CAPÍTULO 4

“Modelado 3D de equipos de instrumentación electrónica disponibles en el Laboratorio de Instrumentación. Aplicaciones VRML de la interfaz virtual. Optimización de los modelos 3D”

1. Introducción

No nos resulta novedoso afirmar que Internet es uno de los grandes artífices de la Era de la Comunicación. Supone una herramienta potente para acercar la infinidad de información y recursos existentes en formato digital al alcance de todos. Y todo ello disponiendo de un punto de acceso a Internet y tras unos cuantos "clics".

Desde el Departamento de Ingeniería Electrónica de la Universidad de Sevilla se pretende potenciar el uso que se hace de esta herramienta, no ya solo encauzándola para conseguir un "expositor de información" online, sino intentando exprimir al máximo las posibilidades que nos ofrecen las múltiples aplicaciones de esta tecnología en auge.

Así, y como primer paso, la plataforma Moodle viene a completar las posibilidades que ofrecía el portal web de la asignatura "Laboratorio de Instrumentación Electrónica", añadiendo opciones de foros de debate, tutorías online a través de *chats*, registro de las personas relacionadas con la asignatura, ofreciendo la posibilidad de realizar un seguimiento del trabajo del alumnado a través de cuestionarios, encuestas, tareas grupales, etc... y pudiendo mantener la confidencialidad de los datos con la configuración de los módulos de seguridad de que dispone el entorno. Todo ello complementa un recurso virtual que dejaría de ocupar un segundo plano para el docente, abriendo ampliamente el abanico de posibilidades.

Sin embargo, no todo queda aquí. Tiempo atrás se inició una línea de investigación ambiciosa a todas luces, que pretendía acercar aún más el campo de la instrumentación electrónica al usuario de Internet. Se trata de una herramienta a través de la cual se le permitiría a cualquier alumno de la asignatura (o a cualquier persona con los permisos necesarios) conectarse a un servidor que ofreciese a su cliente un entorno determinado, habilitándolo para manipular remotamente equipos de instrumentación disponibles físicamente en el laboratorio.

Este proyecto de gran envergadura recopila numerosos aspectos técnicos que se han de cubrir antes de ponerse en funcionamiento. Todo ello para conseguir en un futuro implementar una red de instrumentación electrónica disponible a todos los usuarios de forma remota, suponiendo este hecho el acercamiento definitivo de unos recursos de laboratorio que por sus elevadísimos costes están al alcance de pocos.

1.1. Desarrollo del interfaz virtual

Dentro de los múltiples aspectos técnicos que involucran la implementación de una red de equipos de instrumentación electrónica, el más importante de cara al usuario final es el interfaz virtual. No hay que olvidar que el objetivo que se persigue es el de crear un entorno intuitivo y de fácil uso, que no le exija al usuario unos conocimientos técnicos más allá de los del uso que quiera hace de los equipos en cuestión. Es decir, hemos de suponer que el usuario sólo conoce el funcionamiento exclusivo del equipo de instrumentación,

independientemente de cómo éste se haya implementado en el entorno virtual.

Es por ello que uno de los objetivos principales es crear una interfaz intuitiva, realista y amigable, que invite al usuario a usar el equipo virtual como si del propio equipo físico se tratase.

La creación de un entorno 3D fiel al mundo real supone la solución ideal en nuestro cometido. Para ello, la "Realidad Virtual", de uso cada vez más extendido, se revela como la herramienta que habremos de trabajar para cubrir nuestros objetivos.

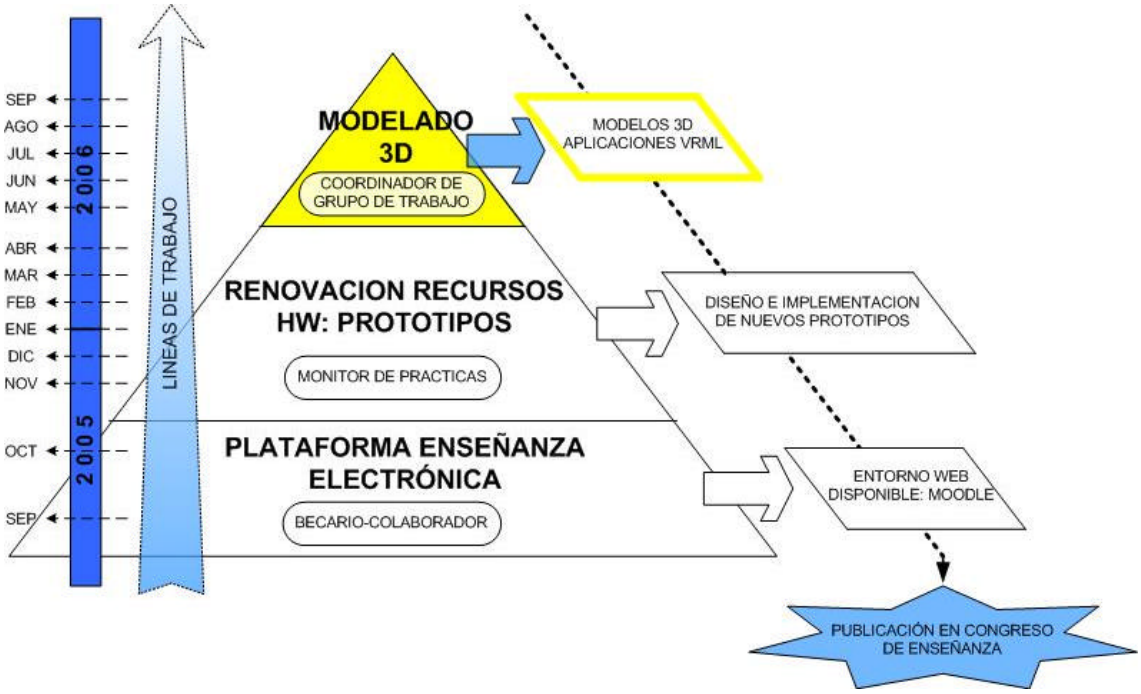


Ilustración 69: ***Tercera etapa de actuación del PFC***

A finales del mes de abril, y para cerrar un proyecto que ha introducido mejoras a la práctica totalidad de la asignatura "Laboratorio de Instrumentación Electrónica" en todos sus niveles, se me propone participar en la línea de investigación mencionada

diseñando y modelando en 3D una interfaz virtual realista de los equipos de instrumentación electrónica disponibles en el laboratorio con posibilidades de gestión remota. Para ello se me asigna como Coordinador de un grupo de trabajo formado por alumnos de la asignatura en el curso 2005 / 06, que me ayudaría a alcanzar los objetivos marcados, y que se desarrollarán a continuación.

2. Estudio previo al modelado 3D de equipos de instrumentación electrónica

El primer paso en la implementación de los modelos 3D requiere un estudio detallado de todos los aspectos, tanto técnicos como organizativos, para optimizar la consecución de objetivos:

2.1. Equipos a modelar

Por razones obvias, el principal requisito que se le exige a un equipo para ser modelado es que pueda ser controlado de forma remota, preferiblemente a través de un puerto GPIB.

Los equipos a modelar forman parte en su mayoría de una reciente adquisición de equipos del Departamento de Instrumentación Electrónica. Para comprobar sus especificaciones, se hizo necesario un estudio de los albaranes de compra y de los manuales de los equipos, comprobando para cada uno de ellos si permitían un control remoto de sus funciones.

Siguiendo estas directrices, los equipos seleccionados fueron:

? Generador de funciones TG2050

? Generador de funciones TG1010AGP

? Multímetro 1705-GP

? Fuente PL330TP

? Osciloscopio Digital WS422

? Generador de funciones Agilent 33220A

? Fuente Agilent E3631A

? Generador de funciones HP 33120A

? Fuente HP 3631A

2.2. Equipo de trabajo

Mi designación como Coordinador de un grupo de trabajo me permitió trabajar al frente de tres alumnos de la asignatura "Laboratorio de Instrumentación Electrónica", de modo que pudiéramos repartir tareas más eficientemente.

