AppModule
AppRoutingModule
const routes: Routes = []
AppComponent
ngOnInit()
@HostListener('window:resize', []) onResize(): void
title: string;
Width: number;
Height: number;
SceneEditorComponent
constructor(private sceneService: ThreeSceneService, private elRef: ElementRef)
ngOnInit()
private onResized(event: ResizedEvent): void
ngAfterViewInit(): void
onMaterialChange(material: Material)
public dropped(files: NgxFileDropEntry[]
public fileOver(event)
public fileLeave(event)
onSceneChange(op: string)
public ResetScene(): void
onLightChanged(light: Light): void
onSelectedTabChange(event: MatTabChangeEvent): void
@Input() Width: number
@Input() Height: number
ViewHeight = 100
ViewWidth = 100
@ViewChild('ThreeJSView', { static: false }) private threeView: SceneViewComponent;
@ViewChild('LightsLibraryEditor', { static: false }) private lightsLirbaryEditor: LightsLibraryEditorComponent;
@ViewChild('MaterialEditor', { static: false }) private materialEditor: MaterialEditorComponent;
MaterialsLibraryEditorComponent
constructor(private libraryService: MaterialLibraryService,
private confirmationDialog: MatDialog,
private sceneService: ThreeSceneService)
ngOnInit()
onAssignMaterialChange(event: MatCheckboxChange): void
onClear(): void
onClone(): void
onDelete(): void
onEditorResized(event: ResizedEvent): void
onGetFromScene(): void
onImport(event: any): void
onLightsChanged(): void
onMaterialChanged(material: Material): void
onNew(): void
onSave(): void
onSelectedTabChange(index: number)
onSelectionChange(change: MatSelectChange): void
setDragControl(force = false): void
@Output() changedMaterial: Material
Library: MaterialLibrary
Material: Material
Materials: Materials
assignMaterial: boolean
deleteDisabled: boolean
MaterialEditorComponent
constructor(private sceneService: ThreeSceneService
private confirmationDialog: MatDialog,
private libraryService: MaterialLibraryService)
ngOnInit(): void
onCopy(): void
onDelete(): void
onMaterialChanged(material: Material): void
onNewMaterial(type: MaterialType): void
onPaste(): void
onSelectionChange(change: MatSelectChange): void
onDiffuseColourChanged(colour: string): void
onEmissiveColourChanged(colour: string): void
onSpecularColourChanged(colour: string): void
onShininessChange(event: MatSliderChange): void
onRoughnessChange(event: MatSliderChange): void
onMetalnessChange(event: MatSliderChange): void
updateSelection(): void
@Output() newMaterial: Material;
@Output() changedMaterial: Material;
@Input() Material: Material
Materials: Materials
sceneJSON: string;
material = new Material(MaterialType.MESH_STANDARD);
materialCopy: Material
LightsLibraryEditorComponent
constructor(private libraryService: LightsLibraryService,
private confirmationDialog: MatDialog,
private sceneService: ThreeSceneService)
ngOnInit()
onClear(): void
onClone(): void
onDelete(): void
onImport(event: any): void
onLightChanged(light: Light): void
onNew(): void
onSave(): void
onSelectedTabChange(index: number)
onSelectionChange(change: MatSelectChange): void
@Output() changedLight: Light
Library: LightsLibrary
Light: Light
Lights: Lights
deleteDisabled: boolean
SceneComponent
constructor()
ngOnInit()
onEditorResized(event: ResizedEvent): void
onSceneChange(op: string): void
@Output() sceneChanged = new EventEmitter<string>()
NewMaterialComponent
constructor()
ngOnInit()
onNew(): void
@Output() newMaterial: MaterialType
