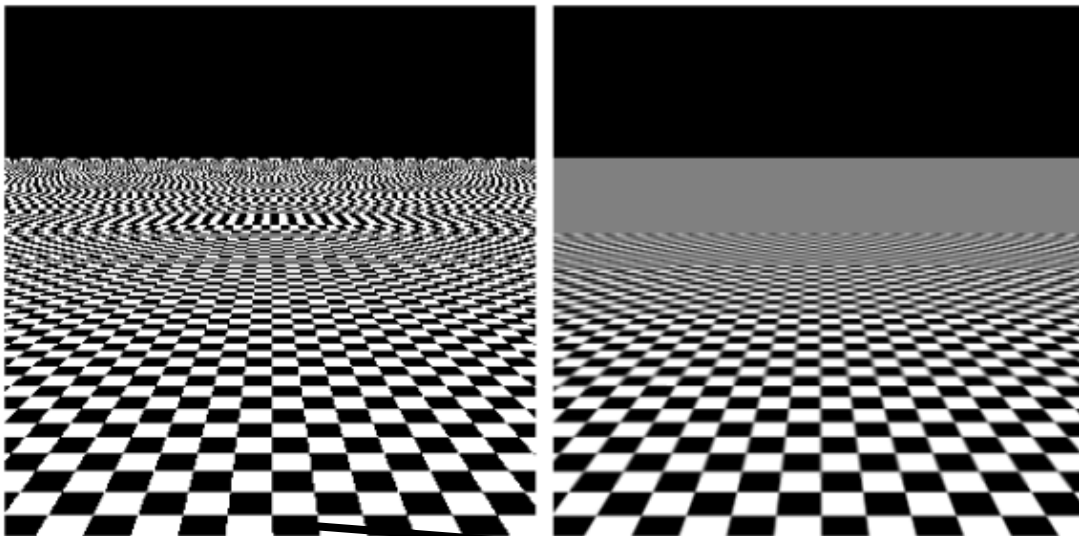


```
function LoadTexture(url) {  
    const texture = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, texture);  
    const pixel = new Uint8Array([0, 0, 255, 255]);  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 1, 1, 0,  
gl.RGBA, gl.UNSIGNED_BYTE, pixel);  
    const image = new Image();  
    image.onload = () => {  
        gl.bindTexture(gl.TEXTURE_2D, texture);  
        gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA,  
gl.RGBA, gl.UNSIGNED_BYTE, image);  
        SetTextureFilters(image);  
    };  
    image.src = url;  
    return texture;  
}
```

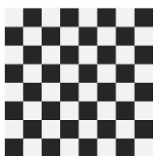
WebGL



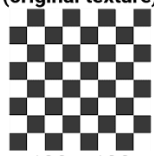
Denne funktion sætter tekstur filter.



Ingen Mipmapping



Mip 0
(original texture)



Mip 1



Mip 2



Mip 3



128 x 128

64 x 64

32 x 32

16 x 16

```
function SetTextureFilters(image)
{
    if (IsPow2(image.width) && IsPow2(image.height))
    {
        gl.generateMipmap(gl.TEXTURE_2D);
    }
    else
    {
        gl.texParameteri(gl.TEXTURE_2D,
            gl.TEXTURE_WRAP_S, gl.CLAMP_TO_EDGE);
        gl.texParameteri(gl.TEXTURE_2D,
            gl.TEXTURE_WRAP_T, gl.CLAMP_TO_EDGE);
        gl.texParameteri(gl.TEXTURE_2D,
            gl.TEXTURE_MIN_FILTER, gl.LINEAR);
    }
}
```

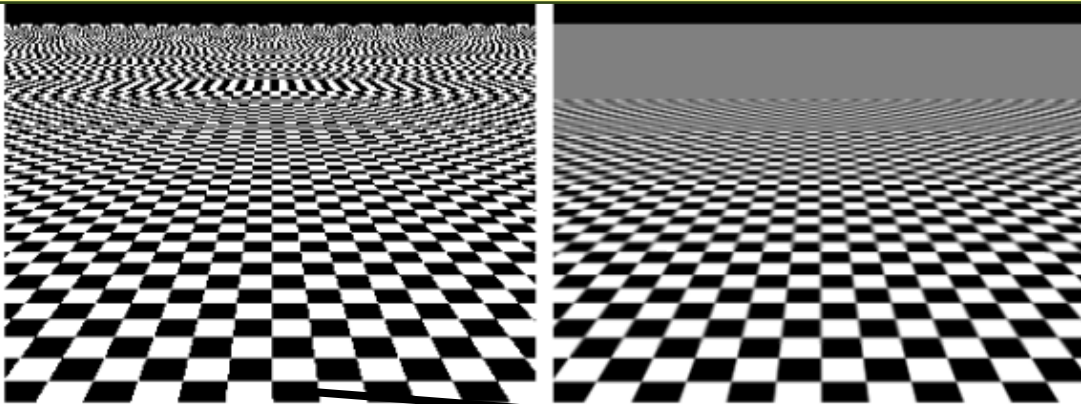


WebGL

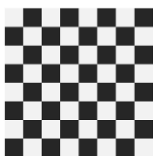


Denne funktion sætter tekstur filter.

