Node
Node
<<Interface>>
MapClassToComponents
[className: string]: Component[]
Serializable
Node
+ name: string
+ mtxWorld: Matrix4x4
+ timestampUpdate: number
- parent: Node | null
- children: Node[]
- components: MapClassToComponents
- listener: MapEventTypeToListener
- capture: MapEventTypeToListener
- active: boolean
+ activate(_on: boolean): void
+ get isActive(): boolean
+ getParent(): Node | null
+ setParent(_parent): void
+ get cmpTransform(): ComponentTransform
+ getChildren(): Node
+ getChildren(sReName_name: string): Node[]
+ appendChild(_node: Node): void
+ removeChild(_node: Node): void
+ findChild(_node: Node): number
+ replaceChild(_replace: Node, _with: Node): boolean
+ getBranch(): IterableIterator<Node>
+ applyAnimation(_mutator: Mutator): void
+ getComponent(_class): Component
+ getComponents(T extends Component(_class: new () => T)): T[]
+ getComponent(T extends Component(_class: new () => T)): T
+ addComponent(_component: Component): void
+ removeComponent(_component: Component): void
+ serialize(): Serialization
+ deserialize(_serialization: Serialization): Serializable
+ addEventListener(_type: EVENT | string, _handler: EventListener, _capture: boolean /| AddEventListenerOptions/ = false): void
+ dispatchEvent(_event: Event): boolean
+ broadcastEvent(_event: Event): void
+ dispatchEventToTarget(_event: Event): void
- getBranchGenerator(): IterableIterator<Node>
SerializableResource
NodeResource
+ idResource: string | undefined
NodeResourceInstance
- idSource: string | undefined
+ reset(): void
+ serialize(): Serialization
+ deserialize(_serialization: Serialization): Serializable
# extl_nodeResource: NodeResource): void
Transfer
Serializer
<<Interface>>
Serializable
serialize(): Serialization
deserialize(_serialization: Serialization): Serializable
<<Interface>>
NamespaceRegister
[name: string]: Object
<<Interface>>
Serialization
[type: string]: General
Serializer
- namespace: NamespaceRegister = {"": FudgeCore}
+ registerNamespace(_namespace: Object): void
+ serializer_object: Serialization | Serializable
+ deserializer_serialization: Serialization | Serializable
+ registry(T, _type: string | string[], _type): T
+ stringify(_serialization: Serialization): string
+ parse(_string: string, _type: string | string[], _type): T
+ toString(_serialization: Serialization): string
+ parse(_type: string, _serialization: Serialization): T
+ getFullRefs(_object: Serializable): string
+ getUniqueRefs(_object: Serializable): string
+ findNamespaces(_object: Object): string[]
+ findNamespaces(_namespaces: Object, _parent: Object): string[]
Mutable
<<Interface>>
Mutator
<<Interface>>
MutatorForAnimation
<<Interface>>
MutatorAttributeTypes
<<Interface>>
MutatorForUserInterface
Mutable
+ get type(): string
+ getMutator(): Mutator
+ getMutatorForAnimation(): MutatorForAnimation
+ getMutatorForUserInterface(): MutatorForUserInterface
+ getAttributes(_mutator: Mutator): MutatorAttributeTypes
+ updateMutator(_mutator: Mutator): void
+ mutate(_mutator: Mutator): void
# reduceMutator(_mutator: Mutator): void
Coat
Coat
Coat
+ name: string
# renderData: [key: string]: unknown
+ mutate(_mutator: Mutator): void
+ useRenderData(_renderShader: RenderShader): void
+ serialize(): Serialization
+ deserialize(_serialization: Serialization): Serializable
# reduceMutator(): void
CoatColored
+ color: Color
CoatMatCap
+ texture: TextureImage
+ tintColor: Color = new Color(0.5, 0.5, 0.5, 1)
+ flatness: number
CoatTextured
+ texture: TextureImage
+ pivot: Matrix3x3 = mat
+ tint: Matrix3x3 = Mat3.IDENTITY
+ tilingX: number
+ tilingY: number
+ repetition: boolean
FileIoBrowserLocal
FileIoBrowserLocal
- selector: HTMLInputElement
+ load(): void
+ save(_files: MapFilenameToContent): void
+ async handleFileSelect(_event: Event): Promise<void>
