A 3D VIEWER FOR COMPLEX HIERARCHICAL PRODUCT MODELS

Documentación: hom3r API

(versión en español)

DOCUMENT IDENTIFIER:

Document	hom3r API
Status	Final
Editors	Daniel González-Toledo (UMA); María Cuevas-Rodríguez (UMA); Carlos Garre del Olmo (UMA); Luis Molina-Tanco (UMA); Arcadio Reyes-Lecuona (UMA)
Document description	**Documento donde se presenta el visualizador 3D hom3r. Se describe en detalle este visualizador, así como su integración y la API que ofrece.**

REVISION TABLE:

Version	Date	Modified pages	Modified Sections	Comments
1.0	18/11/2016	-	-	Primera versión del documento hom3r API

TABLE OF CONTENTS

1 Visualizador 3D (hom3r) 4

1.1 Como integrar hom3r en una aplicación web 4

1.2 hom3r API 6

1 Visualizador 3D (hom3r)

HOM3R (Hierarchical prOduct Model 3D vieweR) en un visualizador 3D diseñado para ser incorporado en una aplicación web.

El objetivo de este módulo es generar un render 3D interactivo de la totalidad del producto, el cual permite al usuario navegar a través de la geometría del mismo, mostrando información proveniente de la aplicación web. Además, dicho módulo es el encargado de cargar el modelo geométrico del producto a petición de la aplicación y administrar la jerarquía del producto. El sistema ofrece una interfaz de usuario, tal y como se ve en la Figura 1, que da acceso a todas las funcionalidades implementadas. El visualizador responde a los eventos de la misma y al igual que a los distintos eventos de ratón y teclado para controlar la navegación.

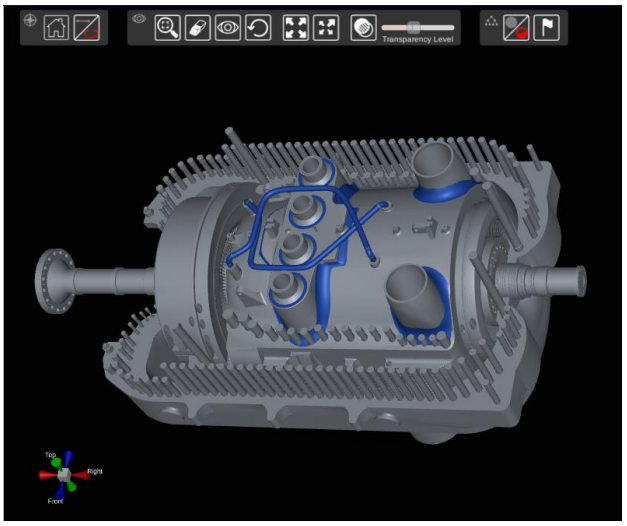

Figura 1. Visualizador 3D (hom3r)

1.1 Como integrar hom3r en una aplicación web

Hom3r está formado por un canvas 3D y una API JavaScript. La implementación del canvas 3D se ha realizado usando la plataforma de desarrollo de juegos Unity 3D en su versión 5 Pro, compilado en WebGL para su posterior integración en una aplicación web. hom3r ofrece un API JavaScript que encarga de: (1) la carga del canvas 3D y (2) de implementar la interfaz que permite el intercambio de información. Dicha API tiene la forma final de varios ficheros, con código JavaScript en su interior, donde se implementa todo el código necesario.

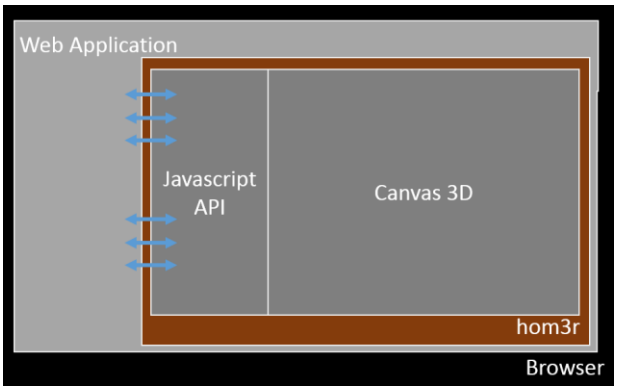

Figura 2. Diagrama de bloques del sistema desarrollado en su contexto

Las funciones encargadas de cargar el canvas 3D están basadas en las plantillas y ejemplos proporcionados por Unity3D. La API de hom3r, descrita en la siguiente sección, se ha implementado haciendo uso de un contenedor de JavaScript, que hace las veces de clase, donde se han definido los distintos métodos y atributos necesarios. Estos métodos se encargan de comunicarse con el módulo 3D, adaptando los tipos de datos cuando es necesario. Además, esta API permite que la aplicación web registre una función *callback* a través de la cual será informada de los distintos eventos que suceden en la interacción con el módulo 3D.

