



Universidad de Valladolid



ESCUELA DE INGENIERÍAS
INDUSTRIALES

GRADO DE INGENIERIA DE DISEÑO INDUSTRIAL Y
DESARROLLO DE PRODUCTO

TRABAJO FIN DE GRADO

**PRESENTACION DE PRODUCTO EN
HTML5**

Autor:

Alfonso Sánchez Sanjuán

Tutor:

David Escudero Mancebo

JULIO de 2013

Me gustaría dedicar este proyecto a mi tutor, D. David Escudero Mancebo, por la orientación y guía, por el apoyo y la paciencia que ha tenido conmigo, y por creer en mí.

A los demás profesores de la facultad, especialmente a D. Jesús Magdaleno Martín, tutor de mi proyecto final en la Ingeniería Técnica, el cual ha sido la base para el presente trabajo.

A mi familia, que ha estado apoyándome en todo momento y que no ha desesperado a pesar del tiempo que me ha llevado realizarlo. De forma especial a mi hermana por la ayuda que me ha prestado.

A mis amigos y compañeros por los buenos ratos, y el apoyo en los no tan buenos. Especialmente en esta ocasión, a mis compañeros del último curso; en esta última etapa en la Universidad.

A todos ellos, muchas gracias

Alfonso

PRESENTACION DE PRODUCTO EN HTML5

Índice

1-Introducción.....	7
1.1-Diagrama resumen del proceso.....	8
1.2-Descripción del proceso.....	10
2-Antecedentes del Patinete COMPACT.....	13
2.1-Reseña del proyecto anterior.....	14
2.2-Detalles del Proyecto.....	15
2.2.1 Versatilidad.....	15
2.2.1-Plegado del Manillar.....	16
2.2.3-Alargamiento/Acortamiento de la Columna de dirección.....	17
2.2.4-Abatimiento de la Columna de dirección.....	17
2.2.5-Giro de las Ruedas.....	18
2.2.6-Estructura de la Base.....	18
2.3-Ingeniería.....	19
2.4-Planos.....	21
2.5-Conclusiones.....	25
3-Render Fotorrealista.....	27
3.1-Importar Archivos a Autodesk 3D Studio Max.....	28
3.2-Jerarquías.....	30
3.3-Entorno.....	32
3.3.1-Mesa.....	32

3.3.2-Sofá	33
3.3.3-Planta	33
3.3.4-Habitáculo	34
3.4-Materiales Max	34
3.4.1-Materiales propios del Patinete	36
3.4.2-Materiales del Habitáculo	39
3.4.3-Materiales de la Mesa	40
3.4.3-Materiales del Sofá	41
3.4.4-Materiales de la Planta	42
3.5-Iluminación	44
3.6-Cámaras	45
3.6.1-Cámara a partir de vista	47
3.7-Casos a destacar	48
3.7.1-Puertas y Ventanas	48
3.7.2-Mapeado de texturas	54
3.8- Configuración de Render	56
3.9-Exportación de Imágenes Renderizadas	57
3.9.1-Imágenes Generales	58
3.9.2-Imágenes de Detalle	63
3.9.3-Imágenes de Funcionamiento	70
4-Creación de Vídeo	73
4.1-Configuración	75

4.2-Entorno de la animación	75
4.2.1-Creación del degradado	76
4.2.2-Degradado como entorno.....	77
4.3-Definición de movimientos de articulaciones.....	79
4.4-Uso de Dope Sheet.....	81
4.5-Casos a destacar	83
4.5.1-Plegado del manillar: Wire Parameters.....	83
4.5.2-Compresión de muelles: Select and Non-Uniform Scale.....	87
4.6-Materiales en las animaciones (Materiales animados).....	89
4.6.1-Transparencia de objetos.....	89
4.6.2-Rreflexión del entorno	92
4.7-Movimientos de Cámara.....	93
4.7.1-Asignar una Trayectoria a la Cámara.....	97
4.8-Exportación de Vídeos.....	99
4.8.1-Vídeos Generales	99
4.8.2-Vídeos de Montaje	102
4.8.3-Vídeos de Funcionamiento	111
5-Web 3D	119
5.1-Antecedentes.....	120
5.1.1-HTML5	120
5.1.2-X3DOM	126
5.2-Proceso de transformación de Modelo 3D (Fichero 3ds) a Modelo 3D Interactivo (X3D)..	129

5.2.1-Diagrama.....	130
5.2.2-Descripción del proceso.....	131
5.3-Proceso detallado: Problemas y particularidades.....	132
5.3.1-Diagrama.....	132
5.3.2-Descripción del proceso.....	134
5.3.3-Transformación de archivos en la práctica	136
5.4-VRML.....	139
5.4.1-Algunas características de VRML	140
5.4.2-Aspecto de los objetos en VRML: Materiales	145
5.4.3-Transformación de archivos 3ds Max a VRML.....	147
5.5-X3D.....	150
5.5.1-Características del archivo X3D	150
5.5.2-Particularidades de la conversión a X3D.....	155
5.5.3-Montaje de conjuntos en X3D	159
5.5.4-Visualizador 360°.....	160
5.5.5-Vistas	168
5.5.6-Materiales del patinete	179
5.5.7-Configuración de producto	183
6-Composición web HTML5	191
6.1-Recursos.....	192
6.1.1-HTML5, CSS3, etc	192
6.1.2-jQuery	196

6.1.3- Responsive Web Design	199
6.2-Montaje final.....	200
7-Conclusiones.....	207
7.1-Imagen	209
7.2-Video.....	209
7.3-Interacción	210
7.3.1-Incompatibilidad / Consenso en la WEB	211
7.3.2-3D a partir de un objeto real	211
7.4-Estrategia	212
7.5-Web.....	213
8-Bibliografía	215
8.1-Antecedentes.....	216
8.2-Render.....	216
8.3-Video.....	218
8.4-HTML / HTML5.....	219
8.4.1-HTML5	220
8.5-Web 3D / 3D Interactivo.....	221
8.5.1- X3DOM	222
8.5.2- VRML.....	223
8.5.3- X3D.....	224
8.6-Consultas generales.....	225



1-Introducción

El objetivo del presente proyecto, se trata de actualizar y perfeccionar un proyecto anterior, completando las secciones que pudieron quedar más inconclusas en materia de representación y presentación de producto. Para ello, se realizan diversas representaciones multimedia, además de una presentación de producto a través de lenguaje de programación compatible con HTML5.

El producto utilizado para dicha presentación, fue diseñado previamente, y los propósitos de este proyecto son:

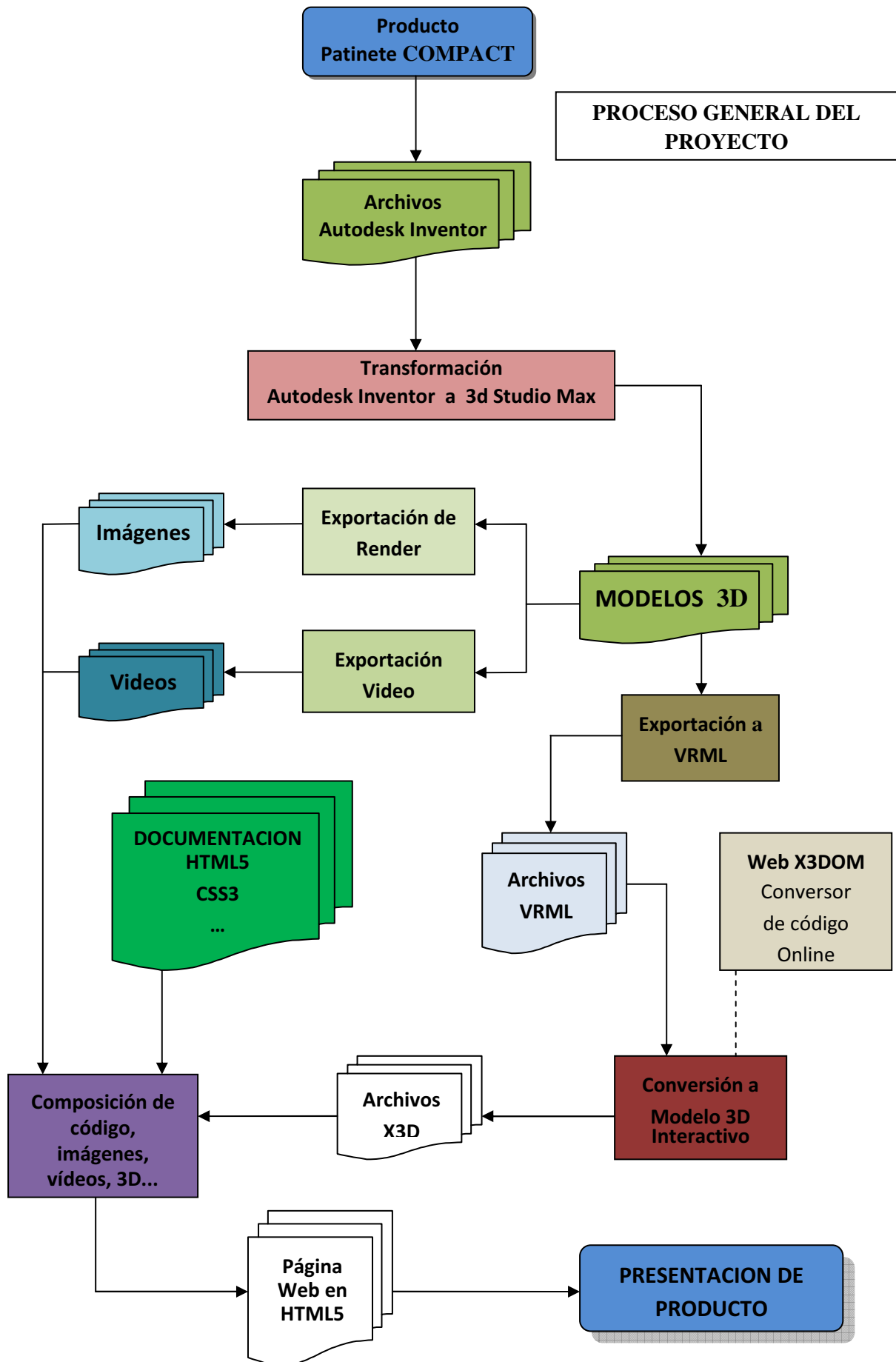
Dar una imagen del producto atractiva y realista, dar detalles con el fin de potenciar las características del producto, ofrecer calidad y cantidad de contenido, permitir la visualización y manipulación del producto de forma interactiva, y permitir la personalización del producto a gusto del cliente.

Para tales fines se utilizan los siguientes recursos:

Se utilizan imágenes generadas por ordenador (render) de alto realismo; tanto generales, como de detalle y primeros planos. Además de vídeos en los que se puede ver el funcionamiento de los distintos componentes y mecanismos del producto, así como el montaje de diversos conjuntos. Se ofrece una presentación X3D, que permite la visualización global del producto en 360 grados, además de cambiar algunas de sus características. Por último, una Web en HTML5, que englobe todas y cada una de las secciones; pudiendo navegar por ellas de forma sencilla y dinámica.

1.1-Diagrama resumen del proceso

En el siguiente diagrama de flujo, queda resumido el proceso llevado a cabo en este Proyecto. De esta forma, se da una visión global de todas y cada una de sus fases, y la obtención de los recursos anteriormente citados para los fines que en él se persiguen.



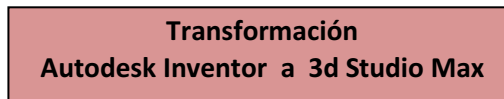
1.2-Descripción del proceso



El elemento de entrada del diagrama, tenemos el Patinete COMPACT. Producto diseñado en un proyecto anterior. Dicho producto se describe en el **capítulo 2**.



Nos serviremos de los archivos correspondientes al modelo 3d, creado en su momento con *Autodesk Inventor Professional 9*. En el **capítulo 2**, se describen las características de estos archivos; así como las limitaciones que se encontraron.



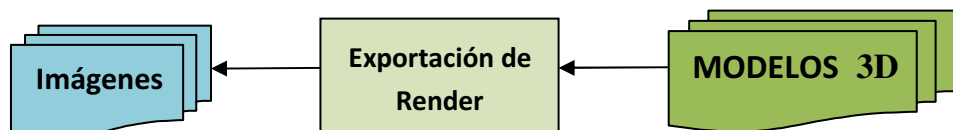
Por motivos de representación, se escoge el programa *Autodesk 3ds Max Design 2013* para este nuevo proyecto. Para ello debemos transformar los archivos procedentes de Autodesk Inventor Professional 9. Este sencillo proceso se describe en el **capítulo 3**.



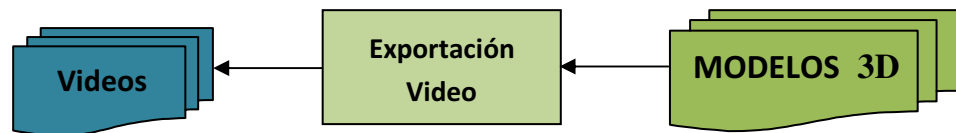
Una vez tenemos los archivos convertidos en Autodesk 3ds Max Design 2013; trabajamos con ellos, para obtener las distintas representaciones que queremos.

Dichas representaciones son las siguientes:

- Configuraremos una escena con otros elementos modelados, materiales e iluminación; en la que integraremos el producto, para poder exportar las imágenes más oportunas y representativas. De este proceso, se habla ampliamente en el **capítulo 3**.



- Animaremos el producto de manera que se aprecie su funcionamiento y montaje, así como los puntos fuertes del mismo. Configuraremos las animaciones de forma que podamos exportar los vídeos necesarios para representar las cualidades anteriormente citadas. Proceso descrito en el **capítulo 4**.



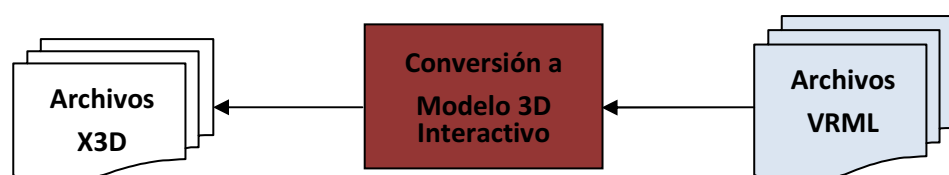
- Montamos archivos en los que aparezcan únicamente el producto o alguna parte de éste; tales como despieces, para exportar archivos **VRML**, que utilizaremos posteriormente. Esta parte del proceso se documentará en el **capítulo 5**.



Web X3DOM
 Conversor
 de código
 Online

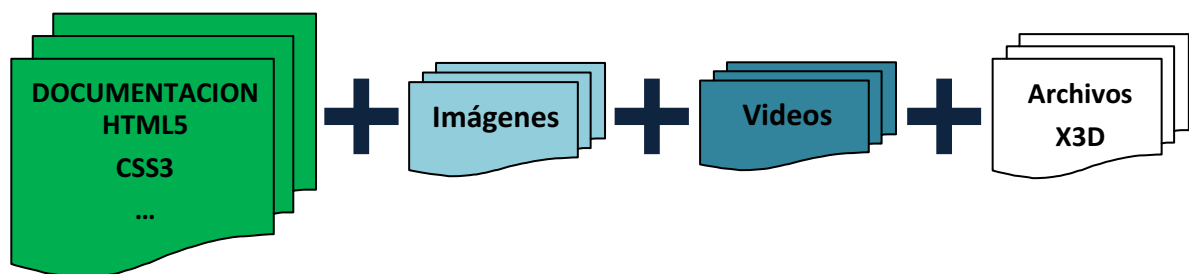
En la búsqueda de información y recursos para la representación 3D en la Web, encontramos la página www.x3dom.org. Dicha página, cuenta con un conversor online de código para tal fin.

Para obtener un modelo 3D interactivo, nos serviremos de los archivos VRML, exportados anteriormente. Tal proceso, nos devolverá como resultado, los archivos **X3D** necesarios. Este proceso, se describe en el **capítulo 5**; así como sus particularidades y dificultades encontradas.

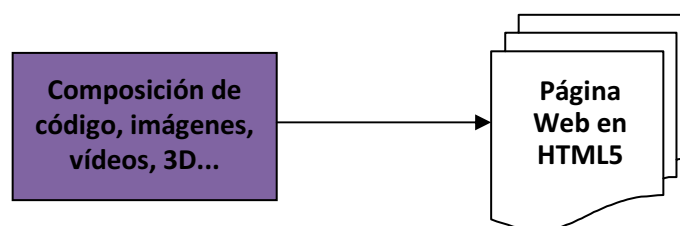


Tendremos en este momento, que recopilar tanta información y archivos de que dispongamos; procedentes de los distintos procesos que hemos llevado a cabo con los modelos 3D. Contenido que se muestra en el **capítulo 6**.

Además necesitaremos datos respecto al producto diseñado; qué encontramos en el anterior proyecto, tales como textos descriptivos, imágenes, datos técnicos...Junto a todo ello, la información necesaria en cuanto a **HTML5** y estilos **CSS3**.

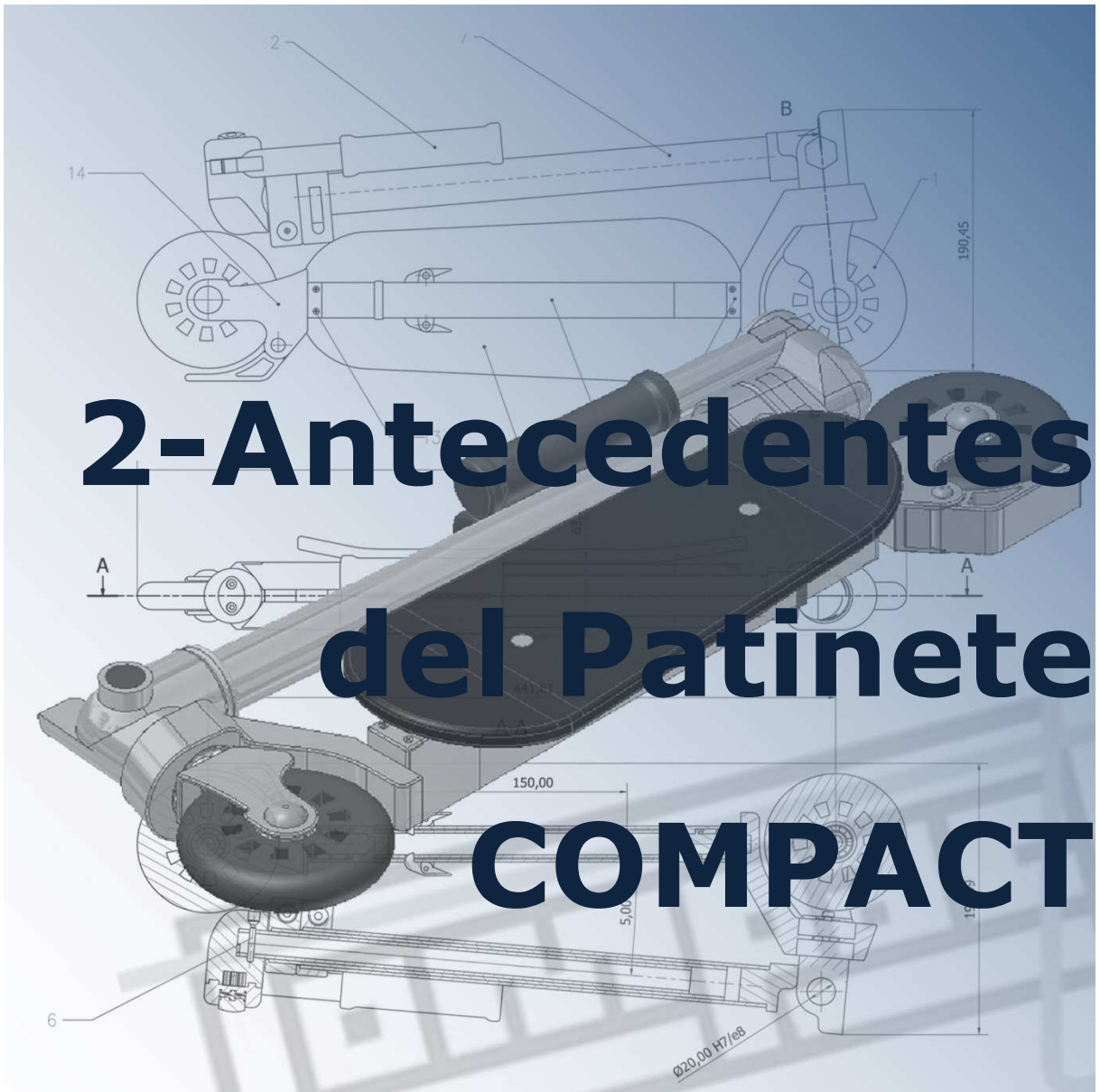


Con la ayuda de algún editor de código, elaboraremos los archivos necesarios en HTML5, integrando todo lo anterior.



PRESENTACION DE PRODUCTO

Como elemento de salida de este diagrama de flujo; y a su vez resultado de todo el proceso que en él se representa, encontramos la presentación de producto. En este caso, una **presentación en HTML5**, visual e interactiva, propósito del presente proyecto.



En este capítulo, se hace un recorrido por el proceso de diseño del Patinete COMPACT. Este es un proyecto anterior y el producto en el que se basa la presentación web que se quiere realizar en este nuevo trabajo.

2.1-Reseña del proyecto anterior

El producto que se presenta se trata de un patinete, el cual fue diseñado por mí, con la colaboración de Héctor Núñez Gómez, alumno también de esta escuela. El diseño de dicho patinete fue presentado como Proyecto Fin de Carrera conjunto, en Octubre de 2007 con el nombre de “PATINETE COMPACT”, y realizado bajo la tutoría de D. Jesús Magdaleno Martín.

El objetivo principal del proyecto era diseñar un producto versátil e innovador. Con el diseño del patinete “COMPACT”, se quiso incorporar algunas innovaciones en cuanto al plegado del patinete, que hagan que nuestro producto sea competitivo en el mercado y atractivo para el consumidor desde el punto de vista de la calidad y de la estética.

El objetivo principal es conseguir que ocupe el mínimo espacio una vez se haya dejado de utilizar. Tratando además de darle una visión y un uso mucho más global; no limitando nuestro mercado a un público infantil, sino abriéndonos a todo tipo de consumidor, sobre todo el joven. No sólo como un producto de entretenimiento, sino como una opción alternativa de transporte; ecológica y saludable.

Como ya hemos dicho, el objetivo principal es que ocupe el menor espacio cuando no se esté utilizando, frente a una amplia variedad de productos ya existentes que pudieran resultar aparatosos o no cumplir con las expectativas del usuario en este aspecto. Para ello incorporamos alguna diferencia en los sistemas de plegado que ofrecen la posibilidad de reducir considerablemente sus dimensiones de forma sencilla y segura.

El principal reclamo hacia el público joven es su diseño. Un diseño con unas formas muy atractivas a la vista, muy dinámico y que da la sensación de solidez.

En principio se ha pensado en ofertar un producto, que por su calidad y prestaciones no tenga un precio muy elevado en comparación con otros productos similares, ampliando así el número de potenciales consumidores.

En el diseño de nuestro patinete se tienen en cuenta materiales y métodos de fabricación para conseguir que sea lo suficientemente resistente y seguro para el usuario, cumpliendo para ello con la norma establecida sobre los esfuerzos que debe soportar este tipo de productos (**norma UNE-EN 14619: Equipo de deporte sobre ruedas. Patinetes. Requisitos de seguridad y métodos de ensayo**).

Haciendo hincapié en el tema de la seguridad también se incluye un manual detallado de utilización y se indicará el equipo de protección recomendado para su uso.

2.2-Detalles del Proyecto

Después de la realización de un exhaustivo estudio de campo y teniendo en cuenta los objetivos antes planteados, procedimos al diseño de nuestro patinete.

2.2.1 Versatilidad

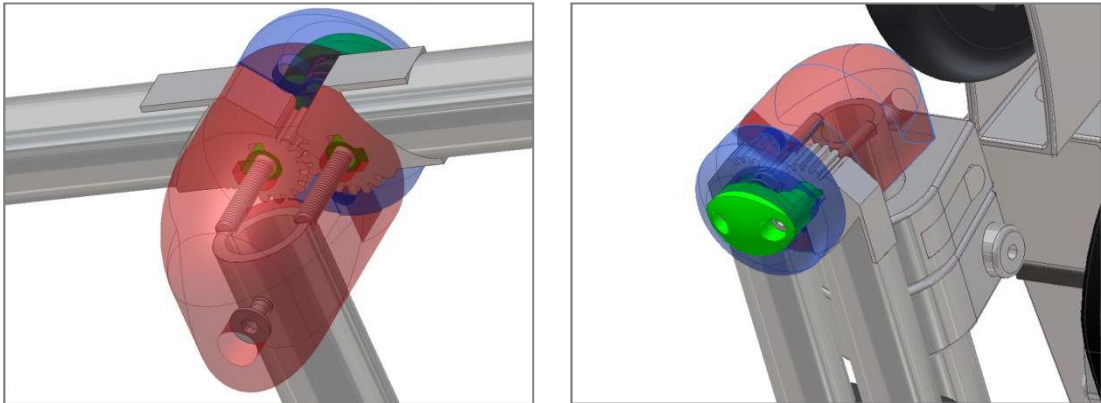
En las siguientes imágenes se muestra su atractivo diseño, así como lo espectacular del patinete totalmente plegado. Destaca mucho la diferencia de dimensiones entre el producto totalmente estirado y el mismo totalmente plegado, dejando patente la labor de diseño realizada:



En la parte superior izquierda, vemos el patinete en su configuración más amplia; totalmente extendido, pudiendo apreciar el detalle de la base en la imagen de abajo. En la parte central vemos el patinete con la estructura cerrada. Y por último, observamos el patinete completamente plegado. Nótese la diferencia de tamaño con respecto a la primera imagen.

Después del estudio de numerosas alternativas, decidimos que las mejores soluciones para cada sistema de plegado son las siguientes:

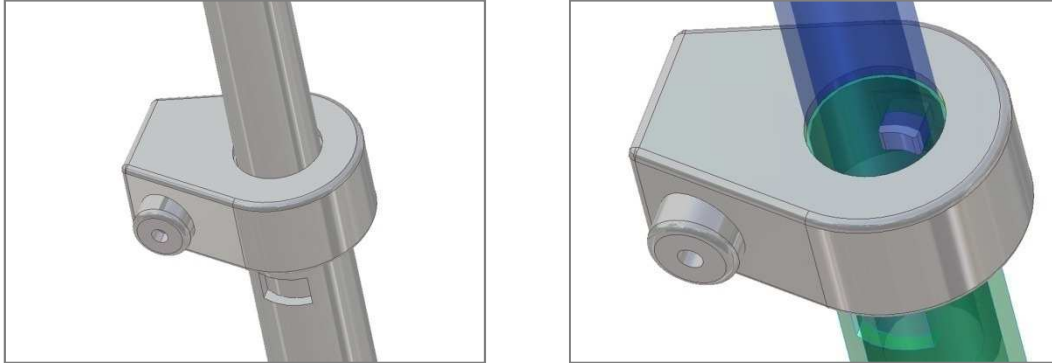
2.2.1-Plegado del Manillar



Para el plegado del manillar se pulsa el botón que aparece de color verde en la imagen; de esta manera se liberan las barras, y mediante un sistema de engranes, las barras giran hasta la posición de plegado. Al engranar una con otra, basta con mover una de ellas para que ambas cambien de posición.

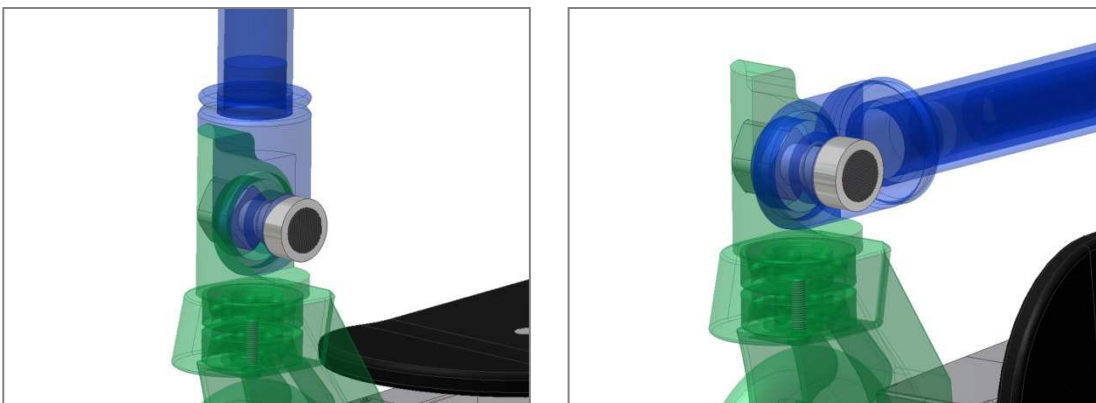
Gracias a unos muelles de compresión, el botón vuelve a su estado inicial, quedando las barras del manillar fijas en esta nueva posición.

2.2.3-Alargamiento/Acortamiento de la Columna de dirección



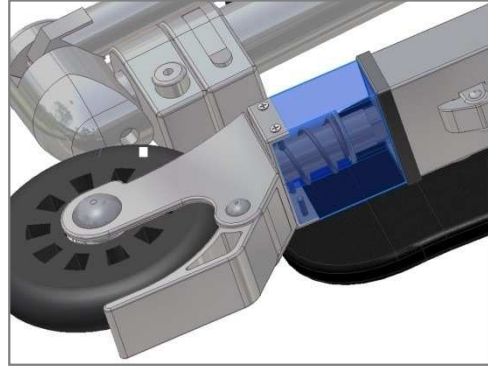
El bloqueo de las barras de dirección es muy sencillo. Pulsando el botón liberamos las dos barras enganchadas, bajamos la barra hasta la posición de plegado, donde la barra queda automáticamente fija, gracias a un muelle que hace que la pieza de bloqueo recupere su posición.

2.2.4-Abatimiento de la Columna de dirección



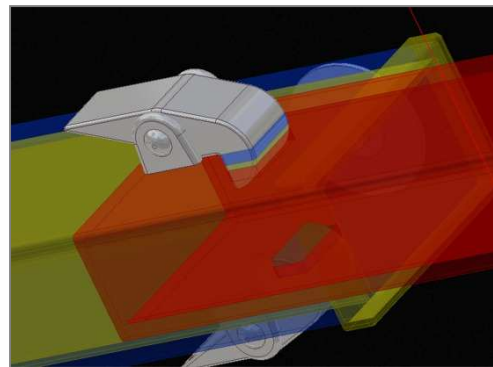
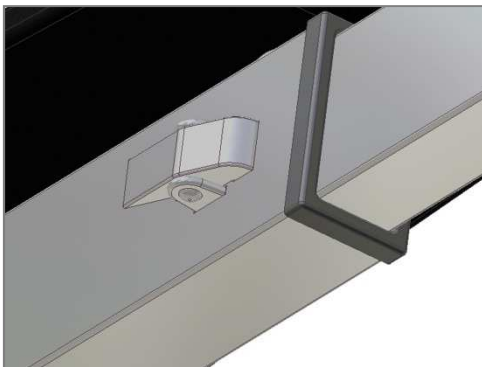
El sistema de abatimiento sigue la misma filosofía que los sistemas anteriores. Una pieza de bloqueo permite dos posiciones; abierta, en su posición de uso, y cerrada o plegada. Tras pulsar el botón, se desbloquea el sistema permitiendo el giro de la dirección. Una vez en la posición plegada, el bloqueo vuelve a actuar por acción del muelle recuperador.

2.2.5-Giro de las Ruedas



Para plegar la horquilla, tanto la trasera como la delantera, se tira hacia fuera hasta que el extremo, de sección cuadrada, quede totalmente liberado. Se gira hacia la posición paralela a la tabla (90°) y se suelta, volviendo a encajar en el alojamiento gracias a un muelle; como en los casos anteriores.

2.2.6-Estructura de la Base



La estructura es telescópica y consta básicamente, de dos tubos de sección rectangular, donde uno se desliza por el interior del otro permitiendo dos posibles longitudes. Para comprimir la estructura de la base, tenemos que actuar sobre dos pinzas laterales. Una vez liberada la pieza de menor sección, se desliza dentro de la de mayor sección hasta llegar a la posición de plegado; soltando las pinzas para que bloqueen nuevamente el conjunto.

Gracias a la combinación de todos estos sistemas conseguimos un patinete que ocupa las dos terceras partes de uno normal, una vez plegado.

2.3-Ingeniería

Los mayores problemas surgieron a la hora de realizar los cálculos de resistencia de nuestro patinete frente a las condiciones que exige la norma. Teniendo que pasar por varios procesos de rediseño de algunas piezas como el sistema de plegado de la barra de dirección o la horquilla, entre otras.

El estudio se llevó a cabo sobre mallas tridimensionales de cada uno de los conjuntos analizados, utilizando modelos de elementos finitos para el cálculo de esfuerzos y deformaciones. También se han calculado todo tipo de elementos normalizados que debía incluir nuestro producto.

En el proceso de diseño, se dimensionaron todas y cada una de las piezas; teniendo en cuenta los ajustes, tolerancias, y estados superficiales necesarios para su fabricación y montaje. De todo este proceso, se dejó constancia en los numerosos planos de conjunto y despiece.

Para que el proyecto estuviese totalmente definido se realizaron diagramas de fabricación, montaje y de desarrollo de las diferentes fases. Dando también información de las máquinas herramientas y procesos utilizados.

Se realizó el correspondiente estudio de seguridad tanto del patinete como del supuesto puesto de trabajo y montaje. Haciéndose las indicaciones necesarias y pudiéndose asegurar el cumplimiento de la normativa a tales efectos. En la realización del proyecto, se utilizó aluminio, acero, madera laminada caucho de silicona y caucho.

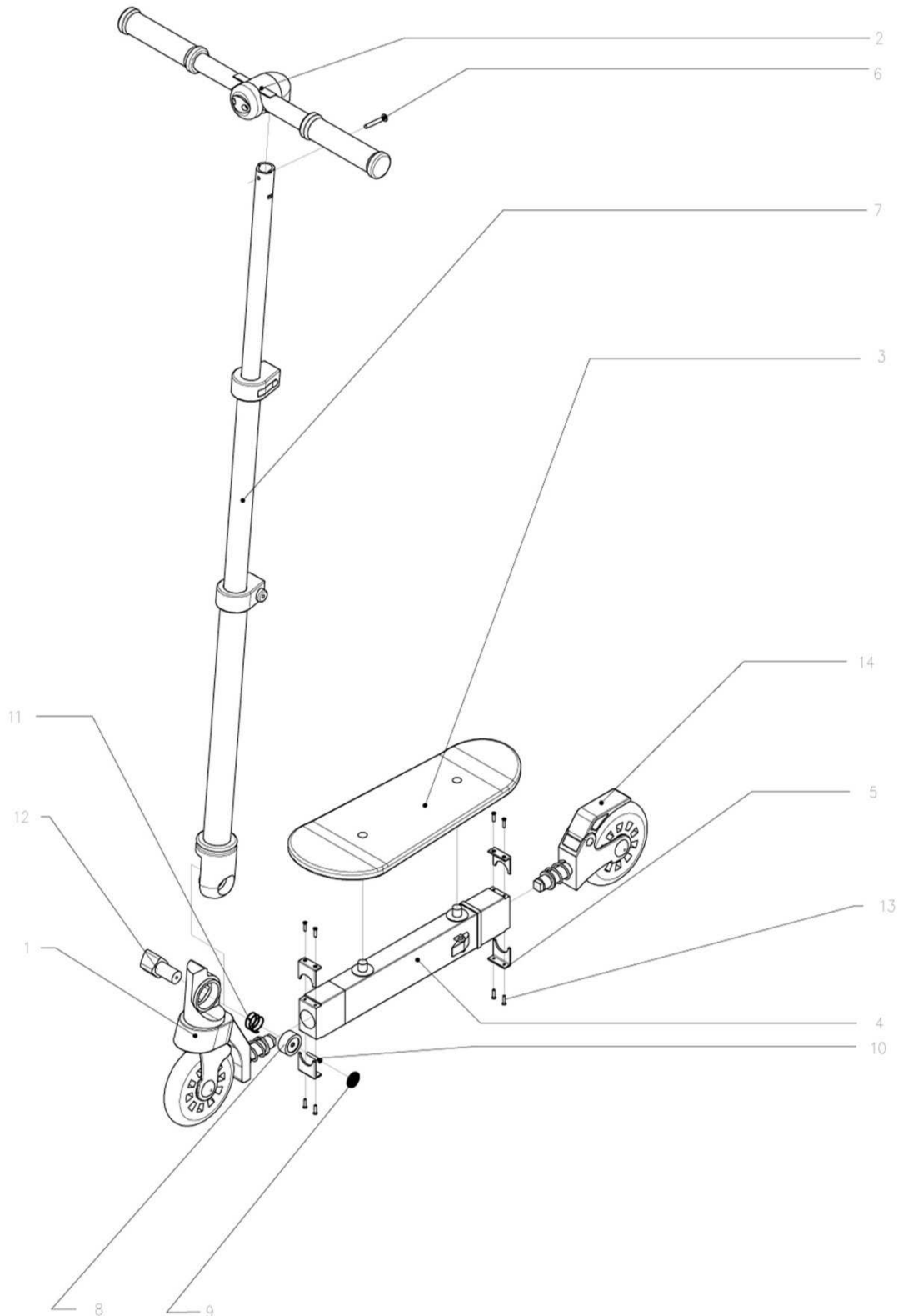
Nuestro producto, así como los materiales de los que está hecho y procesos que se siguen para su fabricación, respetan el medio ambiente, algo fundamental hoy en día. Además en el proyecto se definen algunas formas de reciclado de los materiales utilizados, así como el impacto de algunos de los residuos de estos sobre el medio ambiente.

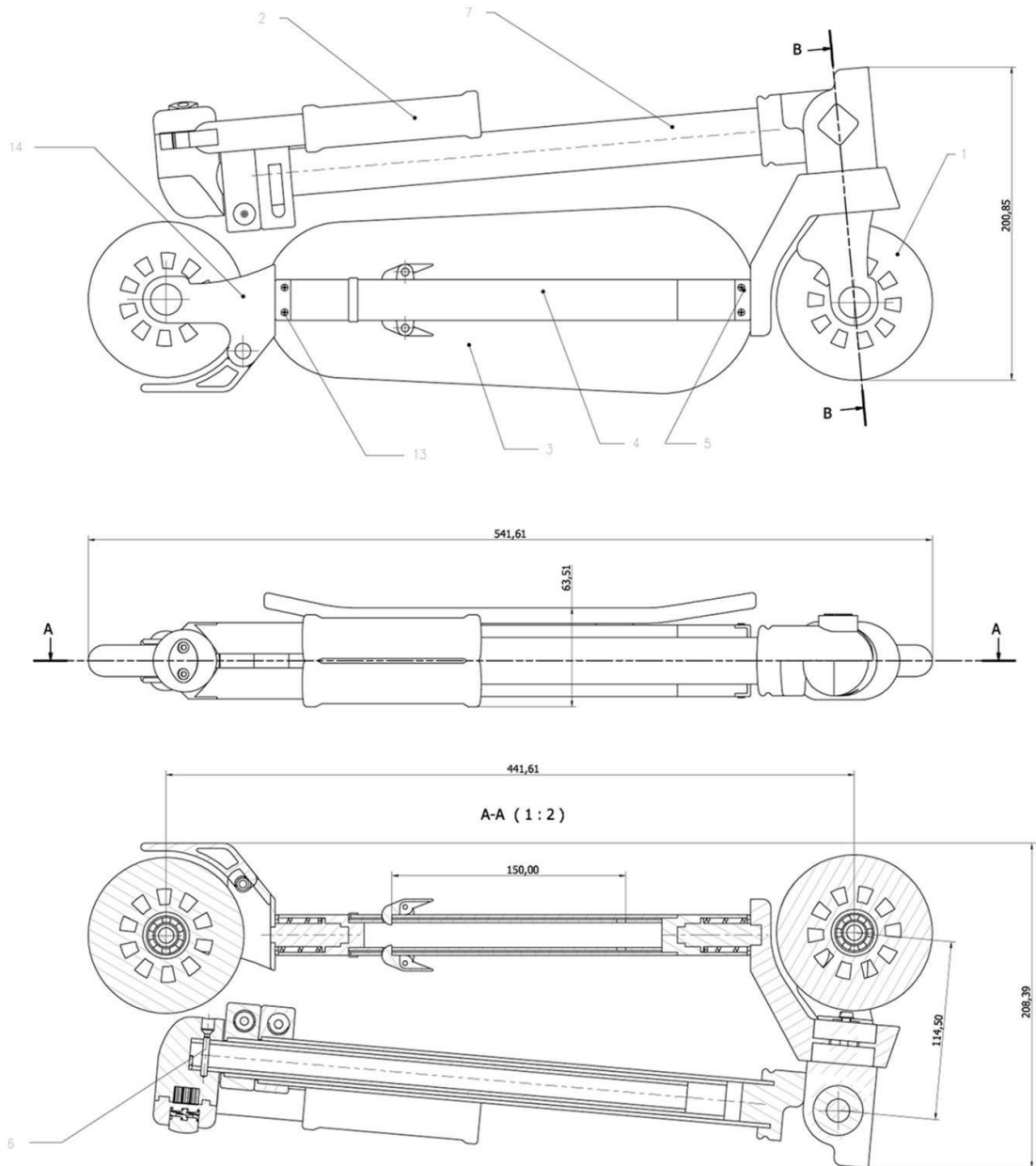


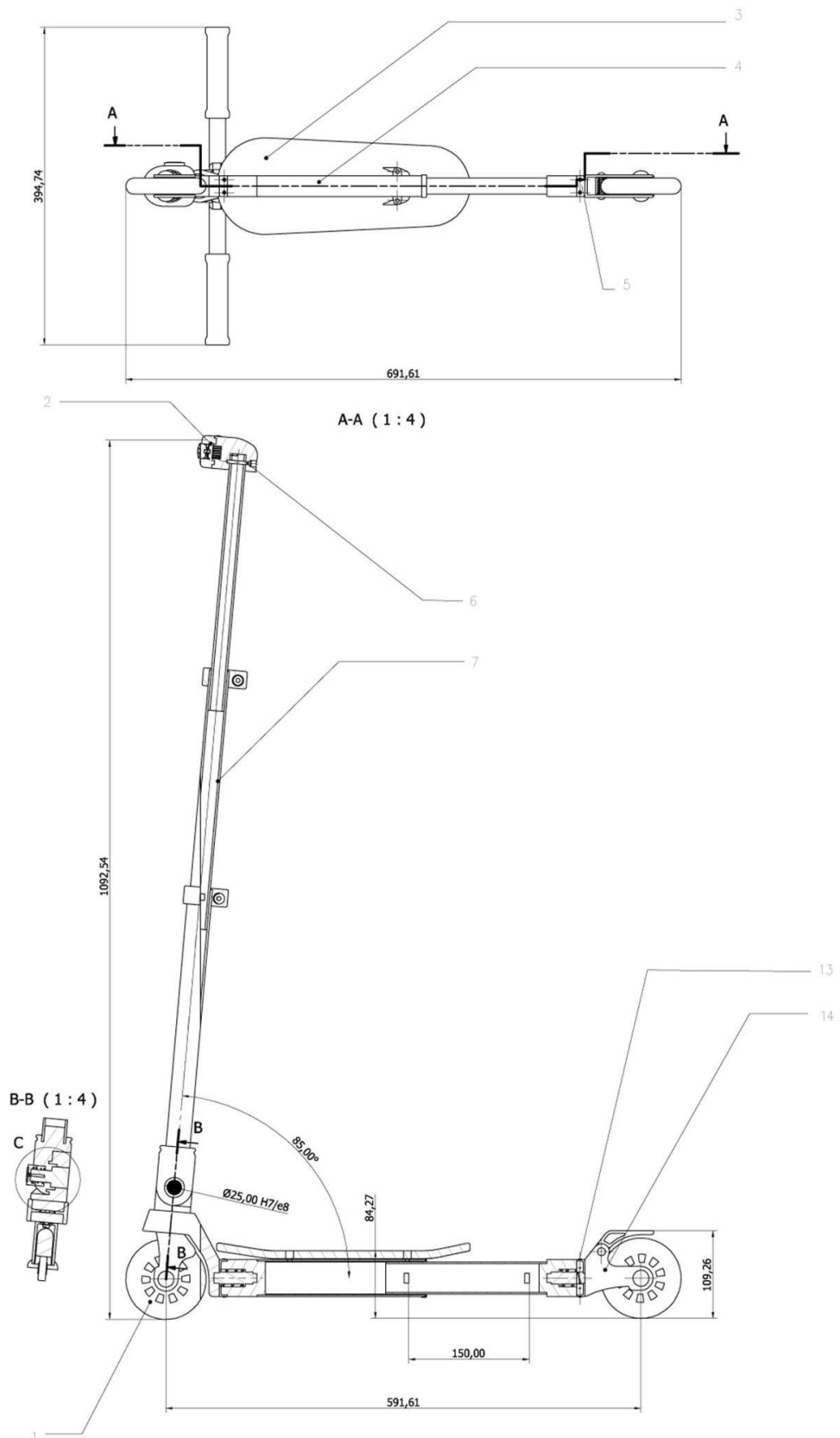
El Patinete fue modelado, en su totalidad, en el programa *Autodesk Inventor Professional 9*. Las piezas originales fueron modeladas, y los elementos normalizados, obtenidos de las bibliotecas propias del programa. Se utilizó este programa, debido a la familiaridad que se tenía con él, al haber sido uno de los programas más utilizados durante la carrera.

A pesar de ello, dicha versión del programa, presentaba algunas limitaciones tales como la dificultad para crear formas y superficies complejas de forma fluida y realista; la carencia de realismo en el acabado de las piezas; viéndose marcadas todas las aristas y las limitadas posibilidades de exportación y animación.

2.4-Planos







1	Subconj. Rueda trasera	14	Planos S03 - S04	
8	Tornillo M3 x 10	13	EN 20225:1991	Acero
1	Bloqueador	12	Plano P38	Acero Inoxidable
1	Muelle-Bloqueador	11	ISO 2161-2	Acero
1	Tornillo M3 x 20	10	EN 20225:1991	Acero
1	Goma	9	Plano P40	Caucho de Silicona
1	Botón Abatimiento	8	Plano P39	Acero Inoxidable
1	Subconj. Columna de dirección	7	Planos S05 - S06	
1	Tornillo M4 x 30	6	EN 20225:1991	Acero
4	Tapa	5	Plano P07	Acero Inoxidable
1	Subconjunto Plataforma	4	Planos S01 - S02	
1	Tabla	3	Plano P17	Madera laminada
1	Subconjunto Manillar	2	Planos S09 - S10	
1	Subconj. Rueda Delantera	1	Planos S07 - S08	
NºPiezas	Denominación	Marca	Referencia	Material
	Fecha	Nombre	UNIVERSIDAD DE VALLADOLID Dpto. EXPRES. GRAF. EN LA ING.	
Dibujado	Ag-2007	Sánchez-Núñez		
Compro.				
Escala			Sustituido por:	
			Sustituye a:	
			Hoja nº:	Nº Hojas:
			Plano nº:	

Ejemplo de cajetín utilizado en uno de los planos generales.

2.5-Conclusiones

Como conclusiones apuntar que desde el punto de vista del diseño nuestro producto tendría una gran aceptación, al haberse conseguido plasmar ese aspecto juvenil, dinámico y sólido.

Por otra parte conseguimos abarcar un amplio rango de alturas; y por lo tanto de usuarios, gracias a las tres posiciones de la columna de dirección. Con una altura máxima del manillar de aproximadamente 1m y mínima de 45cm, así como dos posibles posiciones que tiene la estructura de la base.

Destacar la consecución de nuestro principal objetivo que era el de conseguir un patinete lo más compactable y ligero posible para su fácil transporte y almacenamiento. Reduciendo la longitud total a las dos terceras partes de un patinete normal, con un peso total de menos de 3,9 kilogramos lo que es bastante reducido teniendo en cuenta otros patinetes metálicos. Esto lo hemos conseguido gracias a un innovador diseño y a la utilización de materiales de última generación como es el aluminio Al 7005-T6.

El precio final de venta al público; de 45€, es totalmente asumible para cualquier tipo de usuario que busque un producto de características similares y cierto nivel de calidad.

Otra importante conclusión que sacamos es la de que debido a la complejidad de algunas piezas y mecanismos, y a la alta calidad del producto la tirada de producción tiene que ser baja.

En cuanto a seguridad, el patinete cumple con la normativa al respecto, e incluye información sobre el equipamiento de protección recomendado durante su utilización.

La última conclusión es que gracias a la utilización de materiales totalmente reciclables, el patinete “COMPACT” es respetuoso con el medio ambiente. Tema muy delicado y de actualidad que hemos tenido especialmente en cuenta a lo largo del desarrollo de nuestro proyecto.

Debido a las limitaciones encontradas en el uso del programa *Autodesk Inventor Professional 9*, en cuanto a representación; cómo antes se ha mencionado, y al escaso tiempo del que se disponía, el proyecto se centró así, en la parte de diseño y aspectos técnicos, quedando poco completa la representación del producto, aspecto que aborda este nuevo trabajo. Para ello, hacemos uso del programa *Autodesk 3ds Max Design 2013*, sirviéndonos de sus posibilidades de representación realista, así como de la posibilidad de crear animaciones en las que se muestre el funcionamiento de los componentes y el montaje de las piezas del producto.



En este capítulo se aborda la fase de exportación de imágenes a partir de un modelo 3D. Configuraremos una escena con otros elementos, materiales e iluminación; en la que integraremos el producto, para poder exportar las imágenes más oportunas y representativas.

3.1-Importar Archivos a Autodesk 3D Studio Max

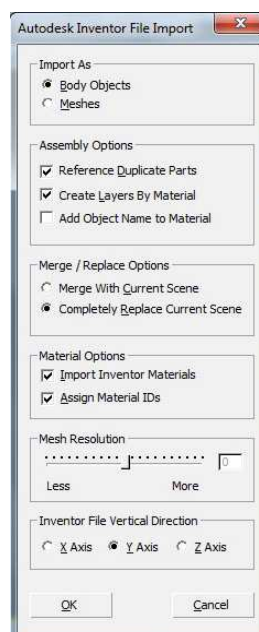
Para obtener unas imágenes de mayor calidad y realismo, se opta por realizar una nueva composición; esta vez en Autodesk 3D Studio. En concreto se usará la versión *Autodesk 3ds Max Design 2013*.

Existen ciertas complicaciones a tal efecto. Los archivos originales, realizados en Autodesk Inventor Professional 9, no se pueden abrir directamente en esta nueva versión de 3ds Max. Para ello se actualizan en la versión nueva de Inventor; *Autodesk Inventor Professional 2013*.

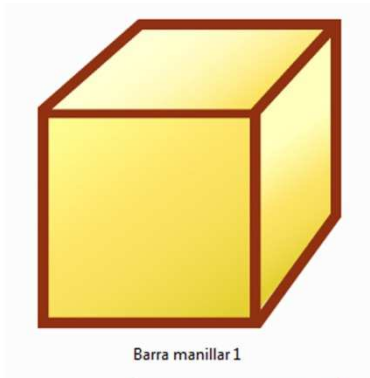
Los archivos guardados en esta nueva versión de Inventor, ya son compatibles con la nueva versión de 3ds Max. En este punto del proceso, se presenta un problema con los elementos normalizados que se utilizaron en la versión antigua. En ocasiones las bibliotecas de los programas varían, dando lugar a posibles pérdidas de elementos. Con lo cual, debemos buscar en las nuevas bibliotecas, elementos que se ajusten a nuestras necesidades. O en caso extremo, modelar nuevos elementos, con las características específicas necesarias.

A pesar de que la nueva versión de Inventor dista mucho de la utilizada en un principio; y ofrece mejores acabados así como un mayor realismo, se sigue optando por el 3ds Max, gracias a sus posibilidades para definir los materiales, propiedades de la escena y posibilidades de animación.

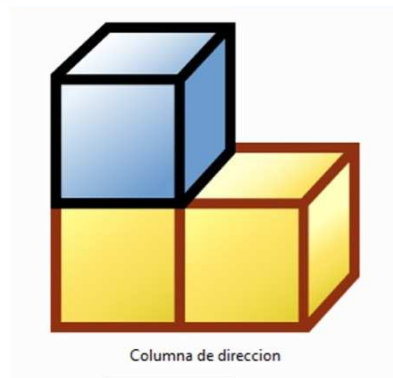
Al abrir o importar las piezas creadas en Inventor, nos aparecerá un cuadro de diálogo similar a este



Podemos trabajar directamente con las opciones por defecto. De esta forma, podemos abrir en 3ds Max, piezas creadas en Inventor (archivos con extensión .ipt), o conjuntos de varias piezas (archivos con extensión .iam).

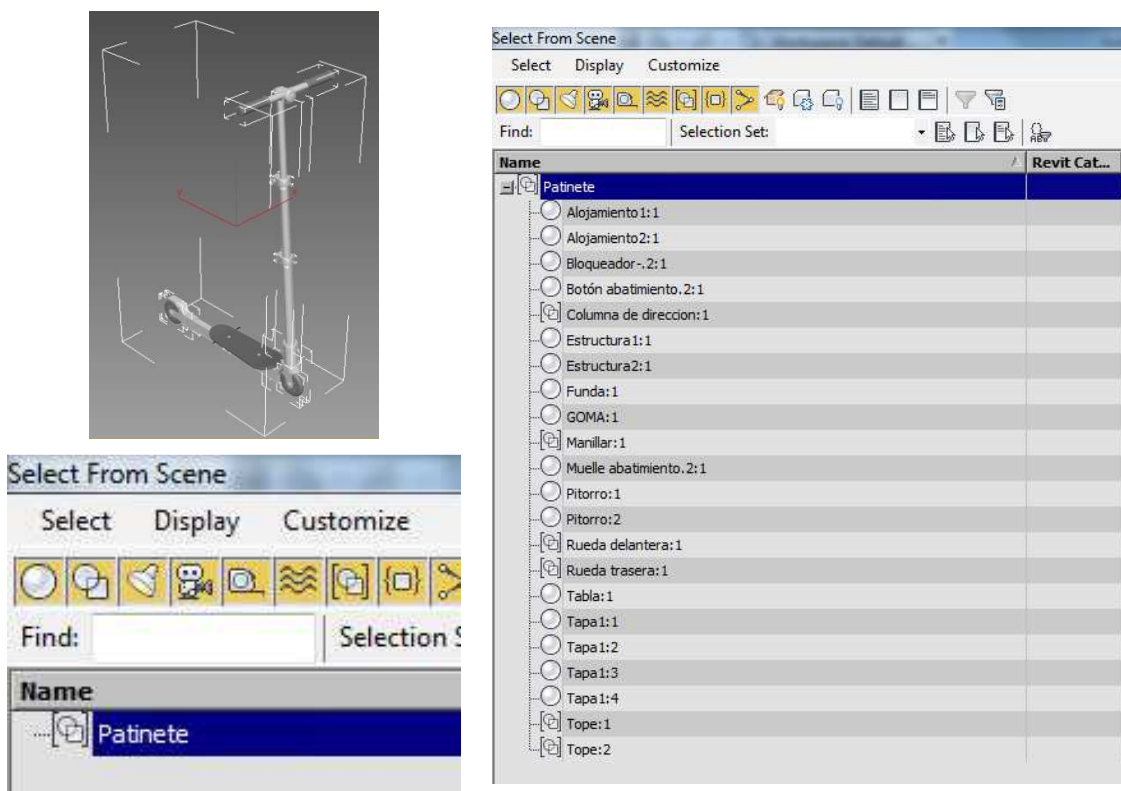


archivos de piezas, con extensión .ipt



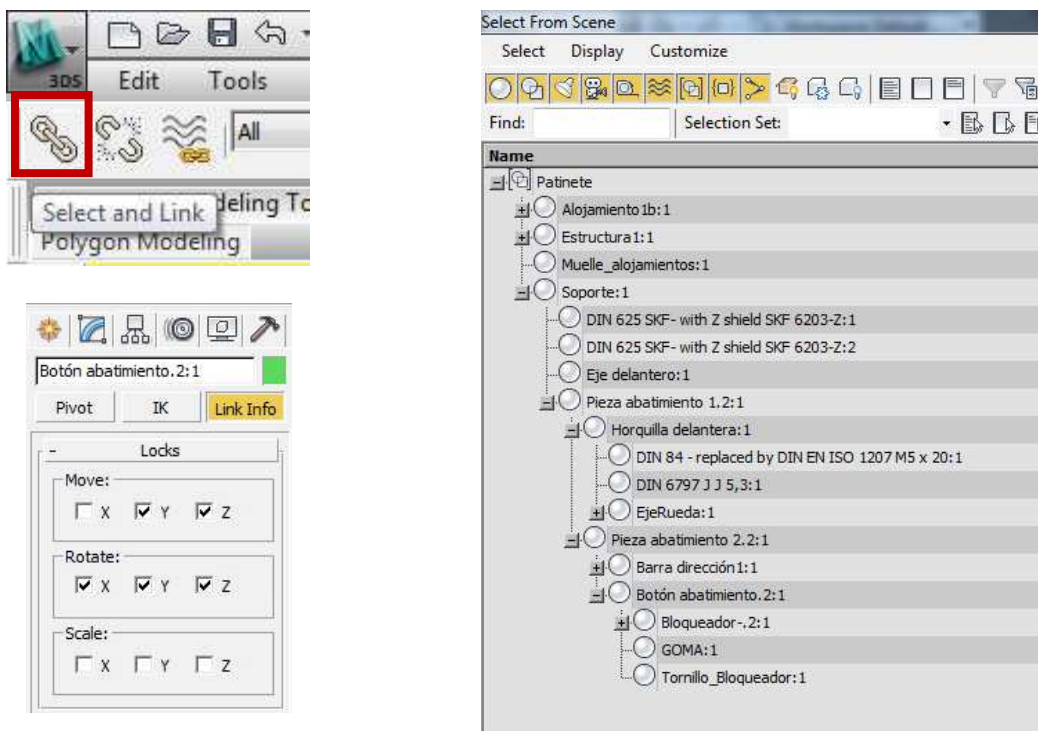
archivos de conjunto, con extensión .iam

Cuando se trate de archivos de conjunto, en 3ds Max, éste aparecerá como un grupo cerrado. Dicho grupo mantendrá el nombre del archivo .iam, y cada uno de los elementos del grupo, mantendrá a su vez el nombre del archivo .ipt en que fueron creados; lo cual facilitará mucho el trabajo.

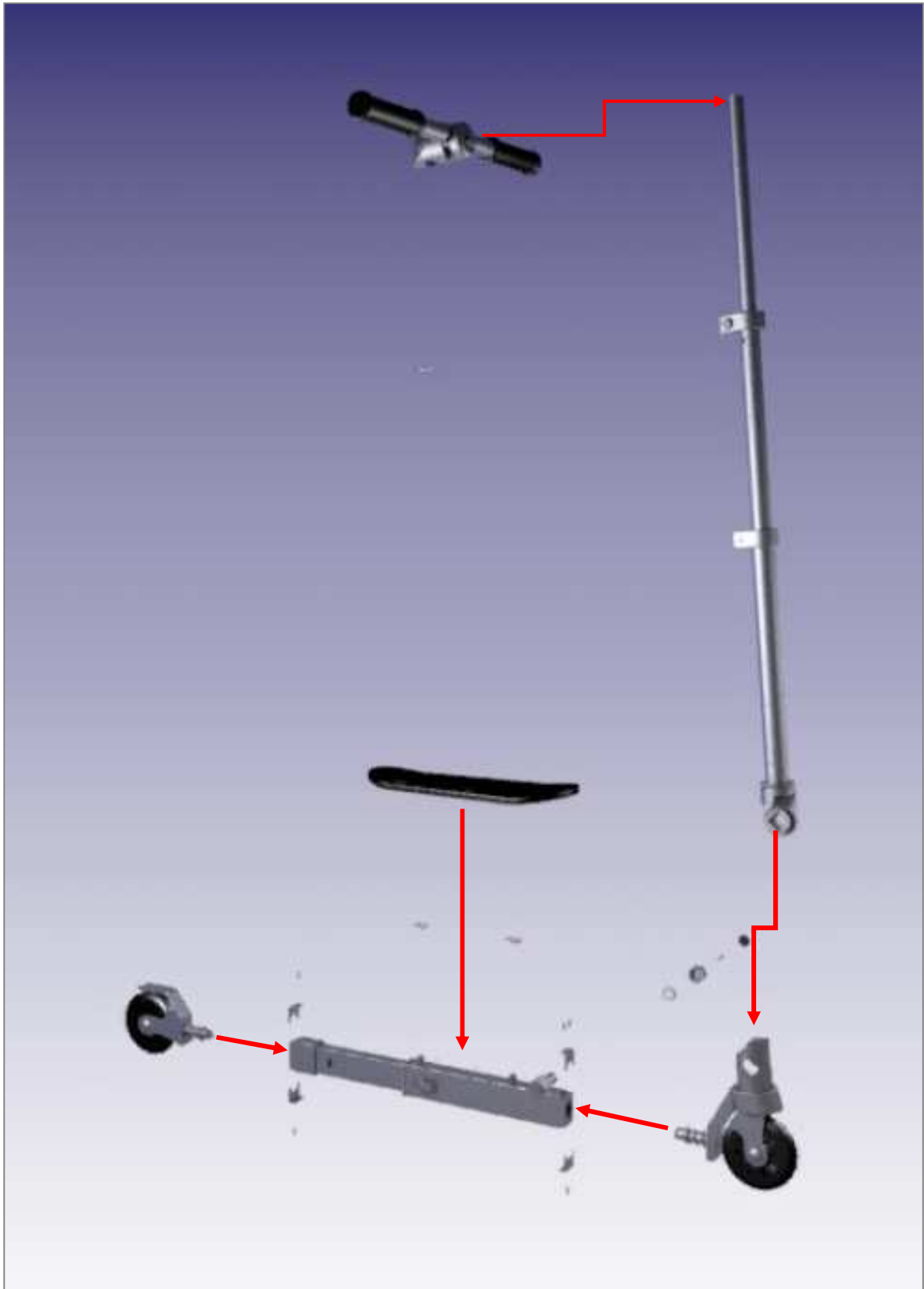


3.2-Jerarquías

Pensando en que posteriormente se animará el producto; es el momento de crear las relaciones necesarias, entre los distintos elementos del mismo. Esto quiere decir, que utilizaremos jerarquías para definir y limitar cada uno de los movimientos de las piezas que intervendrán en la animación.