Se decidió en un principio realizar un reparto equitativo de los equipos a modelar entre los componentes del grupo de trabajo. Cada miembro del equipo de trabajo se comprometió a realizar los siguientes modelos:

? Rafael Inda Cervino: 2 equipos
  - o Generador de funciones TG1010AGP
  - o Fuente PL330TP

? Manuel Jiménez Castro: 2 equipos
  - o Osciloscopio Digital WS422
  - o Fuente HP 3631A

? Cristina Nieto Coronado: 2 equipos
  - o Multímetro 1705-GP

- Fuente Agilent E3631A

? Álvaro Mozo Casado: 3 equipos

- Generador de funciones HP 33120A
- Generador de funciones TG2050
- Generador de funciones Agilent 33220A

Sin embargo, el vencimiento de los plazos marcados para algunos de los miembros del grupo provocó que se alteraran los planes prefijados, que con la inclusión de una nueva alumna al grupo de trabajo dejó el reparto de equipos como sigue:

? Rafael Inda Cervino: 1 equipo

- Generador de funciones TG1010AGP

? Manuel Jiménez Castro: 1 equipo

- Osciloscopio Digital WS422

? Cristina Nieto Coronado: 1 equipo

- Multímetro 1705-GP

? María Dolores Cardoso: 3 equipos

- Fuente PL330TP
- Fuente HP 3631A
- Fuente Agilent E3631A

? Álvaro Mozo Casado: 3 equipos

- Generador de funciones HP 33120A
- Generador de funciones TG2050
- Generador de funciones Agilent 33220A

2.3. Decisiones técnicas del desarrollo 3D

El desarrollo de una interfaz tridimensional útil pasa por cumplir dos objetivos destacados por encima del resto:

- Los modelos 3D de los equipos han de ajustarse lo más fielmente posible a los equipos reales, procurando que el usuario pueda reconocerlos visualmente sin esfuerzo.
- El modelo final formará parte de un entorno 3D con múltiples elementos independientes. La optimización de su implementación es una prioridad, para evitar tiempos elevados de carga en el sistema que hagan inviable su uso interactivo.

Estos dos aspectos pueden ser antagónicos, ya que un mayor detalle en el modelado implica mayor tamaño y complejidad del sistema final, aumentando la sobrecarga del servidor. El estudio pasa por buscar una solución de compromiso que no requiera líneas de alta velocidad al usuario y que sea multiplataforma.

2.3.1. Lenguaje de modelado de Realidad Virtual: VRML

VRML (*Virtual Reality Modeling Language*) es la alternativa elegida para desarrollar nuestra interfaz en 3D. Se trata de un lenguaje de modelado de mundos virtuales en tres dimensiones a los que accedemos utilizando un navegador web.

Dentro de estos mundos desarrollados a través de VRML podremos navegar, e incluso interactuar, como si formásemos parte de él. Las ventajas que nos ofrece este modo de visitar sitios en

Internet son su intuitividad y "humanidad", ya que podremos movernos por los escenarios como lo hacemos en el mundo real.

Tal vez uno de los problemas (o virtud, según se mire) es tener que trabajar en líneas de comando. Esto supone una mayor laboriosidad a la hora de crear objetos 3D con un alto nivel de detalle, si lo comparamos con programas de diseño gráfico en 3D actuales. Por esta razón se decidió apoyar nuestro trabajo en programas de edición 3D que nos facilitase el trabajo de modelado, a costa de aumentar considerablemente el tamaño de los modelos de equipos posteriormente importados a VRML.

2.3.2. Diseño gráfico en 3D: Solid Edge

En el proceso de selección del software adecuado se barajaron tres posibilidades: Catia, 3D Studio y Solid Edge.

La limitación del tiempo de que disponíamos para hacernos con el manejo del software, unido a la sencillez que mostraba Solid Edge para obtener buenos resultados nos hizo decantarnos por este último. Además, algunos de los miembros del grupo de trabajo estaban familiarizados con su uso, debido a que se trata del software de edición 3D utilizado en la Escuela de Ingenieros de Sevilla para impartir la asignatura de libre configuración "Diseño gráfico 3D por computador", cursada por estos. Catia y 3D Studio son alternativas interesantes y muy potentes como software de diseño 3D, aunque los resultados obtenidos con Solid Edge confirman el acierto de nuestra decisión.

Solid Edge es un "sistema de definición de producto y modelado CAD" muy empleado en el diseño de maquinaria, gracias a su precisión

y a lo poderoso e intuitivo que resulta su entorno de modelado. La versión que hemos empleado es la V17 con licencia académica, la misma que se distribuye en la escuela a través de la asignatura antes mencionada.

Los modelos de los equipos que obtuvimos con Solid Edge se ajustan perfectamente al producto real, tanto como se quiera profundizar en el detallado. Nos hemos centrado en conseguir un alto nivel de detalle en el panel frontal de los equipos, así como en la estructura general del equipo, para que pueda ser reconocido en un entorno virtual sin esfuerzo.

Al finalizar el modelado del equipo, se exporta a VRML para que pueda ser visualizado en un navegador web.

2.3.3. Optimización del modelo 3D en VRML

El archivo creado a partir del modelo en Solid Edge exportado a VRML tiene la extensión .vml. El tamaño de dicho archivo es crítico, ya que de él depende directamente el tiempo de descarga del modelo entre servidor y cliente en el sistema final.

Más importante aún es el código fuente VRML que se haya generado tras la exportación. Descubrimos los defectos de apoyarnos en un programa de edición 3D al comprobar que todas las geometrías creadas con Solid Edge son consideradas "conjuntos de puntos" por VRML, con la consiguiente carga que supone el situar cada punto con sus coordenadas en el escenario virtual. Esto se traduce en movimientos poco fluidos del equipo modelado al querer trasladarlo o rotarlo en el entorno 3D. Si escalásemos este problema al diseño de

un laboratorio 3D con múltiples equipos, deducimos que la navegación a su través sería impracticable.

Por estas razones he buscado soluciones para optimizar los resultados partiendo de una codificación no optimizada del modelo del equipo, buscando alternativas a los elementos del equipo más *"pesados"* para el sistema (textos, *displays*, etc), probando programas optimizadotes de código (como Chisel) y compresores del archivo VRML (como Gzip). Los resultados obtenidos, comparándolos con los archivos VRML de los que partíamos, pueden ser considerados muy satisfactorios.

3. Modelado 3D de equipos usando Solid Edge

La principal razón por el que elegimos Solid Edge como software de edición 3D es la facilidad con la que se obtienen buenos resultados. El propio programa dispone de tutoriales para aprender paso a paso todas las posibilidades que ofrece, aunque no es necesario ver más que algunos de ellos para realizar el modelado de los equipos que nos ocupan.

Con miras a la posterior edición de botones que haremos en VRML (con opciones de *"linkado"* al pulsarlos) se decidió editar por separado los botones y el *display* de la estructura principal del equipo (cada elemento del equipo sería una **pieza sólida**: archivo de extensión .par), fusionándolos posteriormente en un equipo completo, una vez editados (formando un **conjunto**: archivo de extensión .asm).

3.1. Diseño de piezas sólidas

Una vez instalado Solid Edge, abrimos el programa y empezamos a editar cada elemento del equipo empezando por la estructura principal o "caja", para tenerlo de referencia a la hora de implementar el resto de elementos. Seleccionamos en el menú principal *Pieza sólida* y se nos mostrará un espacio 3D con los 3 planos de referencia (XY, XZ e YZ). Éste será nuestro campo de trabajo de ahora en adelante.

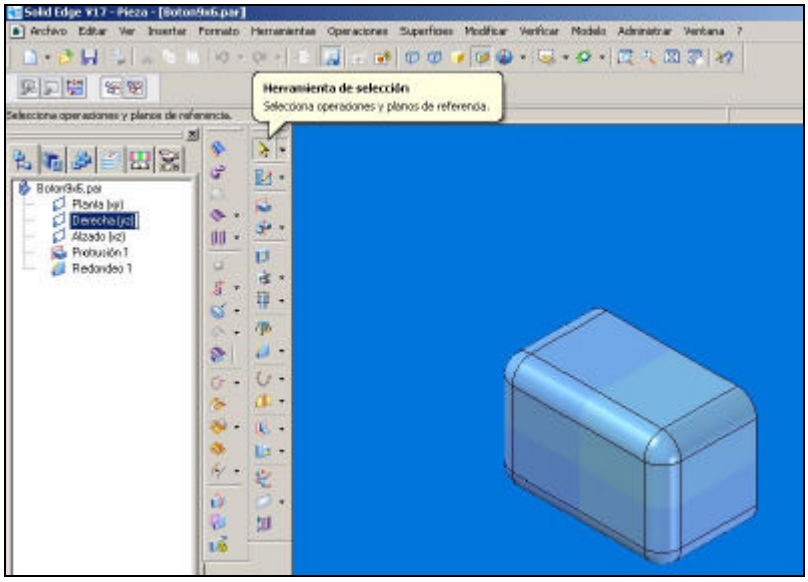


Ilustración 70: ***Pantalla principal de Solid Edge: Edición de Pieza***

En la ventana de la izquierda se nos muestran varias pestañas de las que en principio solo usamos las dos primeras: *Pathfinder de Operaciones,* que nos mostrará todas las operaciones que vayamos realizando en el espacio 3D, y *Biblioteca de Operaciones*, que nos mostrará el árbol de directorios de nuestro disco duro para cargar elementos.