Para hacer uso de la API, el script *Hom3rAPI.js* debe ser incluido por la aplicación web mediante la correspondiente sentencia HTML. Una vez hecho esto, para acceder a los distintos métodos ofrecidos por la API, desde la aplicación, es necesario usar el prefijo "*Hom3rAPI*" de tal forma que, por ejemplo, registrar la función *MyCallbackFunction* desde la aplicación se hace de la siguiente forma: *Hom3rAPI.RegisterCallBack (MyCallBackFunction)*.

Además del script que contiene la API, la aplicación web debe integrar el resto de scripts que contienen el código y permiten la carga de hom3r. A continuación, se detallan todos estos archivos:

- **Hom3rAPI.js**: Contiene la API javascript implementada para comunicarse con el visualizador 3D.
- **UnityConfigModule.js**: Configura los parámetros necesarios para la carga de hom3r, entre ellos: (1) donde se encuentran los ficheros que contienen el código de hom3r y (2) el tamaño de memoria que se reserva para la ejecución, este parámetro hay que ajustar dependiendo del peso de los modelos 3D. Además, en este script es donde se deben hacer los chequeos de compatibilidad del navegador y el manejo de excepciones, si se desea.
- **UnityProgress.js**: Este script contiene el código que se encarga de mostrar la imagen y barra de progreso durante la carga del canvas WebGL generado por Unity, que contiene hom3r. Si se desea cambiar estas imágenes deben cambiarse en ese archivo.
- **UnityLoader.js**: Este script contiene el código del cargador de WebGLs de Unity.

Para añadir hom3r en una aplicación web:

1. Se deben incluir los ficheros JavaScript en el código HTML:

```
<script src="~/Hom3r/Hom3rAPI.js"></script>
<script src="~/Hom3r/UnityConfigModule.js"></script>
<script src="~/Hom3r/ProgressBar/UnityProgress.js"></script>
<script src="~/Hom3r/Release/UnityLoader.js"></script>
```

2. Se debe incluir el canvas, en el código html, allí donde se desee que aparezca:

```
<div id="hom3rwebgl">
    <canvas class="emscripten" id="canvas"
oncontextmenu="event.preventDefault()" ></canvas>
</div>
```

A continuación, se muestra un ejemplo de página web sencilla que incluye hom3r.

```
<!doctype html>

<html lang="en-us">
<head>
<meta charset="utf-8">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>hom3r example</title>
</head>
<body>
<h3> home3r example</h3>
<div id="hom3rwebgl">
<canvas class="emscripten" id="canvas" oncontextmenu="event.preventDefault()"
height="700px" width="900px"></canvas>
</div>
<script src="Hom3rAPI.js"></script>
<script src="UnityConfigModule.js"></script>
<script src="ProgressBar/UnityProgress.js"></script>
<script src="Release/UnityLoader.js"></script>
</body>
</html>
```

1.2 hom3r API

Esta sección define la interfaz (API) que hom3r ofrece a la aplicación web, con el objetivo de lograr que el canvas 3D se pueda integrar en la aplicación web y pueda intercambiar información con la misma. Dicha API podemos dividirla en: (1) Métodos utilizados en la aplicación web para enviar información a hom3r (interfaz de entrada) y (2) métodos que utiliza la aplicación web para recibir información de hom3r (callbaks), tal y como se muestra en la siguiente Figura 3.

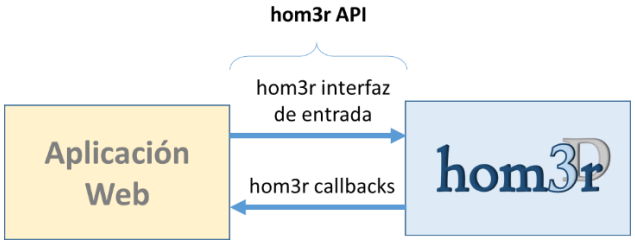

Figura 3. hom3r API

Métodos del interfaz de entrada

La interfaz está formada por los métodos que se muestran en el siguiente diagrama de la Figura 4.