materialType: MaterialType, materialTypes: Map<MaterialType, string>
MeshStandardMaterialEditorComponent
constructor()
ngOnInit()
onBaseMaterialChanged(material: MeshStandardMaterial): void
onColourChanged(colour: string): void
onEmissiveIntensityChanged(event: MatSliderChange): void
onMetalnessChanged(event: MatSliderChange): void
onRoughnessChanged(event: MatSliderChange): void
onWireframeChange(event: MatCheckboxChange): void
onWireframeLinewidthChanged(event: MatSliderChange): void
@Output() materialChange: MeshStandardMaterial
@Input() Material: MeshStandardMaterial
MaterialLibraryService
constructor()
getDefaultLibrary(): MaterialLibrary
importLibrary(library: MaterialLibrary): void
setCurrentMaterials(materials: Materials): void
currentMaterials: Materials Library: MaterialLibrary;
currentIndex = 0;
LightEditorComponent
constructor(private sceneService: ThreeSceneService,
private confirmationDialog: MatDialog,
private libraryService: LightsLibraryService)
ngOnInit()
onIntensityChanged(event: MatSliderChange): void
onLightChanged(event: Light): void onNewLight(type: LightType): void
onCopy(): void
onDelete(): void
onPaste(): void
onSave()
onSaveAll()
onNew()
onLoad()
onSelectionChange(change: MatSelectChange): void
unsetLightHelper(): void
updateSelection(): void
@Output() newLight: Light
@Output() changedLight: Light
lightType: typeof LightType
Lights: Lights
Light: Light
maxIntensity: number
lightCopy: Light
SceneSaverComponent
constructor(private sceneService: ThreeSceneService)
ngOnInit()
onSubmit()
getSceneJSON(): void
saveScene(gltf: any): void
message: string
exportType = ExportFileType.GLTF
SceneModifierComponent
constructor(private sceneService: ThreeSceneService)
ngOnInit()
private onNewScene(): void
private onAddToScene(): void
private onUpdateScene(): void
@Output() sceneChanged = new EventEmitter<string>()
SceneViewComponent
constructor(private sceneService: ThreeSceneService)
ngOnInit()
ngAfterViewInit()
SetMaterial(material: Material)
onOrbitControlChange(): void
Render(): void
setCameraSize(width: number, height: number): void
newScene(): void
UpdateScene(): void
@ViewChild('container', { static: true }) container: ElementRef
@Input() AreaWidth: number
@Input() AreaHeight: number
sceneJSON: string
MaterialPreviewComponent
constructor(private sceneService: ThreeSceneService)
onEditorResized(event: ResizedEvent): void
Render(): void
resetLights(): void
updateMaterial(material: Material): void
updateObject(object: THREE.Object3D): void
@Input() AreaWidth: number
@Input() AreaHeight: number
@Input() Material: Material
TextureEditorComponent
constructor(private sceneService: ThreeSceneService)
ngOnInit()
Render(): void
@Input() AreaWidth: number; @Input() AreaHeight: number; @Input() Material: Material;
BaseMaterialEditorComponent
constructor()
ngOnInit()
onOpacityChanged(event: MatSliderChange): void
onTransparentChange(event: MatCheckboxChange): void
onVisibleChange(event: MatCheckboxChange): void
@Output() materialChange: Material
@Input() Material: Material
MaterialLibrary
constructor(name: string, description: string)
clear(): void
clone(): MaterialLibrary
current: number
name: string;
materials: Materials;
ThreeSceneService
constructor(private lightsLibraryService: LightsLibraryService)
getScene(): THREE.Scene
getNewScene(): THREE.Scene
getSceneJSON(): string
getSceneMaterials(): Materials