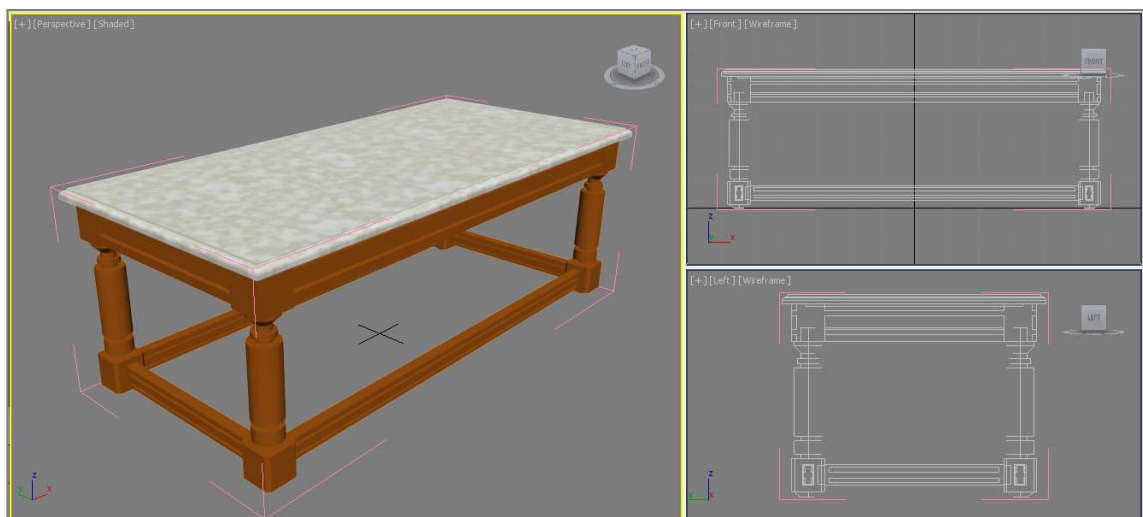
En la siguiente imagen se ve, de forma muy simplificada, la relación que existe entre las diferentes piezas o conjuntos:



3.3-Entorno

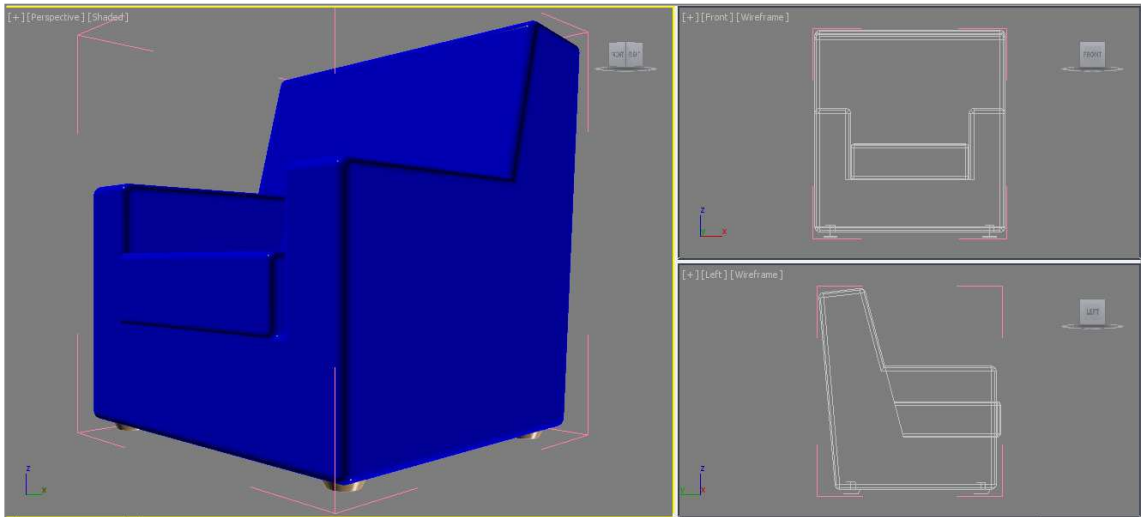
Para completar la escena, se añaden elementos como suelo, paredes y algunos objetos para crear un entorno. Primeramente, crearemos una habitación muy sencilla, donde situar el patinete. Esto es, unas paredes un techo y un suelo. A continuación, crearemos algún otro objeto que acompañe al patinete en la escena, para así crear un entorno.

3.3.1-Mesa



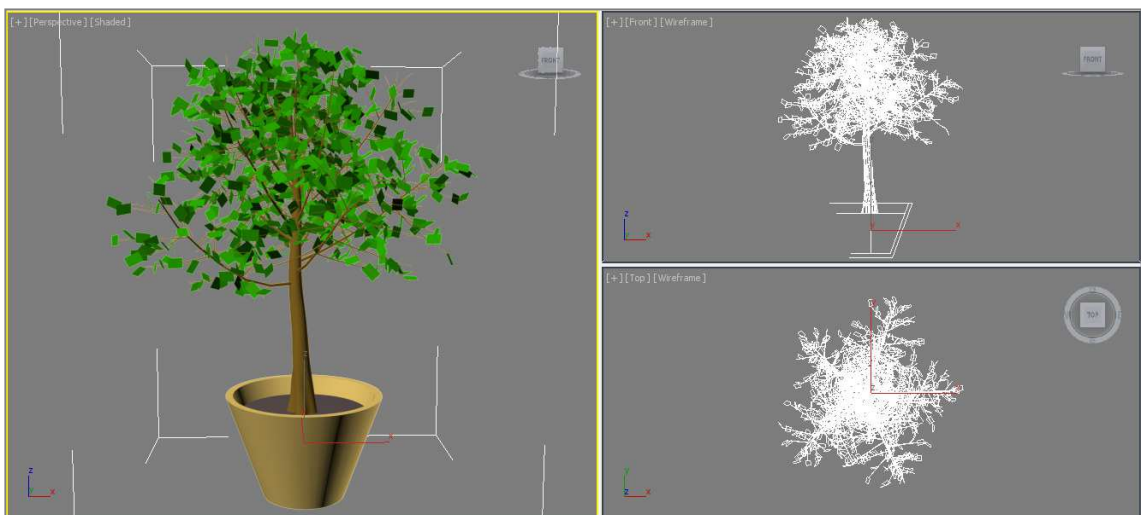
Mesa de madera y mármol. Las dimensiones de la mesa son 120 x 56 x 43cm. Modelada íntegramente para este proyecto; como un elemento más de la escena en la que se integrará el patinete, y de la que se sacarán las imágenes, posteriormente.

3.3.2-Sofá



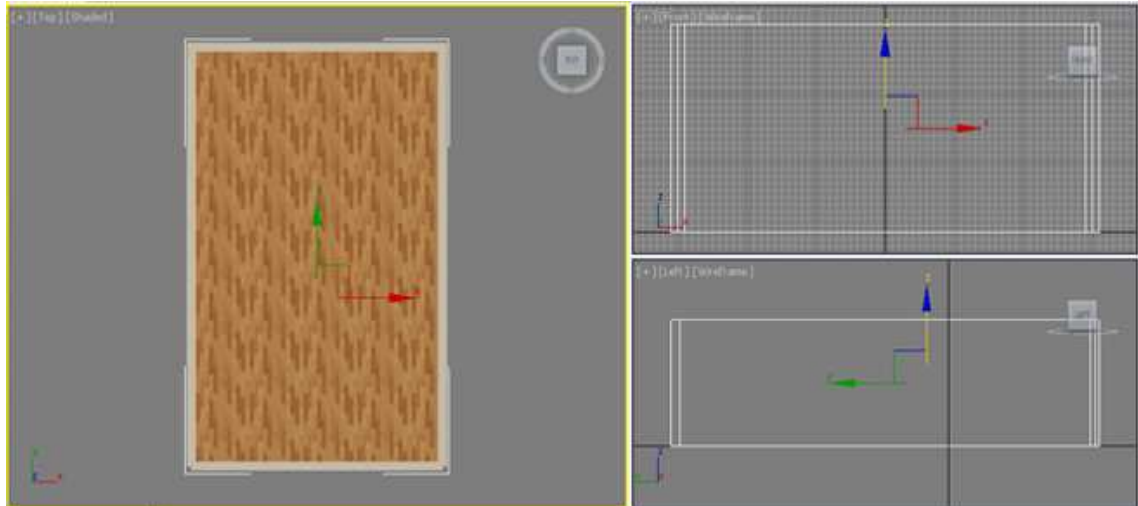
Sofá de tela y patas de madera. Objeto modelado también para la escena, de líneas rectas y geometría sencilla.

3.3.3-Planta



Planta con tiesto. En este caso, se modela un tiesto y la tierra; utilizando un elemento *Foliage*, dentro de *AEC Extended*, en la sección *Geometry* de *Autodesk 3d Studio Max Design*. Se crea, para ello, un árbol de pequeño tamaño.

3.3.4-Habitáculo

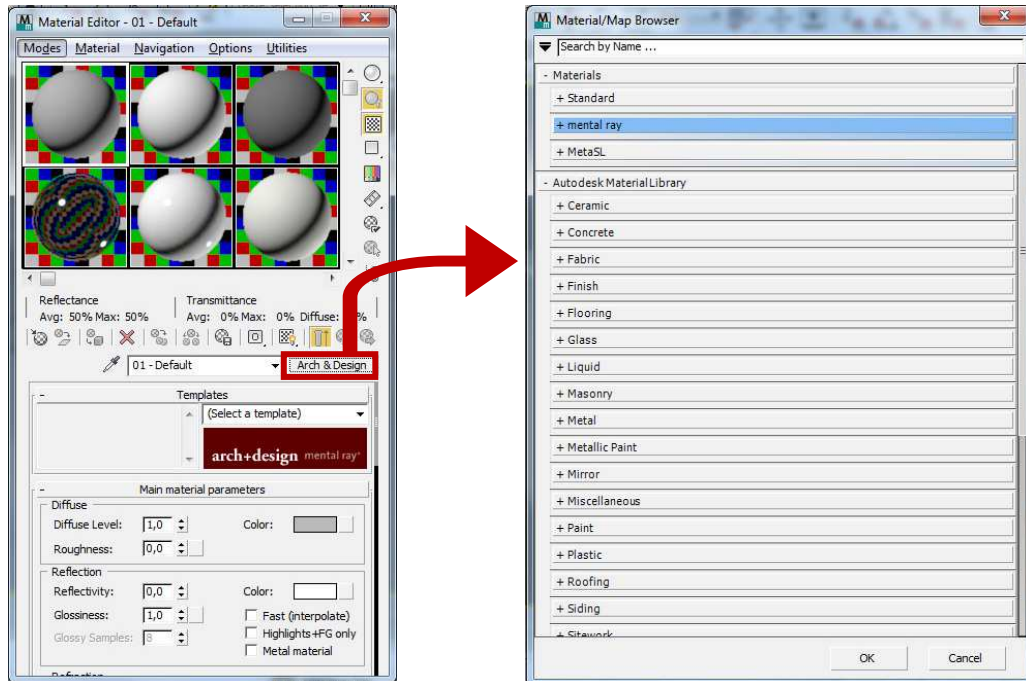


Para contener los demás elementos, se crean cuatro paredes, un suelo y un techo, a modo de habitación.

Una vez tenemos el modelo en 3ds Max, debemos definir las propiedades tales como materiales e iluminación, para posteriormente realizar los render y las animaciones.

3.4-Materiales Max

Para las imágenes y animaciones, se necesitan diferentes acabados y grados de realismo, lo cual se obtendrá con diversos materiales, que se obtienen de las bibliotecas de 3ds max, así como creando materiales nuevos.

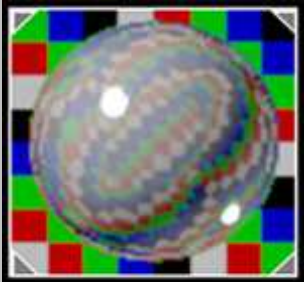
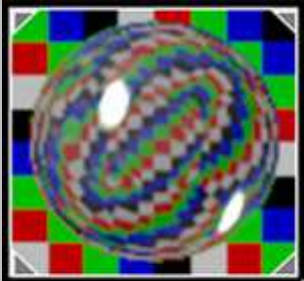


Principalmente, se necesita definir cada uno de los materiales de los que está compuesto el patinete. A continuación se especifican los materiales utilizados en *3ds Max Design*, para la representación de la escena. Éstos están desglosados en materiales propios del patinete; como objeto principal de la escena, y también se especifican los materiales de cada uno de los objetos que le acompañan.

Para cada uno de estos materiales, se tendrán en cuenta propiedades físicas como reflexión y refracción, así como texturas que tendrían éstos en la realidad, para tratar de que el modelo 3D sea lo más fiel posible al producto real.

3.4.1-Materiales propios del Patinete

METAL

	<p>Aluminio:</p> <p>Para las piezas de aluminio, se busca un acabado brillante. Se escoge en el buscador de materiales, un <i>Autodesk Metal</i>; dentro de <i>Mental Ray</i>.</p> <p><u>Type:</u> Chrome</p> <p><u>Finish:</u> Semi-polished</p> <p>En la sección Performance Tuning: Reflection Glossy Samples: 32</p>
	<p>Acero Inoxidable:</p> <p>Para las piezas de acero, se escoge en el buscador de materiales, un <i>Autodesk Metal</i>; dentro de <i>Mental Ray</i>; de la misma forma que el Aluminio, buscando que el acabado sea brillante.</p> <p><u>Type:</u> Stainless Steel</p> <p><u>Finish:</u> Polished</p> <p>En la sección Performance Tuning: Reflection Glossy Samples: 32</p>

PLASTICO



Plástico Blanco:

Existen unas arandelas de plástico, el cual queremos que sea blanco; imitando a POM (polioximetileno). Para ellas, se escoge en el buscador de materiales, un *Autodesk Plastic/Vinyl*; dentro de *Mental Ray*.

Type: Plastic (Solid)

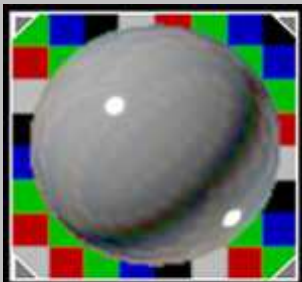
Color: Use Color (Se escoge el blanco en el selector de colores)

Finish: Polished

En la sección Performance Tuning:

Reflection Glossy Samples: 32

Refraction Glossy Samples: 32



Plástico Gris:

Este material, se utilizará para la pieza llamada “funda”, que se encuentra entre los dos tubos de sección rectangular que forma la plataforma del patinete. Gracias a ella, se elimina el roce entre las piezas metálicas, favoreciendo su deslizamiento.

Para esta pieza, se escoge en el buscador de materiales, un *Autodesk Plastic/Vinyl*; dentro de *Mental Ray*, de la misma forma que el Plástico Blanco.

Type: Plastic (Solid)

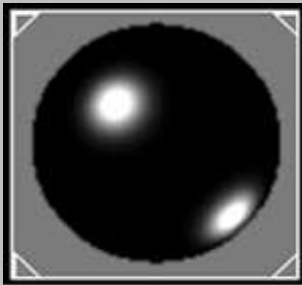
Color: Use Color (Se escoge un gris en el selector de colores)

Finish: Polished

En la sección Performance Tuning:

Reflection Glossy Samples: 32

Refraction Glossy Samples: 32

**Tabla:**

A la tabla, como ejemplo, se le da un acabado negro brillante, como de plástico o resina. Esta vez, crearemos un material nuevo, utilizando un *Standard*.

En las opciones básicas de sombreado, Shader Basic Parameters, elegimos el tipo Blinn, adecuado para plásticos.

Ambient: Negro

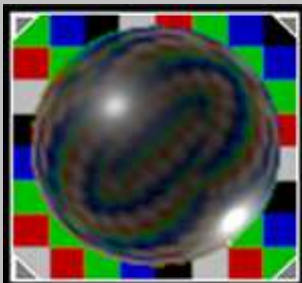
Difusse: Negro (Color del Material)

Specular: Blanco (El color del brillo)

En la sección Specular Highlights:

Specular Level: 100

Glossiness: 46

GOMA**Goma Negra para Manguitos:**

Para las piezas de goma del manillar, se escoge en el buscador de materiales, un *Autodesk Plastic/Vinyl*; dentro de *Mental Ray*.

Type: Plastic (Solid)

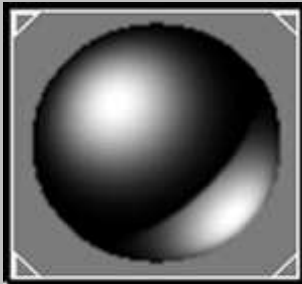
Color: Use Color (Se escoge el negro en el selector de colores)

Finish: Glossy

En la sección Performance Tuning:

Reflection Glossy Samples: 32

Refraction Glossy Samples: 32



Goma Negra para Ruedas:

En esta ocasión, se trata también de un material *Standard*. El objetivo es obtener un material de aspecto similar a una silicona negra y mate.

En las opciones básicas de sombreado, Shader Basic Parameters, elegimos nuevamente, el tipo Blinn.

Ambient: Negro

Difusse: Negro (Color del Material)

Specular: Blanco (El color del brillo)

En la sección Specular Highlights:

Specular Level: 117

Glossiness: 19

Soften: 0,1

Se aplica una imagen como mapa de abombamiento (Bump), para dar algo de rugosidad a la goma.

3.4.2-Materiales del Habitáculo



Paredes (Pintura):

Para la pintura de las paredes, se escoge en el buscador de materiales, un *Autodesk Wall Paint*; dentro de *Mental Ray*.

Pintura de la pared; Wall Paint:

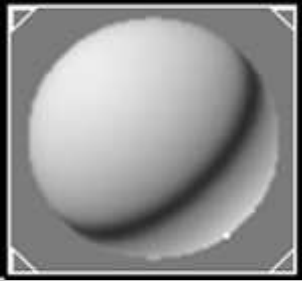
Color: Use Color (En este caso, se escoge un beige claro en el selector de colores)

Finish: Eggshell


Application: Spray

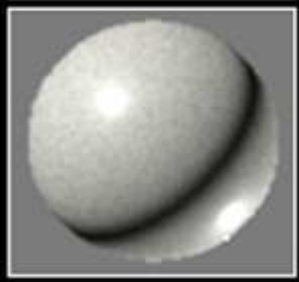
En la sección Performance Tuning:

Reflection Glossy Samples: 32


	<p>Techo (Pintura):</p> <p>Para la pintura del techo, se usará también un <i>Autodesk Wall Paint</i>; dentro de <i>Mental Ray</i>.</p> <p>En la sección Wall Paint:</p> <p><u>Color:</u> Use Color (Para el techo, escogemos blanco en el selector de colores)</p> <p><u>Finish:</u> Flat/Matte</p> <p><u>Application:</u> Spray</p> <p>En la sección Performance Tuning:</p> <p>Reflection Glossy Samples: 32</p>
---	---

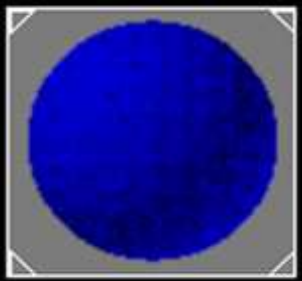
3.4.3-Materiales de la Mesa

	<p>Madera:</p> <p>En este caso, probaremos con un Autodesk Generic; dentro de <i>Mental Ray</i></p> <p><u>Image:</u> Se selecciona un mapa de una de las bibliotecas de 3ds; una imagen predefinida, como muestra del patrón para la textura de la madera: <code>Woods&Plastics.FinishCarpentry.Wood.Cocobolo.jpg</code></p> <p>En la sección Performance Tuning:</p> <p>Reflection Glossy Samples: 12</p> <p>Reflection Glossy Samples: 12</p>
---	--


	<p>Mármol:</p> <p>Para la placa de la mesa, se busca un material similar al mármol. Para ello, se escoge en el buscador de materiales, un <i>Autodesk Stone</i>; dentro de <i>Mental Ray</i>.</p> <p>Para definir el tipo de piedra; Stone:</p> <p><u>Image</u>: Se selecciona un mapa de una de las bibliotecas de 3ds; una imagen predefinida, como muestra del patrón para la textura del mármol.</p> <p>Masonry.Stone.Alabaster.png</p> <p><u>Finish</u>: Polished</p> <p>En la sección Performance Tuning:</p> <p>Reflection Glossy Samples: 32</p>
---	---

3.4.3-Materiales del Sofá

	<p>Madera:</p> <p>Al igual que con la mesa, con un <i>Autodesk Generic</i>; dentro de <i>Mental Ray</i></p> <p><u>Image</u>: Se selecciona un mapa de una de las bibliotecas de 3ds; una imagen predefinida, como muestra del patrón para la textura del mármol:</p> <p>Woods-Plastics.FinishCarpentry.Wood.Panelling.1.jpg (Width:2,8 Height:1 Angle U:68 V:49 W:0)</p> <p>En la sección Performance Tuning:</p> <p>Reflection Glossy Samples: 12</p> <p>Reflection Glossy Samples: 12 .</p>
---	--

	<p>Tela:</p> <p>Para la tela del sofá, utilizamos un material <i>Standard</i>, al que le aplicaremos un mapa de relieve para crear una textura.</p> <p>En las opciones básicas de sombreado, Shader Basic Parameters, elegimos nuevamente, el tipo Blinn.</p> <p><u>Ambient:</u> Negro (En este caso)</p> <p><u>Difusse:</u> Azul (Color del Material)</p> <p><u>Specular:</u> Blanco (El color del brillo)</p> <p>En la sección Specular Highlights:</p> <p>Specular Level: 3</p> <p>Glossiness: 5</p> <p>Soften: 0,1</p> <p>En la sección Maps:</p> <p>Bump al 30%, y seleccionamos un mapa para darle textura. En este caso, una imagen que imita el tejido de la tapicería.</p>
---	--

3.4.4-Materiales de la Planta

	<p>Planta:</p> <p>Utilizaremos un elemento <i>Foliage</i>, dentro de <i>AEC Extended</i>, en la sección <i>Geometry</i>.</p> <p>Se escoge un árbol de pequeño tamaño.</p>
---	--

**Tierra:**

Para la tierra se buscará, un material oscuro y granuloso. Por ejemplo con un material *Standard* y un mapa de abombamiento (Bump).

En este caso un mapa Dent con un tamaño de 200.

Para dar algo de relieve, se le puede aplicar un modificador de desplazamiento (Panel *Modify* > *Displace*).

**Tiesto:**

Para el tiesto, se escoge un material similar al utilizado para la placa de mármol de la mesa. Esto es, un material en la sección *Autodesk Ceramic*; dentro de *Mental Ray*.

Para definir el tipo de piedra; Ceramic:

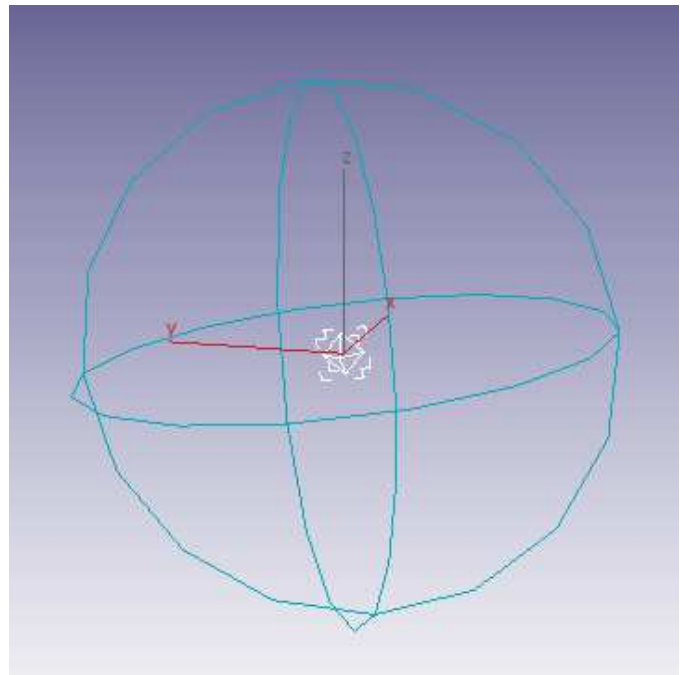
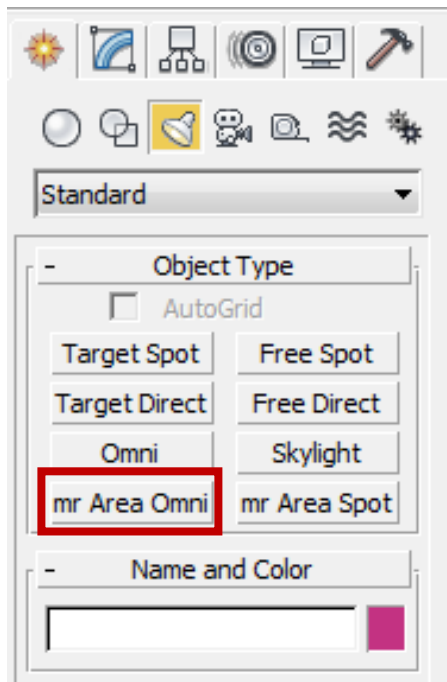
Image: Seleccionaremos alguna imagen que de aspecto de material cerámico.

3.5-Iluminación

Para completar la escena, se colocan unas luces para poder más tarde realizar los render de la misma. Estas luces, se dispondrán alrededor del patinete, por ser el elemento a destacar.

Se elegirán luces del tipo *mr Area Omni*; colocándolas en distintas posiciones y a diferentes alturas, para generar diferentes sombras y reflejos en los materiales definidos.

Para crear las luces, accederemos a la sección *Lights*, dentro del panel *Create*. Escogeremos, en la categoría *Standard*, luces del tipo *mr Area Omni*.



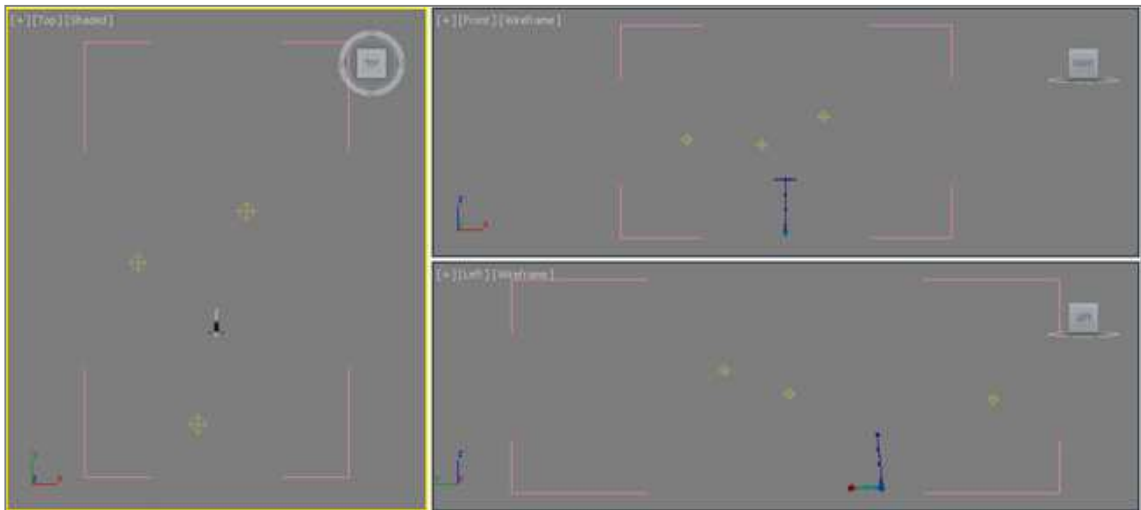
Este tipo de luces, se proyectan desde un área definida (esfera o cilindro) en vez de hacerlo desde un sólo punto. Las luces de área son soportadas solamente por el render Mental Ray.

En la pestaña *Area Light Parameters*, dentro del panel *Modify*; escogeremos en este caso, el tipo esfera (Sphere). De esta forma, la luz es proyectada desde la superficie de una esfera. Jugando con la intensidad de la luz y el radio de la esfera, se obtienen unos efectos u otros.

Para el caso que nos ocupa, tendremos en cuenta lo siguiente:

	Situación	Multiplier (Intensidad)	Forma	Diámetro (mm)
Mr Area Omni 1	(1305,216 , 2545,816 , 2281,52)	2,701	Esfera	40mm
Mr Area Omni 2	(-1305,216 , 1331,139 , 1844,464)	1,041	Esfera	40mm
Mr Area Omni 3	(125,865 , -2545,816 , 1742,024)	1,61	Esfera	40mm

Las luces se disponen de la siguiente forma:

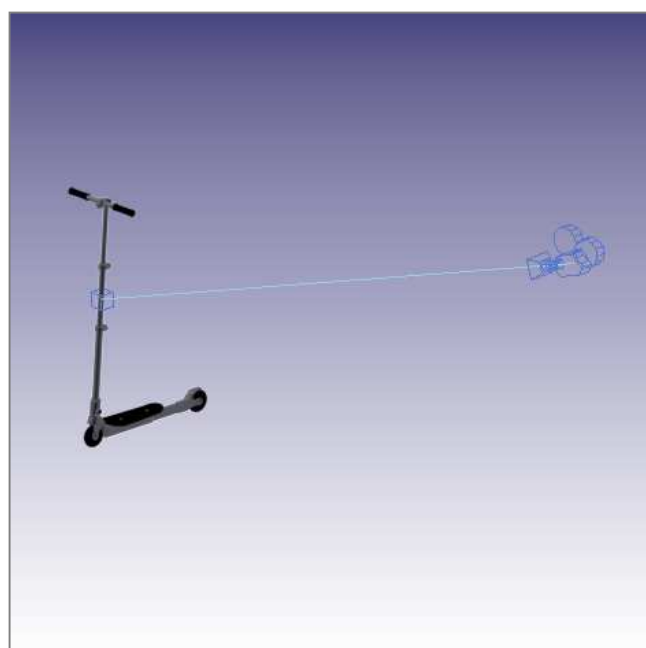
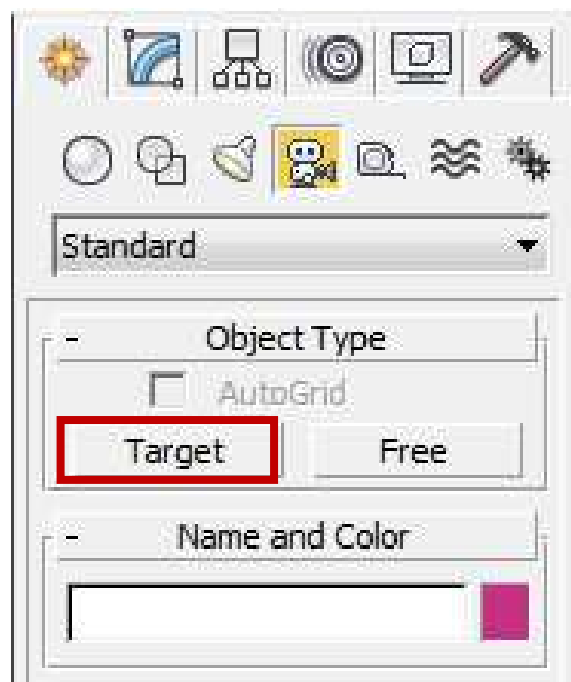


3.6-Cámaras

Para sacar las imágenes utilizaremos cámaras o bien podremos seleccionar las vistas que se quieran con el visor *Perspective*. En este último caso, bastará con colocar el visor en la posición que se quiera, y exportar la imagen.

Para crear la cámara, accederemos a la sección *Cameras*, dentro del panel *Create*. Escogeremos, en la categoría *Standard*, una cámara tipo *Target*.

Lo más adecuado para este propósito es elegir una cámara del tipo *Target*. Este tipo de cámaras tienen un punto de enfoque u objetivo (*Target*), que será muy útil para el tipo de imágenes y vídeos que se quieren realizar; sobre todo, cuando se trate de detalles.

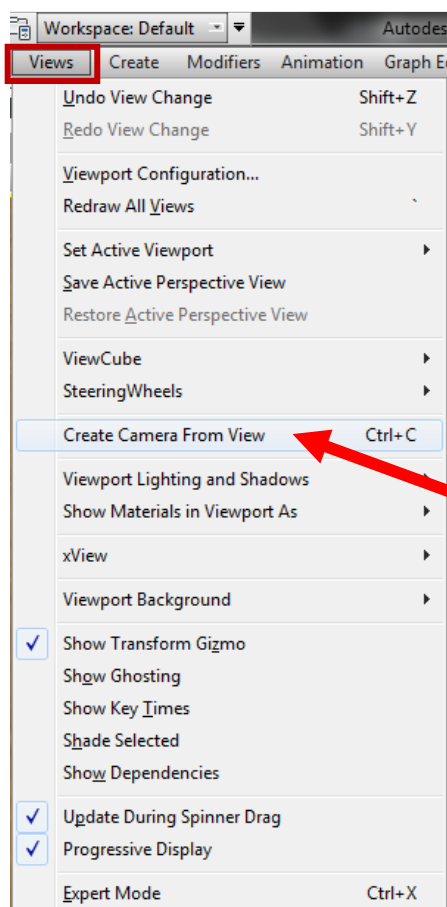


Para obtener diferentes vistas, podremos mover la cámara, el objetivo, o podremos variar el tamaño de la lente o el Campo de visión (FOV; Field Of View); en la pestaña Parameters, dentro del panel Modify.

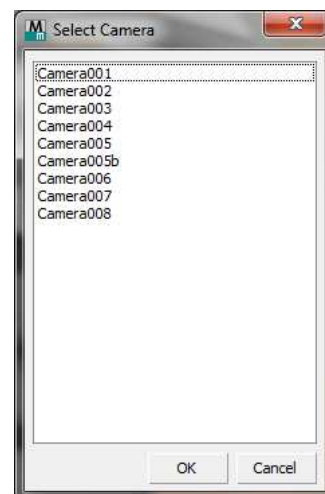
De todo esto, se habla más detalladamente en el capítulo **4-Creación de Video**.

3.6.1-Cámara a partir de vista

Otra forma de crear una cámara, de forma sencilla, será la siguiente. Por ejemplo en la vista *Perspective*, podemos situarnos en cualquier lugar de la escena moviéndonos libremente. Elegido el plano que queremos, podremos crear una cámara a partir de éste, a través del menú *Views*: Views > Create Camera From View.



Podremos cambiar fácilmente de una a otra cámara pulsando la tecla **C**; se abrirá un listado de todas las cámaras que tenemos en la escena, para poder elegir la que se quiera.



3.7-Casos a destacar

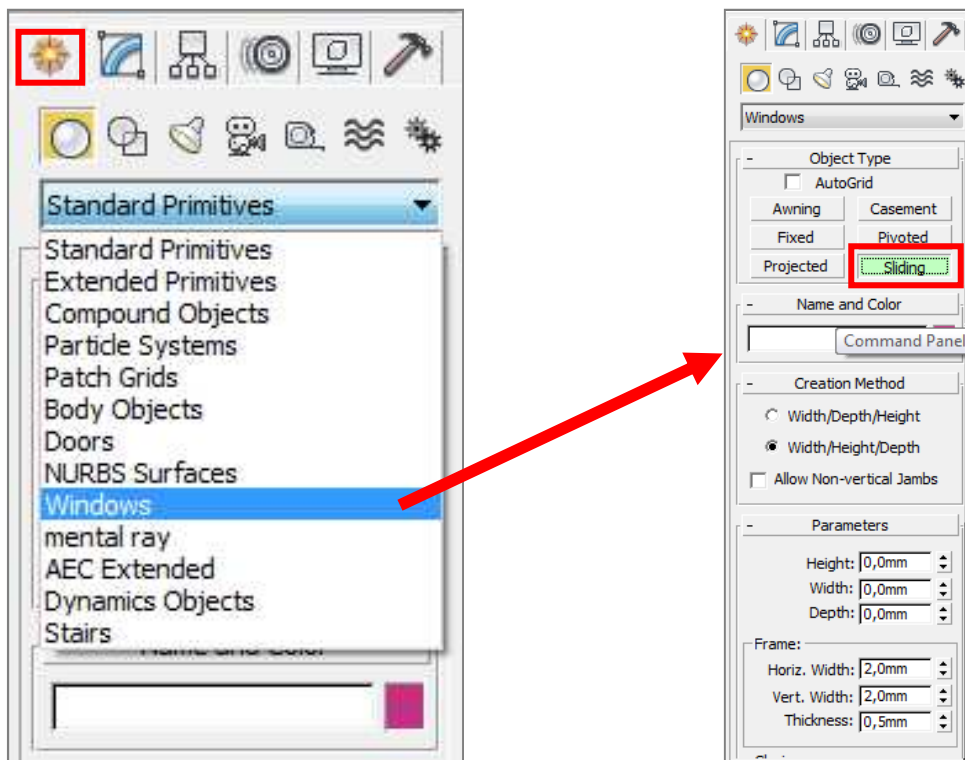
En este apartado, destacaremos algunos de los elementos utilizados en las escenas, por sus características especiales.

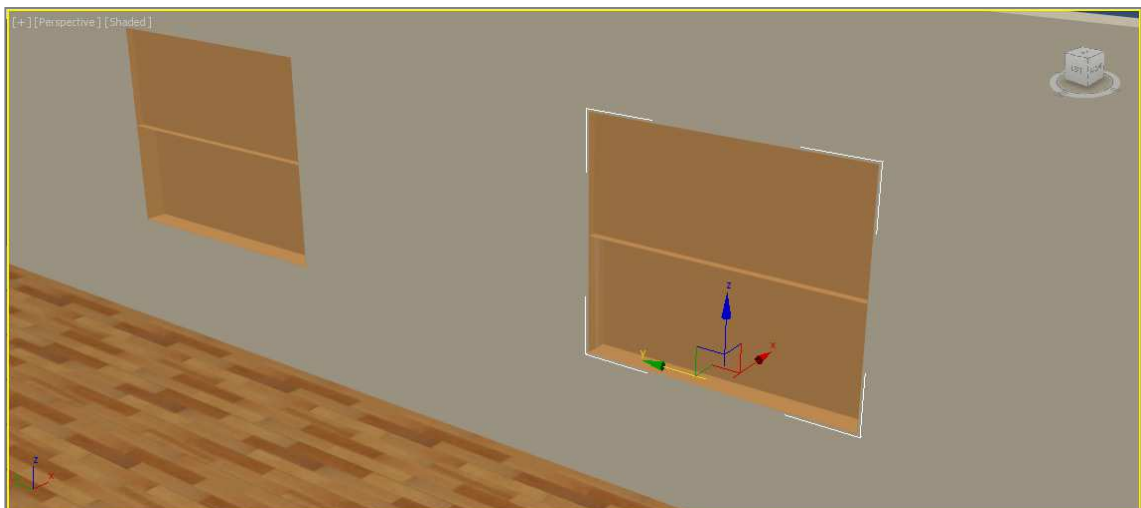
3.7.1-Puertas y Ventanas

Para completar la habitación, se crean algunos elementos como puertas y ventanas.

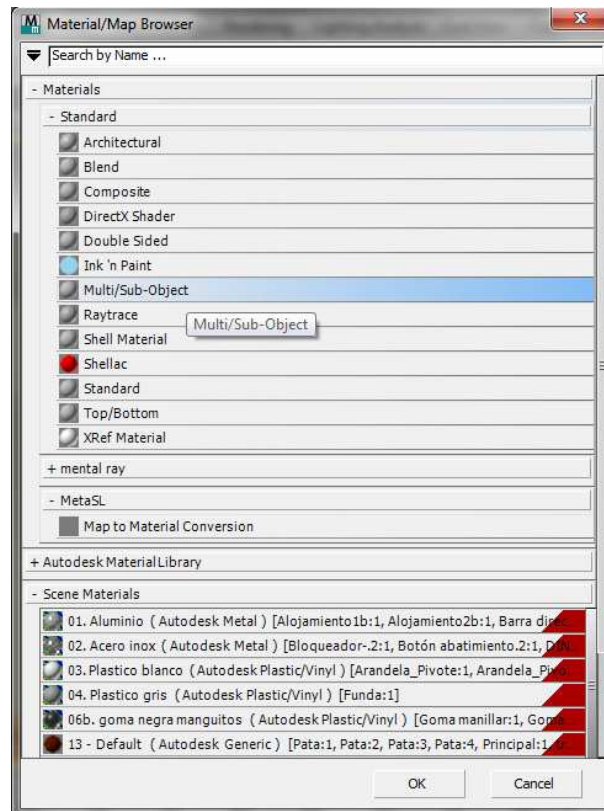
VENTANAS

En la sección *Geometry* del panel de comandos *Create*, seleccionamos la categoría *Windows*. En la persiana *Object Type*, se muestran los seis tipos de ventanas que podremos crear. En este caso crearemos dos ventanas del tipo *Sliding*.



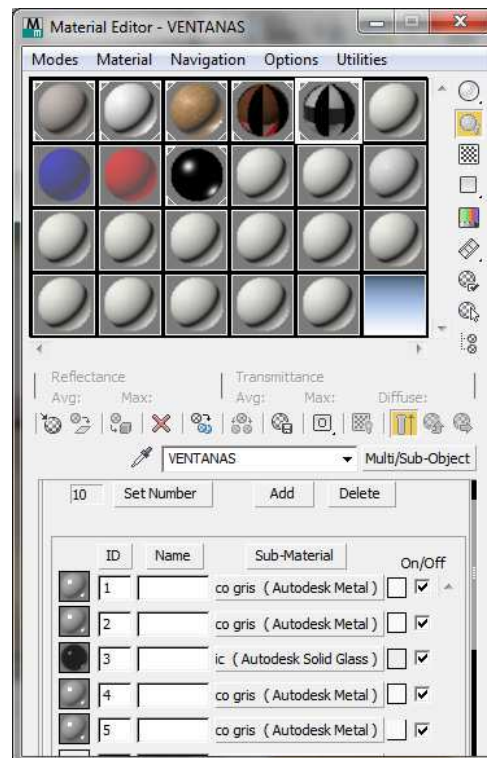


De manera predeterminada, 3ds Max asigna cinco ID diferentes de materiales a las ventanas, de este modo, cuando apliquemos un material *Multi/Sub-Object* (dentro de los materiales *Standard*), podremos configurar los materiales de los distintos elementos de una ventana.



ID de material	Componente de la ventana
1	Listones anteriores
2	Listones posteriores
3	Paneles / Cristal (50% de opacidad)
4	Cerco anterior
5	Cerco posterior

Una vez elegido el material de tipo *Multi/Sub-Object*, escogemos un material para cada uno de los componentes. Podremos elegir uno nuevo, o si queremos utilizar un material que ya exista en la escena, bastará con arrastrar el material que queramos de la muestra hasta el campo del submaterial correspondiente.

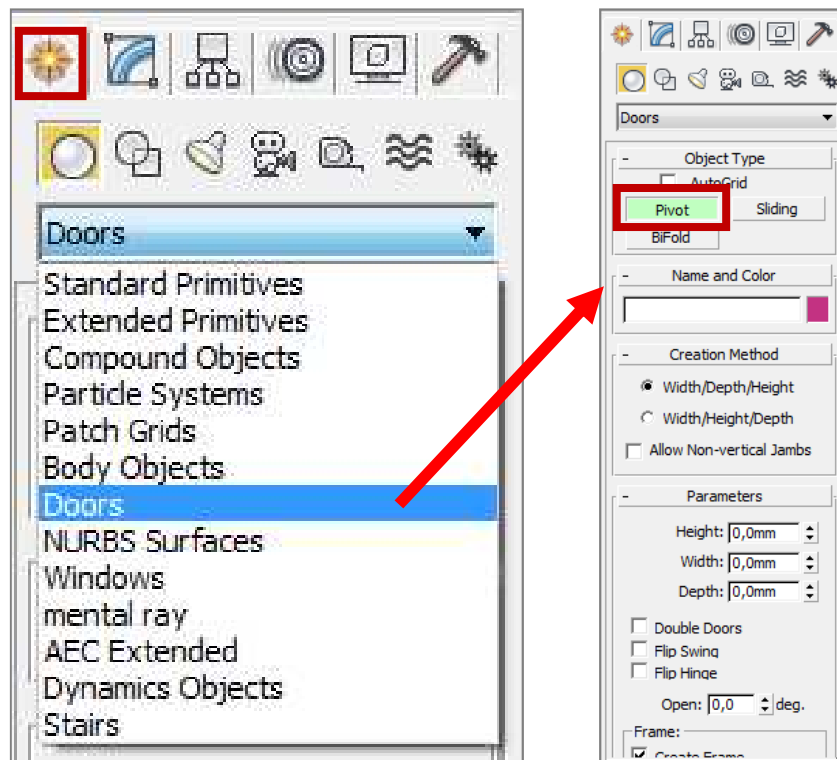


En este caso utilizaremos un aluminio pulido (*Autodesk Metal*, dentro de *Mental Ray*) para las partes metálicas, y un cristal transparente (*Glass*, dentro de *Autodesk Material Library*).



PUERTAS

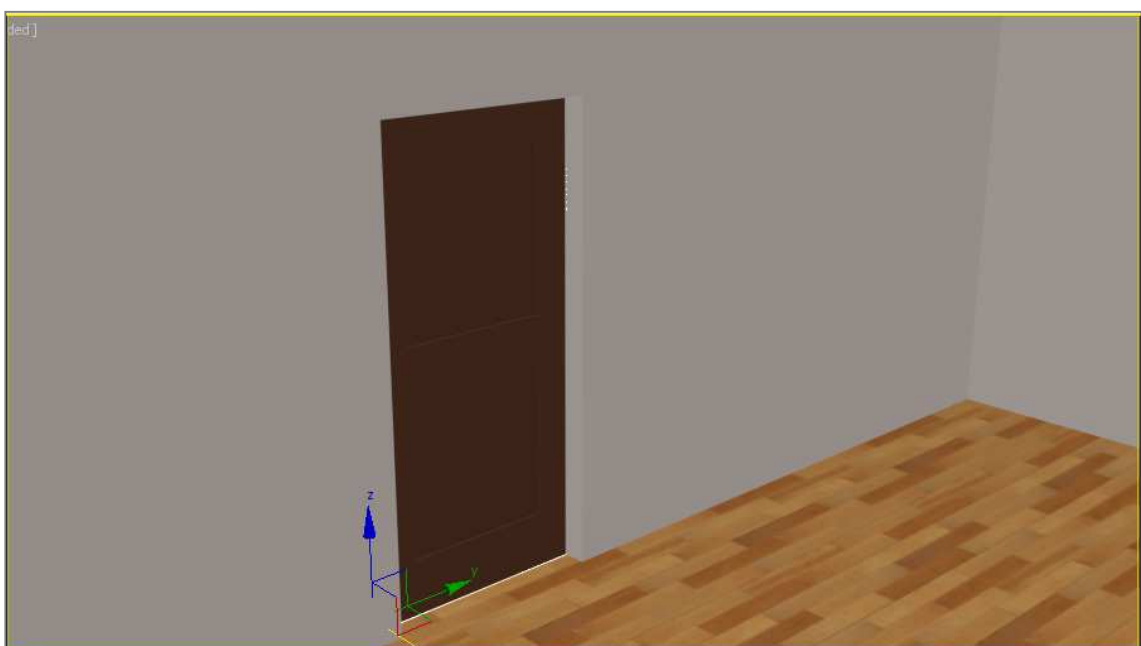
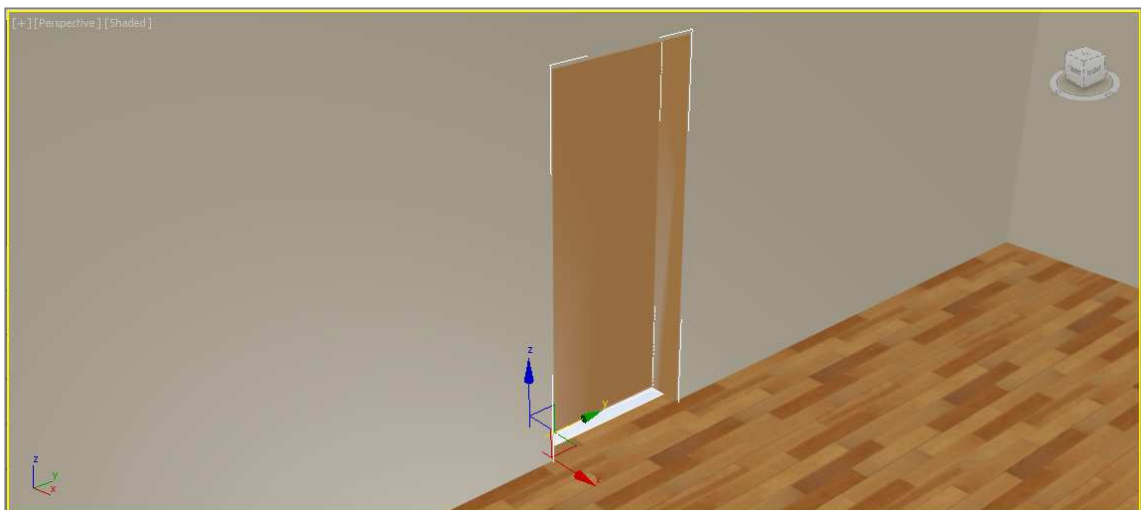
En la sección *Geometry* del panel de comandos *Create*, seleccionamos la categoría *Doors*. En la persiana *Object Type*, se muestran los seis tipos de ventanas que podremos crear. En este caso crearemos una puerta del tipo *Pivot*.



Al igual que en el caso de las ventanas, de manera predeterminada, 3ds Max asigna cinco ID diferentes de materiales a las puertas; de este modo, aplicando un material *Multi/Sub-Object*, podremos configurar los materiales de los distintos elementos. En el caso de las puertas, los identificadores son los siguientes:

ID de material	Componente de la puerta
1	Cuadro interior
2	Cuadro exterior
3	Panel
4	Cerco
5	Bordes

La selección se hará de la misma forma que en el caso de las ventanas.

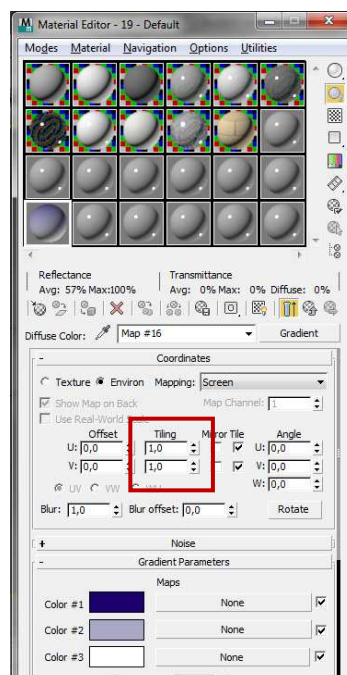


Finalmente, añadimos un pomo, modelado aparte.

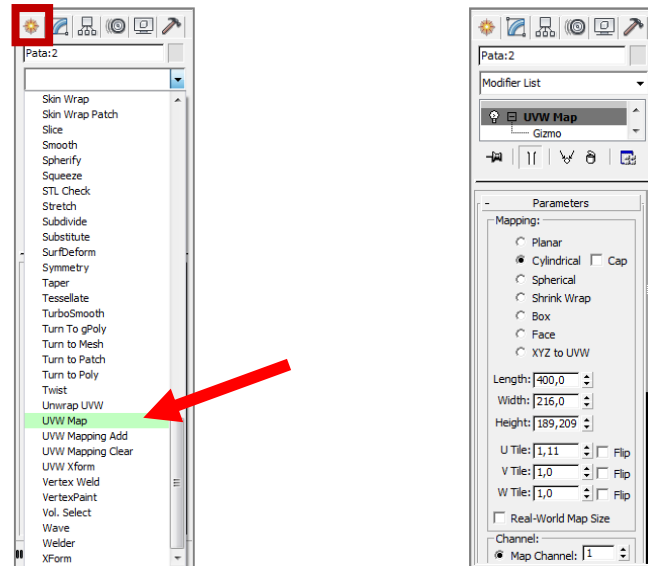


3.7.2-Mapeado de texturas

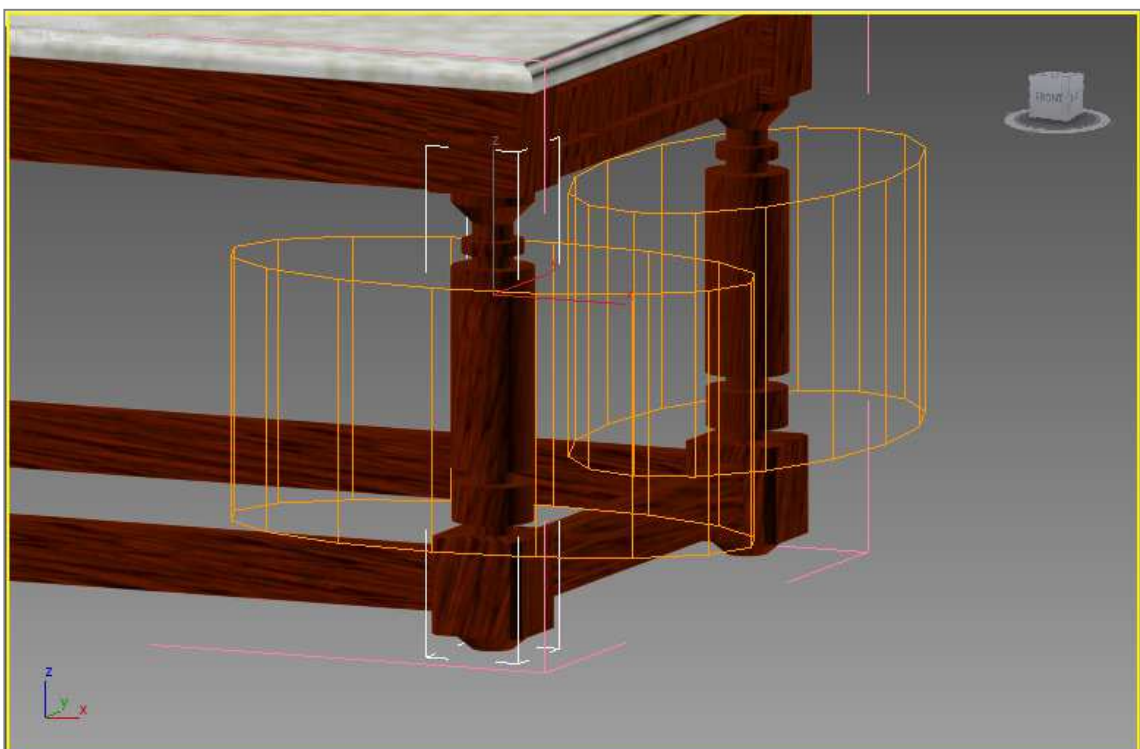
En general, para el ajuste de las texturas en el objeto, habrá que variar los valores del *Tiling* hasta obtener un resultado satisfactorio.



En ocasiones, la mejor forma de ajustar la textura en el objeto es por medio de un modificador de mapeado; *UVW Map*. Dentro del panel *Modify*, elegimos el modificador *UVW Map*, el cual ofrece diferentes tipos de mapeado a partir de primitivas geométricas; cilindro, caja, esfera...

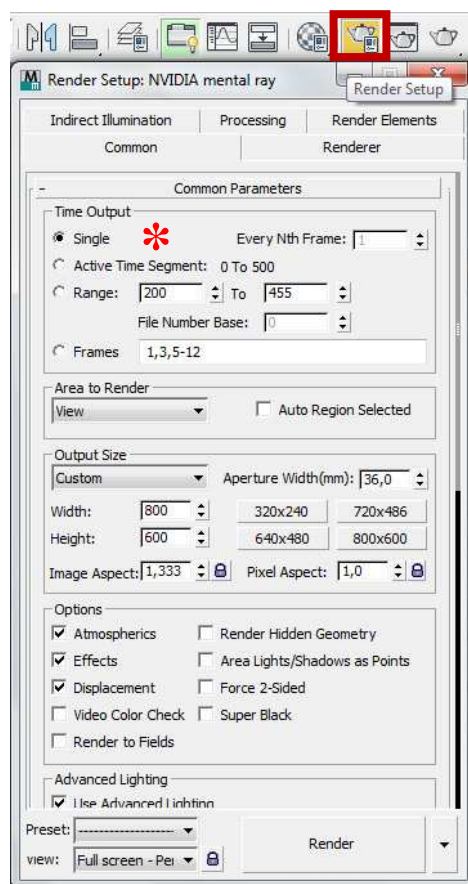


Variando las dimensiones de la primitiva en que se apoya el modificador de mapeado, ajustaremos el tamaño de la textura al objeto. Lo utilizaremos por ejemplo para ajustar el tamaño de la veta de la madera en la textura de la mesa o la puerta.



3.8- Configuración de Render

Para los render de la escena, se escogerán imágenes tanto de conjunto como de detalle. Para ello, hay que valorar la información que se quiere dar a cerca del producto; seleccionando las imágenes que mejor plasmen las características que se quieren destacar.



Accediendo al panel de configuración de renderizado (Render Setup), se ajustan los parámetros para la exportación de las imágenes.

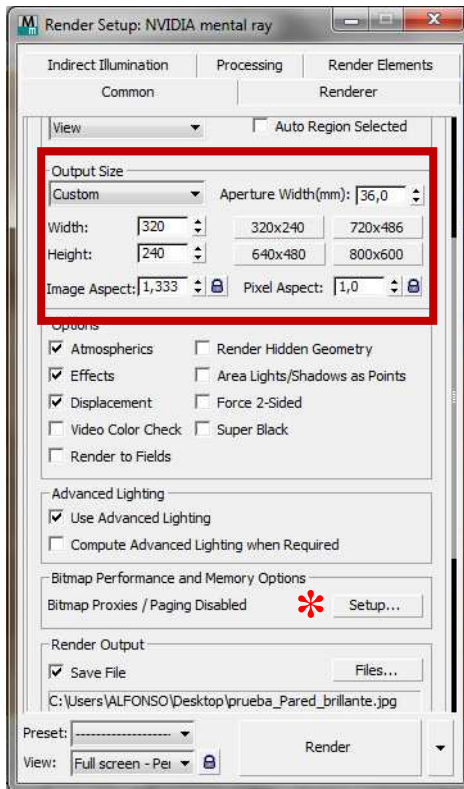
Accederemos a dicho panel, en el icono correspondiente en la Barra de Herramientas:



En el caso de las imágenes, seleccionaremos *Single*, en el área *Time Output*, de la sección *Common Parameters* *. Esto quiere decir, que la imagen que se renderizará, corresponderá al área que vemos en el visor que tenemos activo.

En el caso de un archivo en el que hay una animación, la imagen a renderizar; será el área que vemos en el visor activo, en el fotograma de la animación en el que nos encontremos.

3.9-Exportación de Imágenes Renderizadas



Accederemos al panel de configuración de renderizado (Render Setup), en el icono correspondiente en la Barra de Herramientas:



En el área Output Size, determinaremos el tamaño de las imágenes que queremos exportar y su resolución.

Más abajo, en el área *Render Output*, escogeremos el directorio para guardar los archivos exportados, haciendo clic en el botón Files *. Escogeremos un formato de imagen, normalmente JPEG.