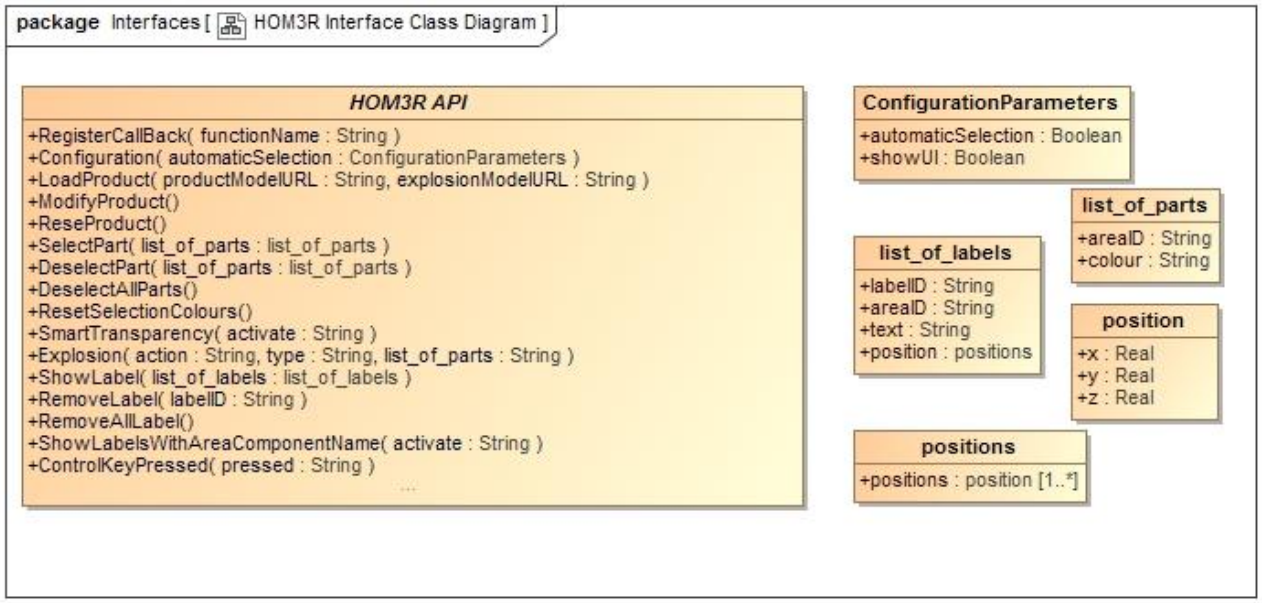

Figura 4. Métodos de la interfaz del sistema con la aplicación web

A continuación se van a explicar cada uno de los métodos y la función que realizan. Esta explicación consta cada una de dos apartados: (1) una descripción de la función que realiza el método y de su funcionamiento y (2) se expondrán los parámetros que usa el método.

RegisterCallBack (functionName)

Descripción: Registra una callback para que la aplicación web reciba información procedente del módulo 3D. La aplicación web debe definir una función y registrarla en la interfaz, a partir de ese momento, el módulo 3D enviará información a la aplicación web a través de la misma. Se puede registrar una callback en cualquier momento, aunque lo habitual será registrarla al principio de la ejecución del DOM de la web. En el apartado siguiente, *Callbacks*, se detalla en mayor medida como debe ser esa callback.

Parámetros: *functionName* es un parámetro de tipo string que indica el nombre de la función de callback que será llamada para pasar información del módulo 3D a la aplicación Web.

Configuration (config) * No implementado todavía

Descripción: Establece en hom3r los parámetros de configuración deseados. Se puede recibir una llamada a este método en cualquier momento. Si no se recibe esta llamada se usará la configuración por defecto.

Parámetros: *config* es un objeto de tipo la clase *ConfigurationParameters* que indica el valor de cada uno de los parámetros de configuración. Si alguno se deja en blanco el sistema usara su valor por defecto.

Los parámetros de configuración disponibles son los siguientes:

Parámetro	Valores	Descripción
automaticSelection	Booleano	Cuando el usuario hace un click de ratón sobre una parte del producto: • Si el valor de este parámetro es *verdadero* el módulo 3D marcara visualmente la pieza como seleccionada e informara posteriormente a la aplicación web de que se ha seleccionado esta pieza. • Si el valor de este parámetro es *falso* el módulo 3D solamente informara a la aplicación web de que el usuario quiere seleccionar esa pieza. Será la aplicación web la responsable de seleccionar la pieza mediante el comando correspondiente. Valor por defecto: verdadero
showUI	Booleano	Parámetro para mostrar o esconder la interfaz de usuario que ofrece hom3r. El parámetro permite mostrarla o esconderla por completo, no parcialmente.

