WebGL Water - Shader Variablen und Konstanten Code-Karte

Water Vertex Shader

WaterVertexShader

Attribute:
- attribute position: vec3

Uniforms:
- uniform perspective: mat4
- uniform model: mat4
- uniform view: mat4
- uniform cameraPos: vec3

- varying clipSpace: vec4
- varying textureCoords: vec2
- varying fromFragmentToCamera: vec3

Konstanten:
- const tiling: float = 4.0

Berechnungen:
clipSpace = perspective * view * worldPosition
fromFragmentToCamera = cameraPos - worldPosition.xyz

Varyings (Output):
worldPosition = model * vec4(position, 1.0)
textureCoords = (position.xz + 0.5) * tiling

clipSpace, textureCoords, fromFragmentToCamera

Water Fragment Shader

WaterFragmentShader

Textur Samplers:
- uniform refractionTexture: sampler2D
- uniform reflectionTexture: sampler2D
- uniform dudvTexture: sampler2D
- uniform normalMap: sampler2D
- uniform waterDepthTexture: sampler2D

Animation & Eigenschaften:
- uniform dudvOffset: float
- uniform waterReflectivity: float
- uniform fresnelStrength: float

- varying clipSpace: vec4
- varying textureCoords: vec2
- varying fromFragmentToCamera: vec3

Konstanten:
- const waterDistortionStrength: float = 0.03
- const shineDamper: float = 20.0

Licht Konstanten:

Wasser Farben:

Funktionen:

Varyings (Input):
- vec3 sunlightColor = vec3(1.0, 1.0, 1.0)
- vec3 sunlightDir = normalize(vec3(-1.0, -1.0, 0.5))
- vec4 shallowWaterColor = vec4(0.0, 0.1, 0.3, 1.0)
- vec4 deepWaterColor = vec4(0.0, 0.1, 0.2, 1.0)
- getNormal(textureCoords): vec3

verwendet

WaterCalculations

NDC Berechnung:

Textur Koordinaten:

Tiefen Berechnung:
near = 0.1, far = 50.0
cameraToFirstThingUnderWater = 2.0 * near * far / ...
angledWaterDepth = underwater - waterDistance

Verzerrung:
distortedTexCoords = dudvTexture + dudvOffset

Fresnel Effekt:

ndc = (clipSpace.xy / clipSpace.w) / 2.0 + 0.5
refractTexCoords = vec2(ndc.x, ndc.y)
reflectTexCoords = vec2(ndc.x, -ndc.y)
totalDistortion = (dudvTexture * 2.0 - 1.0) * strength
refractiveFactor = dot(toCamera, normal)
refractiveFactor = pow(refractiveFactor, fresnelStrength)

Fresnel Formel:
Je steiler der Blickwinkel, desto mehr Reflektion
Je flacher der Blickwinkel, desto mehr Refraktion
fresnelStrength = 2.0 für realistische Wasser

Textur Binding:
TEXTURE0 → refractionTexture
TEXTURE1 → reflectionTexture
TEXTURE2 → dudvTexture
TEXTURE3 → normalMap
TEXTURE4 → waterDepthTexture

Clip Planes:
Negative Y: Alles unter Wasser (Refraktion)
Positive Y: Alles über Wasser (Reflektion)

Shader Pipeline Datenfluss

RenderingPasses

Refraktion Pass:
→ render terrain below water
→ output to refractionTexture

Reflektion Pass:
→ flip view matrix
→ render terrain above water
→ output to reflectionTexture

Wasser Pass:
→ render water plane

clipPlane = vec4(0, -1, 0, 0)
clipPlane = vec4(0, 1, 0, 0)
→ bind refractionTexture (TEXTURE0)
→ bind reflectionTexture (TEXTURE1)
→ bind dudvTexture (TEXTURE2)
→ bind normalMap (TEXTURE3)
→ dudvOffset = (time / 1000.0) * waveSpeed

Mesh Vertex Shader

MeshVertexShader

Attribute:
- attribute position: vec3
- attribute normal: vec3
- attribute uvs: vec2

Uniforms:
- uniform perspective: mat4
- uniform view: mat4
- uniform model: mat4
- uniform clipPlane: vec4

- varying vUvs: vec2
- varying vNormal: vec3
- varying vWorldPos: vec3

Clipping:

Varyings (Output):
gl_ClipDistance[0] = dot(worldPos, clipPlane)

vUvs, vNormal

Mesh Fragment Shader

MeshFragmentShader

Uniforms:
- uniform tex: sampler2D

- varying vUvs: vec2
- varying vNormal: vec3

Output:

Varyings (Input):
gl_FragColor = texture2D(tex, vUvs)

Textured Quad Shader

QuadVertexShader

Attribute:
- attribute position: vec2

Varyings:
- varying textureCoords: vec2

Berechnung:
textureCoords = position * 0.5 + 0.5

gl_Position = vec4(position, 0.0, 1.0)

textureCoords

QuadFragmentShader

Uniforms:
- uniform texture: sampler2D

Varyings:
- varying textureCoords: vec2

Output:

gl_FragColor = texture2D(texture, textureCoords)