En la ventana de la derecha tenemos el espacio 3D sobre el que editaremos nuestras piezas, para posteriormente unirlas en un equipo completo. Es muy importante familiarizarse con las vistas, pudiendo modificarlas a nuestro gusto para facilitarnos el diseño. Podemos hacer zoom directamente con la rueda del ratón, incluso rotar la vista presionando dicha rueda y desplazando el ratón. En la barra de herramientas superior están disponibles botones destinados a la visualización: *Área de zoom, Ajuste, Encuadre*… también se puede modificar el modo de visualización para facilitar ciertas operaciones más complicadas: mostrando y ocultando bordes o caras de la pieza. Por último, puede ser importante modificar y reducir la resolución del equipo con el botón *Mejorar presentación*, si estamos trabajando con un ordenador de inferiores prestaciones.

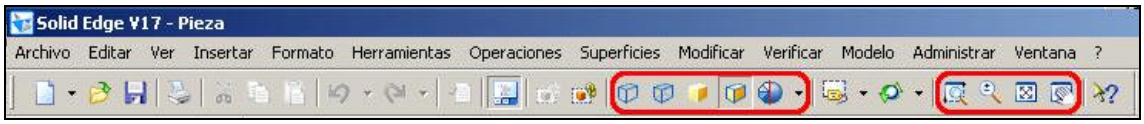


Ilustración 71: ***Barra de herramientas superior de Solid Edge: Edición de Pieza***

Entre ambas ventanas se encuentra la barra de herramientas en la se incluyen todas las operaciones que necesitaremos para modelar las piezas. Una de las mayores ventajas de Solid Edge sale a relucir al activar cualquiera de dichas operaciones: En la parte superior de la ventana se nos irá indicando cada una de los pasos que habrá que seguir para finalizar correctamente la operación seleccionada, guiándonos a modo de tutorial interactivo.

Aprovecharemos esta posibilidad para implementar las siguientes operaciones básicas disponibles, que son las que hemos utilizado para modelar nuestras piezas:

- *Protusión:* seleccionando un plano, realiza una protusión sólida de las dimensiones que se configuren. Operación básica para la creación del cuerpo de cualquier pieza.
- *Vaciado*: se opera igual que con las protusiones, salvo que en este caso se obtiene la operación inversa: de vaciado.
- *Redondear / Achaflanar:* utilizado para moldear las protusiones, ajustándolas al modelo real de la pieza.
- *Copia Simétrica de la Operación:* muy utilizado a la hora de repetir operaciones idénticas, agilizando operaciones como el vaciado de los botones de teclados en el panel frontal de los equipos.
- *Patrón a lo largo de curva:* operación parecida a la anterior, pero con más posibilidades. Su potencia radica en que no está limitada a simetrías, como en el caso anterior.
- *Boceto:* útil a la hora de diseñar líneas guía para la disposición de elementos a lo largo de una misma superficie. No tiene repercusiones físicas en el modelado del equipo.

Durante la edición de protusiones y vaciados (operaciones básicas) surge una nueva barra de herramientas que nos facilitará el diseño de geometrías de diversa índole para posteriormente

protusionarlas o vaciarlas: líneas, arcos, circunferencias, rectángulos… con múltiples métodos de construcción y otras opciones para ajustar las geometrías según ciertas restricciones configurables: acotaciones de distancias, recortes, paralelismos, perpendicularidades, colinealidades, conexión de vértices, geometrías tangentes, circunferencias concéntricas, patrones… todas estas opciones permiten dar un salto de calidad en la precisión de nuestras piezas, debiendo tenerse muy en cuenta a la hora de diseñarlas.

Existe otra herramienta denominada *Pintor de Piezas,* disponible en la barra de menú superior (Formato > Pintor de Piezas) que nos permite editar el color de las piezas o de alguna de sus superficies, ajustándolo aún más al modelo real.

3.2. Diseño del conjunto

Una vez modelados todos los componentes del equipo en cuestión, habremos de fusionarlos para formar el equipo completo. Para ello, volvemos al menú principal del programa y seleccionamos en este caso *Conjunto.*

La disposición de las ventanas es la misma que antes, variando las operaciones disponibles en la barra de herramientas *Operaciones*. De todas las nuevas operaciones que encontramos, el único botón que necesitaremos para nuestro cometido es el de *Ensamblar*.

El ensamblaje correcto exige que entre las piezas a ensamblar se definan al menos 3 puntos de anclaje, los cuales pueden definirse como planos, aristas o vértices, según las opciones de ensamblaje que elijamos en *Tipo de Relaciones* (se verá mas adelante). En el caso de no cumplir esta premisa, las piezas no quedarán correctamente ajustadas y Solid Edge lo indicará cuando selecciones la pieza en el *Pathfinder del conjunto* con el mensaje: *"La ocurrencia X está infrarrestringida".*

Para facilitar el ensamblaje de las piezas en la estructura principal del equipo, se decidió modelar el panel frontal del equipo con unos vaciados correspondientes a cada botón o elemento que se haya de insertar, con sus mismas dimensiones para ajustarlos correctamente.

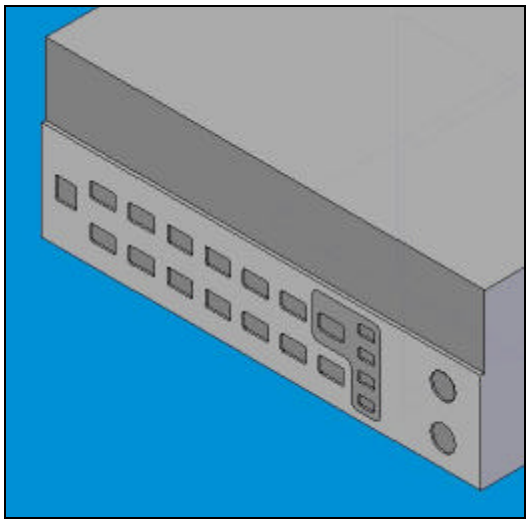


Ilustración 72: ***Detalle de los vaciados realizados en el panel frontal del equipo***

Para empezar, buscaremos en la *Biblioteca de piezas* la que corresponde con la estructura principal del equipo, o "caja". Haciendo clic sobre ella y arrastrándola a la ventana del entorno 3D situada a la

derecha, conseguimos que la "caja" se ajuste a los planos de referencia. Solo faltaría ensamblar el resto de piezas (botones, *display*, etc…) a la "caja" para disponer del conjunto montado y el equipo completo.

El proceso de ensamblado de piezas es como sigue:

- Arrastramos la pieza deseada de la *Biblioteca de piezas* a la ventana del entorno 3D.
- Pulsamos *Ensamblar*. Se desplegará una barra de herramientas superior ofreciéndonos varias posibilidades de ensamblado, eligiendo la que deseemos en el botón *Tipo de Relaciones.*
  - Se recomienda emplear el tipo de relación *Conectar,* estableciendo el modo de visualización del entorno (barra superior, mencionado anteriormente) en *Bordes visibles y ocultos,* tratando de ensamblar vértices de las piezas entre sí. Nos facilitará mucho el proceso, ya que es más sencillo seleccionar vértices que superficies.