LoadProduct (productModelURL, explosionModelURL)

Descripción: Indica a hom3r que empiece la carga del modelo de un producto, el modelo contendrá la información de la geometría, la explosión y otras. Se puede recibir una llamada a este método en cualquier momento. Si se recibe más de una llamada solo la primera llamada será atendida, el resto serán ignoradas. Es responsabilidad de la aplicación que los datos contenidos en el modelo sean correctos, si no lo son el modelo será descartado. Si las URLs contenidas en los parámetros no funcionan serán descartadas. Una vez completada la acción, el sistema responderá su resultado a través de la callback registrada por la aplicación.

```
productModelURL = "/Home/ProductModelHom3r";
explosionModelURL = "/Content/turbineexplosion.xml";
Hom3rAPI.LoadProduct(productModelURL, explosionModelURL);
```

Parámetros: *productModelURL* es un parámetro de tipo cadena de caracteres que indica la URL donde el sistema se podrá descargar el json que contiene toda la información del modelo del producto.

explosionModelURL es un parámetro de tipo cadena de caracteres que indica la URL donde el sistema se podrá descargar el XML que contiene toda la información del modelo de explosión del producto.

ModifyProduct (urlJson) * No implementado todavía

Descripción: Indica a hom3r que actualice el modelo del producto actualmente cargado. Para ello, hom3r modifica la información contenida en las partes del modelo de producto actualmente cargado con los valores contenidos en el nuevo modelo. Se puede recibir una llamada a este método en cualquier momento. Es responsabilidad de la aplicación que los datos contenidos en el modelo sean correctos, si no lo son el modelo será descartado. Este método no añade nuevas partes al modelo. Si la url contenida en el parámetro *jsonUrl* no funciona el parámetro será descartado. Una vez completada la acción el sistema responderá su resultado a través de la callback registrada por la aplicación.

Parámetros: urlJson es un parámetro de tipo cadena de caracteres que indica la url donde el sistema se podrá descargar el json que contiene toda la información del modelo.

ResetProduct () * No implementado todavía

Descripción: Indica a homer que borre el modelo actualmente almacenado y mostrado. Tras la ejecución de este método no se renderizará ninguna geometría 3D hasta que se cargue un nuevo modelo. Una vez completada la acción el sistema responderá su resultado a través de la callback registrada por la aplicación.

SelectPart (list_of_parts)

Descripción: Cambia al estado seleccionado (también llamado estado confirmado) las partes del producto indicadas mediante en el parámetro *list_of_parts*. Se puede recibir una llamada a este método en cualquier momento. Las partes que se indican, para ser confirmadas, pueden estar a cualquier nivel del árbol del producto. Es decir, podrán ser nodos, hojas o áreas. Si una parte indicada tiene hijos, es decir, es nodo u hoja, todas sus partes hijas serán también confirmadas.

Parámetros: *list_of_parts* es un parámetro de tipo lista o array que contiene en cada posición: (1) *areaID* parámetro de tipo cadena que representa el identificador de la parte a seleccionar y (2) *colour* parámetro de tipo cadena que indica el color*, con que se representará la selección, en formato hexadecimal (por ejemplo, #ff0000).

***** Si no se indica ningún color el sistema usará el color por defecto. Una vez indicado un color de selección para una parte, ese pasa a ser el color por defecto para seleccionar esa parte. Se pueden borrar todos los colores por defecto de todas las partes mediante el comando *ResetSelectionColours*.

```
var newSelectedNodes = [];
newSelectedNodes.push({"areaID": partID, "colour": "#ff0000"});
…
Hom3rAPI.SelectPart(newSelectedNodes);
```

DeselectPart (list_of_parts)

Descripción: Cambia al estado no seleccionado (también llamado no confirmado) las partes del producto indicadas mediante en el parámetro *list_of_parts*. Se puede recibir una llamada a este método en cualquier momento. Las partes que se indican, para ser deseleccionadas, pueden estar a cualquier nivel del árbol del producto. Es decir, podrán ser nodos, hojas o áreas. Si una parte indicada tiene hijos, es decir es nodo u hoja, todas sus partes hijas serán también deseleccionadas.

Parámetros: *list_of_parts* es un parámetro de tipo lista o array que contiene en cada posición el identificador de la parte deseleccionar.

```
var DeSelectNodes = [];
DeSelectNodes.push({"areaID": partID, "colour": ""});
…
Hom3rAPI.SelectPart(DeSelectNodes);
```

DeselectAllParts ()

Descripción: Cambia al estado no seleccionado / no confirmado todas las partes del producto confirmadas.

Parámetros: *sin parámetros de entrada.*

ResetSelectionColours()

Descripción: Se reinicia, al color por defecto, el color de selección que se encuentra almacenado en algunas partes del producto que han sido previamente seleccionadas utilizando dicho color.

Parámetros: *sin parámetros*

SmartTransparency (activate)

Descripción: Enciende o apaga la transparencia inteligente en función del parámetro recibido. Nada más recibir este parámetro el sistema modificará el estado de la transparencia inteligente. Se puede recibir una llamada a este método en cualquier momento. Si el parámetro *activate* no contiene alguno de los valores validos esta llamada será descartada.