+ async loadFiles(_files: FileList | FileList[], _loaded: MapFilenameToContent): Promise<void>
<<Interface>>
Interface
[filename: string]: string
Mutable
<<Interface>>
Mutator
<<Interface>>
MutatorForAnimation
<<Interface>>
MutatorAttributeTypes
<<Interface>>
MutatorForUserInterface
Mutable
+ get type(): string
+ getMutator(): Mutator
+ getMutatorForAnimation(): MutatorForAnimation
+ getMutatorForUserInterface(): MutatorForUserInterface
+ getAttributes(_mutator: Mutator): MutatorAttributeTypes
+ updateMutator(_mutator: Mutator): void
+ mutate(_mutator: Mutator): void
# reduceMutator(_mutator: Mutator): void
Engine
Viewport
<<Interface>>
MapLightTypeToLight.lut
[type: typeof Light]: ComponentLight[]
Viewport
- focus: Viewport
+ rectSource: Rectangle
+ camera: ComponentCamera
+ rectSource: Rectangle
+ rectDestination: Rectangle
+ frameClientToCanvas: FramingScaled = new FramingScaled()
+ frameDestinationToClient: FramingComplex = new FramingComplex()
+ frameDestinationToSource: FramingScaled = new FramingScaled()
+ frameSourceToRender: FramingScaled = new FramingScaled()
+ adjustingFrames: boolean = true
+ adjustingCamera: boolean = true
+ lights: MapLightTypeToLight.lut = null
- branch: Node = null
- crc2: CanvasRenderingContext2D = null
- canvas: HTMLCanvasElement = null
- pickBuffers: PickBuffer[] = []
+ initialize(_name: string, _branch: Node, _camera: ComponentCamera, _canvas: HTMLCanvasElement): void
+ get crc2(): CanvasRenderingContext2D
+ getCanvasRectangle(): Rectangle
+ getClientRectangle(): Rectangle
+ setBranch(_branch: Node): void
+ draw(_drawNodes): void
+ showNode(_node): void
+ createPickBuffers(): void
+ pickNode(_pos: Vector2): PickRef[]
+ pickRay(_ray): void
+ adjustCamera(): void
+ pointClientToSource(_client: Vector2): Vector2
+ pointSourceToRender(_source: Vector2): Vector2
+ pointClientToRender(_client: Vector2): Vector2
+ pointRenderToSource(_render: Vector2): Vector2
+ pointClipToClient(_normed: Vector2): Vector2
+ pointClipToCanvas(_normed: Vector2): Vector2
+ pointClientToSource(_client: Vector2): Vector2
+ get hasFocus(): boolean
+ activeInstance(): void
+ activatePointerEvent(_type: EVENT_POINTER, _on: boolean): void
+ activateKeyboardEvent(_type: EVENT_KEYBOARD, _on: boolean): void
+ activateDragDropEvent(_type: EVENT_DRAGDROP, _on: boolean): void
+ activateWheelEvent(_type: EVENT_WHEEL, _on: boolean): void
+ hndPointerEvent(_event): void
- addCanvasPointerevent: EventPointer | EventDragDrop): void
- hndPointerEvent: EventListener
- hndKeyboardEvent: EventListener
- hndWheelEvent: EventListener
+ addEventListener(_type: string, _handler: EventListener, _on: boolean): void
- hndComponentEvent(_event: Event): void
- collectLights(): void
- createSceneGraph(_fudgeNode: Node): string
ResourceManager
<<Interface>>
SerializableResource
idResource: string
<<Interface>>
Resource
[idResource: string]: SerializableResource
<<Interface>>
SerializationOfResources
[idResource: string]: Serialization
ResourceManager
+ resources: Resources
+ serialization: SerializationOfResources
+ serialization: SerializationOfResources
+ registerResource(_resource: SerializableResource): void
+ registerNewResourceFromSerialization(_serialization): void
+ isResource(_object: Serializable): boolean
+ deleteResource(_idResource: string): void
+ getSerializationOfResources(): SerializationOfResources
+ deleteUnreferencedResources(): void
+ serializeResource(_serialization: Serializable): SerializableResource
+ deserializeResource(_serialization: Serializable): SerializableResource
Color
Color
- crc2: CanvasRenderingContext2D
+ r: number
+ g: number
+ b: number
+ a: number
+ [static] getHexFromCSSKeyword(_keyword: string): string
+ [static] CSS(_keyword: string, _alpha: number = 1): Color
+ [static] RGB(_r: number, _g: number, _b: number, _a: number): Color