addCurrentLights(scene: THREE.Scene): void
addFiles(files: NgxFileDropEntry[], finished: CallbackFinished): void
addFile(file: NgxFileDropEntry, files: NgxFileDropEntry[], finished: CallbackFinished): void
addColladaFile(blob: ViewerFile, file: NgxFileDropEntry, files: NgxFileDropEntry[], scene: THREE.Scene, finished: CallbackFinished): void
addGLTFFile(blob: ViewerFile, file: NgxFileDropEntry, files: NgxFileDropEntry[], scene: THREE.Scene, finished: CallbackFinished): void
addJSONFile(blob: ViewerFile, file: NgxFileDropEntry, files: NgxFileDropEntry[], scene: THREE.Scene, finished: CallbackFinished): void
addSTLFile(blob: ViewerFile, file: NgxFileDropEntry, files: NgxFileDropEntry[], scene: THREE.Scene, finished: CallbackFinished): void
cancelHideTransform(): void
delayHideTransform(): void
getDragControl(objects: THREE.Object3D[]): DragControls
setMaterial(material: THREE.Material): void
getMaterial(): THREE.Material
getLights(): Lights
getSceneBox(): THREE.Box3
getSelectableObjects(): THREE.Object3D[]
onDragHoveron(event: DragEvent): void
onTransformDraggingChanged(event: DraggingChangedEvent): void
removeObjectFromScene(object: THREE.Object3D, scene: THREE.Scene = null): void
rescaleScene(camera: THREE.OrthographicCamera = null, box: THREE.Box3 = null, orbitControls: OrbitControls = null): void
resetLights(scene: THREE.Scene = null): void
camera: THREE.OrthographicCamera
public orbitControls: OrbitControls
renderer: THREE.WebGLRenderer
transformControl: TransformControls
NewLightComponent
constructor()
ngOnInit()
onNew(): void
lightType: LightType, lightTypes: Map<LightType, string>
RectareaLightEditorComponent
constructor()
ngOnInit()
DirectionalLightHelper
constructor(light: THREE.DirectionalLight, scale: number, camera: THREE.Camera)
update(light: THREE.DirectionalLight): void
updateMatrixWorld(): void
light: THREE.DirectionalLight
positionSphere: THREE.Mesh
targetSphere: THREE.Mesh
HemisphereLightHelper
constructor(light: THREE.HemisphereLight, scale: number, camera: THREE.Camera)
update(light: THREE.HemisphereLight): void
updateMatrixWorld(): void
light: THREE.HemisphereLight
positionSphere: THREE.Mesh
PointLightHelper
constructor(light: THREE.PointLight, scale: number, camera: THREE.Camera)
update(light: THREE.PointLight): void
updateMatrixWorld(): void
light: THREE.PointLight
positionSphere: THREE.Mesh
SpotLightHelper
constructor(light: THREE.SpotLight, scale: number, camera: THREE.Camera)
update(light: THREE.SpotLight): void
updateMatrixWorld(): void
light: THREE.SpotLight
positionSphere: THREE.Mesh
targetSphere: THREE.Mesh
ConfirmationDialogComponent
constructor(public dialogRef: MatDialogRef<ConfirmationDialogComponent>,
@Inject(MAT_DIALOG_DATA) public data: DialogData)
ngOnInit()
ExportFileType
GLTF = 'gltf'
THREE_SCENE = 'three'
LightsLibraryService
constructor(private http: HttpClient,
private errorDialog: MatDialog)
getDefaultLibrary(): LightsLibrary
importLibrary(library: LightsLibrary): void
setCurrentLights(lights: Lights)
Library: LightsLibrary
currentLights: Lights
DragControls
OrbitControls
STLLoader
DraggingChangedEvent
target: TransformControls
type: string
value: boolean
ColladaLoader
DRACOLoader
GLTFLoader
ViewerFile extends File
relativePath: string
reader: CompoundReader
contentSetter: ContentSetter
service: ThreeSceneService
Three.Scene
TransformControls
DragEvent