3.9.1-Imágenes Generales

Planos generales o planos medios del patinete, desde diferentes ángulos y en distintas posiciones.









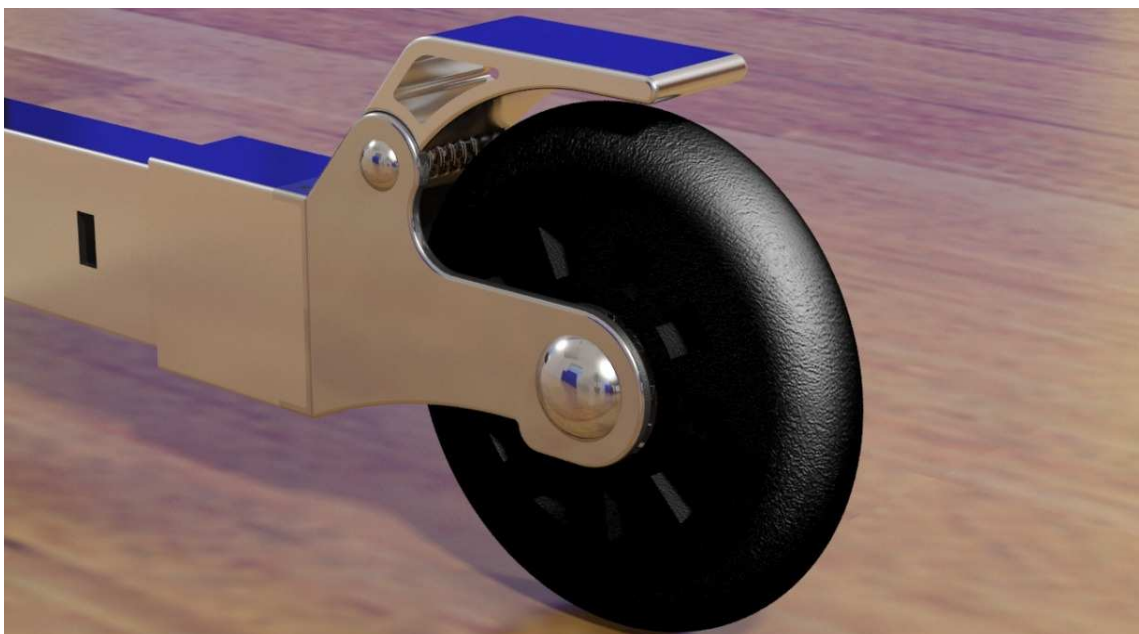




3.9.2-Imágenes de Detalle

Primeros planos del producto. Imágenes de detalle que muestran características del producto a destacar.

Primer plano de la rueda trasera.



Manillar plegado para ocupar menos espacio.



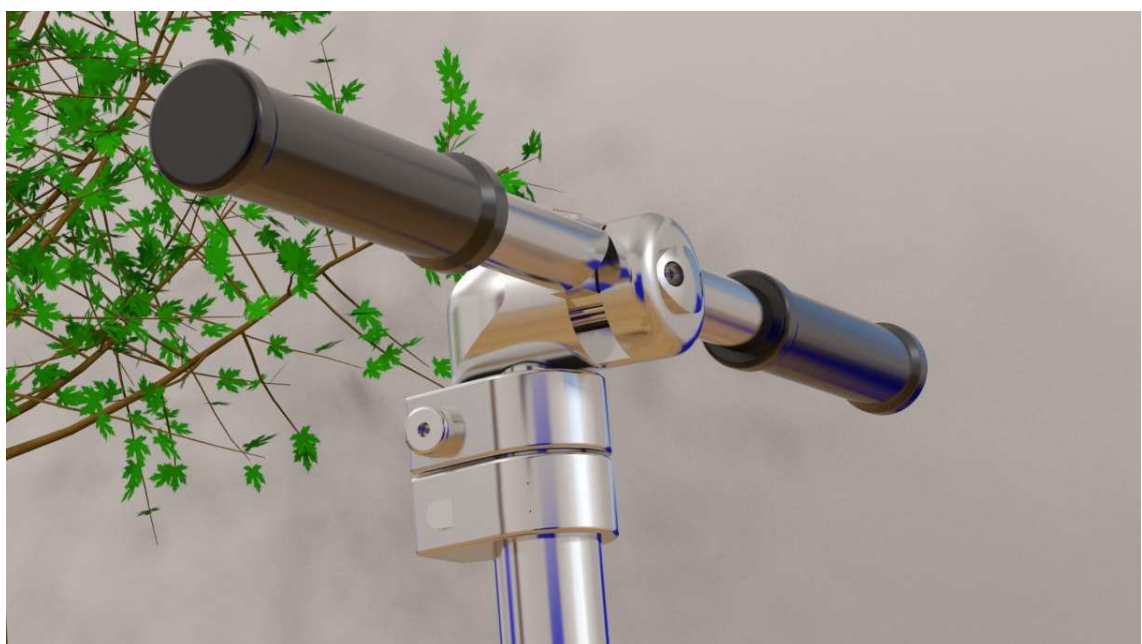
Detalle de reflexiones; patinete plegado sobre mesa.



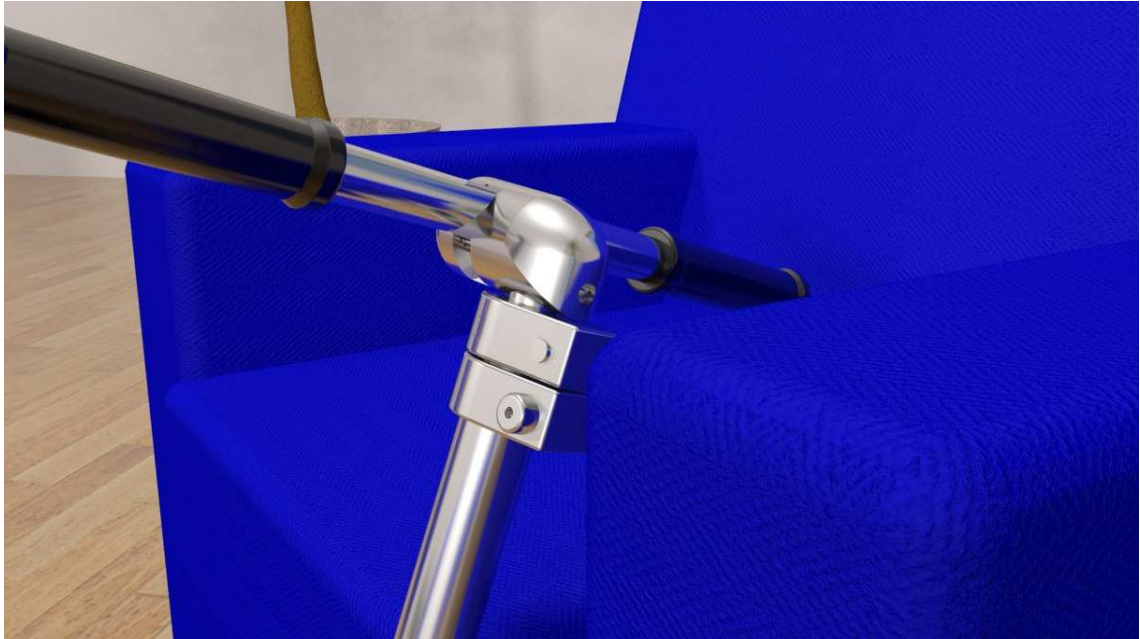
Primer plano de la parte delantera del patinete plegado.



Primer plano del manillar.



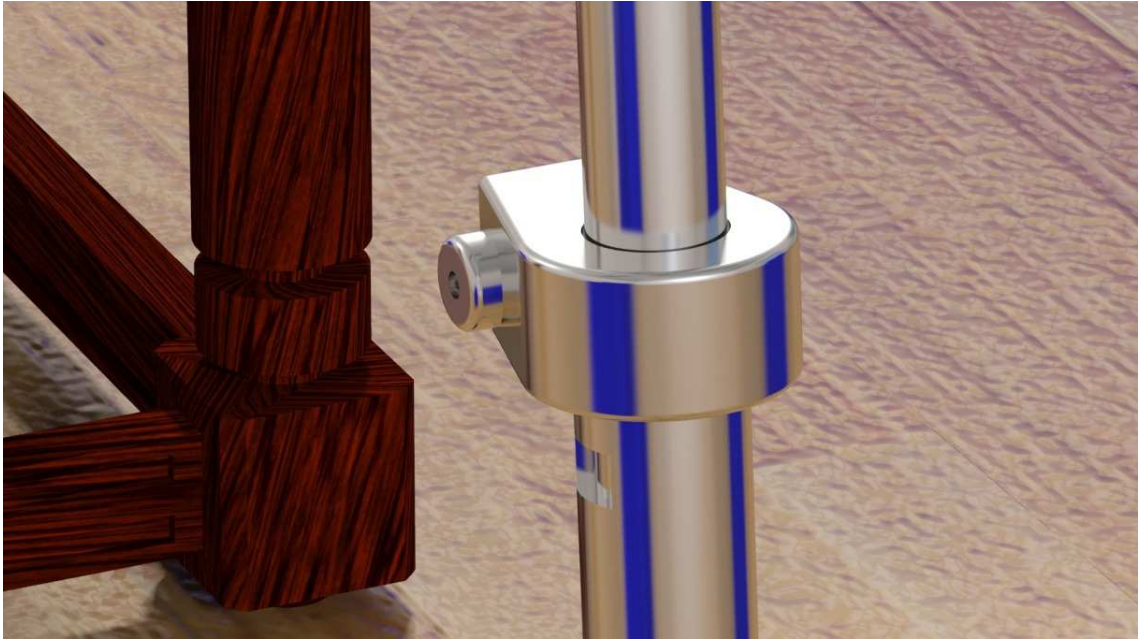
Columna de dirección plegada y manillar abierto. Detalle de reflexión y textura.



Plano medio de la plataforma extendida.



Primer plano de uno de los bloqueos de la dirección.

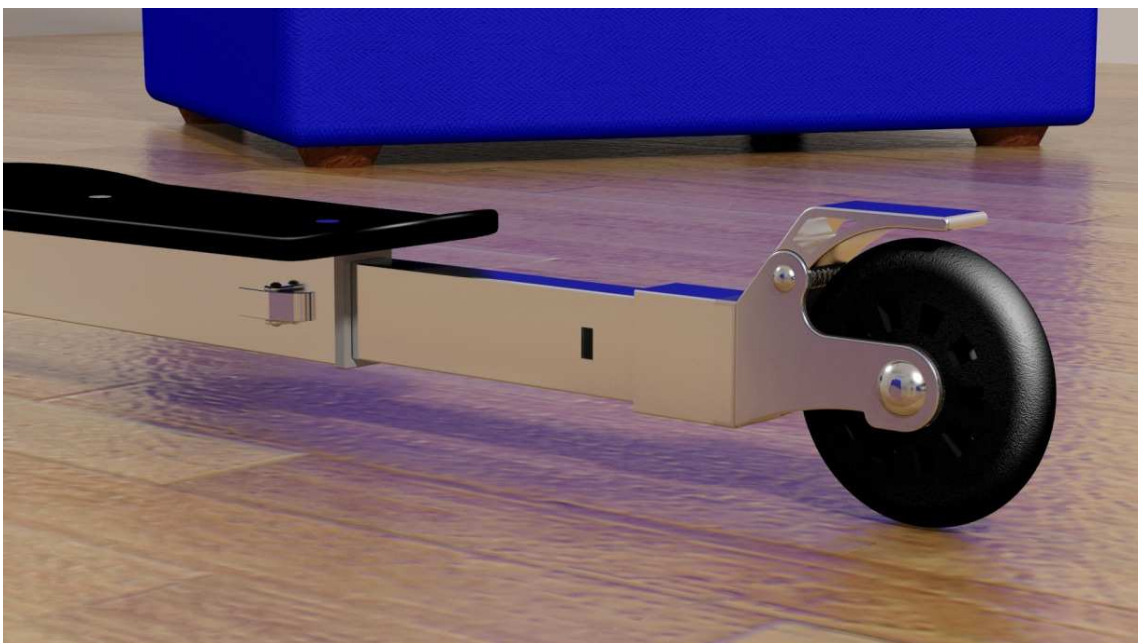


Patinete entre sofá y pared, mostrando lo fácil que es de guardar en cualquier sitio.

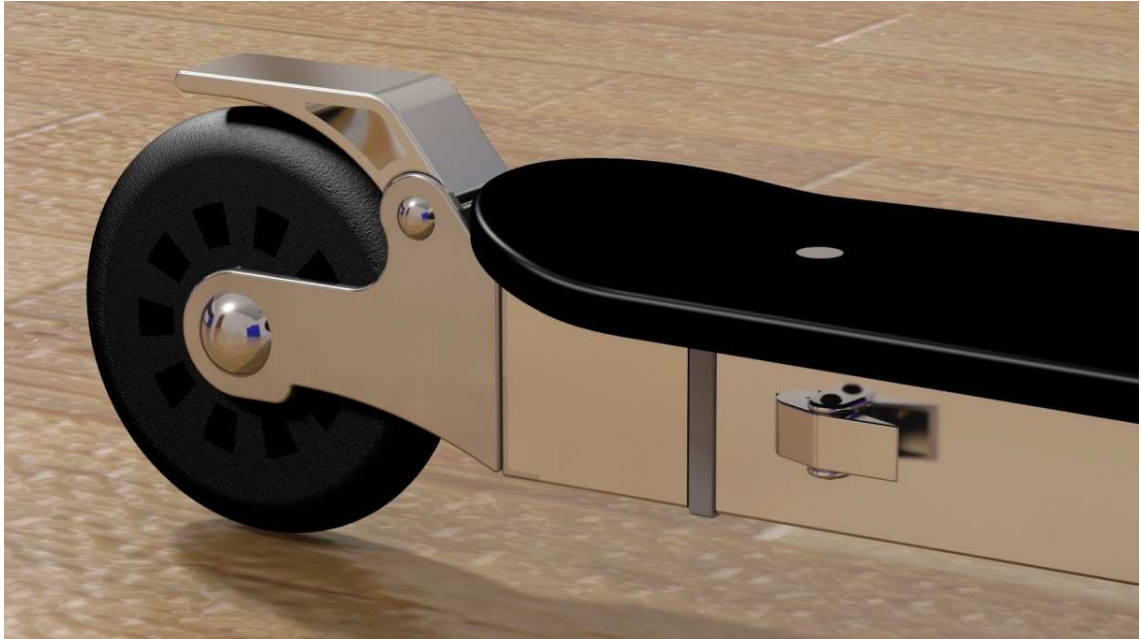




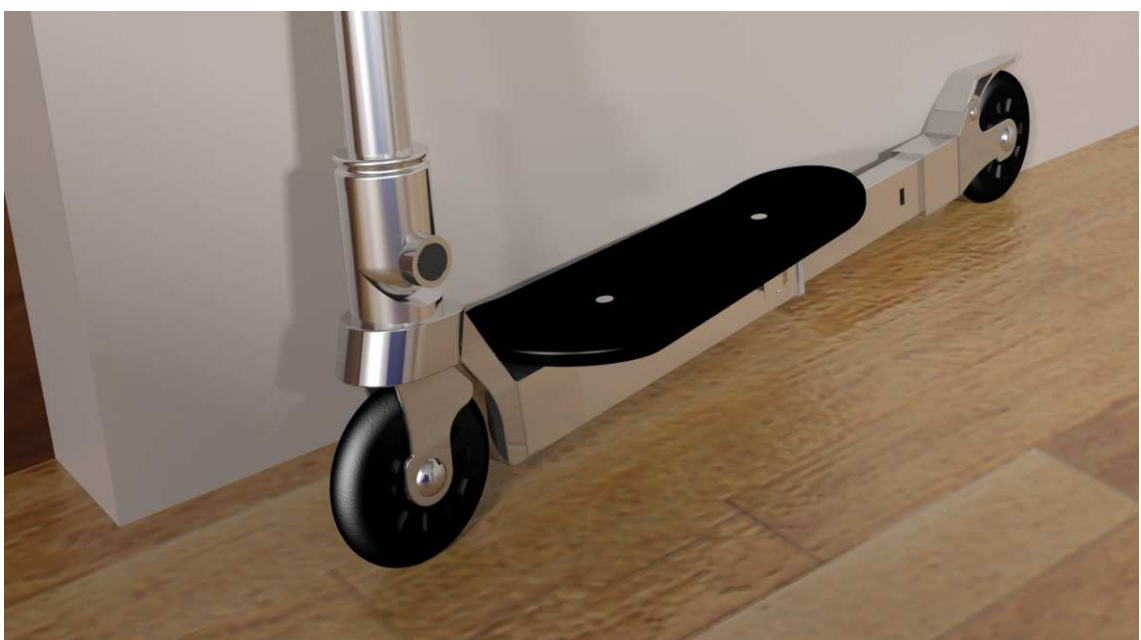
Vista trasera de la estructura en su posición extendida.



Detalle de la estructura en su posición más reducida.



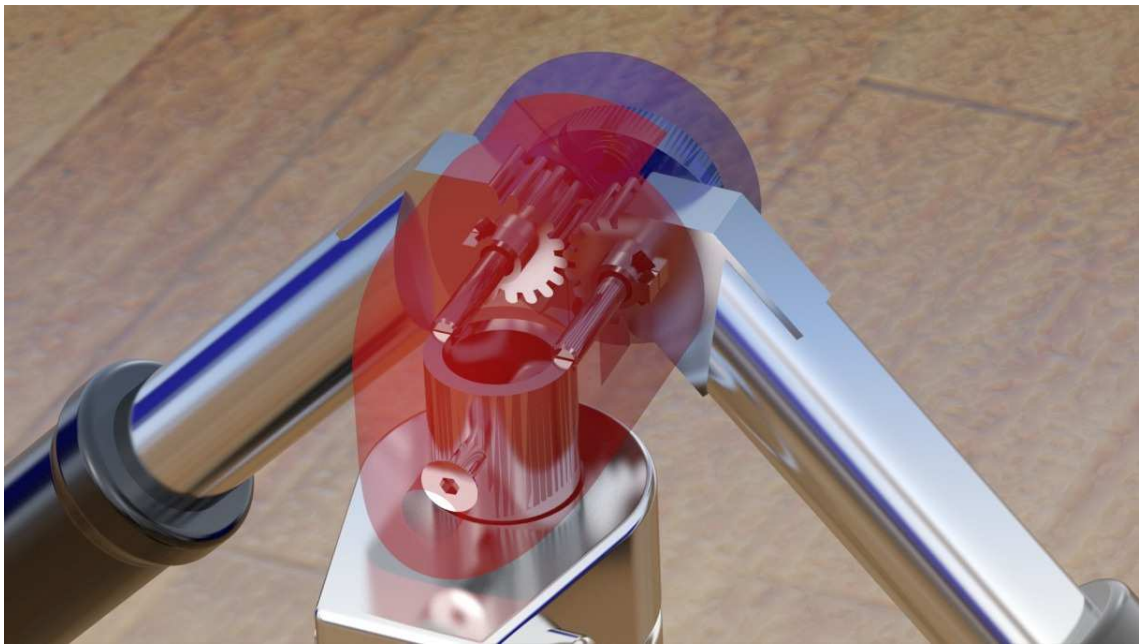
Plano medio de la base del patinete, apoyado en la pared.



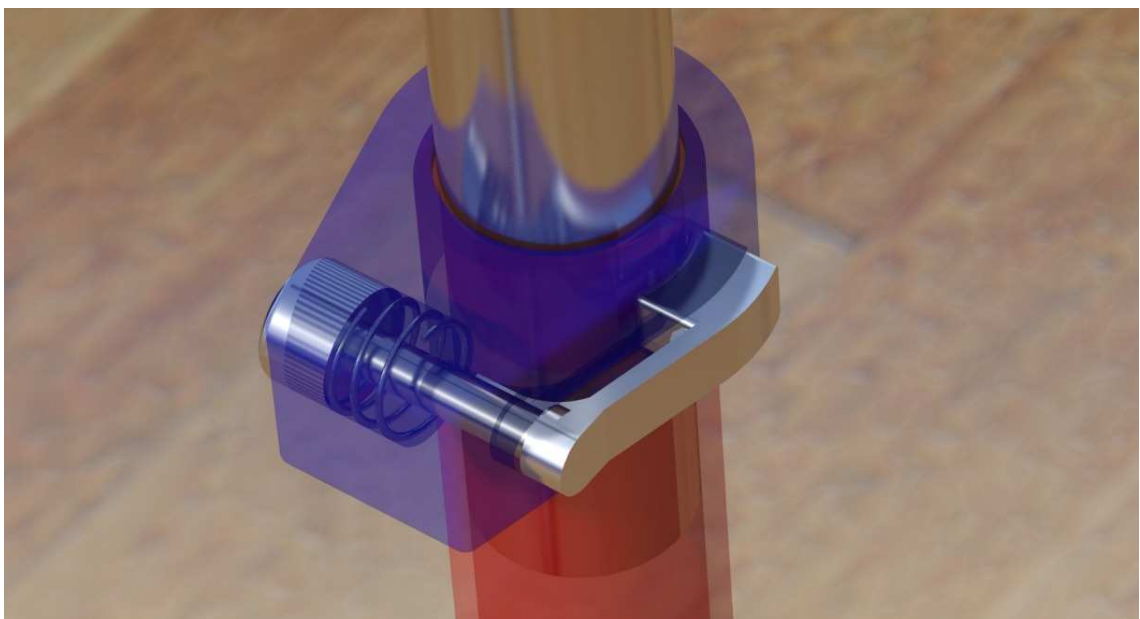
3.9.3-Imágenes de Funcionamiento

Imágenes explicativas. En ellas se pueden ver detalles de las piezas o mecanismos, que de otra manera no sería posible. Esto se consigue haciendo que algunas piezas sean transparentes, permitiendo ver lo que ocurre en su interior.

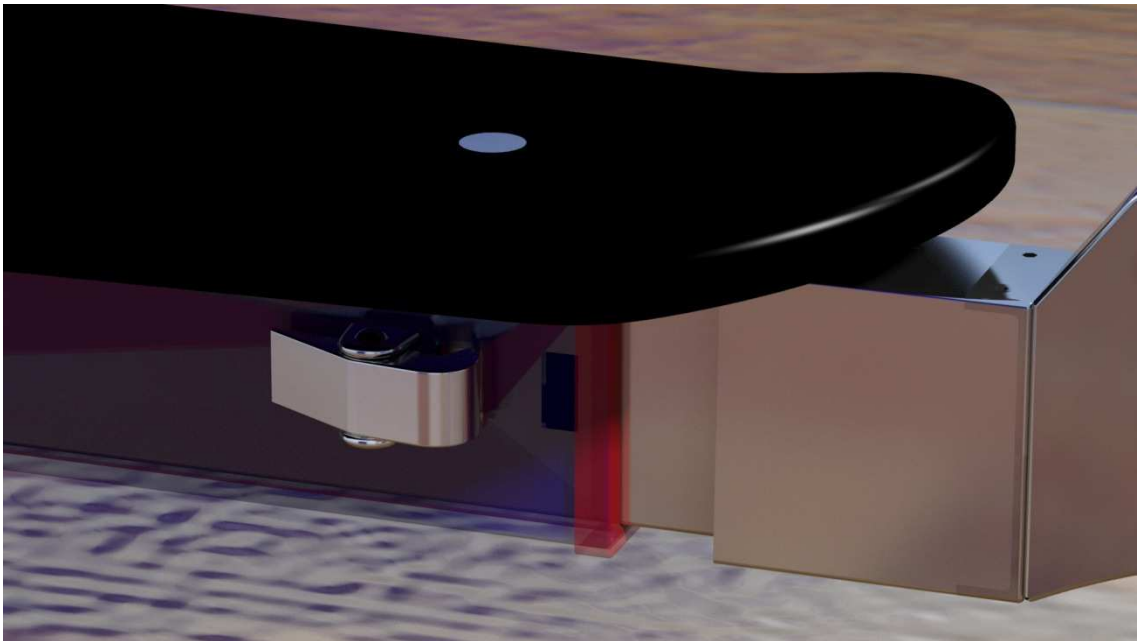
Detalle del mecanismo interior del manillar.



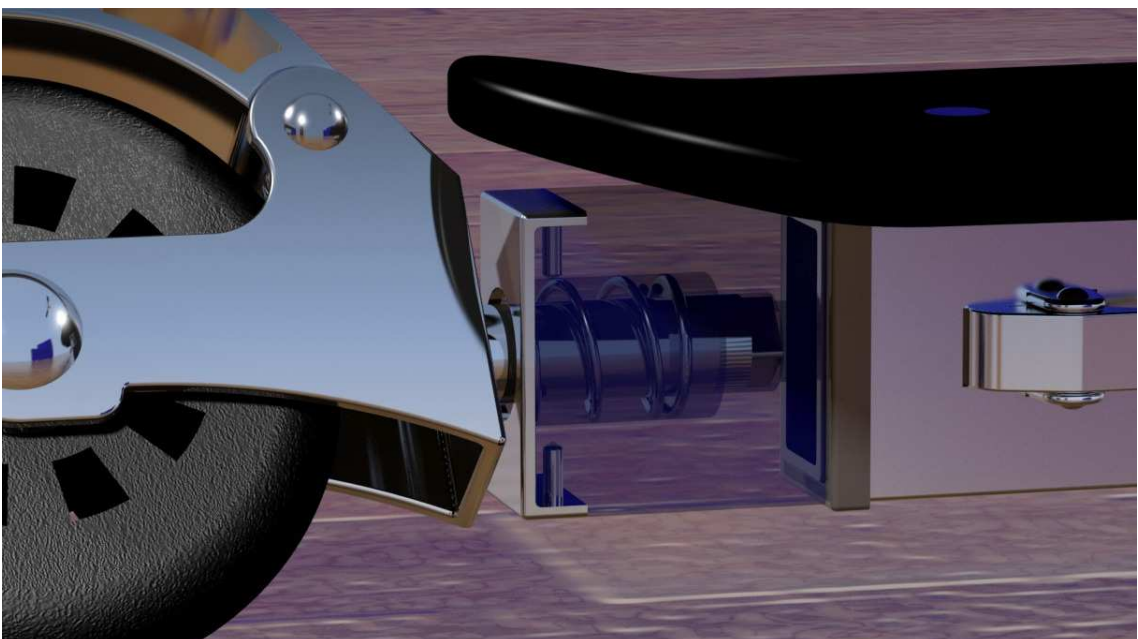
Detalle del funcionamiento de uno de los bloqueos de la dirección.



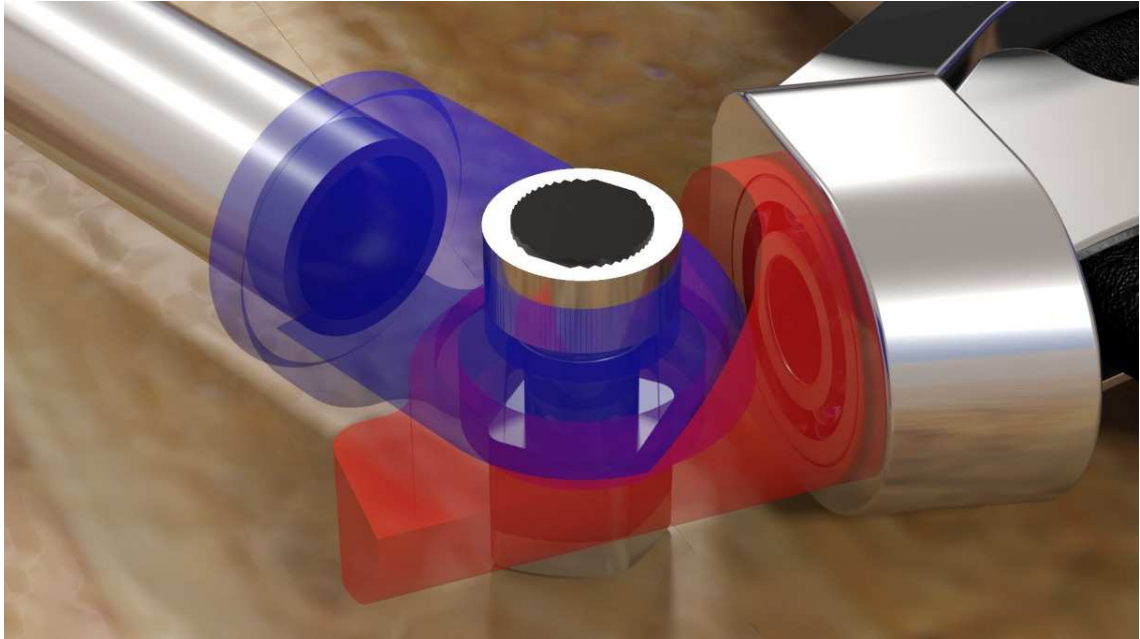
Detalle del tramo más estrecho de la estructura de la plataforma, deslizándose por el interior del tramo de sección mayor.



Detalle del giro de la rueda trasera, en el interior de la estructura de la base.



Detalle del funcionamiento del botón de abatimiento de la dirección.

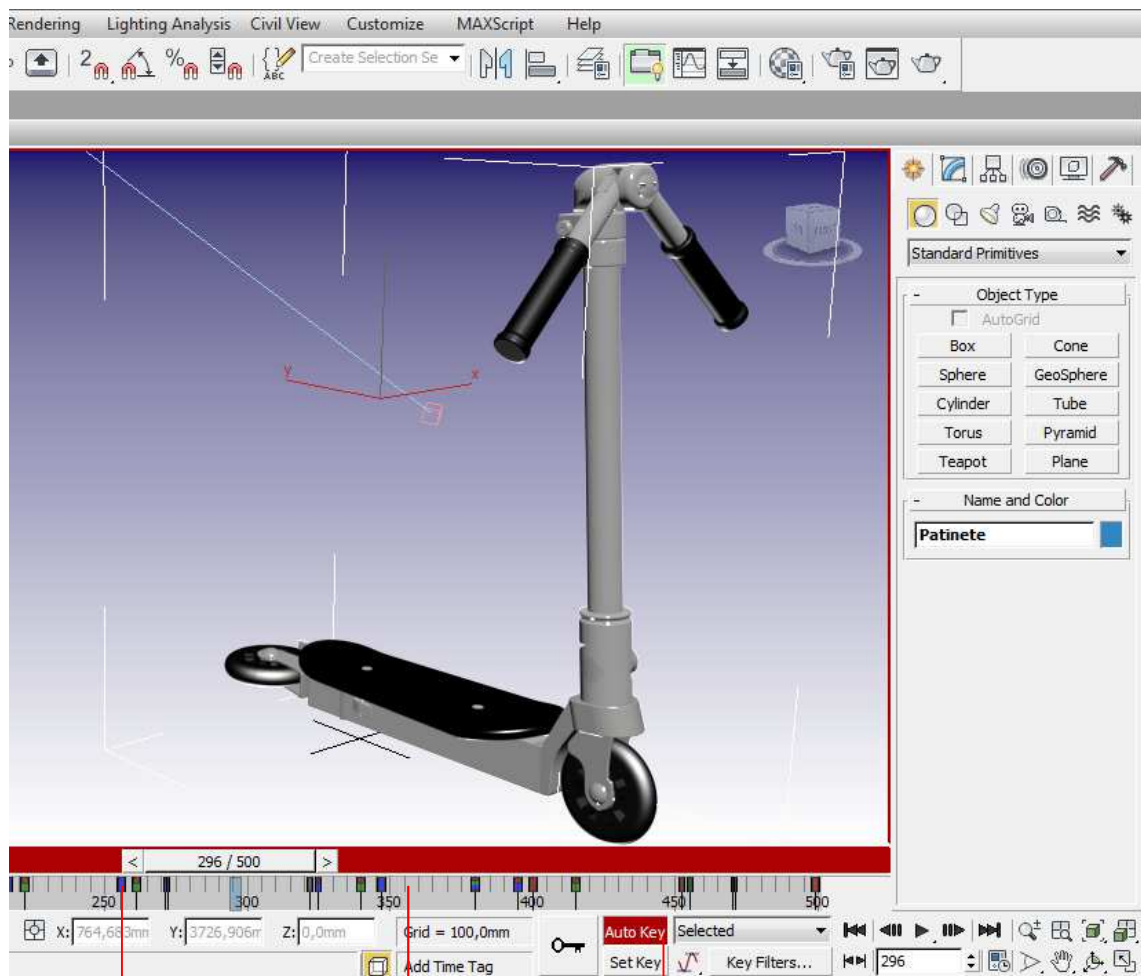




Para mostrar determinadas características de un producto, más aún cuando se trata de un objeto mecánico, la vía más adecuada, es a través de una animación. De esta forma, vemos cómo interactúan las distintas piezas, así como su montaje. Con los videos veremos los diferentes mecanismos del patinete en funcionamiento; lo cual da una información importantísima acerca del producto.

Las animaciones se realizarán en 3d Studio, activando el botón *Auto Key*. De esta forma, iremos modificando la escena, creando “fotogramas clave”. Con esta herramienta, tenemos el control sobre los elementos que configuran la escena, pudiendo cambiar su posición o rotación, en el momento que deseemos.

En la siguiente imagen se muestran los elementos a tener en cuenta en el proceso de animación. Para comenzar a animar cualquier elemento, activaremos el botón *Auto Key*, en la parte inferior de la interfaz de *3ds Max*. Al activarlo el visor activo se enmarca en rojo. En este momento podemos comenzar con la animación; lo cual haremos moviéndonos a lo largo de la línea de tiempo. Cada vez que modifiquemos algún elemento de la escena, se generará un fotograma clave; lo que permitirá el ajuste de tiempos y más adelante veremos cómo permitirá también modificar de forma precisa, propiedades como la posición, rotación o escala de dichos elementos.



Fotogramas Clave Línea de tiempo Botón *Auto Key*

4.1-Configuración

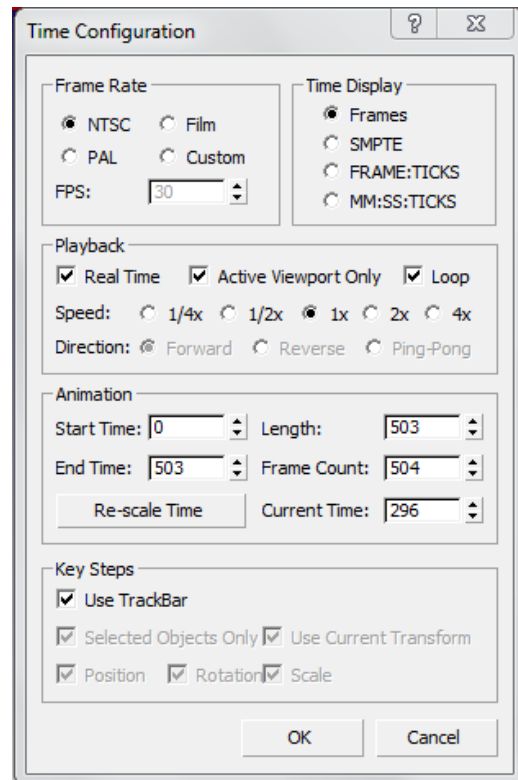
En la parte inferior de la interfaz de 3d Studio*, nos encontramos con los controles de la animación. Bajo éstos, una casilla en la que aparece un número; correspondiente al fotograma en el que nos encontramos.



A su derecha, aparece el icono de Configuración de Tiempo (*Time Configuration*).



Haciendo clic en dicho icono, accedemos al cuadro; que se muestra a la derecha.



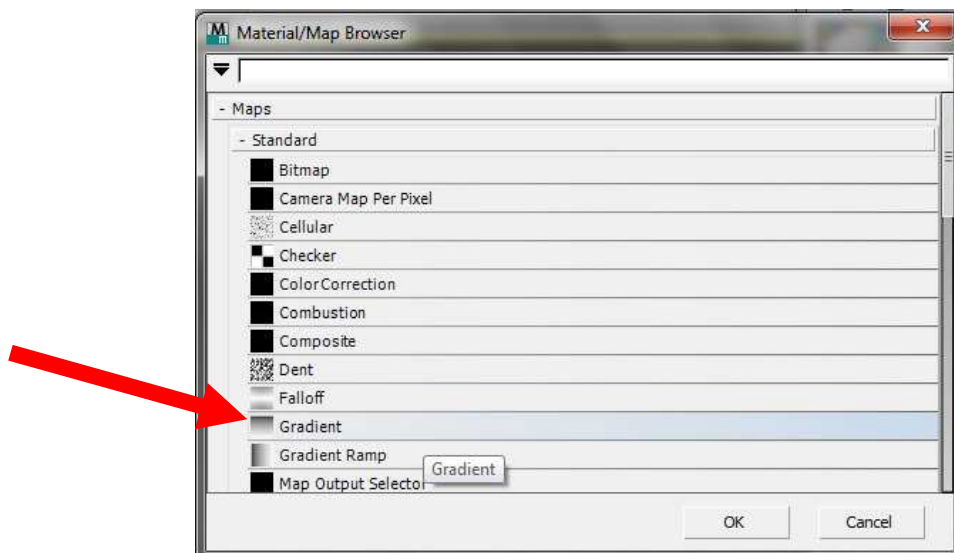
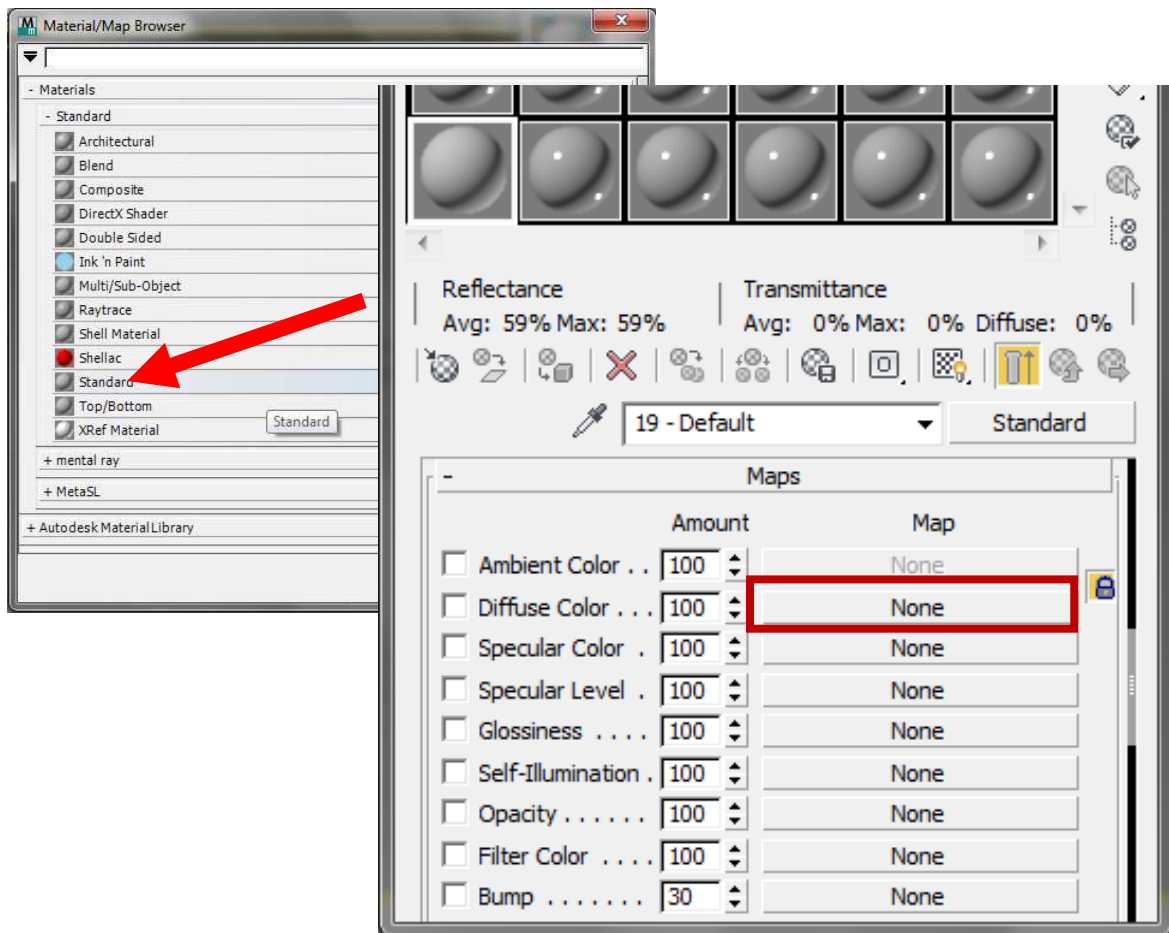
En él, podemos establecer, entre otros parámetros, la velocidad de reproducción de la animación y el tiempo que durará ésta.

4.2-Entorno de la animación



Para las animaciones elegiremos un entorno distinto. En este caso, optaremos por una escena diáfana; en la que sólo aparecerá el patinete, y como entorno un degradado, que crearemos con el editor de materiales. El patinete se encontrará flotando en la escena, por lo que la cámara podrá moverse libremente para grabar desde cualquier ángulo que se quiera. Crearemos un cielo; un degradado de azul oscuro a blanco.

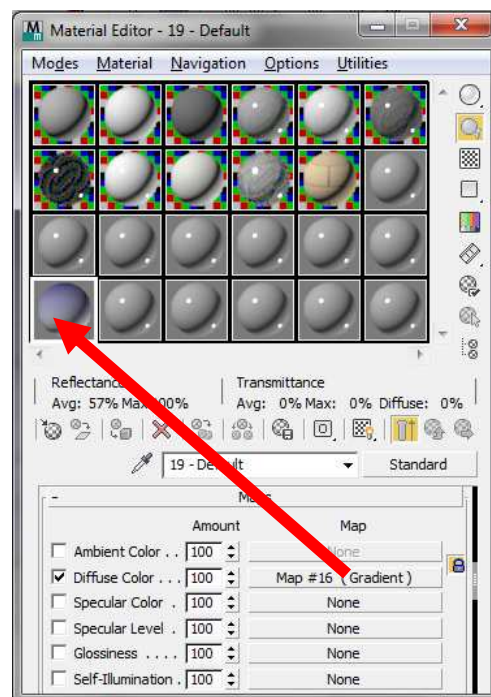
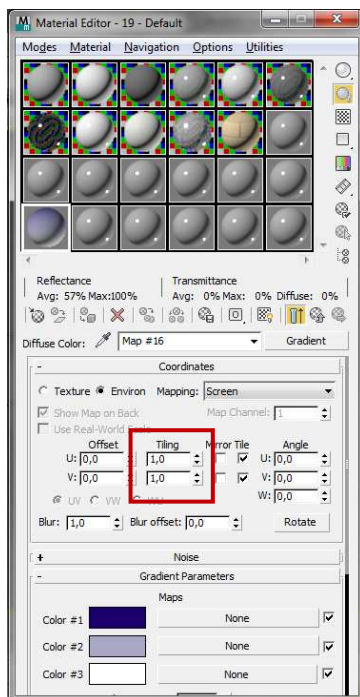
4.2.1-Creación del degradado

En el editor de materiales de *3ds Max*, escogeremos un material estándar. En la sección Maps, como mapa de color difuso seleccionamos Gradient; dentro de *Maps > Standard*.



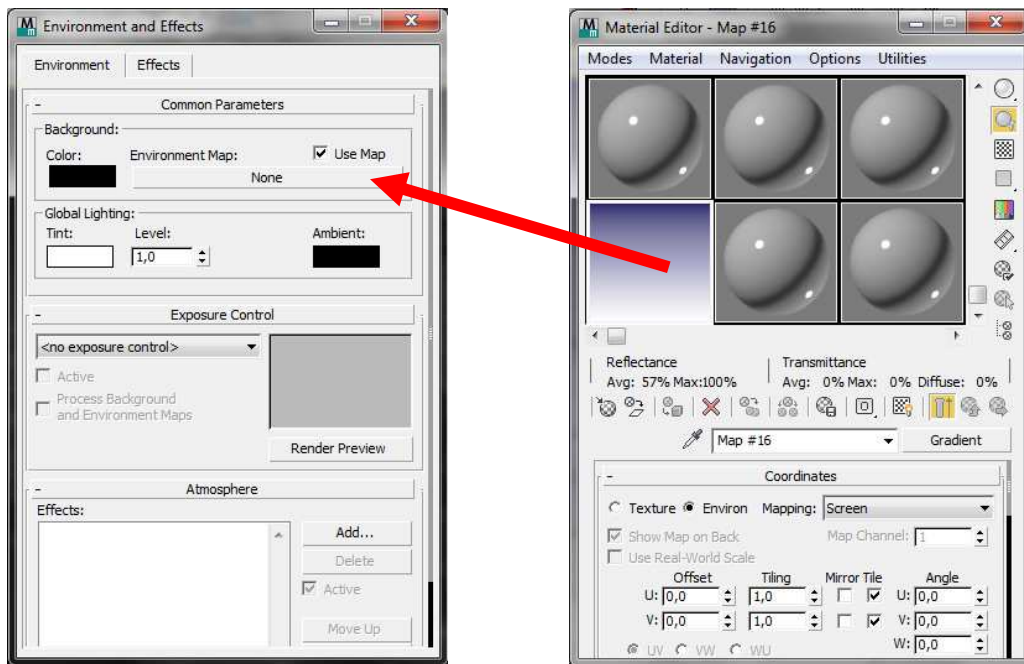
En la sección Coordinates, seleccionamos Environment y Screen como Mapping. Pondremos un valor de 1 para el Tiling de U y V. Y en Los parámetros de degradado escogeremos Un **azul oscuro, azul medio/claro y blanco**. El resto de opciones las dejaremos por defecto.

A continuación subimos un nivel; volviendo al material estándar con el botón *Go to parent* . Arrastraremos el mapa de degradado creado, desde el botón de Color difuso hasta la miniatura del material (seleccionaremos *Copy* en el menú emergente). Finalmente, tendremos lo siguiente; y sólo lo usaremos como material del entorno. 

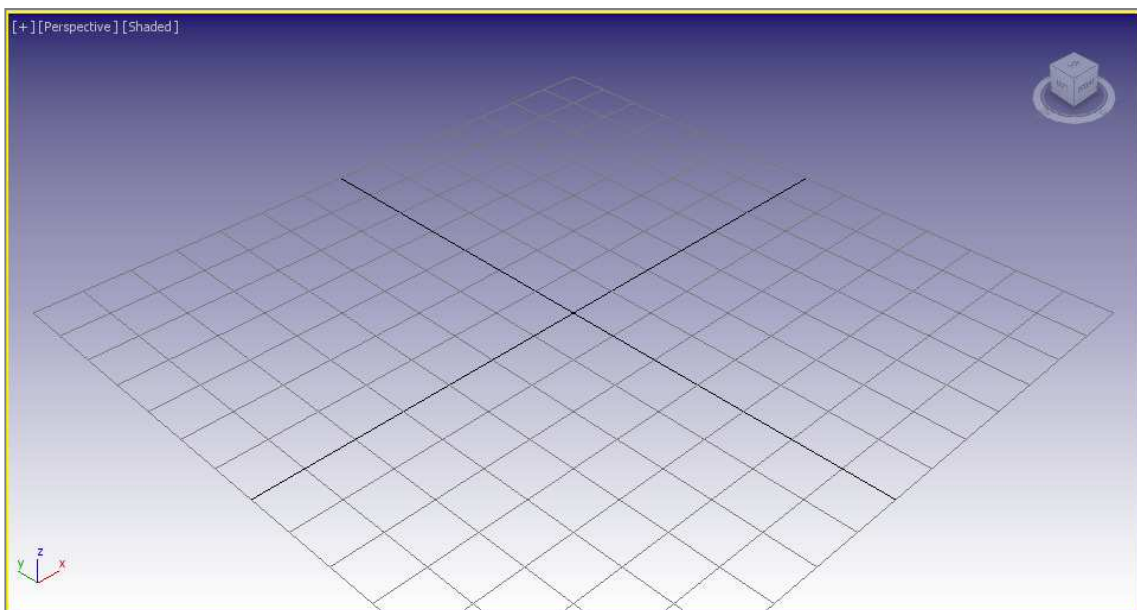


4.2.2-Degradado como entorno

Para poner el degradado que hemos creado, como entorno de la escena; tendremos que hacer lo siguiente. La forma más sencilla de hacerlo, será abriendo el menú de Entorno (*Rendering > Environment... o pulsando la tecla 8*). Con el editor de materiales todavía abierto, bastará con arrastrar nuestro degradado hasta el mapa de entorno, como se muestra a continuación.



Para ver en la escena el entorno que hemos creado, tendremos que seleccionarlo como fondo del visor en el que nos encontramos; a través del menú *Views*: *Views > Viewport Background > Environment Background*. La escena quedará de la siguiente forma:

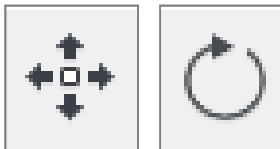


4.3-Definición de movimientos de articulaciones

Una vez activado el Auto Key, nos disponemos a animar los objetos (el visor activo, quedará enmarcado en color rojo).

Como las piezas guardan relación entre ellas, gracias a las jerarquías creadas; y los desplazamientos y giros están restringidos, los movimientos de la animación serán coherentes.

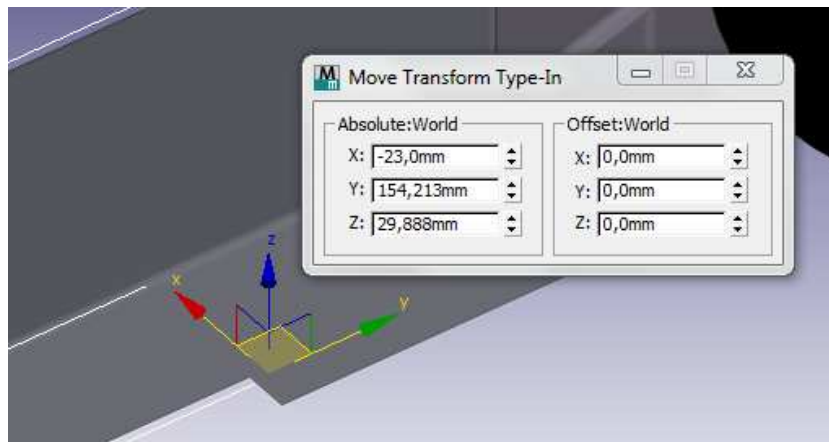
Para realizar los diferentes movimientos, utilizaremos el Sistema de Coordenadas *Local*, que se refiere al punto pivote de la pieza que estemos manipulando. Es posible utilizar en alguna ocasión, el Sistema de Coordenadas *Parent*, donde los movimientos se realizan respecto del elemento “padre” del que estamos manipulando. Es decir, el elemento al que éste está vinculado (en caso de no estar vinculado a ningún otro objeto, se comportaría como el Sistema de Coordenadas *World*, o Universal).



Utilizaremos Select and Move y Select and Rotate; de la barra de herramientas, para realizar los desplazamientos y giros, respectivamente.

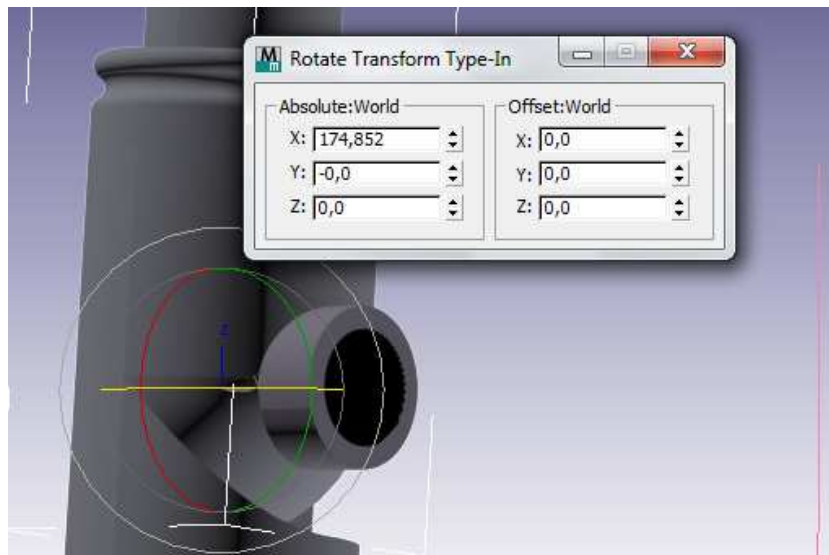
Haciendo clic en el icono Select and Move y a continuación en un objeto, podremos moverlo manualmente según nos lo permitan las restricciones y jerarquías.

Si por el contrario, utilizamos el botón derecho del ratón, accederemos a un cuadro de diálogo, donde podremos introducir datos por teclado. De esta forma, introduciremos valores de desplazamiento, de forma precisa.



Haciendo clic en el icono Select and Rotate y a continuación en un objeto, podremos girarlo manualmente según nos lo permitan las restricciones y jerarquías.

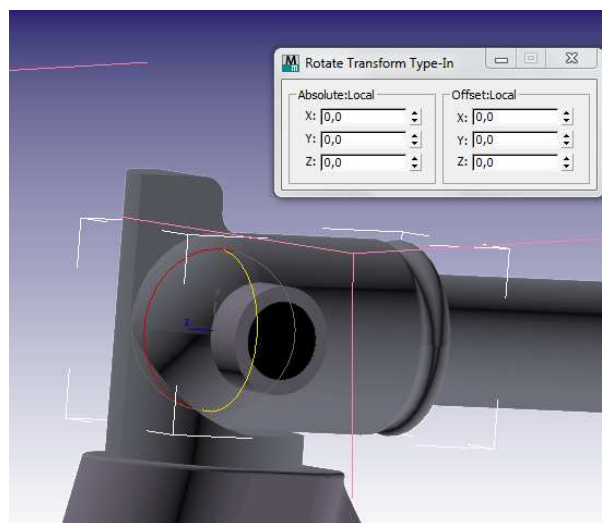
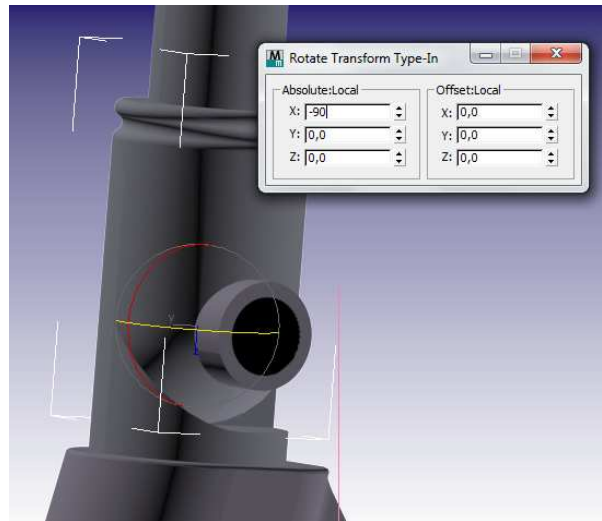
Si por el contrario, utilizamos el botón derecho del ratón, accederemos a un cuadro de diálogo, donde podremos introducir datos por teclado. De esta forma, introduciremos ángulos de giro, de forma precisa.



En dichos cuadros, se podrán introducir coordenadas absolutas (*Absolute*), o también podremos incrementar o reducir en un valor determinado las ya existentes (*Offset*).

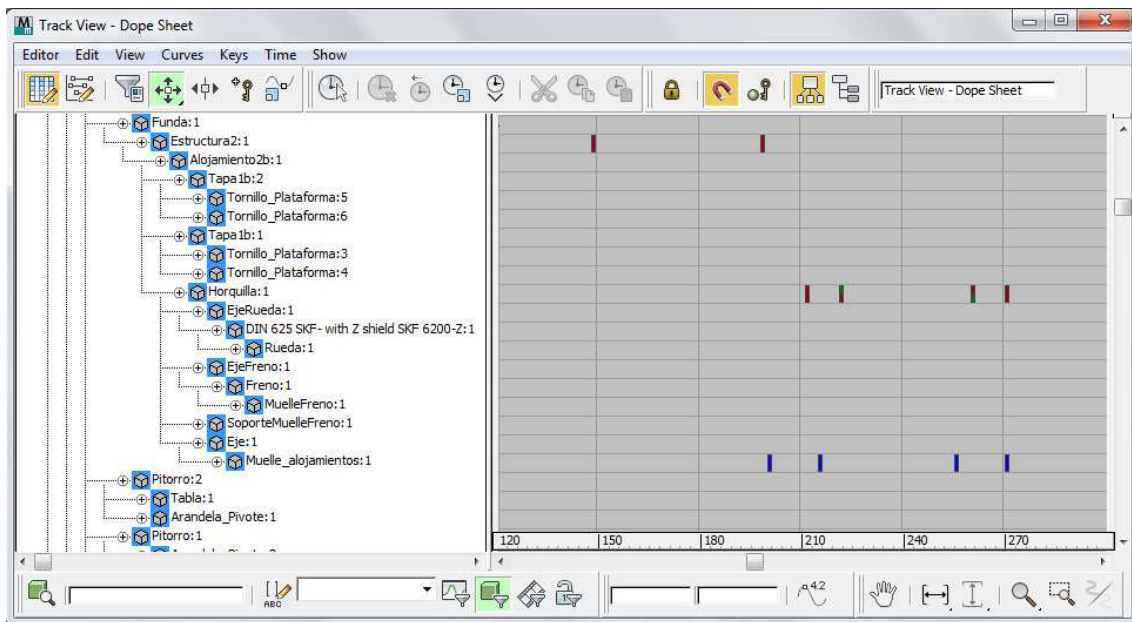
Hay que tener en cuenta, en estos casos, que si utilizamos el Sistema de Coordenadas de Referencia Local, los valores que introduzcamos serán siempre relativos; con respecto a la posición en la que se encuentre el objeto en ese momento.

Así por ejemplo, si seleccionamos la pieza de abatimiento de la dirección e introducimos un valor de 90° en el sentido correcto del eje X del Sistema de Coordenadas Local, conseguiremos que la columna de dirección quede en su posición plegada.

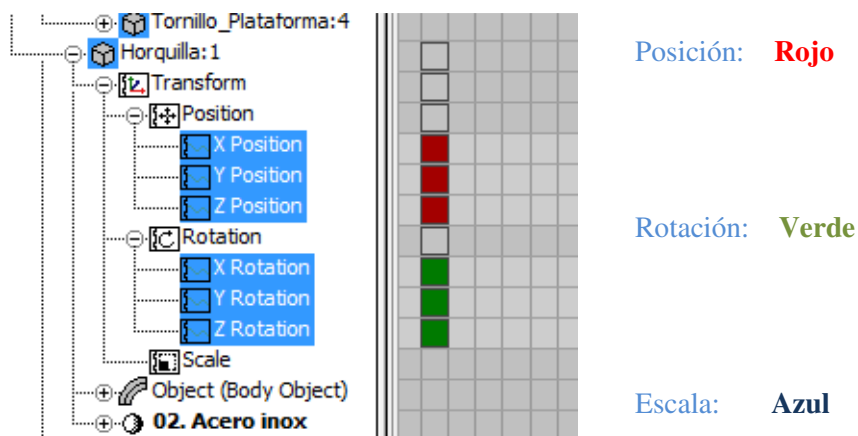


4.4-Uso de Dope Sheet

Para manipular los fotogramas clave más fácilmente; accederemos a la “*Dope Sheet*”, dentro del menú *Graph Editors* de la barra de herramientas. En la parte izquierda del cuadro, aparece un árbol con los elementos de la escena; y en la derecha, la línea de tiempo con los fotogramas clave de cada elemento.



Si se desglosa el árbol, vemos que para cada elemento, existen las secciones Posición, Rotación y Escala; entre otras. Dichas propiedades, responden a un código de colores; y en el desglose, se puede tener acceso a cada tipo de transformación, en cada uno de los ejes.



Variaremos la posición de los fotogramas clave para ajustar el tiempo que duran los distintos movimientos de la animación. Se intentará no solapar distintas acciones, para poder ver con claridad cada una de ellas de forma consecutiva.

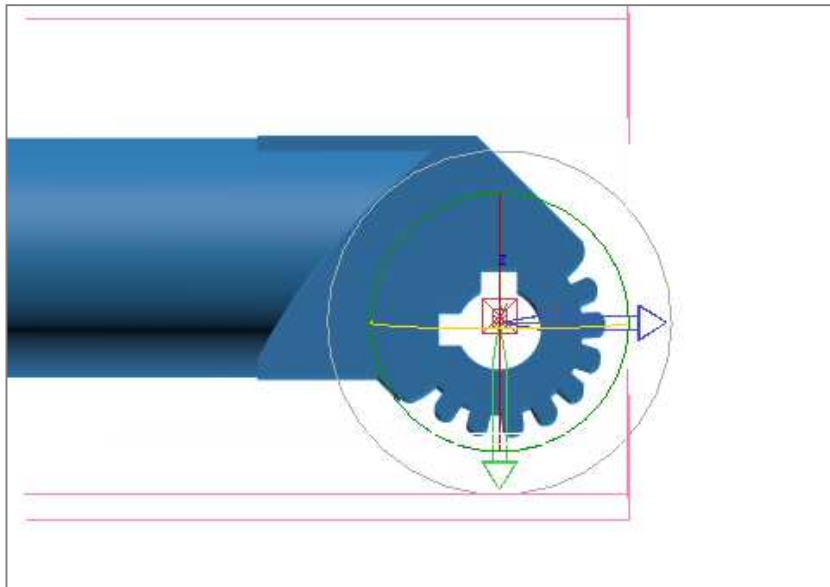
4.5-Casos a destacar

En este apartado, se destacan dos de los métodos seguidos para realizar algunos de los movimientos para generar la animación del patinete. En concreto, se trata de la animación de muelles y las barras del manillar.