+ setNormRGBA(_r: number, _g: number, _b: number, _a: number): void
+ setByteRGBA(_r: number, _g: number, _b: number, _a: number): void
+ getArray(): Float32Array
+ setArrayNormRGBA(_color: Float32Array): void
+ getArrayByteRGBA(): Uint8ClampedArray
+ getArrayByteRGBA(): Uint8ClampedArray
+ add(_color: Color): void
+ getCSS(): string
# reduceMutator(_mutator: Mutator): void
Recycler
Recycler
- depot: [[type: string], Object[]]
+ getInstance(_type: new () => T): T
+ storeInstance(_instance: Object): void
+ dumpAll(): void
Material
SerializableResource
Material
+ name: string
+ idResource: string
- shaderType: typeof Shader
- coat: Coat
+ createCoreMatchingShader(): Coat
+ setCoat(_coat: Coat): void
+ getCoat(): Coat
+ setShader(_shaderType: typeof Shader): void
+ getShader(): typeof Shader
+ serialize(): Serialization
+ deserialize(_serialization: Serialization): Serializable
# reduceMutator(_mutator: Mutator): void
Light
Light
+ color: Color
+ getType(): TypeOfLight
# reduceMutator(): void
LightAmbient
LightDirectional
LightPoint
+ range: number
LightSpot
Animation
Animation
NodeResourceInstance
- idSource: string | undefined
NodeResource
Animation
+ idResource: string
+ name: string
+ labels: AnimationLabel
+ fps: number = 60
+ animationStructure: AnimationStructure
+ events: AnimationEventTrigger
- framesPerSecond: number
<<Interface>>
AnimationLabel
[name: string]: number
<<Interface>>
AnimationEventTrigger
[name: string]: number
<<Interface>>
AnimationStructure
[attribute: string]: Serialization | AnimationSequence
AnimationKey
AnimationKey
- time: number
- value: number
- constant: boolean = false
- slopeIn: number = 0
- slopeOut: number = 0
AnimationFunction
AnimationFunction
- a: number = 0
- b: number = 0
- c: number = 0
- d: number = 0
- getKey(): AnimationKey
- keyOut: AnimationKey
AnimationSequence
AnimationSequence
- keys: AnimationKey[] = []
Texture
Texture
protected reduceMutator(): void
TextureCanvas
TextureImage
+ image: HTMLImageElement
TextureSketch
TextureHTML
Math
Matrix4x4
Matrix4x4
- data: Float32Array = new Float32Array(16)
- mutator: Mutator = null
- vectors: VectorRepresentation
+ get translation(): Vector3
+ set translation(_translation: Vector3)
+ get rotation(): Vector3
+ set rotation(_rotation: Vector3)
+ get scaling(): Vector3
+ set scaling(_scaling: Vector3)
+ get IDENTITY(): Matrix4x4
+ MULTIPLICATION(_a: Matrix4x4, _b: Matrix4x4): Matrix4x4
+ INVERSE(_original: Matrix4x4): Matrix4x4
+ LOOK_AT(_transformPosition: Vector3, _targetPosition: Vector3, _up: Vector3 = Vector3.Y()): Matrix4x4
+ TRANSLATION(_translate: Vector3): Matrix4x4
+ ROTATION_X(_angleInDegrees: number): Matrix4x4
+ ROTATION_Y(_angleInDegrees: number): Matrix4x4
+ ROTATION_Z(_angleInDegrees: number): Matrix4x4
+ SCALING(_scaler: Vector3): Matrix4x4
+ PROJECTION_CENTRAL(_aspect: number, _fieldOfViewInDegrees: number, _near: number, _far: number, _direction: FIELD_OF_VIEW): Matrix4x4
+ PROJECTION_ORTHOGRAPHIC(_left: number, _right: number, _bottom: number, _top: number, _near: number = -400, _far: number = 400): Matrix4x4
+ rotate(_by: Vector3, _fromLeft: boolean = false): void
+ rotateX(_angleInDegrees: number, _fromLeft: boolean = false): void
+ rotateY(_angleInDegrees: number, _fromLeft: boolean = false): void
+ rotateZ(_angleInDegrees: number, _fromLeft: boolean = false): void
+ lookAt(_target: Vector3, _up: Vector3 = Vector3.Y()): void
+ translate(_by: Vector3): void
+ translateX(_x: number): void
+ translateY(_y: number): void
+ translateZ(_z: number): void
+ scale(_by: Vector3): void
+ scaleX(_by: number): void
+ scaleY(_by: number): void
+ scaleZ(_by: number): void
+ multiply(_matrix: Matrix4x4, _fromLeft: boolean = false): void
+ getScaling(): Vector3
+ get _to.Matrix4x4(): void
+ toString(): string