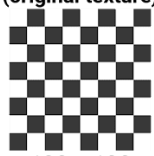
Tjekker størrelse på tekstur



Ingen Mipmapping



Mip 0
(original texture)



128 x 128

Mip 1



64 x 64

Mip 2



32 x 32

Mip 3



16 x 16

```
function SetTextureFilters(image)
{
  if (IsPow2(image.width) && IsPow2(image.height))
  {
    gl.generateMipmap(gl.TEXTURE_2D);
  }
  else
  {
    gl.texParameteri(gl.TEXTURE_2D,
gl.TEXTURE_WRAP_S, gl.CLAMP_TO_EDGE);
    gl.texParameteri(gl.TEXTURE_2D,
gl.TEXTURE_WRAP_T, gl.CLAMP_TO_EDGE);
    gl.texParameteri(gl.TEXTURE_2D,
gl.TEXTURE_MIN_FILTER, gl.LINEAR);
  }
}
```

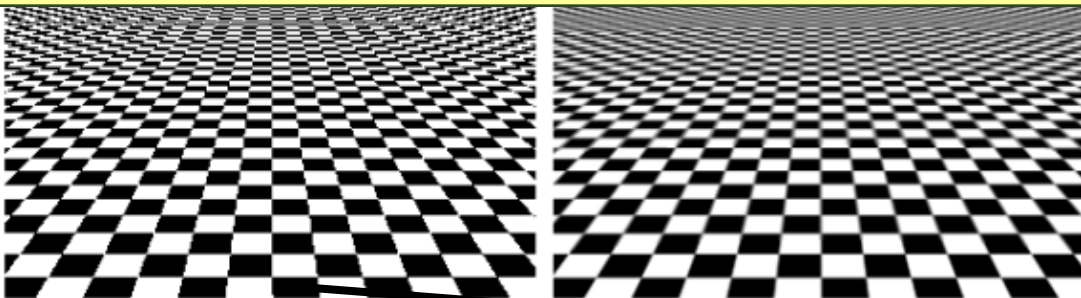


WebGL

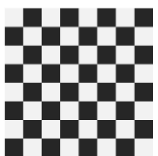


Denne funktion sætter tekstur filter.

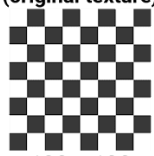
Fortæl grafikort at generere mipmap tekstur



Ingen Mipmapping



Mip 0
(original texture)



Mip 1



Mip 2



Mip 3

128 x 128

64 x 64

32 x 32

16 x 16

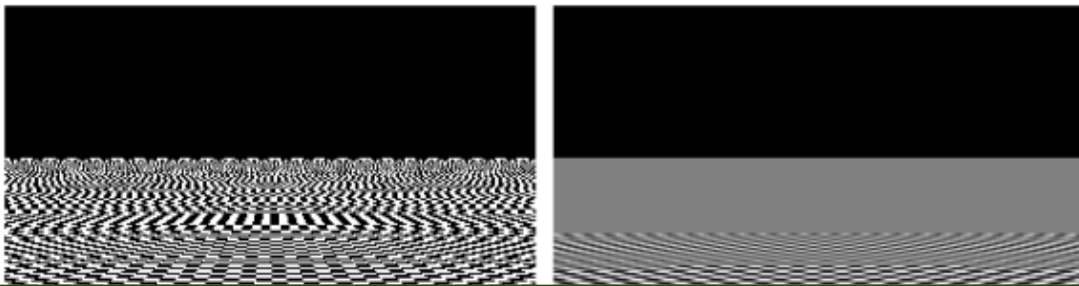
```
function SetTextureFilters(image)
{
    if (IsPow2(image.width) && IsPow2(image.height))
    {
        gl.generateMipmap(gl.TEXTURE_2D);
    }
    else
    {
        gl.texParameteri(gl.TEXTURE_2D,
            gl.TEXTURE_WRAP_S, gl.CLAMP_TO_EDGE);
        gl.texParameteri(gl.TEXTURE_2D,
            gl.TEXTURE_WRAP_T, gl.CLAMP_TO_EDGE);
        gl.texParameteri(gl.TEXTURE_2D,
            gl.TEXTURE_MIN_FILTER, gl.LINEAR);
    }
}
```



WebGL



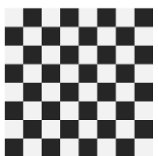
Denne funktion sætter tekstur filter.