4.5.1-Plegado del manillar: Wire Parameters

Como se ha descrito, las barras del manillar engranan entre sí. De esta forma, al desbloquearlas, podremos moverlas simultáneamente hasta la posición que deseemos, ya sea la de plegado o de uso.

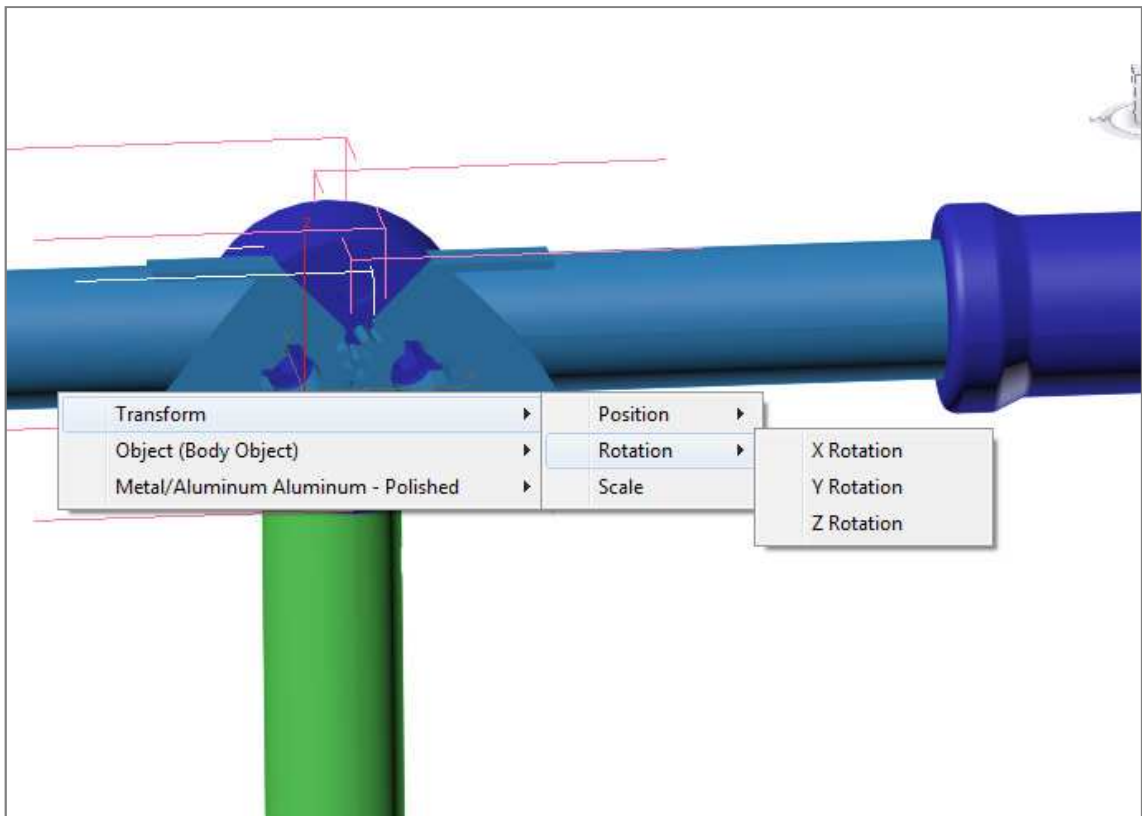
Lo primero que tendremos que hacer, será cambiar la posición del pivote de las barras a su posición correcta; en el centro del engrane. Esto, lo haremos teniendo en cuenta las coordenadas del mismo, y la posición del tornillo que sujeta cada barra al soporte del manillar, ya que serán concéntricos.



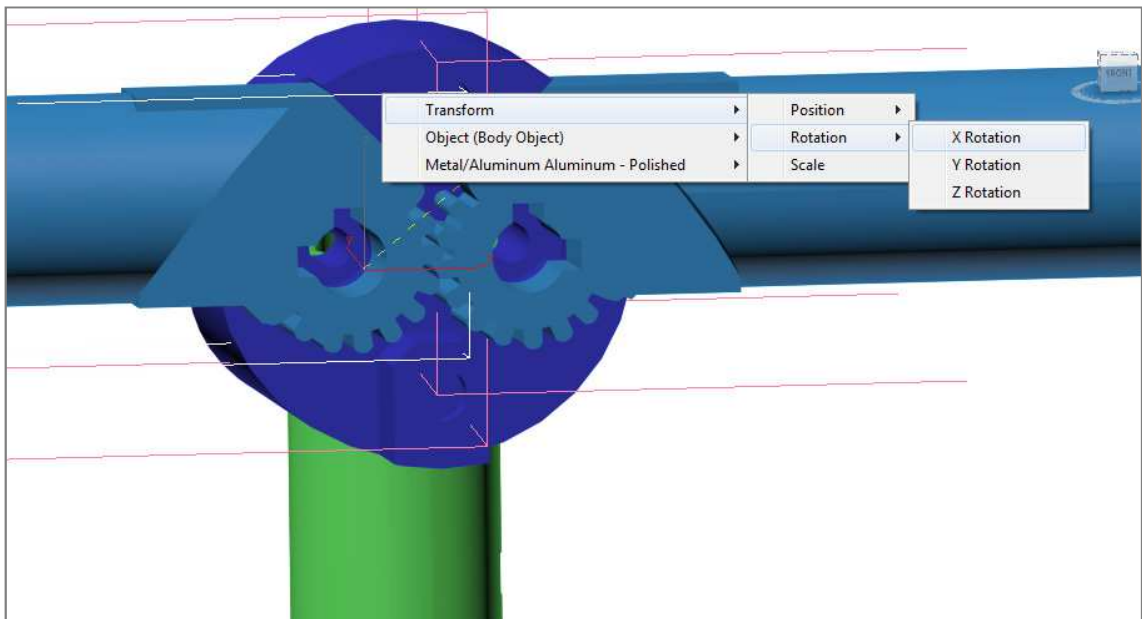
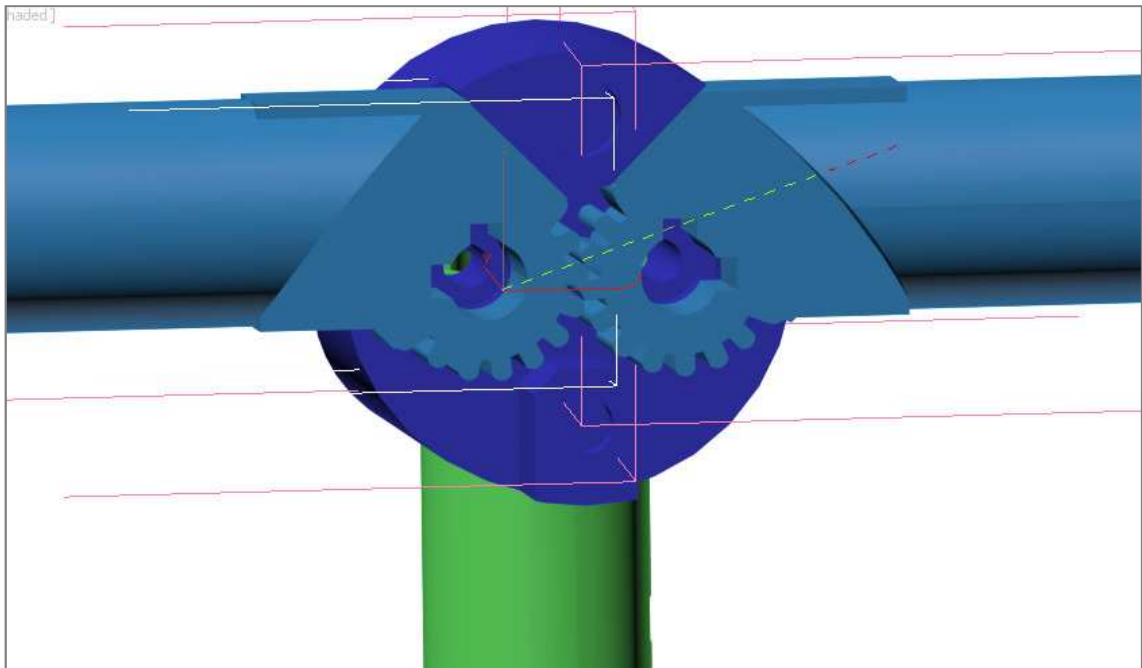
Para animarlas, asignaremos un parámetro que relacione el movimiento de una de ellas con la otra. Esto lo haremos con el comando *Wire Parámetros*. Habrá dos opciones; a través del menú Animación:

Animation > Wire Parameters > Wire Parameters, o haciendo clic con el botón derecho sobre la pieza en cuestión.

En este caso haremos clic con el botón derecho en una de ellas, elegiremos las siguientes opciones en los menús emergentes: *Wire Parameters > Transform > Rotation > (Eje correspondiente)*.



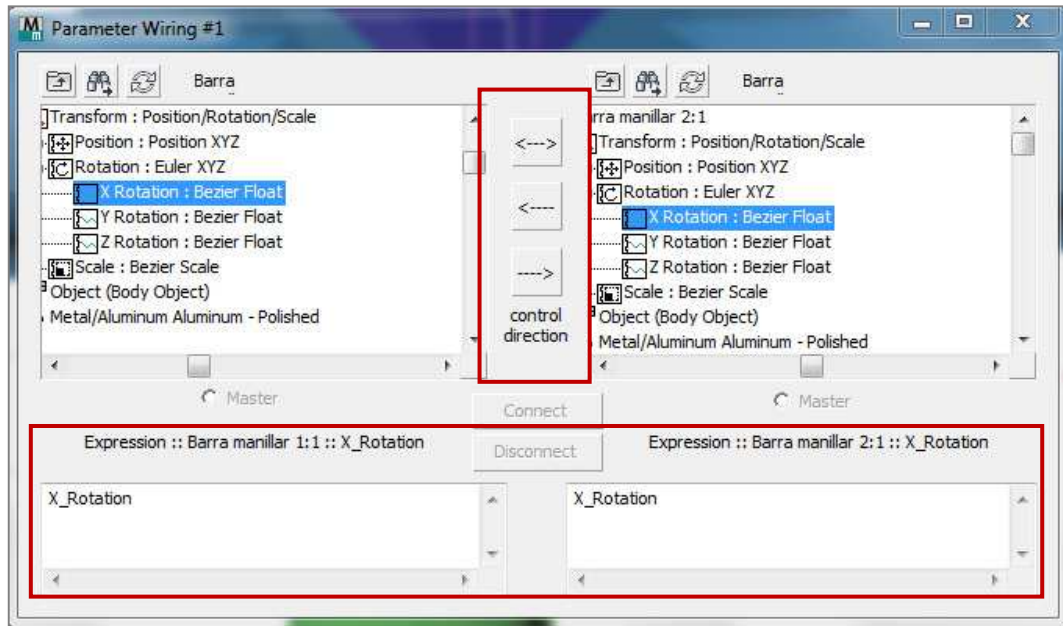
Aparece una línea de puntos que uniremos con la otra barra; a la que queremos relacionar la primera. Otra vez, nos aparecerán los menús de tipo de transformación, y eje.



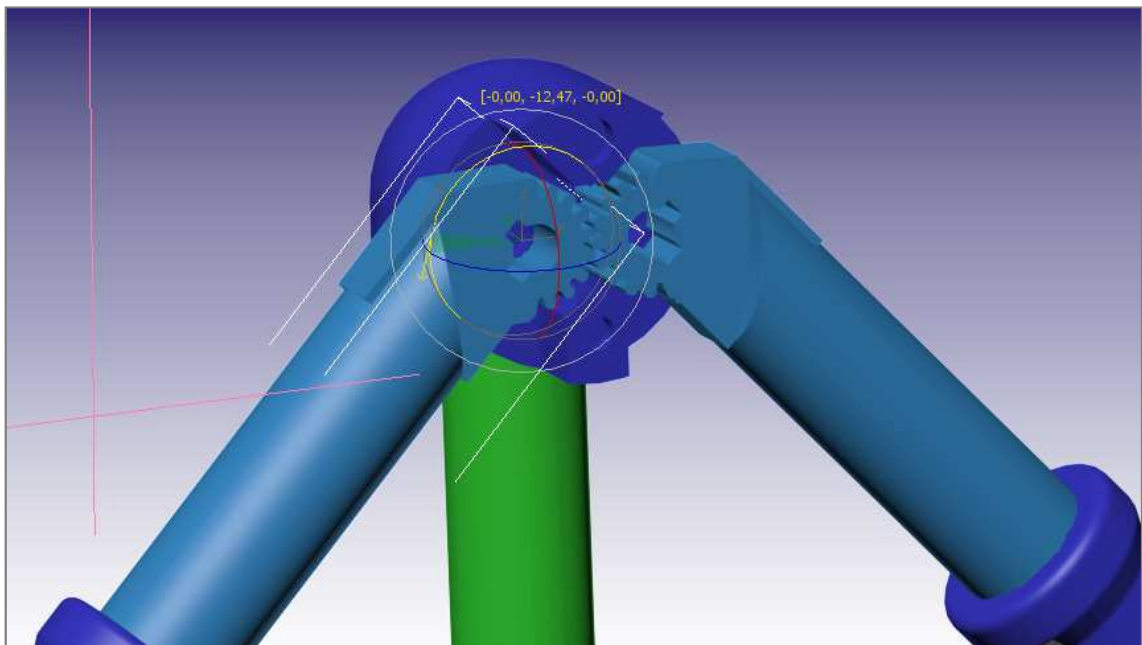
Aparecerá el cuadro Parameter Wirin, dividido en dos secciones, donde vemos representados los dos objetos, y el tipo de relación que queremos definir entre ellos. Debería aparecer sombreado el mismo eje de rotación para ambas barras.

Al ser un sistema de engranes de dientes rectos exteriores, el giro de una barra es igual y de sentido contrario al de la otra. Por lo que habrá que especificar esto en el cuadro. En ocasiones, para que una pieza gire en sentido contrario a otra; se hace añadiendo un signo negativo (-), en uno de los campos

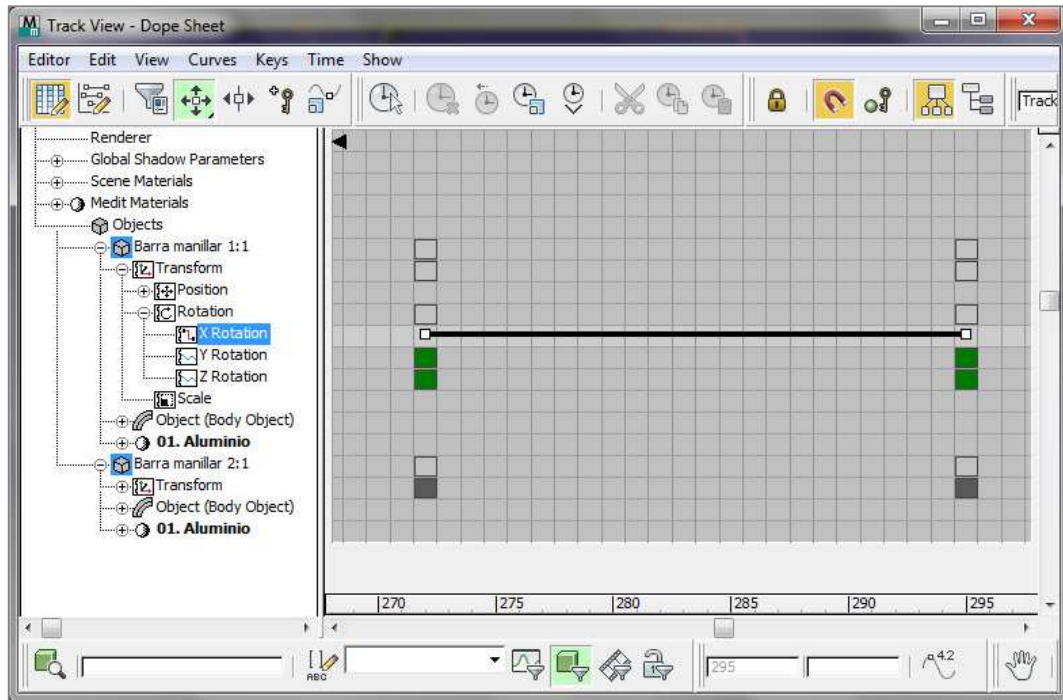
antes del tipo de rotación especificada, en la parte inferior del cuadro. Además en la parte central del cuadro, podremos controlar la relación entre las barras; que al mover una, se mueva la otra o viceversa, o que la relación sea recíproca. Definida la relación, pulsamos *Connect*.



En el momento de animar las barras, nos basta con mover una de ellas; ya que la otra seguirá el movimiento de la primera, de un fotograma clave a otro.

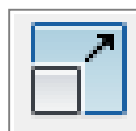


Como vimos en el apartado **4.4-Uso de Dope Sheet**, las transformaciones de rotación, se representarán en color verde. La línea negra que une los fotogramas clave, responde a la relación creada por *Wire Parameters*.



4.5.2-Compresión de muelles: Select and Non-Uniform Scale

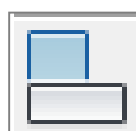
Para simular la compresión de los diferentes muelles, animaremos éstos con el control de transformación de escala; en el icono correspondiente de la barra de herramientas. En este caso, el control de transformación de escala, se trata de un botón desplegable con tres opciones:



Select and Uniform Scale

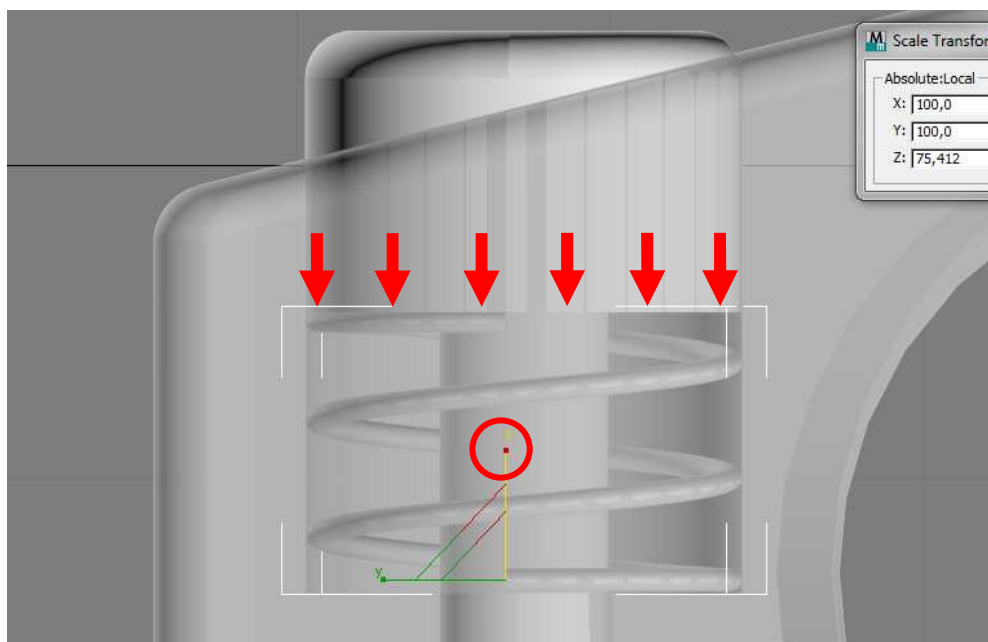
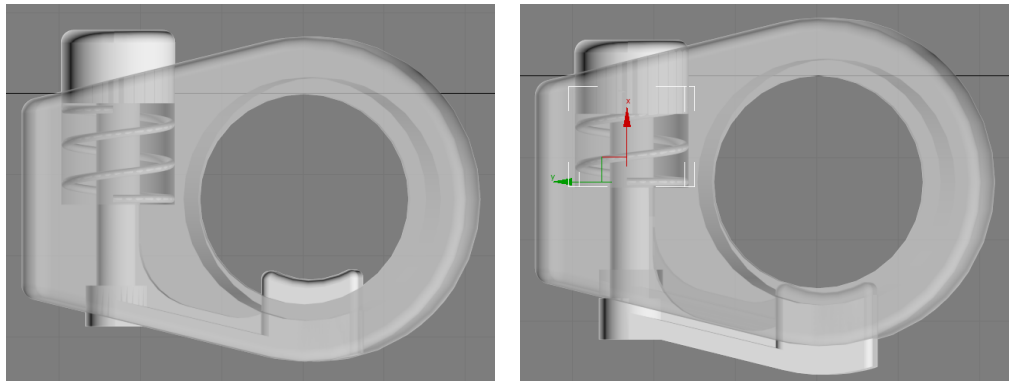


Select and Non-Uniform Scale

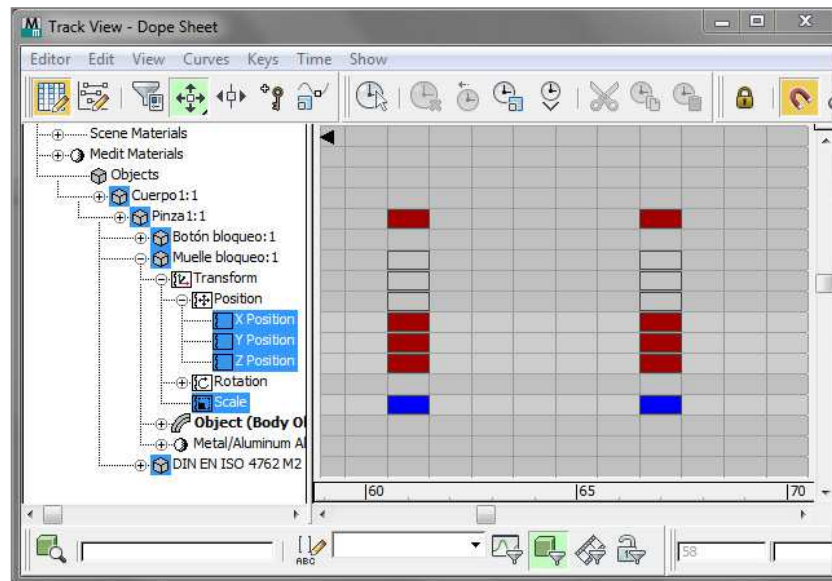


Select and Squash

En este caso se elige la opción *Select and Non-Uniform Scale*, ya que buscamos que la deformación sea sólo en una dirección. Bastará con elegir la pieza a deformar, y a continuación hacer clic sobre el gizmo de transformación, en el eje deseado.



Como vimos en el apartado **4.4-Usa de Dope Sheet**, las transformaciones de escala, se representarán en color azul.



4.6-Materiales en las animaciones (Materiales animados)

Mencionar en este apartado, que para las animaciones, en algunos casos se han utilizado materiales distintos a los empleados para los render. Esto se debe principalmente a los siguientes factores:

4.6.1-Transparencia de objetos

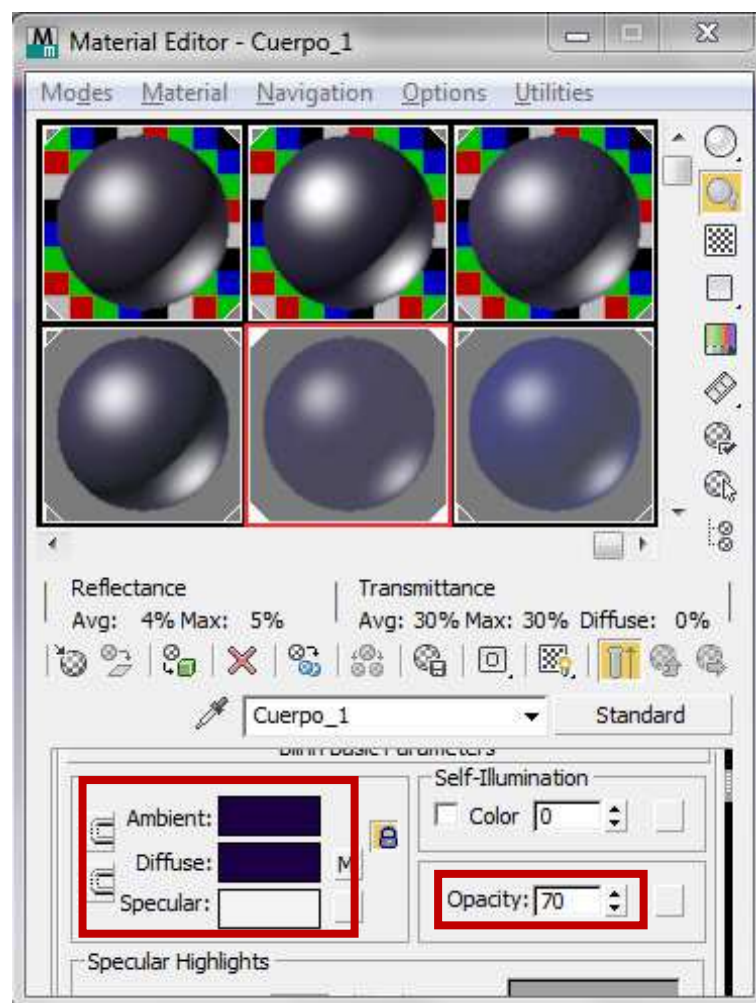
Uno de los propósitos de las animaciones es mostrar el funcionamiento de los distintos mecanismos de plegado del patinete. Se pretende para ello, hacer que alguna de las piezas se haga transparente, para dejar ver lo que ocurre en su interior; mostrando cómo interactúan las piezas entre sí.

Observamos, que algunos de los materiales que se usaron para los render, no permiten esto; por lo que se opta por materiales estándar, a los que se aplica la transparencia. Ha de haber una transición en la transparencia de los objetos; esto quiere decir que ha de animarse el cambio de transparencia en el material.

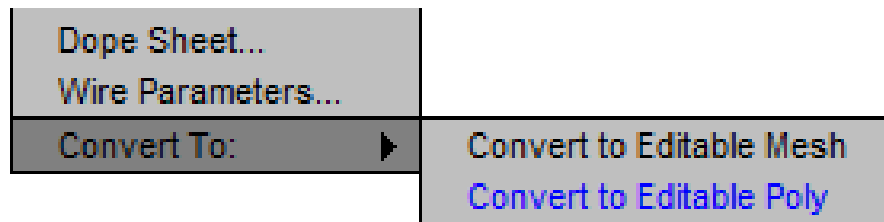
Al igual que si se tratara de un objeto, un material también se puede animar cambiando sus parámetros; como el color, brillo u otras propiedades. La característica que queremos cambiar

principalmente en este caso, es la transparencia; o como se define en el editor de materiales, la opacidad.

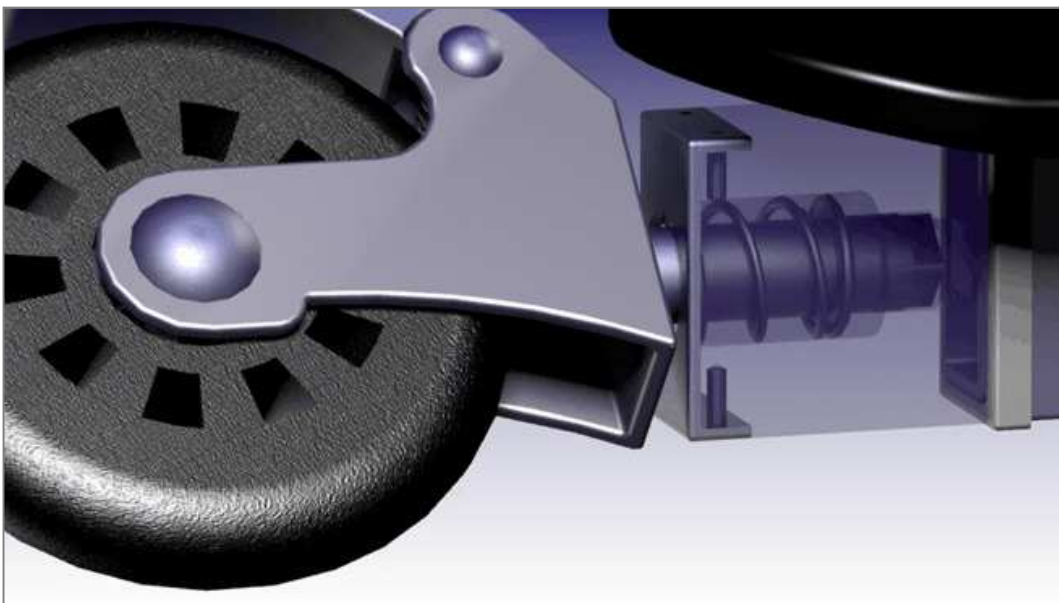
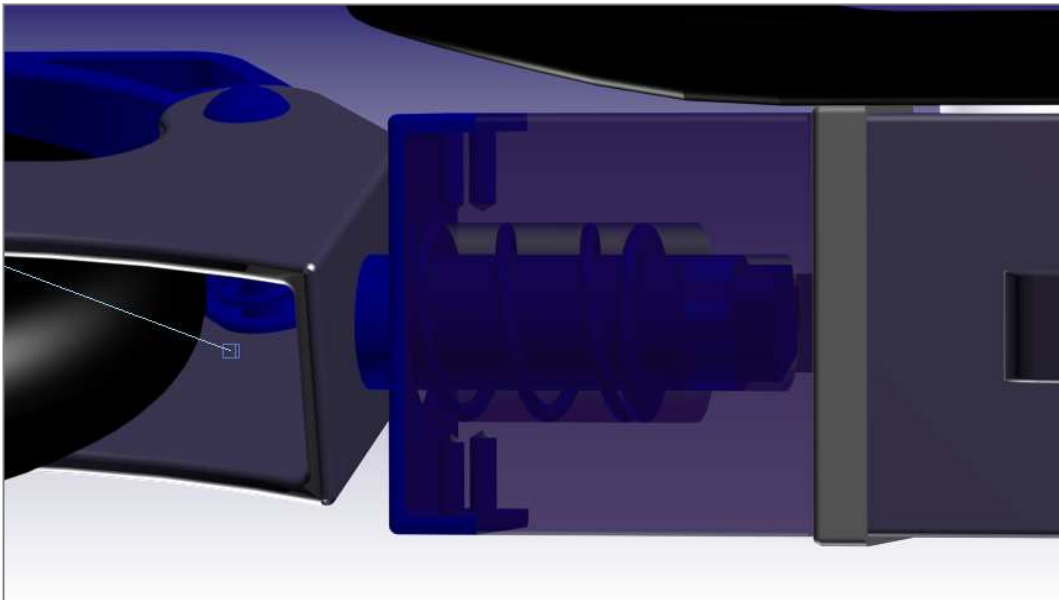
Activando el *Auto Key*, y seleccionando un material en el editor de materiales, vemos que además del visor activo, también se enmarca en rojo la muestra de material. Variando alguno de los parámetros, crearemos un fotograma clave para ese material; comportándose igual que si fuera un objeto.



Bajando el valor de la opacidad, vemos cómo la pieza va haciéndose transparente. También podemos cambiar los colores difuso y ambiental, para mejorar la estética del objeto. Para ver como el objeto se transparenta, antes debemos convertirlo a malla editable (*Editable Mesh*) o conjunto de polígonos editable (*Editable Poly*)



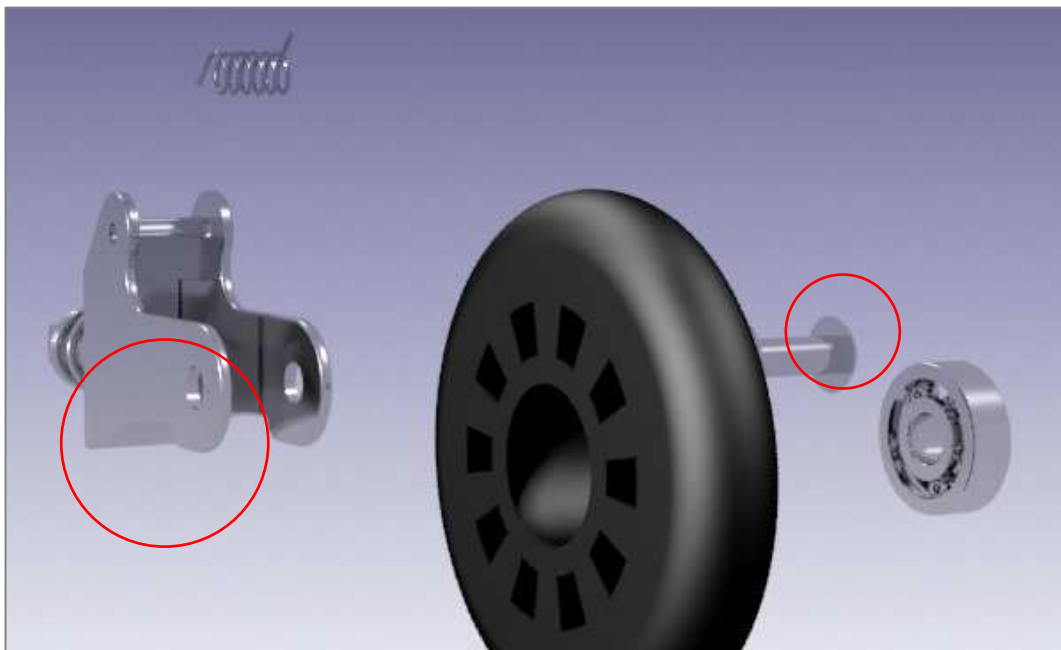
Así, podemos ver lo que pasa en el interior de las piezas a las que apliquemos un material editado de esta manera.



4.6.2-Reflexión del entorno

Como se explicó en el apartado **4.2-Entorno de la animación**; en algunas de las animaciones, se opta por un entorno diáfano; un degradado de azul oscuro a blanco. Esto, junto con los materiales elegidos para los render; los cuales tienen una alta reflexión, dará lugar a algunos problemas.

A pesar de que las piezas metálicas, tiene mucho brillo y alta reflexión, no reflejan casi nada; ya que no tienen elementos cercanos que reflejar. Por el contrario, reflejan totalmente el entorno; pero como este es un degradado, dará lugar a zonas poco definidas y pérdidas de geometría. En ocasiones, algunas partes de las piezas llegan incluso a confundirse con el fondo.



Además de esto, en los montajes también necesitamos alguna transparencia puntual. Cambiando los materiales que teníamos de antes, a materiales estándar, junto con algún ajuste en la iluminación de la escena; obtenemos resultados bastante satisfactorios. Los materiales escogidos esta vez, son menos brillantes, pero las piezas se ven más definidas.



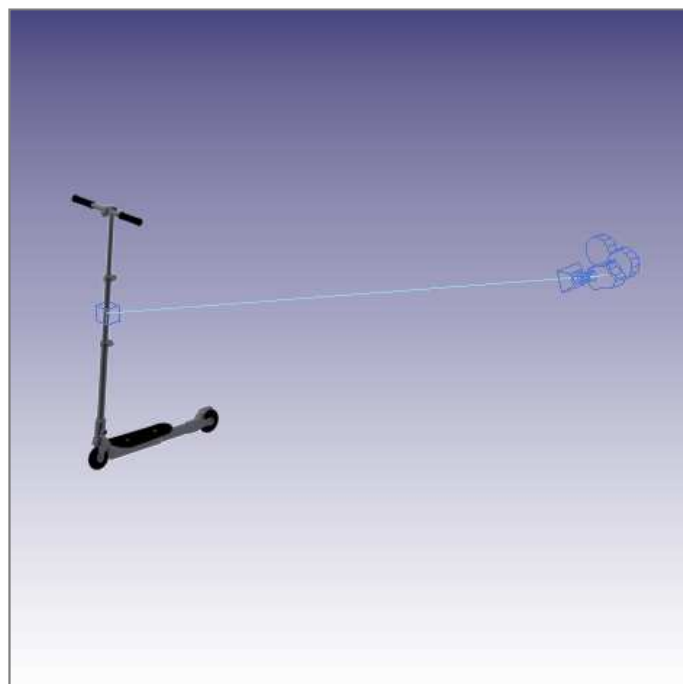
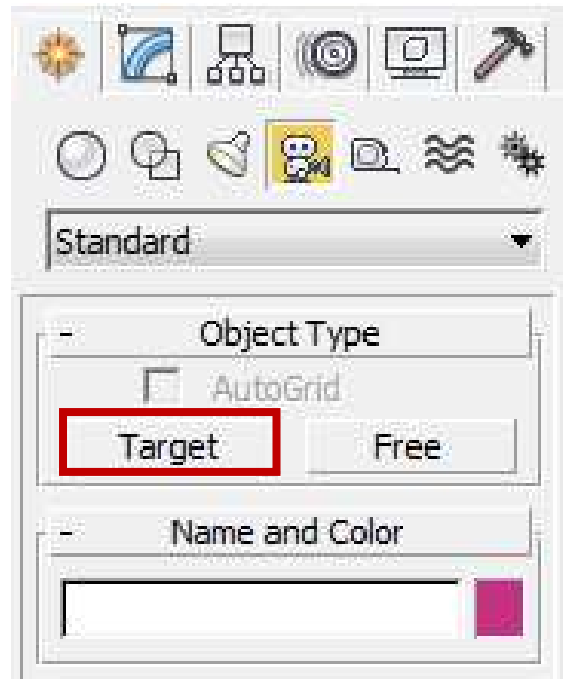
Puede que aplicar estos materiales, suponga perder algo de realismo. Esto no es muy importante en este tipo de animaciones; ya que lo que se busca aquí es mostrar el funcionamiento o montaje de las piezas, de forma clara y definida, lo cual se ha conseguido.

4.7-Movimientos de Cámara

Para realizar los vídeos, colocaremos normalmente una cámara, que también se animará para dar los planos que se deseen, según el momento y/o la acción que se quiera captar.

Se elige una cámara del tipo Target. Este tipo de cámaras tienen un punto de enfoque u objetivo (Target), que será muy útil para el tipo de vídeos que se quieren realizar; sobre todo, cuando se trate de detalles.

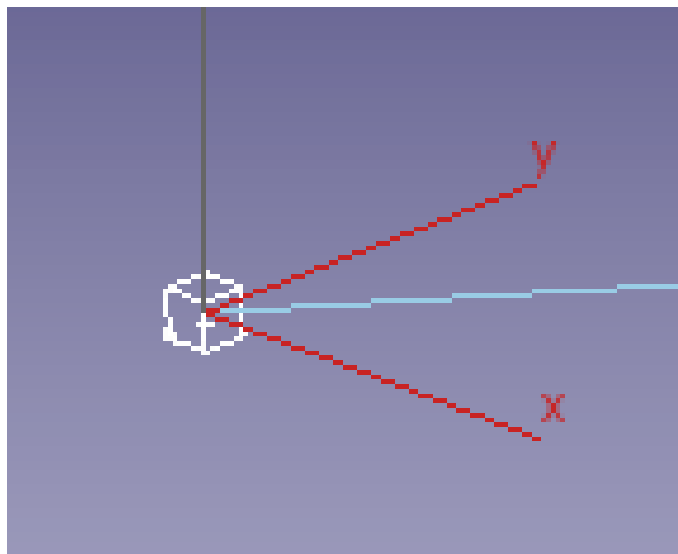
Para crear la cámara, accederemos a la sección *Cameras*, dentro del panel *Create*. Escogeremos, en la categoría *Standard*, una cámara tipo Target.



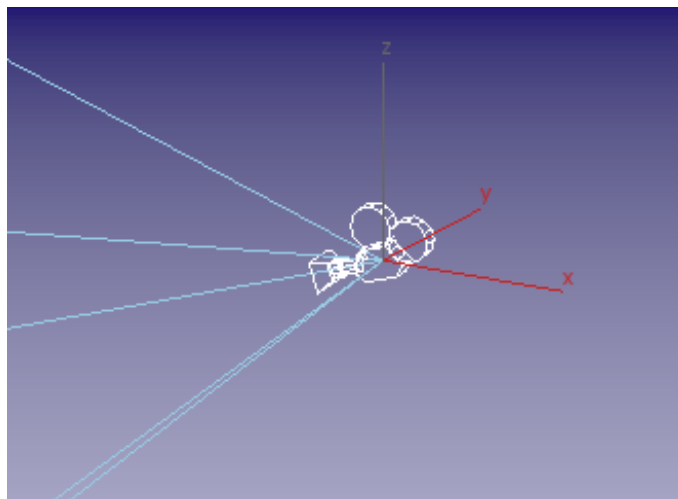
Una vez seleccionado el tipo, en cualquiera de los visores, se pica con el ratón para crear la cámara. A continuación y sin soltar de ratón, se arrastra para situar el objetivo o target. En la parte superior izquierda del visor; o con la tecla C, podremos seleccionar la vista Camera. De esta forma, veremos lo que se capta a través de ella.

Al igual que con cualquier otro elemento de la escena, podremos animar la cámara. Para ello, activaremos el botón *Auto Key*, como hicimos anteriormente. Como cámara y objetivo, son objetos independientes, podremos variar la posición de cada uno de ellos indistintamente. Con esto, conseguiremos enfoques distintos.

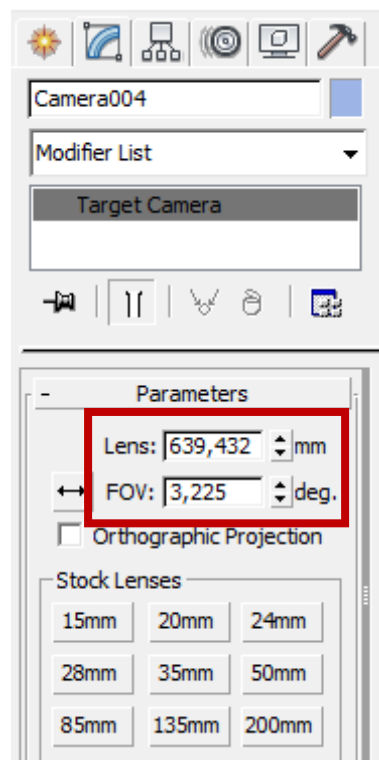
Objetivo (Target):



Cámara:

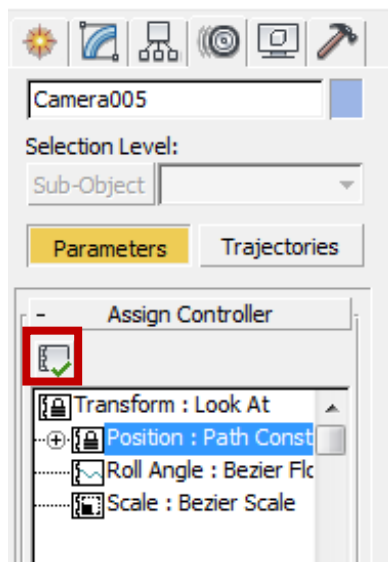


En la pestaña Parameters, dentro del panel *Modify*; podremos variar el tamaño de la lente o el Campo de visión (Field Of View); con lo que conseguiremos planos más amplios para vistas generales, o más cortos cuando se trate de resaltar detalles. Lentes de menor o mayor tamaño, respectivamente. Así se podrá variar el encuadre, sin necesidad de modificar la posición de la cámara y/o el objetivo.



4.7.1-Asignar una Trayectoria a la Cámara

Otro método que se utiliza, es asignar a la cámara una trayectoria. Como en este caso, se requiere que la cámara gire alrededor del patinete para ofrecer distintas vistas y detalles de todo él, se optará por una trayectoria circular o helicoidal. Para asignar la trayectoria, se siguen los siguientes pasos.



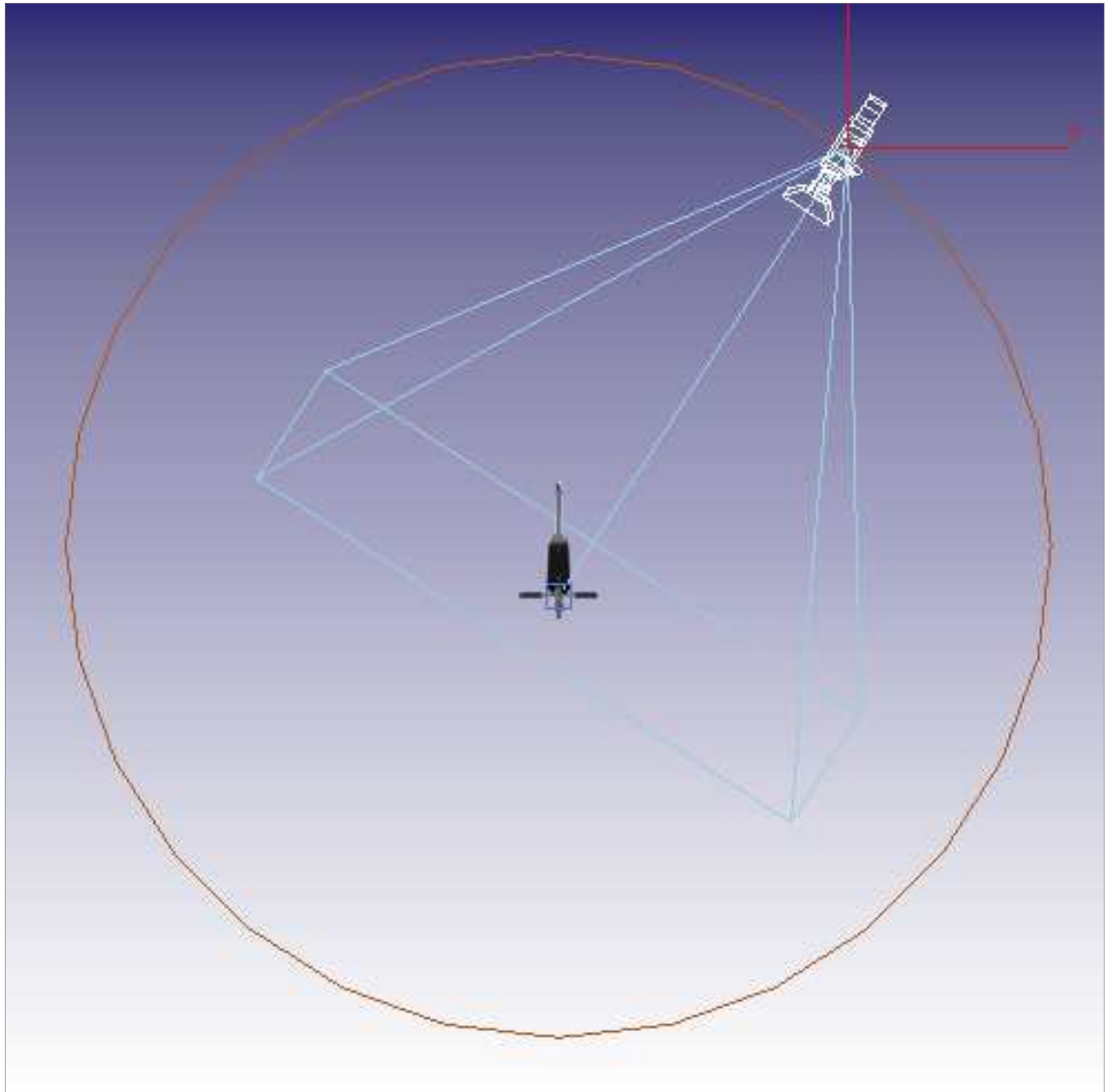
Dentro del panel *Motion*; con el botón *Parameters* resaltado en naranja y la cámara seleccionada, escogeremos *Position* en *Assign Controller* y pulsaremos el icono para determinar el tipo de controlador que queremos asignar a la cámara.

En la lista de controladores que aparecerá al hacer clic en el icono, seleccionaremos *Path Constraint* (Restricción de Trayectoria).

A continuación haremos clic sobre la línea que hayamos dibujado previamente y que será la trayectoria que seguirá la cámara.

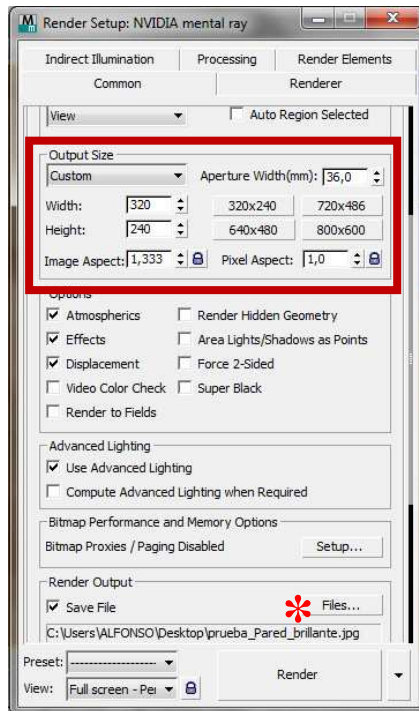
En el área *Path Options*, del panel *Motion*, podremos definir la posición de la cámara a lo largo de la trayectoria. Con el *Auto Key* activado, podremos animarla.

En la siguiente imagen se muestra la cámara, que sigue una trayectoria circular alrededor del patinete.



Las animaciones se harán con el *Auto Key* activado y combinando la animación de la cámara a lo largo de la trayectoria, el Campo de Visión o el tamaño de la lente, e incluso animando la trayectoria. Para mover la cámara a lo largo de la trayectoria de forma precisa, lo haremos dentro del panel *Motion*, en la sección *Path Options* ajustaremos el valor para el parámetro Along Path.

4.8-Exportación de Vídeos



De la misma forma que para las imágenes, accederemos al panel de configuración de renderizado (Render Setup), en el icono correspondiente en la Barra de Herramientas:

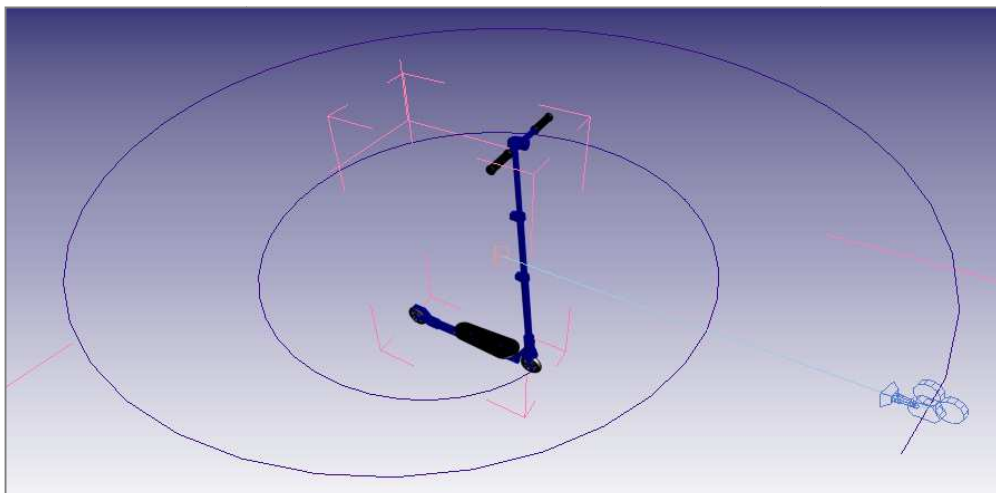


En el área Output Size, determinaremos el tamaño los videos que queremos exportar y su resolución.

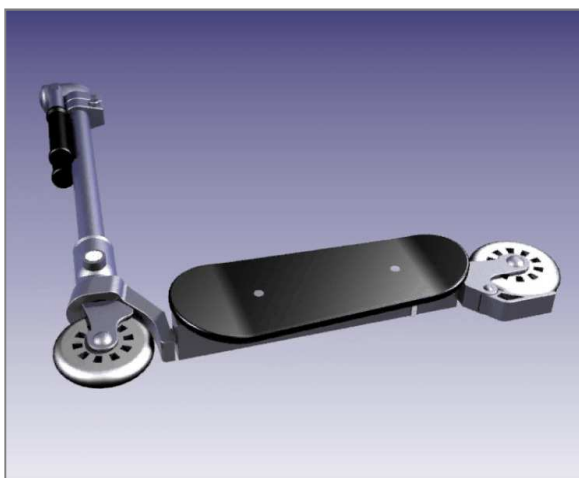
Más abajo, en el área *Render Output*, escogeremos el directorio para guardar los archivos exportados, haciendo clic en el botón Files *. Escogeremos un formato de video, normalmente AVI.

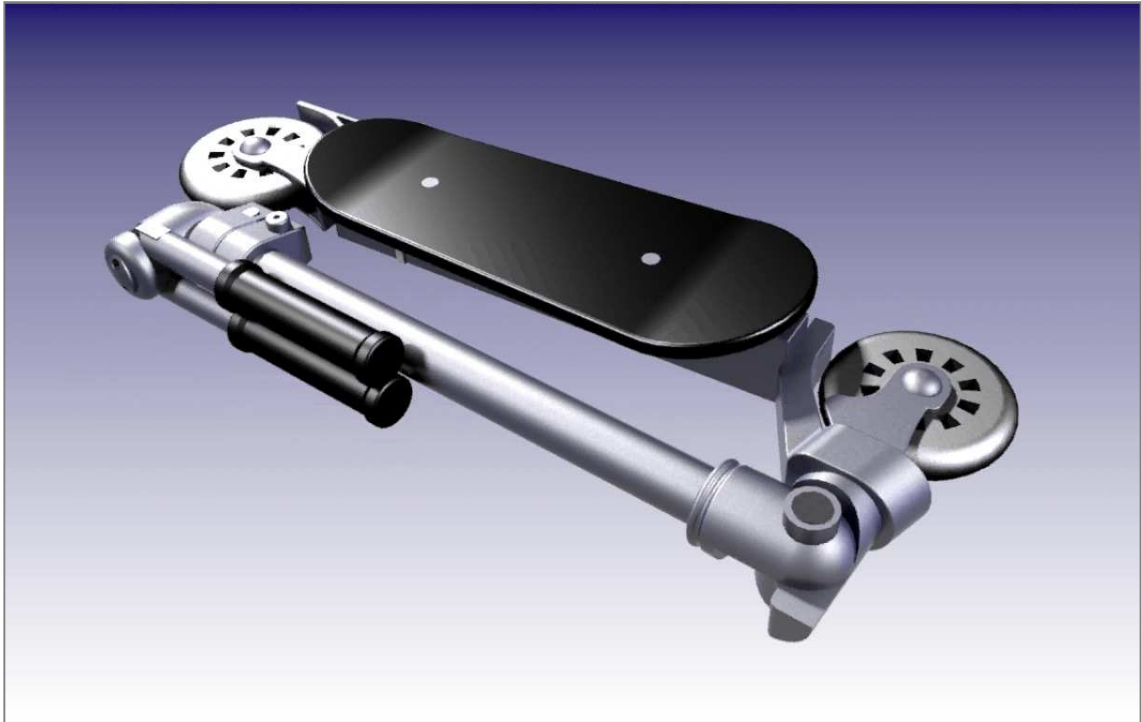
4.8.1-Videos Generales

Se hará una animación de conjunto, en la que veremos una vista general del patinete, así como la forma de plegado. En este caso se utiliza una trayectoria en forma de hélice, además de animar el objetivo.









4.8.2-Vídeos de Montaje

Igual que se hizo para elaborar los planos del producto, se divide éste en subconjuntos. Se hace una animación en la que se puede ver como se monta cada uno de ellos, y una del montaje general.

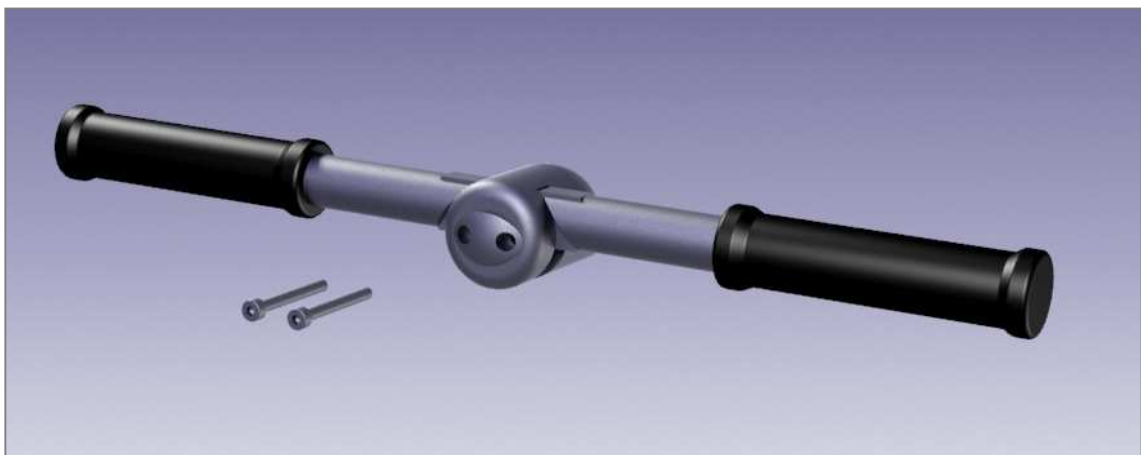
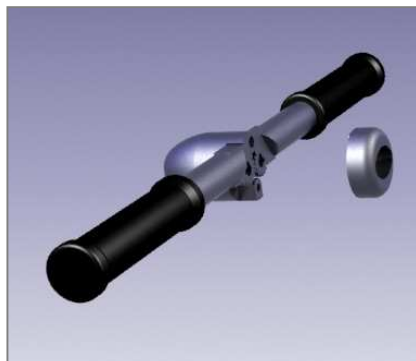
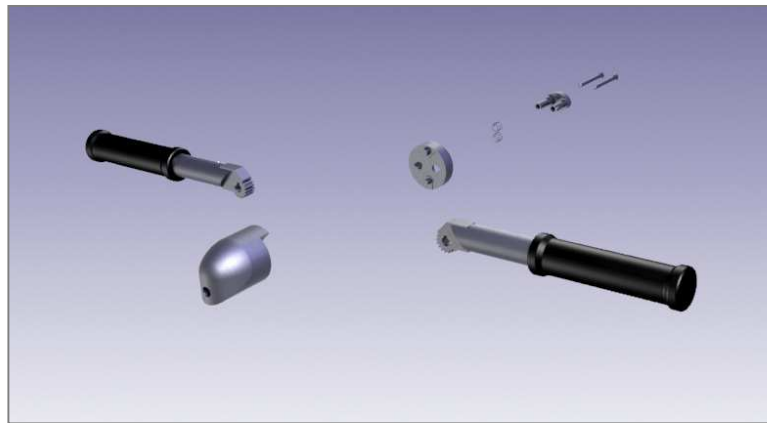
En estas animaciones, se utiliza una trayectoria circular. De esta forma, la cámara gira alrededor de las piezas y vemos como van encajando unas con otras desde diferentes ángulos, además de variar el Campo de Visión (*FOV*) en determinados momentos.

El video empezará haciendo un plano general; alrededor del conjunto explosionado, para poco a poco ir viendo las diferentes fases del proceso de montaje y las piezas que intervienen. Terminado el proceso, se gira alrededor nuevamente, para ver el conjunto una vez montado.

En ocasiones aparecerán piezas ya montadas en la animación; esto ocurrirá con aquellas, que por ejemplo, necesiten ser soldadas.

MANILLAR

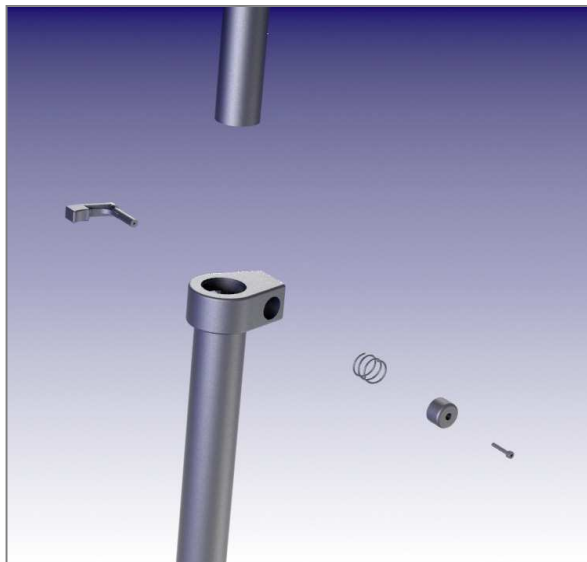
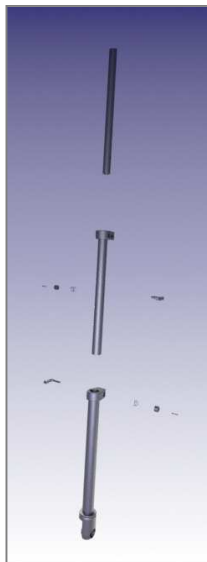
El subconjunto manillar consta de 11 piezas (los manguitos ya vienen colocados en cada una de las barras). En un primer momento, la cámara gira sobre el despiece explosionado, para ver todas las piezas que lo componen de forma general. A continuación, la animación muestra la forma en que se montan, dando finalmente una visión global del conjunto, una vez montado.