+ get(): Float32Array
+ serialize(): Serialization
+ deserialize(_serialization): Serializable
+ public getMutator(): Mutator
+ mutate(_mutator: Mutator): void
+ getMutatorAttributeType(_mutator: Mutator): MutatorAttributeTypes
# reduceMutator(_mutator: Mutator): void
# resetCache(): void
<<Interface>>
VectorRepresentation
translation: Vector3
rotation: Vector3
scaling: Vector3
Serializable
Matrix3x3
Matrix3x3
- data: Float32Array
- mutator: Mutator
- vectors: VectorRepresentation
+ get translation(): Vector2
+ set translation(_translation: Vector2)
+ get rotation(): number
+ set rotation(_rotation: number)
+ get scaling(): Vector2
+ set scaling(_scaling: Vector2)
+ PROJECTION(_width: number, _height: number): Matrix3x3
+ get IDENTITY(): Matrix3x3
+ TRANSLATION(_translate: Vector2): Matrix3x3
+ ROTATION(_angleInDegrees: number): Matrix3x3
+ SCALING(_scaler: Vector2): Matrix3x3
+ MULTIPLICATION(_a: Matrix3x3, _b: Matrix3x3): Matrix3x3
+ translate(_by: Vector2): void
+ translateX(_x: number): void
+ translateY(_y: number): void
+ scale(_by: Vector2): void
+ scaleX(_by: number): void
+ scaleY(_by: number): void
+ rotate(_angleInDegrees: number): void
+ multiply(_matrix: Matrix3x3): void
+ getScaling(): Vector2
+ get(): Float32Array
+ serialize(): Serialization
+ deserialize(_serialization): Serializable
+ getMutator(): Mutator
+ mutate(_mutator: Mutator): void
+ getMutatorAttributeType(_mutator: Mutator): MutatorAttributeTypes
# reduceMutator(_mutator: Mutator): void
# resetCache(): void
<<Interface>>
VectorRepresentation
translation: number
rotation: number
scaling: Vector2
Random
Random
- generate: Function = Math.random
- const random: Random
+ createGenerator(_seed: number): Function
+ getRaw(): number
+ getRange(_min: number, _max: number): number
+ getInteger(_min: number, _max: number): number
+ getBoolean(): boolean
+ getSign(): number
+ getElement(T(_array: Array<T>): number
+ getProperty(T(_map: Map<T, Unc>): T
+ getPropertyName(_object: Object): string
+ getPropertySymbol(_object: Object): symbol
Rectangle
<<Enum>>
ORIGIN2D
Rectangle
+ position: Vector2 = Recycler.get(Vector2)
+ size: Vector2 = Recycler.get(Vector2)
+ static GET(_x: number = 0, _y: number = 0, _width: number = 1, _height: number = 1, _origin: ORIGIN2D = ORIGIN2D.TOPLEFT): Rectangle
+ setPosition(Array(_x: number = 0, _y: number = 0, _width: number = 1, _height: number = 1, _origin: ORIGIN2D = ORIGIN2D.TOPLEFT): void
+ getRectangleIntersection(_target: Rectangle): Vector2
+ get copy(): Rectangle
+ minmax(_point: Vector2): boolean
+ collides(_rect: Rectangle): boolean
+ toString(): string
# reduceMutator(_mutator: Mutator): void
Vector2
Vector2
- data:Float32Array;
+ ZERO(): Vector2;
+ ONE(_scale: number = 1): Vector2;
+ X(_scale: number = 1): Vector2;
+ Y(_scale: number = 1): Vector2;
+ TRANSformation(_vector: Vector2, _matrix: Matrix3x3, _includeTranslation: boolean = true): Vector2;
+ NORMALIZATION(_vector: Vector2, _length: number = 1): Vector2;
+ SCALE(_vector: Vector2, _scale: number): Vector2;
+ SUM(_...vectors: Vector2[]): Vector2;
+ DIFFERENCE(_a: Vector2, _b: Vector2): Vector2;
+ DOT(_a: Vector2, _b: Vector2): number;
+ SCALED(_vector: Vector2, _scalar: number): Vector2;
+ EQUals(_a: Vector2, _b: Vector2): boolean;
+ DOT(_a: Vector2, _b: Vector2): number;
+ REFLECTION(_incention: Vector2, _normal: Vector2): Vector2;
+ equals(_compare: Vector2, _tolerance: number = Number.EPSILON): boolean
+ add(_addend: Vector2): void
+ subtract(_subtrahend: Vector2): void
+ scale(_scaler: number): void
+ normalize(_length: number = 1): void
+ set(_x: number = 0, _y: number = 0): void
+ get(): Float32Array
+ get copy(): Vector2;
+ transform(_matrix: Matrix4x4, _includeTranslation: boolean = true): void
+ toVector3(): Vector3
+ reflect(_normal: Vector3): void
