Departamento de Ingeniería de Sistemas y Automática

Universitat Politècnica de València

Simulación en robótica con *VirtualRobot*. Introducción a *VRM*

Martin Mellado Arteche

Escuela Técnica Superior de Ingeniería Informática

Septiembre, 2012

Contenido

Introducción al sistema de modelado gráfico para robótica *VRM*

1. Objetivo

El objetivo de este documento es presentar un tutorial para introducir al lector en el manejo del sistema de modelado de objetos, piezas y entornos de robots denominado *VirtualRobot Modeller* (*VRM*) y su uso dentro de *VirtualRobot Simulator* (*VRS*). *VRM* es un conjunto de programas o aplicaciones externas de modelado gráfico orientado a aplicaciones de robótica para su uso en *VRS*. *VRM* se instala al instalar *VRS*. En la parte inicial del tutorial se muestra el aprendizaje de dichos programas con una serie de tareas a realizar. Posteriormente se muestra como se pueden utilizar los resultados en *VRS*. Se supone que el usuario está familiarizado con el programa *VRS* y es capaz de ejecutar aplicaciones en dicho entorno.

2. Introducción. Aplicaciones de *VRM*

En *VRS* se puede cargar un entorno para hacer más realista la simulación. Los entornos están compuestos de objetos y piezas. La diferencia entre objetos y piezas es que mientras que los objetos son elementos meramente "decorativos", es decir, con información sólo gráfica para mejorar el aspecto visual de una simulación, las piezas contienen más información. En concreto, tienen asociados sistemas de coordenadas que se utilizan para que los robots puedan actuar sobre ellos, tanto mediante movimientos relativos como para realizar agarre de las piezas.

Las aplicaciones *VRM* sirven para crear y editar objetos y piezas y poder construir entornos con ellos. *VRM* está compuesto, principalmente, de las siguientes tres aplicaciones ***VRM Tools*** (ejecutables desde la opción **File>>VRM Tools**):

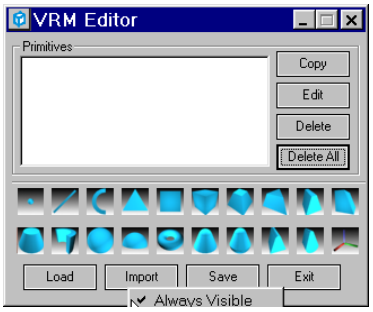
- ***VRM Editor*** para crear y editar tanto objetos como piezas.
- ***VRM Object Mover*** para localizar, borrar y salvar objetos de un entorno cargado en *VRS*.
- ***VRM Part Mover*** para localizar, borrar y salvar piezas de un entorno cargado en *VRS*.

Con estas aplicaciones se pueden generar entornos que posteriormente se salven con la opción correspondiente de ***VRS Loader***.

3. Modelado de objetos con *VRM Editor*.

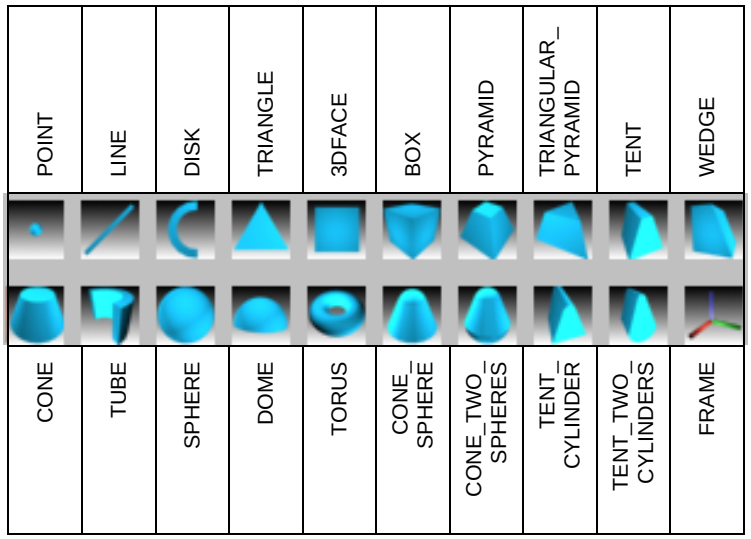
Para el modelado de objetos en *VRS* se dispone de la *VRM Tool* llamada ***VRM Editor***. ***VRM Editor*** es una aplicación externa que sirve modelar objetos para crear entornos. Como programa externo, su ejecutable **VRMEditor.exe** se puede ejecutar directamente desde el operativo, estando localizado en la carpeta **Applications\VRMTools** a partir del directorio de *VRS*. En la configuración de instalación por defecto de *VRS* se encuentra accesible con la opción del menú **File>>VRM Tools>>VRM Editor**. Una vez ejecutado, es un programa como otro cualquiera, con acceso desde la barra de tareas. Si ***VRM Editor*** está en ejecución se volverá a abrir en caso de lanzarse a ejecución de nuevo.

Su interfaz es el que se muestra a la derecha, disponiendo, como todas las aplicaciones de tipo *VRM Tool*, de un menú propio de tipo *pop-up*, accesible al pulsar sobre la ventana (fuera de los botones) con el botón derecho del ratón, que permite, si se activa la opción **Always Visible** (como aparece por defecto), tener esta aplicación siempre visible por encima de cualquier otra ventana, incluida la de *VRS*. El interfaz se compone de cuatro partes: listado de primitivas, botones de edición, botones de creación de primitivas y botones de ficheros.