COLUMNA DE DIRECCION

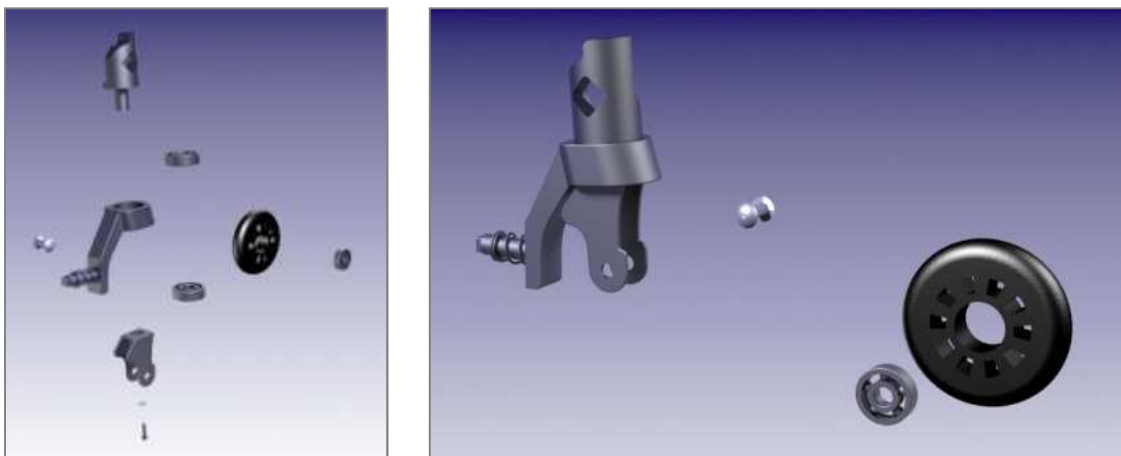
En este caso, el subconjunto columna de dirección está formado por 14 piezas (La unión de algunas de ellas no se muestra en la animación). Esta vez, como el conjunto es de mayor tamaño que el manillar; además de los giros de cámara alrededor del conjunto, haremos barridos a lo largo de éste, para enfocar las zonas que interesen en cada momento. Esto último lo hacemos combinando por una parte, diferentes campos de visión para ampliar o reducir el enfoque, la animación del objetivo (ya que usamos una cámara tipo *Target*), y animando también en ocasiones, la trayectoria circular por la que discurre la cámara.

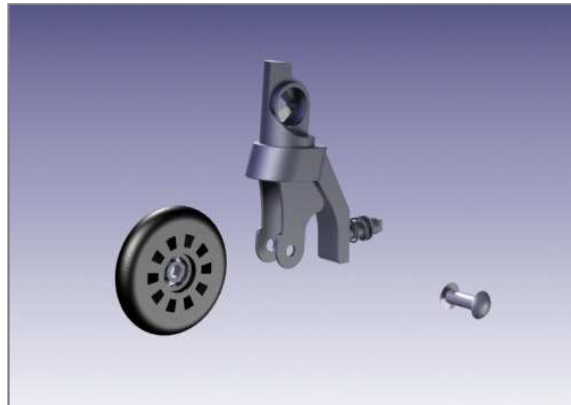




RUEDA DELANTERA

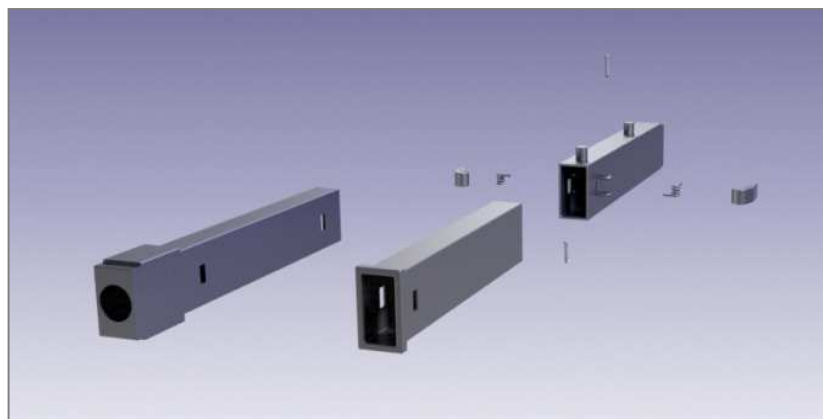
El subconjunto rueda delantera consta de 12 piezas; tres de ellas ya aparecen unidas en la animación. La cámara gira alrededor del despiece, mostrando el proceso de montaje y dando finalmente una visión global del conjunto, una vez montado. Se combinan los giros, con diferentes enfoques.

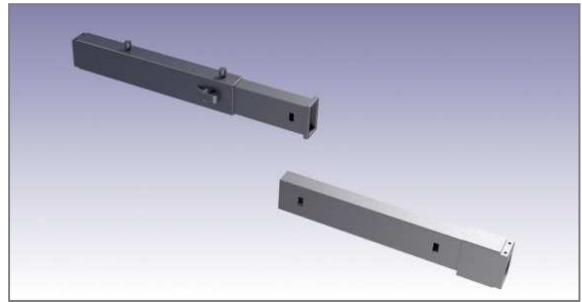
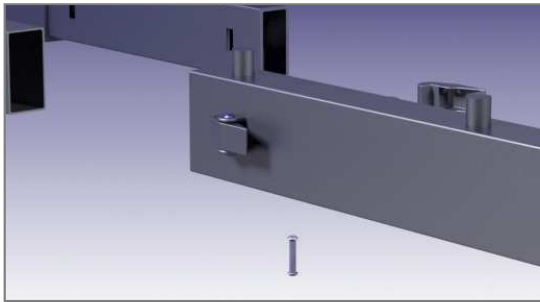




PLATADFORMA

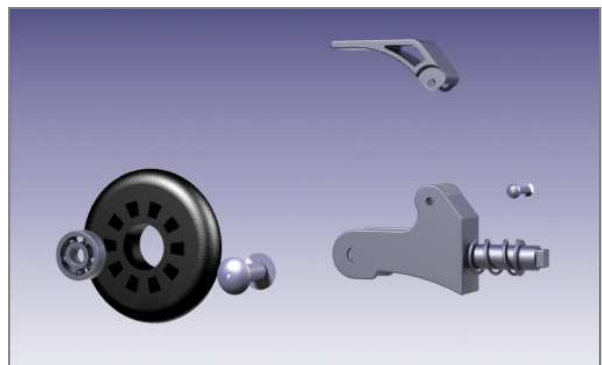
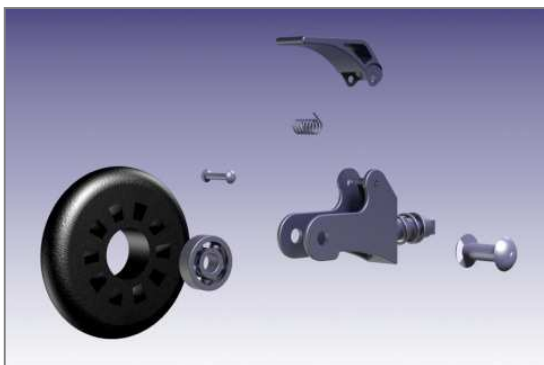
De las 17 piezas que componen el subconjunto plataforma, algunas ya han sido soldadas previamente. De igual forma que para la rueda delantera, se combinan los giros de la cámara alrededor del subconjunto, con diferentes enfoques.

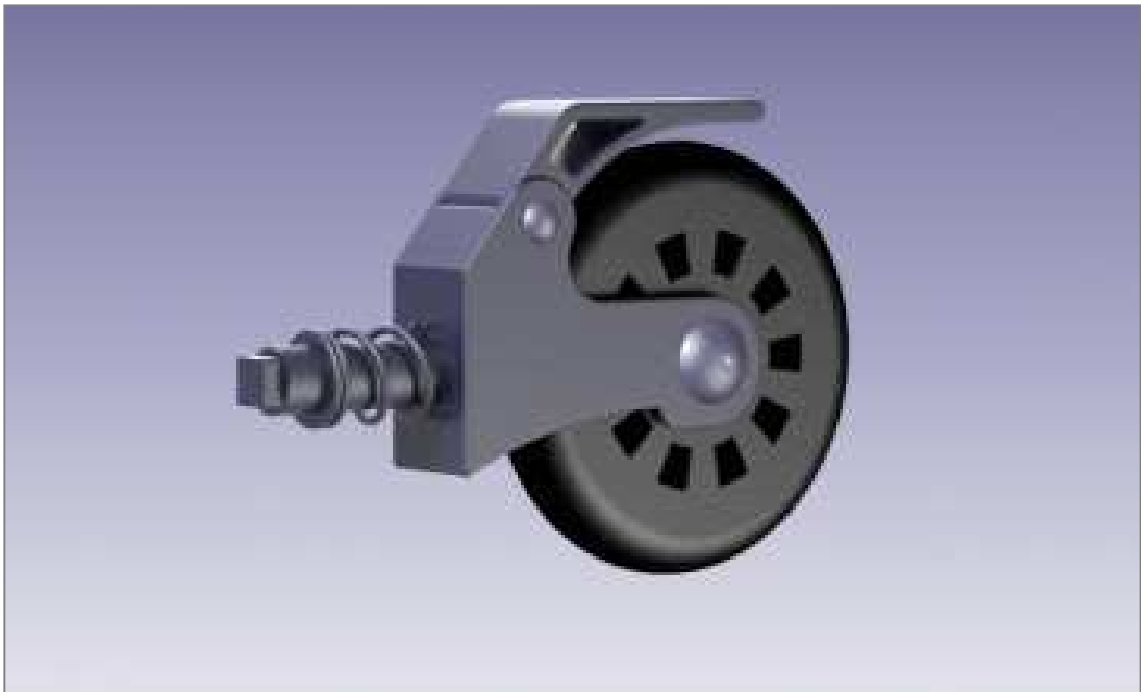




RUEDA TRASERA

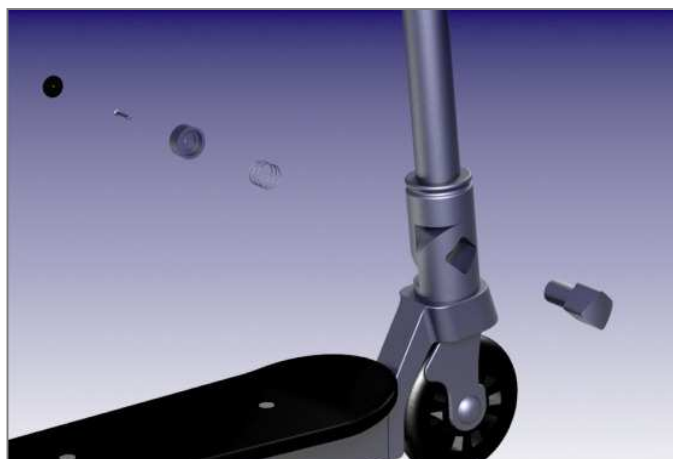
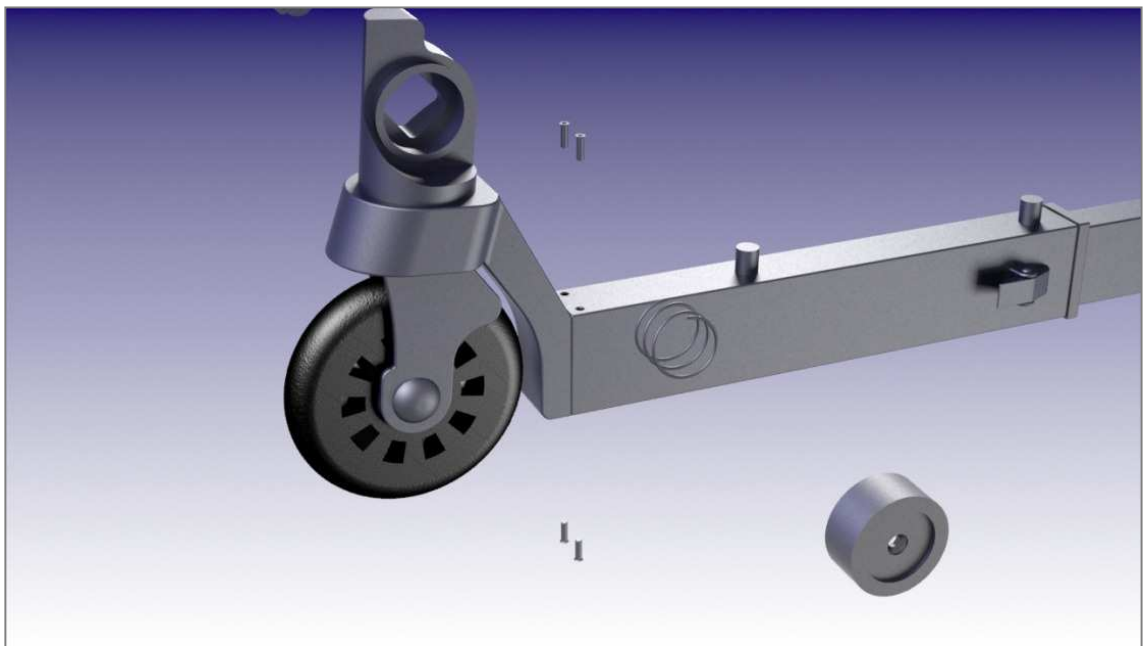
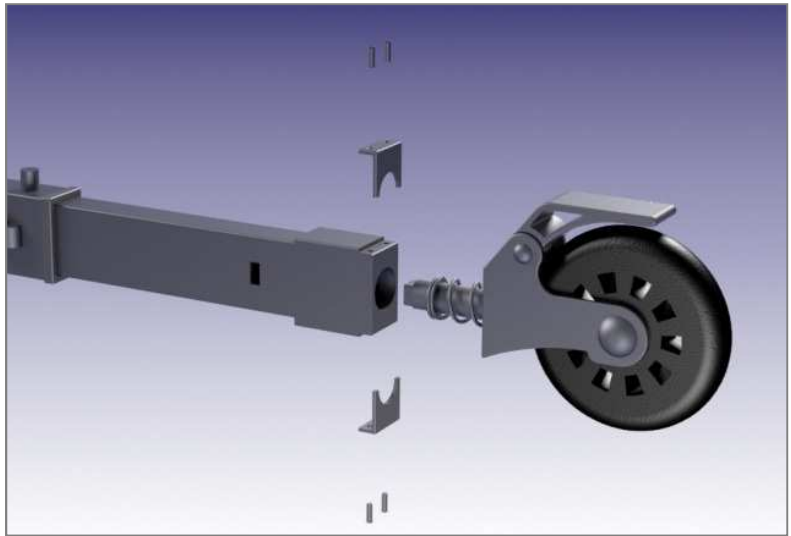
La rueda trasera estará formada por 10 piezas; cuatro de las cuales, aparecen ya unidas. La animación es similar a las anteriores, girando alrededor de las piezas mientras se muestra el proceso de montaje.

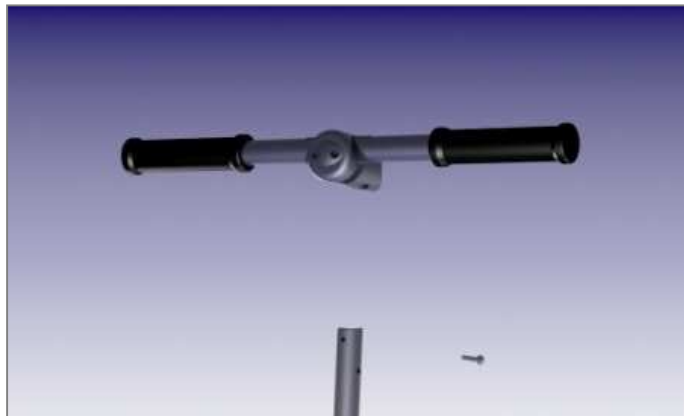
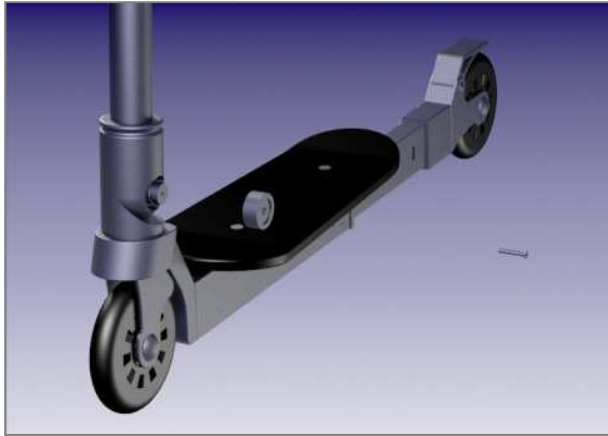




PATINETE COMPLETO

El patinete, consta de de los cinco subconjuntos citados anteriormente, las piezas que unen éstos entre sí y la tabla; en total 25 elementos. En este caso, se combina todo lo utilizado anteriormente para las grabaciones de los subconjuntos; es decir; animación de la trayectoria de la cámara (en el caso de los montajes, escogimos una trayectoria circular), animación del objetivo (*Target*), y variaciones en el campo de visión (*FOV*). Lo cual nos permite cubrir todas las fases del montaje; desplazándonos al lugar en que sucede la acción y enfocando las zonas necesarias.



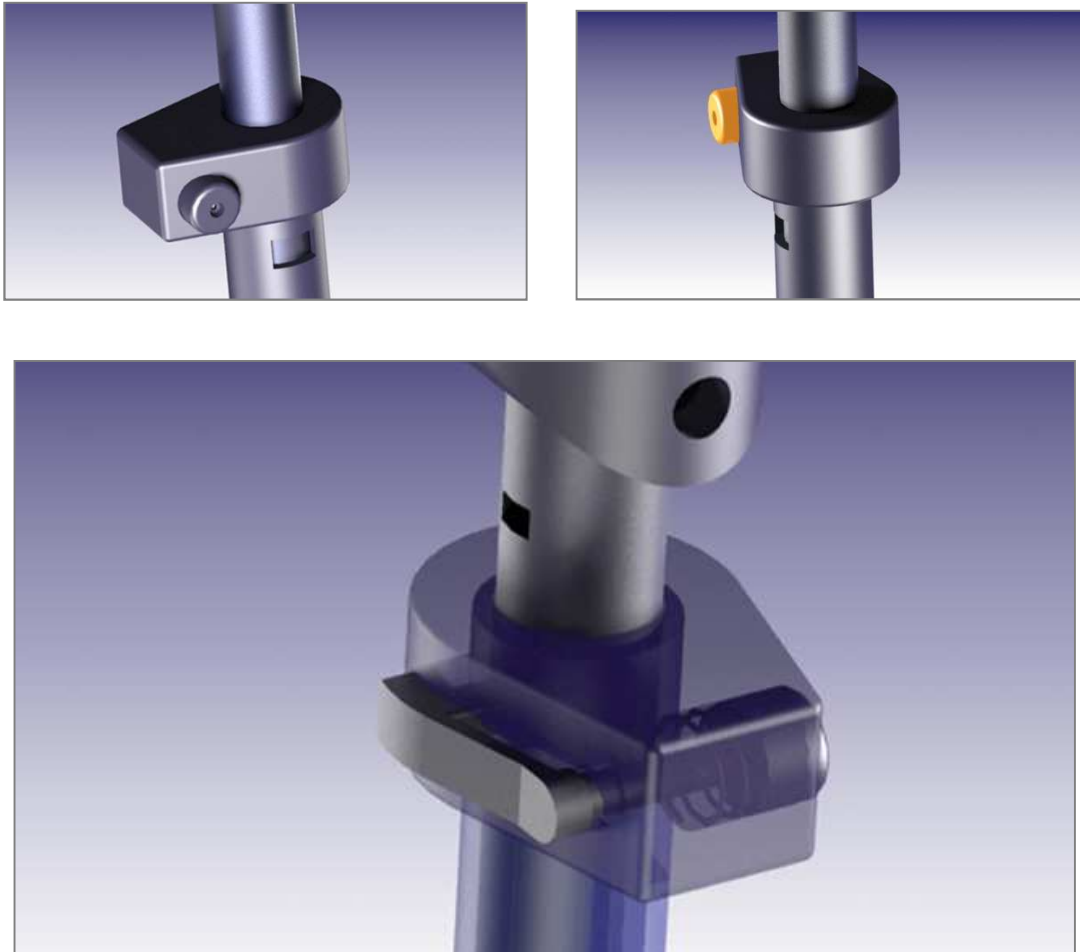


4.8.3-Vídeos de Funcionamiento

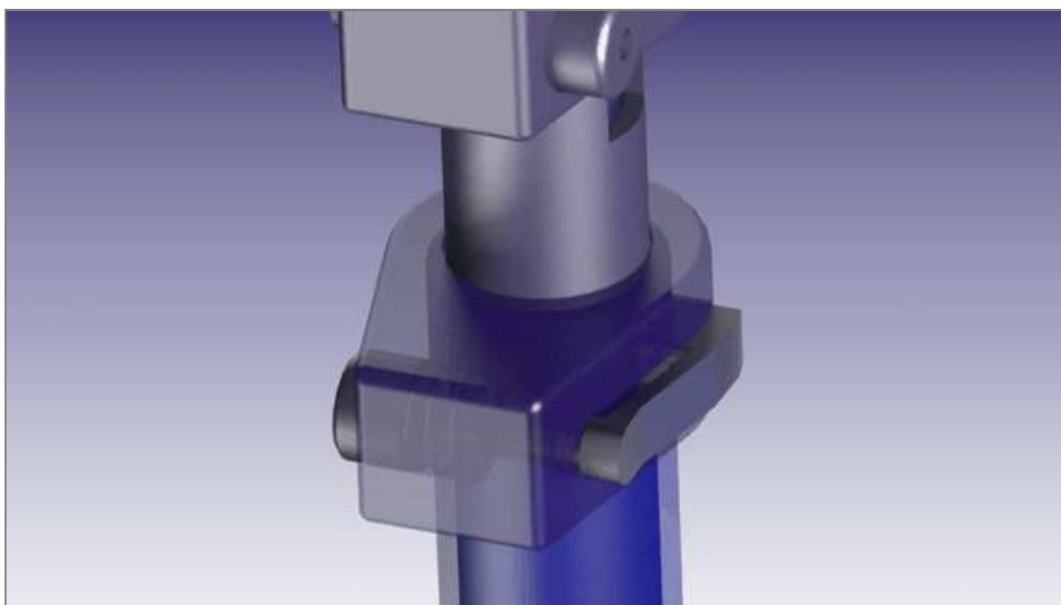
Animación en la que se muestra una vista general del producto, para a continuación ir haciendo primeros planos sobre aquellas piezas que hay que accionar para llevar a cabo las distintas operaciones de plegado.

Para esta animación, se utiliza también una trayectoria circular, así como la variación del Campo de Visión para los efectos de zoom. El barrido sobre los distintos mecanismos, se hace animando el objetivo (*Target*). Para dar una idea clara de cómo funcionan los sistemas de plegado, se resaltarán las piezas que hay que accionar en cada momento. Desde su posición más extendida, habrá que realizar 6 movimientos; es decir, habrá que accionar 6 mecanismos diferentes.

Desbloqueo y plegado del tercer tramo de la dirección. El botón se resalta, indicando que esta pieza la que hay que accionar para acortar la dirección, a su posición intermedia.

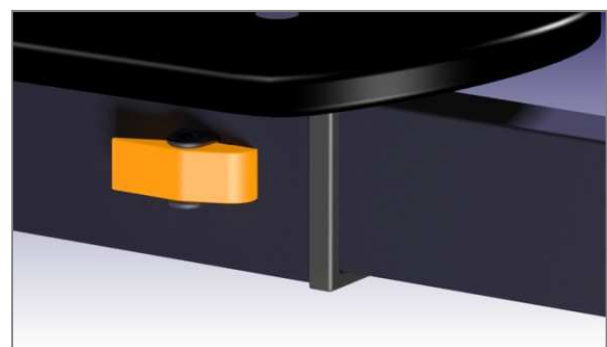
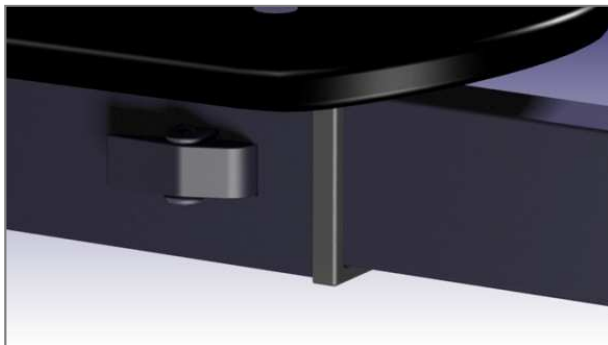


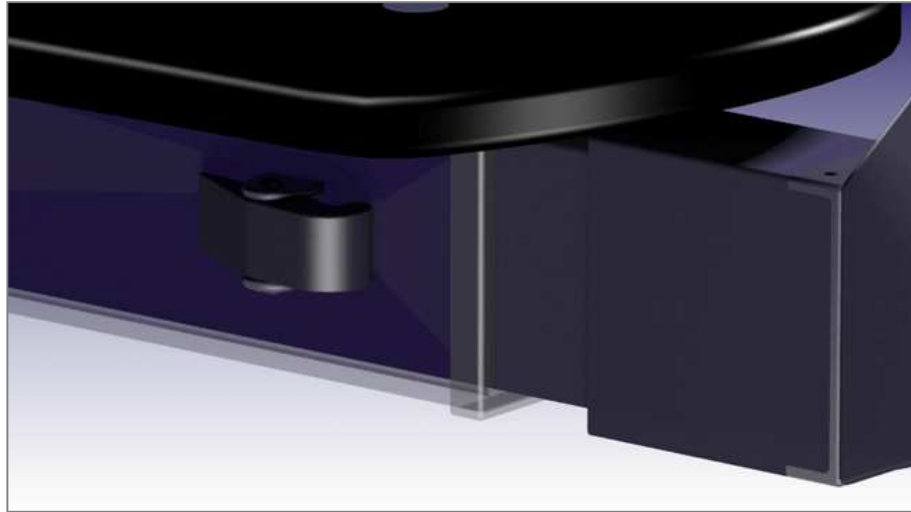
Desbloqueo y plegado del segundo tramo de la dirección. De igual forma que el anterior, el botón se resalta, indicando que esta pieza la que hay que accionar para acortar la dirección, esta vez a su posición más baja.





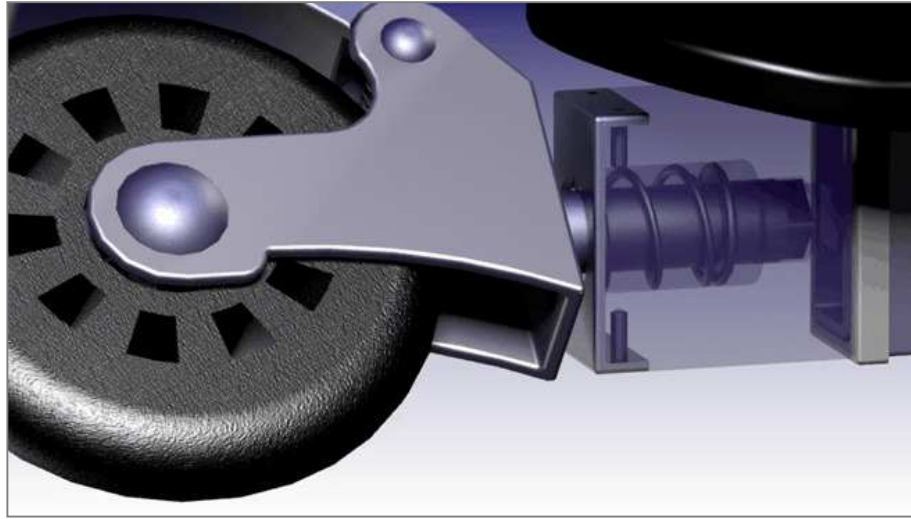
Plegado de la base. En este caso, se resaltan las pinzas laterales en el tramo de mayor sección de la plataforma; de esta forma el tramo de menor sección puede deslizarse por el primero, para compactar la base.





Giro de la rueda trasera. A continuación, hay que girar 90 grados la rueda trasera; de forma que quede paralela a la tabla. Para ello, habrá que accionar la horquilla trasera, la cual se resalta en la animación.



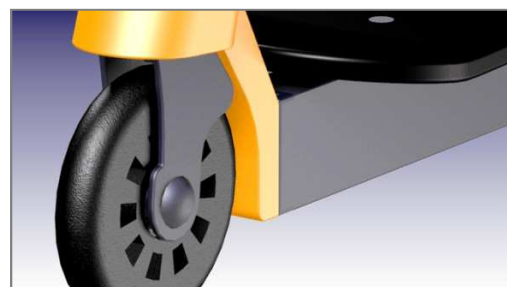
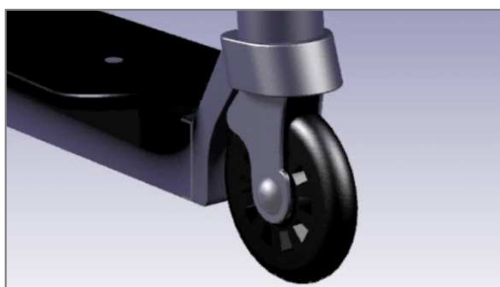


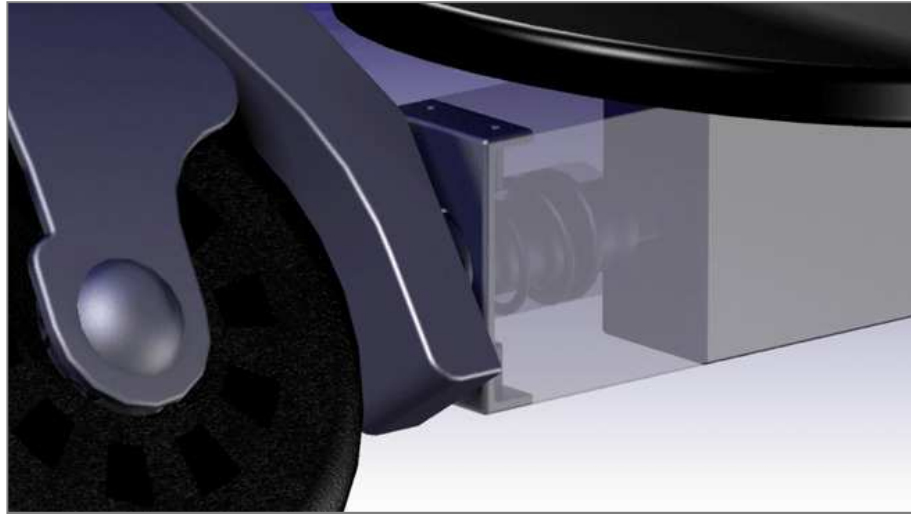
Plegado del manillar. Debemos plegar el manillar, de forma que ambas barras quedarán alineadas con la dirección. El botón que se resalta en la animación, desbloquea las barras. Bastará con accionar una de ellas para realizar esta operación, ya que engrana una con otra.





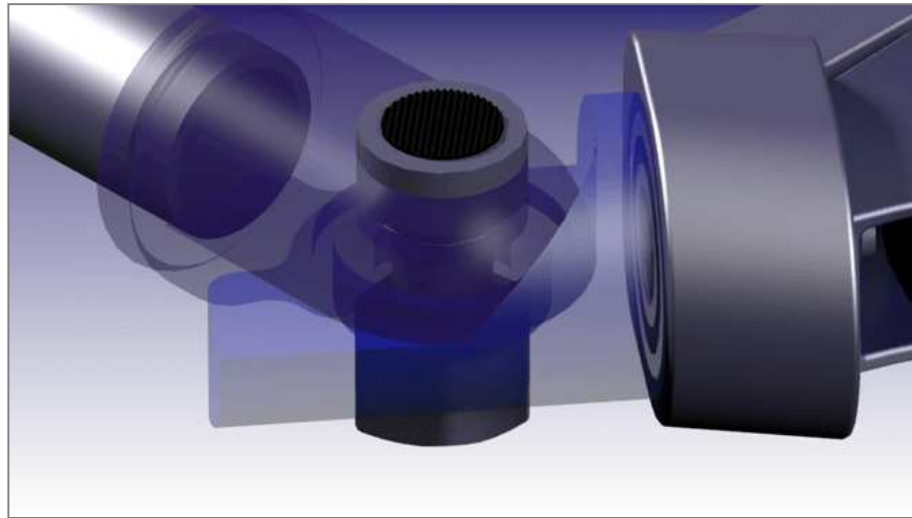
Giro de la rueda delantera. Al igual que con la rueda trasera, la rueda delantera tendrá que girar 90 grados. En la animación se resaltará la pieza soporte; que hay que accionar de la misma forma que se hizo con la horquilla trasera. Hay que tener en cuenta que este giro ha de ser en sentido contrario al de la rueda trasera, para poder más adelante, plegar la columna de dirección.

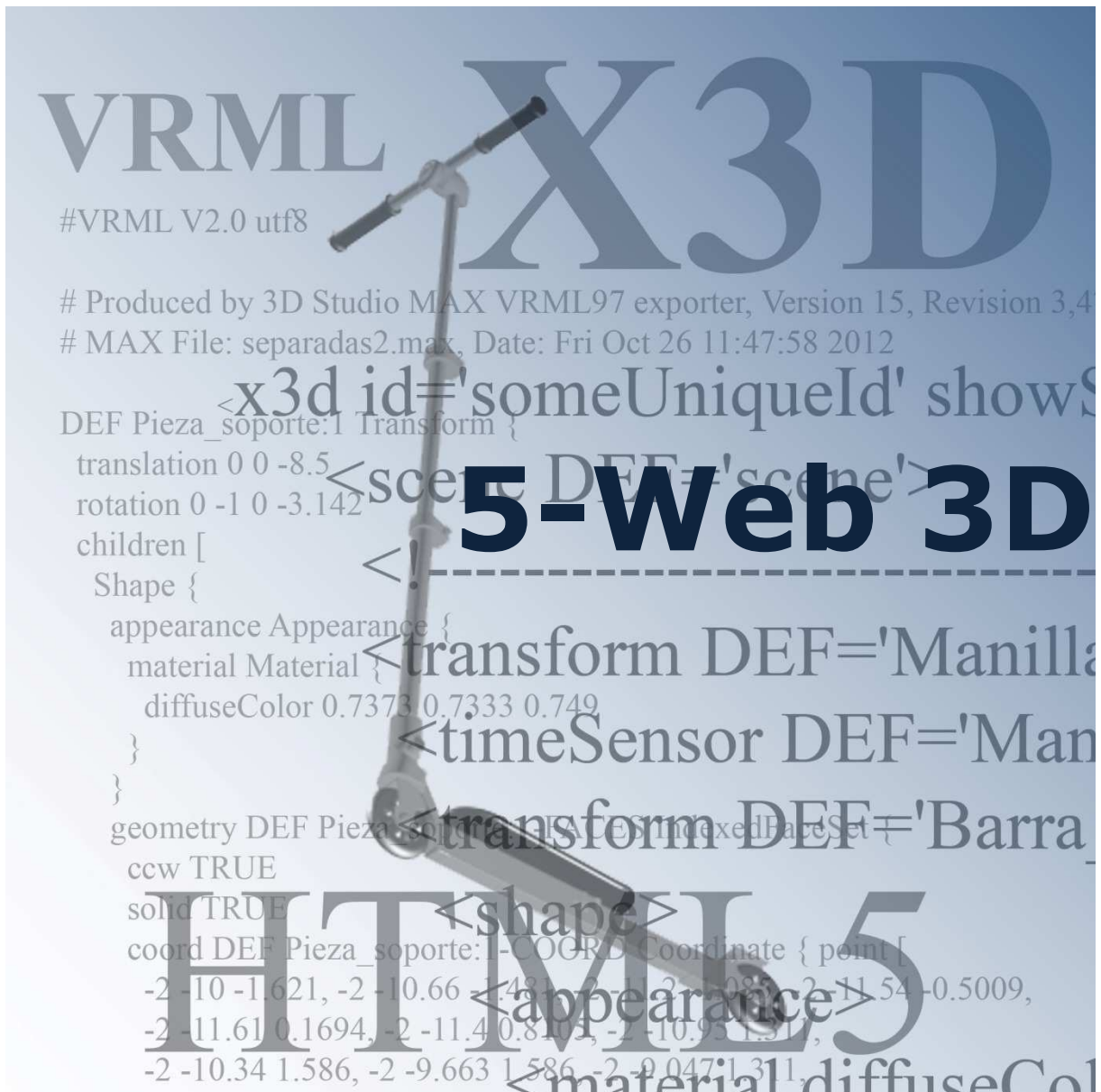




Plegado de la dirección. El último paso, será accionar el botón, resaltado en la animación, que bloquea la dirección en su posición de uso. Hecho este paso, el patinete queda totalmente plegado.







En este apartado, se hace referencia a la última versión del Lenguaje de de Hipertexto; HTML5, y las novedades que se aportan. También se habla de formatos y lenguajes compatibles con él; para la visualización de modelos 3D interactivos, como el X3D. Además de esto, se detalla el proceso de transformación de los archivos de modelado, a este formato X3D, para más adelante integrarlo en la Web.

5.1-Antecedentes

5.1.1-HTML5

El HTML5 (*HyperText Markup Language*, versión 5) es la quinta revisión del lenguaje de programación “básico” HTML, de la *World Wide Web*. Esta nueva versión pretende remplazar al actual (X)HTML, corrigiendo problemas con los que los desarrolladores web se encuentran, actualizándose a nuevas necesidades que surgen en la Web hoy en día.

Actualmente el HTML5 se encuentra en fase experimental, aunque cada vez más empresas están desarrollando sus sitios web en esta nueva versión del lenguaje. A diferencia de otras versiones de HTML, los cambios en HTML5 comienzan añadiendo semántica y accesibilidad implícitas, eliminando cualquier ambigüedad. Se tiene en cuenta el dinamismo de sitios web como redes sociales; donde su aspecto y funcionalidad se asemeja más a una aplicación, que a un documento.

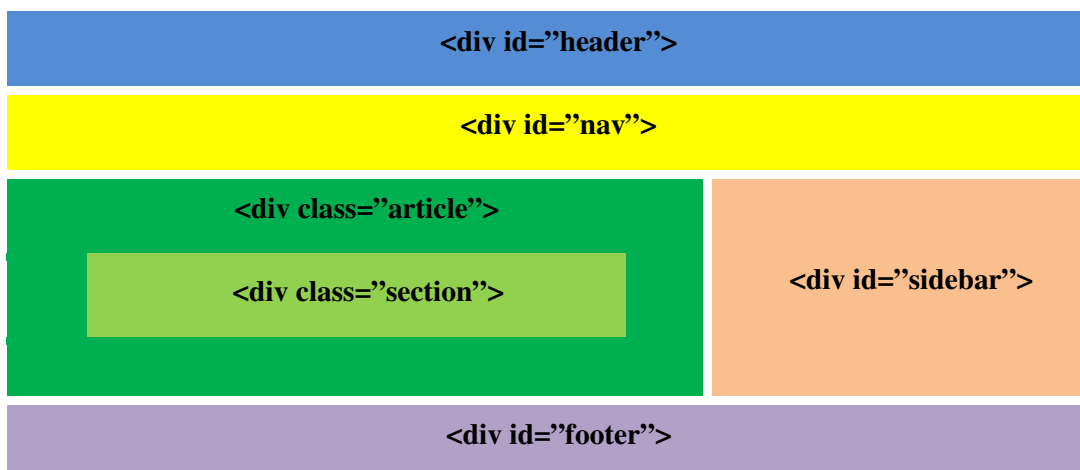
Debido a un uso abusivo de etiquetas DIV para crear una estructura en bloques, HTML5 ofrece varios elementos que perfeccionan esta estructuración determinando qué es cada sección, y eliminando así DIV innecesarios. Esto conllevará a que la estructura de la web sea más coherente y fácil de entender; dando mayor importancia a según qué secciones de la web, con lo que se facilita también la tarea de los buscadores y aplicaciones que interpretan sitios web. Entre estos nuevos elementos, destacan:

- **<article></article>**: Para representar una sección general dentro de un documento o aplicación. Puede contener subsecciones; y podemos estructurar mejor toda la página creando jerarquías del contenido con etiquetas de encabezado h1-h6.
- **<section></section>**: Artículo independiente de contenido. Composición autónoma que pueda ser reutilizado y repetido. Pueden ir anidados.
- **<aside></aside>**: Se le puede considerar un contenido independiente. Puede utilizarse para efectos tipográficos, barras laterales, elementos publicitarios, para grupos de elementos de la navegación, u otro contenido que se considere separado del contenido principal de la página.
- **<header></header>**: Cabecera del sitio. Grupo de artículos introductorios o de navegación.

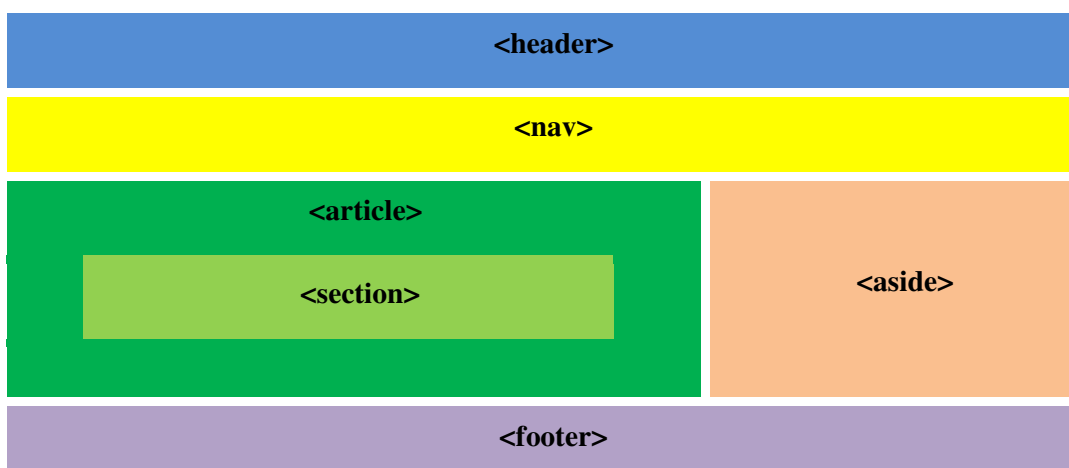
- `<nav></nav>`: Sección con links de navegación. Aparecerán enlaces que consisten en bloques principales de la navegación.
- `<footer></footer>`: Representa el pie de página, y puede contener información acerca del autor, copyright, etc.
-

DIFERENCIAS ENTRE HTML Y HTML5

HTML



HTML5



ELEMENTOS INTERESANTES:

- **<audio> y <video>:** Nuevos elementos que permitirán insertar contenido multimedia de sonido o de vídeo, respectivamente. Es una de las novedades más importantes e interesantes en este HTML5, ya que permite reproducir y controlar vídeos y audio sin necesidad de plugins como el de Adobe Flash o QuickTime.

El elemento <video> ha sido diseñado para utilizarlo sin la necesidad de que tenga que detectar ningún script. Se pueden especificar múltiples ficheros de video y los navegadores que soporten el video en HTML5 escogerán uno basado en el formato que soporte.

Formatos de video

Los formatos de video son como los lenguajes escritos. Cada navegador leerá un tipo de “lenguaje”. Los lenguajes de los videos se llaman “codecs”; un algoritmo utilizado para compactar un video.

Existen docenas de codecs aunque destacan dos, principalmente. Uno de ellos requiere el pago de licencia, y funciona en Safari y los I-Phones. El otro es gratuito y de código abierto, funcionando en navegadores como Chrome y Firefox.

- **<embed>:** Se emplea para contenido insertado que necesite plugins como el Flash. Los navegadores ya reconocen este elemento; pero ahora al formar parte de un estándar, no habrá conflicto con <object>.
- **<canvas>:** Este es un elemento complejo que permite generar gráficos dibujando en su interior. Es utilizado por ejemplo en Google Maps y en un futuro permitirá a los desarrolladores crear aplicaciones muy interesantes.

HTML 5 define el elemento <canvas> como un rectángulo donde se puede utilizar Java Script para dibujar. También determina un grupo de funciones (canvas API) para dibujar formas, así como para crear gradientes y transformaciones.

Texto Canvas

Aunque el navegador las API (*Application Programming Interface*; en español *Interfaz de Programación de Aplicaciones*) de canvas, no quiere decir que pueda soportar las API para

texto-canvas. Las API de texto se han añadido posteriormente, por lo que es posible que algunos navegadores no tengan integrado las API para texto.

Geolocalización

Estima la posición geográfica en un determinado momento.

MEJORAS EN FORMULARIOS:

Como norma para todos, un formulario es una etiqueta `<form>` y en su interior algunos elementos `<input type="text">` o `<input type="password">` finalizado con un botón `<input type="submit">` .

Se incorporan diversos elementos a “type”, con lo que la etiqueta `<input>` adquiere gran relevancia, lo que nos permitirá tener campos “inteligentes” en los formularios.

- `<input type="search">` para cajas de búsqueda.
- `<input type="number">` para adicionar o restar números mediante botones.
- `<input type="range">` para seleccionar un valor entre dos valores predeterminados.
- `<input type="color">` seleccionar un color.
- `<input type="tel">` números telefónicos.
- `<input type="url">` direcciones web.
- `<input type="email">` direcciones de email.
- `<input type="date">` para seleccionar un día en un calendario.
- `<input type="month">` para meses.
- `<input type="week">` para semanas.
- `<input type="time">` para fechas.
- `<input type="datetime">` para una fecha exacta, absoluta y tiempo.
- `<input type="datetime-local">` para fechas locales y frecuencia.

Placeholder

Se utiliza dentro de los campos input. Con él, se mostrará un texto dentro del input siempre y cuando el campo esté vacío o no esté señalado. En cuanto se haga clic dentro del campo; o se llegue por tabulador, el texto preestablecido desaparecerá.

Firefox no da soporte a esta propiedad, al igual que Internet Explorer y Opera; sólo es compatible, a día de hoy, con Safari y Chrome. Aquellos navegadores que no soporten placeholder simplemente lo ignorarán y no mostrarán nada.

Aquí hay un ejemplo de cómo se puede incluir placeholder en un formulario:

```
<form>
<input placeholder="Buscar en la base de datos">
<input value="Buscar">
</form>
```

Autofocus

El atributo de autofocus permite al usuario decidir y controlar qué campo de texto debe ser enfocado (señalado, activado) en cuanto la página es cargada o se esté cargando. No deberá haber más de un elemento con autofocus en la página.

HTML5 introduce el atributo de control de autofocus en los formularios. En cuanto la web se comienza a cargar, dicho atributo mueve el cursor y así la atención del usuario a un campo <input> en particular.

A día de hoy, autofocus sólo lo soportan Safari, Chrome y Opera; Firefox e Internet Explorer, lo ignoran.

```
<form>
<input name="b" autofocus>
<input type="submit" value="Search">
</form>
```

MODERNIZR

Como HTML5 no está implantado del todo, existen navegadores antiguos que no lo soportan. Por ello, podremos detectar si los navegadores soportan cada uno de los elementos por separado.

Los navegadores, al renderizar la página web, construyen un “**Modelo de Objeto de Documento**” (*Document Object Model - DOM*). Esto es una colección de objetos que representan los elementos del HTML en la página. Cada elemento - `<p>`, `<div>`, `` - es representado en el **DOM** por un objeto diferente.

Modernizr es una librería de *JavaScript* con licencia MIT (MIT Media Lab; laboratorio dentro de la Escuela de Arquitectura y Planificación en el Instituto de Tecnología de Massachusetts), que sirve para detectar si muchos de los elementos son compatibles con HTML5 y CSS3.

Dicha librería se irá actualizando y para utilizarla solo hay que incluir el siguiente código en el `<head><script>` de tu página:

```
<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Dive Into HTML5</title>

<script src="modernizr.min.js"></script>

</head>

<body>

...

</body>

</html>
```

APLICACIONES OFFLINE

Gracias al HTML5 se puede crear una aplicación web que funcione offline (sin necesidad de conexión a Internet).

Las aplicaciones web offline se ejecutan como una aplicación online. La primera vez que se visita una web offline que esté disponible, el servidor web le dirá a al navegador los ficheros que necesita

para poder trabajar desconectado. Estos ficheros pueden ser, HTML, JavaScript, imágenes y hasta videos. Una vez que el navegador ha descargado los ficheros necesarios podrás volver a visitar la web aunque no estés conectado a Internet. El navegador reconocerá que estás desconectado y utilizará los ficheros que había descargado con anterioridad. La próxima vez que te conectes, si has realizado cambios en la web offline, estos se subirán al servidor actualizándolo.

5.1.2-X3DOM

Además de las imágenes y vídeos exportados de 3d Studio, en el propósito de realizar una presentación de producto completa y visual; se busca poder incluir un modelo 3D, que pueda ser manipulado por el usuario. De esta forma, el usuario interactúa con el producto; pudiendo resaltar los aspectos que estime oportunos.

En concreto se buscan dos situaciones:

- La posibilidad de tener un elemento 3D que se pueda mover y rotar.
- Incorporar interactividad con el usuario, tal como poder cambiar el color de ciertos elementos del modelo. De esta forma, el usuario puede hacer una simulación, y configurar el producto a su gusto.

Anteriormente, esto era posible gracias a *Flash* y programación *JavaScript*.

Hoy en día, HTML5 incorpora el elemento *canvas*; un elemento complejo donde se puede utilizar Java Script para dibujar, generar gráficos, dibujar formas, así como para crear gradientes y transformaciones. Además es capaz de renderizar en los navegadores más importantes (Mozilla, Chrome, Opera, Safari e Internet Explorer) elementos 3D.

W3b3D

Web3D surgió inicialmente como la idea de visualizar y navegar plenamente sitios web con 3D. Ahora, se refiere a todo el contenido 3D interactivo que se incrusta en páginas web html que podemos ver en un navegador. Por lo general, la tecnología Web3D, requiere la instalación de un visor o plugin, para ver el contenido.

Hoy en día, hay disponibles diversos formatos y herramientas. En la búsqueda de documentación, se encuentra que WebGL y X3DOM, son las plataformas más comunes en la investigación de Web 3D.

WebGL

WebGL es una especificación estándar en desarrollo, para mostrar gráficos en 3D en navegadores web. El WebGL permite mostrar gráficos en 3D acelerados por hardware (GPU) en páginas web, sin la necesidad de plug-ins en cualquier plataforma que soporte OpenGL 2.0 u OpenGL ES 2.0. Técnicamente es un API (*Interfaz de programación de aplicaciones*) para javascript que permite usar implementación nativa de OpenGL ES 2.0 que se incorporará en los navegadores. WebGL utiliza el elemento Canvas del HTML.

OpenGL (*Open Graphics Library*) es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. Puede usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples. Se usa ampliamente en CAD, realidad virtual, representación científica, visualización de información y simulación de vuelo, así como en desarrollo de videojuegos, donde compite con *Direct3D* en plataformas Microsoft Windows.

OpenGL ES (*Open Graphics Library for Embedded Systems*) es una variante simplificada de OpenGL diseñada para dispositivos integrados tales como teléfonos móviles, PDAs y consolas de videojuegos. La define y promueve el Grupo Khronos, un consorcio de empresas dedicadas a hardware y software gráfico interesadas en APIs gráficas y multimedia. Khronos gestiona también WebGL.

WebGL creció desde los experimentos del canvas 3D comenzados por Mozilla. El cual, en 2006, mostró un primer prototip, y a finales de 2007, tanto Mozilla como Opera, habían hecho sus propias implementaciones por separado. A principios de 2009 Mozilla y Khronos formaron el *WebGL Working Group* (*Grupo de Trabajo del WebGL*).

La mayoría de navegadores son miembros del WebGL Working Group, y ya está presente en Mozilla Firefox, Mozilla Fennec, Google Chrome y también en la versión 5.1 de Safari incorporada en OS X Lion.

X3DOM

Optaremos por X3DOM, dado que ofrece múltiples posibilidades para el fin que queremos conseguir.

Podremos llevar a cabo la transformación de nuestro modelo 3D generado en *Autodesk 3ds MaxDesign*, a un modelo 3D interactivo, de forma relativamente sencilla. En la página web www.x3dom.org, encontraremos la documentación y recursos necesarios para el proceso de transformación de nuestro modelo.

X3DOM se define como una estructura experimental de código abierto y tiempo de ejecución, para la integración de HTML5 y contenido 3D declarativo.

X3DOM (pronunciado *X-Freedom*) es una estructura experimental de código abierto y tiempo de ejecución, para apoyar el debate que está en marcha en la Web3D y las comunidades del W3C (*World Wide Web Consortium*), sobre cómo podría ser la integración de HTML5 y contenido 3D declarativo. Trata de cumplir las actuales especificaciones de HTML5 para contenido 3D, y permite incluir elementos X3D como parte de cualquier árbol del DOM de HTML5.

El objetivo aquí es tener una escena “viva” de X3D en el DOM del HTML, lo que permite manipular el contenido 3D con sólo añadir, quitar o cambiar los elementos del DOM. Sin interfaz de plugin o complemento específico (como el SAI –*Scene Access Interface*- específico de X3D) necesario. También es compatible con la mayoría de los eventos de HTML en objetos 3D. El modelo de integración total aún está en evolución y abierto a debate.

Se espera iniciar un proceso similar a cómo evolucionó el SVG (*Scalable Vector Graphics*) en HTML5:

- Proporcionar una visión y un tiempo de ejecución con el que experimentar y desarrollar un modelo de integración para 3D declarativo en HTML5
- Conseguir que el debate en las comunidades de HTML5 y X3D continúe y evolucione el sistema y modelo de integración.

- Por último, sería parte del estándar HTML5 y contaría con total apoyo de todos los principales navegadores de forma nativa.

Más información arquitectónica y de fondo se puede encontrar en los periódicos X3DOM (publicado en las conferencias anuales Web3D 2009-2012):

- X3DOM: un modelo basado en DOM integración HTML5/X3D
- Una arquitectura escalable para la integración X3DOM modelo HTML5/X3D
- Aspectos dinámicos e interactivos de X3DOM
- Uso de imágenes y contenedores Binary explícita para la entrega eficiente e incremental de declarativas escenas en 3D en la Web

Hay también dos artículos sobre el proyecto en la revista alemana iX, una en la edición de noviembre de 2010 y una en la edición de diciembre de 2010 (Fuentes de información en el apartado **8.5-Web 3D**).

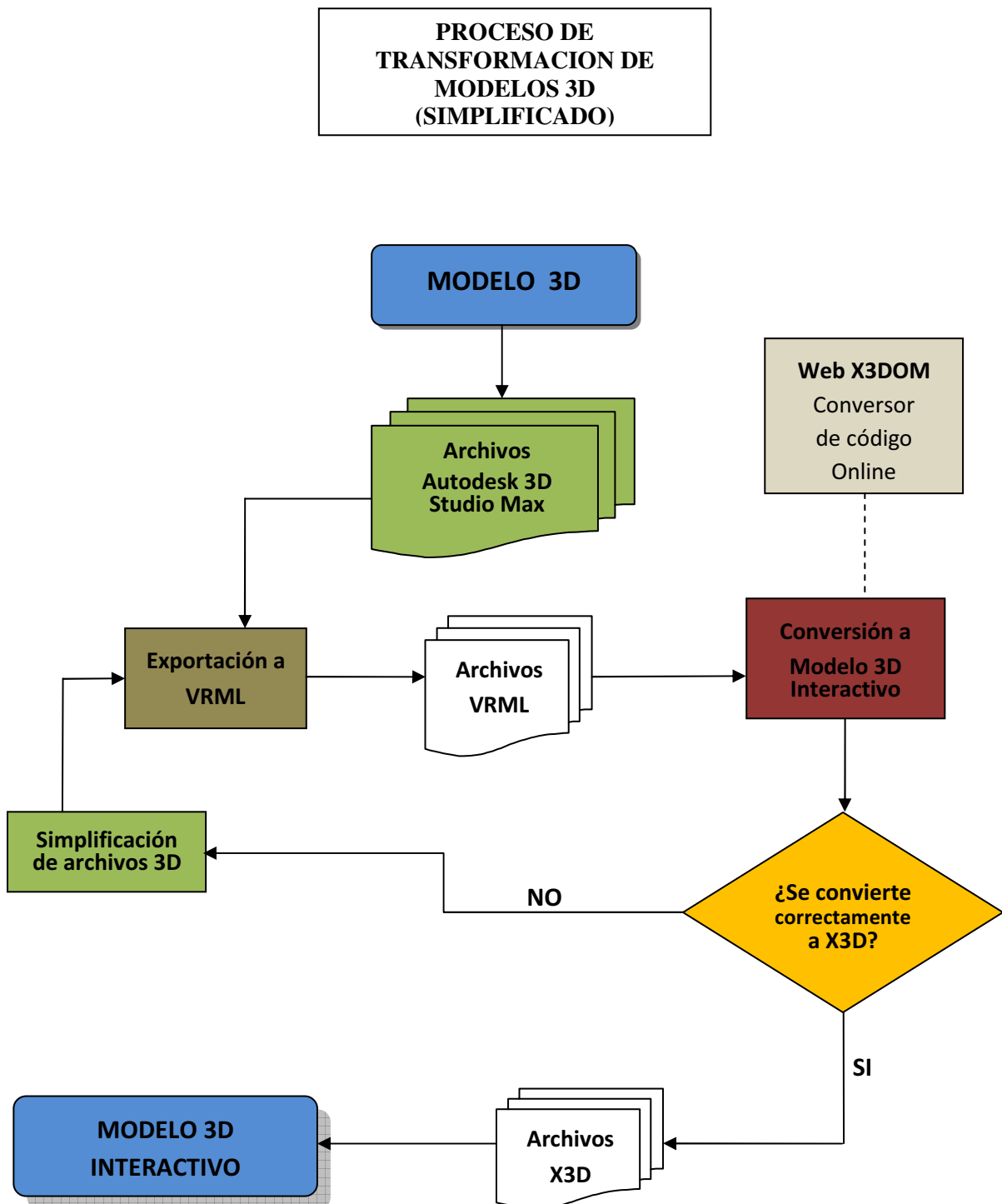
Alternativamente, usted, como desarrollador web, también puede simplemente utilizar el actual sistema para crear páginas web y aplicaciones, que incluyen contenido (X)3D declarativo que se representarán, aceleradas por hardware (gracias a WebGL), sin la necesidad de utilizar ningún plugin.

5.2-Proceso de transformación de Modelo 3D (Fichero 3ds) a Modelo 3D Interactivo (X3D)

En este apartado veremos el proceso general para poder pasar, de un archivo de modelado 3D, a un modelo que podamos visualizar y manipular en la Web. Para ello se necesitará seguir un proceso de exportación y transformación de archivos.

5.2.1-Diagrama

En el siguiente diagrama de flujo, se describe el proceso de transformación de forma simplificada.



5.2.2-Descripción del proceso

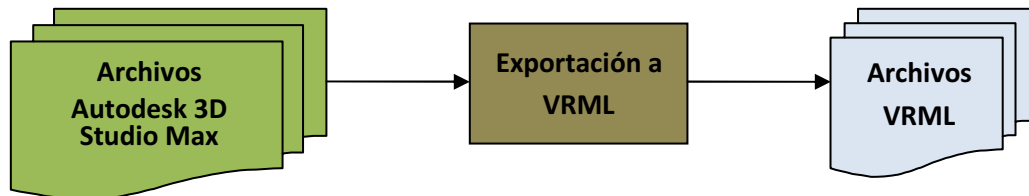
MODELO 3D

Al ser el proceso de transformación del modelo de *3ds Max* a 3D interactivo, como entrada del diagrama tendremos el **modelo 3D**.



Como hemos visto anteriormente, en esta fase del proceso, necesitaremos trabajar con los archivos de *Autodesk 3ds Max Design 2013*. Montaremos archivos en los que aparezcan únicamente el producto o alguna parte de éste; tales como despieces. La escena ha de ser sencilla; por ello, no se tendrán en cuenta materiales ni iluminación.

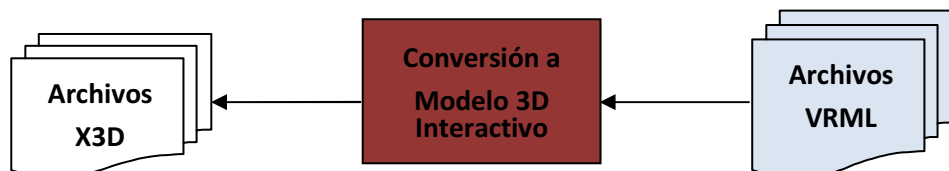
En las opciones de exportación de *Autodesk 3ds Max*, escogeremos **VRML** (esto es archivos con extensión **.WRL**). Son los que necesitaremos a continuación.