+ toString(): string
+ getMutator(_value: number, index: number, array: Float32Array) => number): Vector3
+ getMutator(): Mutator
# reduceMutator(_mutator: Mutator): void
Vector3
Vector3
- data: Float32Array
+ X(_scale: number = 1): Vector3
+ Y(_scale: number = 1): Vector3
+ Z(_scale: number = 1): Vector3
+ ZERO(): Vector3
+ ONE(_scale: number = 1): Vector3
+ TRANSformation(_vector: Vector3, _matrix: Matrix4x4, _includeTranslation: boolean = true): Vector3
+ NORMALIZATION(_vector: Vector3, _length: number = 1): Vector3
+ SUM(_...vectors: Vector3[]): Vector3
+ DIFFERENCE(_a: Vector3, _b: Vector3): Vector3
+ SCALED(_vector: Vector3, _scalar: number): Vector3
+ CROSS(_a: Vector3, _b: Vector3): Vector3
+ DOT(_a: Vector3, _b: Vector3): number
+ REFLECTION(_incention: Vector3, _normal: Vector3): Vector3
+ equals(_compare: Vector3, _tolerance: number = Number.EPSILON): boolean
+ add(_addend: Vector3): void
+ subtract(_subtrahend: Vector3): void
+ scale(_scaler: number): void
+ normalize(_length: number = 1): void
+ set(_x: number = 0, _y: number = 0, _z: number = 0): void
+ get(): Float32Array
+ get copy(): Vector3;
+ transform(_matrix: Matrix4x4, _includeTranslation: boolean = true): void
+ toVector2(): Vector2
+ reflect(_normal: Vector3): void
+ toString(): string
+ getMutator(_value: number, index: number, array: Float32Array) => number): Vector3
+ getMutator(): Mutator
# reduceMutator(_mutator: Mutator): void
Framing
<<Interface>>
Border
left: number
right: number
top: number
bottom: number
Framing
+ getPoint(_pointInFrame: Vector2, _rectFrame: Rectangle): Vector2
+ getPointInverse(_point: Vector2, _rect: Rectangle): Vector2
+ getRect(_rectFrame: Rectangle): Rectangle
# reduceMutator(_mutator: Mutator): void
FramingScaled
+ nominWidth: number = 1.0
+ nominHeight: number = 1.0
+ setScale(_nominWidth: number, _nominHeight: number): void
+ getPoint(_pointInFrame: Vector2, _rectFrame: Rectangle): Vector2
+ getPointInverse(_point: Vector2, _rect: Rectangle): Vector2
+ getRect(_rectFrame: Rectangle): Rectangle
FramingComplex
+ margin: Border = {left: 0, top: 0, right: 0, bottom: 0 }
+ padding: Border = {left: 0, top: 0, right: 0, bottom: 0 }
+ getPoint(_pointInFrame: Vector2, _rectFrame: Rectangle): Vector2
+ getPointInverse(_point: Vector2, _rect: Rectangle): Vector2
+ getRect(_rectFrame: Rectangle): Rectangle
+ getMutator(): Mutator
FramingFixed
+ width: number = 300
+ height: number = 150
+ setSize(_width: number, _height: number): void
+ getPoint(_pointInFrame: Vector2, _rectFrame: Rectangle): Vector2
+ getPointInverse(_point: Vector2, _rect: Rectangle): Vector2
+ getRect(_rectFrame: Rectangle): Rectangle
Component
ComponentMaterial
+ material: Material
ComponentMesh
+ pivot: Matrix4x4
+ mesh: Mesh
ComponentTransform
+ local: Matrix4x4
ComponentScript
ComponentLight
+ pivot: Matrix4x4 = Matrix4x4.IDENTITY
+ light: Light
+ setType(T extends Light=(_class: new () => T)): void
Component
Component
singleton: boolean
- container: Node | null
- active: boolean
+ activate(_on: boolean): void
+ get isActive(): boolean
+ get isSingleton(): boolean
+ get isComponent(): boolean
+ setContainer(_container: Node | null): void
+ serialize(): Serialization
+ deserialize(_serialization): Serializable
# reduceMutator(_mutator: Mutator): void
Serializable
ComponentTransform
+ local: Matrix4x4
# reduceMutator(_mutator: Mutator): void
ComponentAnimator
<<Interface>>
ANIMATION_PLAYMODE
<<Interface>>
ANIMATION_PLAYBACK
ComponentAnimator
animation: Animation
playmode: ANIMATION_PLAYMODE
playback: ANIMATION_PLAYBACK
speedScaleWhiteGlobalSpeed: boolean
- localTime: TIME
- speedScale: number = 1
- lastTime: number = 0
- updateAnimation(cmpAc_e: Event, _time: number): [Mutator, number]
- executeEvents(_events: string[]): void