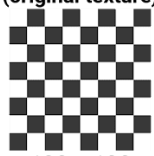
Sæt tekstur filter op uden mipmaps
(S & T er lig med U & V)



Ingen Mipmapping



Mip 0
(original texture)



Mip 1



Mip 2



Mip 3



128 x 128

64 x 64

32 x 32

16 x 16

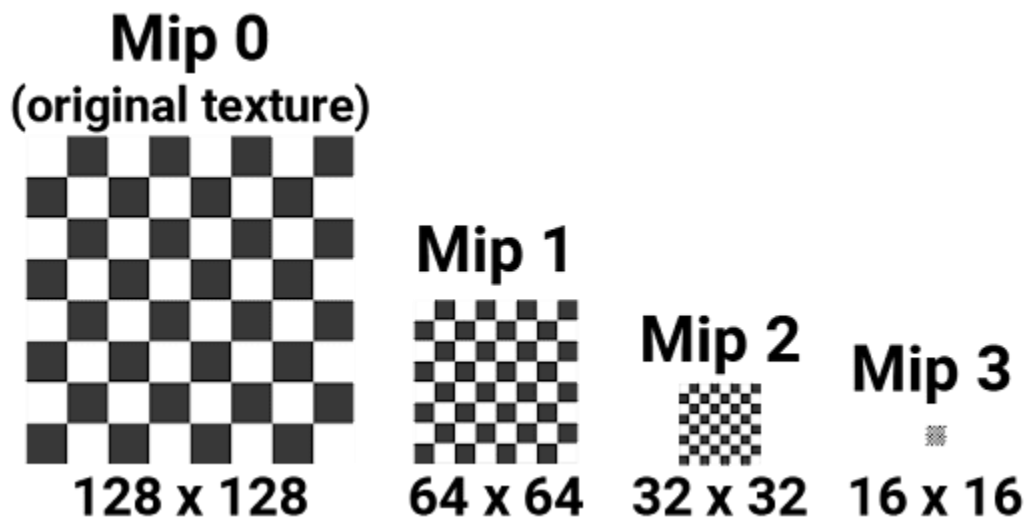
```
function SetTextureFilters(image)
{
    if (IsPow2(image.width) && IsPow2(image.height))
    {
        gl.generateMipmap(gl.TEXTURE_2D);
    }
    else
    {
        gl.texParameteri(gl.TEXTURE_2D,
            gl.TEXTURE_WRAP_S, gl.CLAMP_TO_EDGE);
        gl.texParameteri(gl.TEXTURE_2D,
            gl.TEXTURE_WRAP_T, gl.CLAMP_TO_EDGE);
        gl.texParameteri(gl.TEXTURE_2D,
            gl.TEXTURE_MIN_FILTER, gl.LINEAR);
    }
}
```



WebGL



Denne funktion tjekker om værdi er power of 2
Kun tekstur med størrelse af power of 2 kan
generere (hardware accelererende) mipmaps.



```
function IsPow2(value)
{
    return (value & (value - 1)) === 0;
}
```



WebGL



Tilføj UV koordinater til AddVertex()

En vertex består nu i alt af 8 floats

```
// Triangle
const vertices = [X Y Z R G B U V
0.0, 0.5, 0.0, 1.0, 0.0, 0.0, 0.5, 1.0,
-0.5,-0.5, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
0.5,-0.5, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0];
```

```
function AddVertex(x, y, z, r, g, b, u, v)
{
    const index = vertices.length;
    vertices.length += 8;
    vertices[index + 0] = x;
    vertices[index + 1] = y;
    vertices[index + 2] = z;
    vertices[index + 3] = r;
    vertices[index + 4] = g;
    vertices[index + 5] = b;
    vertices[index + 6] = u;
    vertices[index + 7] = v;
}
```



WebGL



Tilføj også UV koordinater til AddTriangle().

```
function AddTriangle(x1, y1, z1, r1, g1, b1, u1, v1,  
                    x2, y2, z2, r2, g2, b2, u2, v2,  
                    x3, y3, z3, r3, g3, b3, u3, v3)  
{  
    AddVertex(x1, y1, z1, r1, g1, b1, u1, v1);  
    AddVertex(x2, y2, z2, r2, g2, b2, u2, v2);  
    AddVertex(x3, y3, z3, r3, g3, b3, u3, v3);  
}
```