Web X3DOM
 Conversor
 de código
 Online

Como ya sabemos, nos apoyaremos en la página www.x3dom.org, que cuenta con un conversor online de código para tal fin.

Para obtener un modelo 3D interactivo, nos serviremos de los archivos VRML, exportados anteriormente. Tal proceso, nos devolverá como resultado, los archivos **X3D** necesarios.



La entrada del conversor, será el archivo VRML que hemos exportado del archivo modelado en *Autodesk 3ds Max*; y como resultado, nos devolverá un archivo X3D, el cual ya se puede visualizar en el navegador.



**Simplificación
de archivos 3D**

En caso de que la conversión **no** se llegue a realizar correctamente, habrá que llevar a cabo algunas modificaciones en el modelo 3D; es decir, en los archivos de *Autodesk 3ds Max*, con el fin de simplificarlos. Una vez simplificados, se realizará de nuevo el proceso hasta conseguir los archivos X3D necesarios. Las particularidades del proceso y problemas encontrados, se describen más adelante.



**MODELO 3D
INTERACTIVO**

Como elemento de salida de este diagrama de flujo; y a su vez resultado del proceso de transformación de archivos que en él se representa, encontramos el **Modelo 3D Interactivo**. Éste, en formato X3D, se integrará después en una página HTML5.

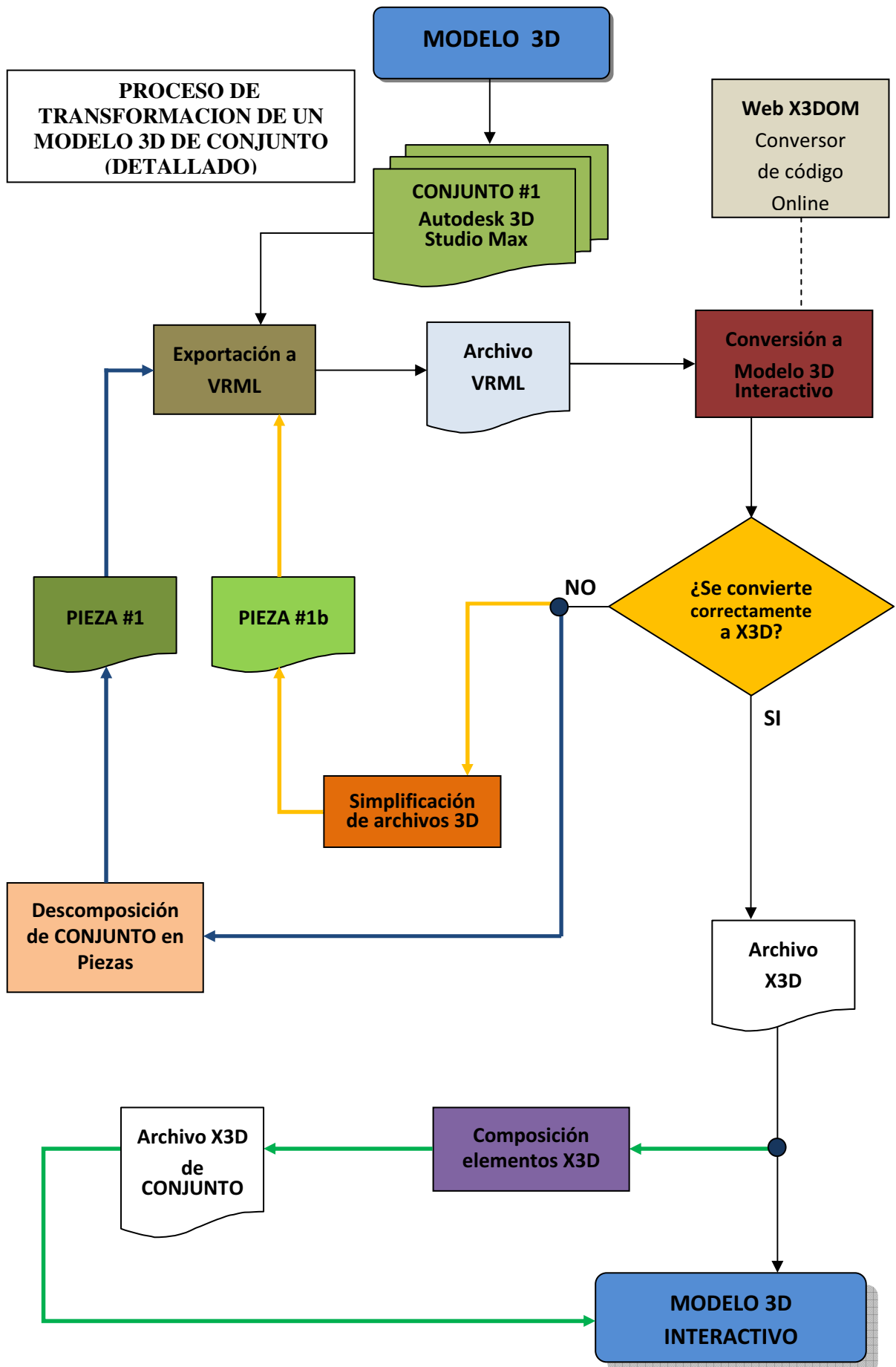
El proceso que se representa en el diagrama, se refiere a archivos *3ds Max* en general. Da una idea de los pasos a seguir para obtener un modelo interactivo que podamos integrar en una página web, más adelante.

5.3-Proceso detallado: Problemas y particularidades

En este apartado, veremos el proceso de transformación de archivos a X3D, de forma más detallada y teniendo en cuenta los casos especiales y problemas que pueden surgir.

5.3.1-Diagrama

Representación gráfica del proceso de transformación de archivos; con los posibles problemas que pueden surgir y la forma de corregirlos; recorriendo el gráfico por distintas “rutas”.



5.3.2-Descripción del proceso

En el apartado anterior, hemos visto el proceso de transformación general para archivos *Autodesk 3ds Max*, en archivos X3D. El proceso que se representa en el diagrama detallado, se refiere a archivos *3ds Max* de conjunto. Podemos observar que el proceso es similar a la versión simplificada; descrita anteriormente, por lo que obviaremos los primeros pasos.

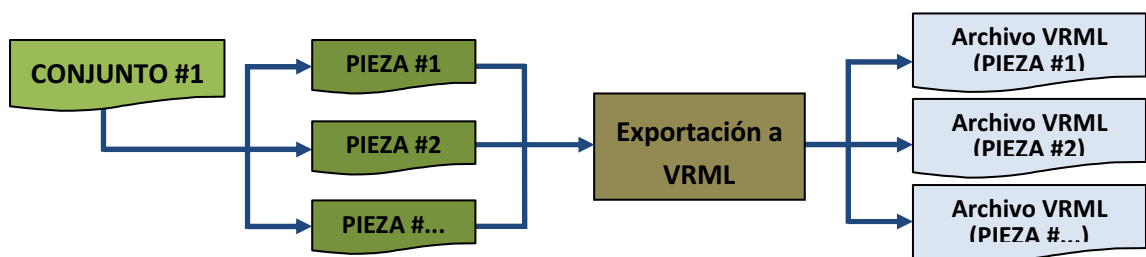
La diferencia la encontramos al observar cómo se comporta el archivo **VRML**, al tratar de transformarlo a **X3D**, con el conversor de la página www.x3dom.org.

NO se convierte correctamente a X3D. En caso de que la conversión de archivos VRML no sea satisfactoria, habrá que simplificar los archivos como se explicó antes. Tendremos que actuar de la siguiente forma:

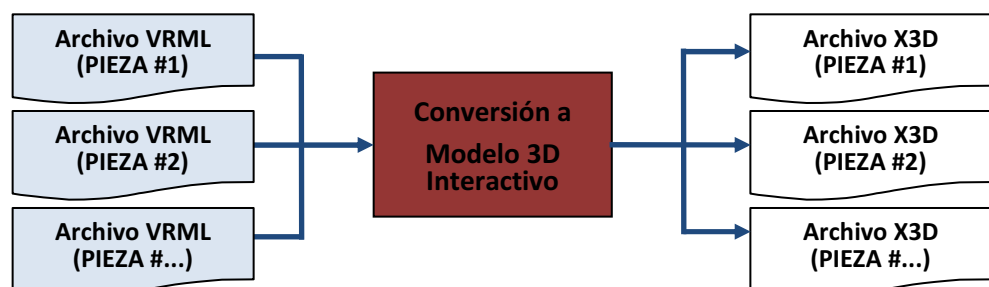
PASO1:

Descomposición de CONJUNTO en Piezas

Si en el caso de que el archivo VRML; exportado de un archivo de *3ds Max* de conjunto, no se convierta satisfactoriamente a X3D, tendremos que dividir este archivo de conjunto en tantos archivos individuales como piezas formen el conjunto. Una vez descompuesto el conjunto en piezas, habrá que exportar un nuevo archivo VRML para cada una de ellas. Esto es:



Habrà que transformar ahora cada uno de esos archivos en X3D.



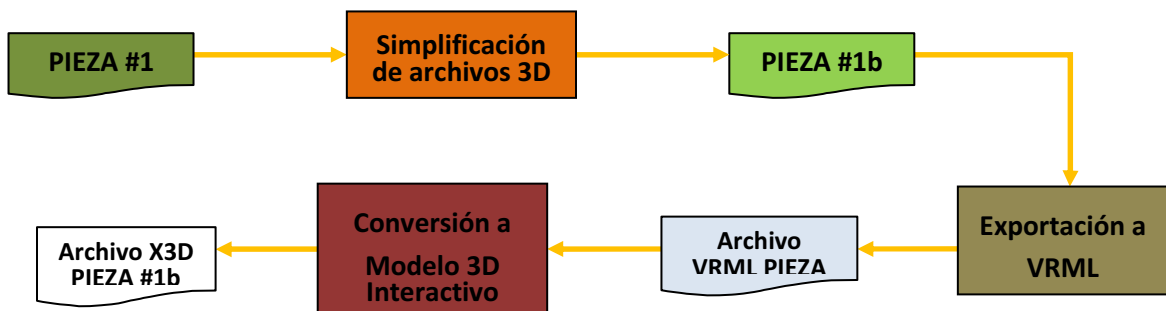
NO se convierte correctamente a X3D. En caso de que la conversión de archivos VRML no sea satisfactoria, habrá que simplificar nuevamente los archivos. Esta vez actuaremos así:

PASO2:

Simplificación de archivos 3D

Si habiendo descompuesto el archivo de conjunto en piezas; y exportando éstas a VRML, aún no se convierte correctamente a X3D, habrá que modificar de alguna forma el archivo de 3ds Max de aquella o aquellas piezas que den problemas. Esto supondrá un cambio en la geometría de la pieza, simplificándola en la medida de lo posible.

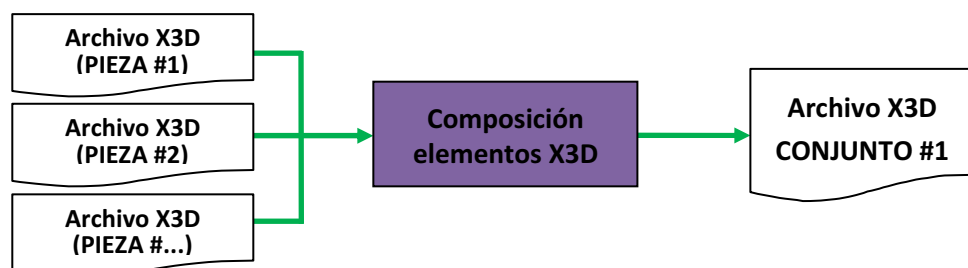
Una vez hecho esto, se seguirá el proceso de la misma forma que antes:



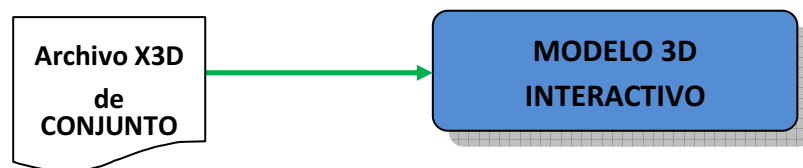
Este proceso se realizará tantas veces sea necesario para conseguir el archivo X3D necesario.

COMPOSICION DE ELEMENTOS X3D

Suponemos que partimos de un archivo de conjunto. Si la transformación a X3D; del archivo VRML exportado, se lleva cabo con éxito, el proceso de transformación del modelo termina aquí. En caso contrario, tanto si hay que hacer el **PASO 1**; como si también hay que hacer el **PASO 2**, lo cual es más desfavorable y complica aún más el proceso, tendremos que continuar de la siguiente manera:



Esto significa que debemos hacer el “**proceso inverso**”; es decir que tendremos que montar el conjunto de nuevo, esta vez a través del código X3D. Este nuevo archivo, con las correcciones necesarias, ya podrá visualizarse correctamente en la web. De esta forma obtenemos el **Modelo 3D Interactivo**; elemento de salida de este diagrama de flujo; y a su vez resultado del proceso de transformación de archivos. Éste, en formato X3D, se integrará después en una página HTML5.



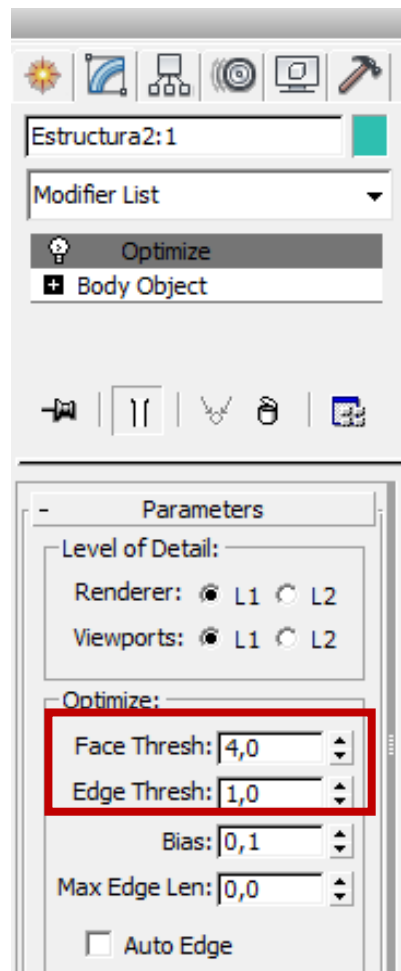
El proceso de composición de archivos X3D de conjunto, se llevará a cabo tantas veces como sea necesario. Esto supone que habrá que realizarlo tantas veces como haya modelos 3D de conjunto que no se puedan exportar y transformar correctamente.

5.3.3-Transformación de archivos en la práctica

Principalmente queremos una representación completa del patinete, por lo que lo ideal sería poder exportar y posteriormente transformar, todo el patinete de una sola vez. En primer lugar, exportaremos a VRML un archivo de *3ds Max* en el que aparece únicamente el patinete. La exportación a VRML no supone ningún problema. Es en el momento de introducir el código de VRML en el convertidor online, cuando aparecen dificultades. **No se convierte correctamente a X3D.**

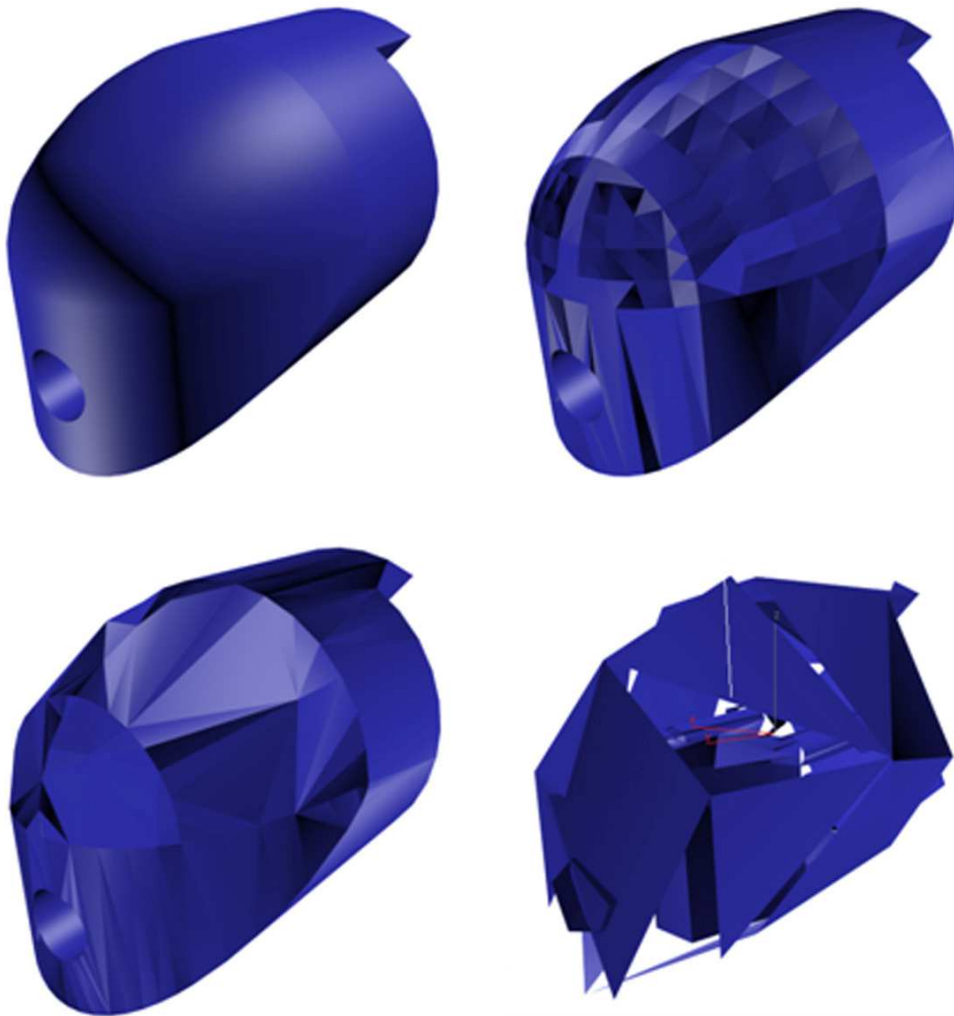
1. En un primer momento, se intenta resolver el problema, simplificando el modelo. Esto se hará dividiendo el modelo general en subconjuntos.
2. Repetimos la operación de exportación a VRML sin ningún problema. Pero en ocasiones, esos subconjuntos tampoco se convierten correctamente a X3D. Con lo cual se prueban otras soluciones.
3. Se intenta solucionarlo esta vez, simplificando la geometría de algunas de las piezas del conjunto que da problemas a la hora de convertir a X3D. Esto se lleva a cabo por medio de un modificador; en el panel *Modify* se escoge el modificador *Optimize*. Con ello, podemos

ajustar el número de caras y aristas de las piezas del conjunto, con el objetivo de simplificarlo.



Esto, no dará buen resultado. Encontraremos que para conseguir que se convierta a X3D, habrá que simplificar mucho la geometría de algunas piezas; lo que dará lugar por una parte a superficies con demasiadas caras; y en ocasiones, a ausencia de geometría y vacíos en algunas zonas de las piezas.

Al decir que las piezas tienen muchas caras, lo que en realidad ocurre es lo contrario. Se reduce el número de caras o polígonos que forman la superficie de la pieza; por lo que visualmente, parece que hay muchas caras en vez de una sola superficie lisa. Reduciendo mucho el número de polígonos o aristas, empezarán a producirse deformaciones y a desaparecer geometría.



Como no todos los elementos necesitarán el mismo grado de optimización, y puesto que este proceso sería de prueba / error, se descarta por lo tedioso que resultaría, así por no poder obtener un resultado satisfactorio y suficientemente uniforme para todas las piezas. Para poder convertir correctamente el conjunto, unas piezas quedarían muy irregulares mientras que otras prácticamente no necesitarían simplificación.

4. A continuación se prueba exportando grupos con menor número de piezas. Pero esta solución se descarta rápidamente ya que no se llega a una solución “lógica”. Vemos que no influye el número de piezas, pues en ciertos casos se convierten grupos con mayor número de piezas que otros, y geometrías similares en cuanto a la complejidad de superficies. No podemos establecer una “regla”.

5. Después de numerosas pruebas y puesto que no podemos llegar a ninguna conclusión de porqué se convierten o no; y de qué depende que esto suceda, actuamos de la siguiente forma:

Exportamos el subconjunto a VRML. Si no se convierte a X3D, descomponemos el conjunto en cada una de sus piezas, o de dos en dos piezas, lo que no supone ningún problema para convertirlas. Y seguimos el mismo proceso con todas ellas.

6. Salvo en muy raras ocasiones, las piezas individuales sí se convierten a X3D. En caso de que esto no suceda; se procederá a simplificar dichas piezas lo mínimo posible, eliminando geometría que no sea indispensable en esa representación.

Una vez exportadas todas las piezas, tendremos que “montarlas” de nuevo en el archivo X3D. Una vez montado el patinete completo, bastará entonces, con realizar algunos pequeños cambios en cuanto a la visualización del modelo, o la configuración. Esto se detallará en el apartado **5.5-X3D**.

5.4-VRML

VRML (Virtual Reality Modeling Language) o Lenguaje de Modelado de Realidad Virtual, es un formato estándar para mostrar modelos 3D en la Web. La versión actual de VRML es la VRML 2.0 o VRML97. VRML usa la extensión de archivo .wrl. Para visualizar VRML en Internet Explorer, Firefox, Chrome y otros, habrá que descargar e instalar un plugin.

En 1995, VRML se convirtió en el primer formato basado en 3D. VRML era único porque soportaba geometría 3D, animación y “scripting”. En 1997, VRML fue certificado ISO y continuó atrayendo a un gran número de seguidores entre artistas e ingenieros. VRML es el formato de soporte 3D más extendido.

En 2001, X3D se unió como una codificación XML del VRML. X3D añadió sombras, geolocalización, y otras características vanguardistas, junto a otras de soporte personalizado para usuarios en medicina, CAD, GIS (Geographic Information System), y otras áreas importantes.

5.4.1-Algunas características de VRML

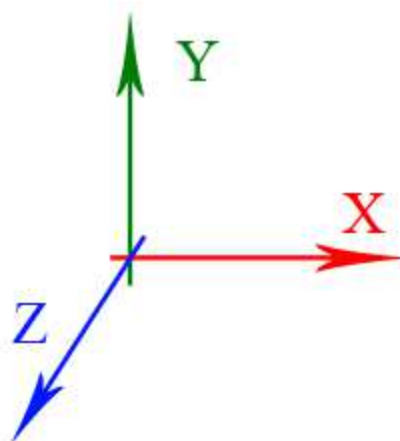
En VRML97, las escenas que se construyen se conocen comúnmente como *worlds* (mundos); de ahí que la extensión de los archivos sea *.wrl*. Todos los archivos deben incluir la cabecera: #VRML V2.0 utf8.

Se tratan de archivos de texto ASCII, por lo que pueden editarse con cualquier editor de texto. Existen editores como *V-Realm Builder*, que interpretan el código y permiten ver la escena al tiempo que se edita.

A continuación se hace una breve descripción de algunas características de las escenas VRML.

EJES Y UNIDADES

En VRML97, el eje X positivo apunta a la derecha, el eje Y positivo hacia arriba y el Z positivo apunta hacia fuera de la pantalla; como se muestra en la siguiente imagen.



VRML97 funciona mediante un sistema “local” para cada objeto que se define, y uno “global” para relacionar todos los objetos de un mismo entorno. De esta forma, si solo definimos un objeto, al que no se aplica ninguna transformación, el sistema de ejes “local”, coincidirá con el sistema de ejes “global” o del mundo.

Por convenio, las distancias lineales están en metros, los ángulos en radianes, el tiempo en segundos y los colores en RGB. Estas y otras características las veremos en el apartado **5.5.2-Particularidades de la conversión a X3D**, ya que son iguales para ambos códigos.

PRIMITIVAS

En VRML97 existen una serie de objetos predefinidos o primitivas; los cuales son:

- Box (Paralelepípedos; cubos, cajas)
- Sphere (Esfera)
- Cone (Cono)
- Cylinder (Cilindro)

Estas primitivas se definen en el nodo **Shape**, el cual tiene dos campos; **geometry** y **appearance**. El primero define la geometría del objeto 3D y el segundo, el color o textura del objeto. Este último lo veremos en el siguiente apartado más detalladamente.

<p>BOX:</p> <pre>Shape{ appearance Appearance { material Material { diffuseColor 1 0 0 } } geometry Box { size 5 2 6 } }</pre>	<p>Caja de cinco metros de ancho, dos metros de alto y seis metros de fondo. Al no aplicarse transformaciones, se encuentra en el origen de coordenadas globales.</p>
---	---

<p>SPHERE:</p> <pre>Shape{ geometry Sphere { radius 4 } }</pre>	<p>Esfera de cuatro metros de radio; centrada en el origen coordenadas.</p>
<p>CONE:</p> <pre>Shape{ geometry Cone { bottomRadius 3 height 5 } }</pre>	<p>Cono con una base de tres metros de radio y cinco metros de altura, centrado en el origen.</p>
<p>CYLINDER:</p> <pre>Shape{ geometry Cylinder { height 6 radius 3 } }</pre>	<p>Cilindro con una base de tres metros de radio y seis metros de altura, centrado en el origen de coordenadas.</p>

OBJETOS Y LINEAS

VRML97 cuenta con estructuras para dibujar geometrías arbitrarias, con las que definir la geometría de un objeto 3D a partir de sus vértices y caras. Esto se hace por medio del nodo **IndexedFaceSet**, en el campo **geometry** de un nodo **Shape**.

<pre>Shape{ geometry IndexedFaceSet { coord Coordinate { point [0.5 0.5 0.5, 0.5 0.5 -0.5, -0.5 0.5 -0.5, -0.5 0.5 0.5, 0.5 -0.5 0.5, 0.5 -0.5 -0.5, -0.5 -0.5 -0.5, -0.5 -0.5 0.5] } coorIndex [0, 1, 2, 3, -1, 0, 4, 5, -1, 1, 5, 6, 2, -1, 2, 6, 7, 3, -1, 3, 7, 4, 0, -1, 4, 7, 6, 5, -1] } } }</pre>	<p>En este ejemplo vemos un cubo, definido a partir de ocho vértices y seis caras (normalmente esto no se hace, ya que se cuenta con una primitiva a tal efecto).</p> <p>En primer lugar, se definen los vértices en el campo coord; usando el campo point, dentro del nodo Coordinate, para listar los puntos 3D que forman los vértices del objeto. A continuación se definen los polígonos que forman las caras, mediante el campo coorIndex. La lista de vértices de una cara, se separa mediante un -1. (El número mínimo de vértices por cara es tres)</p> <p>Los vértices se listan en orden antihorario; deben ser consecutivos y enlazando el primero con el último.</p>
---	---

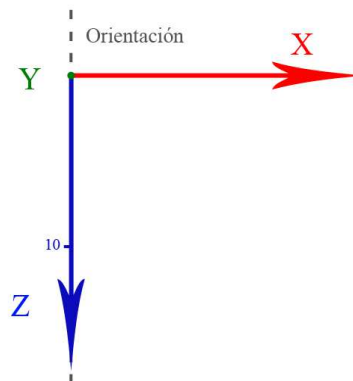
De forma similar a como se definen objetos sólidos, también se puede con objetos “alámbricos” o de “alambre”; objetos formados simplemente por aristas sin caras planas. Esto se hace análogamente, por medio del nodo **IndexedLineSet**.

PUNTO DE VISTA

Aunque las escenas de VRML97 permiten navegar libremente, existe la posibilidad de definir diversos puntos de vista. Esto se hace mediante el nodo **Viewpoint**.

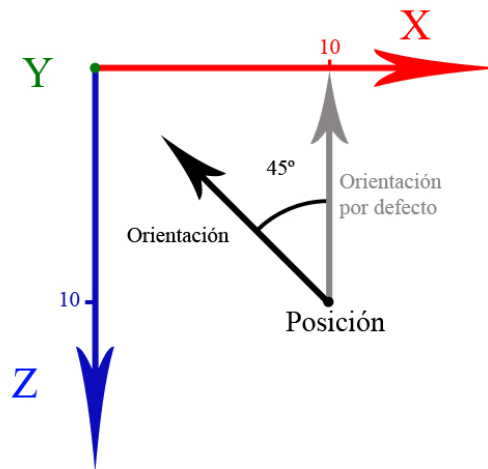
```
Viewpoint {
    position 0 0 0
    orientation 0 0 1 0
    description "PuntoDeVista_PorDefecto"
}
```

Los valores por defecto sitúan el observador sobre el eje Z positivo a diez metros del origen y mirando hacia el sentido negativo de éste; hacia el origen. El vector director de la orientación del punto de vista por defecto, es un vector unitario $[0, 0, -1]$, para X, Y, Z respectivamente. El ángulo de rotación es 0, ya que el observador está orientado hacia el origen sobre el eje Z.



En el siguiente ejemplo, se define la posición del observador a diez metros en el eje X, y diez en el eje Z; orientado hacia el centro de coordenadas. Esto supone que el vector director de la orientación en este caso, equivale a una rotación de 45 grados del vector director por defecto.

```
Viewpoint {
    position 10 0 10
    orientation 0 1 0 0.7854 ← 45 grados en radianes
    description "PuntoDeVista_Ejemplo"
}
```



Al igual que en una cámara se puede cambiar la lente, para tener un ángulo de visión más o menos amplio, VRML97 permite variar el ángulo de visión mediante el campo **fieldOfView**. Por ejemplo:

```
Viewpoint {
    fieldOfView 2.618
    description "Gran_Angular"
}
Viewpoint {
    fieldOfView 1.3962667
    description "80mm"
}
Viewpoint {
    fieldOfView 0.31416
    description "Telefoto"
}
```

5.4.2-Aspecto de los objetos en VRML: Materiales

Pondremos especial atención a la forma de definir el material de los objetos. A continuación vemos un ejemplo de código VRML:

```

#VRML V2.0 utf8
Shape {
  appearance Appearance {
    material Material { diffuseColor 1 0.8 0.8
                        ambientIntensity 0.6
                        transparency 0.5
                        emissiveColor 0 1 0
                        shininess 0.6
                        specularColor 1 1 0.0 }
    }
  geometry Cone { bottomRadius 2
                  height 4 }
}

```

NODO SHAPE (FORMA)

Cómo hemos visto, todos los objetos visibles están definidos en el nodo **Shape**. Este nodo tiene dos campos **appearance** y **geometry**.

El campo **appearance** especifica un nodo **Appearance**, el cual se usa para definir el color textura y demás, para aplicarlo a la geometría. El campo **geometry**, indica qué forma se dibujará.

NODO APPEARANCE (APARIENCIA)

El nodo **Appearance** define el aspecto de la geometría, y sólo podrá ir dentro del nodo **Shape**.

Los campos incluidos en el nodo son **material**, **texture**, y **textureTransform**. Todos los campos son opcionales, pero hay que especificar al menos uno.

El campo **material** contiene el nodo **Material**. Este nodo especifica el color de la geometría asociada, y cómo la geometría refleja la luz.

El campo **texture** contiene uno de los nodos de textura compatibles: **ImageTexture**, **MovieTexture**, o **PixelTexture**. Si este campo está vacío o no existe, no se aplicará ninguna textura a la geometría.

El campo **textureTransform** contiene el nodo **TextureTransform**. Este nodo especifica cómo se aplica la textura a la geometría.

NODO MATERIAL

El nodo **Material** especifica el color, reflexión de la luz y transparencia. Este nodo sólo se puede definir dentro de un nodo **Appearance**.

Este nodo tiene seis campos: **diffuseColor**, **emissiveColor**, **ambientIntensity**, **shininess**, **specularColor**, y **transparency**.

El campo **diffuseColor**, define el color de la geometría. Este campo se ignora si se usan texturas.

El campo **emissiveColor** se usa para definir objetos que brillan por sí mismos; la figura representada parece estar iluminada desde el interior y brillar en la oscuridad.

El campo **ambientIntensity** especifica la cantidad de luz que refleja la geometría.

El campo **specularColor** define el color de las zonas brillantes de la geometría.

El campo **shininess** controla la intensidad de la luz de los puntos brillantes; pequeños valores representan brillos suaves, mientras que los valores altos definen zonas destacadas más pequeñas y nítidas.

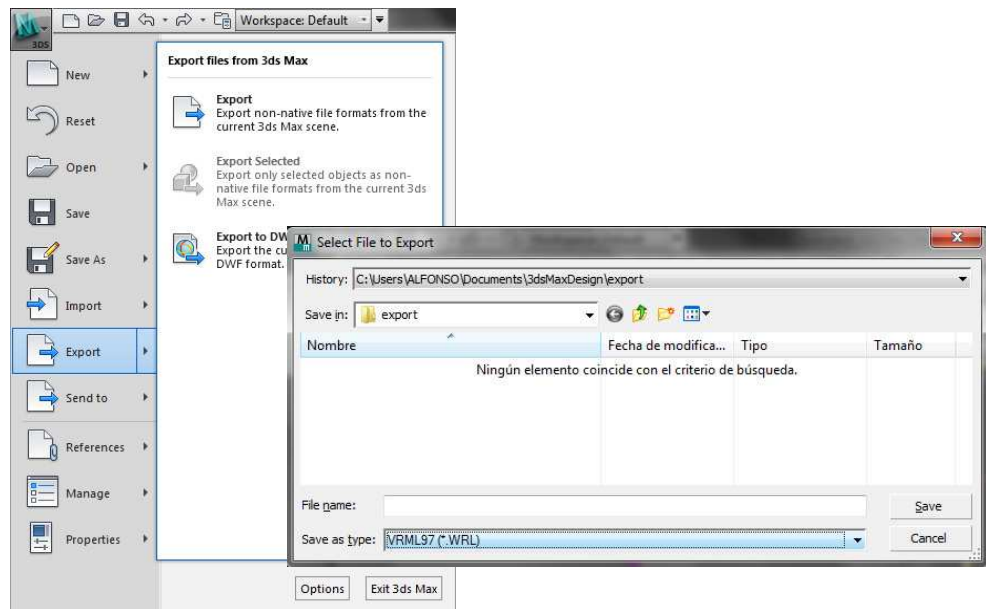
El campo **transparency** controla la transparencia de la geometría asociada. Un valor de 0.0 supone que la geometría es totalmente opaca, y un valor de 1 significa que la geometría es invisible.

Todos los campos de “Color” tienen un valor RGB asociado; es decir, tres valores entre 0,0 y 1,0. Los otros campos tienen un valor único entre 0.0 y 1.0.

5.4.3-Transformación de archivos 3ds Max a VRML

Directamente de *3ds Max*, exportaremos a un archivo VRML, es decir con extensión .WRL. Esto se hará en Archivo > Exportar. Nos guardará un archivo que podremos abrir con el bloc de notas o un

editor de código; para más tarde, copiarlo y pegarlo en el conversor de código con el fin de transformarlo en X3D.



Se trata de un archivo en el que está codificada toda la información relativa al archivo de origen, del modelo 3D que se exporta.

```

Barra_Direccion_1: Bloc de notas
Archivo Edición Formato Ver Ayuda
#VRML V2.0 utf8
# Produced by 3D studio MAX VRML97 exporte
# MAX File: Barra_Direccion_1.max, Date: T

DEF barra_direccion01 Transform {
  translation -13 370.5 294.9
  rotation 0.0444 -0.998 0.0444 -1.573
  children [
    shape {
      appearance Appearance {
        material Material {
          diffuseColor 0.1059 0.6941 0.105
        }
      }
      geometry DEF barra_direccion01-FACES
        ccw TRUE
        solid TRUE
        coord DEF barra_direccion01-COORD
          -6 159 -13.75, -6 159 -10.97, -6
            4.601 159 -11.62, -4.601 159 -11
              6 159 -13.75, 6 159 -10.97, 5.52
                -5.521 159 -13.95, -2.275 159 -1
                  0 159 -12.5, 0 159 -15, 2.275 15
                    6 151 -13.75, 6 151 -10.97, 6 15
                      -4.601 151 -11.62, 4.601 151 -11

```

Información sobre la versión, fecha de creación, programa desde el que se ha exportado el archivo y nombre del archivo de origen (*3ds Max*).

Información sobre el objeto; nombre, coordenadas de posición y rotación, y material.

Información de la geometría; coordenadas de todas las caras y vértices de la pieza.

Estos archivos no son los que necesitamos; sino que servirán como transición entre el modelo que hemos creado en *3ds Max* y el modelo 3d interactivo que queremos conseguir.

Accedemos al conversor de código online a través de la página web www.x3dom.org (http://doc.instantreality.org/tools/x3d_encoding_converter/). El conversor nos pide un código de entrada; que en este caso será el código VRML, y nos devolverá uno de salida, que puede ser X3D o una página en HTML5 con el código X3D ya integrado.

X3D encoding converter

1 | choose input encoding
Classic encoding (VRML97) ▼

2 | paste input code
SE COPIA EL CODIGO
VRML Y SE PEGA AQUI

3 | choose output encoding
XML encoding (X3D) ▼
Convert encoding Reset

4 | encoded output

Código de entrada (VRML)

Código de salida

En las opciones de salida podemos elegir:

XML encoding (X3D) → Código X3D. Se apoya en referencias externas de www.web3d.org

HTML5 encoded webpage (x3dom html5)

XHTML5 encoded webpage (x3dom xhtml5)

Código X3D integrado en HTML5 o XHTML5, respectivamente. Se apoya en referencias externas de www.x3dom.org

(Fuentes de información en el apartado **8.5.2-VRML**)

5.5-X3D

X3D es un formato de archivo de estándares abiertos y arquitectura de tiempo de ejecución, para representar escenas y objetos 3D utilizando XML. Es un estándar ISO que proporciona un sistema de almacenamiento, recuperación y reproducción en tiempo real, de contenido gráfico insertado en aplicaciones, todo ello en una arquitectura abierta para soportar una amplia gama de ámbitos y escenarios de usuario.

X3D tiene un rico conjunto de características que pueden adaptarse para el uso en ingeniería y visualización científica, CAD y arquitectura, visualización médica, entrenamiento y simulación, multimedia, entretenimiento, educación, y mucho más.

El desarrollo de la comunicación en tiempo real de los datos 3D a través de todas las aplicaciones y aplicaciones de red, ha evolucionado desde sus inicios como el Virtual Reality Modeling Language (VRML), hasta el estándar X3D; mucho más maduro y refinado.

X3D puede considerarse como la próxima generación de VRML. Los archivos X3D se pueden mostrar con un plugin de X3D, visor, navegador, programa o conjunto de herramientas específicos, aunque la mayoría de los plugins VRML también puede mostrar archivos X3D.

5.5.1-Características del archivo X3D

CODIGO

En el código, vemos que la estructura del archivo X3D es muy similar al archivo VRML. En el caso del archivo VRML, las secciones estaban definidas entre llaves; {...}. En X3D, las secciones van entre etiquetas de apertura y cierre, típicas de HTML; **<X3D>....</X3D>**.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN"
<X3D xmlns:xsd="http://www.w3.org/2001/XMLSchema-i
<Scene DEF='scene'>
  <Transform DEF='barra_direccion01' rotation='0
  <Shape>
    <Appearance>
      <Material diffuseColor='0.1059 0.6941 0.
    </Appearance>
    <IndexedFaceSet DEF='barra_direccion01-FAC
2 91 89 -1 95 91 92 -1 95 93 91 -1 95 94 93 -1 96
7 -1 162 159 160 -1 162 161 159 -1 164 161 162 -1
232 228 230 -1 232 231 228 -1 234 231 232 -1 234 2
295 296 -1 298 297 295 -1 300 297 298 -1 300 299 2
    <Coordinate DEF='barra_direccion01-COORD
-12.29 165 2.275 -11.62 -165 4.601 -11.62 165 4.60
6.84 -165 -10.46 -12.55 -165 -8.208 -8.839 -165 -8
-165 -5.521 -13.95 165 -5.521 -12.55 -165 -8.208 -
5 5.521 165 -13.95 6 151 -13.75 5.521 151 -13.95 5
-6.84 165 -10.46 -2.729 165 -14.75 -4.601 165 -11.
  </IndexedFaceSet>
</Shape>
</Transform>
</Scene>
</X3D>

```

Información sobre la versión, codificación, etc.

Información sobre el objeto; nombre, coordenadas de posición y rotación, y material.

Información de la geometría; coordenadas de todas las caras y vértices de la pieza.

La imagen anterior corresponde a un ejemplo de una única pieza transformada a X3D. Se trata de una de las barras de la dirección del patinete. En el proceso de transformación se escoge, como código de salida en el convertor online de www.x3dom.org, **XML encoding (X3D)**.

Aquí podemos ver el código, tal cual nos lo devuelve el convertor online:

3 | choose output encoding

XML encoding (X3D)

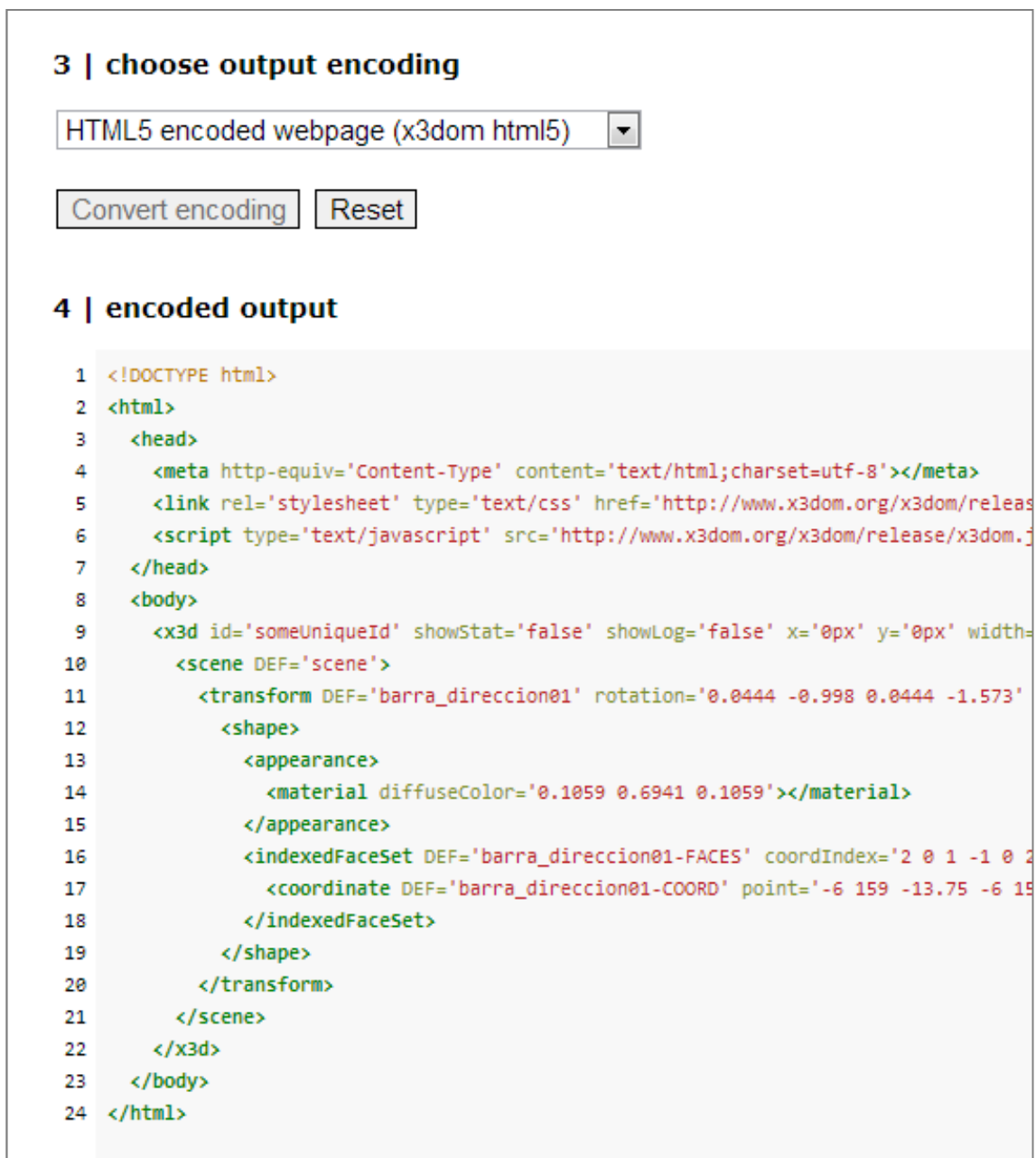
4 | encoded output

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifica
3 <X3D xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance" profile='Full' version=
4   <Scene DEF='scene'>
5     <Transform DEF='barra_direccion01' rotation='0.0444 -0.998 0.0444 -1.573' tran
6     <Shape>
7       <Appearance>
8         <Material diffuseColor='0.1059 0.6941 0.1059' />
9       </Appearance>
10      <IndexedFaceSet DEF='barra_direccion01-FACES' coordIndex='2 0 1 -1 0 2 3 -
11      <Coordinate DEF='barra_direccion01-COORD' point='-6 159 -13.75 -6 159 -1
12    </IndexedFaceSet>
13    </Shape>
14  </Transform>
15  </Scene>
16 </X3D>

```


A continuación, vemos la misma pieza transformada a X3D. En este caso, como código de salida en el conversor online de www.x3dom.org, se ha escogido **HTML5 encoded webpage (x3dom html5)**. El código X3D va integrado en código HTML5, por lo que ya se puede visualizar en el navegador, al ser una página web en sí misma. Bastará con copiar el código y guardarlo con extensión **.html**.



3 | choose output encoding

HTML5 encoded webpage (x3dom html5) ▼

Convert encoding Reset

4 | encoded output

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta http-equiv='Content-Type' content='text/html; charset=utf-8'></meta>
5     <link rel='stylesheet' type='text/css' href='http://www.x3dom.org/x3dom/release/x3dom.css'>
6     <script type='text/javascript' src='http://www.x3dom.org/x3dom/release/x3dom.js'></script>
7   </head>
8   <body>
9     <x3d id='someUniqueId' showStat='false' showLog='false' x='0px' y='0px' width='100%' height='100%'>
10      <scene DEF='scene'>
11        <transform DEF='barra_direccion01' rotation='0.0444 -0.998 0.0444 -1.573'>
12          <shape>
13            <appearance>
14              <material diffuseColor='0.1059 0.6941 0.1059'></material>
15            </appearance>
16            <indexedFaceSet DEF='barra_direccion01-FACES' coordIndex='2 0 1 -1 0 2'>
17              <coordinate DEF='barra_direccion01-COORD' point='-6 159 -13.75 -6 159 -13.75'></coordinate>
18            </indexedFaceSet>
19          </shape>
20        </transform>
21      </scene>
22    </x3d>
23  </body>
24 </html>
```

NAVEGADOR

Al abrir dicha página web el navegador, observaremos lo siguiente (siempre y cuando el navegador soporte x3dom, sistema en el que se apoya el código obtenido):



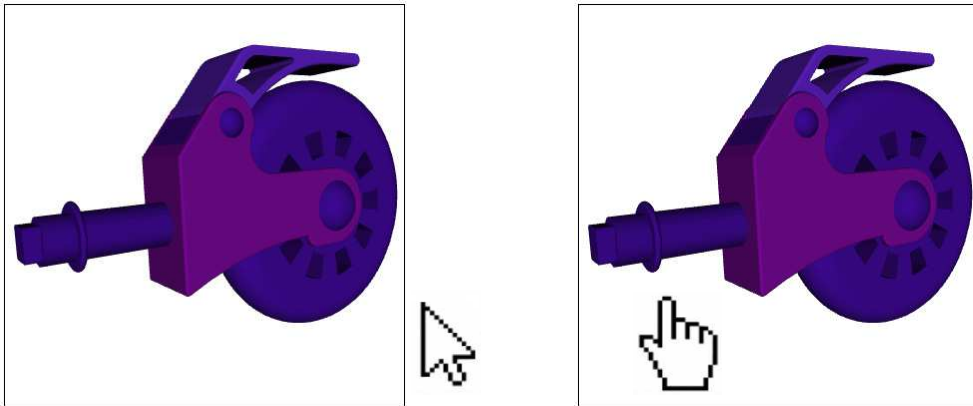
Nos encontramos con una página en blanco en la que aparece una pequeña ventana donde se visualiza el modelo transformado. Esta ventana tiene un tamaño predefinido de 400 x 400 píxeles; que viene indicado en la etiqueta **<x3d>**:

```
<x3d id='someUniqueId' showStat='false' showLog='false' x='0px' y='0px' width='400px' height='400px'>
```

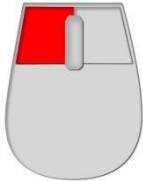
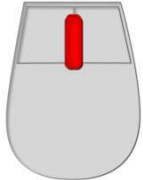
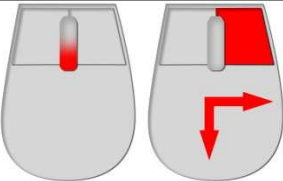
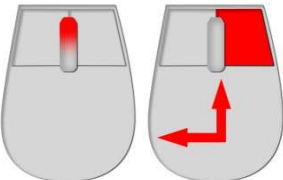
Al igual que otras características del archivo, esto se podrá variar con facilidad.

CONTROLES

El modelo 3D que aparece en pantalla, directamente del conversor, ya se puede manipular haciendo uso del ratón. En cuanto nos ponemos encima de la ventana de X3D con el ratón; el puntero cambia a forma de mano, indicando así que podemos actuar sobre la escena.



Los controles de la escena serán los siguientes:

GIRAR	
	Botón izquierdo del ratón y movimientos en cualquier dirección. Se gira el modelo 3D.
MOVER	
	Botón central del ratón y movimientos en cualquier dirección. Se desplaza el modelo 3D.
AMPLIAR	
	Girando la rueda del ratón hacia abajo o pulsando botón derecho y arrastrando hacia abajo o hacia la derecha.
REDUCIR	
	Girando la rueda del ratón hacia arriba o pulsando botón derecho y arrastrando hacia arriba o hacia la izquierda.

*Podremos centrar la escena haciendo doble clic en alguna zona de ésta.

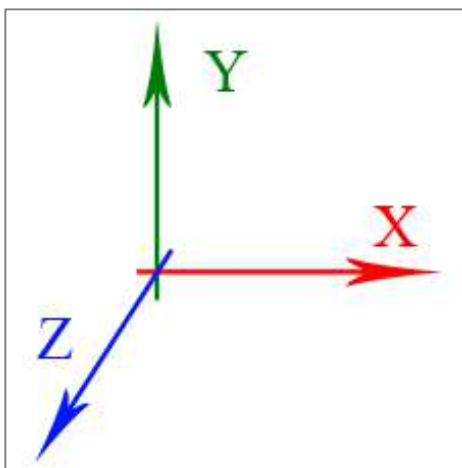
5.5.2-Particularidades de la conversión a X3D

A continuación se mencionan algunos puntos a tener en cuenta, o problemas encontrados en el proceso. Esto complicará algo más la transformación de nuestro modelo.

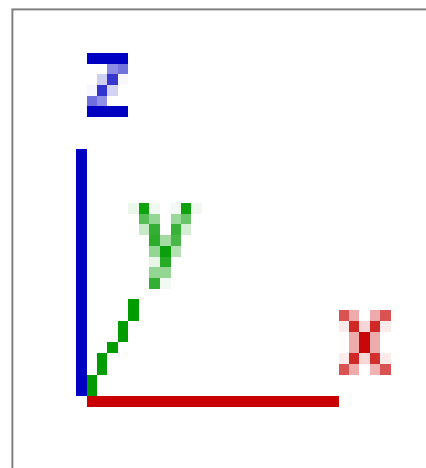
COORDENADAS

Hay que tener en cuenta, que en la transformación del modelo, cambian las coordenadas. Si por ejemplo en el archivo de *3ds Max* tenemos un objeto con las coordenadas **(2,291 -7,36 0)** para X, Y, y Z respectivamente, en el código tendremos **translation 2.291 0 7.36**. Las coordenadas pasarán de ser (X, Y, Z) a (X, Z, -Y).

Las coordenadas en X3D serán las mismas que en VRML. En VRML97 el eje X positivo apunta a la derecha, el eje Y positivo apunta hacia arriba y el eje Z positivo apunta hacia fuera de la pantalla, tal y como muestra la siguiente imagen.



VRML y X3D



3ds Max


(Sistema de Coordenadas *World*, o Universal)

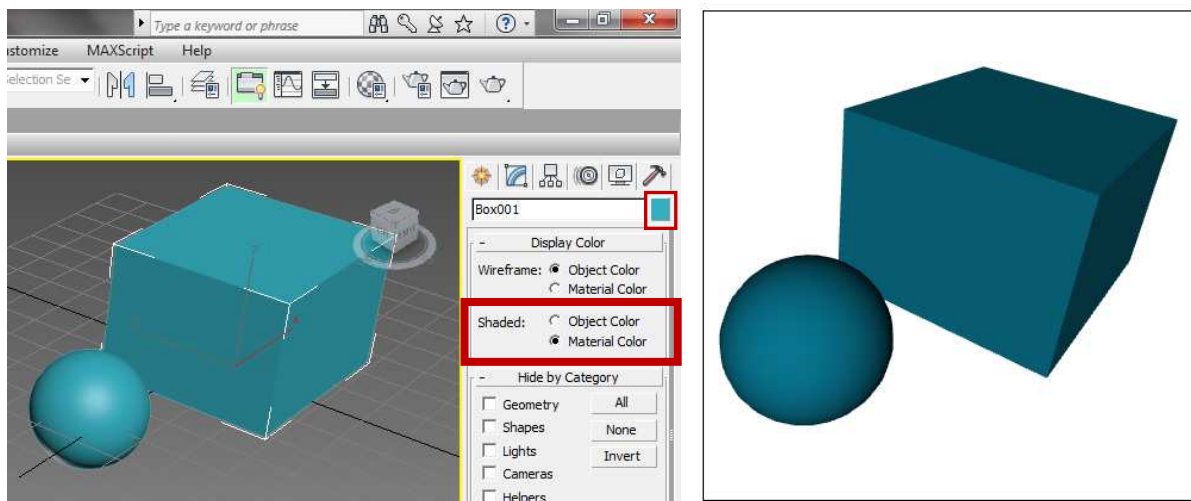
(Fuentes de información en los apartados 5 y 6 del capítulo 8-Bibliografía)

PUNTO DE VISTA

Por defecto, el punto de vista de la escena estará en el origen de coordenadas. Por lo que habrá que modificar el punto de vista, como se explica más adelante (apartados **5.5.4-Visualizador 360°** y **5.5.5-Vistas**).

MATERIALES

Por razones desconocidas, los materiales aplicados a los objetos del archivo de *3ds Max*, no se representan en la escena de X3D. En cambio, sí mantiene el color del objeto; el que se le asigna al objeto al crearlo. Si en *3ds Max*, accedemos al panel *Display* , y en la sección *Display Color* escogemos Object Color, ese será el color con el que aparecerá el objeto en la escena X3D.



Lo que vemos a continuación, es la conversión a X3D del código VRML que vimos en el apartado **5.4.2-Aspecto de los objetos en VRML: Materiales**. En este ejemplo, aparece resaltada la etiqueta `<material...></material>`, equivalente al nodo **Material** en VRML, y que definirá el aspecto del objeto en la escena.

```

<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv='Content-Type' content='text/html; charset=utf-8'></meta>
    <link rel='stylesheet' type='text/css'
      href='http://www.x3dom.org/x3dom/release/x3dom.css'></link>
    <script type='text/javascript'
      src='http://www.x3dom.org/x3dom/release/x3dom.js'></script>
  </head>
  <body>
    <x3d id='someUniqueId' showStat='false' showLog='false' x='0px' y='0px' width='400px'
      height='400px'>
      <scene DEF='scene'>
        <shape>
          <appearance>
            <material ambientIntensity='0.6' diffuseColor='1 0.8 0.8' emissiveColor='0 1 0'
              shininess='0.6' specularColor='1 1 0' transparency='0.5'></material>
          </appearance>
          <cone bottomRadius='2' height='4'></cone>
        </shape>
      </scene>
    </x3d>
  </body>
</html>

```

SUAVIZADO DE SUPERFICIES

X3dom necesita las normales de los vértices para calcular el sombreado de las superficies. El sistema trata de crear (y actualizar) estas normales, si no son dadas por el usuario. El proceso de creación de las normales, puede controlarse con un valor de **creaseAngle** por cada elemento; definido por el usuario. El campo **creaseAngle** define un ángulo (en radianes) para determinar si polígonos adyacentes se dibujan con bordes afilados o un sombreado suave. Si el ángulo entre las normales de dos polígonos adyacentes es menor que **creaseAngle**, un sombreado suave se renderizará a través del segmento de línea compartido.

Algunos sistemas (por ejemplo, WebGL-backend) sólo soportarán valores entre 0.0 y 3.14. Es decir, $\text{creaseAngle}=0$ significa que todos los bordes serán afilados y $\text{creaseAngle}=3.14$, todos ellos suavizados.

A continuación vemos el mismo elemento con diferentes valores: ($\text{creaseAngle}=0$ y $\text{creaseAngle}=1$, respectivamente). Este valor se añadirá dentro de la etiqueta `<indexedFaceSet....>` `</indexedFaceSet>`; contenida a su vez en `<shape...>``</shape>`.



Esto será muy interesante, porque dará un aspecto mucho más atractivo, y aportará algo más de realismo a las piezas.

5.5.3-Montaje de conjuntos en X3D

Ya que cómo hemos visto antes; hemos tenido que “desmontar” el modelo para poder exportarlo satisfactoriamente, ahora debemos volver a “montarlo”. Esto lo haremos copiando el código de cada una de las piezas o conjuntos que hayamos convertido, y pegándolo en una misma etiqueta X3D.

Para poder hacerlo de forma ordenada, se pueden poner comentarios a modo de guía. Un comentario en HTML se escribe de la siguiente forma: **<!--Comentario-->**. El texto entre los símbolos, sólo aparecerá en el código fuente, no se verá en el navegador.

Se copiará el código de cada archivo X3D correspondiente a cada transformación; es decir, correspondiente a cada pieza o conjunto que se haya exportado correctamente, dentro de una misma escena (**<scene>...</scene>**).