- applyPlaymode(_time: number): number
- updateDirection(_time: number): number
- updateScale(): void
ComponentAudioListener
ComponentAudioListener
+ field: type
+ method(type): type
ComponentCamera
<<Enum>>
FIELD_OF_VIEW
<<Enum>>
PROJECTION
ComponentCamera
+ pivot: Matrix4x4 = Matrix4x4.IDENTITY
- backgroundColor: Color
- backgroundEnabled: boolean
- transform: Matrix4x4
- fieldOfView: number
- projection: PROJECTION
- direction: FIELD_OF_VIEW
- backgroundColor: Color
- backgroundEnabled: boolean
+ get Projection(): PROJECTION
+ getBackgroundEnabled: boolean
+ get Transform()
+ getFieldOfView(): number
+ getDirection(): FIELD_OF_VIEW
+ get ViewProjectionMatrix(): Matrix4x4
+ projectCentral(_aspect: number = this.aspectRatio, _fieldOfView: number = this.fieldOfView, _direction: FIELD_OF_VIEW = this.direction): void
+ projectOrthographic(_left: number = -1, _right: number = RenderManager.getCanvas().clientWidth, _bottom: number = RenderManager.getCanvas().clientHeight, _top: number = 0): void
+ getProjectionRectangle(): Rectangle
+ project_pointInWorldSpace: Vector3): Vector3
+ getMutatorAttributeTypes(_mutator: Mutator): MutatorAttributeTypes
+ mutate(_mutator: Mutator): void
# reduceMutator(_mutator: Mutator): void
ComponentAudio
ComponentAudio
+ field: type
+ method(type): type
Time
<<Enum>>
LOOP_MODE
Object
Time
<<Interface>>
Timers
[if number]: Timer
Timer
Timer
- active: boolean
+ count: number
- handler: TimerHandler
+ cycle: number
+ elapse: number
- event: EventTimer
- timeoutRef: number
- isWindow: number
+ get id(): number
+ get remain(): number
+ setLeft(): Timer
[← clear(): void
Time
- (nameTime: Time = new Time())
+ name: Time
- scale: number
- offset: number
- lastCallToExpired: number
- timers: Timers = {}
- nTimerNext: number = 0
+ get name(): Time
+ get time, milliseconds: number): TimeUnits
+ get(): number
+ getBetween(_to: number): number
+ set(_time: number = 0): void
+ setScale(_scale: number = 1.0): void
+ getScale(): number
+ getOffset(): number
+ getElapsedSincePreviousCall(): number
- delay(_lapse: number): Promise<void>
+ clearAllTimers(): void
+ setTimeout(_lapse: number, _id: number): void
+ setTimeout(_lapse: number, _count: number, _handler: TimerHandler, _...arguments: Object[]): number
+ deleteTimer(_id: number): void
+ getTimers(): Timers
+ hasTimers(): boolean
- recalcAllTimers(): void
Loop
Loop
+ timeStartGame: number
+ timeStartReal: number
+ timeFrameGame: number
+ timeFrameReal: number
- timeLastFrameGame: number
- timeLastFrameReal: number
- timeLastFrameGameRun: number
- timeLastFrameRealRun: number
- running: boolean
- mode: LOOP_MODE
- timeScaled: number
- idRequest: number
- frameRequested: number
- framesToAnalyzeFps: number
- (mode:LOOP_MODE, _framesToAnalyzeFps: number)
+ start(_mode: LOOP_MODE = LOOP_MODE.FRAME_REQUEST, _fps: number = 60, _evoke: boolean = false): void
+ stop(): void
+ getFrameRequestAnimation(): number
+ getFpsRealAverage(): number
- loop(): void
- loopFrame(): void
- loopTime(): void
Ray
Ray
+ origin: Vector3
+ direction: Vector3
+ length: number
RayHit
+ node: Node
+ face: number
+ zBuffer: number
Shader
Shader
+ getCoat(): typeof Coat
+ getVertexShaderSource(): string
+ getFragmentShaderSource(): string
ShaderUniColor
+ getCoat(): typeof Coat
+ getVertexShaderSource(): string
+ getFragmentShaderSource(): string
ShaderMatCap
+ getCoat(): typeof Coat
+ getVertexShaderSource(): string
+ getFragmentShaderSource(): string
ShaderTexture
+ getCoat(): typeof Coat
+ getVertexShaderSource(): string
+ getFragmentShaderSource(): string
ShaderRayCast
+ getVertexShaderSource(): string
+ getFragmentShaderSource(): string
ShaderFlat
+ getCoat(): typeof Coat
+ getVertexShaderSource(): string
+ getFragmentShaderSource(): string
Debug
Debug
- delegates: [[filter: number], MapDebugTarget[]]