Parámetros: *activate* es un parámetro de tipo cadena que indica el nuevo estado de la transparencia inteligente. Sus valores validos "true" (para activar la transparencia inteligente) o "false" (para desactivar la transparencia inteligente).

Explosion (action, type, partID) * No implementado todavía

Descripción: Activa la animación que la explosión o implosión de las diferentes piezas del modelo con el objetivo de evitar oclusiones entre piezas. Esta animación dependerá de la acción que se envíe en el parámetro *action*. Nada más recibir este parámetro el sistema modificará el estado de la explosión. Se puede recibir una llamada a este método en cualquier momento. Si los parámetros no contienen alguno de los valores validos esta llamada será descartada. Existen dos tipos de explosión: global, donde se aplicará la animación de explosión a todas las piezas, y local, donde solo las piezas seleccionadas sufrirán la animación. La explosión local implica el desensamblado de partes no seleccionadas en el caso de que estas ocluyan a la seleccionada.

Parámetros: *action*: es un parámetro de tipo cadena que indica, que puede tener los valores: (1) *explode*, el cual provocará que la animación realice un desensamblado de las piezas, y (2) *implode*, el cual provocará que la animación realice un desensamblado de las piezas.

type es una cadena de caracteres que indica el tipo de animación en la explosión. Los valores válidos son: "global", desensamblaje o emsamblaje (según la *action* elegida) de todas las partes y "local", solo las partes indicadas en el parámetro *list_of_parts*) son desensambaladas o ensambladas (según la *action* elegida).

list_of_parts array de las partes de modelos elegidas para aplicar la explión o implosion. Este parámetro es ignorado si el tipo de explosión es global.

ShowLabel(list_of_labels)

Descripción: Crea una etiqueta 3D asociada a cada una de las partes del producto indicada en el parámetro de entrada *list_of_labels*, el texto mostrado en la etiqueta se envía también a través del parámetro de entrada. Este método puede ser llamado en cualquier momento.

Parámetros: *list_of_labels* es un parámetro de tipo lista o array que contiene en cada posición: (1) *labelID* el identificador de la etiqueta, (2) *areaID* el identificador del área del producto al que se asocia la etiqueta, (3) *text* el texto que debe mostrar la etiqueta y (4) *position* la posición (x, y, z)

donde se conecta la etiqueta con el área de la parte del producto. Si no se conoce dicha posición se debe enviar una posición por defecto (0, 0, 0), la cual hará que el sistema conecte la etiqueta con el centro geométrico del área.

```
var position = [];
position.push({"x": 0, "y": 0, "z": 0});
var newlabels = [];
newlabels.push({ "labelID": labelID, "areaID": areaID, "text": text, "position":
position });

Hom3rAPI.ShowLabel(newlabels);
```

RemoveLabel(labelID)

Descripción: Elimina una etiqueta 3D determinada, la cual es indicada por el parámetro de entrada del método. Para volver a visualizarla es necesario volver a mandar todos los datos necesarios a hom3r mediante el comando *ShowLabel*. Este método puede ser llamado en cualquier momento.

Parámetros: labelID es una cadena de caracteres que indica el identificador de la etiqueta que se va a dejar de visualizar.

RemoveAllLabel()

Descripción: Elimina todas las etiquetas 3D que se estén mostrando en ese momento. Para volver a visualizarlas es necesario volver a mandarlas mediante el comando ShowLabel.

Parámetros: *sin parámetros de entrada.*

ShowLabelsWithAreaComponentName(activate)

Descripción: Enciende o apaga el modo “mostrar etiqueta con nombre del componente o área” de hom3r. El cual crea una etiqueta 3D asociado a cada parte del producto seleccionado mostrando el nombre de este. Nada más recibir este parámetro el sistema modificará su estado mostrando las etiquetas que correspondan. Si el parámetro *activate* no contiene alguno de los valores validos esta llamada será descartada.

Parámetros: *activate* es un parámetro de tipo cadena que indica el nuevo estado de este modo. Sus valores validos “true” (para activar el modo) o “false” (para desactivar el modo).

ControlKeyPressed(pressed)

Descripción: Para no interferir con el funcionamiento normal de la aplicación web y debido a una limitación de Unity WebGL, hom3r no es capaz de leer las pulsaciones de teclado. De todas formas, la única tecla que hom3r necesita para su funcionamiento es la pulsación de la tecla control para la selección múltiple. Es por ello que se ha decidido crear una interfaz para informar a hom3r del uso de esta tecla. Es la aplicación web la que debe informar a hom3r del cada vez que se pulse esta tecla. No es necesario él envío continuo de este comando, solo es necesario enviarlo cuando hay un cambio de estado.