object: THREE.Object3D
target: DragControls
type: string
Materials
clone(): Materials
name: string;
materials: Material[];
LightsLibrary
clear(): void
clone(): LightsLibrary
name: string
lights: Lights[]
ErrorDialogComponent
constructor(public dialogRef: MatDialogRef<ErrorDialogComponent>,
@Inject(MAT_DIALOG_DATA) public data: DialogData)
ngOnInit()
File
MeshPhysicalMaterial
constructor()
clone(): MeshPhysicalMaterial
copy(material: MeshPhysicalMaterial): void
clearcoat: number
clearcoatNormalMap: THREE.Texture
clearcoatNormalScale: THREE.Vector2
clearcoatRoughness: number
reflectivity: number
MeshBasicMaterial
constructor()
clone(): MeshBasicMaterial
copy(material: MeshBasicMaterial): void
alphaMap: THREE.Texture
aoMap: THREE.Texture
aoMapIntensity: number
colour: string
combine: THREE.Combine
envMap: THREE.Texture
map: THREE.Texture
morphTargets: boolean
reflectivity: number
refractionRatio: number
skinning: boolean
specularMap: THREE.Texture
wireframe: boolean
wireframeLinecap: string
wireframeLinejoin: string
wireframeLinewidth: number
MeshDepthMaterial
constructor()
clone(): MeshDepthMaterial
copy(material: MeshDepthMaterial): void
displacementMap: THREE.Texture
displacementScale: number
displacementBias: number
fog: boolean
wireframe: boolean
wireframeLinewidth: number
MeshLambertMaterial
constructor()
clone(): MeshLambertMaterial
copy(material: MeshLambertMaterial): void
alphaMap: THREE.Texture
aoMap: THREE.Texture
aoMapIntensity: number
colour: string
combine: THREE.Combine
emissive: string
emissiveMap: THREE.Texture
emissiveIntensity: number
envMap: THREE.Texture
lightMap: THREE.Texture
lightMapIntensity: number
map: THREE.Texture
morphNormals: boolean
morphTargets: boolean
reflectivity: number
refractionRatio: number
skinning: boolean
specularMap: THREE.Texture
wireframe: boolean
wireframeLinecap: string
wireframeLinejoin: string
wireframeLinewidth: number
MeshMatcapMaterial
constructor()
clone(): MeshMatcapMaterial
copy(material: MeshMatcapMaterial): void
alphaMap: THREE.Texture
bumpMap: THREE.Texture
bumpScale: number
colour: string
displacementMap: THREE.Texture
displacementScale: number
displacementBias: number
map: THREE.Texture
matcap: THREE.Texture
morphNormals: boolean
morphTargets: boolean
normalMap: THREE.Texture
normalMapType: THREE.NormalMapTypes
normalScale: THREE.Vector2
skinning: boolean
wireframe: boolean
wireframeLinewidth: number
MeshNormalMaterial
constructor()
clone(): MeshNormalMaterial
copy(material: MeshNormalMaterial): void
bumpMap: THREE.Texture
bumpScale: number
displacementMap: THREE.Texture
displacementScale: number
displacementBias: number
fog: boolean
morphNormals: boolean
morphTargets: boolean
normalMap: THREE.Texture
normalMapType: THREE.NormalMapTypes
normalScale: THREE.Vector2
skinning: boolean
MeshPhongMaterial
constructor()
clone(): MeshPhongMaterial
copy(material: MeshPhongMaterial): void
alphaMap: THREE.Texture
aoMap: THREE.Texture
aoMapIntensity: number
bumpMap: THREE.Texture
bumpScale: number
colour: string
combine: THREE.Combine
displacementMap: THREE.Texture
displacementScale: number
displacementBias: number
emissive: string
emissiveMap: THREE.Texture
emissiveIntensity: number
envMap: THREE.Texture
lightMap: THREE.Texture
lightMapIntensity: number
map: THREE.Texture
morphNormals: boolean
morphTargets: boolean
normalMap: THREE.Texture
normalMapType: THREE.NormalMapTypes