```

<!DOCTYPE html>
<html>
....
<body>
  <x3d .....>
    <scene DEF='scene'>
      <viewpoint ...>...</viewpoint>
<!--PATINETE-->
<!--CONJUNTO_1-->
  <transform DEF='PIEZA_1-1' ...>...</transform>
  <transform DEF='PIEZA_1-2' ...>...</transform>
  ....
<!--CONJUNTO_2-->
  <transform DEF='PIEZA_2-1' ...>...</transform>
  <transform DEF='PIEZA_2-2' ...>...</transform>
  ....
</scene>
</x3d>
</body>
</html>

```


5.5.4-Visualizador 360°

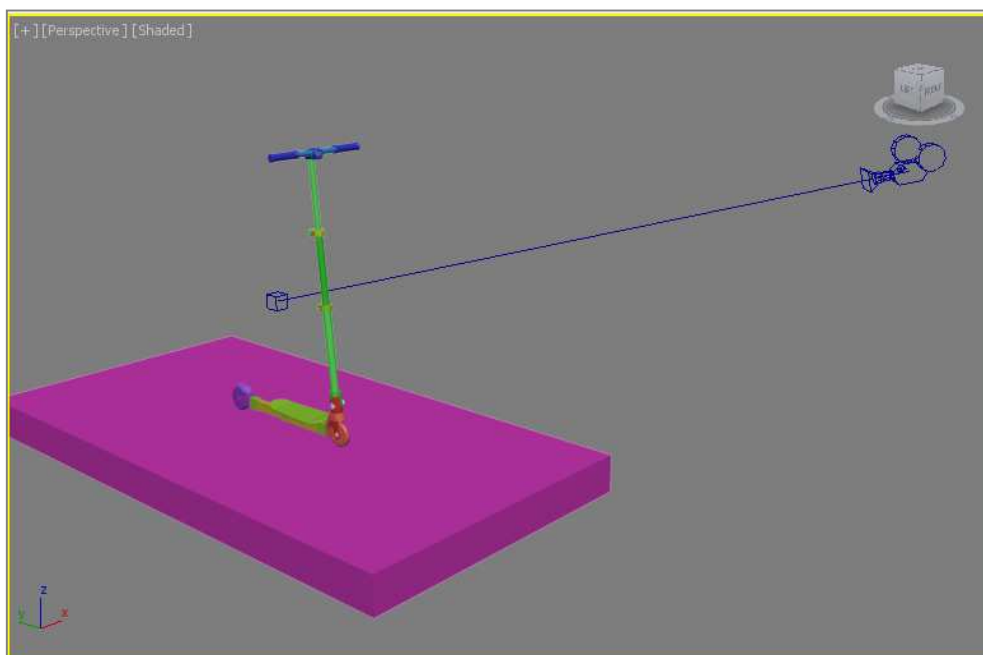
CAMBIO DE PUNTO DE VISTA

Como ya hemos visto en el apartado **5.5.2-Particularidades de la conversión a X3D**; por defecto, tanto en VRML como en X3D, el punto de vista de la escena estará en el origen de coordenadas. Tendremos que modificarlo manualmente.

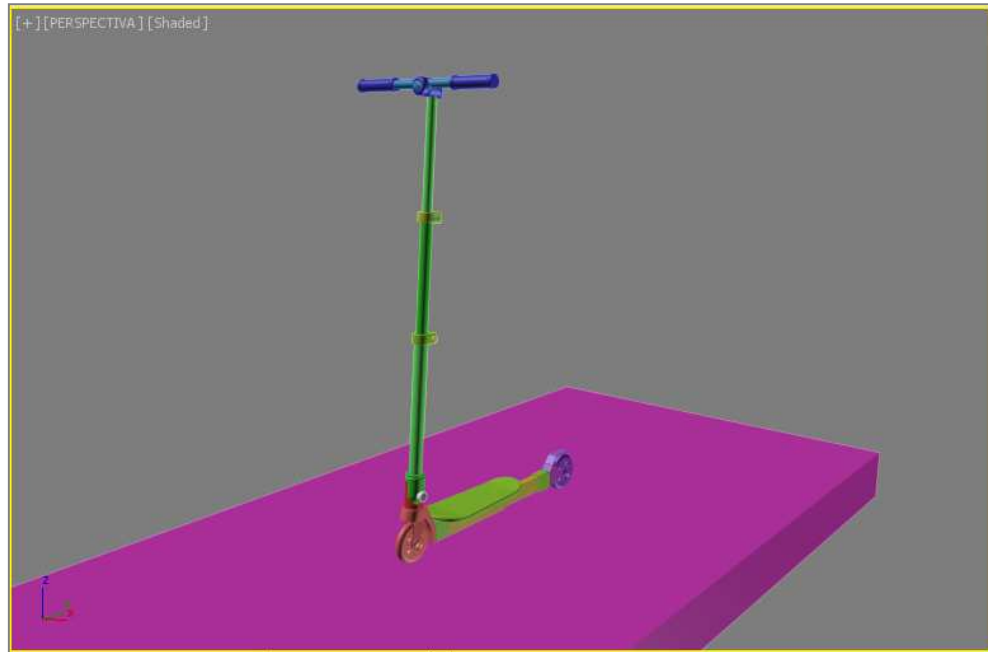
Una posibilidad es introducir en el código, nuevas coordenadas y orientación para el punto de vista. Ya que no estamos familiarizados con este lenguaje, optamos por hacerlo desde *3ds Max*. El proceso será el siguiente.

Crearemos una cámara con la posición y orientación que estimemos convenientes. Nos quedaremos sólo con los elementos de la escena que necesitemos para este propósito; por ejemplo si utilizamos “*Target Camera*” nos quedaríamos con la cámara y el elemento *Target*. Si por el contrario estamos usando una “*Free Camera*”, nos quedamos sólo con ésta. (También podemos simplificar la escena y sustituir el patinete por un objeto sencillo; caja o esfera por ejemplo, de forma que podamos visualizar el resultado más tarde).

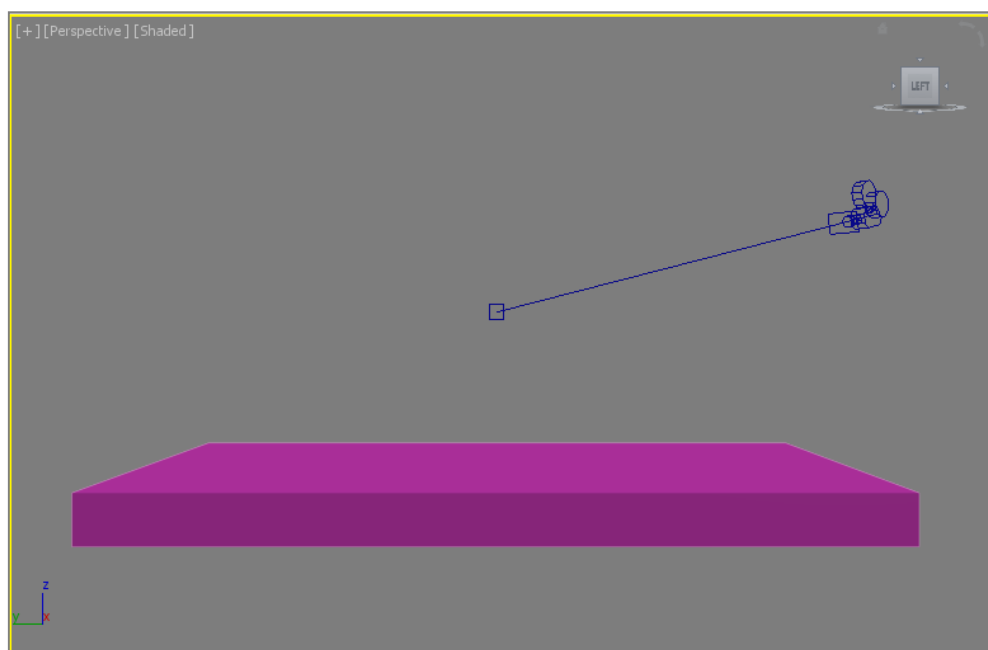
En este caso, se trata de una “*Target Camera*”; colocada en posición más o menos elevada, dando una perspectiva del patinete.



En la siguiente imagen vemos lo que capta dicha cámara.



Se aíslan los elementos a exportar.



A continuación, exportaremos un nuevo archivo VRML y lo convertiremos a X3D de la misma forma que hicimos antes con el resto de objetos (piezas independientes y conjuntos). Esto nos devolverá una etiqueta `<viewpoint ...></viewpoint>`, la cual corresponde a la nueva orientación del punto de vista de la escena X3D, es decir los datos de la cámara que hemos creado en el archivo de *3ds Max*.

VRML:

```
#VRML V2.0 utf8
```

```
# Produced by 3D Studio MAX VRML97 exporter, Version 15, Revision 3,47
# MAX File: PATINETE_CAMARAS_(solo_camaras).max, Date: ....
```

```
DEF PERSPECTIVA Viewpoint {
  position 1452 1000 1787
  orientation 0.2852 -0.953 -0.1019 -0.7177
  fieldOfView 0.7363
  description "PERSPECTIVA"
}
DEF Box001 Transform {
  translation -13 0 0
  children [
    Transform {
      translation 0 -71.67 0
      children [
        Shape {
          appearance Appearance {
            material Material {
              diffuseColor 0.5412 0.03137 0.4314
            }
          }
          geometry Box { size 1190 143.3 2268 }
        }
      ]
    }
  ]
}
```

X3D:

```

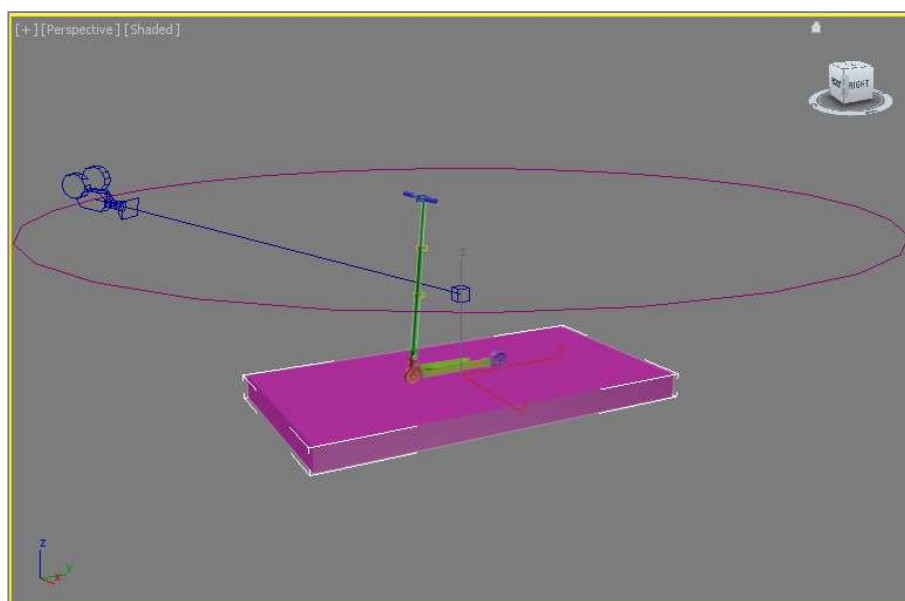
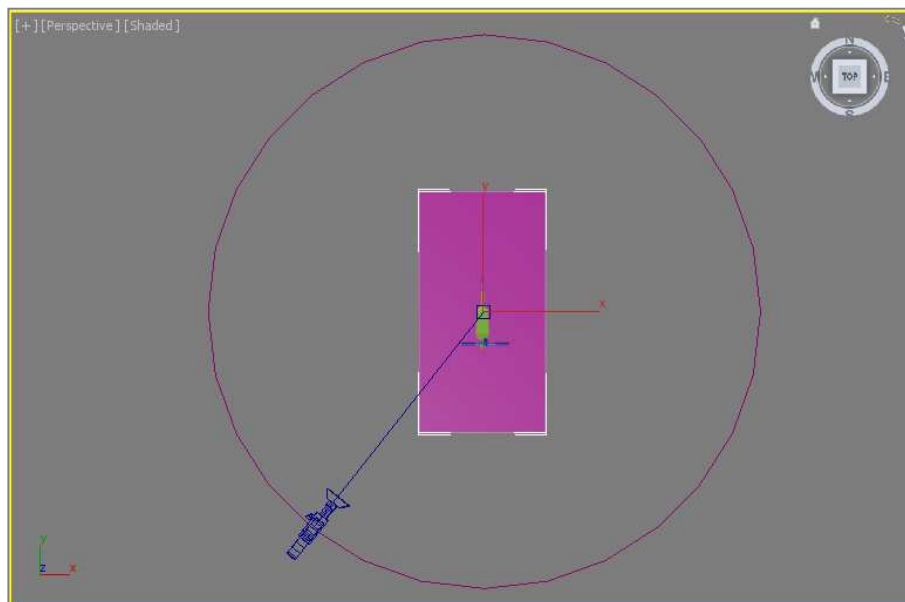
<!DOCTYPE html>
<html>
  <head>
    <meta ....></meta>
    <link.....></link>
    <script .....></script>
  </head>
  <body>
    <x3d .....>
      <scene DEF='scene'>
        <viewpoint      DEF='PERSPECTIVA'      description='PERSPECTIVA'
          orientation='0.2852  -0.953  -0.1019  -0.7177'  position='1452  1000  1787'
          fieldOfView='0.7363'></viewpoint>
        <transform DEF='Box001' translation='-13 0 0'>
          <transform translation='0 -71.67 0'>
            <shape>
              <appearance>
                <material diffuseColor='0.5412 0.03137 0.4314'></material>
              </appearance>
              <box size='1190 143.3 2268'></box>
            </shape>
          </transform>
        </transform>
      </scene>
    </x3d>
  </body>
</html>

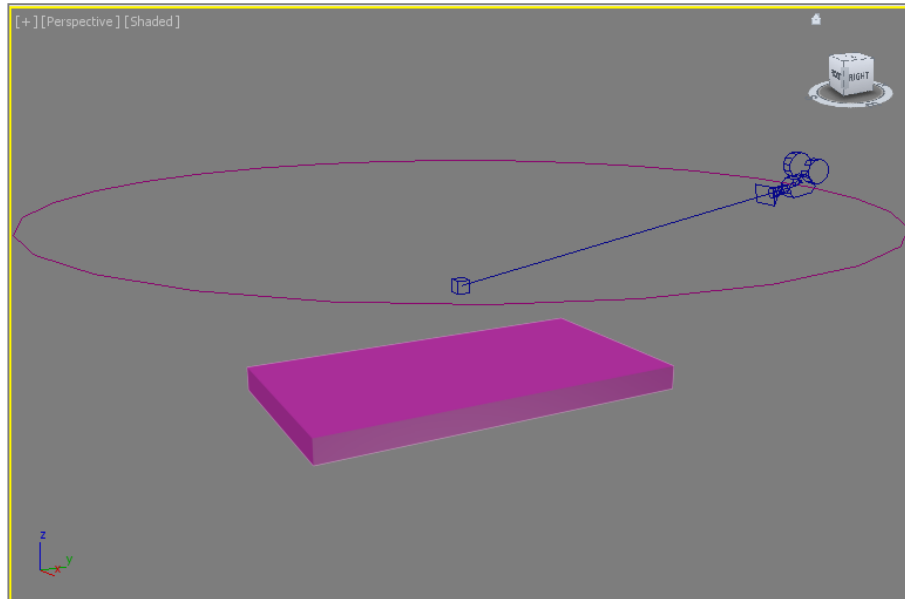
```

En el ejemplo, sustituimos el patinete por una caja, a modo de referencia, aunque no sería necesario. Lo que haremos será copiar el código correspondiente a la cámara y pegarlo en el archivo de nuestra escena. Hay que tener en cuenta que la etiqueta **<viewpoint ...></viewpoint>** debe ir inmediatamente después de la etiqueta **<scene DEF='scene'>**; de apertura de la escena.

VISTA DE 360 GRADOS

Para tener una vista de 360 grados automática; es decir que la cámara gire indefinidamente alrededor del patinete, necesitaremos crear una cámara en *3ds Max*, que haga esto mismo. De la misma forma que vimos en el capítulo 4 (4.7.1-Asignar una Trayectoria a la Cámara), crearemos para ello, una “*Target Camera*”, y un círculo. El centro del círculo y el objetivo (*Target*) estarán centrados en el patinete. A continuación asignaremos a la cámara, el círculo a modo de trayectoria, y creamos una animación en la que la cámara dé una vuelta completa. Sólo queda exportar a VRML y transformar a X3D, como antes.





Al exportar y transformar la cámara animada, además de la etiqueta `<viewpoint...></viewpoint>`, encontramos las etiquetas `<timeSensor...></timeSensor>`, `<positionInterpolator...></positionInterpolator>`, y `<orientationInterpolator ... ></orientationInterpolator>`. Estas etiquetas irán en este orden, inmediatamente después de la etiqueta `<scene DEF='scene'>`; de apertura de la escena.

Después de la etiqueta de transformación, y antes de la etiqueta de cierre de la escena `</scene>`, encontramos las etiquetas `<ROUTE...></ROUTE>`, referentes también a la animación de la cámara.

VRML:

```
#VRML V2.0 utf8
```

```
# Produced by 3D Studio MAX VRML97 exporter, Version 15, Revision 3,47
```

```
# MAX File: PATINETE_CAMARAS_(solo_camaras).max, Date: Tue May 14 19:18:25  
2013
```

```
DEF _60 Viewpoint {  
  position 2281 1000 -6.349  
  orientation 0.1062 -0.9886 -0.1065 -1.585  
  fieldOfView 0.6024
```

```

    description "_60"
  }
  DEF _60-TIMER TimeSensor { loop TRUE cycleInterval 6.667 },
  DEF _60-POS-INTERP PositionInterpolator {
    key [... ]
    keyValue [.....
    ] },
  DEF _60-ROT-INTERP OrientationInterpolator {
    key [.....]
    keyValue [.....
    ] },
  ROUTE _60-TIMER.fraction_changed TO _60-POS-INTERP.set_fraction
  ROUTE _60-POS-INTERP.value_changed TO _60.set_position
  ROUTE _60-TIMER.fraction_changed TO _60-ROT-INTERP.set_fraction
  ROUTE _60-ROT-INTERP.value_changed TO _60.set_orientation
  DEF Box001 Transform {
    translation -13 0 0
    children [
      Transform {
        translation 0 -71.67 0
        children [
          Shape {
            appearance Appearance {
              material Material {
                diffuseColor 0.5412 0.03137 0.4314
              }
            }
            geometry Box { size 1190 143.3 2268 }
          }
        ]
      }
    ]
  }
}

```

X3D:

```

<!DOCTYPE html>
<html>
<head>
  <meta http-equiv='Content-Type' content='text/html; charset=utf-8'></meta>
  <link rel='stylesheet' type='text/css'
    href='http://www.x3dom.org/x3dom/release/x3dom.css'></link>
  <script type='text/javascript'
    src='http://www.x3dom.org/x3dom/release/x3dom.js'></script>

```

```

</head>
<body>
  <x3d id='someUniqueId' showStat='false' showLog='false' x='0px' y='0px'
    width='400px' height='400px'>
    <scene DEF='scene'>
      <viewpoint DEF='_60' description='_60' orientation='0.1062 -0.9886 -0.1065 -
        1.585' position='2281 1000 -6.349' fieldOfView='0.6024'></viewpoint>
      <timeSensor DEF='_60-TIMER' cycleInterval='6.667'
        loop='true'></timeSensor>
      <positionInterpolator DEF='_60-POS-INTERP' key='.....'
        keyValue='.....'></positionInterpolator>
      <orientationInterpolator DEF='_60-ROT-INTERP' key='.....'
        keyValue='.....'></orientationInterpolator>
      <transform DEF='Box001' translation='-13 0 0'>
        <transform translation='0 -71.67 0'>
          <shape>
            <appearance>
              <material diffuseColor='0.5412 0.03137 0.4314'></material>
            </appearance>
            <box size='1190 143.3 2268'></box>
          </shape>
        </transform>
      </transform>
      <ROUTE fromNode='_60-TIMER' fromField='fraction_changed' toNode='_60-
        POS-INTERP' toField='set_fraction'></ROUTE>
      <ROUTE fromNode='_60-ROT-INTERP' fromField='value_changed'
        toNode='_60' toField='set_orientation'></ROUTE>
      <ROUTE fromNode='_60-POS-INTERP' fromField='value_changed'
        toNode='_60' toField='set_position'></ROUTE>
      <ROUTE fromNode='_60-TIMER' fromField='fraction_changed' toNode='_60-
        ROT-INTERP' toField='set_fraction'></ROUTE>
    </scene>
  </x3d>
</body>
</html>

```

Lo que haremos será copiar el código correspondiente a la cámara y pegarlo, donde proceda, en el archivo de nuestra escena.

Si manipulamos la escena de 360 grados; haciendo doble clic en cualquier lugar de ella, el visor regresará a su estado inicial.

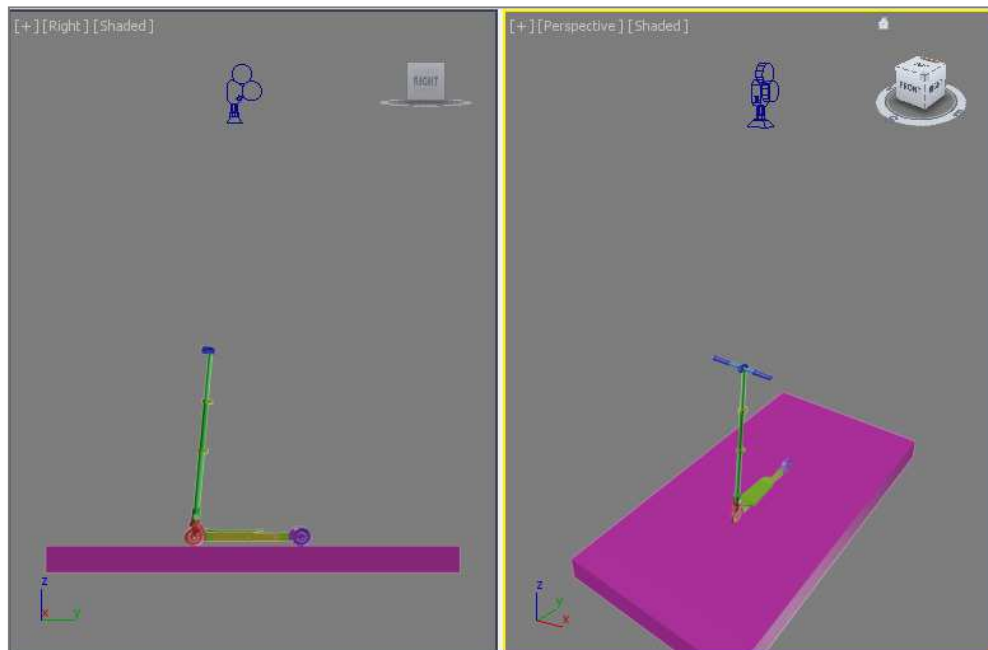
5.5.5-Vistas

Aunque cualquier vista se puede manipular haciendo uso del ratón; permitiendo cambiar el zoom y la posición o rotación de los objetos, otra de las posibilidades es ofrecer diferentes vistas predeterminadas.

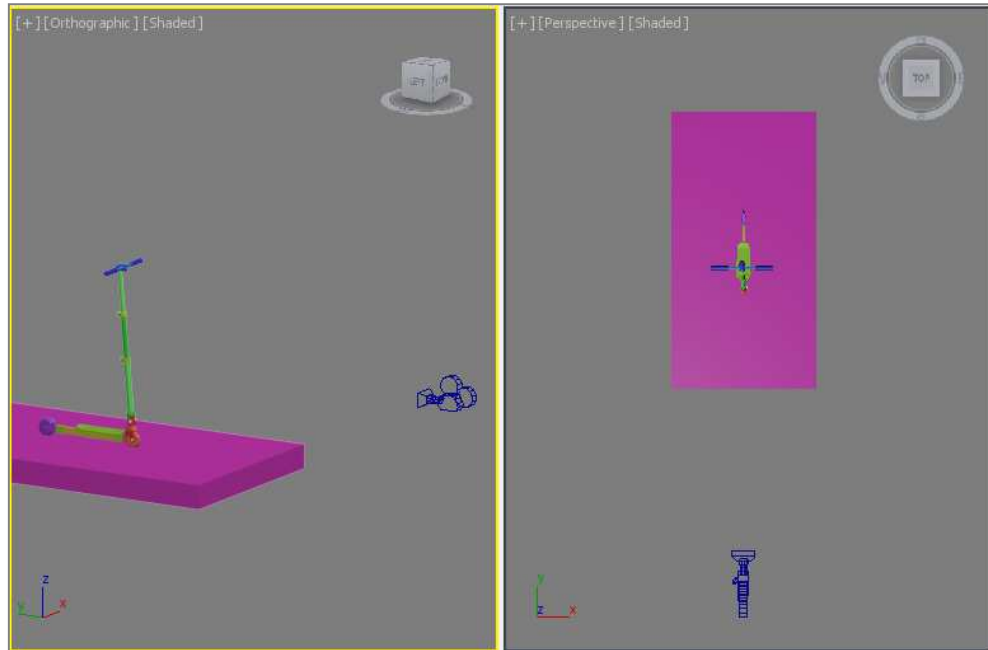
Con este fin, seguiremos el proceso descrito en el apartado anterior; y crearemos en un archivo de *3ds Max*, tantas cámaras como vistas queramos dar. En nuestro caso ofreceremos la vista de 360 grados descrita anteriormente, una perspectiva, y las seis vistas ortogonales (**técnicamente no serán ortogonales; tendrán algo de perspectiva, por motivos estéticos**).

Podremos crear todas las cámaras en un mismo archivo de *3ds Max*. Como el nombre que tienen los elementos en *3ds Max*, se mantiene en VRML y X3D, nombraremos cada cámara según la vista que queramos dar con ella).

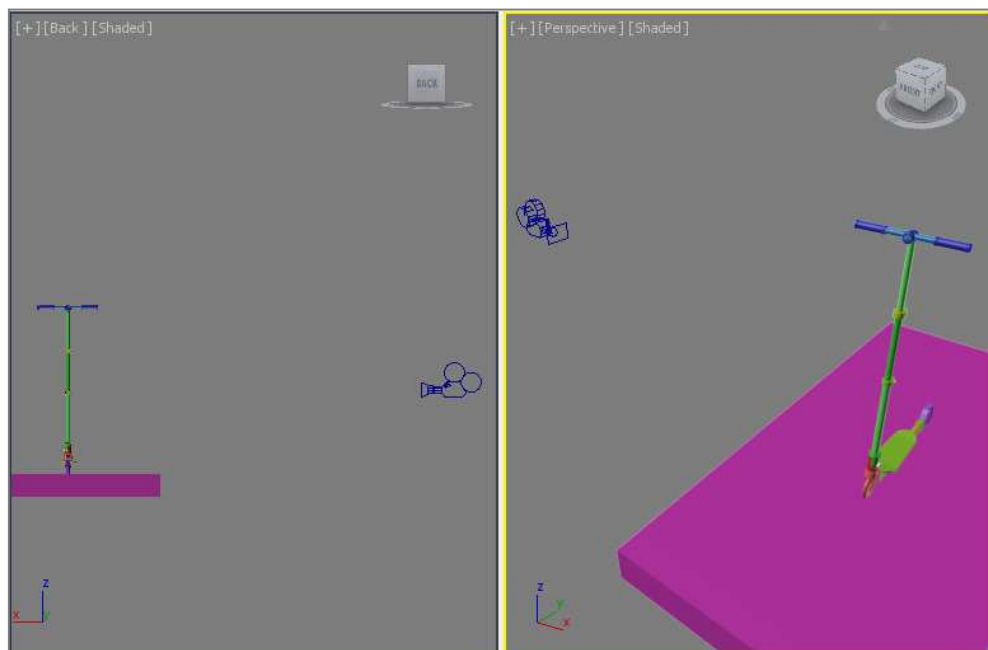
TOP: Cámara que da una vista superior del patinete. En esta ocasión la cámara es una “*Free Camera*”.



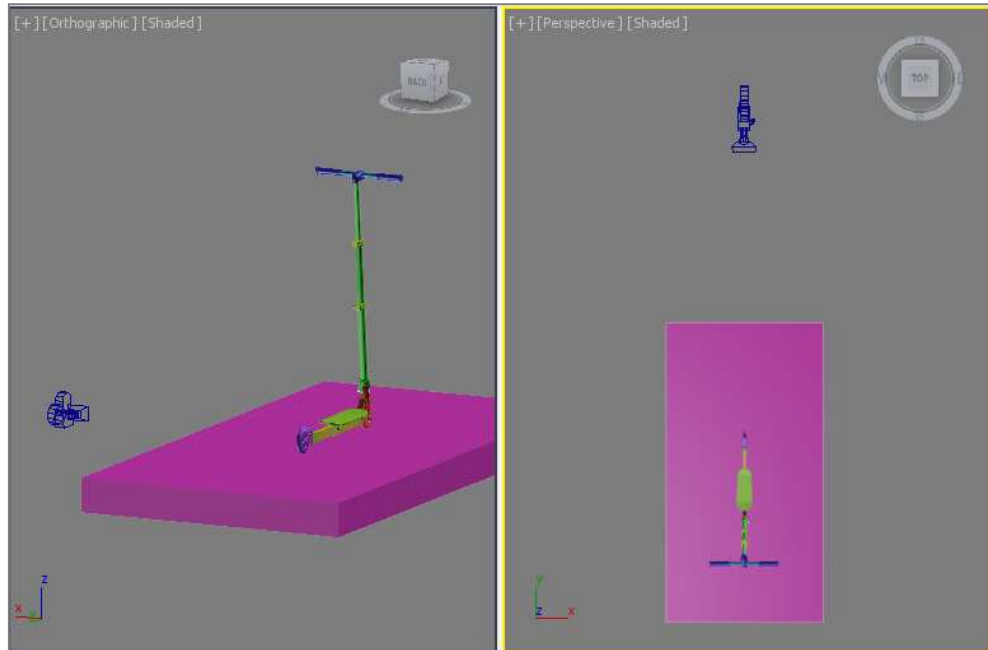
FRONT: “Free Camera” para mostrar la parte frontal del patinete.



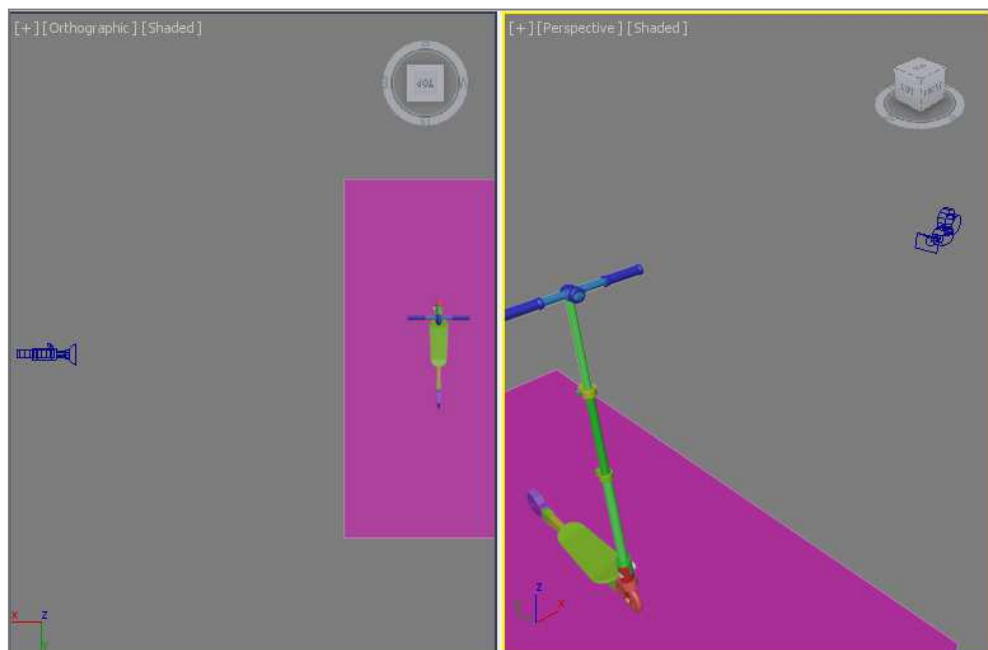
LEFT: “Free Camera” situada en uno de los laterales; perfil a la izquierda de la vista Front.



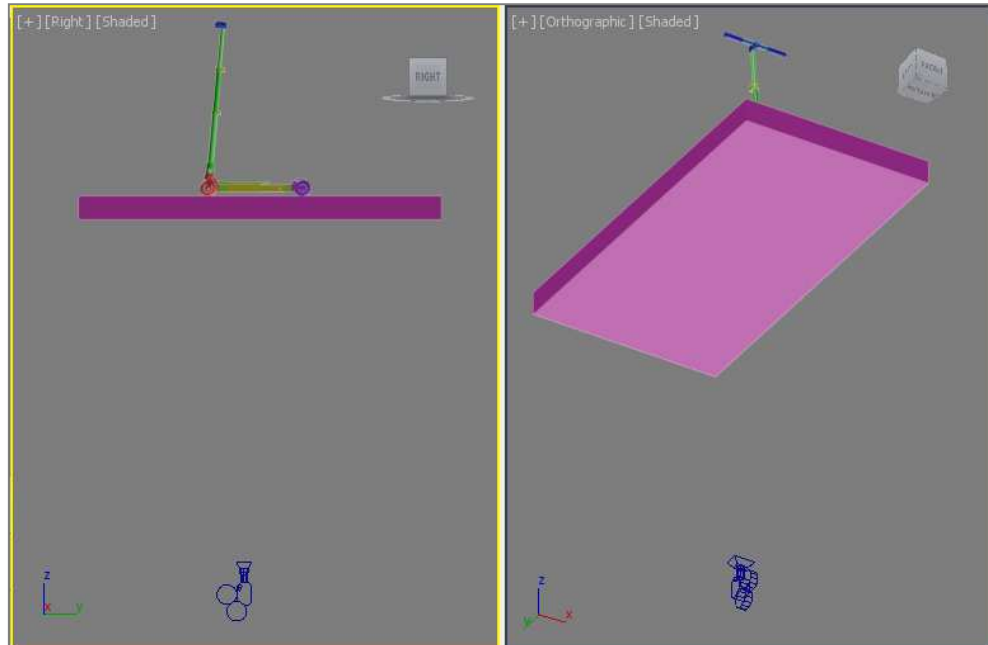
BACK: “Free Camera” que muestra la parte trasera del patinete.



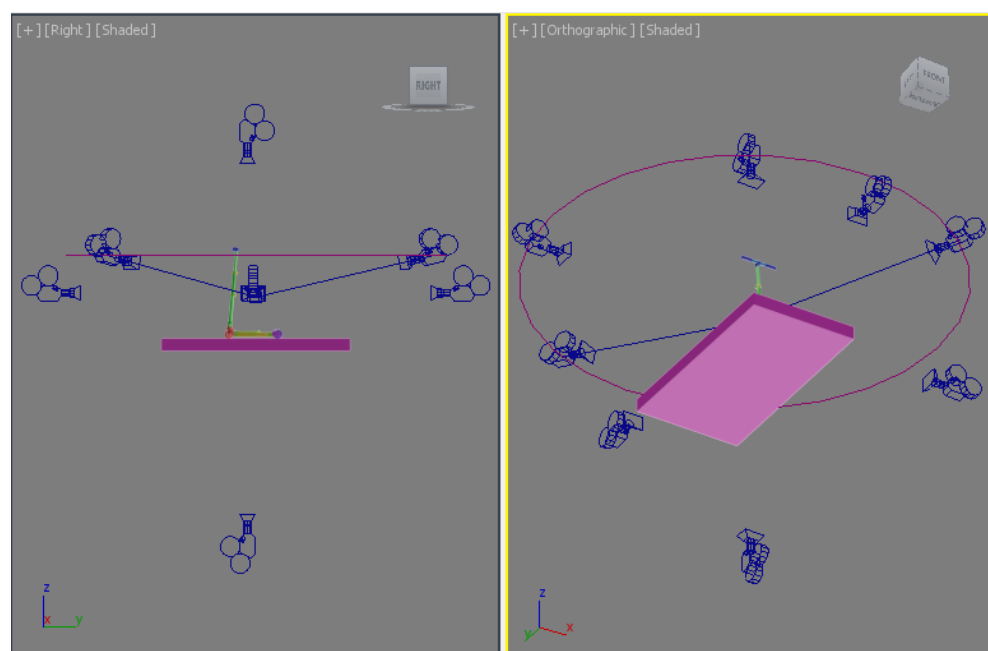
RIGHT: “Free Camera” situada en uno de los laterales; perfil a la derecha de la vista Front.



BOTTOM: “Free Camera” para mostrar la parte inferior del patinete.



Todas las cámaras utilizadas. Seis “Free Camera” para las vistas comunes, una “Target Camera” para una vista en perspectiva, y una “Target Camera” animada sobre una trayectoria circular, para la vista de 360 grados.



Como siempre, sólo falta exportar a VRML y a continuación, transformar a X3D.

VRML:

```
#VRML V2.0 utf8
```

```
# Produced by 3D Studio MAX VRML97 exporter, Version 15, Revision 3,47
```

```
# MAX File: PATINETE_CAMARAS_(solo_camaras).max, Date: Tue May 14 19:18:25  
2013
```

```
DEF TOP Viewpoint {  
  position -13 2500 100  
  orientation 1 0 0 -1.571  
  fieldOfView 0.6024  
  description "TOP"  
}  
DEF FRONT Viewpoint {  
  position -13 550 2500  
  orientation 1 0 0 0  
  fieldOfView 0.6024  
  description "FRONT"  
}  
DEF BACK Viewpoint {  
  position -13 550 -2500  
  orientation 0 -1 0 -3.142  
  fieldOfView 0.6024  
  description "BACK"  
}  
DEF LEFT Viewpoint {  
  position -2513 550 30  
  orientation 0 1 0 -1.571  
  fieldOfView 0.6024  
  description "LEFT"  
}  
DEF RIGHT Viewpoint {  
  position 2487 550 30  
  orientation 0 -1 0 -1.571  
  fieldOfView 0.6024  
  description "RIGHT"  
}  
DEF BOTTOM Viewpoint {  
  position -13 -2500 100  
  orientation -1 0 0 -1.571  
  fieldOfView 0.6024
```

```

    description "BOTTOM"
  }
  DEF PERSPECTIVA Viewpoint {
    position 1452 1000 1787
    orientation 0.2852 -0.953 -0.1019 -0.7177
    fieldOfView 0.7363
    description "PERSPECTIVA"
  }
  DEF _60 Viewpoint {
    position 2281 1000 -6.349
    orientation 0.1062 -0.9886 -0.1065 -1.585
    fieldOfView 0.6024
    description "_60"
  }
  DEF _60-TIMER TimeSensor { loop TRUE cycleInterval 6.667 },
  DEF _60-POS-INTERP PositionInterpolator {
    key [.....
  ]
    keyValue [.....
  ]},
  DEF _60-ROT-INTERP OrientationInterpolator {
    key [.....]
    keyValue [.....
  ]},
  ROUTE _60-TIMER.fraction_changed TO _60-POS-INTERP.set_fraction
  ROUTE _60-POS-INTERP.value_changed TO _60.set_position
  ROUTE _60-TIMER.fraction_changed TO _60-ROT-INTERP.set_fraction
  ROUTE _60-ROT-INTERP.value_changed TO _60.set_orientation
  DEF Box001 Transform {
    translation -13 0 0
    children [
      Transform {
        translation 0 -71.67 0
        children [
          Shape {
            appearance Appearance {
              material Material {
                diffuseColor 0.5412 0.03137 0.4314
              }
            }
            geometry Box { size 1190 143.3 2268 }
          }
        ]
      }
    ]
  }
]

```

}

X3D:

```

<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv='Content-Type' content='text/html;charset=utf-8'></meta>
    <link rel='stylesheet' type='text/css'
      href='http://www.x3dom.org/x3dom/release/x3dom.css'></link>
    <script type='text/javascript'
      src='http://www.x3dom.org/x3dom/release/x3dom.js'></script>
  </head>
  <body>
    <x3d id='someUniqueId' showStat='false' showLog='false' x='0px' y='0px' width='400px'
      height='400px'>
      <scene DEF='scene'>
        <viewpoint DEF='TOP' description='TOP' orientation='1 0 0 -1.571' position='-13
          2500 100' fieldOfView='0.6024'></viewpoint>
        <viewpoint DEF='FRONT' description='FRONT' position='-13 550 2500'
          fieldOfView='0.6024'></viewpoint>
        <viewpoint DEF='BACK' description='BACK' orientation='0 -1 0 -3.142'
          position='-13 550 -2500' fieldOfView='0.6024'></viewpoint>
        <viewpoint DEF='LEFT' description='LEFT' orientation='0 1 0 -1.571' position='-
          2513 550 30' fieldOfView='0.6024'></viewpoint>
        <viewpoint DEF='RIGHT' description='RIGHT' orientation='0 -1 0 -1.571'
          position='2487 550 30' fieldOfView='0.6024'></viewpoint>
        <viewpoint DEF='BOTTOM' description='BOTTOM' orientation='-1 0 0 -1.571'
          position='-13 -2500 100' fieldOfView='0.6024'></viewpoint>
        <viewpoint DEF='PERSPECTIVA' description='PERSPECTIVA'
          orientation='0.2852 -0.953 -0.1019 -0.7177' position='1452 1000 1787'
          fieldOfView='0.7363'></viewpoint>
        <viewpoint DEF='_60' description='_60' orientation='0.1062 -0.9886 -0.1065 -1.585'
          position='2281 1000 -6.349' fieldOfView='0.6024'></viewpoint>
        <timeSensor DEF='_60-TIMER' cycleInterval='6.667' loop='true'></timeSensor>
        <positionInterpolator DEF='_60-POS-INTERP' key='.....'
          keyValue='.....'></positionInterpolator>
        <orientationInterpolator DEF='_60-ROT-INTERP'
          key='.....'
          keyValue='.....'></orientationInterpolator>
        <transform DEF='Box001' translation='-13 0 0'>
          <transform translation='0 -71.67 0'>
            <shape>

```

```

    <appearance>
      <material diffuseColor='0.5412 0.03137 0.4314'></material>
    </appearance>
    <box size='1190 143.3 2268'></box>
  </shape>
</transform>
</transform>
<ROUTE fromNode='_60-TIMER' fromField='fraction_changed' toNode='_60-
POS-INTERP' toField='set_fraction'></ROUTE>
<ROUTE fromNode='_60-POS-INTERP' fromField='value_changed' toNode='_60'
toField='set_position'></ROUTE>
<ROUTE fromNode='_60-TIMER' fromField='fraction_changed' toNode='_60-
ROT-INTERP' toField='set_fraction'></ROUTE>
<ROUTE fromNode='_60-ROT-INTERP' fromField='value_changed' toNode='_60'
toField='set_orientation'></ROUTE>
</scene>
</x3d>
</body>
</html>

```

De la misma forma que antes, lo que haremos será copiar el código correspondiente a las cámaras y pegarlo en el archivo de nuestra escena. Habrá que tener en cuenta el orden en que se colocan las etiquetas relativas a las cámaras; ya que la primera que aparezca en el código, después de **<scene DEF='scene'>**, será la vista por defecto.

SELECCIONAR VISTA

Con el fin de poder seleccionar distintas vistas, tendremos que crear tantos botones como vistas queramos ofrecer. Para ello, asignaremos un identificador a cada una de ellas (**id='camara_1'**); y a cada botón, un comando para cambiar la vista según dicho identificador (**<input type="button" value="CAMARA 1" onclick="document.getElementById('camara_1').setAttribute('set_bind','true');"/>**). Los botones irán incluidos en una etiqueta **<div></div>**.

En nuestro caso, tendremos lo siguiente:

Asignar un **identificador** para cada cámara en la etiqueta `<viewpoint ...></viewpoint>` correspondiente:

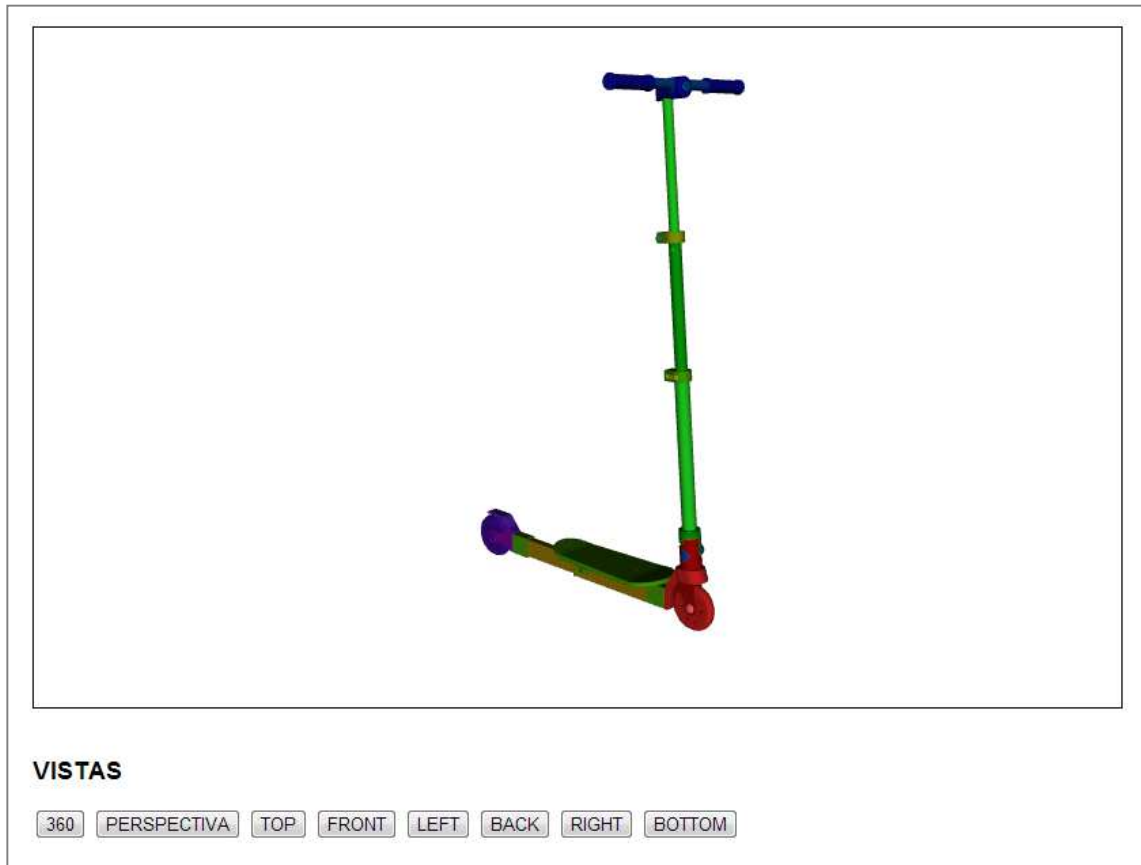
```
<viewpoint DEF='_60' id='360_cam'
.....></viewpoint>
<timeSensor DEF='_60-TIMER' cycleInterval='6.667' loop='true'></timeSensor>
<positionInterpolator DEF='_60-POS-INTERP'
.....></positionInterpolator>
<orientationInterpolator DEF='_60-ROT-INTERP'
.....></orientationInterpolator>
<viewpoint DEF='TOP' id='top_cam'
.....></viewpoint>
<viewpoint DEF='FRONT' id='front_cam'
.....></viewpoint>
<viewpoint DEF='BACK' id='back_cam'
.....></viewpoint>
<viewpoint DEF='LEFT' id='left_cam'
.....></viewpoint>
<viewpoint DEF='RIGHT' id='right_cam'
.....></viewpoint>
<viewpoint DEF='BOTTOM' id='bottom_cam'
.....></viewpoint>
<viewpoint DEF='PERSPECTIVA' id='perspectiva_cam'
.....></viewpoint>
```

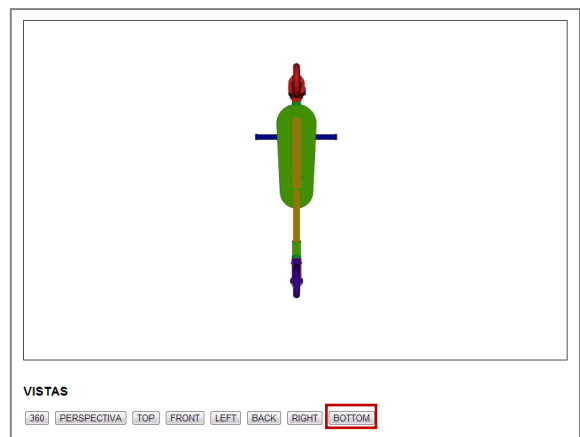
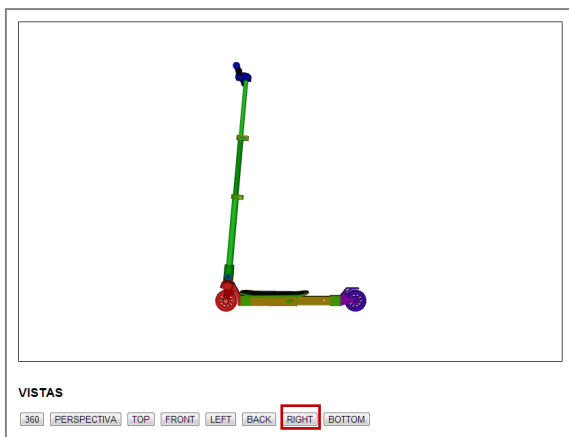
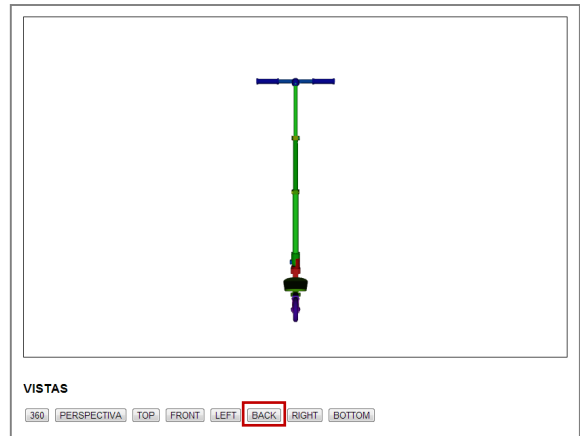
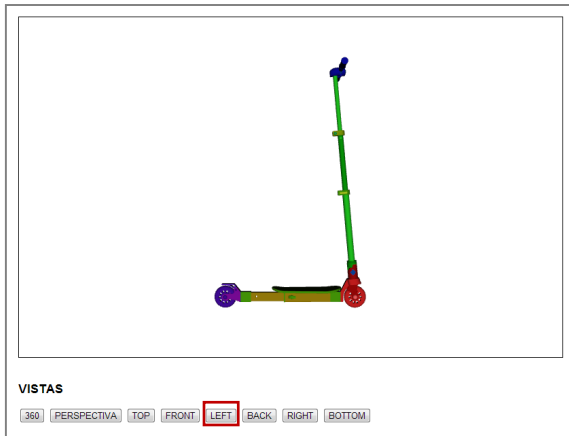
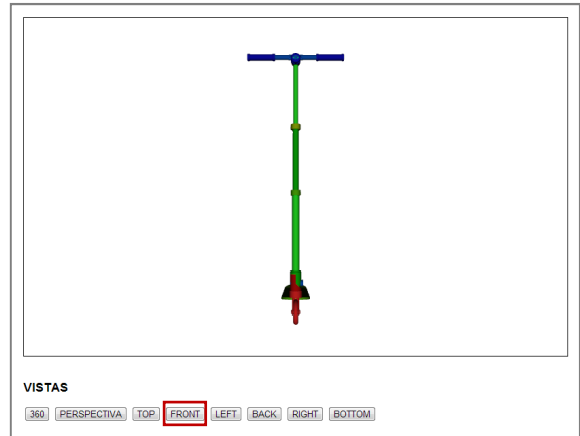
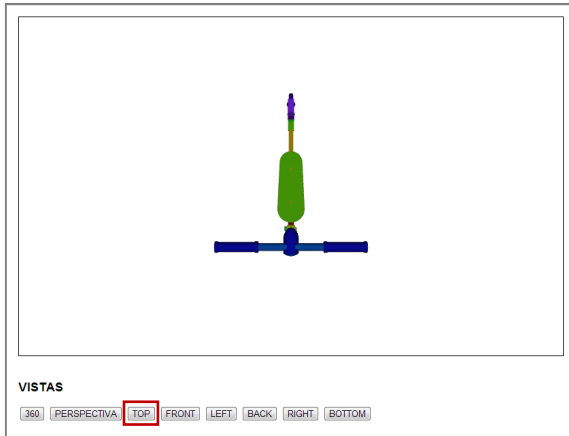
Etiqueta **div** que contenga a los botones de cambio de vista. Esta etiqueta irá fuera de la escena X3D, es decir de su correspondiente etiqueta `<x3d....></x3d>`.

```
<div class="buttons group">
  <input type="button" value="360"
  onclick="document.getElementById('360_cam').setAttribute('set_bind','true');"/>
  <input type="button" value="PERSPECTIVA"
  onclick="document.getElementById('perspectiva_cam').setAttribute('set_bind','true');"/>
  <input type="button" value="TOP"
  onclick="document.getElementById('top_cam').setAttribute('set_bind','true');"/>
  <input type="button" value="FRONT"
```

```
onclick="document.getElementById('front_cam').setAttribute('set_bind','true');"/>
<input type="button" value="LEFT"
onclick="document.getElementById('left_cam').setAttribute('set_bind','true');"/>
<input type="button" value="BACK"
onclick="document.getElementById('back_cam').setAttribute('set_bind','true');"/>
<input type="button" value="RIGHT"
onclick="document.getElementById('right_cam').setAttribute('set_bind','true');"/>
<input type="button" value="BOTTOM"
onclick="document.getElementById('bottom_cam').setAttribute('set_bind','true');"/>
</div>
```

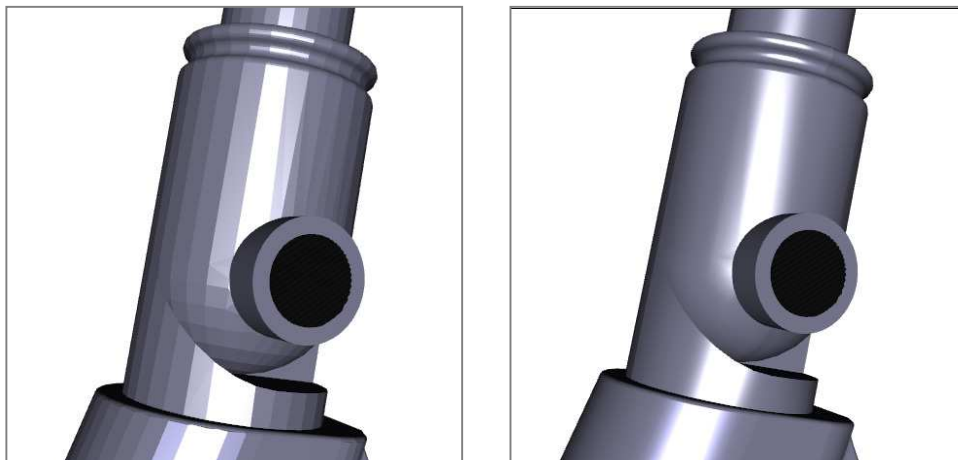
El atributo **value**, corresponde al texto que aparecerá en el botón. El comando activa en el visor, la vista correspondiente a cada identificador. El resultado será el siguiente:





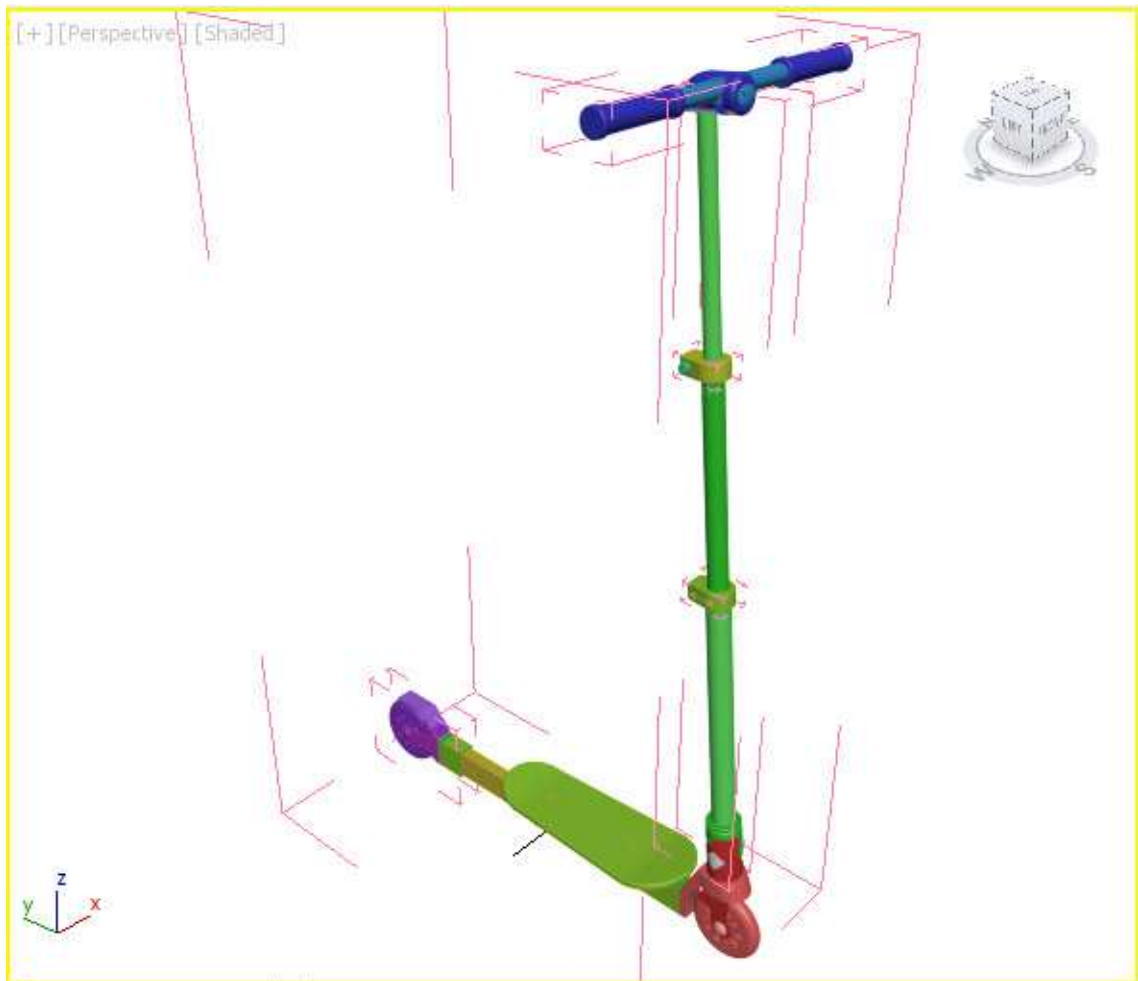
5.5.6-Materiales del patinete

Además del material, para que las piezas adquieran realismo, añadiremos el valor **creaseAngle**, como vimos en el apartado **5.5.2-Particularidades de la conversión a X3D**. De esta forma, las piezas tendrán un acabado más fino.



Como ya hemos visto, en la transformación no se respetan los materiales que habíamos asignado a los objetos para los render; por lo que se exportarán con los colores con que se hayan creado, y más adelante, se cambiarán la etiqueta **<material...>**, para que el modelo sea más atractivo.

En el caso del patinete, a modo de guía, escogeremos una gama de colores para cada subconjunto en que se divide. De esta forma, será más fácil reconocer que piezas vamos cambiando, cuando añadamos los materiales en el código.



A continuación, buscaremos valores para los parámetros que definen el material:

MATERIALES BASICOS O POR DEFECTO:METALES:

Por simplicidad, se asigna la misma etiqueta **<material...></material>**, tanto al aluminio como al acero.

```
<material ambientIntensity='0' diffuseColor='0.6667 0.6667 0.7804' shininess='0.4015'  
specularColor='1.46 1.46 1.46'></material>
```

RUEDAS:

```
<material ambientIntensity='0' diffuseColor='0.07059 0.07059 0.07059' shininess='0.24'  
specularColor='0.9 0.9 0.9'></material>
```

MANGUITOS:

Este material sera asignado tanto a los manguitos como a la goma del botón de abatimiento.

