**ML PLAK**

**Documentación sobre Visor 3D**

**Autor: Marcelo Tuller**

**Correo electronico:** [**marscrophimself@protonmail.com**](mailto:marscrophimself@protonmail.com)

**Indice**

**1 …………………………………………………………………………………………… Instalacion en el entorno local**

**5 …………………………………………………………………………………………… Como usar el entorno de** [**diseño**](https://www.google.com/search?sxsrf=AOaemvLDuFYz3Ab63gWlrGNGswPBT50XNQ:1639665066999&q=dise%C3%B1o&spell=1&sa=X&ved=2ahUKEwjmpfmuxOj0AhVBp5UCHYypDI4QBSgAegQIAhA2)

**6 …………………………………………………………………………………………… Compatibilidad con Three.js**

**7 …………………………………………………………………………………………… Viewer**

**22 …………………………………………………………………………………………… Room**

**26 …………………………………………………………………………………………… Contribuir a este documento**

**Instalacion en el entorno local**

**Dependencias:**

* NVM: esta dependencia sirve para ajustar las versiones de NodeJS y NPM en la raiz del proyecto
* NodeJS: debido a dependencias funcionales, el sistema requiere utilizar en la raiz del proyecto la version 12.5.0 de NodeJS. Para conocer la version de NodeJS que se utiliza en la raiz de su proyecto puede abrir una interfaz de comandos y utilize node -v, comando que retorna la version de NodeJS que se utiliza para instalar paquetes en la raiz del proyecto. Se puede cambiar la version de la misma usando nvm use <Out: node-v>.
* NPM: debido a dependencias funcionales, se necesita utilizar la version 6.9.0. La dependencia sirve para instalar nuevas versiones de NodeJS y tambien para instalar las dependencias de JS (tales como VueJS o las especificas de Typescript).

**Configuracion de entorno:**

Para instalar y trabajar correctamente con el entorno local es necesario tener un servidor activo de ML PLAK que procese todas las solicitudes que se hacen dentro del entorno. Para eso se utiliza la opcion BACKEND\_BASE\_URL que esta en config/dev.env.js . Si la misma apunta a un servidor que tiene problemas para procesar las solicitudes que necesitamos es necesario ajustar esta variable por otra funcional o ver si es necesario debuggear el servidor de base.

La opcion por defecto para correr la aplicacion es <https://staging.mlplak.com:442/server> la cual procesara todas las solicitudes entre el entorno local y el subdominio en ML PLAK.

**Pasos para instalar:**

* En lo posible, verifique la integridad de las dependencias en package.json, dado que algunas dependencias estan forzadas a implementar una version que depreca a las demas dependencias automaticamente, por lo que se debe sacar en esos casos el uso de ^ y de esa manera no instalar una version relativa a la que se busca.
* Utilice **npm install** para instalar las dependencias. Si no hay dependencias deprecando al resto, la instalacion debe ser valida
* Utilice **npm run dev** para utilizar la aplicacion en localhost:8080

**Como usar el entorno de** [**diseño**](https://www.google.com/search?sxsrf=AOaemvLDuFYz3Ab63gWlrGNGswPBT50XNQ:1639665066999&q=dise%C3%B1o&spell=1&sa=X&ved=2ahUKEwjmpfmuxOj0AhVBp5UCHYypDI4QBSgAegQIAhA2)

**Inicio de sesion:**

* Para iniciar sesion en ML PLAK es necesario tener credenciales validas. Se puede hablar con alguien del equipo que genere las mismas en la base de datos remota y las asigne al desarrollador.

**Uso del Visor 3D:**

* Una vez que ingresa a la seccion **Listado de asistencias** seleccione el recuadro en el tope izquierdo e ingrese a la seccion **Vistas de** [**diseño**](https://www.google.com/search?sxsrf=AOaemvLDuFYz3Ab63gWlrGNGswPBT50XNQ:1639665066999&q=dise%C3%B1o&spell=1&sa=X&ved=2ahUKEwjmpfmuxOj0AhVBp5UCHYypDI4QBSgAegQIAhA2)
* La seccion redigira automaticamente al editor 3D, en el cual se crean y actualizan nuevos muebles y habitaciones.