normalScale: THREE.Vector2
reflectivity: number
refractionRatio: number
shininess: number
skinning: boolean
specular: string
specularMap: THREE.Texture
wireframe: boolean
wireframeLinecap: string
wireframeLinejoin: string
wireframeLinewidth: number
MeshToonMaterial
constructor()
clone(): MeshToonMaterial
copy(material: MeshToonMaterial): void
alphaMap: THREE.Texture
aoMap: THREE.Texture
aoMapIntensity: number
bumpMap: THREE.Texture
bumpScale: number
colour: string
displacementMap: THREE.Texture
displacementScale: number
displacementBias: number
emissive: string
emissiveMap: THREE.Texture
emissiveIntensity: number
envMap: THREE.Texture
gradientMap: THREE.Texture
lightMap: THREE.Texture
lightMapIntensity: number
map: THREE.Texture
morphNormals: boolean
morphTargets: boolean
normalMap: THREE.Texture
normalMapType: THREE.NormalMapTypes
normalScale: THREE.Vector2
refractionRatio: number
shininess: number
skinning: boolean
specular: string
specularMap: THREE.Texture
wireframe: boolean
wireframeLinecap: string
wireframeLinejoin: string
wireframeLinewidth: number
PointsMaterial
constructor()
clone(): PointsMaterial
copy(material: PointsMaterial): void
colour: string
map: THREE.Texture
size: number
sizeAttenuation: boolean
ShadowMaterial
constructor()
clone(): ShadowMaterial
copy(material: ShadowMaterial): void
transparent: boolean
SpriteMaterial
constructor()
clone(): SpriteMaterial
copy(material: SpriteMaterial): void
colour: string
fog: boolean
map: THREE.Texture
rotation: number
sizeAttenuation: boolean
LineDashedMaterial
constructor()
clone(): LineDashedMaterial
copy(material: LineDashedMaterial): void
dashSize: number
gapSize: number
scale: number
LineBasicMaterial
MeshStandardMaterial
constructor()
clone(): MeshStandardMaterial
copy(material: MeshStandardMaterial): void
alphaMap: THREE.Texture
aoMap: THREE.Texture
aoMapIntensity: number
bumpMap: THREE.Texture
bumpScale: number
colour: string
displacementMap: THREE.Texture
displacementScale: number
displacementBias: number
emissive: string
emissiveMap: THREE.Texture
emissiveIntensity: number
envMap: THREE.Texture
envMapIntensity: number
lightMap: THREE.Texture
lightMapIntensity: number
map: THREE.Texture
metalness: number
metalnessMap: THREE.Texture
morphNormals: boolean
morphTargets: boolean
normalMap: THREE.Texture
normalMapType: THREE.NormalMapTypes
normalScale: THREE.Vector2
refractionRatio: number
roughness: number
roughnessMap: THREE.Texture
skinning: boolean
vertexTangents: boolean
wireframe: boolean
wireframeLinewidth: number
MaterialService
DialogData
title: string
label: string
message: string
cancel: boolean
close: boolean
DirectionalLightEditorComponent
constructor()
ngOnInit()
onChange(event: MatCheckboxChange): void
onColourChanged(colour: string): void
onPositionChange(position: Point3): void
onTargetChange(position: Point3): void
onShadowChange(event: MatCheckboxChange): void
@Output() lightChange: Light
@Input() Light: DirectionalLight
HemisphereLightEditorComponent
constructor()
ngOnInit()
onColourChanged(colour: string): void
onGroundColourChanged(colour: string): void
onPositionChange(position: Point3): void
onShadowChange(event: MatCheckboxChange): void
@Output() lightChange: Light
@Input() Light: PointLight
Lights
clone(): Lights
name: string
lights: Array<Light>
SpotLightEditorComponent
constructor()
ngOnInit()
onAngleChanged(event: MatSliderChange): void