Parámetros: *pressed* es un parámetro de tipo cadena que indica el estado pulsado o no de la tecla control. Sus valores validos son: “true” indica que la tecla ha sido pulsada y “false ” indica que la tecla control no está siendo pulsada.

```
document.onkeydown = function (data) {
    if (data.keyCode === 17 && lastKeyDown !== data.keyCode) {
        Hom3rAPI.ControlKeyPressed("true");
        lastKeyDown = data.keyCode;
    }
};
document.onkeyup = function (_data) {
    Hom3rAPI.ControlKeyPressed("false");
    lastKeyDown = "";
}
```

Callbacks

La comunicación entre el sistema y la aplicación web se lleva a cabo a través del uso de callbacks. Las callbacks permiten que la aplicación pueda definir una función en su sistema con el nombre que se desee y que implemente las respuestas a los mensajes como más le convenga.

La aplicación debe, por tanto, definir una función para posteriormente registrar en la API del sistema como *callback*. Será decisión de la aplicación web el nombre que le da a esta función y como está implementada. Sin embargo, su definición debe ser tal que esté formada por dos parámetros de entrada de tipo cadena, tal que así: *nombre_de_la_function(message, value)*. A través del parámetro *message* el sistema indica a la aplicación web cuál es el mensaje que le quiere transmitir y a través del parámetro *value* indica cuál es el valor que toma ese mensaje.

Ejemplo de uso de una callback:

```
//Register Callback at the beginning.
Hom3rAPI.RegisterCallBack(MyCallBackFunction);

//Callback.
function MyCallBackFunction(message, value) {
    if (message === "hom3r") {
        if (value === "ok") {
            LoadProduct();                      //User function
        } else if (value === "error")
        {
            console.log(vale);
        }
    }
    if (message === "product") {
        if (value === "ok") {
            console.log("Product model loaded correctly!!!");
        } else if (value === "error") {
            console.log("Error: product model cannot be loaded.");
        }
    }
    if (message === "selectPart") {
        FromHom3r_SelectPartWithID(value);      //User function.
    }
    if (message === "deselectPart") {
        FromHom3r_DeselectPartWithID(value);    //User function
    }
    if (message === "deselectAllParts") {
        FromHom3r_DeselectAllParts();           //User function
    }
    if (message === "SaveSinglePoint"){
        FromHom3r_SaveSinglePoint(value);       //User function
    }
    if (message === "RemoveSinglePoint") {
        FromHom3r_RemoveSinglePoint(value);     //User function
    }
    if (message === "EndSinglePoint") {
        FromHom3r_EndSinglePoint(value);        //User function
    }
    if (message === "RemoveLabel") {
        FromHom3r_RemoveLabel(value);           //User function
    }
}
```

En la Tabla 1 se pueden ver los distintos tipos de mensajes y sus valores posibles.

Tabla 1: Función CallBack. *Mensajes y valores posibles en cada caso.*

Message	Descripción	Valor

hom3r	Se recibe este mensaje cuando el sistema se ha cargado correctamente y está a la espera de recibir el modelo del producto. Si no se registra la callback a tiempo puede que no se reciba este mensaje.	Valores: *ok/error*
		Ejemplo de método al que podemos llamar cuando recibamos este mensaje, este método se encarga de cargar el json que contiene el modelo del producto, además del xml que contiene el modelo de la explosión. Ejemplo:
		`function LoadProduct() {`
		`  productModelURL = "/Home/ProductModelHom3r";`
		`  explosionModelURL = "/Content/turbineexplosion.xml";`
		`  Hom3rAPI.LoadProduct(productModelURL, explosionModelURL);`
		`}`

product	Se recibe este mensaje cuando el sistema ha cargado correctamente el modelo del producto o se ha producido un error al cargarlo.	Valores: *ok/error*
		Ejemplo de una posible callback:
		`if (message === "product") {`
		`  if (value === "ok") {`
		`  console.log("Product model loaded correctly!!!");`
		`  } else if (value === "error") {`
		`  console.log("Error: product model cannot be loaded.");;`
		`  }`
		`}`

selectPart	Se recibe este mensaje cuando el usuario ha solicitado la selección de una parte o varias del producto durante la interacción 3D, indicando que parte ha sido seleccionada.	Valores: un objeto lista o Array que contiene el conjunto de partes del producto que han sido seleccionadas. Cada posición de este Array contiene dos campos: (1) *areaID* parámetro de tipo cadena que representa el identificador de la parte a seleccionar y (2) *colour* parámetro de tipo cadena que no se usa y por tanto vendrá en blanco.
		Ejemplo de cómo manejar el valor recibido:
		`function FromHom3r_SelectPartWithID(list_of_parts) {`
		`  for (var i = 0; i < list_of_parts.data.length; i++) {`
		`  console.log("This part has been selected: " + list_of_parts.data[i].areaID);`
		`  }`
		`}`