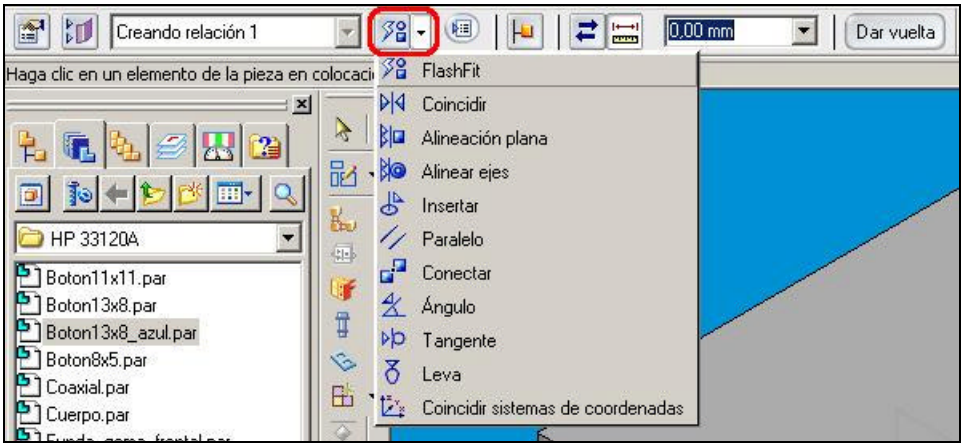
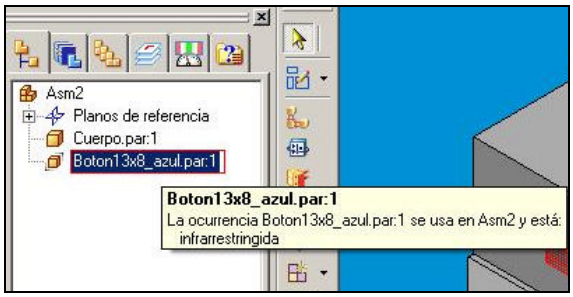


Ilustración 73: ***Menú desplegable de "tipos de relación" entre superficies***

? Seleccionado el tipo de relación que queramos, hacer clic consecutivamente en las superficies, aristas y/o vértices que queramos ensamblar, estableciendo un mínimo de tres anclajes para fijar correctamente las piezas entre sí.



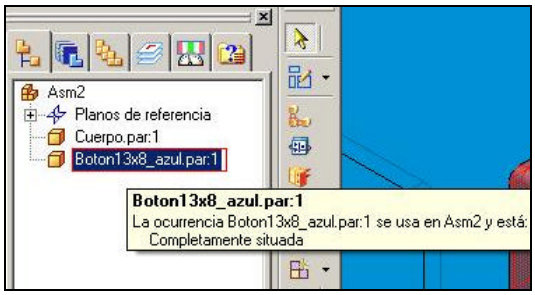


Ilustración 74: ***Anclaje de pieza: "ocurrencia infrarrestringida"***

Ilustración 75: ***Anclaje de pieza: "ocurrencia completamente situada"***

? Si el anclaje es correcto, al seleccionar la pieza en el *Pathfinder de operaciones* obtendremos el siguiente mensaje: *"Completamente situada".*

? Repetimos el proceso con todas las piezas del equipo hasta completar su montaje.

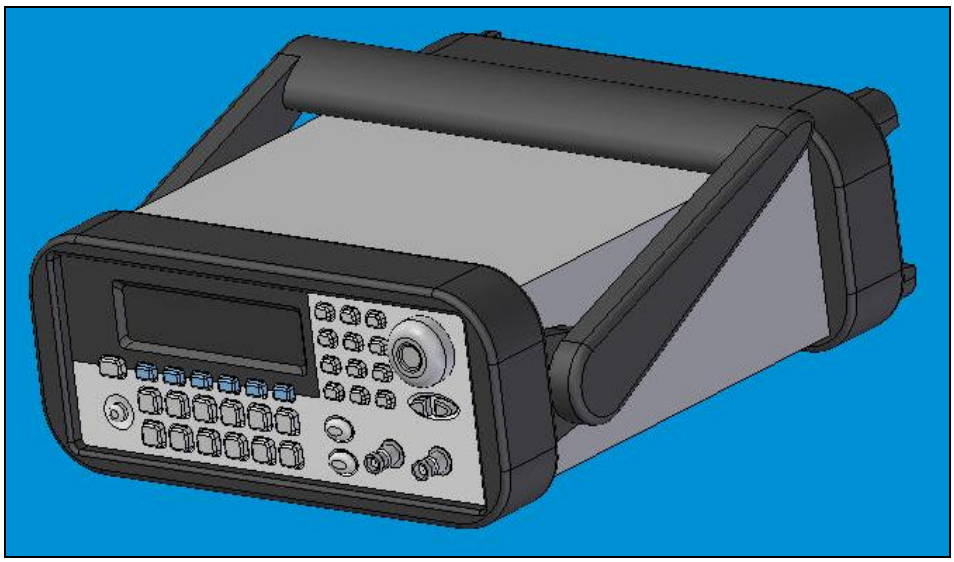


Ilustración 76: ***Equipo de instrumentación electrónica Agilent 33220A, modelado en 3D mediante Solid Edge***

3.3. Guardar el conjunto creado. Exportación a VRML

Finalizado el montaje del equipo con todas sus piezas, solo falta guardar los cambios (Archivo > Guardar como…), para finalmente disponer de nuestro equipo de instrumentación electrónica modelado en 3D con Solid Edge.

Tomando como ejemplo el equipo Agilent 33220A, podemos observar que el tamaño de los archivos que lo componen (.par, .cfg y .asm) superan los 5,22 MB, sin haber incluido textos (veremos posteriormente por qué), lo que nos da una idea de la carga que supondría a un sistema trabajar con un laboratorio virtual repleto de equipos similares. Sin embargo, nuestra intención es trabajar en el formato VRML, por lo que exportaremos el equipo primeramente a VRML, comprobando los resultados que obtenemos.

Para ello, una vez abierto el archivo que almacena el equipo al completo (de extensión .asm), seleccionaremos Archivo > Guardar imagen como… estableciendo el tipo de archivo "Documento VRML", de extensión .wrl. Con ello obtenemos un archivo de un tamaño muy inferior: 818 KB, bastante más manejable de cara a un sistema cliente-servidor, a falta de ser editado en VRML y comprimido.

4. Edición del código VRML

4.1. Edición del código VRML generado a partir de los modelos Solid Edge

VRML fue desarrollado a modo de estándar para la Realidad Virtual en Internet. En este sentido, no requiere más que el bloc de notas para crear mundos virtuales y un navegador web para visualizarlos. Sí se hace necesario instalar un pequeño módulo o *Plug-in* en los navegadores más habituales (Explorer, Mozilla, Opera, etc…) para visualizar mundos VRML sin problemas. Así mismo, existen editores de texto VRML con numerosas utilidades para programar mundos virtuales.

En el caso que nos ocupa, se descargó el programa Cortona Versión 4.2, desarrollada por la empresa Parallel Graphics y de distribución gratuita, habilitando al navegador web Explorer para acceder a mundos VRML. También se procuró la instalación del editor de textos VrmlPad Versión 2.1 (versión completa), ya que la versión de prueba no permitía guardar cambios en archivos de extensión mayor a 64KB, ni copiar o cortar selecciones de más de 32K al portapapeles, con sesiones limitadas a 2 minutos. La elección de VrmlPad como editor de textos se debe a que cumple con las características que buscamos en un editor de este tipo: soporte avanzado de búsquedas y reemplazos, ayuda para la sintaxis, autocompletado de funciones y detección dinámica de errores.

4.2. Análisis del código fuente original

Instalado el software necesario para trabajar con VRML, nuestro primer paso será abrir el archivo que contiene el equipo modelado en Solid Edge, recién exportado a VRML: *Agilent 33220A.wrl*. Estudiando el contenido de sus líneas de código, podremos analizar los puntos sobre los que tendremos que actuar añadiendo, modificando o eliminando código, según los objetivos buscados.