WebGL



Tilføj også UV koordinater til AddQuad().

```
function AddQuad(x1, y1, z1, r1, g1, b1, u1, v1,  
                 x2, y2, z2, r2, g2, b2, u2, v2,  
                 x3, y3, z3, r3, g3, b3, u3, v3,  
                 x4, y4, z4, r4, g4, b4, u4, v4)  
{  
    AddTriangle(x1, y1, z1, r1, g1, b1, u1, v1,  
                x2, y2, z2, r2, g2, b2, u2, v2,  
                x3, y3, z3, r3, g3, b3, u3, v3);  
  
    AddTriangle(x3, y3, z3, r3, g3, b3, u3, v3,  
                x4, y4, z4, r4, g4, b4, u4, v4,  
                x1, y1, z1, r1, g1, b1, u1, v1);  
}
```




WebGL



Tilføj også UV koordinater til CreateTriangle().

```
function CreateTriangle(width, height)
{
    vertices.length = 0;
    const w = width * 0.5;
    const h = height * 0.5;
    AddTriangle(0.0, h, 0.0, 1.0, 0.0, 0.0, 0.5, 1.0,
               -w, -h, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
               w, -h, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0);
}
```



WebGL



Tilføj også UV koordinater til CreateQuad().

```
function CreateQuad(width, height)
{
    vertices.length = 0;
    const w = width * 0.5;
    const h = height * 0.5;
    AddQuad(-w, h, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0,
            -w, -h, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
            w, -h, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0,
            w, h, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0);
}
```



WebGL



Vi skal også have opdateret CreateVBO() funktionen.

```
function CreateVBO(program, vert)
{
    let vbo = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, vbo);
    gl.bufferData(gl.ARRAY_BUFFER, vert, gl.STATIC_DRAW);

    const s = 8 * Float32Array.BYTES_PER_ELEMENT;
```

```
// Create shader attribute: Pos
let p = gl.getAttribLocation(program, 'Pos');
gl.vertexAttribPointer(p, 3, gl.FLOAT, gl.FALSE, s, 0);
gl.enableVertexAttribArray(p);

// Create shader attribute: Color
const o = 3 * Float32Array.BYTES_PER_ELEMENT;
let c = gl.getAttribLocation(program, 'Color');
gl.vertexAttribPointer(c, 3, gl.FLOAT, gl.FALSE, s, o);
gl.enableVertexAttribArray(c);
// Create shader attribute: UV
const o2 = o * 2;
let u = gl.getAttribLocation(program, 'UV');
gl.vertexAttribPointer(u, 2, gl.FLOAT, gl.FALSE, s, o2);
gl.enableVertexAttribArray(u);
}
```



WebGL



Husk også at opdater CreateGeometryBuffers()

```
function CreateGeometryBuffers(program) {  
    // Generate selected geometry and UI  
    CreateGeometryUI();  
    // Create GPU buffer (VBO)  
    CreateVBO(program, new Float32Array(vertices));  
    angleGL = gl.getUniformLocation(program, 'Angle');  
    CreateTexture(program, 'img/tekstur.jpg');  
    // Activate shader program  
    gl.useProgram(program);  
    // Update view rotation  
    gl.uniform4fv(angleGL, new Float32Array(angle));  
    // Update display options  
    gl.uniform4fv(displayGL, new Float32Array(display));  
    // Display geometry on screen  
    Render();  
}
```



WebGL



Lad os gøre det muligt for bruger at ændre mellem at se tekstur eller se vertex colors på renderet geometri.

```
function Update()  
{  
    // Show texture (boolean) last element  
    const t = document.getElementById('t');  
    display[3] = t.checked ? 1.0 : 0.0;  
    // Update array to graphics card and render  
    gl.uniform4fv(displayGL, new Float32Array(display));  
    Render();  
}
```




WebGL



Tilføj et HTML **checkbox** element med id: **t**

```
<body onload="InitWebGL();">
  <canvas id="gl" width="800px" height="600px">
    WebGL is not supported!
  </canvas>
  <textarea id="vs" spellcheck="false"></textarea>
  <textarea id="fs" spellcheck="false"></textarea>
  <select id="shape" onchange="InitShaders();">
    <option selected>Triangle</option>
    <option>Quad</option>
  </select>
  <input type="checkbox" id="t" onchange="Update();">
  Show Texture<br>
  <div id="ui">Generate UI for geometry here!</div>
  <script src="js/webgl.js" defer></script>
</body>
```