**Compatibilidad con Three.js**

Los objetos graficos del visor en ML PLAK utilizan Three.js para importar objetos y crear otros propios de ML PLAK. Las funciones de $store (las que sirven para actualizar estados de los nodos/componentes de VueJS) estan en Typescript, mientras que los objetos que realizan las operaciones graficas estan en Three.js

**Viewer**

El Visor 3D configura, prepara y pone a la vista todos los cambios graficos que se realizan en el editor.

## Constructor

### Viewer(container, existsCanvas, payloadCamera, config, fireEvents)

container - un elemento en el DOM donde todos los procesos graficos se llevan a cabo

existsCanvas - Boolean que sirve para ajustar las dimensiones del visor

payloadCamera - Camera (vista truncada) existente

config - diccionario de parametros de configuracion del Visor. Los mismos son:

enablePieces: Boolean que indica la renderizacion por piezas

enableWireframe: Boolean que indica la vista estructurada de contornos

enableTextures: Boolean que indica el uso de texturas, las cuales son imagenes rasterizadas sobre la superficie de un objeto 3D

enableLights: (deprecada, usar luces de sala) Boolean que indica el uso de luces de ambiente

otherPassType: String que sirve para cambiar la calidad de la imagen (utiliza SAOPass por defecto)

renderPassType: String que indica el tipo de composicion de imagen (renderPass por defecto)

ShowGrid: Boolean que indica mostrar/ocultar la rejilla en el editor

fireEvents - parametro que indica limpiar labels en el visor

## Propiedades

### .renderer: WebGLRenderer

Abstraccion de un renderizador WebGL con sombras ajustables

### .centerPivot : Object3D

Abstraccion de un objeto 3D que sirve para pivotar el centro de rotacion

### .group : Group

Coleccion de todos los modelos de ML PLAK que se estan en la grafica

### .labels : Group

Coleccion de todas las etiquetas del visor

### .dimensionLabels : Group

Coleccion de las etiquetas de dimension del visor

### .gridHelper : GridHelper

Este atributo sirve para configurar la rejilla

### .materials : Dictionary

Este diccionario define tres tipos de materiales que el visor soporta: ‘wood’ (Mesh que simula la madera), ‘wireframe’ (LineBasicMaterial que simula lineas gruesas), ‘grid’ (LineBasicMaterial que simula la vista de bordes)

### .textureSize : Dictionary

Este diccionario sirve para configurar el tamaño de la textura

### .Export\_scale\_factor : Float

Esta variable sirve para configurar la escala en la cual se exporta la edicion a DAE

### .currentActiveView : String

Esta variable configura el tipo de vista/camara (perspectiva, frente, trasera, izquierda, derecha)

### .editorModelsList : null

Esta variable guarda todos los modelos en el editor

### .scene : Scene

Este array guarda una secuencia de objetos y luces de la vista grafica en la cual se incluyen, entre otros, el centro de pivot de la camara, el grupo, las etiquetas, luces , las etiquetas dimensionales y la configuracion de rejilla

### .composer : EffectComposer

Abstraccion de EffectComposer que sirve para componer la imagen del renderizador

### .autoRotate: bool

Variable para rotar automaticamente

### .controls: OrbitControls

Abstraccion de controles orbitales para rotar la vista truncada alrededor del editor

### .orthographicCamera: OrtographicCamera

Abstraccion de la camara ortografica (mantener las dimensiones a cualquier distancia) para el editor