Creación de primitivas

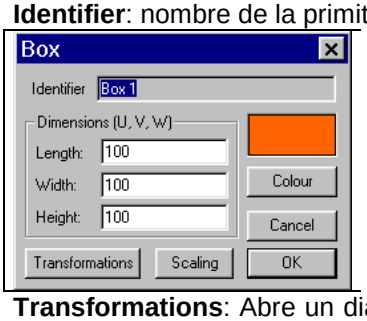
El modelador ***VRM Editor*** permite crear objetos como unión de primitivas gráficas. Estas primitivas se crean con los botones gráficos correspondientes, uno para cada tipo de primitiva dando un total de 20 botones. Las primitivas gráficas disponibles y sus iconos son:



Cada primitiva geométrica se define mediante una serie de parámetros geométricos (dimensiones). La definición de los parámetros geométricos de cada primitiva viene explicada en el **apéndice A**.

Para crear una primitiva se selecciona el botón correspondiente. Por ejemplo, al seleccionar el icono de la primitiva **Box** aparece el siguiente diálogo para introducir datos:

Identifier: nombre de la primitiva (este campo no es modificable)



Dimensions (U, V, W): dimensiones o parámetros geométricos. En el caso del prisma son: **Length** (longitud), **Width** (anchura), **Height** (altura)
Colour: Abre un diálogo para seleccionar el color de la primitiva. El color seleccionado está representado en el rectángulo superior
Cancel: cancela la definición de la primitiva
OK: Guarda y termina la definición de la primitiva
Transformations: Abre un diálogo para permitir localizar la primitiva. Este diálogo se comenta a continuación
Scaling: Abre un diálogo que permite escalar la primitiva (multiplica todos sus parámetros geométricos por un factor de escala mayor que cero)

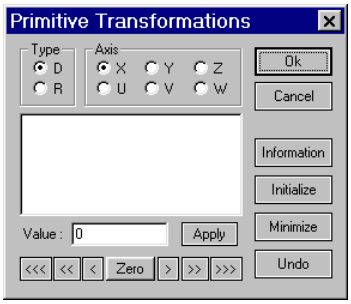
En este diálogo se deben introducir las dimensiones (opcionalmente, escalarlas) y seleccionar el color adecuado. Para localizar las primitivas se dispone de las transformaciones tal como se explica en la siguiente sección.

Tarea 3.1:
- Pulsa el icono Box para definir un prisma (**NO pulses el botón OK** en esta tarea, ya que continúa en la Tarea 3.2).
- Prueba a cambiar sus dimensiones para ver cómo se ajusta dinámicamente.
- Cambia el color de la primitiva.
- Escala la primitiva a tamaño doble (**Scale Factor**=2) y mitad (**Scale Factor**=0.5).

Localización de primitivas

El botón **Transformations** abre el siguiente diálogo:

Type: Define el tipo de transformación: Desplazamiento (D) o Rotación (R)
Axis: Selecciona respecto a qué eje se aplicará la transformación pudiendo ser un eje global (X,Y,Z) o local (U,V,W)
Ok: termina la definición de transformaciones y vuelve al diálogo anterior (creación de primitiva) salvando las transformaciones realizadas



Cancel: cancela la definición de las transformaciones y vuelve al diálogo anterior (creación de primitiva) recuperando el estado en que se encontraban las transformaciones al iniciar este diálogo
Information: Muestra la matriz de transformación homogénea
Initialize: Elimina todas las transformaciones de la lista
Minimize: Minimiza la transformación a seis o menos transformaciones básicas (RX,RY,RZ,DX,DY,DZ)
Undo: Deshace la última transformación básica de la lista eliminándola
Value: Introduce un valor de desplazamiento o ángulo (en grados)
Apply: Aplica la transformación básica definida con la terna (**Type**, **Axis**, **Value**) y la añade al final de la lista
Zero: Pone a cero el valor del campo **Value**
<<<, **<<**, **<**: restan 100,10,1 respectivamente al valor del campo **Value**
>, **>>**, **>>>**: suman 1,10,100 respectivamente al valor del campo **Value**

El uso de este diálogo, que aparenta cierta dificultad, es más sencillo de lo que parece. Permite aplicar desplazamientos y rotaciones del objeto respecto a los ejes principales del mismo U,V,W (que se mueven solidariamente con el objeto) o los ejes globales del "mundo" X,Y,Z (que permanecen estáticos o fijos). Como ayuda al uso del diálogo, aparecen dibujados (inicialmente solapados) los ejes de ambos sistemas de coordenadas, usando la nomenclatura de tres ejes con los colores rojo, verde y azul (RGB), que representan los ejes X,Y,Z o U,V,W respectivamente de dichos sistemas.

Sobre cada uno de estos ejes se pueden realizar operaciones de traslación o de rotación. A no ser que dos ejes sean paralelos, el efecto de las traslaciones respecto a uno y otro será diferente. Igualmente, a no ser que dos ejes sean coincidentes, el efecto de las rotaciones será distinto.

Para comprender su funcionamiento, realizar las siguientes tareas.

Tarea 3.2:
- Realiza desplazamientos (Type=D) en los ejes X,Y,Z (ejes fijos) de valor 100 (o cualquier otro) hasta familiarizarse con su uso. Fijaros que una vez se define el tipo, el eje y el valor de la transformación básica, se debe aplicar con el botón **Apply**.
- Realiza desplazamientos en los ejes U,V,W. Nótese que el efecto es el mismo (desplazar en U hace lo mismo que desplazar en X) ya que los ejes se mantienen paralelos.
- Prueba las opciones **Information**, **Undo**, **Minimize** e **Initialize**.

