WebGL Water Tutorial - Component Class Diagram

REST API Endpoints:
- GET / (main page)
- GET /static/* (JS/CSS)
- GET /assets/* (textures)
- GET /shaders/* (GLSL)
- GET /api/meshes
- GET /api/state
- POST /api/state/water
- POST /api/state/camera
- WebSocket /ws

Go Backend Components

Server
- router: *mux.Router
- assets: *Assets
- appState: *State
- upgrader: websocket.Upgrader
- clients: map[*websocket.Conn]bool
- staticPath: string
- port: int

- NewServer(): *Server
- Start(): error
- setupRoutes()
- handleIndex()
- handleAssetFile()
- handleShader()
- handleWebSocket()

Assets
- meshes: map[string]*Mesh
- textures: map[string]*Texture
- basePath: string

- NewAssets(): *Assets
- Initialize(): error
- CreateWaterMesh(): *Mesh
- CreateTerrainMesh(): *Mesh
- GetMesh(): (*Mesh, error)
- ListMeshes(): []string

State
- mu: sync.RWMutex
- clock: Clock
- camera: *Camera
- water: Water
- scenery: Scenery

- NewState(): *State
- Update(Message)
- GetCamera(): *Camera
- GetWater(): Water

Mesh
- Name: string
- Vertices: []float32
- Normals: []float32
- TexCoords: []float32
- Indices: []uint16
- VertexCount: int
- TriangleCount: int

Camera
- position: math3d.Vec3
- target: math3d.Vec3
- up: math3d.Vec3
- distance: float32
- yaw: float32
- pitch: float32
- minDistance: float32
- maxDistance: float32

- NewCamera(): *Camera
- OrbitLeftRight(delta: float32)
- OrbitUpDown(delta: float32)
- Zoom(delta: float32)
- GetViewMatrix(): math3d.Mat4

Water
- Reflectivity: float32
- FresnelStrength: float32
- WaveSpeed: float32
- UseReflection: bool
- UseRefraction: bool

Math3D Components

Vec3
- X: float32
- Y: float32
- Z: float32

- NewVec3(): Vec3
- Add(): Vec3
- Sub(): Vec3
- Cross(): Vec3
- Normalize(): Vec3

Mat4
- elements: [16]float32

- NewMat4(): Mat4
- Multiply(): Mat4
- Perspective(): Mat4
- LookAt(): Mat4
- ToSlice(): []float32

Asset/State data

HTTP/WebSocket

Shader Components

WaterVertexShader
- attribute position: vec3
- uniform perspective: mat4
- uniform view: mat4
- uniform model: mat4
- varying clipSpace: vec4
- varying textureCoords: vec2

Pipeline

WaterFragmentShader
- uniform refractionTexture: sampler2D
- uniform reflectionTexture: sampler2D
- uniform dudvTexture: sampler2D
- uniform normalMap: sampler2D
- uniform waterReflectivity: float
- uniform fresnelStrength: float
- uniform dudvOffset: float

MeshVertexShader
- attribute position: vec3
- attribute normal: vec3
- attribute uvs: vec2
- uniform clipPlane: vec4

Pipeline

MeshFragmentShader
- uniform tex: sampler2D
- varying vUvs: vec2

JavaScript Frontend Components

WebGLWaterApp
- gl: WebGLRenderingContext
- canvas: HTMLCanvasElement
- programs: object
- meshes: object
- textures: object
- framebuffers: object
- state: object
- ws: WebSocket

- init(): Promise<void>
- loadShaders(): Promise<void>
- loadAssets(): Promise<void>
- setupFramebuffers()
- render()
- renderWater()
- renderMeshes()
- connectWebSocket()

ShaderProgram
- program: WebGLProgram
- uniformLocations: object
- attribLocations: object

- createProgram(): WebGLProgram
- compileShader(): WebGLShader
- linkProgram()

MeshBuffer
- vertexBuffer: WebGLBuffer
- normalBuffer: WebGLBuffer
- texCoordBuffer: WebGLBuffer
- indexBuffer: WebGLBuffer
- indexCount: number

- createMeshBuffers(): MeshBuffer

Framebuffer
- framebuffer: WebGLFramebuffer
- colorTexture: WebGLTexture
- depthTexture: WebGLTexture
- width: number
- height: number

- createFramebuffer(): Framebuffer

TextureManager
- textures: Map<string, WebGLTexture>

- loadTexture(): Promise<WebGLTexture>
- bindTexture()

Rendering Pipeline:
1. Refraction pass → framebuffer
2. Reflection pass → framebuffer
3. Main scene → screen
4. Water surface with multi-texturing
5. Real-time animation loop (60fps)