## Methods

### .setupRenderer ( )

Esta funcion crea un nuevo renderizador

### .setupContainer (container: DOM object, existsCanvas: Bool)

Esta funcion permite que el contenedor herede las propiedades de la vista

### .setupCamera (payloadCamera: Camera)

Esta funcion actualiza la camara en el visor 3D

### .setupCenterPivot ()

Esta funcion crea un centro de pivot para el visor

### .setupModelsGroup ()

Esta funcion crea una coleccion de modelos para el visor

### .setupLabels ()

Esta funcion crea etiquetas para el visor

### .setupDimensionLabels ()

Esta funcion crea etiquetas de dimension para el visor

### .setupGridHelper ()

Esta funcion crea la configuracion de rejillas del visor

### .setupMaterials ()

Esta funcion crea los materiales para el visor

### .setupConfig ()

Esta funcion crea el diccionario de configuracion para el visor

### .setupFireEvents ()

Esta funcion crea la limpieza de etiquetas

### .setupGlobalAttributes ()

Esta funcion crea propiedades globales para el visor

### (deprecada) .setupLights ()

Esta funcion crea luces del visor

### .setupScene ()

Esta funcion crea la escena del visor

### .setupComposer ()

Esta funcion crea el compositor del visor

### .setupListeners ()

Esta funcion crea eventos para escuchar las interacciones con el cursor

### .renderFrame ()

Esta funcion rasteriza todas las variables en el compositor y lo reinicia

### .setRenderPass ()

Esta funcion configura el renderPass del compositor

### .setShaderPass ()

Esta funcion configura el shaderPass del compositor

### .getSelectedItemsOutlinePass ()

Esta funcion configura el outlinePass del compositor

### .cubeDr (a: Float de tamaño, x: Float, y: Float, z:Float)

Esta funcion retorna un cubo con posicion y sombra

### .loadTextures (textures: Objetos de imagenes rasterizadas)

Esta funcion retorna un cubo con posicion y sombra

### .refreshTextures ()

Esta funcion recarga las texturas en el visor

### .addLight (x: Float, y: Float, z: Float)

Esta funcion carga una luz de punto en el visor

### .createModels (data: Dictionary)

Esta funcion utiliza un diccionario para crear un modelo nuevo

### .findModelByTag (tag: String)

Esta funcion sirve para buscar modelos por etiqueta

### .removeModel (id: String)

Esta funcion sirve para remove un model de acuerdo a su ID

### .addModel (data: Dictionary)

Esta funcion sirve para crear un nuevo modelo con un diccionario de datos

### .createModel (data: Dictionary)

Esta funcion retorna un modelo con un diccionario de datos

### .rotate (r: Float)

Esta funcion sirve para cambiar el centro de pivot dado un angulo de rotacion

### .createPiece (data: Dictionary)

Esta funcion sirve para crear una pieza nueva con un diccionario de datos

### .updatePiece (data: Dictionary)

Esta funcion sirve para actualizar una pieza nueva con un diccionario de datos

### .updateMLPlackPiece (data: Dictionary)

Esta funcion sirve para actualizar una pieza nueva de ML PLAK

### .updateMLPlackModel (data: Dictionary)

Esta funcion sirve para actualizar un modelo nuevo de ML PLAK

### .clearGroup ()

Esta funcion sirve para crear un grupo nuevo en el visor

### .loadAllModulesForRender (modules: Dictionary)

Esta funcion sirve para limpiar la sesion previa y actualizar las piezas y el modelo en ML PLAK

### .loadAllModulesForRender (modules: Dictionary)

Esta funcion sirve para limpiar la sesion previa y actualizar las piezas y el modelo en ML PLAK

### .updateModel (d: Dictionary que guarda datos del modelo, visibilidad y posiciones)

Esta funcion sirve para actualizar un modelo especifico con datos de posicion

### .updateModelVisibility (d: Dictionary que guarda datos del modelo y visibilidad)

Esta funcion sirve para actualizar la visibilidad de un modelo especifico con datos de posicion