WebGL

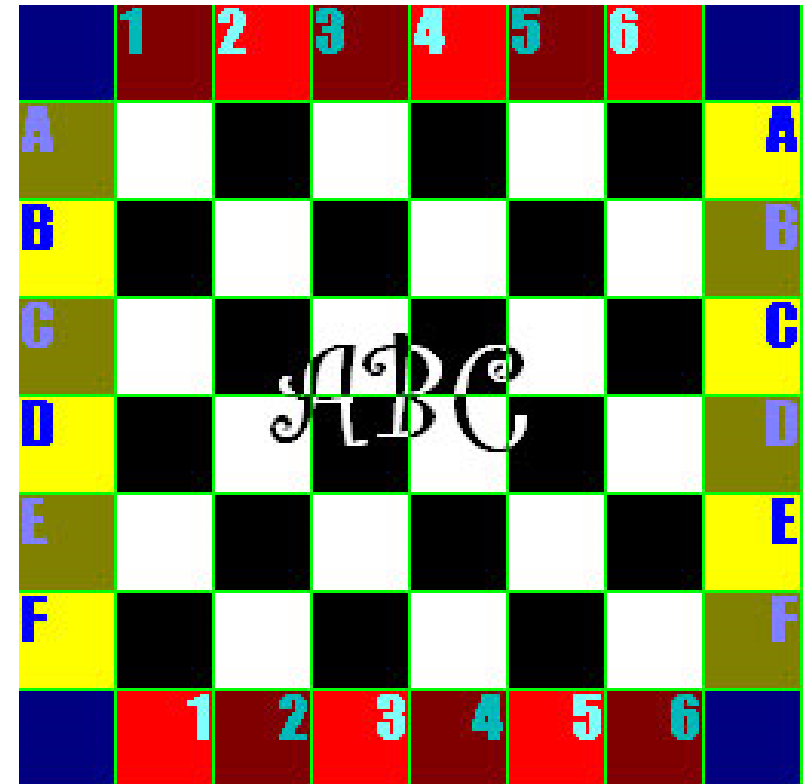


Ofte bruger jeg denne tekstur til test af følgende grunde:

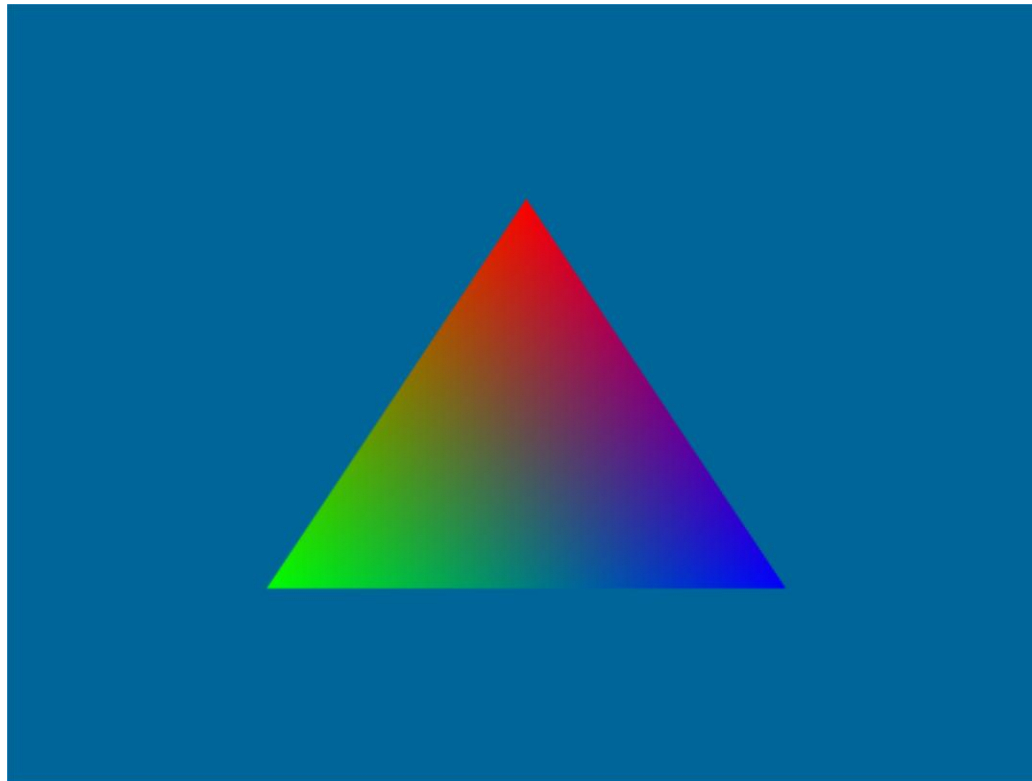
- Let at orientere sig
- RGB farver vist korrekt
- Spejlvendt
- Størrelse er 256x256 (Power of two)

Kan hentes her:

<http://www.udvikleren.dk/images/articleimages/1812.jpg>



WebGL



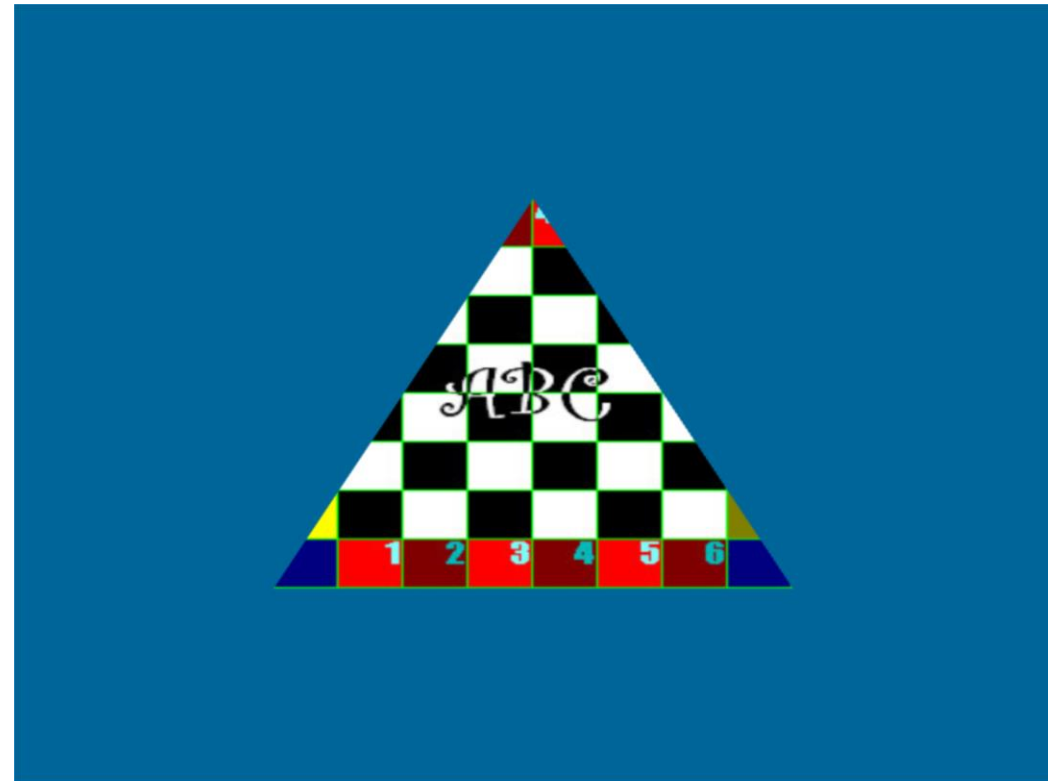
precision
mediump float;

precision
mediump float;

Triangle ☐ Show Texture

Width: 1

Height: 1



precision
mediump float;

precision
mediump float;