+ setFilter(_target: DebugTarget, _filter: DEBUG_FILTER): void
+ addFilter(_target: DebugTarget, _filter: DEBUG_FILTER): void
+ error(_message: Object, _error: Object): void
+ warn(_message: Object, _error: Object): void
+ fail(_message: Object, _error: Object): void
+ clear(): void
+ clear(_target: string): void
+ groupCollapsed(_name: string): void
+ groupEnd(): void
- delegate(_filter: DEBUG_FILTER, _message: Object, _error: Object): void
<<Interface>>
MapDebugTargetToDelegate
<<Enum>>
DEBUG_FILTER
<<Interface>>
MapDebugFilterToDelegate
DebugTarget
- delegates: MapDebugFilterToDelegate
+ mapKeyArrayToArray(_message: Object
DebugAlert
- delegates: MapDebugFilterToDelegate
+ createDelegate(_heading: string): Function
DebugTextarea
+ textArea: HTMLTextAreaElement
- delegates: MapDebugFilterToDelegate
- groups: string[]
+ clear(): void
+ group(_name: string)
+ groupEnd(): void
+ addDelegate(_type: string, _heading: string): Function
- setDelegate(_level: number): string
+ print(_text: string): void
DebugDialog
DebugConsole
- delegates: MapDebugFilterToDelegate
Mesh
Mesh
- vertices: Float32Array
+ Indices: Uint16Array
- textureUVs: Float32Array
- normalFace: Float32Array
+ idResource: string
+ getBufferSpecification(): BufferSpecification
+ getVertices(): Float32Array
+ getIndexCount(): number
+ serialize(): Serialization
+ deserialize(_serialization): Serializable
+ abstract create(): void
+ abstract getVertices(): Float32Array
+ abstract getTextureUVs(): Float32Array
+ abstract getIndices(): Uint16Array
+ abstract getNormalsFace(): Float32Array
SerializableResource
MeshQuad
+ create(): void
+ createVertices(): Float32Array
+ createIndices(): Uint16Array
+ createTextureUVs(): Float32Array
+ createFaceNormals(): Float32Array
MeshCube
+ create(): void
+ createVertices(): Float32Array
+ createIndices(): Uint16Array
+ createTextureUVs(): Float32Array
+ createFaceNormals(): Float32Array
MeshSprite
+ create(): void
+ createVertices(): Float32Array
# createIndices(): Uint16Array
# createTextureUVs(): Float32Array
# createFaceNormals(): Float32Array
MeshPyramid
+ create(): void
+ createVertices(): Float32Array
+ createIndices(): Uint16Array
+ createTextureUVs(): Float32Array
+ createFaceNormals(): Float32Array
Browser
EventTarget
HTMLCanvasElement
WebGL2RenderingContext
CanvasRenderingContext2D
Event
Event
EventTarget
<<Enum>>
EventPointer
<<Enum>>
EventWheel
<<Enum>>
EventDragDrop
<<Enum>>
EventKeyboard
<<Enum>>
EventListener
<<Enum>>
Event
<<Interface>>
MapEventTypeToListener
[eventType: string]: EventListener[]
EventTargetStatic
targetStatic: EventTargetStatic
+ addEventListener(_type: string, _handler: EventListener): void
+ removeEventListener(_type: string, _handler: EventListener): void
+ dispatchEvent(_event: Event): Boolean
EventTarget
DragEvent
EventDragDrop
+ pointerX: number
+ pointerY: number
+ canvasX: number
+ canvasY: number
+ clientRect: ClientRect
<<Enum>>
EVENT_DRAGDROP
EventKeyboard
EventKeyboard
<<Enum>>
EVENT_KEYBOARD
<<Enum>>
KEYBOARD_CODE
KeyboardEvent
EventTimer
<<Enum>>
EVENT_TIMER
EventTimer
+ type: EVENT_TIMER = EVENT_TIMER.CALL
+ target: Timer
+ arguments: Object[]
+ firstCall: boolean = true
+ lastCall: boolean = false
EventPointer
<<Enum>>
EVENT_POINTER
EventPointer
+ pointerX: number
+ pointerY: number
+ canvasX: number
+ canvasY: number
+ clientRect: ClientRect
PointerEvent
EventWheel
<<Enum>>
EVENT_WHEEL
EventWheel
WheelEvent
Render
RenderInjector
RenderInjector
- injections: [[className: string], CoatInjection]
+ decorateCoat(_constructor: Function): void
- checkRenderDataForCoatColoredThis(_Coat: RenderShader): RenderShader | void
- checkRenderDataForCoatMatCapThis(_Coat: RenderShader): RenderShader | void
- checkRenderDataForCoatTexturedThis(_Coat: RenderShader): RenderShader | void
CoatInjection