### .rotateModel (group: grupo a rotar, data: Diccionario con variables de rotacion)

Esta funcion sirve para rotar un grupo segun coordenadas

### .setEdgeColor (v: String de color hexagesimal)

Esta funcion sirve para cambiar el color de un contorno

### .selectItem (selected: Objeto seleccionado)

Esta funcion sirve para seleccionar un objeto en el visor con un outlinePass

### .getModuleByName (moduleName: String de modulo)

Dado que el visor abstrae un modulo de ML PLAK, esta funcion retorna el modulo en ML PLAK como visor

### .selectItemById (id: String de item)

Esta funcion selecciona un item en ML PLAK por su ID

### .toggleRotation (rotate: Bool)

Esta funcion activa o desactiva la rotacion automatica durante un toque

### .createLine (s,d)

Esta funcion crea dos lineas en la rejilla de ML PLAK

### .createOrigin ()

Esta funcion crea puntos de origen en el visor

### .toggleGrid (show: Boolean)

Esta funcion mantiene visible la rejilla en el visor durante un toque

### .exportToObj (filename: String de nombre de archivo, callback: Funcion a ejecutar despues de exportar)

Esta funcion guarda el objeto en el editor con todos los objetos de Mesh

### .exportToObjDAE (filename: String de nombre de archivo, callback: Funcion a ejecutar despues de exportar)

Esta funcion guarda en un archivo DAE en el editor con todos los objetos de Mesh

### .exportToGLTF (callbackWithResult: Funcion a ejecutar despues de exportar)

Esta funcion guarda en un archivo GLTF en el editor con todos los objetos de Mesh

### .loadGLTF (resourceUrl: String de URL)

Esta funcion carga una edicion de ML PLAK guardada en GLTF

### .togglePiecesVisibility (v: String de visibilidad)

Esta funcion activa o desactiva la visibilidad durante un toque

### .exportImage (filename: String de nombre de archivo)

Esta funcion exporta una edicion a un archivo

### .getCurrentImage ()

Esta funcion sirve para cambiar la imagen cargada en el visor

### .updateRoomElement (data: Dictionary)

Esta funcion sirve para actualizar la Habitacion de la edicion segun las variables

### .removeRoomElement (data: Dictionary)

Esta funcion busca el tag del modelo en el diccionario y lo elimina

### .rotateAroundWorldAxis (object: Objecto en el visor, axis: Objeto con ejes, radians: Float de angulo)

Esta funcion rota un objeto dado segun los ejes y luego lo rota a los angulos dados

### .extractTextures (data: Dictionary con piezas)

Esta funcion extrae la textura de la pieza seleccionada

### .createModelLabel (model: Objeto de ML PLAK)

Esta funcion utiliza la etiqueta del modelo como etiqueta de la edicion

### .showLabels (show: Boolean)

Esta funcion cambia la visibilidad de las etiquetas

### .updateModelLabel (model: Objeto en el visor)

Esta funcion actualiza la etiqueta de un modelo de ML PLAK

### .showDimensions (show: undefined,m: Objeto de ML PLAK)

Esta funcion sirve para cambiar las etiquetas de un modelo de ML PLAK

### .getPatternImg (m: Objeto de ML PLAK)

Esta funcion sirve para cambiar la imagen en el editor

### .getSelectedObject ()

Esta funcion retorna el objeto seleccionado

### .clearSelection ()

Esta funcion retorna una seleccion indefinida

### .clearOrtographicCamera ()

Esta funcion elimina la camara ortografica

### .cameraOrtographic (view: Visor en un componente)

Esta funcion realiza una camara ortografica sobre el componente visualizado

### .cameraCreateOrthographicViewCamera ()

Esta funcion crea una camara ortografica nueva

### .setCameraPosition ()

Esta funcion configura la posicion de camara basandose en la vista del momento

### .fireMarkerCountUpdatedEvent ()

Esta funcion ejecuta el cambio de camara en la vista corriente

### .adjustDimensions (data: Dictionary con medidas y orientaciones del objeto)

Esta funcion ejecuta cambia las dimensiones del objeto en el visor segun la orientacion de sus vectores de posicion