Recuerda que estás usando VRS, con lo que tienes todas sus facilidades gráficas de visualización a tu disposición (zoom, scroll, point of view, etc).

Tarea 3.3:
- Inicializa la transformación (botón **Initialize**).
- Realiza una rotación (Type=R) en el eje Z de ángulo 60.
- Realiza desplazamientos en los ejes X y U. Deduce la diferencia entre los desplazamientos respecto a ejes fijos y locales.
- Realiza desplazamientos en los ejes Y y V. Deduce la diferencia entre los desplazamientos respecto a ejes fijos y locales.

Tarea 3.4:
- Inicializa la transformación (botón **Initialize**).
- Realiza desplazamientos en X, Y, Z.
- Realiza rotaciones en los ejes U,V,W (ejes principales del objeto) eligiendo un ángulo deseado.
- Realiza rotaciones en los ejes X,Y,Z (ejes fijos del mundo). Deduce la diferencia entre las rotaciones respecto a ejes fijos y locales.

Dado que este diálogo es básico para localizar las primitivas que componen los objetos, resulta necesario tener claro su manejo antes de proseguir con la siguiente sección.

Tarea 3.5:
- Determina la localización de la primitiva como desees.
- Cierra el diálogo de transformaciones (botón **Ok**).
- Cierra el diálogo de definición de la primitiva **Box** (botón **OK**). Desde este momento se dispone de una primitiva en el modelador.

Copia, edición y borrado de primitivas

Las primitivas ya creadas pueden duplicarse con el botón **Copy**. Basta con seleccionar la primitiva a duplicar de la lista de primitivas existentes y pulsar el botón **Copy**. Se abrirá el diálogo de creación para poder cambiar las dimensiones, el color o la localización de la replica (en otro caso ambas primitivas quedarán solapadas). En caso de cancelar se pierde la copia.

Cualquier primitiva introducida se puede editar, esto es, modificar cualquiera de sus dimensiones o parámetros geométricos, cambiarle el color, escalarla o modificar su localización. Para ello se debe seleccionar su nombre en la lista de primitivas y pulsar el botón **Edit** y se abre el diálogo de creación de la primitiva para poder cambiar las dimensiones, el color o la localización de la primitiva.

Por otra parte, cualquier primitiva introducida se puede eliminar. Para ello se debe seleccionar su nombre en la lista de primitivas y pulsar el botón **Delete**. Nota que NO pide confirmación y que no se puede recuperar una primitiva una vez eliminada.

Finalmente, también se pueden eliminar todas las primitivas creadas en *VRS* mediante el botón **Delete All**, que pedirá confirmación antes de cumplir la acción.

Tarea 3.6:
- Introduce varias primitivas de tipos diferentes (excepto **Frame**) y localízalas en el espacio haciendo uso de diferentes opciones.
- Prueba las opciones de edición, copia y borrado de primitivas para familiarizarte con su uso.

Almacenamiento de objetos

Una vez se han introducido todas las primitivas que forman un objeto, éste se puede salvar en un fichero con el botón **Save**. Tras un diálogo para seleccionar el directorio e introducir el nombre del fichero, las primitivas existentes se guardan en un fichero con extensión **OBF**. Para mantener una estructura de ficheros coherente, se recomienda guardar los ficheros en la carpeta **Models\User** (a partir de la cual se pueden organizar los ficheros por carpetas). El

objeto se salva en un fichero con formato ASCII (modificable por tanto con cualquier editor de texto). El formato **OBF** está comentado en el apéndice B.

Tarea 3.7:
- Modela libremente una mesa con dimensiones 120cm x 60cm y 75cm de alto (recuerda usar milímetros al modelar el objeto).
- Guarda la mesa en un fichero (en la carpeta **Models\User**).

Carga e importación de las primitivas de un objeto

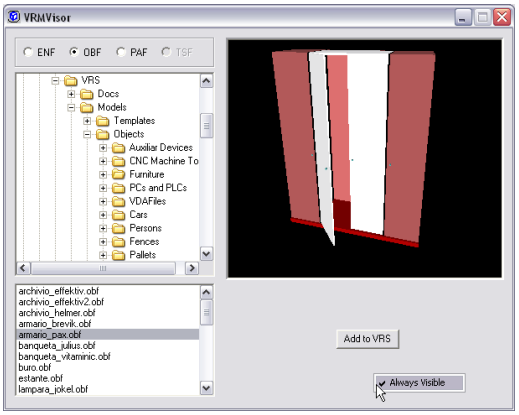
Las primitivas salvadas en ficheros de objetos se pueden cargar con el botón **Load**. Si las primitivas creadas no se han salvado pedirá confirmación, ya que en otro caso se perderán. Por otra parte, durante la creación de un objeto se pueden importar todas las primitivas de un objeto ya creado y guardado en fichero mediante el botón **Import** que permitirá seleccionar el fichero del objeto. Todas las primitivas que compongan el objeto almacenado en el fichero se añaden al objeto en creación actual.

Tarea 3.8:
- Modela libremente un par de objetos sencillos (máximo 4 primitivas) y guárdalos en dos ficheros.
- Crea un nuevo objeto importando la mesa y los objetos anteriores. Observa que una vez se importan los objetos sólo se puede acceder a sus primitivas y pierden su condición de agrupación, con lo que no se puede re-localizar todo el objeto como un conjunto (esto se resuelve con la definición de entornos comentada en la siguiente sección).
- Elimina este nuevo objeto.