onColourChanged(colour: string): void
onDecayChanged(event: MatSliderChange): void
onPenumbraChanged(event: MatSliderChange): void
onPositionChange(position: Point3): void
onTargetChange(position: Point3): void
onShadowChange(event: MatCheckboxChange): void
@Output() lightChange: Light
@Input() Light: SpotLight
Material
constructor(type: MaterialType)
static CreateMaterial(mat: THREE.Material): Material
clone(): Material
copy(material: Material): void
type: MaterialType
alphaTest: number
blendDst: number
blendDstAlpha: number
blendEquation: THREE.BlendingEquation
blendEquationAlpha: number
blending: THREE.Blending
blendSrc: THREE.BlendingDstFactor | THREE.BlendingSrcFactor
blendSrcAlpha: number
clipIntersection: boolean
clippingPlanes: THREE.Plane[]
clipShadows: boolean
colourWrite: boolean
depthFunc: THREE.DepthModes
depthTest: booleane
depthWrite: boolean
stencilWrite: boolean
flatShading: boolean
fog: boolean
name: string
opacity: number
polygonOffset: boolean
polygonOffsetFactor: number
polygonOffsetUnits: number
premultipliedAlpha: boolean
dithering: boolean
side = THREE.FrontSide
toneMapped: boolean
transparent: boolean
vertexColors = THREE.NoColors
visible: boolean
// ThreeJS material
material: THREE.Material
DirectionalLight
constructor()
clone(): DirectionalLight
copy(light: DirectionalLight): void
castShadow: boolean
HemisphereLight
constructor()
clone(): HemisphereLight
copy(light: HemisphereLight): void
groundColour: string
castShadow: boolean
SpotLight
constructor()
clone(): SpotLight
copy(light: SpotLight): void
castShadow: boolean
angle: number
decay: number
distance: number
penumbra: number
power: number
PointLightEditorComponent
constructor()
ngOnInit()
onColourChanged(colour: string): void
onDecayChanged(event: MatSliderChange): void
onPositionChange(position: Point3): void
onShadowChange(event: MatCheckboxChange): void
@Output() lightChange: Light
@Input() Light: PointLight
AmbientLightEditorComponent
constructor()
ngOnInit()
onColourChanged(colour: string): void
@Output() lightChange: Light
@Input() Light: Light
Colour: string
PointLight
constructor()
clone(): PointLight
copy(light: PointLight): void
castShadow: boolean
decay: number
distance: number
power: number
MaterialType
LINE_BASIC = 'Line Basic'
LINE_DASHED = 'Line Dashed'
MESH_BASIC = 'Mesh Basic'
MESH_DEPTH = 'Mesh Depth'
MESH_LAMBERT = 'Mesh Lambert'
MESH_MATCAP = 'Mesh Matcap'
MESH_NORMAL = 'Mesh Normal'
MESH_PHONG = 'Mesh Phong'
MESH_PHYSICAL = 'Mesh Physical'
MESH_STANDARD = 'Mesh Standard'
MESH_TOON = 'Mesh Toon'
POINTS = 'Mesh Points'
SHADOW = 'Shadow'
SPRITE = 'Sprite'
Point3DComponent
constructor()
ngOnInit()
onChange(): void
@Output() changePoint: Point3
@Input() Point: Point3
@Input() Label:string
@Input() enableX:boolean
@Input() enableY:boolean
@Input() enableZ:boolean
@Input() enableCheckboxes
X:number
Y:number
Z:number
Light
constructor(type: LightType)
static CreateLight(light: THREE.Light): Light
clone(): Light
copy(light: Light): void
toJSON(): LightExport
name: string
intensity: number
type: LightType
light: THREE.Light
colour: string
position: Point3
target: Point3
Point3
constructor(x: number, y: number, z: number)
X: number
Y: number
Z: number
Three.Light
LightType
AMBIENT = 'Ambient'
DIRECTIONAL = 'Directional'
HEMISPHERE = 'Hemisphere'
POINT = 'Point'
RECT_AREA = 'Rectangular Area'
SPOT = 'Spot'