deselectPart	Se recibe este mensaje cuando el usuario ha solicitado la de-selección	Valores: un objeto lista o Array que contiene el conjunto de partes del producto que han sido deseleccionadas. Cada posición de este Array contiene dos campos: (1) *areaID*

	de una parte del producto durante la interacción 3D, indicando que parte ha sido seleccionada.	parámetro de tipo cadena que representa el identificador de la parte a seleccionar y (2) *colour* parámetro de tipo cadena que no se usa y por tanto vendrá en blanco.
		Ejemplo de cómo manejar el valor recibido.
		`function FromHom3r_DeselectPartWithID(list_of_parts) {`
		`  for (var i = 0; i < list_of_parts.data.length; i++) {`
		`  console.log("This part has been de-selected: " + list_of_parts.data[i].areaID);`
		`  }`
		`}`

`deselectAllPart`	Se recibe este mensaje cuando el usuario ha solicitado la de-selección de todas las partes del producto.	Values: *Sin parámetro*
RemoveLabel	Se recibe este mensaje cuando el usuario ha eliminado una etiqueta 3D, mediante la interfaz de usuario, que había sido previamente introducida en hom3r por la aplicación mediante el comando *ShowLabel*.	Valores: cadena de texto que indica el identificador de la etiqueta que se desea eliminar. Ese identificador le fue proporcionado a hom3r por la aplicación web mediante el comando *ShowLabel*.

Comportamiento del interfaz

En este apartado se va a mostrar el comportamiento del sistema a través del intercambio de mensajes en los escenarios principales a los que se enfrenta.

Para lograr este objetivo, se van a presentar distintos diagramas de secuencia, en los cuales se observará el flujo de datos para dar respuesta a los escenarios básicos a los que el sistema debe ser capaz de hacer frente. Por tanto, a continuación, se va a ordena el apartado según los distintos escenarios principales.

Escenario principal de inicialización y carga de modelos

Este escenario involucra a la interfaz cuando se carga el sistema dentro de la aplicación web. El sistema será cargado haciendo uso de un codigo específico que sera entregado dentro del conjunto de ficheros que forman la API. De forma paralera a la carga del canvas 3D se debe registrar la callback en la API.

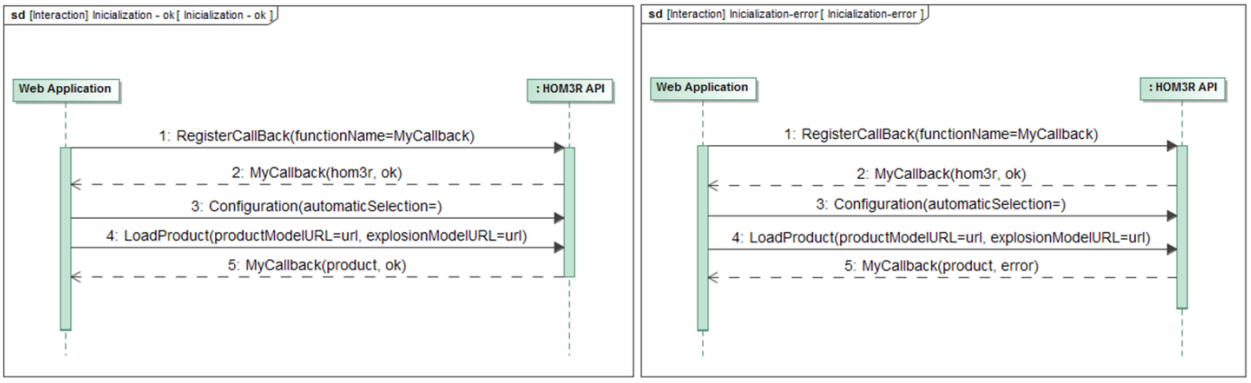

Figura 5. Diagrama de secuencia de inicialización. (a) Se muestra una inicialización sin contratiempos. (b) Se muestra una inicialización en la que se ha producido un error en la cargar del modelo del producto.

Supongamos que la carga del canvas 3D se ha realizado sin ningún contratiempo, tal y como se ve en la Figura 5, el sistema hom3r enviará a la aplicación web mediante la callback (registrada como *MyCallback*) un mensaje (mensaje 2 en la figura) indicando que está listo y esperando recibir comandos. En caso de que la carga no se realizara correctamente este mensaje o bien no se recibiría o bien se recibirá un *error* en su lugar.