Tomando un extracto del código fuente original (el código completo tiene 11114 líneas), podemos hacernos una idea de su estructura:

```
#VRML V2.0 utf8 Solid Edge VRML Export V1.2
```

La línea de cabecera informa de que está recibiendo un archivo de VRML, la versión correspondiente del lenguaje (la versión 2.0. es sensible a mayúsculas y minúsculas), **utf8** indica al navegador qué estándar de cadenas de texto debe emplear. Por último nos hace referencia a la procedencia del código fuente (exportado de Solid Edge).

```
Group {
 children [
```

Un mundo VRML está hecho a partir de nodos, que son tipos de objeto. Dentro de estos nodos existen campos, que son propiedades del objeto. En este caso el código empieza con un nodo *Group*, que es el nodo más simple y lo único que hace es agrupar otros nodos. En su interior tenemos un campo *children* que puede contener otros nodos, anidándose uno debajo de otros formando una jerarquía de nodos.

```
WorldInfo { title "Produced using Solid Edge VRML Export, by UGS PLM Solutions." }
```

El primer nodo que aparece es el nodo *WorldInfo***,** que contiene información general sobre el entorno 3D. Éste se despliega en la barra del título de la ventana del navegador, pudiendo contener más información sobre el archivo.

```
DEF Main Viewpoint {
 position 18.9 -21.9 9.79
 orientation    0.848 0.373 0.377 1.36
 fieldOfView    0.00793
 description "Main View"
}
DEF Top Viewpoint {
 position 0 0.0065 27.9
 orientation    1 0 0 0
 fieldOfView    0.00793
 description "Top View"
}
DEF Bottom Viewpoint {
...
```

En primer lugar tenemos una serie de nodos que comienzan con la cláusula DEF. Hacen alusión a la definición de nodos para su posterior reutilización, sin necesidad de redefinirlas, empleando en esta ocasión la cláusula USE.

En segundo lugar, los nodos *Viewpoint* que se están definiendo implementan cada uno un punto de vista desde el cual el usuario puede situarse por defecto: vista principal, vista superior, inferior… Para alternar de uno a otro, solo hay que hacer clic sobre las flechas

que hay en la barra de herramientas inferior de la pantalla, una vez que estemos visualizando el entorno 3D desde un navegador.

```
NavigationInfo {
 type [ "EXAMINE","ANY" ]
 headlight TRUE
}
```

El nodo *NavigationInfo* se configura para establecer el modo de navegación del usuario a través del entorno: usuario inmóvil con posibilidad de rotar la vista, sin limitación alguna, etc.

```
Background { skyColor 1 1 1 }
Group {
 children [
  DEF RightLight DirectionalLight {
        direction -0.632455 0.547723 -0.547723
        intensity  0.35
  }
  DEF LeftLight DirectionalLight {
        direction  0.632455 0.547723 -0.547723
        intensity  0.35
  }
  DEF BackLight DirectionalLight {
        direction  0 -0.707107 -0.707107
        intensity  0.35
  }
  DEF BottomLight DirectionalLight {
        direction  0 0 1
        intensity  0.35
```

```
}
```

El primer nodo que tenemos es *Background*, que determina el color del fondo de nuestro entorno.

El segundo nodo es un *Group* anidado, que vuelve a abrirse para recoger nuevos nodos. Entre ellos, los cuatro primeros son definiciones de nodos *DirectionalLight*, que establecen iluminación unidireccional y uniforme (como la luz solar) que incide sobre todo aquel nodo que sea *child* de su nodo padre (en este caso, todos los Shapes que vendrán a continuación, es decir, todo el equipo modelado).

```
Shape {
 appearance DEF S0x80000009 Appearance {
  material Material {
   ambientIntensity 0.2
   diffuseColor 0.2 0.2 0.2
   emissiveColor 0.0471 0.0471 0.0471
   specularColor 0.243 0.243 0.243
   shininess 0.056
   transparency 0
  }
 }
 geometry IndexedFaceSet {
  solid FALSE
  normal Normal { vector [ .707 -.707 0,.707 -.707 0,.707 -.707 0,.707 -.707 0,.707 -.707 0 ] }
  coord Coordinate { point [ -.0854 -.1299 .0498,-.0854 -.1299 .0751,-.0839 -.1284 .0736,-.0824 -.127 .0528,-.0824 -.127 .0721 ] }
  coordIndex [ 1,0,2,-1,3,2,0,-1,2,3,4,-1 ] }
```

```
}
Shape {
appearance USE S0x80000009
geometry IndexedFaceSet {
solid FALSE
normal Normal { vector [ 0 -.707 -.707,-.279 -.804 -.524,-.295 -.806 -.511,-.511 -.806 -.295,-.524 -.804 -.279,-.707 -.707 0,0 -.707 -.707,-.707 -.707 0,-.279 -.804 -.524,-.295 -.806 -.511,-.511 -.806 -.295,-.524 -.804 -.279,-.707 -.707 0 ] }
coord Coordinate { point [ .0116 -.127 .075,.0131 -.127 .0746,.0132 -.127 .0746,.0141 -.127 .0737,.0141 -.127 .0736,.0145 -.127 .0721,.0146 -.1299 .0779,.0159 -.1284 .0736,.0161 -.1299 .0776,.0162 -.1299 .0776,.0171 -.1299 .0767,.0171 -.1299 .0766,.0174 -.1299 .0751 ] }
coordIndex [ 7,12,11,-1,10,9,8,-1,5,7,4,-1,11,4,7,-1,4,11,3,-1,2,3,11,-1,11,10,2,-1,8,2,10,-1,2,8,1,-1,6,1,8,-1,1,6,0,-1 ] }
}
Shape {
...
```

Por último, comprobamos que de la línea de código 95 a la 11114 nos encontramos cientos de nodos *Shapes*, cada uno de los cuales define una superficie de todas las que conforman el equipo modelado en Solid Edge.

En el primer nodo se define el primer campo, *appearance,* para ser reutilizado en todos los nodos *Shape* cuyas superficies tengan sus mismas características físicas: brillo, transparencia, color, textura, etc. El segundo campo, *geometry*, hace referencia a la geometría de la superficie, y es aquí donde resaltan los defectos de exportar el modelo desde Solid Edge. Geometrías sencillas como un cuadrado o una

circunferencia son tratadas como figuras irregulares, siendo definidas como un conjunto de múltiples puntos, sobrecargando el archivo con miles de líneas de datos presumiblemente innecesarios.

Otro defecto, derivado directamente del anterior, se debe a que multitud de superficies del equipo idénticas entre sí no son predefinidas para ser reutilizadas con la cláusula USE, ya que como hemos visto en el código, no se reconoce a las superficies como tales. Se pierde así una oportunidad de reducir enormemente el tamaño del código, puesto que lo único que se reutiliza es el campo *appearance* de los *Shapes*.

4.3. Objetivos perseguidos y edición del código fuente

Una vez interpretado el código VRML que definía al equipo, se intentaron alcanzar una serie de objetivos funcionales y estéticos:

4.3.1. Inclusión de los textos informativos en el panel frontal de los equipos

La experiencia de incluir los textos como protusiones del equipo en Solid Edge resultó ser una mala idea, como pudimos comprobar en el tamaño de los archivos obtenidos en Solid Edge (35,5MB, 7 veces mayor que sin textos) y en su exportación a VRML (3,51MB, más de 4 veces mayor que sin textos), con la consiguiente pérdida de interactividad por sobrecarga al visualizarlo en un navegador web.

Se analizaron dos alternativas para incluir textos en VRML:

? Como texturas aplicadas a las superficies de los botones: Se editaron los textos como imágenes de extensión .JPG

(almacenados en el directorio "Textos") y se aplicaron como texturas a las superficies frontales de los botones. El código del nodo Shape referente a la superficie con textos quedaría así:

```
Shape {
  appearance Appearance {
   texture ImageTexture {
     url "Textos/Mod.jpg"
       }
 }
 ...
 }
```

En contra de esta medida tenemos que el tiempo de carga de las imágenes es algo mayor que el del equipo (2-3 segundos mayor), y que los efectos de iluminación sobre la superficie del botón no se aplican a las imágenes, que habría que tratarlas por separado. Además, estamos creando nuevos archivos (de imagen) de los que dependerá la correcta visualización del equipo, aumentando también el tamaño en memoria total del equipo.

A favor, que no es necesario manipular coordenadas, rotaciones ni traslaciones para situar los textos correctamente sobre los botones.

- Directamente como textos: se haría necesario crear un nuevo nodo *Transform*, para situar el texto correctamente en el espacio (traslación y rotación), y crear un *Shape* anidado dentro de *Transform,* donde editar el texto a través del campo *geometry Text*. El código quedaría así:

```
Transform {
```

```
translation     -0.0715 -.1335 .0092
rotation 1 0 0 1.57
children [
        Shape {
                appearance Appearance {
                material Material {
                        ambientIntensity 0
                        shininess 0
                        diffuseColor 0 0 0
                        }
                }
                geometry Text{
                        string "Mod"
                        fontStyle FontStyle   {
                                justify "MIDDLE"
                                size .0035
                                style "BOLD"
                        }
                }
                }
        ]
}
```

La desventaja clara de este método radica en tener que situar los textos correctamente sobre la superficie del botón en cuestión (un poco separado para evitar un efecto de *"flickering"* entre el texto y la superficie del botón). El hecho de que el equipo haya sido exportado desde Solid Edge provoca que las coordenadas del escenario VRML

sean desconocidas en principio, implicando una colocación de cada texto por el método "prueba y error" hasta conseguir ajustarlo correctamente.