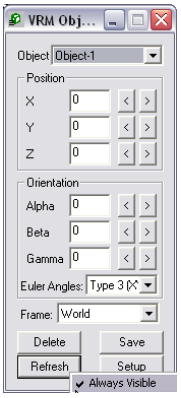
4. Modelado de entornos con objetos

Tal como se ha visto en la sección anterior, con la ayuda de *VRM Editor* se pueden representar en *VRS* primitivas para modelar objetos y almacenarlos en ficheros. Si se desea que *VRS* pueda representar estos objetos como entidades agrupadas, hay que formar ficheros de entorno. Un entorno es una unión de objetos (posteriormente se verá que también se pueden incorporar piezas).

Para cargar objetos que definan entornos se recomienda usar la opción de buscar entornos disponible en *VRS Loader* (botón ) que ejecuta el programa ***VRM Visor***. Su interfaz es el que se muestra a la derecha, disponiendo, como todas las aplicaciones de tipo *VRM Tool*, de un menú propio de tipo *pop-up*, accesible al pulsar sobre la ventana (fuera de los botones) con el botón derecho del ratón, que permite, si se activa la opción **Always Visible** (como aparece por defecto), tener esta aplicación siempre visible por encima de cualquier otra ventana, incluida la de *VRS*.



Con esta aplicación se pueden visualizar y cargar ficheros de entorno en *VRS* activando la opción **ENF** y ficheros de objetos activando la opción **OBF**. También sirve para ficheros de piezas activando la opción **PAF**. ***VRM Visor*** muestra a la izquierda el árbol de directorios (parte superior) y el contenido del directorio (parte inferior) y a la derecha una visualización del contenido del fichero en una ventana gráfica. Pulsando dos veces con el ratón sobre la ventana gráfica se puede configurar el uso del ratón sobre la ventana gráfica. Con el botón ***Add to VRS*** se pueden añadir objetos al entorno actual disponible en *VRS*.



Una vez se tiene al menos un objeto cargado en *VRS* (en otro caso no se arranca), se puede ejecutar la aplicación ***VRM Object Mover***, accesible con la opción del menú **File>>VRM Tools>>VRM Object Mover**. Esta aplicación permite localizar, borrar y salvar objetos de un entorno, si bien su función principal es para cambiar la posición y orientación de un objeto ya existente en el entorno y permitir el modelado de entornos actuando de esta forma. Su interfaz es el que se muestra a la derecha, disponiendo, como todas las aplicaciones de tipo *VRM Tool*, de un menú propio de tipo *pop-up*, accesible al pulsar sobre la ventana (fuera de los botones) con el botón derecho del ratón, que permite, si se activa la opción **Always Visible** (como aparece por defecto), tener esta aplicación siempre visible por encima de cualquier otra ventana, incluida la de *VRS*.

En el campo **Object** se puede seleccionar el objeto a re-localizar y con los botones "**<**" y "**>**" o tecleando directamente se puede cambiar el valor de las componentes x,y,z de posición o α,β,γ de orientación. La orientación se puede especificar en cualquiera de los tres tipos de ángulos de Euler y la localización se puede representar respecto al sistema del mundo o el del entorno (inicialmente coincidentes, sólo varían si se ha modificado esta relación con **Place Environment**). Se dispone de los botones **Delete** para eliminar un objeto del entorno, **Save** para salvar un objeto en su localización actual, **Refresh** para actualizar la lista de objetos (necesario cuando otra aplicación ha cargado nuevos objetos) y **Setup** para configurar los incrementos de los botones "**<**" y "**>**".

Con estas dos aplicaciones se pueden cargar y localizar objetos en VRS para formar entornos que se pueden salvar con el botón **Save Environment** de ***VRS Loader***. Se recomienda guardar los ficheros en la carpeta **Models\User**. Los entornos se salvan en ficheros con extensión **ENF** cuyo formato está descrito en el apéndice B. Como ya se ha comentado, estos ficheros se pueden cargar en *VRS* con el botón **Load Environment** de ***VRS Loader***.

Tarea 4.1:
- Asegúrate de tener un entorno vacío en *VRS*.
- Añade en *VRS* la mesa definida en la sección anterior.
- Añade en *VRS* uno de los objetos definidos en la sección anterior.
- Localiza este objeto para que se sitúe encima de la mesa.
- Añade en *VRS* el otro objeto y localízalo también encima de la mesa.
- Salva el entorno en la carpeta **Models\User** y elimina el entorno.
- Carga el entorno creado.

5. Modelado de piezas y de entornos con piezas

Los objetos modelados con *VRM* podrán visualizarse en *VRS* para dar la sensación de que el robot se integra en un entorno con el que trabaja. Sin embargo, estos objetos son objetos inertes, sobre los que un robot no podrá operar. Por el contrario, con *VRM* se pueden definir piezas, que pueden ser operadas (por ejemplo manipuladas) por robots en *VRS*. La diferencia entre objetos y piezas es que estos disponen, además de un conjunto de primitivas para definir su forma geométrica, de al menos un sistema de coordenadas que servirá para que el robot realice movimientos relativos a este sistema de coordenadas o agarre la pieza por este sistema de coordenadas.

Para definir piezas, se trabaja con VRM Editor igual que para definir objetos, con la única salvedad de que debe crearse al menos un **Frame** con el botón correspondiente. Una vez se active este tipo, *VRM* añade un sistema de coordenadas de operación de la pieza (*operation frame*) inicialmente con transformación identidad. El campo **Display Size** así como la opción de escalado sólo sirve para la visualización del **Frame**, sin ser un campo de definición de un parámetro geométrico. Para localizar el **Frame** se debe hacer uso del diálogo de **Primitive Transformations** accesible con el botón **Transformations**.