provides
RenderOperator
RenderOperator
crc3: WebGL2RenderingContext
- rectViewport: Rectangle
- renderShaderRayCast: RenderShader
+ assertTrue(_value: T | null, _message: string = ""): T
+ initialize(_antialias: boolean = false, _alpha: boolean = true): void
+ getCanvasRectangle(): Rectangle
+ getRenderingContext(): WebGL2RenderingContext
+ setViewportRect(_rectangle: Rectangle, _height: number): void
+ getViewportRectangle(_rect: Rectangle): void
+ resetViewportRectangle(): Rectangle
# createRenderLights(_lights: MapLightTypeToLightList): RenderLights
# drawLights(_renderShader: RenderShader, _renderLights: RenderLights, _uniformsLightList): void
# draw(_renderShader: RenderShader, _renderBuffers: RenderBuffers, _uniformsCoat: RenderCoat, _world: Matrix4x4, _projection: Matrix4x4): void
# drawGizmos(_renderShader: RenderShader, _renderBuffers: RenderBuffers, _world: Matrix4x4, _projection: Matrix4x4): void
# createProgram(_shaderClass: typeof Shader): RenderShader
# useProgram(_shaderInfo: RenderShader): void
# deleteProgram(_shaderInfo: RenderShader): void
# deleteBuffers(_mesh: Mesh): RenderBuffers
# getBuffers(_renderBuffers: RenderBuffers): void
# deleteBuffers(_renderBuffers: RenderBuffers): void
# createParameter(_coat: Coat): RenderCoat
# updateParameter(_coat: RenderCoat): void
# deleteParameter(_coat: RenderCoat): void
# setAttributeStructure(_attributeLocation: number, _bufferSpecification: BufferSpecification): void
<<Interface>>
BufferSpecification
size: number
dataType: number
normalize: boolean
stride: number
offset: number
<<Interface>>
RenderCoat
coat: Coat
<<Interface>>
RenderShader
program: WebGLProgram
attributes: [[name: string], number]
uniforms: [[name: string], WebGLUniformLocation]
<<Interface>>
RenderBuffers
vertices: WebGLBuffer
indices: WebGLBuffer
nIndices: number
textureUVs: WebGLBuffer
normalsFace: WebGLBuffer
<<Interface>>
RenderLights
[type: string]: Float32Array
RenderManager
RenderManger
+ rectClip: Rectangle = new Rectangle(-1, 1, 2, -2)
- renderShaders: Map<typeof Shader, Reference<RenderShader>> = new Map()
- renderCoats: Map<Coat, Reference<RenderCoat>> = new Map()
- renderBuffers: Map<Mesh, Reference<RenderBuffers>> = new Map()
- nodes: MapNodeToNodeDeReferences = new Map()
- lastUpdate: number
- pickBuffers: PickBuffer[]
+ addNode(_node: Node): void
+ addBranch(_node: Node): boolean
+ addEventsTo(_node: Node): void
+ removeBranch(_node: Node): void
+ updateNode(_node: Node): void
+ updateBranch(_node: Node): void
+ setLights(_lights: MapLightTypeToLightList): void
+ get Node(): Node
+ clear(_color: Color = null): void
+ resetFrameBuffer(_color: Color = null): void
+ drawBranch(_node: Node, _cmpCamera: ComponentCamera, _drawNode: Function = RenderManager.drawNode): void
+ drawBranchForRayCast(_node: Node, _cmpCamera: ComponentCamera): PickBuffer[]
+ drawNode(_node: Node, _finalTransform: Matrix4x4, _projection: Matrix4x4): void
+ drawNodeForRayCast(_node: Node, _finalTransform: Matrix4x4, _projection: Matrix4x4): void
+ drawNodeForRayCast(_node: Node, _finalTransform: Matrix4x4, _projection: Matrix4x4): void
+ getRayCastTexture(): WebGLTexture
+ recalculateAllNodeTransforms(): void
+ recalculateTransformsOfNodeAndChildren(_node: Node, _world: Matrix4x4): void
+ removeReference<KeyType, ReferenceType>(_in: Map<KeyType, Reference<ReferenceType>>, _key: KeyType, _deletor: Function): void
+ createReference<KeyType, ReferenceType>(_in: Map<KeyType, Reference<ReferenceType>>, _key: KeyType, _creator: Function): void
<<Interface>>
NodeReferences
shader: typeof Shader
coat: Coat
mesh: Mesh
<<Interface>>
PickBuffer
node: Node
texture: WebGLTexture
frameBuffer: WebGLFrameBuffer
Reference_
- reference: T
- count: number
+ getReference(): T
+ decreaseCounter(): number
+ decreaseCounter(): number