La ventaja, aparte de que toda la información del equipo está contenida en el código (sin necesitar archivos externos como en el caso anterior), está en la carga simultánea de textos y equipo, además de que ambos reciben el mismo trato a efectos de iluminación. Ésta es la solución escogida para nuestros equipos.

4.3.2. Linkado de botones con páginas web externas

Uno de los objetivos principales de toda esta línea de actuación radica en dotar al panel frontal del equipo de funcionalidades interactivas.

En este caso hemos experimentado con el *linkado* de los botones a una web diseñada a tal propósito, a la que se accederá cada vez que se haga clic sobre cualquiera de los botones del panel frontal durante la navegación por el entorno 3D.

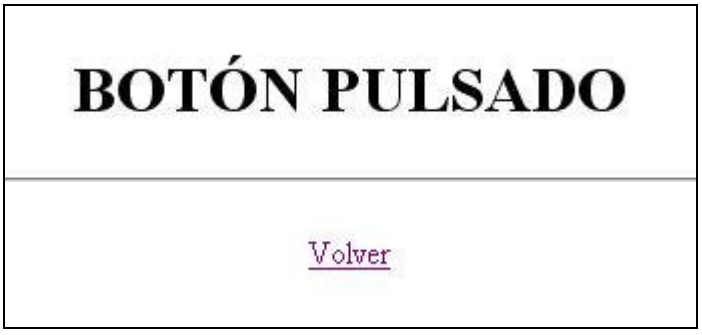


Ilustración 77: ***Web linkada a los botones de los equipos, con información de "Botón pulsado" y posibilidad de volver a la vista principal del equipo***

Para ello, hemos escogido una de las alternativas disponibles para tal fin: el nodo *Anchor*. Como su propio nombre indica, es un nodo que "ancla" a todos aquellos nodos que tenga anidados en su interior.

A continuación, un trozo de código relativo a un botón *linkado* a la web.

```
Anchor {
 children [
  Shape {
   appearance USE S0x80000023
   geometry IndexedFaceSet {
    solid FALSE
    normal Normal { vector [ 0 -1 0,0 -1 0,0 -1 0,0 -1 0,0 -1 0,0 -1 0,0 -1 0,0 -1 0,0 -1 0,0 -1 0,0 -1 0,0 -1 0,0 -1 0,0 -1 0,0 -1 0,0 -1 0,0 -1 0 ] }
    coord Coordinate { point [ .0956 -.133 .0419,.0956 -.133 .042,.0956 -.133 .0417,.0955 -.133 .0422,.0954 -.133 .0415,.0953 -.133 .0424,.0948 -.133 .0409,.0947 -.133 .0429,.0941 -.133 .0432,.0932 -.133 .0401,.0932 -.133 .0436,.0924 -.133 .0399,.0918 -.133 .0441,.0913 -.133 .0442,.0892 -.133 .0392,.089 -.133 .0392,.089 -.133 .0446 ] }
    coordIndex [ 0,1,2,-1,3,2,1,-1,2,3,4,-1,5,4,3,-1,4,5,6,-1,7,6,5,-1,6,7,9,-1,8,9,7,-1,9,8,11,-1,10,11,8,-1,11,10,14,-1,16,15,13,-1,14,13,15,-1,13,14,12,-1,10,12,14,-1 ] }
   }
 ]
 description "CURSOR DERECHA"
 url "boton_pulsado.html"
 }
```

Y el código HTML de la web a la que se accede (programada con Dreamweaver 7), que muestra un mensaje de "Botón pulsado" al usuario, es el siguiente:

```
<html>
<head>
<meta http-equiv="Content-Language" content="es">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>BOTÓN PULSADO</title>
</head>
<body>
<p align="center"> </p>
<p align="center"><b><font size="6">BOTÓN PULSADO</font></b></p>
<hr>
<p align="center"><a href="Agilent 33220A.wrl">Volver</a></p>
</body>
</html>
```

4.3.3. Otras modificaciones al código fuente original

? Ajuste de la vista principal, centrándola en la pantalla.

```
DEF Main Viewpoint {
  position 18.83 -21.9 9.86
  orientation   0.848 0.373 0.377 1.36
  fieldOfView   0.00793
  description "Main View"
 }
```

? Edición del color de fondo del entorno a un tono azulado (blanco por defecto), para evitar confusiones con algunas superficies del equipo.

```
Background { skyColor 0.8 1 1 }
```

? Configuración de la iluminación incidente en el equipo 3D.

```
Group {
  children [
   DEF RightLight DirectionalLight {
        direction -0.632455 0.547723 -0.547723
        intensity  0.2
   }
   DEF LeftLight DirectionalLight {
        direction  0.632455 0.547723 -0.547723
        intensity  0.2
   }
...
```

? Edición de las texturas de los materiales empleados, para una visualización más clara del panel frontal.

```
Shape {
   appearance DEF S0x8000001F Appearance {
    material Material {
     ambientIntensity 0.933
     diffuseColor 0.4 0.698 1
     emissiveColor 0 0 0
```

```
        specularColor 0 0 0
        shininess 1
        transparency 0
       }
      }
...
```

4.4. Resultados finales y conclusiones

Observando el tamaño en memoria del archivo final, 818KB, comprobamos lo conveniente de emplear textos editados en VRML en lugar de Solid Edge, ya que unido a todos los demás añadidos que hemos incluido en el archivo exportado original, apenas lo hemos agrandado 15 KB (en comparación con los 3,5MB que ocupaba el archivo con textos en Solid Edge).

Podemos concluir que la edición directa de equipos en VRML nos daría unos resultados óptimos de tamaño en memoria, aunque también podemos afirmar que conseguir un alto detalle en el modelado de equipos con VRML puede resultar un tarea bastante engorrosa, a causa de las desventajas que en este sentido tiene trabajar en línea de comandos.

5. Optimización de los modelos 3D en VRML

Aunque hemos conseguido unos buenos resultados de los modelos de equipos tras exportarlos a VRML y editarlos posteriormente, hemos comprobado que la codificación VRML del archivo obtenido no es en absoluto óptima, por lo que los 818KB finales que obtenemos se nos antoja excesivo. Trataremos de buscar alguna alternativa para tratar de "aligerar el peso" del archivo, haciéndolo más manejable.

5.1. Optimizador de modelos: Chisel 2.1.2

Entre los optimizadotes de código VRML disponibles por Internet (Rational Reducer, VizUp, Chisel…), se ha seleccionado éste último para comprobar su utilidad.

Atendiendo a sus características, Chisel es un programa que aumenta la calidad, fiabilidad y rendimiento de los mundos VRML, ofreciendo al programador un amplio rango de herramientas para trabajar con archivos VRML: herramientas de diagnóstico para depurar errores, depuradores para eliminar código redundante, inútil o innecesario, reorganizadores de código, compresores…

Las pruebas realizadas han sido las siguientes:

- Tras la instalación, ejecutamos el programa Chisel 2.1.2
- Seleccionando File > Open, cargamos el archivo VRML sobre el que queremos trabajar: Agilent 33220A_V2.wrl (818KB).
- Se inicia un proceso de validación (campo *Validate*) en el que se comprueba la sintaxis, los posibles errores, el código redundante, etc. Al finalizar, se ilumina el campo

correspondiente al depurador (*Clean*), debido a que se han detectados elementos del código depurables.