La creación de piezas se realiza en su parte geométrica igual que la creación de objetos, pudiendo introducir, editar, copiar o borrar primitivas de la misma forma. La única exigencia es que se haya creado una primitiva **Frame**. Las piezas se salvan en ficheros con extensión **PAF** con el formato descrito en el apéndice B. Para la creación de piezas se pueden importar ficheros de objetos **OBF** y/o ficheros de piezas **PAF**. Se recomienda guardar los ficheros de piezas en la carpeta **Models\User**.

Las piezas se pueden incluir en los ficheros de entornos que se visualizan en *VRS* haciendo uso de la aplicación ***VRM Visor*** que se ha comentado anteriormente. Hay que activar la opción PAF para poder visualizar ficheros de piezas existentes. Las piezas cargadas en el modelador se pueden localizar y borrar (incluso salvar en su nueva localización) mediante la aplicación ***VRM Part Mover***, accesible con la opción del menú **File>>VRM Tools>>VRM Part Mover** y de manejo similar a la aplicación ***VRM Object Mover*** comentada anteriormente. Ahora se pueden salvar entornos con objetos y piezas usando el botón **Save Environment** de ***VRS Loader***.

Tarea 5.1:
- Asegúrate de tener un entorno vacío en *VRS*.
- Modela una pieza sencilla (un par de primitivas y un sistema de operación).
- Salva la pieza a fichero en la carpeta **Models\User**.

Tarea 5.2:
- Asegúrate de tener un entorno vacío en *VRS*.
- Carga el entorno creado en la **Tarea 4.1**.
- Añade al entorno la pieza creada en la tarea anterior.
- Carga el robot **CRS A465.rkf** en *VRS* y sitúalo encima de la mesa.
- Activa el modo de visualización de sistemas de operación activando el campo **PARTS** de la opción del menú **Display>>Frames...**.

6. Operación sobre las piezas en *VRS*

La definición de los sistemas de operación es imprescindible para poder programar en *VRS* las operaciones de:
- Mover un robot para que el sistema de coordenadas activo de una de sus herramientas sea coincidente con el sistema de operación de una pieza (*MoveToPart*).
- Mover un robot para que el sistema de coordenadas activo de una de sus herramientas se aproxime al sistema de operación de una pieza (*ApproxToPart*).
- Hacer que una herramienta del robot coja una pieza forzando a que la relación entre el sistema de coordenadas activo de una de sus herramientas y el sistema de operación de una pieza sea constante (*PickPart*).

Para definir dónde se coloca un sistema de operación de una pieza que sea útil hay que saber qué tipo de operación se va a realizar sobre esta pieza y cómo está situado el sistema de la herramienta del robot que vaya a realizar la operación.

Por ejemplo, supóngase que se desea diseñar una pieza que pueda ser manipulada por el robot CRS A465 que dispone de una herramienta tipo ventosa neumática (Figura 2). El sistema de coordenadas de esta herramienta está modelado en *VRS* con su origen situado en el centro de su cara exterior, el eje X apuntando hacia el exterior de la ventosa y el eje Z hacia arriba verticalmente en la posición de reposo del robot.

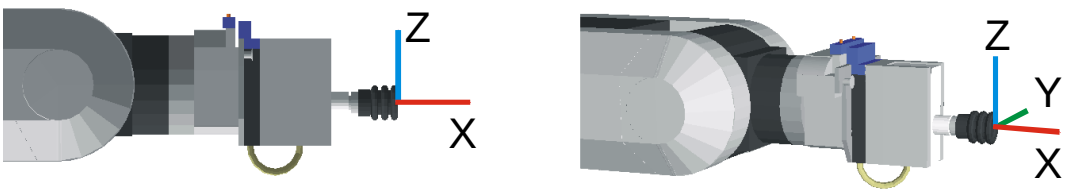


Figura 2. Modelo de la ventosa del CRS A465 y su ToolFrame.

Para simular que el robot con la ventosa coge una pieza habrá que situar un sistema de operación en la pieza como se muestra en la Figura 3a, de forma que se pueda ordenar al robot un movimiento que sitúe el sistema de coordenadas de la ventosa coincidente al sistema de operación tal como se ve en la Figura 3b, para permitir que la ventosa coja la pieza.

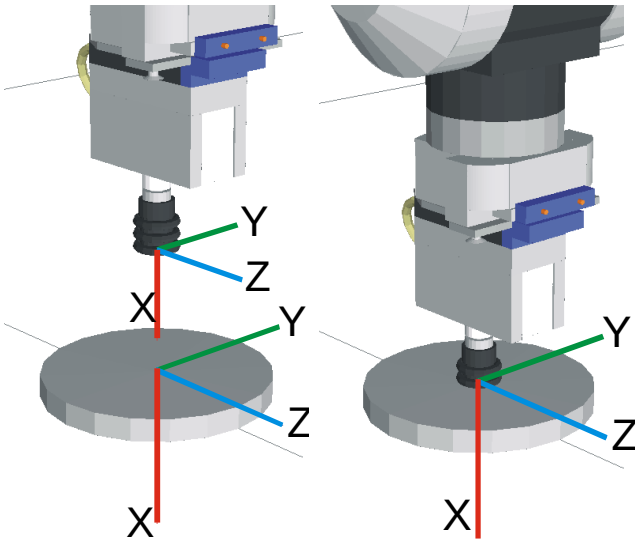


Figura 3. a) Sistema de operación. b) Situación de cogida.

Si la pieza es un cilindro situado en el suelo, bastará con aplicar las dos siguientes transformaciones:

- Rotación en V de 90º para orientar el sistema con el eje X vertical hacia abajo.
- Desplazar en Z la altura del cilindro, para situar el origen del sistema de operación en la cara superior del cilindro.