```
<material ambientIntensity='0' diffuseColor='0.07059 0.07059 0.07059' shininess='0.4585'  
specularColor='1.134 1.134 1.134'></material>
```

TABLA:

```
<material ambientIntensity='0' diffuseColor='0.07059 0.07059 0.07059' shininess='0.487'  
specularColor='1.908 1.908 1.908'></material>
```









Los siguientes, se representan en la escena pero no se tienen en cuenta en el configurador de materiales:

PLASTICOS:

Gris: `<material ambientIntensity='0' diffuseColor='0.4078 0.4078 0.4078' shininess='0.5345' specularColor='0.954 0.954 0.954'></material>`

Blanco: `<material ambientIntensity='0' diffuseColor='0.8549 0.8549 0.8549' shininess='0.5345' specularColor='0.954 0.954 0.954'></material>`

En el configurador de materiales dispondremos de los siguientes colores para poder asignar a distintas partes del patinete:

	COLOR	RGB (X3D)	RGB	HTML
	NEGRO	0.07059 0.07059 0.07059	(18 18 18)	#121212
	AZUL	0.0392 0.0902 0.4627	(15 23 118)	#0f1776
	ROJO	---	(132 17 17)	#841111
	VERDE	0.0431 0.3686 0.1804	(11 94 46)	#0b5e2e
	MORADO	---	(94 10 85)	#5e0a55
	NARANJA	0.8000 0.4000 0.0078	(204 102 2)	#cc6602
	METAL CLARO	0.6667 0.6667 0.7804	(170 170 199)	#aaaac7
	METAL OSCURO	0.3216 0.3216 0.3725	(82 82 95)	#52525f

El color RGB es la cantidad de rojo(R), verde(G) y azul(A), que tiene un color; con valores desde 0 a 255. De esta forma; el negro sería la ausencia de color, es decir (0 0 0), y el blanco sería (255 255 255).

En el caso de X3D, los valores corresponden a dividir el valor de cada componente de color por 255; así por ejemplo en este caso, el blanco sería 1 1 1.

5.5.7-Configuración de producto

Uno de los propósitos de este proyecto era poder interactuar con un modelo 3D, con la posibilidad de cambiar los colores del patinete, a modo de configurador de producto. De esta forma, el cliente puede elegir y ver en tiempo real, como quedaría el producto a su elección.

Esto se conseguirá con una función. Habrá que asignar al material de cada pieza, que ofrezca la posibilidad de cambio, un identificador. En el configurador aparecerá una muestra de cada color posible, la cual llevará asignada a su vez un comando. Al hacer clic en la muestra (normalmente será un pequeño cuadro con el color correspondiente), se activa el comando, que llama a la función; haciendo que el color que tenía anteriormente la pieza, se cambie por el de la muestra.

A continuación vemos un ejemplo de lo descrito:

EN LA PIEZA:

```
...
<transform DEF='PIEZA_1' rotation='....' translation='.....'>
  <shape>
    <appearance>
      <material id='color_PIEZA_1' .....></material>
    </appearance>
  </shape>
</transform>
....
```

Ej: A la **PIEZA_1** se le asigna el identificador color_PIEZA_1.

EN LA MUESTRA:

```

```

Supongamos que la muestra es un pequeño cuadro de color rojo. La muestra llevará asignado un comando, el cual llamará a la función.

FUNCION:

```
<script type="text/javascript">
```

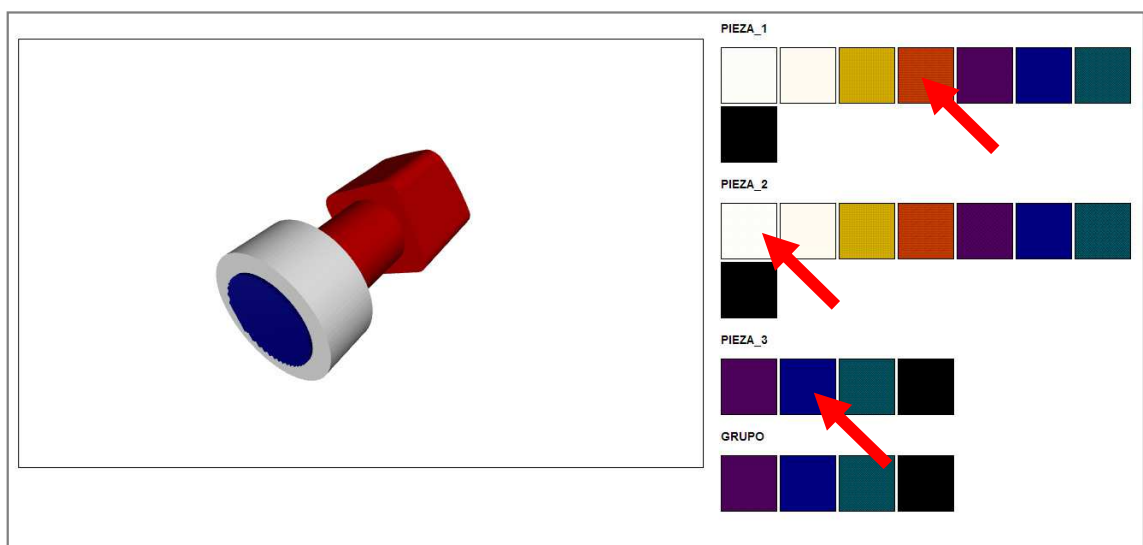
```

function changePIEZA_1(col)
    {
        document.getElementById('color_PIEZA_1').setAttribute("diffuseColor",
col);
    }
</script>

```

La función cambiará el color difuso de la PIEZA_1, por el color que lleva asignado el comando correspondiente a la muestra. En este ejemplo, la función cambia el color difuso a rojo; que es el color que indica el comando.

Habrá entonces, una función para cada pieza o grupo de piezas que compartan muestrario de colores, y un comando para cada muestra.



En el caso de un grupo de piezas que compartan el mismo muestrario, tendremos la siguiente función:

EN LA MUESTRA:

```

```

FUNCION:

```
<script type="text/javascript">
```

```
function changeGRUPO_1(col)
```

```
{
```

```
document.getElementById('color_PIEZA_1').setAttribute("diffuseColor", col);
```

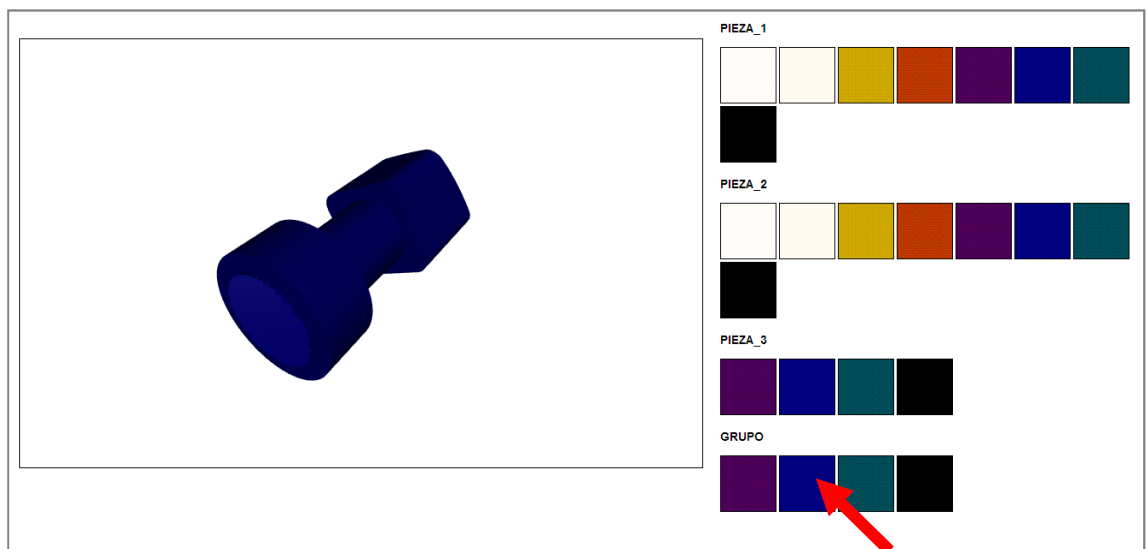
```
document.getElementById('color_PIEZA_2').setAttribute("diffuseColor", col);
```

```
document.getElementById('color_PIEZA_3').setAttribute("diffuseColor", col);
```

```
}
```

```
</script>
```

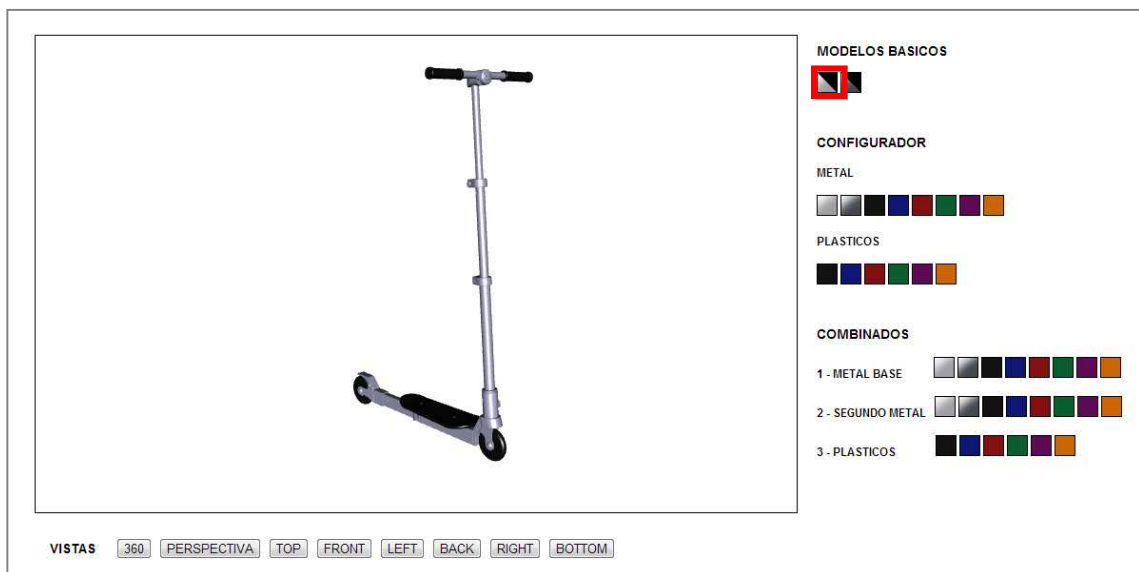
Según el ejemplo, el grupo de piezas, que comparten el mismo muestrario, cambiarían su color a azul como ordena el comando de la muestra elegida.



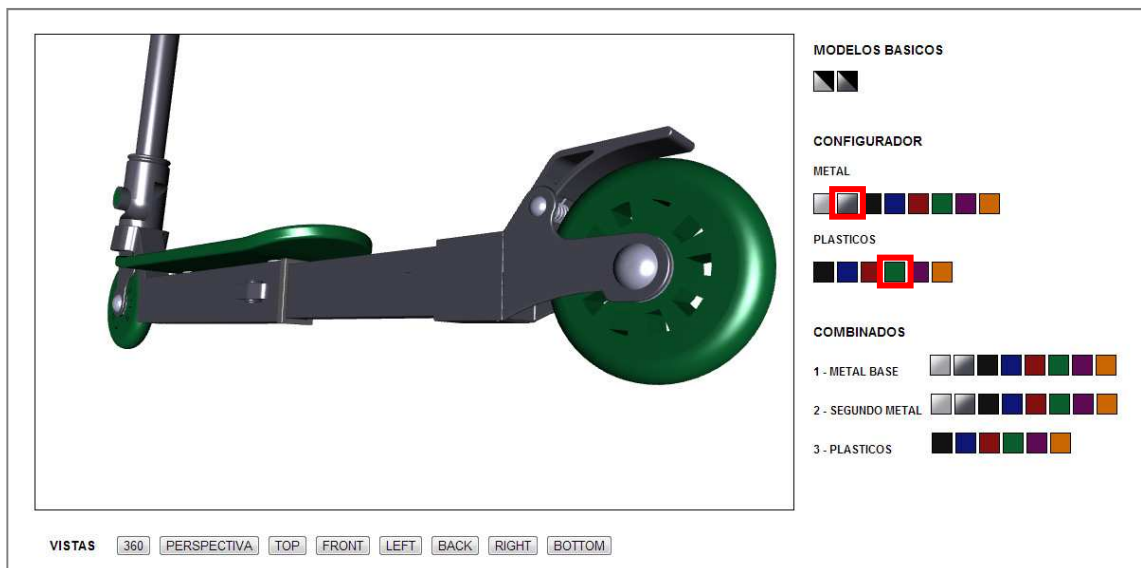
En el caso del patinete, tendremos una sección para modelos predefinidos; y otra donde el usuario puede hacer una configuración más personalizada. Al igual que en el ejemplo, lo que se cambiará, será el color difuso del material; dejando igual el resto de atributos.

Habrá que asignar un identificador para cada pieza, como vimos en el ejemplo. De esta forma, en el caso por ejemplo de los metales, la función será bastante extensa; ya que tendremos un comando para cada pieza.

Se ofrecen dos modelos básicos; en los que las partes no metálicas son de color negro; uno de ellos con el metal al natural, y otro con el metal oscurecido.



A continuación, se puede elegir entre distintos colores tanto para las piezas metálicas, como para las que no lo son; cambiando ambos grupos con un solo clic.



Por último, podremos combinar las partes metálicas en dos colores. Primero, elegiremos el color metálico base y a continuación el segundo metal; que corresponderá a partes como el cuerpo del manillar, los bloqueos de la dirección y las horquillas. Finalmente, se elige el color de las partes no metálicas.



MODELOS BASICOS
[Basic model icons]

CONFIGURADOR

METAL
[Metal color swatches: silver, black, blue, red, green, purple, orange]

PLASTICOS
[Plastic color swatches: black, blue, red, green, purple, orange]


COMBINADOS

1 - METAL BASE [Red box around silver swatch] [Black] [Blue] [Red] [Green] [Purple] [Orange]

2 - SEGUNDO METAL [Red box around black swatch] [Silver] [Blue] [Red] [Green] [Purple] [Orange]

3 - PLASTICOS [Red box around black swatch] [Blue] [Red] [Green] [Purple] [Orange]

VISTAS [360] [PERSPECTIVA] [TOP] [FRONT] [LEFT] [BACK] [RIGHT] [BOTTOM]



MODELOS BASICOS
[Basic model icons]

CONFIGURADOR

METAL
[Metal color swatches: silver, black, blue, red, green, purple, orange]

PLASTICOS
[Plastic color swatches: black, blue, red, green, purple, orange]

COMBINADOS

1 - METAL BASE [Red box around silver swatch] [Black] [Blue] [Red] [Green] [Purple] [Orange]

2 - SEGUNDO METAL [Red box around black swatch] [Silver] [Blue] [Red] [Green] [Purple] [Orange]

3 - PLASTICOS [Red box around black swatch] [Blue] [Red] [Green] [Purple] [Orange]

VISTAS [360] [PERSPECTIVA] [TOP] [FRONT] [LEFT] [BACK] [RIGHT] [BOTTOM]

Para los modelos básicos, necesitamos que a partir de una única muestra se activen dos funciones distintas; esto es que queremos todas las piezas de metal con un color, y las no metálicas con otro. Esto se hará llamando a las dos funciones a la vez en el mismo comando



MODELO BASICO CON METAL CLARO:

Todas las piezas metálicas en metal claro.

Todas las piezas no metálicas en negro.

```

```



MODELO BASICO CON METAL OSCURO:

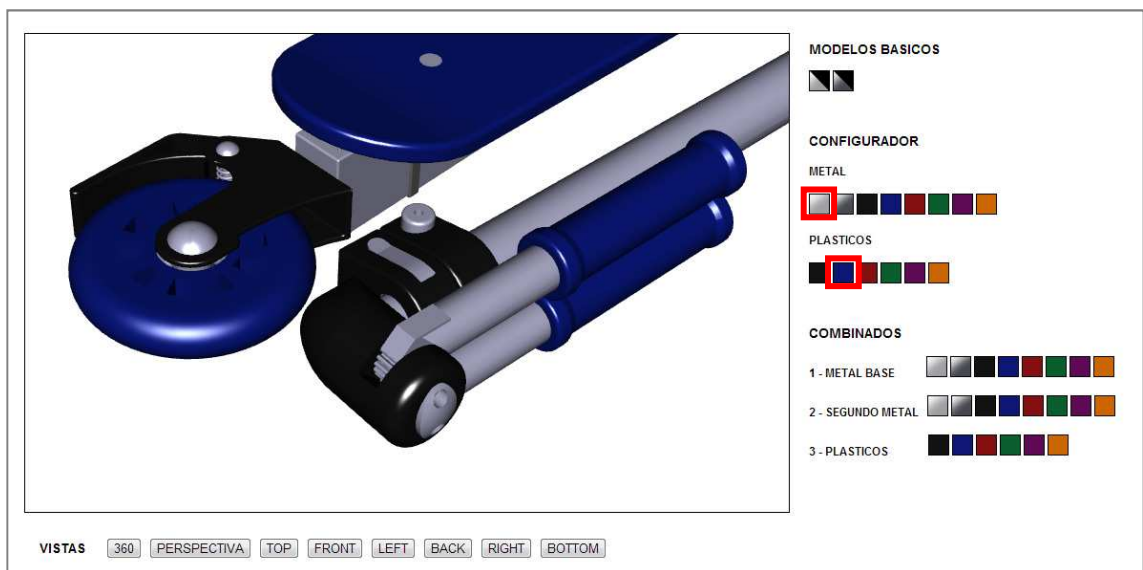
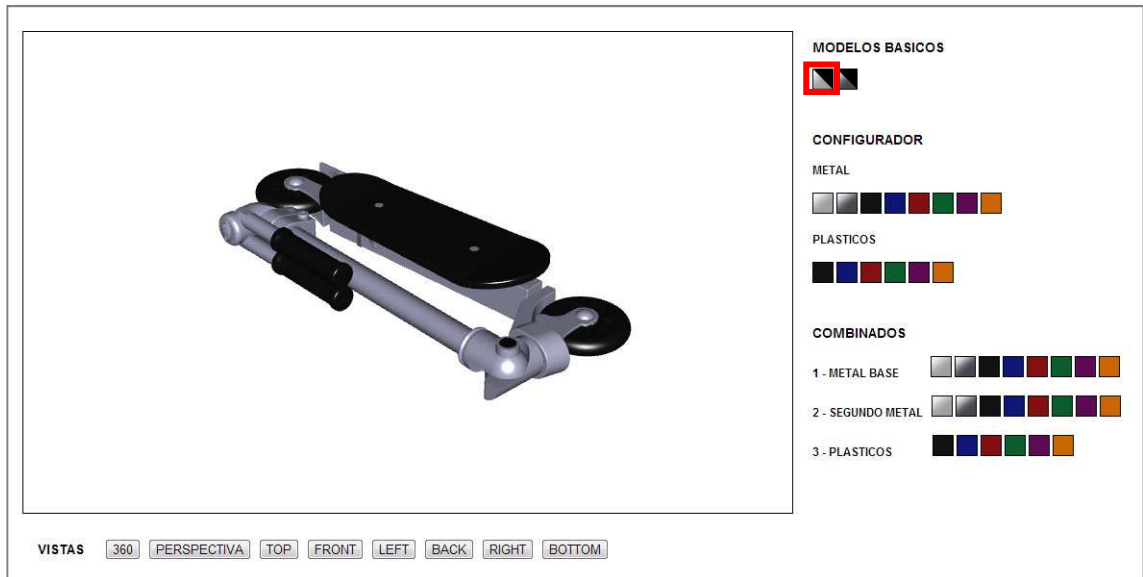
Todas las piezas metálicas en metal oscuro.

Todas las piezas no metálicas en negro.

```

```

Haremos lo mismo para el modelo plegado.





Recopilando las imágenes y vídeos realizados hasta ahora, se hace una página web que englobe y complete la presentación. Ésta, está hecha en HTML 5 y pretende poder visualizarse en cualquier dispositivo.

6.1-Recursos

A continuación se explican brevemente los recursos utilizados para el montaje de la página web.

6.1.1-HTML5, CSS3, etc

HTML5

Haremos uso de las nuevas etiquetas HTML5, explicadas en el apartado **5.1.1-HTML5**, para estructurar la página, integrar vídeo, y mejorar el uso de formularios, entre otros. En cuanto a los formularios, gracias al HTML5, podremos requerir que los campos sean cubiertos con un formato específico; por ejemplo, podremos requerir una dirección de email y si lo que se introduce en el campo, no tiene la estructura de email, no lo da por válido, avisando del error. Esto antes había que hacerlo recurriendo a programación con *Javascript*.

También se han aplicado transformaciones, por ejemplo a los botones del menú; en las que hay una transición con transparencia, para las que antes había que recurrir a *Adobe Flash*.

CSS3

La versión 3 de *Cascading Style Sheets*, o en español *Hojas de Estilos en Cascada*. Con ellas daremos formato a los distintos elementos de la página. Si lo comparáramos con un procesador de textos, el lenguaje HTML sería el texto en sí. Con las hojas de estilo CSS, definiríamos el color, el tamaño de letra, interlineado, etc.

Dependiendo de la hoja de estilos CSS aplica al HTML, podremos obtener resultados muy distintos visualmente, unos de otros. Es decir que un mismo contenido; el código HTML, puede adquirir diversos formatos y composiciones.

MODELO DE CAJAS

Es común el uso del modelo de cajas. Este consiste en asignar a cada elemento de la web, unas dimensiones con respecto a lo que hay tanto a su alrededor, como en su interior. En la siguiente imagen vemos un elemento o caja blanca de borde rojo.



Tenemos el **margin**, o *margen exterior*. Esto define la distancia de la caja a los elementos contiguos. Así mismo hay un *margen interior* o **padding**, el cual establece la distancia del **border** o borde de la caja, al contenido de esta.

A todos estos elementos se les puede dar formato gracias a los estilos CSS. Así podremos variar el grosor, color, tipo de línea del border; el color de fondo de la caja, hacer que las esquinas de la caja sean redondeadas, etc.

HEAD Y METADATOS

Dentro de la etiqueta **head** irán el título, y datos internos de la página; como el enlace a la hoja de estilos CSS, una descripción del contenido de la web, relación a otros archivos como por ejemplo el **sitemap** o a la miniatura **favicon** q aparece junto al título en el navegador, etc.

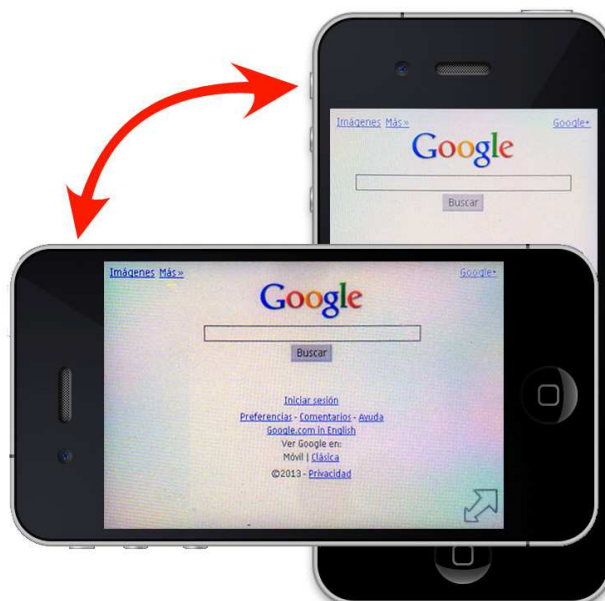
El uso de metadatos o etiquetas **meta**, ayuda al posicionamiento de la web en buscadores etc. Por ejemplo la etiqueta meta de descripción o la de *sitemap*. El sitemap, es un archivo con extensión **.xml**, el cual hace referencia a todas las secciones de que se compone la web. Esto ayudará a los motores de búsqueda a posicionar la web según su contenido.

Dentro del head también irán las funciones o referencias a bibliotecas externas, las funciones y contenidos de programación, en caso de necesitarlo; etiquetas **<script>...</script>**.

Debido al propósito del proyecto, destacaremos la siguiente equiqueta:

```
<meta name="viewport" content="width=device-width,initial-scale=1">
```

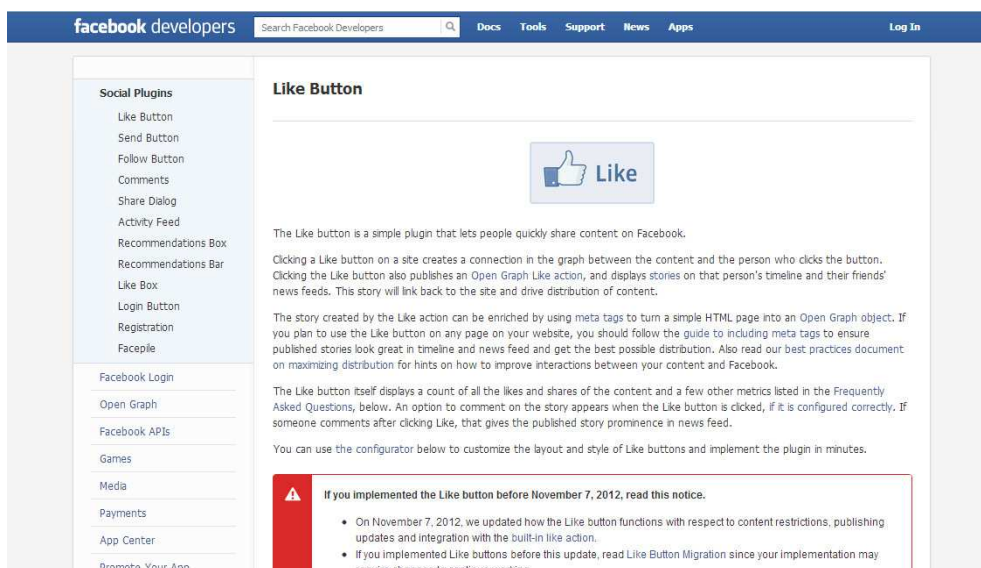
Esta etiqueta nos servirá para que la página se visualice correctamente en **tablets** o **smartphones**, tanto en orientación *landscape* (“paisaje”; horizontal o apaisado), como en *portrait* (“retrato”, vertical).



WIDGETS SOCIALES:

Los **Widgets Sociales**, son los botones que solemos encontrar en las páginas web, con los cuales podemos enlazar con las correspondientes redes sociales. De esta forma podremos seguir o dar “me gusta”, directamente haciendo clic sobre ellos, sin necesidad de ir a la página en la red social correspondiente.

Hoy en día el uso de estos botones es muy acusado, y las propias redes sociales tienen una sección para desarrolladores web, que proporcionan el código para integrar en una web. Normalmente tienen un configurador en el que se introduce la dirección de la página en cuestión, y a continuación se le da formato, eligiendo de entre diversas opciones.



Ejemplo de las páginas para configuración de widgets sociales en *Twitter* y *Facebook*.

6.1.2-jQuery

jQuery es una biblioteca JavaScript rápida, pequeña y rico en funciones. Permite que el control de eventos, manipulación y animaciones, más simple y con una **API** (*Interfaz de Programación de aplicaciones*), más fácil de usar y que funciona a través de multitud de navegadores. Da lugar a un uso de *JavaScript* más flexible. Utilizaremos jQuery en dos ocasiones para esta web:

FLEXSLIDER:

Un **FlexSlider**, también llamado *carrusel de imágenes*; es una zona definida en la que aparecen una serie de imágenes más o menos relevantes. Normalmente tiene unas flechas de avance y retroceso, y se puede además colocar una descripción de la imagen o comentario, además de imágenes como enlaces.



Como hemos visto, el script encargado de llamar a la librería correspondiente de jQuery, irá en la etiqueta **head** de la página. En este caso:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>
<script>
    !window.jQuery      &&                document.write("<script
    src='js/jquery.min.js'></scrip>");
</script>
<script src="js/jquery.flexslider.js"></script>
<script type="text/javascript">
    $(window).load(function() {
        $(".flexslider").flexslider();
    });
</script>
```

En esta ocasión, se incluye una función con la cual se invoca el JavaScript de forma local en caso de que falle la biblioteca online o la conexión a internet. Además es habitual cargar una hoja de estilos específica para el **FlexSlider**, con la que se le dará formato a la ventana, las flechas, pie de foto si lo lleva, tiempo que se visualiza la imagen o duración de la transición entre imágenes, etc. Esta hoja de estilos suele ir aparte de la hoja de estilos generales para la web. Esto irá también en el **head**.

```
<link rel="stylesheet" href="css/flexslider.css"/>
```

FANCYBOX:

Es un tipo de **Lightbox**; una ventana emergente que se abre para mostrar una serie de contenidos. Aunque generalmente se muestran imágenes, también se pueden incluir *películas swf de Flash* e *iframes*, entre otros.



Al igual que el FlexSlider, necesitará el correspondiente JavaScript, y su hoja de estilos.

```
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.4/jquery.min.js"></script>
<script>
    !window.jQuery && document.write("<script
    src='js/jquery.min.js'></scrip>");
</script>
<script src="js/jquery.mousewheel-3.0.4.pack.js"></script>
<script src="js/jquery.fancybox-1.3.4.js"></script>

<script>
    $(document).ready(function(){
        //fancybox
        $("a[rel=capturas]").fancybox({
            "transitionIn":"fade",
            "transitionOut":"elastic",
            "titlePosition":"over",
            "titleFormat":function(title, currentArray, currentIndex,
            currentOpts){
                return"<span id='fancybox-title-over'> Imagen "+(currentIndex+1)+"
                de"+currentArray.length +(title.length?"&nbsp; "+"title:")+"</span>";
            }
        });
    });
</script>
....

<link rel="stylesheet" href="css/fancybox.css"/>
```

Se incluye una función con la cual se invoca el JavaScript de forma local en caso de que falle la biblioteca online o la conexión a internet.

6.1.3- Responsive Web Design

El **Responsive Web Design**, puede traducirse como *Diseño Sensible, Fluido* o *Adaptativo*. Esta práctica o tendencia está orientada sobre todo a los dispositivos móviles, ya sean ordenadores portátiles, netbooks, tablets o smartphones. El Responsive Design, consiste en que la web se adapte a la pantalla del dispositivo en que se esté visualizando.



MEDIA QUERIES:

La adaptación de la web al dispositivo, se hará a través de las **Media Queries**. Estas son fragmentos de código, incluidos en la hoja de estilos CSS, que actúan como una función; modificando el estilo de los elementos y por consiguiente la configuración de la web, en función de un parámetro o variable. Generalmente, esa variable es la anchura en píxeles.

```
@media(max-width: determinada anchura en px){  
    etiqueta a la que afecta{  
        estilo1: valor;  
        estilo2: valor;  
    }  
    .....  
}
```


Dentro de las llaves de la Media Query, irán todas aquellas etiquetas, clases o id's; cuyos atributos se quieran modificar, en función de una determinada anchura. Dicha anchura, irá como variable de la función, dentro del paréntesis.

La idea es reestructurar la página en un tamaño más reducido, sin perder contenido. Se establecerá una organización diferente de los elementos componentes de la página, e incluso se eliminarán aquellos que no sean imprescindibles, siempre y cuando el contenido no deje de tener sentido y estar organizado.

Para todo esto, se establecerán unos anchos. Estos anchos se llaman **breakpoints**, y supone el punto de cambio de una estructura a otra. Por lo que tendremos una Media Query para cada breakpoint. Normalmente, a criterio del diseñador; los breakpoints tienden a ser un estándar en función de los dispositivos existentes, facilitando así el trabajo.

Esto, junto con la etiqueta meta **name="viewport"**; descrita antes y que supone que la página responda a la orientación del dispositivo, hacen que la web cuente con un diseño adaptativo.

6.2-Montaje final

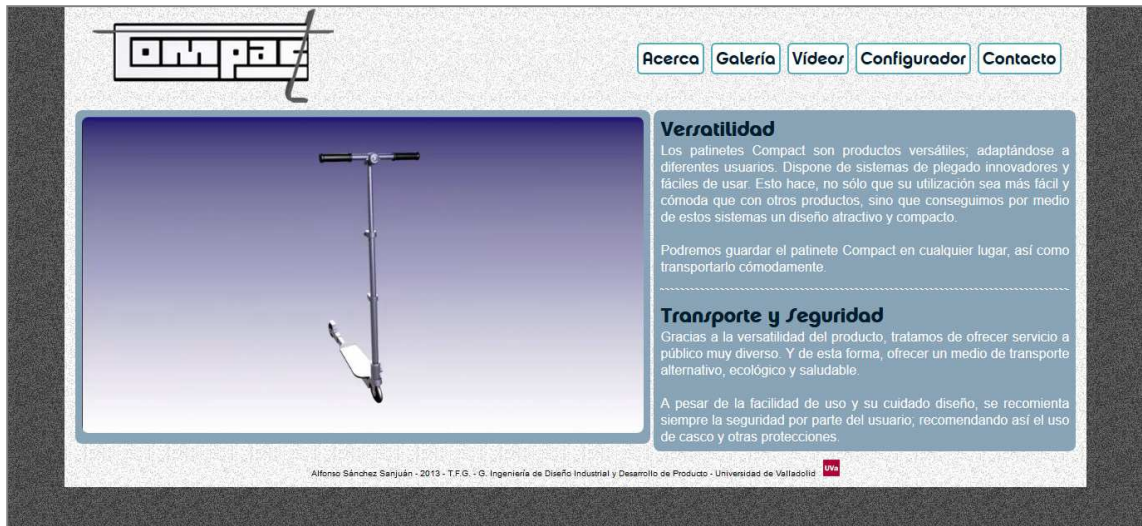
Para esta web, se han recopilado los contenidos descritos anteriormente; esto es las imágenes y vídeos, así como el modelo 3D interactivo. Todo ello, para dar a conocer una imagen representativa del producto. La estructura de la web es la siguiente:

INICIO:



Archivo *index.html*. En la página de inicio, encontramos un *FlexSlider*, con imágenes representativas del producto a modo de introducción. En este caso, ninguna de las imágenes actúa como enlace; se ha añadido un pie de foto y se han deshabilitado las flechas de avance y retroceso. Cuenta también con una sección en la que aparecen *Widgets Sociales* a diferentes redes.



ACERCA:

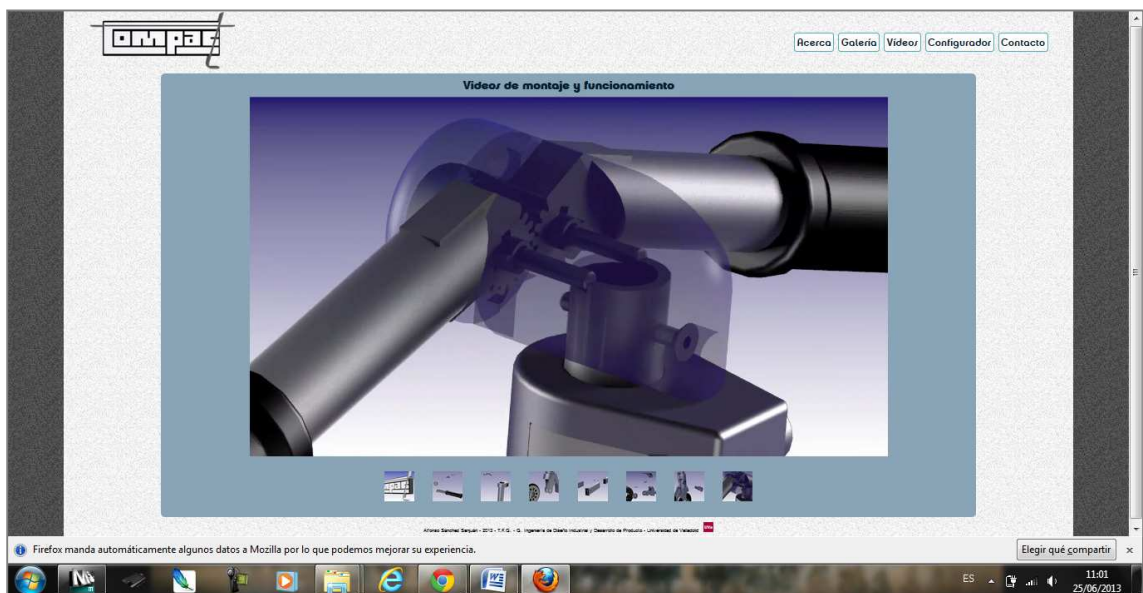
Archivo [acerca.html](#). Aquí veremos un video de la animación general de plegado, y unas descripciones en cuanto a características del patinete.

GALERIA:

Archivo [render.html](#). Galería de imágenes seleccionadas de entre los render exportados de *3d Studio Max*. Las imágenes se abren con una *Fancybox*, y algunas de ellas cuentan con una descripción al pie.

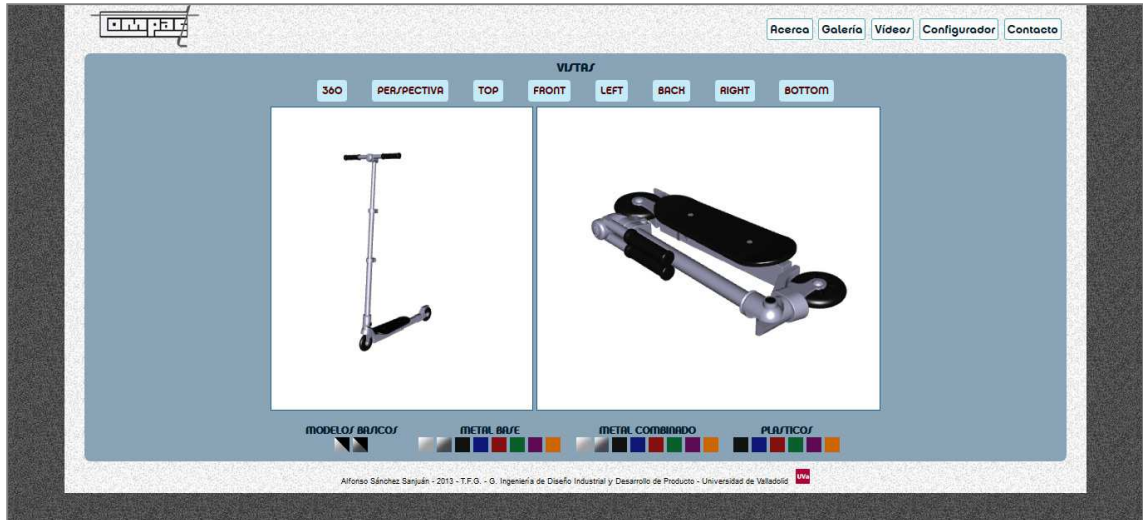


VIDEOS:

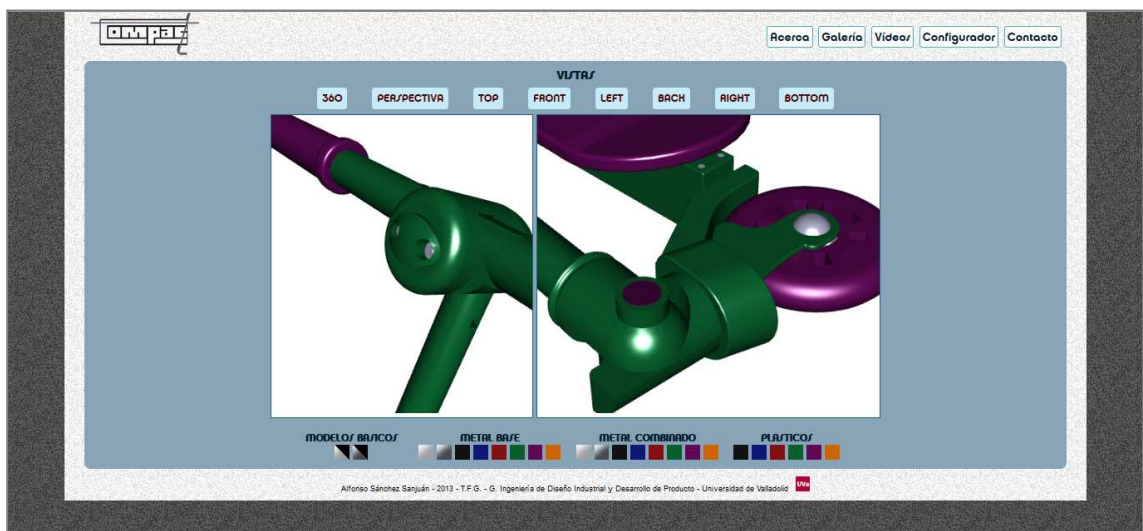


Archivo [animaciones.html](#). Galería vídeos de las animaciones realizadas con *3d Studio Max*. El video irá en una etiqueta `<video>...</video>`, de las que incorpora HTML5. En ella se abrirán todos ellos, utilizando una función que cambia la ruta del video, haciendo clic en la miniatura correspondiente.

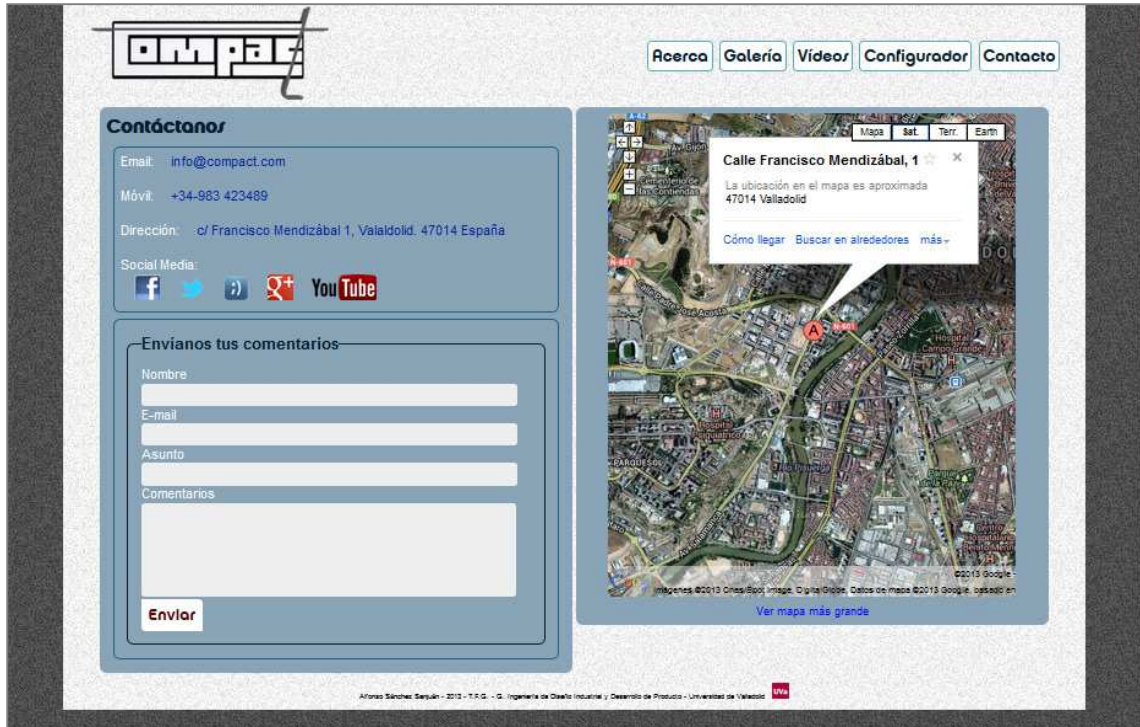
```
<video id="video" controls preload autoplay>
  <source src="vdo/VIDEO.mp4" type="video/mp4"/>
  .....
</video>
.....
<script type="text/javascript">
  function changeVideo(vdo)
  {
    document.getElementById('video').setAttribute("src", vdo);
  }
</script>
```

CONFIGURADOR:

Archivo *configurador.html*. Aquí encontramos el modelo 3D interactivo que hemos preparado para usarlo a modo de configurador, donde poder cambiar la combinación de color a nuestra elección. Se opta por mostrar a la vez el patinete plegado y extendido.



Los botones de cambio de vista y de color de las piezas, afectan a ambas ventanas; es decir tanto al modelo extendido como al plegado. Ambos son manipulables en todas sus vistas; pudiendo alejar y acercar el modelo, girarlo, etc.

CONTACTO:

Archivo *contacto.html*. En este último apartado, encontramos un formulario de contacto, un plano de situación, y enlaces a las páginas en diferentes redes sociales.

Como ejemplo, se han tomado para esta web, datos de las redes sociales de la Universidad de Valladolid; tanto para los enlaces, como para los widgets.



En este capítulo, se analiza el proyecto desde el punto de vista de los objetivos que se tenían inicialmente; pasando a valorar los resultados obtenidos en el desarrollo del mismo.

Como se describe en el segundo capítulo de esta memoria, partimos de un proyecto anterior. En él se hace el diseño de un producto; en concreto un patinete. Las premisas de dicho diseño, eran que el patinete fuera versátil, atractivo y que se plegara lo máximo posible.

Estas premisas se cumplieron en su momento, gracias al trabajo realizado:

- Se diseñó el producto conforme a la norma establecida (**norma UNE-EN 14619: Equipo de deporte sobre ruedas. Patinetes. Requisitos de seguridad y métodos de ensayo**), en cuanto a materia de seguridad; con sus respectivos ensayos de resistencia.
- Se hizo una exhaustiva descripción de cada una de las piezas que lo constituyen, así como de todo el conjunto, a través de diversos planos y modelos 3D.
- Se buscaron materiales empleados en productos similares, que cumplieran con los requisitos técnicos.
- Se describieron los procesos de fabricación y montaje.
- Se realizaron cálculos de producción, etc.

Con todo ello, obtuvimos un resultado muy satisfactorio en cuanto a los objetivos propuestos; consiguiendo un producto compacto y atractivo, seguro y de fácil utilización; mejorando a productos similares del momento y ofreciendo servicio a público muy diverso con un único producto.

Todo ello, dejaba patente el trabajo de Ingeniería realizado. Sin embargo, debido a las carencias de las herramientas gráficas utilizadas en aquel momento, no se pudo dar una imagen del producto tan vistosa como quizás requería. Este es el porqué del presente trabajo y premisa general del mismo. Hoy en día no es suficiente que un producto se diseñe correctamente; hay que saber darlo a conocer.

Para el análisis del presente trabajo, estableceremos cinco categorías o puntos principalmente. Dichos puntos pueden verse como los objetivos o parámetros para este trabajo.

7.1-Imagen

En este caso, mejoramos con creces la presentación estática del producto. La diferencia de las imágenes de las que partíamos, con los render que hemos obtenido, es más que notable. Partíamos de imágenes directamente del programa; es decir, del espacio de trabajo del programa, ya que éste carecía de la posibilidad de exportar imágenes de calidad. Ahora, disponemos de imágenes en las que se configura un entorno; una escena en la que se tienen en cuenta materiales y luces, con las que ofrecer una visual atractiva y realista. En las imágenes, se ofrecen diversas vistas del producto, mostrando éste en diferentes situaciones y posiciones; detalles e imágenes de funcionamiento.

Se ha utilizado un programa que, aunque habiendo trabajado con él en el aula, ha requerido un tiempo de adaptación, debido al tiempo de no utilización del mismo, y a la diferencia entre las versiones utilizadas entonces y ahora.

7.2-Video

Los vídeos de los que partíamos, mostraban lo propio; pero la forma de hacerlo era simplemente explicativa, no intervenía para nada el factor visual. Es decir, que no resultaban atractivos en comparación con lo obtenido en esta ocasión. Pasamos de animaciones exportadas del espacio de trabajo de un programa de diseño mecánico, en el que los movimientos están limitados (pensado en la representación de montajes o movimientos restringidos), a unos vídeos dinámicos y con gran carga visual. En ellos se recrean los movimientos propios del producto; para explicación del funcionamiento y montaje del mismo, pero también se animan las cámaras para ofrecer diferentes puntos de vista, e incluso se anima la opacidad de algunas piezas para mostrar lo que ocurre en su interior.

Se diferencian materiales en imágenes y vídeos. En las imágenes queremos representar el producto en sí, de forma muy vistosa, reproduciendo situaciones o dando detalles característicos del producto.

En las animaciones o vídeos sin embargo, se quiere claridad. Los materiales no son tan vistosos, pero no se necesita en este momento; se requiere una explicación gráfica del funcionamiento y montajes.

Hay que decir que el programa utilizado tanto para la obtención de imágenes como de vídeos, *3d Studio Max*; tiene un potencial enorme, y el manejo del mismo requiere de especialización.

7.3-Interacción

Este podría ser el punto fuerte del proyecto, ya que ha requerido más trabajo. Lo que se pretendía era obtener un modelo 3D interactivo a partir de archivos de *3d Studio Max*. Y en estos momentos, en que la tecnología 3D está en auge en gran diversidad de campos, parece casi imprescindible, ofrecer una presentación de producto 3D. Sumado a esto, la aparición de HTML 5, que se va implantando poco a poco, y que incorpora soporte para 3D en el nuevo elemento *canvas*, hace presagiar que el futuro de las páginas web; y en consecuencia la presentación de productos en Internet, tiende a ser en tres dimensiones.

Para el desarrollo de este punto, ha sido requisito el auto aprendizaje de diferentes formatos y tecnologías, desconocidas hasta el momento; iniciándose en HTML 5 y X3D. Trabajando con los archivos de los que ya disponíamos, hemos conseguido un modelo con el que poder interactuar, moverse libremente, cambiar vistas, y con la ayuda de una sencilla programación en javascript, poder cambiar propiedades a este modelo; en este caso, el color de las piezas.

Se han encontrado numerosos problemas en este proceso, ya que como se ha explicado, por razones desconocidas, la exportación de archivos no se llevaba a cabo de forma correcta en todos los casos. Esto requería manipular el código, y de alguna manera “montar” las piezas en el nuevo formato. Tarea complicada debido al desconocimiento de programación y comportamiento de estos formatos.

7.3.1-Incompatibilidad / Consenso en la WEB

El 3D en la web no es algo nuevo, ya que en 1995 se da a conocer la primera versión de VRML, y desde entonces esta tecnología sigue en desarrollo, pasando por diferentes formatos y múltiples aplicaciones, hoy en día. Sin embargo, no llega a establecerse totalmente. La causa de esto, probablemente sea la cantidad de formatos y soportes para los distintos navegadores. En este sentido ocurre lo mismo con el lenguaje HTML 5. Todavía existe incompatibilidad con algunos navegadores y se requiere de una programación web específica para unos u otros.

Todo ello podría solucionarse si se establecieran unos estándares comunes a todos los navegadores, con los que poder ofrecer la misma información en unos y otros. La diferencia entre ellos, debiera ser sólo visual y no de contenido; de esta forma dispondríamos de la información en cualquier momento, y situación a través de cualquier dispositivo.

También hay que tener en cuenta que HTML5 aún no está establecido, pero cada vez más páginas web están incorporando este nuevo lenguaje; sobre todo las grandes. Uno de los factores importantes que influyen en el cambio a HTML5 es la no necesidad de *Adobe Flash Player*, ya que el nuevo lenguaje incorpora video de forma nativa, sin necesidad de plugins o controladores.

7.3.2-3D a partir de un objeto real

Simplemente como dato, mencionar que cada vez se está extendiendo más la visualización en 360 grados de productos en tiendas online. Existen técnicas por medio de las cuales, a partir de un objeto real, se obtiene una simulación de giro 360 grados.

Material:

- **CAJA DE LUZ:** caja blanca de material translúcido en la que se coloca el objeto, y este es iluminado desde el exterior.
- **CAMARA**
- **MESA GIRATORIA:** dispositivo que gira el objeto en el interior de la caja de luz
- **SOFTWARE:** para la elaboración de la simulación
- **PRODUCTO**

La idea es que la cámara y la mesa giratoria están controladas por el propio software. De esta forma, el programa le indica a la mesa qué grados girar, mientras que a la cámara, le ordena cuando disparar. Una vez obtenidas todas las fotografías, se elabora el archivo de simulación.

Este método es muy conveniente, rápido y fácil de usar cuando se dispone del producto. En nuestro caso, al ser un modelo virtual, el proceso es totalmente distinto. Ya que hay que partir de dicho modelo y realizar distintas transformaciones. El proceso es no sólo muy diferente, sino mucho más elaborado.

7.4-Estrategia

Entre el proyecto de partida y el actual, hay un proceso también ingenieril, nada despreciable. Esta diferencia se debe a las herramientas empleadas y el trabajo realizado. El resultado obtenido, dista mucho del anterior. Se han conseguido imágenes y videos con gran impacto visual, que explican gráficamente las características y prestaciones del producto.

La presentación de producto es estratégica en Diseño Industrial; el éxito puede depender de cómo se venda o difunda el producto. Las nuevas tecnologías ofrecen unos canales de comunicación óptimos para la estrategia de Diseño Industrial. Habrá que identificar quien es el receptor de este mensaje, ya que la estrategia será diferente en unos casos y otros. Por ejemplo podemos hacer la siguiente diferenciación principalmente:

DISEÑADOR  **EMPRESA**

Por ejemplo en el caso de un diseñador *freelance*, que quiere darse a conocer a empresas para vender el producto, la información ha de ser lo más completa posible. En nuestro caso, toda la información sería necesaria; desde lo elaborado en este proyecto para dar una primera presentación e impacto visual, como la parte de ingeniería desarrollada en el proyecto anterior, para una descripción más técnica orientada a su producción, dado el caso.

(En este caso el diseñador tendrá que asegurarse de la protección del diseño, bajo patente o registro del mismo, para evitar la vulneración de los derechos de la propiedad intelectual e industrial)

EMPRESA  **USUARIO**

Si lo que se pretende es dirigirse a posibles compradores o usuarios del producto, la información ha de ser lo más visual y clara posible. Tendrá que ser una información conceptual, dejando claro qué es el producto, qué prestaciones ofrece y cómo se utiliza.

En este caso por ejemplo, no interesaría tanto, como se fabrica o monta el producto; pero sí como se usa, y que prestaciones ofrece. La mejor forma de ver esto, es gracias a videos e imágenes como las que se han elaborado en este trabajo. Cuanta más información se dé a cerca de un producto, más fácil será que el receptor de dicha información comprenda de qué se trata, y se interese por él. Más aún, cuando se carece de un producto físico que poder mostrar in situ. Es sobre todo, en este último caso, cuando se constata la importancia y utilidad de una representación 3D interactiva.

7.5-Web

Gracias al uso de las nuevas etiquetas de **HTML5**, a complementos como **FlexSlider** o **Fancybox**, y a la práctica de **Responsive Web Desig**; por medio de las **Media Queries**, así como la etiqueta meta **name="viewport"**; se consigue una web adaptable a la pantalla del dispositivo, tanto en anchura como en orientación.

La página en general, se visualiza en la mayoría de navegadores. El X3D en particular, no es compatible con algunos de ellos. Al igual que con el HTML 5, y como ya hemos visto, cada vez son más los navegadores que se están adaptando.

Con esta web, se completa el trabajo anterior; englobando todas sus partes y dando una imagen del producto, atractiva, dinámica y que se ajusta en la medida de lo posible, a las nuevas tecnologías y dispositivos actuales.



A continuación se hace una reseña de la bibliografía consultada para documentar el presente proyecto, clasificada por contenidos.

8.1-Antecedentes

Documentación sobre el producto objeto de la presentación desarrollada en éste proyecto, y fuente para el desarrollo del **capítulo 2** de esta memoria.

Proyecto “PATINETE COMPACT”.

Tutor: Jesús Magdaleno Martín

Alumnos: Héctor Núñez Gómez y Alfonso Sánchez Sanjuán

Escuela Universitaria Politécnica

Valladolid, Octubre de 2007

8.2-Render

Fuentes consultadas, como apoyo en el proceso de renderizado y uso de *3d Studio Max*; el cual se describe en el **capítulo 3** de esta memoria.

3D Studio Max 9

Instituto Europeo de Desarrollo y Tecnología; INEDETTEC S.L.

Copyright © 2008

Rendering With Mental Ray & 3ds Max

Joep Van Der Steen

Focal Press

El gran libro de 3ds Max 2010

©MEDIAactive

Barcelona : Marcombo, 2010

ISBN: 978-84-267-1628-6

3ds max 5

Ted Boardam (traductor, Fernando Ortega Escolar)

Anaya Multimedia, D.L. 2003

ISBN: 84-415-1521-2

Maxapuntes 3D

<http://maxapuntes3d.blogspot.com.es/>

Youtube (Diversos canales)

8.3-Video

Fuentes consultadas, como apoyo en el proceso de animación y uso de *3d Studio Max*; el cual se describe en el **capítulo 4** de esta memoria.

3D Studio Max 9

Instituto Europeo de Desarrollo y Tecnología; INEDETTEC S.L.

Copyright © 2008

Rendering With Mental Ray & 3ds Max

Joep Van Der Steen

Focal Press

El gran libro de 3ds Max 2010

©MEDIAactive

Barcelona : Marcombo, 2010

ISBN: 978-84-267-1628-6

3ds max 5

Ted Boardam (traductor, Fernando Ortega Escolar)

Anaya Multimedia, D.L. 2003

ISBN: 84-415-1521-2

PLYCOUNT

<http://www.polycount.com>

FORO 3D

<http://www.foro3d.com/archive/index.php/t-91463.html>

Youtube (Diversos canales)**8.4-HTML / HTML5**

Información acerca del lenguaje de marcado de hipertexto (HTML); tanto de versiones antiguas, como de la quinta y última, que se está introduciendo actualmente. Información sobre el consorcio World Wide Web, y otros datos acerca de la WEB. Datos que se reflejan en el **capítulo 5**.

HTML

<http://es.wikipedia.org/wiki/HTML>

World Wide Web

http://es.wikipedia.org/wiki/World_Wide_Web

World Wide Web Consortium

http://en.wikipedia.org/wiki/World_Wide_Web_Consortium

http://es.wikipedia.org/wiki/World_Wide_Web_Consortium

DOM (Document Object Model)

http://es.wikipedia.org/wiki/Document_Object_Model

¿Qué es el Modelo de Objetos del Documento?

<http://html.conclase.net/w3c/dom1-es/introduction.html>

Extensible Markup Language

http://es.wikipedia.org/wiki/Extensible_Markup_Language

XHTML

<http://es.wikipedia.org/wiki/XHTML>

MIT Media Lab

http://es.wikipedia.org/wiki/MIT_Media_Lab

SGML

<http://es.wikipedia.org/wiki/SGML>

API (Application Programming Interface) / Interfaz de programación de aplicaciones

<http://es.wikipedia.org/wiki/API>

8.4.1-HTML5**HTML5**

<http://es.wikipedia.org/wiki/HTML5>

HTML5

<http://theproc.es/>

Nuevos estándares en el desarrollo de sitios web: HTML5 y CSS3

<http://desarrolloweb.dlsi.ua.es/cursos/2012/nuevos-estandares-desarrollo-sitios-web/html5-nuevos-elementos>

Los nuevos elementos estructurales de HTML5

<http://mosaic.uoc.edu/ac/le/es/m8/ud2/>

Etiquetas en Html 5

<http://roarpc.blogspot.com.es/2012/01/etiquetas-en-html-5.html>

Nuevas tags HTML5

<http://software.intel.com/es-es/blogs/2011/12/07/nuevas-tags-html5>

Cuáles son las etiquetas nuevas del lenguaje HTML5, con una breve descripción sobre su utilidad y clasificación.

<http://www.desarrolloweb.com/articulos/nuevas-etiquetas-html5.html>

Youtube (Diversos canales)

8.5-Web 3D / 3D Interactivo

Fuentes de información al respecto de entornos 3D en la Web. Diferentes formatos y códigos. Documentación descrita en el **capítulo 5**.

Creating virtual 360 Panorama

<http://renderstuff.com/creating-virtual-360-panorama-cg-tutorial/>

Put 3D objects at your visitors' fingertips: UVaM on the iPad

<http://www.idea.org/blog/2011/11/29/put-3d-objects-at-your-visitors-fingertips-uvam-on-the-ipad/>

Scalable Vector Graphics

http://es.wikipedia.org/wiki/Scalable_Vector_Graphics

Web3D

<http://en.wikipedia.org/wiki/Web3D>

WebGL

<http://es.wikipedia.org/wiki/WebGL>

OpenGL

<http://es.wikipedia.org/wiki/OpenGL>

OpenGL ES

http://es.wikipedia.org/wiki/OpenGL_ES

SAI (Scene Access Interface) Tutorial

http://www.xj3d.org/tutorials/general_sai.html#SGWalk

8.5.1- X3DOM**X3DOM Instant 3D the HTML way**

<http://www.x3dom.org/>

About X3Dom

http://www.x3dom.org/?page_id=2

X3D Encoding Converter

http://doc.instantreality.org/tools/x3d_encoding_converter/

iX (magazine)

[http://en.wikipedia.org/wiki/IX_\(magazine\)](http://en.wikipedia.org/wiki/IX_(magazine))

ix. Magazin für professionelle informationstechnik

<http://www.heise.de/ix/>

https://www.heise.de/artikel-archiv/ix/2010/11/042_In-die-Tiefe-gehen

https://www.heise.de/artikel-archiv/ix/2010/11/054_Browser-t-raeume

https://www.heise.de/artikel-archiv/ix/2010/11/046_Freies-Malen

https://www.heise.de/artikel-archiv/ix/2010/12/116_Erdverbunden

8.5.2- VRML**X3D Encoding Converter**

http://doc.instantreality.org/tools/x3d_encoding_converter/

VRML Plugin and Browser Detector

<http://cic.nist.gov/vrml/vbdetect.html>

Applications, Players and Plugins for X3D / VRML Viewing

www.web3d.org/x3d/content/examples/X3dResources.html#Conversions

Applications, Players and Plugins for X3D / VRML Viewing

www.web3d.org/x3d/content/examples/X3dResources.html

VRML Plugin and Browser Detector / VRML and X3D Plugins, Programs, and Toolkits

<http://cic.nist.gov/vrml/vbdetect.html>

VRML Tutorials

<http://www.lighthouse3d.com/vrml/tutorial/>

Heriot Watt University / MSc DMIS

<http://www.macs.hw.ac.uk/~hamish/9ig2/>

Tutorial VRML 2.0

<http://www.di.ujaen.es/~rsegura/igai/vrml/documentos/tema5.htm>

8.5.3- X3D**X3D**

<http://es.wikipedia.org/wiki/X3D>

What is X3D?

<http://www.web3d.org/realtime-3d/x3d/what-x3d>

Getting Started with X3D

<http://www.web3d.org/realtime-3d/x3d/getting-started>

Player support for X3D components

http://www.web3d.org/wiki/index.php/Player_support_for_X3D_components

X3D International Standards

<http://www.web3d.org/x3d/specifications/>

X3D Encoding Converter

http://doc.instantreality.org/tools/x3d_encoding_converter/

Applications, Players and Plugins for X3D / VRML Viewing

www.web3d.org/x3d/content/examples/X3dResources.html#Conversions

Applications, Players and Plugins for X3D / VRML Viewing

www.web3d.org/x3d/content/examples/X3dResources.html

VRML Plugin and Browser Detector / VRML and X3D Plugins, Programs, and Toolkits

<http://cic.nist.gov/vrml/vbdetect.html>

Extensible 3D (X3D) 3.1 Tooltips en Español

<http://www.web3d.org/x3d/content/X3dTooltipsSpanish.html>

X3D Example Archives: Conformance Nist, Appearance, Material, rgb texture material transparency first

http://www.web3d.org/x3d/content/examples/Conformance/Appearance/Material/_pages/page21.html

SwirlX3D Tutorials

<http://www.swirlx3d.com/tut03.php>

8.6-Consultas generales

Fundamentos de Informática Gráfica

David Escudero Mancebo

Cano Pina,S.L Ediciones Ceysa