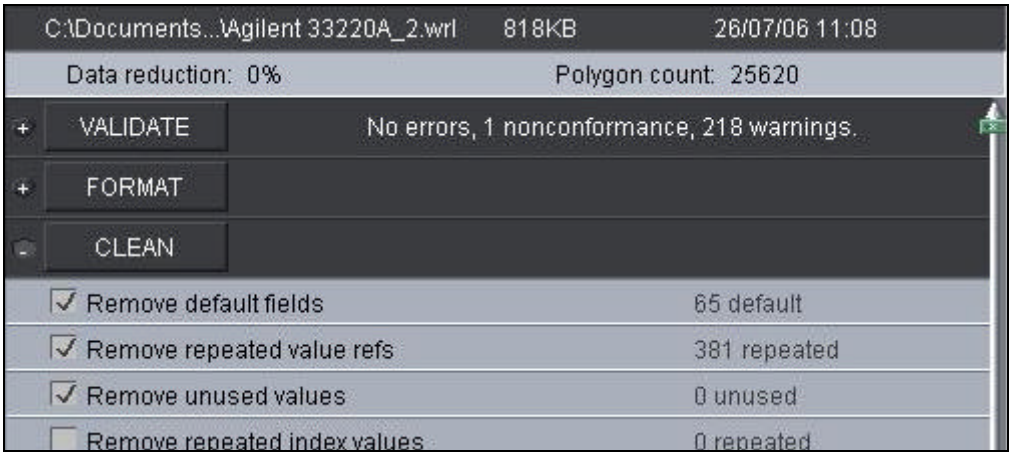


Ilustración 78: ***Detalle del campo Clean en la ventana principal del programa Chisel***

En la ventana de la izquierda se muestra el código completo, y las anotaciones de *"warnings"* y demás defectos encontrados en el código.

? Abrimos el submenú del campo *Clean*, y observamos la información obtenida del proceso de validación. Marcamos las tres primeras casillas (eliminar valores por defecto, valores de referencia repetidos y valores no usados) y pulsamos sobre *Clean*.

? Como podemos comprobar, hemos conseguido una reducción de un 3% del tamaño total del archivo tras el proceso de la depuración.

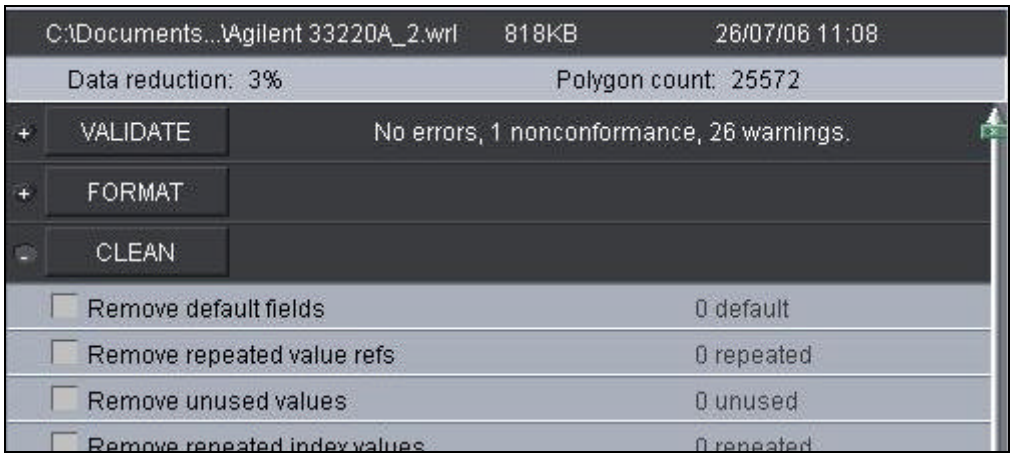


Ilustración 79: ***Reducción del 3% del peso del archivo tras la depuración (Clean)***

? El siguiente paso que seguiremos será el de condensado (campo Condense). Para este proceso se han probado 3 configuraciones distintas, verificando posteriormente los resultados obtenidos tanto en tamaño de memoria total como en los resultados visuales obtenidos del modelo optimizado.

Las opciones que configuraremos serán las siguientes:

- *Adjust numeric resolution*: Ajuste del valor numérico en múltiples aspectos de resolución: coordenadas, color, texturas y nodos normales.
- *Create DEF / USE*: crea definiciones para reutilizar nodos repetidos a lo largo del código.
- *Create index files*: crea listas de valores índice para evitar redundancias en determinadas listas de valores.
- *Remove normals*: elimina las normales creadas en el código, provocando que sea el navegador el que las

tenga que calcular. En ocasiones esta medida acelera el tiempo de carga del modelo.

- ✍ *Remove unused DEFs*: elimina definiciones no reutilizadas (aunque las creadas para *Viewpoints* son respetadas).
- ✍ *Shorten DEFs names*: acorta los nombres de las definiciones.

? A continuación se muestran las configuraciones adoptadas y sus respectivos resultados, con las reducciones aplicadas al archivo recién depurado:

- **Configuración 1**: reducción del 15% (archivo: 699KB)

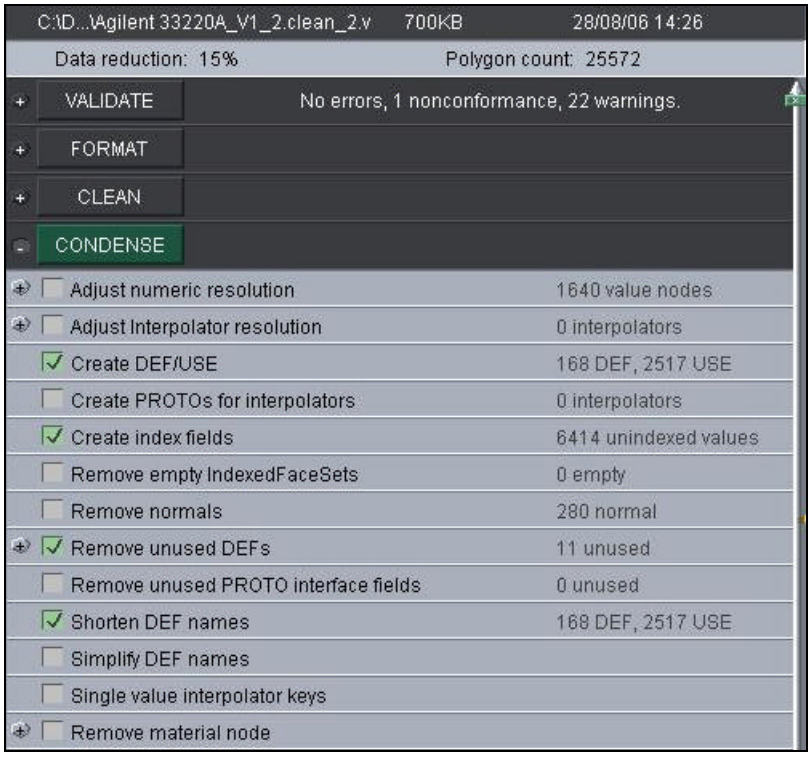


Ilustración 80: ***Configuración 1 de condensado de código***

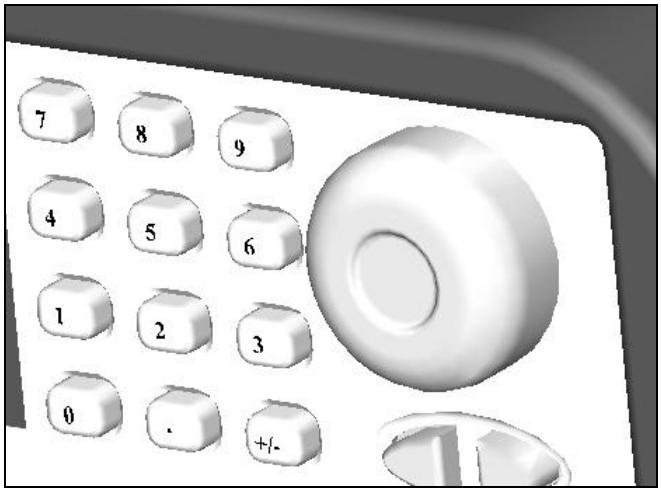


Ilustración 81: ***Detalle del equipo usando la configuración 1***

- **Configuración 2:** reducción del 27% (archivo: 596KB)