Tarea 6.1:

- Modela una pieza compuesta de un cilindro que pueda manipular el CRS A465 con la ventosa mediante un sistema de operación.
- Salva la pieza a fichero.
- Crea un entorno que tenga tantos objetos como desees pero como única pieza el cilindro anterior situado con una única transformación que sea un desplazamiento de 500mm en el eje X.
- Salva el entorno en la carpeta **Models\User**.
- Carga el robot **CRS A465 Vacuum.rfk** de la carpeta **Models\Tutorial**.
- Con ***VRS PartHandling***, manipula la pieza.

7. Ejercicios prácticos

El CRS A465 también puede disponer como herramienta de una pinza electromecánica servocontrolada. El sistema de coordenadas de esta herramienta (Figura 4) está modelado en *VRS* con su origen situado en el plano en que se tocan los dedos al cerrarse, sobresaliendo 21mm desde la placa base de los dedos y hundido 4mm desde el extremo de los dedos (de longitud 25mm). La orientación es similar a la del sistema de coordenadas de herramienta de la ventosa.

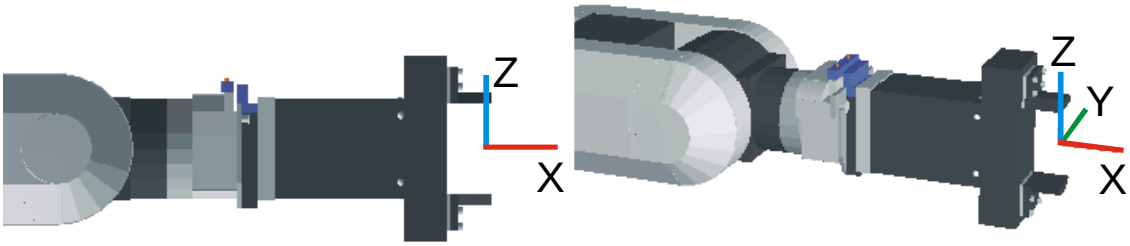


Figura 4. Modelo de la herramienta pinza del CRS A465 y su ToolFrame.

El movimiento de los dedos de la pinza está modelado mediante 4 estadios con aperturas entre los dedos de 51mm (pinza abierta), 33mm (semiabierta), 15mm (semicerrada) y 1mm (pinza cerrada) tal como se muestra en la Figura 5.

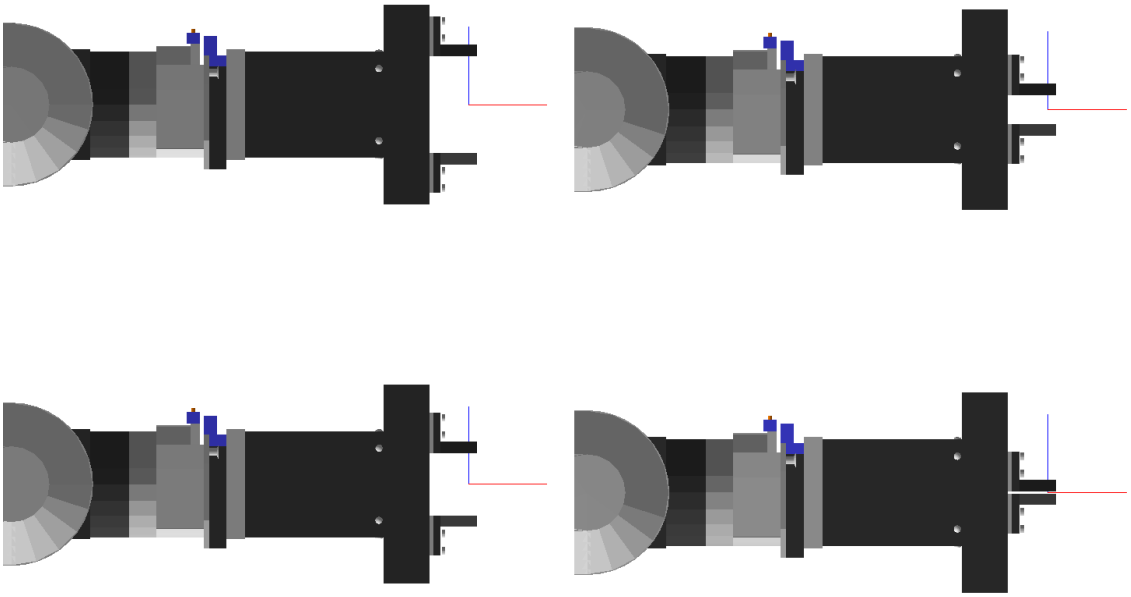


Figura 5. Modelos de los estados de la pinza del CRS A465.

Ejercicio 7.1:
Modela un entorno que tenga **dos piezas** para la manipulación del robot CRS A465 (sitúa la pieza en algún lugar alcanzable por el robot) de la siguiente forma:
- La primera pieza se debe agarrar correctamente con la pinza en estado semiabierto.
- La segunda pieza se debe agarra correctamente con la pinza en estado semicerrado.

Manipula las piezas en *VRS* con ***VRS PartHandling***.

Ejercicio 7.2:
Modela un entorno que tenga **una pieza** para la manipulación del robot CRS A465 de la siguiente forma:
- Con un sistema de operación la pieza se debe agarrar correctamente con la pinza en estado semiabierto.
- Con otro sistema de operación la pieza se debe agarrar correctamente con la pinza en estado semicerrado.

