WebGL Water Tutorial - Komponenten Klassendiagramm Shader Komponenten REST API Endpunkte: (C) WaterVertexShader - GET / (Hauptseite) (C) MeshVertexShader - GET /static/* (JS/CSS) oattribute position: vec3 - GET /assets/* (Texturen) oattribute position: vec3 ouniform perspective: mat4 - GET /shaders/* (GLSL) ouniform view: mat4 oattribute normal: vec3 GET /api/meshes ouniform model: mat4 oattribute uvs: vec2 - GET /api/state ovarying clipSpace: vec4 ouniform clipPlane: vec4 POST /api/state/water ovarying textureCoords: vec2 POST /api/state/camera WebSocket/ws Pipeline |Pipeline Go Backend Komponenten (C) Server □router: *mux.Router □assets: *Assets (C) WaterFragmentShader □appState: *State □upgrader: websocket.Upgrader ouniform refractionTexture: sampler2D □ clients: map[*websocket.Conn]bool (C) MeshFragmentShader ouniform reflectionTexture: sampler2D □ static Path: string ouniform dudvTexture: sampler2D port: int ouniform tex: sampler2D ouniform normalMap: sampler2D ovarying vUvs: vec2 o New Server(): *Server ouniform waterReflectivity: float oStart(): error ouniform fresnelStrength: float ouniform dudvOffset: float osetupRoutes() ohandleIndex() ohandleAssetFile() ohandleShader() ohandleWebSocket() \ HTTP/WebSocket Asset/Status Daten JavaScript Frontend Komponenten (C) WebGLWaterApp □gl: WebGLRenderingContext canvas: HTMLCanvasElement (C) Assets C State □programs: object meshes: object □mu: sync.RW Mutex □meshes: map[string]*Mesh ntextures: object clock: Clock □textures: map[string]*Texture □ framebuffers: object □basePath: string □camera: *Camera □state: object water: Water o New Assets(): *Assets □ws: WebSocket □scenery: Scenery o Initialize(): error oinit(): Promise<void> o CreateWaterMesh(): *Mesh oNewState(): *State oloadShaders(): Promise<void> o CreateTerrainMesh(): *Mesh oUpdate(Message) oloadAssets(): Promise<void> oGetMesh(): (*Mesh, error) oGetCamera(): *Camera osetupFramebuffers() oGetWater(): Water oListMeshes(): []string orender() orenderWater() orenderMeshes() oconnectWebSocket() (C) Camera position: math3d.Vec3 □target: math3d.Vec3 (C) Mesh (C) MeshBuffer (C) ShaderProgram (C) Framebuffer □up: math3d.Vec3 (C) Water o Name: string □distance: float32 Rendering Pipeline: (C) TextureManager oframebuffer: WebGLFramebuffer oprogram: WebGLProgram overtexBuffer: WebGLBuffer o Reflectivity: float32 o Vertices: []float32 □yaw: float32 1. Refraktion Pass → Framebuffer onormalBuffer: WebGLBuffer ocolorTexture: WebGLTexture ouniformLocations: object □textures: Map<string, WebGLTexture> oNormals: []float32 pitch: float32 oFresnelStrength: float32 odepthTexture: WebGLTexture otexCoordBuffer: WebGLBuffer oattribLocations: object 3. Hauptszene → Bildschirm oTexCoords: []float32 □minDistance: float32 oWaveSpeed: float32 owidth: number oindexBuffer: WebGLBuffer oloadTexture(): Promise<WebGLTexture> 4. Wasseroberfläche mit Multi-Texturing oIndices: []uint16 maxDistance: float32 o UseReflection: bool ocreateProgram(): WebGLProgram oindexCount: number oheight: number obindTexture() ocompileShader(): WebGLShader 5. Echtzeit Animationsschleife (60fps) o Vertex Count: int o UseRefraction: bool o New Camera(): *Camera ocreateFramebuffer(): Framebuffer oTriangleCount: int olinkProgram() ocreateMeshBuffers(): MeshBuffer oOrbitLeftRight(delta: float32) oOrbitUpDown(delta: float32) oZoom(delta: float32) oGetViewMatrix(): math3d.Mat4 Math3D Komponenten **(C)** V e c 3 **(C)** M at 4 oX: float32 oY: float32 □elements: [16]float32 oZ: float32 o New Mat4(): Mat4 o Multiply(): Mat4 o Perspective(): Mat4 oNewVec3(): Vec3

oAdd(): Vec3 oSub(): Vec3

oCross(): Vec3 oNormalize(): Vec3

oLookAt(): Mat4 oToSlice(): []float32