### .setTextureToGeometry (geometry: Objeto geometrico, dimensions: Dictionary con medidas de objeto,textureSize: Dictionary con medidas de textura)

Esta funcion ejecuta cambia las facetas de un objeto segun la relacion entre las dimensiones y medidas de textura dadas

### .scaleModelDAE (value: Float)

Esta funcion escala el grupo y la escena del visor segun el valor dado para exportar el visor a DAE

### .clearPreviousCanvas (container: Nodo del DOM)

Esta funcion elimina el canvas del contenedor dado

### .changeCameraType(typeCamera: String de tipo de camara)

Esta funcion cambia el tipo de camara al tipo que especifica el usuario

## Eventos

Estos eventos se encargan de completar los procesos que implican diferentes acciones en el editor.

.onchange: llama al compositor a renderizar

.onload: carga la imagen en el visor como promesa

.onerror: captura el error en la promesa

## Ruta

src/components/DesignDashboard/Tabs/3D/classes/viewer[.](https://github.com/mrdoob/three.js/blob/master/src/animation/AnimationAction.js)js

**Room**

Esta clase abstrae toda la estructura de una habitacion para ser incluida dentro de un visor

## Constructor

### Room(w,h,l,params)

options - este atributo toma el diccionario params para las opciones de configuracion

availableObjects - lista de objetos disponibles

scene - Group que debe estar en la escena del visor

scaleFactor - factor para escalar la escena de la habitacion

objects - lista de objetos

materials - Diccionario con objetos Mesh para los diferentes materiales de la habitacion (‘wall’ , ‘ceil’, ‘floor’, ‘editorBox’)

geometry - objeto geometrico para la habitacion

mesh - Mesh de la escena de la habitacion con sombras

## Propiedades

### .light: DirectionalLight

Abstraccion de una luz direccional con ubicacion y direccion

## Metodos

### .updateObject (data: Dictionary de objetos )

Esta funcion actualiza los objetos dados los datos en un diccionario

### .removeTag (tag: String)

Esta funcion remueve un objeto de acuerdo a su etiqueta

### .updateData (obj: Objeto, data: Diccionario con coordenadas)

Esta funcion rota y cambia las posiciones de un objeto segun las coordenadas de posicion y rotacion de un objeto dado

### .loadObject (data: Dictionary con datos)

Esta funcion crea carga un objeto en la escena segun los datos dados

### .loadWindow ()

Esta funcion carga una ventana en la escena

### .setSize (data: Dictionary con coordenadas)

Esta funcion cambia las medidas del Mesh de la sala

### .setColors (data: Dictionary con datos)

Esta funcion cambia los colores del material en la escena segun los datos dados

### .loadTextures (textureName)

Esta funcion cambia la textura de la pared del material

### .getMesh ()

Esta funcion devuelve el Mesh de la escena

### .updateMaterials ()

Esta funcion cambia los materiales en el Mesh de acuerdo a la configuracion de los materiales

### .setLight (data: Dictionary con datos de luces)

Esta funcion cambia la intensidad y el color de la luz en la instancia de sala

### .findObjectByTag ()

Esta funcion retorna un objeto dado segun su tag

## Ruta

src/components/DesignDashboard/Tabs/3D/classes/room[.](https://github.com/mrdoob/three.js/blob/master/src/animation/AnimationAction.js)js

**Contribuir a este documento**

No hay reglas sobre como contribuir a este documento. Lo unico que se pide es seguir las especificaciones de la documentacion para mantener la legibilidad. Las contribuciones se pueden hacer en base a nuevas funciones o funciones que ya estan en el repositorio, necesitan documentacion y aun no la tienen.