Manipula las piezas en *VRS* con ***VRS PartHandling***.

A. Definición de primitivas

Las primitivas geométricas y sus parámetros de definición están detallados en el documento VRPD (VirtualRobot Primitive Definition) pero a modo de ayuda a continuación se muestran una gráfica resumen de los mismos.

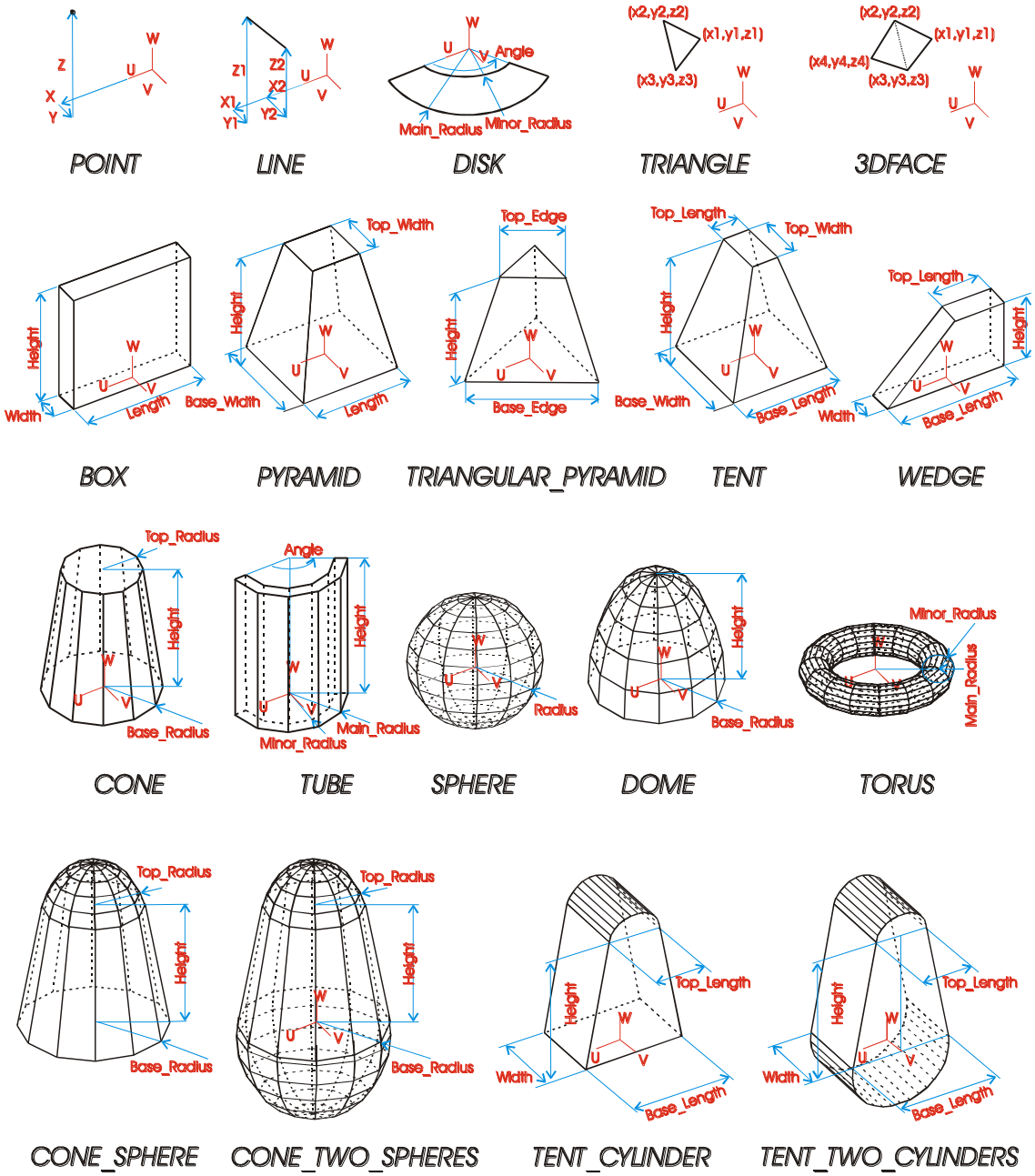


Figura A1. Primitivas Geométricas y Parámetros de Definición.