Triangle ☒ Show Texture

Width: 1

Height: 1

WebGL

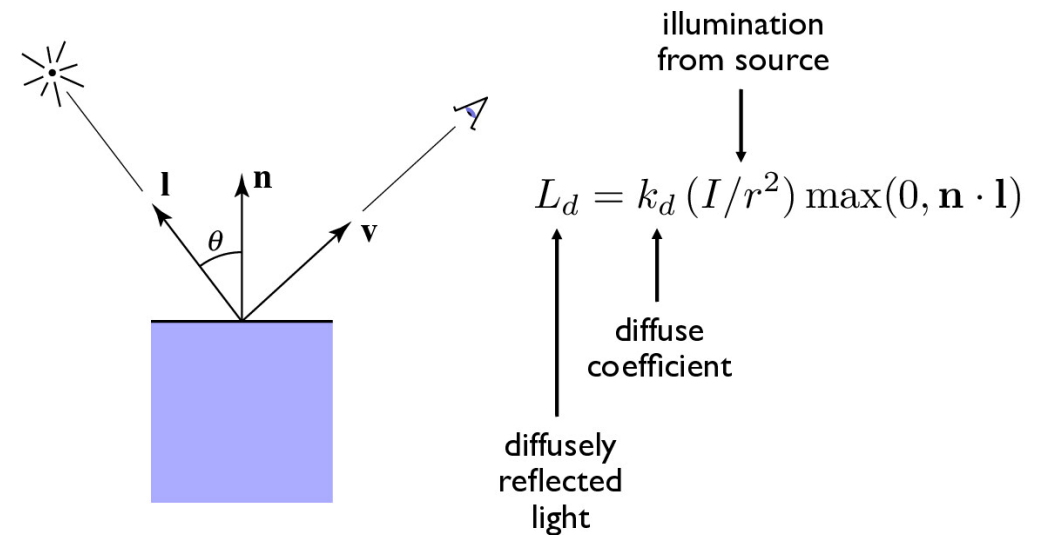


**Hvordan belyses overflader i
3D grafik?**

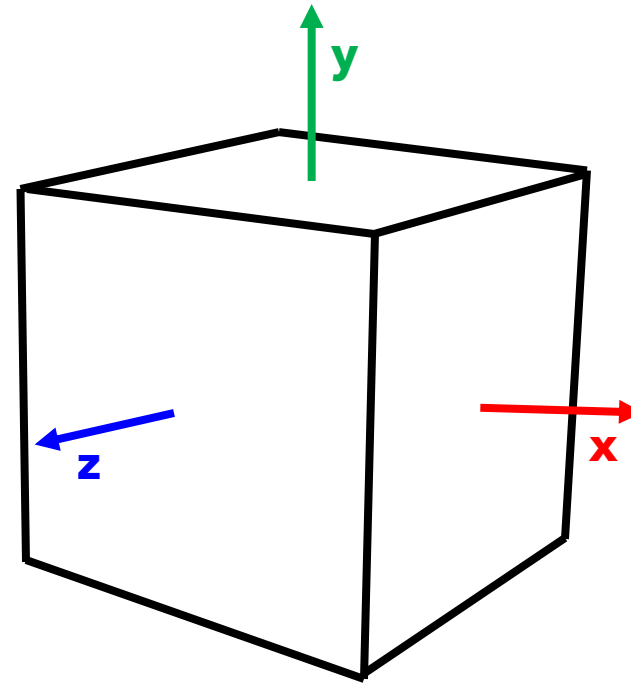


Lambertian (Diffuse) Shading

Shading independent of view direction



WebGL

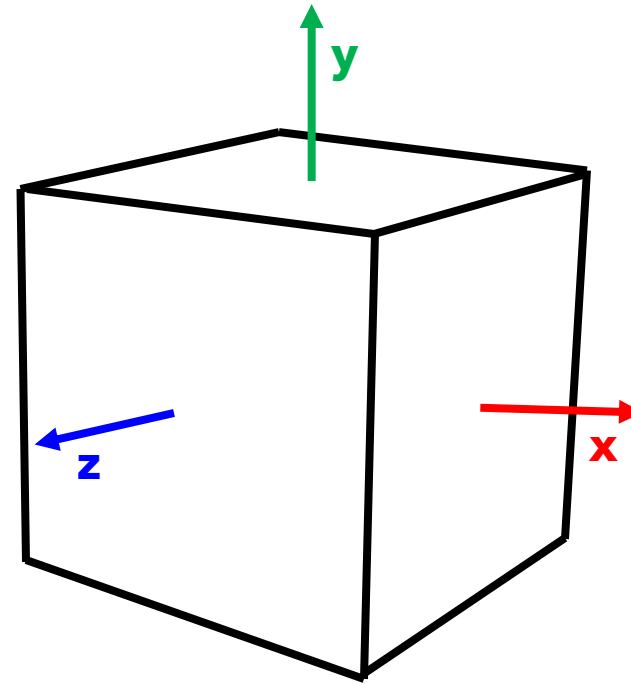


Alle normaler for en 3D kasse vil altid flugte X, Y og Z akserne.
Eksempelvis vil normal for toppen af kassen flugte Y akse: **0.0, 1.0, 0.0**

WebGL



Husk at en normal altid skal være 1 unit lang!



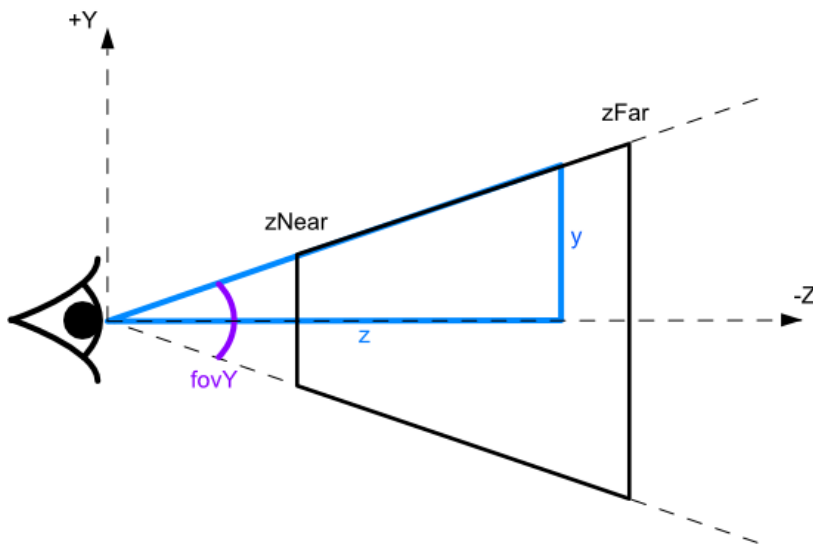
Alle normaler for en 3D kasse vil altid flugte X, Y og Z akserne.
Eksempelvis vil normal for toppen af kassen flugte Y akse: **0.0, 1.0, 0.0**