A partir de ese momento y cuando la aplicación lo desee puede enviar los parámetros de configuración. Si no lo hace, el sistema funcionara con los parámetros de configuración por defecto. Tras esto, la aplicación debe indicar al sistema que cargue el modelo del producto, que incluye la geometría y los datos adjuntos, mediante el comando *LoadProduct*. Este comando indica al sistema las *URLs* donde puede acceder a los ficheros *json* y xml para descargar los modelos. Si todo ocurre sin ningún contratiempo, se recibirá un mensaje de confirmación, en caso contrario se recibirá un mensaje de error.

Escenario principal de interacción

Suponiendo que la carga del canvas 3D y del producto se ha realizado sin ningún contratiempo, nos encontraríamos en la situación mostrada en la Figura 5 (a). A partir de ese momento, el sistema está visualizando el modelo geométrico del producto y el usuario se encontrará interaccionando tanto con la aplicación web como con el sistema.

El intercambio de mensajes es bidireccional, por un lado la aplicación envía mensajes a traves de los metodos definidos en la interfaz y por el otro el sistema envía mensajes a traves de la callback a la aplicación web. En el apartado anterior se definieron los distintos mensajes y las condiciones determinadas en que se envía cada

uno de ellos, al igual que los parámetros de cada método. En el siguiente diagrama se muestra un ejemplo de interacción entre hom3r y la aplicación web.

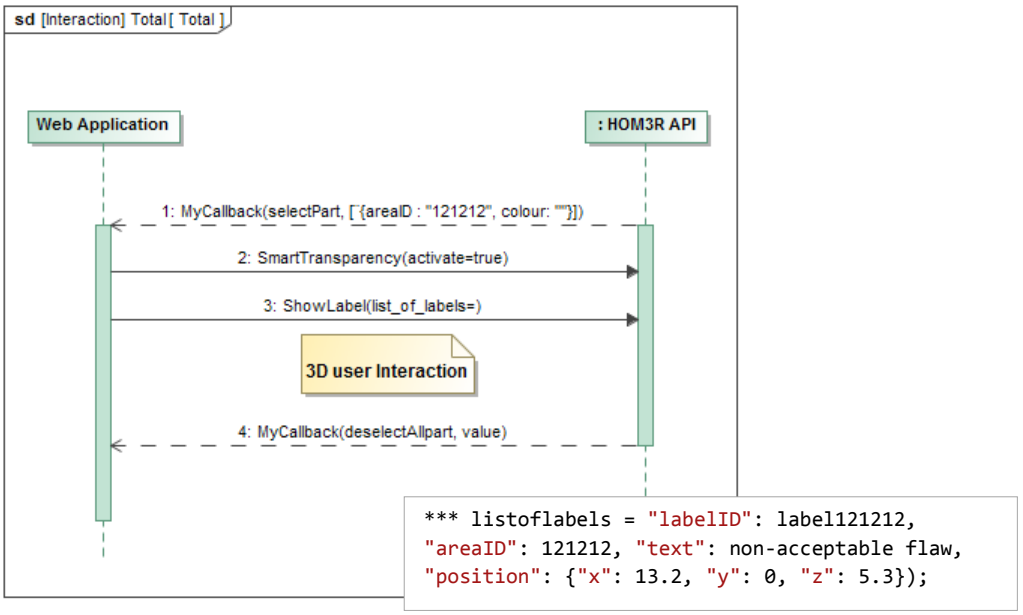

Figura 6. *Diagrama de secuencia de interacción en la interfaz entre la aplicación web y el sistema. Se muestra un ejemplo del posible intercambio de mensajes*

En la Figura 6 se muestra un ejemplo de interacción, solo a título ilustrativo, el orden de los mensajes podría ser otro totalmente diferente y seguirían siendo válidos, además, no se muestran todos los mensajes disponibles, solo un pequeño grupo de ellos.

Una vez cargado el modelo del producto (ilustrado en la Figura 5), comienza la parte de interacción del usuario, tanto con la aplicación web como con el modelo 3D. El usuario, mediante la interacción 3D (haciendo uso del ratón), ha seleccionado la pieza con ID igual a 121212, el sistema envía esta información a la aplicación web. La aplicación web recibe este mensaje mediante la callback registrada.

El siguiente ejemplo de interacción que se muestra es la activación de la transparencia inteligente, lo cual permitirá la visualización completa de la pieza seleccionada. Para ello la aplicación web envía un comando (*2. SmartTransparency*) a hom3r, con valor true.

En el siguiente paso, la aplicación web indica a hom3r que muestre una etiqueta concreta (*3. ShowLabel*), indicándole el identificador de la etiqueta, el texto que debe aparecer en ella y la posición donde debe conectarse con la representación geométrica del modelo. Para finalizar, hom3r indica a la aplicación web que se han deseleccionado todas las partes del producto.