B. Formato de ficheros

El formato de los ficheros OBF, PAF y ENF están detallados en el documento VRFFD (VirtualRobot File Format Definition) y se resumen a continuación (los ";" indican comentario hasta fin de línea, los tabuladores no son significativos).

```
; OBJECT DESCRIPTION FILE (OBF)
; This file describes an Object as a set of Primitives

[GENERAL]                                   ; General Information
        Name            = Table             ; Object Name
        Manufacturer    =                   ; Robot Manufacturer (for robot components)
        Model           =                   ; Robot Model (for robot components)
        Author          = Juan Vte. Catret  ; Author
        Company         = DISA - UPV        ; Company
        Date            = 9/2/99            ; Date

[PRIMITIVE1]                                ; First Primitive of Object
        Type    = BOX                       ; Type of Primitive:
            ; POINT (x, y, z)
            ; LINE (x1, y1, z1, x2, y2, z2)
            ; DISK (Main_Radius>0, Minor_Radius>=0, 0<Angle<=360 [Main_Radius>=Minor_Radius])
            ; TRIANGLE (x1, y1, z1, x2, y2, z2, x3, y3, z3)
            ; 3DFACE (x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4)
            ; BOX (Length>0, Width>0, Height>0)
            ; PYRAMID (Length>0, Base_Width>0, Top_Width>=0, Height>0)
            ; TRIANGULAR_PYRAMID (Base_Edge>0, Top_Edge>=0, Height>0)
            ; TENT (Base_Length>0, Base_Width>0, Top_Length>0, Top_Width>=0, Height>0)
            ; WEDGE (Base_Length>0, Width>0, Top_Length>0, Height>0)
            ; CONE (Base_Radius>0, Top_ Radius>=0, Height>0)
            ; TUBE (Main_Radius>0, Minor_Radius>=0, 0<Angle<=360 [Main_Radius>Minor_Radius] , Height>0)
            ; SPHERE (Radius>0)
            ; DOME (Base_Radius>0, Height>0)
            ; TORUS (Main_Radius>0, Minor_Radius>0)
            ; CONE_SPHERE (Base_Radius>=0, Top_Radius>0, Height>0)
            ; CONE_TWO_SPHERES (Base_Radius>0, Top_Radius>0, Height>0)
            ; TENT_CYLINDER (Width>0, Base_Length>=0, Top_Length>0, Height>0)
            ; TENT_TWO_CYLINDERS (Width>0, Base_Length>0, Top_Length>0, Height>0)
            ; VDA_CURVE (No_Segments, ...)
            ; VDA_SURF (No_Patches, ...)
            ; VDA_FILE (FileName, Thickness>0, Curve_Sections>0, Surface_Sections>0)
            ; OBF_FILE (FileName)
            ;       For VDA_FILE and OBF_FILE, the file name must include extension and start from VRS Path
        RGB     = 255,0,0           ; Red-Green-Blue (RGB) components for Colour (0..255,0..255,0..255)
                            ; This field can be avoided for OBF_FILE primitive (it has no effect if exists)
        Length  = 30.00             ; Name and value of parameters according to type of Primitive
        Width   = 100.00            ; Dimensions on mm (degrees for angles)
        Height  = 150.00
        Transformations     = DX>5.000, RZ>90.000       ; Location of Primitive1 related to Object Frame
                ; Defined with as many Basic Transformations as required
                ; Transformations can be: Displacements (D) or Rotations (R)
        ; The axis is indicated with (X,Y,Z,U,V,W) before the symbol '>'
        ; Transformations related to Object (X,Y,Z) or Primitive (U,V,W) Frames
        ; The transformations will be applied in the specified order
        ; Dimensions expressed on mm or degrees (real numbers)
        ; Transformations = I must be used for identity (no rotation or translation)

[PRIMITIVE2]                                ; Second Primitive of Object
        ..
        ..                  ; As many Primitives as required to define Object (at least one)
```

; PART DESCRIPTION FILE (PAF)
; This file describes a Part as an Object with a Operating Frame defined for Robot Operation

```
[GENERAL]                                   ; General Information
        Name            = Screw             ; Part Name
        Manufacturer    =                   ; Robot Manufacturer (for robot components)
        Model           =                   ; Robot Model (for robot components)
        Author          = David Puig        ; Author
        Company         = DISA - UPV        ; Company
        Date            = 9/2/99            ; Date
```

```
[OPERATION1]                                ; Location of Operation Frame 1 related to Part Frame
        Transformations = I                 ; Defined with Basic Transformations (related to Part Frame)
```

```
[OPERATION2]                                ; Location of Operation Frame 2 related to Part Frame
        Transformations = I                 ; Defined with Basic Transformations (related to Part Frame)
```

```
        ..                                  ; As many Operation Frames as required to define Part
                                            ;       (at least one)
```

```
[PRIMITIVE1]                    ; Defined as a Primitive in Object Geometric Description File
        ..                      ; Transformations related to Part (X,Y,Z) or Primitive (U,V,W) Frames
```

```
[PRIMITIVE2]
        ..
```

```
        ..                                  ; As many Primitives as required to define Part (at least one)
```

; ENVIRONMENT GEOMETRIC DESCRIPTION FILE (ENF)
; This file describes an Environment as a set of Objects and Parts

```
[GENERAL]                                   ; General Information
        Name            = RoboticsLaboratory    ; Environment Name
        Manufacturer    =                       ; No sense for Environment
        Model           =                       ; No sense for Environment
        Author          = Carlos Correcher      ; Author
        Company         = DISA - UPV            ; Company
        Date            = 9/2/99                ; Date
```

```
[OBJECT1]
        Name            = Table     ; Object Name
        Transformations = I         ; Location of Object1 related to Environment Frame
[OBJECT1_PRIMITIVE1]                ; Defined as a Primitive in Object Geometric Description File
        ..                          ; Transformations related to Object (X,Y,Z) or
                                    ; Primitive (U,V,W) Frames
[OBJECT1_PRIMITIVE2]                ; As many Primitives as required to define Object1 (at least 1)
        ..
```

```
[OBJECT2]
        ..
```

```
        ..                          ; As many Objects as required to define Environment (even none)
```

```
[PART1]
        Name            = Screw     ; Part Name
        Transformations = I         ; Location of Part1 related to Environment Frame
[PART1_OPERATION1]
        Transformations = I         ; Location of Operation Frame 1 related to Object Frame
[PART1_OPERATION2]                  ; As many Operation Frames as required to define part (at least 1)
        ..
[PART1_PRIMITIVE1]                  ; Defined as a Primitive in Object Geometric Description File
        ..                          ; Transformations related to Part (X,Y,Z) or Primitive (U,V,W) Frames
[PART1_PRIMITIVE2]                  ; As many Primitives as required to define Part1 (at least 1)
        ..
```

```
[PART2]
        ..
```

```
        ..                          ; As many Parts as required to define Environment (even none)
```