WebGL



Vi kan implementere Lambertian (diffuse) shading i fragment shaderen, således:

Læg mærke til at lysets retning er langs z-aksen.



```
precision mediump float;
uniform sampler2D Texture;
uniform vec4 Display;
varying vec3 vertexColor;
varying vec2 uv;
varying vec3 normal;
void main()
{
    vec3 lightDirection = vec3(0.0, 0.0, 1.0);
    float lambert = dot(lightDirection, normal);
    vec3 shade = Display.rgb * lambert;
    float p = abs(Display.w);
    vec3 texture = texture2D(Texture, uv).rgb;
    vec3 color = vertexColor;
    gl_FragColor = vec4(mix(color, texture, p) *shade, 1.0);
}
```



WebGL



Tilføj normal til AddVertex() ... samt AddTriangle(), AddQuad(), CreateTriangle(), CreateQuad(), CreateVBO() osv.

En vertex består nu i alt af 11 floats

```
// Triangle
const vertices = [X   Y   Z   R   G   B   U   V   Nx  Ny  Nz,
                  0.0, 0.5, 0.0, 1.0, 0.0, 0.0, 0.5, 1.0, 0.0, 0.0, 1.0,
                  -0.5,-0.5, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0,
                  0.5,-0.5, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0];
```

```
function AddVertex(x, y, z, r, g, b, u, v, nx, ny, nz)
{
    const index = vertices.length;
    vertices.length += 11;
    vertices[index + 0] = x;
    vertices[index + 1] = y;
    vertices[index + 2] = z;
    vertices[index + 3] = r;
    vertices[index + 4] = g;
    vertices[index + 5] = b;
    vertices[index + 6] = u;
    vertices[index + 7] = v;
    vertices[index + 8] = nx;
    vertices[index + 9] = ny;
    vertices[index + 10] = nz;
}
```

WebGL



Opdatering af lysets farve, ved at konvertere hex decimaltal til en RGB farve

```
function Update()  
{  
    // Show texture (boolean) last element  
    const t = document.getElementById('t');  
    display[3] = t.checked ? 1.0 : 0.0;  
    // Light color (convert hex to RGB)  
    const l = document.getElementById('l').value;  
    display[0] = parseInt(l.substring(1,3),16) / 255.0;  
    display[1] = parseInt(l.substring(3,5),16) / 255.0;  
    display[2] = parseInt(l.substring(5,7),16) / 255.0;  
    // Update array to graphics card and render  
    gl.uniform4fv(displayGL,new Float32Array(display));  
    Render();  
}
```



WebGL



Tilføj et HTML **color** element med id: **l**

```
<body onload="InitWebGL();">
  <canvas id="gl" width="800px" height="600px">
    WebGL is not supported!
  </canvas>
  <textarea id="vs" spellcheck="false"></textarea>
  <textarea id="fs" spellcheck="false"></textarea>
  <select id="shape" onchange="InitShaders();">
    <option selected>Triangle</option>
    <option>Quad</option>
  </select>
  <input type="checkbox" id="t" onchange="Update();">
  Show Texture<input type="color" id="l"
value="#f6b73c" onchange="Update ()">Light Color<br>
  <div id="ui">Generate UI for geometry here!</div>
  <script src="js/webgl.js" defer></script>
</body>
```



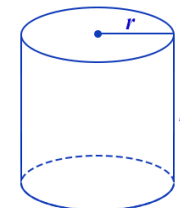
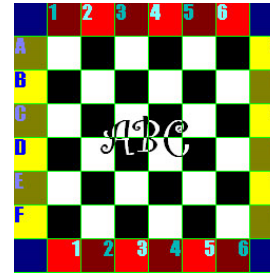
WebGL



Gør dig selv en tjeneste og implementer tekstur først, og få det til at virke, før du begynder at implementer belysning.

Øvelse: **Tekstur & Belysning**

- 1) Implementer tekstur ved at tilføje UV koordinater og funktionerne `CreateTexture()`, `LoadTexture()` som anvist i oplæg.
- 2) Implementer belysning ved at tilføje normals og lambertian shading i fragment shader
- 3) Implementer en 3D cylinder ved brug af `Math.sin()` & `Math.cos()`



Det er lettere
end du tror!