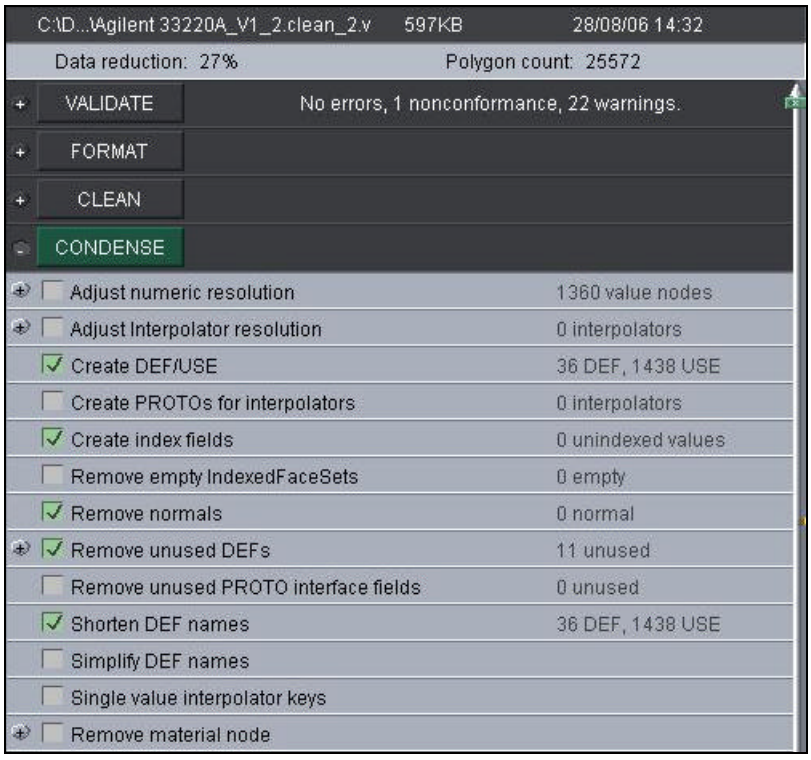


Ilustración 82: ***Configuración 2 de condensado de código***

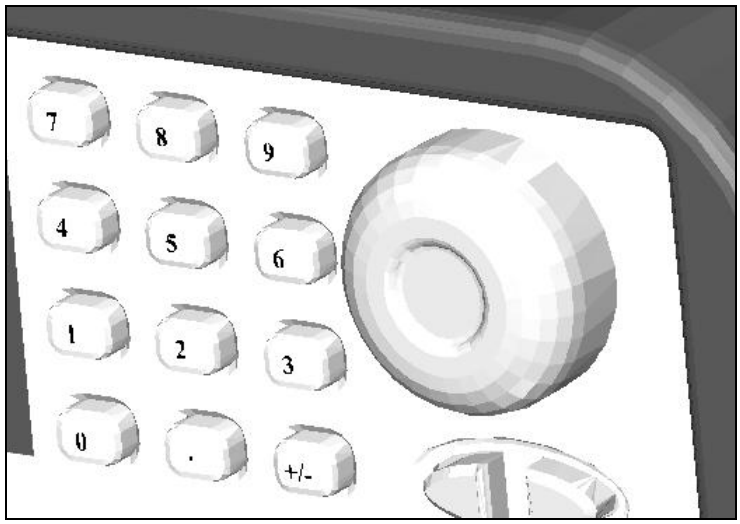


Ilustración 83: ***Detalle del equipo usando la configuración 2***

- **Configuración 3:** reducción del 36% (archivo: 526KB)

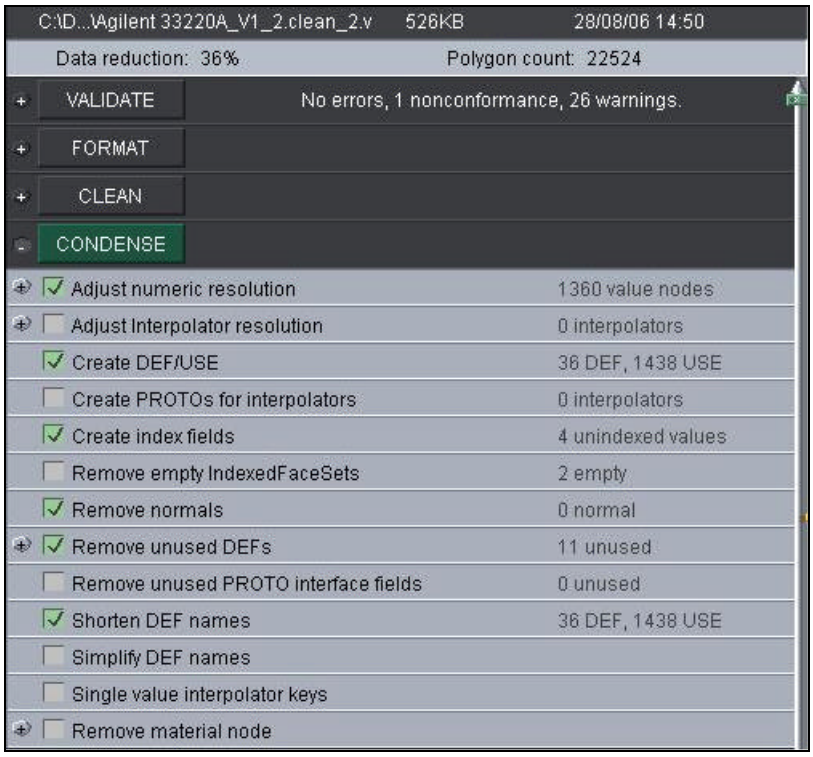


Ilustración 84: ***Configuración 3 de condensado de código***

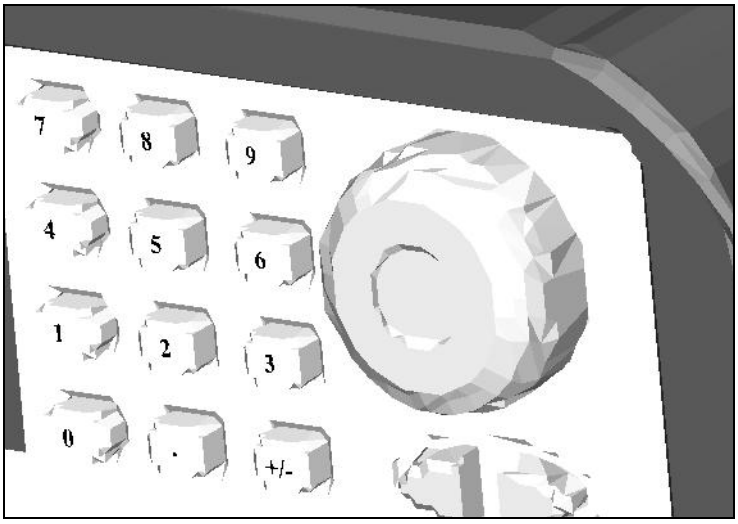


Ilustración 85: ***Detalle del equipo usando la configuración 3***

- Por último, se almacena el nuevo archivo optimizado seleccionando en la barra superior de herramientas en *Save current file* o *Save under new name*.

5.2. Compresor de archivos VRML: Gzip

La extensión estándar de los archivos VRML es .wrl, aunque es posible emplear una nueva extensión .wrz para VRML comprimido. La ventaja de la compresión de los mundos VRML de esta manera se encuentra en que los navegadores web pueden leer archivos VRML al igual que los archivos normales, suponiendo una ventaja clara para el desarrollador de códigos.

Esto lo podemos comprobar al intentar comprimir el archivo recién editado en VRML (previamente a ser tratado con Chisel), con extensión en memoria de 818KB. Para ello utilizaremos también el programa Chisel, que entre sus herramientas cuenta con un compresor Gzip.

- Simplemente tendremos que abrir el programa Chisel y cargar el archivo deseado: Agilent 33220A.wrl
- Seleccionamos el botón *Save gzipped under new name* y especificamos un nuevo nombre para nuestro archivo, que tendrá una nueva extensión: .wrz.

Los resultados son satisfactorios a todas luces: el nuevo archivo ocupa una extensión de 101KB, más de 8 veces menor en extensión que el original, con una reducción del 88%. Y todo ello sin afectar en absoluto a la resolución del equipo modelado al ser visualizado en un navegador web, y sin aumentar el tiempo de carga del archivo.

Ilustración 86: ***Equipo de instrumentación electrónica Agilent 33220A modelado en VRML y visualizado en un navegador web***

Una última opción a considerar sería optimizar en primer lugar el archivo VRML con Chisel, para posteriormente comprimirlo con Gzip, bajando de los 100KB el nuevo archivo obtenido. Estas nuevas cotas de tamaño en memoria son perfectamente tratables en un sistema cliente-servidor, que es lo que se pretende.